Science Applications International Corporation
An Employee-Owned Company

National Data Buoy Center

Tropical Atmosphere Ocean Project

(TAO Array Volume 2)

DMAC/QC Delayed Mode Operating Procedures Manual

September 2005

Science Applications International Corporation
An Employee-Owned Company

Title Page

Document Name: ***Tropical Atmosphere Ocean Project***
DMAC/QC Delayed Mode Operating Procedures

Publication Date: ***September 2005***

Contract Number: XX-XXXX-XXXXXXXXX

Project Number: Task: 05-TOS-03

Prepared by: Richard Gagne/Janice Boyd

Approval: ___________________________

Name and Organization

Concurrence: __________________________

Name and Organization

Science Applications International Corporation
An Employee-Owned Company

Change Control Page

The following information is being used to control and track modifications made to this document.

Revision #	Date	Author	Authorization	Summary of Change
1.0D01	8-23-05	Richard Gagne, Mark Livingston, James Singer, Paul Blankinship, Janice Boyd	TAO Project	Initial Draft
1.0D02	9-21-05	Richard Gagne, Janice Boyd	TAO Project	Draft after Technical Review

Science Applications International Corporation
An Employee-Owned Company

Table of Contents

Science Applications International Corporation
An Employee-Owned Company

Preface

Document Version Control: It is the developer's responsibility to ensure that the latest version of the operational procedure is contained in this manual. Questions should be directed to the developer of the manual or section lead.

This manual was generated by the ***Tropical Atmosphere Ocean (TAO)*** project team. TAO manual will be developed for the ***Data Quality Assurance (DQA)*** section of the ***National Data Buoy Center (NDBC)*** of the ***National Oceanic and Atmospheric Administration (NOAA)***.

Approval: ***A complete stage review /exit will constitute approval of this document.***

Developer: The primary contact for questions regarding this manual is:

Janice Boyd, SAIC, Oceanographic Data Analyst

TAO Team

Phone: ***(228) 688-3049***

Email: ***janice.boyd@noaa.gov***

Privacy Information

This manual may contain information of a sensitive nature. This information should not be given to persons other than those who are involved with the ***TAO*** data or who will be involved with performing tasks on ***TAO*** buoys.

1. Introduction

1.1 Scope

The TAO operating procedures manual will encompass the following procedural areas:

- Manual delayed-mode (RAM) checks
- Documentation
- Non-QC/analysis procedures

1.2 Objective

The TAO array has moved from research to operational status and there has been a transition from TAO being operated by Pacific Marine Environment Laboratory (PMEL) to the NDBC. The purpose of this operating procedures manual is to direct the user on the procedures of the real-time, delayed mode, and Acoustic Doppler Current Profiler (ADCP) data quality assurance checks and the analysis of the data for delivery to the customer.

1.3 Background

Development of the Tropical Atmosphere Ocean (TAO) array was motivated by the 1982-1983 El Nino event, the strongest of the century up to that time, which was neither predicted nor detected until nearly at its peak. The event highlighted the need for real-time data from the tropical Pacific for both monitoring, prediction, and improved understanding of El Nino. As a result, with support from NOAA's Equatorial Pacific Ocean Climate Studies (EPOCS) program, PMEL began development of the ATLAS (Autonomous Temperature Line Acquisition System) mooring. This low-cost deep ocean mooring was designed to measure surface meteorological and subsurface oceanic parameters, and to transmit all data to shore in real-time via satellite relay. The mooring also designed to last one year in the water before needing to be recovered for maintenance. Under the direction of PMEL scientist Stan Hayes, prototype ATLAS were field tested in early 1984, and a modest scale array was deployed along 110W in late 1984. Additional ATLAS deployments were made beginning in 1985 at the start of the 10-year (1985-94) international Tropical Ocean Global Atmosphere (TOGA) program. The array, named the Tropical Atmosphere Ocean (TAO) array, grew slowly during the first half of TOGA as the proof of concept for a sustained buoy observing system was evaluated. Initial successes led to a rapid expansion of the array during the second half of TOGA with the widespread support of the climate community. The full array of nearly 70 moorings was not completed until the final month of TOGA (Dec 1994). During the 10 years in which the array was under development, over 400 buoys were deployed on 83 cruises, using 17 different ships from 6 different countries. Accomplishing this feat required a multi-national partnership of institutions in the US, Japan, France, Taiwan, and Korea.

After TOGA ended in 1994, the TAO array continued under sponsorship of the international Climate Variability and Predictability (CLIVAR) program, the Global Ocean Observing

System (GOOS), and the Global Climate Observing System (GCOS). In 1996, the NOAA Ship KA'IMIMOANA was commissioned to service the TAO array east of 165E. In 1997, the US Congress authorized long term sustained support of the TAO array as part of an operational El Nino/Southern Oscillation (ENSO) observing system. On 1 January 2000, the TAO array officially became the TAO/TRITON array, with sites west of 165E occupied by TRITON (Triangle Trans Ocean Buoy Network) buoys maintained by the Japan Agency for Marine-Earth Science and Technology (JAMSTEC).

The operationally supported measurements of the TAO/TRITON array consist of winds, sea surface temperature, relative humidity, air temperature, and subsurface temperature at 10 depths in the upper 500 m. Five moorings along the equator also measure ocean velocity. Additional moorings and/or enhancements to the basic measurement suite are often incorporated to the operational array in support of research studies to understand specific physical processes not well measured by the existing network. Other measurements may be made for satellite or numerical model validation purposes. These research efforts are usually of limited duration and/or geographical scope, and done in collaboration with other institutions in the US and abroad.

To meet the demands of both operational and research measurements in the TAO array, an engineering redesign of the ATLAS was initiated in 1994 to update it with greater measurement capabilities, improved ocean temperature sensor accuracies, and more modular construction. The Next Generation ATLAS now has the capability to measure and transmit in real-time salinity, rain rate, long and shortwave radiation, barometric pressure, and ocean velocity. These measurements are made at selected sites to meet the needs of specialized research experiments. A robust high-latitude version of the ATLAS mooring has also been designed, capable of deployments for up to one year in the more energetic oceanic regimes of the extra-tropics.

The TAO/TRITON array is currently supported by the US (NOAA), Japan (JAMSTEC), and France (IRD), **reference 9**.

1.4 Target Audience

The following is the target audience that the DQA Operating Procedures Manual will impact:

Organizational

- National Data Buoy Center (NDBC)
- National Oceanographic and Atmospheric Administration (NOAA)
- Science Applications International Corporation (SAIC)

Personnel

- Data Quality Analyst, Section Management
- Data Quality Analyst, Delayed Mode

1.5 References

PMEL

Reference Number	Author	Title
1	Brian J. Lake, Sonya M. Noor, H. Paul Freitag, and Michael J. McPhaden	Calibration Procedures and Instrument Accuracy Estimates of ATLAS Air Temperature and Relative Humidity Measurements
2	L.J. Mangum, H. Paul Freitag, and Michael J. McPhaden	TOGA TAO Array Sampling Schemes and Sensor Evaluations
3	H.P. Freitag, M. O'Haleck, G.C. Thomas, and M.J. McPhaden	Calibration Procedures and Instrumental Accuracies for ATLAS Wind Measurement
4	H.P. Freitag, M.E. McCarty, C. Nosse, R. Lukas, M.J. McPhaden, and M.F. Cronin	COARE Seacat Data: Calibrations and Quality Control Procedures
5	P.N. A'Hearn. H.P. Freitag, and M.J. McPhaden	ATLAS Module Temperature Bias Due to Solar Heating
6	Yolande. L Serra, Patrick A'Hearn, H.P. Freitag, and Michael J. McPhaden	ATLAS Self-Siphoning Rain Gauge Error Estimates
7	H.P.Freitag, M.J. McPhaden, C.Meinig, and P.Plimpton	Mooring Motion Bias of Point Doppler Current Meter Measurements
8	P.E. Plimpton, H.P. Freitag, and M.J. McPhaden	Processing of Subsurface ADCP Data in the Equatorial Pacific

1 **Websites**

Reference Number	Author	Title
9	PMEL	http://www.pmel.noaa.gov/tao/
10	NDBC	http://port01

2 **Other**

Reference Number	Author	Title
11	Fofonoff, P., and R. C. Millard Jr.	Algorithms for computation of fundamental properties of seawater

1.6 Acronyms

AGL	*Above Ground Level*
AOI	*Average Optimum Interpolation*
ASCII	*American Standard Code for Information Interchange*
AT	*Air Temperature*
ATLAS	*Autonomous Temperature Line Acquisition System*
COADS	*Comprehensive Ocean-Atmosphere Data Set*
CTD	*Conductivity, Temperature, Depth*
DMAC	*Data Management and Communication*
DQA	*Data Quality Assurance*
GTS	*Global Telecommunications System*
JAMSTEC	*Japan Agency for Marine Earth Science and Technology*
NDBC	*National Data Buoy Center*
NOAA	*National Oceanic and Atmospheric Administration*
NTSC	*NDBC Technical Service Contract*
PMEL	*Pacific Marine Environmental Laboratory*
QA	*Quality Assurance*
QC	*Quality Check*
RH	*Relative Humidity*
RMS	*Root Mean Square*
RTD	*Resistance Temperature Detector*
SST	*Sea Surface Temperature*
SWR	*Shortwave Radiation*
T	*Subsurface Temperature*
TAO	*Tropical Atmosphere Ocean*
TRITON	*Triangle Trans Ocean Buoy Network*
XBT	*Expandable Bathy Thermograph*

2. Manual QC Procedures

2.1 Procedures for Database Management

Within the daily operations there are database management procedures ties to the analysis of data to ensure a quality product is distributed to the end customers. Performing the database updates involve manual manipulation of scripts within the LINUX Fedora operating environment. The initiated scripts will perform updates to the GTS calibrated file data once the analyst has flagged the data appropriately upon completion of their manual delayed mode procedures.

2.2 Procedures for Climatology Management

Climatology in the traditional sense is not a factor in how the TAO system maintains historical measurement data. For the purpose of TAO, climatology is really historical information averages stored under the directory *tao/realtime/sitedb* and derived by web-based applications for display using plotting scripts. The ATLASCheck program and web-based applications will utilize this historic measurement information to determine standard deviation values for automated quality checking and web-derived applications. There are outside sources which derive data provided by NDBC to perform true climatologic functions. Since these agencies are outside the NDBC organization, updates to information, access, and control of the databases are not coordinated or managed by NDBC entities. The following 6 paragraphs descriptions will cover the out-sourced sites that are used to create gridded climatologic averages up to the sites most recent model year, which in most cases are years old and not very conducive to the quality check of the raw real-time and delayed-mode data that is ingested daily in the case of real-time data or five to six weeks old delayed-mode and ADCP data.

2.2.1 Comprehensive Ocean-Atmosphere Data Set (COADS): COADS climatologic information is maintained by the NOAA Climate Diagnostics Center. The current release, 2.1, is valid from 1784-2002. Prior to the TAO project, climate records were observed from ships within the area and therefore sparse. The TAO system uses a 1 degree by 1degree grid reference to determine a climatologic average. This is done using FORTRAN programs that read the NetCDF format climatologic data that is stored within the climatology database in the TAO system at NDBC. The measurements that are maintained within COADS for TAO use are winds, barometric pressure, relative humidity, and air temperature.

2.2.2 Reynolds and Smith AOI SST Climatology: Reynolds and Smith AOI SST Climatology was developed by and maintained by the National Meteorological Center. The current release is valid from 1971 to 2000. The TAO system uses a 1 degree by 1degree grid reference to determine a climatologic average. This is done using FORTRAN programs that read the NetCDF format climatologic data that is stored within the climatology database in the TAO system at NDBC. The measurements that are maintained within Reynolds and Smith AOI SST Climatology for TAO use are for SST only.

2.2.3 Kessler Temperature Climatology: The Kessler temperature climatology is a 4-D objective analysis of historical XBT and CTD temperatures, which uses best estimates of the correlation lengths in Longitude, Latitude, and Time to grid the data. This climatology was developed by William S. Kessler at PMEL. Since the TAO project was originally developed at PMEL, this climatologic database is updated daily. A daily cronjob run automatically by the TAO system updates the MySql database, shell scripts transfer and convert the data into the intranet TAO system, and FORTRAN scripts convert the information into NetCDF files for use by the web-based TAO systems. The measurements associated to the Kessler Temperature Climatology for TAO use are subsurface temperatures, dynamic height, and 20 degree isotherm depth.

2.2.4 World Ocean Atlas Climatology: The World Ocean Atlas Climatology was developed by and maintained by the National Oceanographic Data Center. The current database has been updated to 2001. This information is used to check for anomalies in the salinity measurement data received on a daily real-time basis and during delayed mode processing. The TAO system uses a 1 degree by 1degree grid reference to determine a climatologic average. This is done using FORTRAN programs that read the NetCDF format climatologic data that is stored within the climatology database in the TAO system at NDBC.

2.2.5 Atlas of Surface Marine Data: The Atlas of Surface Marine Data is used to determine anomalies within the shortwave radiation measurement. The current analysis of surface marine data covers the years from 1945 to 1989. Since shortwave radiation from the sun is fairly consistent and mainly impacted by atmospheric constituents within the near term, the information 1X1 degree averages will still be effective for current information being received into the TAO system. The site information and database are maintained and updated by the National Oceanographic Data Center.

2.2.6 Xie and Arkin Precipitation Climatology: The Xie and Arkin Precipitation Climatology are used to determine anomalies within the precipitation measurement. The current climatologic data covers the years from 1979-2001. There is an indication that the precipitation data after 1996 is taking a downward trend and may not be very reliable for daily real-time quality checks of the TAO precipitation measurement. This would also be true when performing comparisons of the TAO delayed-mode data. The site information and database are maintained by the National Center for Environmental Prediction in the production of the 1X1 degree grid ASII format.

2.3 Procedures for Data Processing

Data processing encompasses two realms of responsibility, real-time and delayed mode. Real-time includes the daily, weekly, and monthly processing of data that is ingested daily from service Argos. Delayed mode, also known as RAM data, is the processing of the high resolution data that is downloaded from the A2 modules and A2 tube directly from off the ATLAS mooring itself. This volume will cover the delayed mode processing.

Delayed Mode (RAM): Raw data recovered from the internal memory are first processed using computer programs that apply pre-deployment calibrations and generate time series in engineering units. These programs also search for missing data and perform gross error checks for data that fall outside physically realistic ranges. A computer log of potential data problems is automatically generated as a result of these procedures.

Next, time series plots, spectral plots, and histograms are generated for all data. Plots of differences between adjacent subsurface temperature measurements are also generated. Statistics, including the mean, median, standard deviation, variance, minimum and maximum are calculated for each time series.

Individual time series and statistical summaries are examined by trained analysts. Data that have passed gross error checks but which are unusual relative to neighboring data in the time series, and/or which are statistical outliers, are examined on a case-by-case basis. Mooring deployment and recovery logs are searched for corroborating information such as problems with battery failures, vandalism, damaged sensors, or incorrect clocks. Consistency with other variables is also checked. Data points that are ultimately judged to be erroneous are then flagged.

Examine all the plots with to look for problems. The best way to find problems is by having looked at many, many of these plots before to know what they "should" look like, and what sort of problems are common. The next paragraphs will discuss the general quality control that occurs with all measurements processed during the RAM upload.

The first step in quality control is to look at the log files (pm###X.log and mod###X.log) for problems during the processing.

Tube problems: While checking the pm###X.log from processA2Tube, note how many tube resets occurred (indicated by how many buffers are listed as being in fast sampling mode). Resets that cannot be accounted for by deployment or repair visits should be checked out; the cause of spontaneous tube resets is unknown but is a symptom of a tube problem, occurring after the tube has been unresponsive for a period of time. Unexplained gaps in data are usually associated with such resets. Spontaneous tube resets can be verified and the dates of the resets can be found by looking at the 13th column of hex real-time data, under the TAO internal home page, Data - Data monitoring files - check latest transmissions. Simply enter the ptt number (located at the top of the cal and doc files) and a large enough number of 'days previous' to span the deployment you want to check, then select the 'wind/humidity/conductivity' data type. The 13th column of the wt###X_r.dat file produced by RAM processing (to match the formatting of real-time data) should also contain any spontaneous tube resets. A '0001' will appear when the first reset occurs, then '0002', etc. Manual (human-intended) resets will not be listed, so unless there are problems, this column should remain at '0000'.

Second, look at the plots. Most of the statistics on the plots generated by plottube and plotmod are fairly self explanatory. The most useful are the bottom 5 "minimum" and "maximum" point out values which are obviously out of range (air temperature of 1 degree, for instance). This is particularly important for the plots with a fixed Y-axis (all of the tube data). The module data plots (except for salinity and density) are auto-scaling, so the minimum and maximum values are always plotted. You just need to look at the Y-axis to examine the data range. "n" is the number of samples used for calculation of the statistics. That is, all samples that aren't flagged.

The number of samples flagged is listed also. To calculate the percentage of good data divide "n" by "n"+"flagged". *If* the data record covers the whole deployment period (checked by the dates in the upper corners of the plot box), this is the percentage of good data. If the record ends early (much more often true of tube data), then you need to multiply the calculated percentage by the percentage of the deployment period you have data for. The "outliers" number (which is only produced for the fixed-axis tube data) tells you how many data points were off the plot scale, and likely bad.

Track down all the outliers and flagged values. Outliers *can* be legitimate data (particularly for humidity), but often are not. Find out when and, if possible, why these points are what they are. This is sometimes most easily done by looking at the actual files (e.g. loading the .RAM file into EMACS (LINUX editor) and searching for "1e35" or searching for the value listed under "min" or "max"). It is sometimes more easily done by plotting the data in MATLAB. Check the plotting page for more information on how to do this. Also you can use MATLAB to find flagged values (with isnan(y)) or to find samples with a range of values (e.g. newatlas(x(find(y<100 & y>0))), which can be useful for locating bad points that need to be hand flagged (see flag_data). Two programs that can be useful if the data seem to be particularly noisy, with random values scattered throughout, are get_lows_1 (for flagging individual bad points) and get_lows_3 (for flagging larger spikes and dips). Both are based on a user-input noise-finding threshold, and produce a (usually long) list of flags that can be pasted into the flag###X.txt file (explained in more detail in the flag_data section). The cause of the noise should be looked into.

Also be sure to check if you have an accurate representation of the data the tube stored. Look at the raw dump file and see if the there were problems in the data transfer at the time in question, or if the data in the dump are formatted correctly but have bad values (e.g. FFFF). A short continuous chunk of missing data in all tube sensors is usually from a tube reset. Check the pm###X.log file to find out when resets and missed samples were. A frequent problem with the XMODEM binary dumps is that people FTP them without having "binary" set (rather than ASCII) on their FTP program. If transferring from or to Windows, this causes translation problems where a new line byte is translated into a carriage return *and* a line feed byte, or vice-versa. This introduces extra bytes into the binary dump, or removes bytes from a binary dump, either one of which can cause lots of problems.

Frequently surface sensors are unplugged from the tube before the "official" recovery time (the rightmost time in the first line of the pm###X.cal file), and there may be a few flagged (or otherwise bad) readings at the very end of the file and sometimes at the very beginning, too. Be sure to check on either side of, but particularly previous to, flagged periods. Often a reset or otherwise garbled buffer will have some bad data that needs to be hand flagged (see flag_data), but some will be already flagged by the processing software. *Always* find out exactly where and why any samples have been flagged. Generally, one shouldn't have *any* flagged samples.

If you determine that the deployment or recovery time is incorrect (quite possible), have the Cal file changed for re-processing.

Sensors can go bad in a variety of ways. Sometime failure is abrupt, sometimes gradual. Abrupt failures are usually either "self-flagging" (i.e. raw counts go to FFFF and are automatically converted to 1e35), or easily located, and all data points afterwards can be flagged. Gradual failure or drift is much more difficult to locate. Plot the data for the period in question at a variety of time scales to try to locate an appropriate time to start flagging. A problem obvious when a whole year is plotted may not be so when you only look at one week. Sometimes picking a time to begin flagging is rather arbitrary.

Sensors can be flagged, i.e. thrown out completely, if they are obviously bad and would be useless (or worse) to data users. If the data are questionable or unusual, but nothing is definitely wrong with it, you can lower the quality of the data, so users know that there may be a problem. For instance, if the humidity sensor seems to be working throughout the deployment, but there is one week of low (say, 50%) humidity's that does not show up at neighboring buoys, you might want to lower the quality. Flags and quality codes can be altered in the ~.flg files by running flag_data, which uses flags you've entered into the flag~.txt file.

Tube software prior to version 2.21, deployed before about March, 2000, had an incompatibility with the module software. The tube communicated inductively with the modules, setting their clocks each night just before midnight. The first night a tube could communicate it would set the module times, but *not* the day. So if the module (or tube) clocks had drifted more than 30 seconds or so, they could actually have already change to the next day, and suddenly the tube ordered them to change the time to 23:59:30. This could cause a variety of problems, depending on the module software. Sometimes modules just stopped logging for 24 hours, but kept their original (now wrong) dates. Sometimes they would skip a day of data, but change their internal date, just leaving a 144 sample gap in the buffers. These can be very confusing and difficult to track down. Look at plots at the beginning and end of the deployment to determine which modules had problems, next, start looking through the data for where the problem occurred. It is usually at the time of a tube swap, when a new tube was attached to the mooring with a clock that was set 1 minute differently from the previous tube. There's a matlab script, `correlate_modules.m`, that plots the correlations between the temperature time series for adjacent (in depth) modules. This can be very useful for finding clock problems, if suddenly 2 modules that were well correlated become de-correlated at some time; that's probably when the clock problem occurred.

A sometimes useful technique for comparing sensors to nearby buoys or climatologic means is to look at the web pages, http://www.pmel.noaa.gov/tao/jsdisplay/ for the Pacific (click on "time-series plots"), and http://www.pmel.noaa.gov/pirata/display.html for Atlantic (PIRATA) buoys.

A2 Module and Tube Processing: This outlines the major processing procedures for the meteorological ("tube") and oceanographic ("module") data downloaded from the internal memory of the ATLAS 2 moorings ("RAM delayed mode data"). Figure 1 summarizes the typical processing steps. The analyst may have to do additional processing and data editing, depending upon the problems encountered in individual data sets.

Summary of processing steps:

1. Obtain calibration data and mooring documentation
2. Process the subsurface ("module") data: processA2Mod
3. Process surface and meteorological ("tube") data: processA2Tube
4. Check log files
5. Trim times using flag_data, if necessary
6. Process rain gauge data
7. Plot and examine surface and met ("tube") data
8. Plot and examine subsurface oceanographic ("module") data
9. Complete quality control and data flagging
10. Complete processing of salinity data
11. Compute daily averages
12. Compare ram delayed mode data with real time data
13. Complete the archiving steps

Figure 1, Summary of Processing Steps

These processing steps require scripts and programs ordinarily kept on tao04 and run on that computer. The software must be in directories and subdirectories /opt/tao/delayed, /opt/tao/perl5lib, and /opt/tao/util. The data need to be in subdirectories of /taodata/recovered/ . The .bash_profile file of the tao04 home directory used for processing the data must have at the end the line /opt/tao/util/ bin/tao_env

Typical Processing Steps:

1. Obtain calibration data and mooring documentation

A calibration file, or pm###X.cal, contains all the calibration information for a mooring's sensors, as well as information about the deployment and recovery times, location etc. Documentation on .cal files is given in Appendix "Calibration File Description." *Cal files are obtained from ******.* A separate cal file should exist for each version (a, b, c, etc) of the mooring, although the post-cruise calibration file for the "a" deployment may be the pre-cruise calibration file for the "b" deployment, and so on. The programs listed here expect each cal file to be placed in the appropriate tao04 directory for that deployment (/taodata/recovered/[site]/[pm###]). At least a pre-cruise calibration file must be available. For best correction of salinity data, a post-cruise calibration file containing the conductivity cell calibration coefficients should also be available.

A mooring documentation file, or pm###.doc file, is a text file containing notes made about a given mooring during the time(s) it was deployed. It documents, for instance, if a certain sensor went bad and at what time. It also specifies when sensors were replaced during ship visits and other things of this sort. If a, b, etc, versions of the mooring occurred, information on all configurations of the mooring should be included in the doc file. *Doc files are obtained from* **********. As with cal files they should be placed the appropriate tao04 directory for that deployment (/taodata/recovered/ [site]/ [pm###]).

These two documents provide information on what data should be present and anything unusual that has been observed so far that may have implications for the processing steps below. The `.cal` file contains all the serial numbers of the equipment that was deployed. *It can be used to set up the mooring summary file pm###x.sum*. The cal file actually used in processing (since it might change at a later date) is written to the processing log files, but a printed copy should also be kept in the deployment processing documentation

Also examine the deployment and recovery logs for the deployment. NDBC should be provided the cruise logs in Filemaker format. These logs appear to come in 4 parts: cruise info, deployment information, recovered information, and repair information (for example, gp105kaCruiseInfo, gp105kaDep, gp105kaRec, and gp105kaRep). *{We still need to determine: will PMEL send these automatically and how and where will we store them.}* Examination of the recovered logs gives additional ideas of the problems that may appear during the processing steps below.

Make hardcopy printouts of the .cal, .doc. and deployment/recovery/repair logs to keep in the deployment processing documentation.

2. Process subsurface ("module") data: processA2Mod

C`d` to the working directory to process the data, taodata/recovered/[site]/[pm###]/working/. Type "`processA2Mod`".

The program requests the following inputs:

- Enter values for site and buoy(mooring) number (separated by /)(e.g. 095w/029) :
- Enter value for deployment letter (a):
- Enter directory path for calibration file (*default is /taodata/recovered/[site]/[pm###]/)*:
- Enter name for calibration file (pm###x.cal):
- Enter directory path for temporary files (*default is /taodata/recovered/[site]/[pm###]/working)*:):
- Enter directory path for module ram (raw data) files (*default is /taodata/recovered/[site]/[pm###]/rawdata/*):

The values in parentheses are the defaults, which are used if "Return" is entered..

As the program executes it sends messages to the screen and to the log file.

The outputs are 5 types of ASCII files: `cond###X.ram`, `pres###X.ram`, `sal###X.ram`, and `temp###X.ram and mod###X.log`, corresponding to conductivity, pressure, salinity, temperature, and the log file. All trapped errors should be noted somewhere in `mod###X.log`. All oceanographic modules have temperature, so the temp###X.ram file has multiple temperature values corresponding to a variety of depths for each date-time, as shown in the example below:

```
  770  395 2003263000000 2003263205000 2004143170600
 DEPLOYED AT  0 00.1N, 139 53.5W ON GP5-03-KA
 TUBE   639, AIR 56603, WIND 19798, RAIN   682, EPPLEY 31615
          DATE    12374    12869    13144    13249    13145    13250    13146    13251    13141
13252
                      1        5       10       20       25       40       45       60       80
100
  2003263225000   26.002  26.007  25.913  25.782  25.766  25.708  25.657  24.943  23.645
21.271
  2003263230000   26.000  25.991  25.920  25.781  25.764  25.704  25.664  25.017  23.725
21.638
  2003263231000   25.978  25.984  25.944  25.782  25.766  25.703  25.654  25.008  23.817
22.083
  2003263232000   25.996  25.965  25.900  25.809  25.770  25.701  25.636  24.900  23.657
21.235
```

Typically pressure sensors will only be at two depths, giving a `pres###X.ram` file that appears

```
  770  395 2003263000000 2003263205000 2004143170600
 DEPLOYED AT  0 00.1N, 139 53.5W ON GP5-03-KA
 TUBE   639, AIR 56603, WIND 19798, RAIN   682, EPPLEY 31615
          DATE    03017    03022 QQ SS
                    300      500
  2003263225000   298.19  497.46
  2003263230000   297.72  496.95
  2003263231000   297.72  497.28
```

Conductivity (and thus salinity) also occurs at only a limited number of depths.

Generally, processing of the oceanographic module data should be done first using processA2Mod. After launching, processA2Mod automatically finds and processes all of the module files in the rawdata directory (/taodata/recovered/[site]/ [pm###]/ rawdata/), as long as their filename extensions are of the form `.txt`, `.dat`, or `.bin` (uppercase or lowercase) and they contain the module serial number. If there is more than one raw data file ("dump") for a module, make sure the one to be processed has the correct extension and the other has an altered extension, such as '.baddates'. It is usually preferable to use binary files for processing (unless only a text dump is available), but occasionally a bad binary file will cause the program to shut down. In this case, convert the file using `module_bin2hex` (which will output a text file with extension `.binhex`), and examine it to find the source of the problem. It is possible (but rare) for the text conversion of a bad binary file to process correctly, since the text headers are read via matching, while the binary headers are read via strict set-length fields. (So, for example, if there is garble before the 'CAFE' starting each regular header, the garble will be ignored in the text reading, but will hang up the binary reading with a "couldn't find mode in dump configuration" error). Regardless, it can be useful to see the dump in text format, and sometimes bad buffers are apparent and can be removed by hand to aid processing.

3. Process surface and meteorological ("tube") data: processA2Tube

If necessary, cd to the working directory to process the data, located in taodata/recovered/[site]/[pm###]/working/. Type "`processA2Tube`".

The program requests the following inputs:

- Enter values for site and buoy(mooring) number (separated by /)(e.g. 095w/029) :
- Enter value for deployment letter (a):
- Enter directory path for calibration file (*default is /taodata/recovered/[site]/[pm###]/)*:
- Enter name for calibration file (pm###x.cal):
- Enter value for directory path for source data file (raw data file) (*default is /taodata/recovered/[site]/[pm###]/rawdata/*):
- Enter value for source data file (xxxx.bin where xxx or xxxx is tube serial number from cal file (e.g., 639.bin):
- Enter value for directory for output files (*default is taodata/recovered/[site]/[pm###]/working/)*:):
- Enter value for y or n for spawning the sub-processes (*default is n*): ****see below***
- Enter value for y or n for emailing (*default is n*):
- Enter value for y or n for background processing (*default is n*):

The values in parentheses are the defaults, which are used if "Return" is entered..

As the program executes, it sends messages to the screen and to the log file pm###X.log.

The outputs are ASCII files met###X.ram (meteorological data), rain###X.ram (rain gauge data), rad###X.ram (radiation data), and pm###X.log, along with *t###X_r.dat files, which are "tube data dumps" and may usually be ignored unless there were serious problems with the data acquisition. All trapped errors should be be written out somewhere in `pm###X.log`.

`*** Additional processing is done if processA2Tube is run in "spawned" mode, which basically incorporates steps 3, 4, and 6 into one processing run. The bad data before and after the deployment and recovery times are removed from all *.ram files (creating *.flg files) as described in step 5 below; and the rain gauge data is processed by rain_desiphon, rain_smooth, and rain_rate (step 6) to give a pm###X.rat file.`

4. Check log files

As they are created, check the program log files such as `pm###x.log`, `mod###x.log` and `rain###x.log` to see if any error messages or other anomalies are noted. Take appropriate action to resolve any problems.

5. Trim times using flag_data, if necessary

Run flag_data to trim files to the deployment times and remove some diagnostic columns of data. This is run automatically if processA2Tube is run in spawned mode in step 3.

6. Process rain gauge data

If not done in step 4, the rain gauge data must be converted to rain rate data using scripts rain_desiphon, rain_smooth, and rain_rate. This must be done after invalid data before and after deployment has been removed to create the rain###X.flg file. The result of the step is the "rain rate" file rain###X.rat. For further information refer to the Appendix "Rain Rate Processing."

- rain_desiphon

Cd to the /working directory (make sure `rain###X.flg is there)`, type `rain_desiphon and answer prompts to produce the "accumulation"` file `rain###X.acm`. This program removes siphon events from the rain volume time series. It produces a file `rain###x.acm` which is a generally increasing time series of 1 minute accumulation measurements.

Siphon events are found by running a window over the data and looking for volume drops of 100 ml or more between two adjacent good samples separated by less than 10 minutes. If such a drop occurs, and there is an overall drop of 350 ml over 10 samples, the sample before the 100ml drop and the 2 afterwards are flagged. The maximum drop between the higher of the 2 points before the drop and the lowest of 3 points after the drop are added back into the running accumulation. Warnings are generated in the log file if there is a large increase or decrease in volume that does not appear to be a siphon. This algorithm *can* make mistakes, so be wary of it.

- rain_smooth

Cd to the /working directory (make sure `rain###X.acm is there)`, type `rain_smooth and answer prompts to produce the "filtered"` file `rain###X.flt`. This script filters the accumulation file (`.acm`)(units are ml of volume; negative implies evaporation) with a 16 minute hanning window, applied every 10 minutes. This makes individual data points non-random. It should be run on all rain data. The program creates a 10 min resolution filtered file with data points at :05, :15, :25 ... minutes after the hour, having the extension `.flt`.

- rain_rate

Cd to the /working directory, type `rain_rate and answer prompts to produce the rain rate` file `rain###X.flt`. This program differences the accumulation file (`.acm` or `.flt`), producing an output file of rain or evaporation rates in mm/hr. It works equally well for 1 minute or 10 minute (smoothed) data files. The program creates an output file (`.rat`) with time stamps midway between the original timestamps. In practice, it is only run on the smoothed `.flt` files with 10 min sampling. Examine for times where rate < -3, which probably contain erroneous data, and determine what action to take.

7. Plot and examine meteorological and surface ("tube") data

Open Matlab on tao04. Type "plottube", select the working directory containing the data files (typically /taodata/recovered/[site/[pm###]/working/) and reply to the prompts. One or more of the possible plot types can be produced, including: wind, air temperature, relative humidity, sort wave radiation, long wave radiation, barometric pressure, rain accumulation, rain rate, Examine (and probably print and save) the plots for any anomalies. Note these for probable correction in step 9 where the quality control and data flagging is completed. Make appropriate hardcopy plots to keep in the deployment processing documentation.

8. Plot and examine subsurface oceanographic ("module") data

Open Matlab. Type plotmod, select the working directory containing the data files (typically /taodata/recovered/[site/[pm###]/working/) and reply to the prompts. Examine (and probably print and save) the plots for any anomalies. Note these for probable correction in step 9 where the quality control and data flagging is completed. Make appropriate hardcopy plots to keep in the deployment processing documentation.

9. Complete quality control and data flagging

The script to do this is flag_data. Depending upon the commands given in the flag.txt file, flag_data may delete data lines, change data lines, or "flag" data lines. Flag_data creates new files with the same name as the input file but with extension .flg. It takes as input only .ram files. An example of a .flg file is given below:

Line Number	Example .flg file for meteorological data
1	`20974  477 2004272000000 2004308201100 2005114134500`
2	`DEPLOYED AT   5 04.2S, 95 03.9W ON GP6-04-RB`
3	`TUBE   636, AIR 40322, WIND 28519, RAIN   883, EPPLEY`
4	`DATE   U   V   SPEED   DIR   AIRT   HUM QQQQ SSSS`
5	`-4   -4   -4   -4   -3   -3`
6	`2004309040000   -5.02   2.43   5.58   295.8   21.83   82.14 2222 5555`
7	`2004309041000   -5.00   2.62   5.64   297.7   21.84   81.27 2222 5555`
8 and similar lines to file end	`2004309042000   -4.83   2.40   5.39   296.4   21.87   81.82 2222 5555`

	. . . .
	Contents of line
1	**PTT# ^ Deployment# ^ Date-Time Group at Deployment ^ Date-Time Group at Recovery**
2	**Deployment location and cruise number**
3	**Equipment serial numbers**
4	**Headers for the columns in the rest of the file. Q is quality code; S is data mode**
5	**(TBD)**
6 - end	**Data lines for the data-time group in the first field. Subsequent line fields are values for the parameters listed in line 4. Q fields contain data quality codes for all preceding data parameters (see Fig. 2). S fields contain data mode codes for all preceding data parameters (see Fig. 2).**

The flag_data program does several things:

a.) It always trims the beginning and end of all the data (`.ram`) files, using the dates in the top line of the header. These dates are the deployment and recovery times. The flagged data file should begin at the deployment time + 2 hours, and end at the recovery time. If here is no file `flag###X.txt` found by flag_data, only actions a.) through c.) are performed.

b.) It removes certain columns of diagnostic data from certain source files, as listed below:

```
Meteorological files, MET###X.ram:  CO,VA,SP,FLAG, if present
LW radiation files, RAD###X.ram:    N, if present
Rain files, RAIN###x.ram:           N, if present
LW radiation files, LWR###X.ram:    NN, NC, ND, if present
```

c.) It adds columns of quality control and mode information, described in Fig. 1. For instance, an output met file has the format `DATE U V SPD DIR AT RH QQQQ SSSS`, where the 4 `Q`s would correspond to the qualities of SPD, DIR, AT and RH. For RAM-derived data, the source code is always 0, 5, 6 or 7, usually 5. So SSSS would normally be `5555`.. A subsurface module temperature file has something like `DATE 25.0 20.0 15.0 10.0 5.0 22222 55555`, with a variable number of columns depending on how many sensors were present. Except quality controlled salinity files there is one `Q` and one `S` value for each data column

d.) If a file `flag###X.txt` exists in an appropriate place (see below), certain data types can be flagged for specific date-time groups.

An example of a flag file is:

```
MET
1998333145100 - end             F       5
```

```
        start - 1998326113000           F       1..4
        begin - end                     +3.8    4
        RAIN
        1999027201434 - 1999027201434   "22.43" 1
        RAD
        1998350061000 - 1998350065400   I       1 2
        BARO
        begin - end                     Q4
        TEMP
        begin - end                     DAY     2..4
        1998351000000 - 1998351010000   Q4      5..end
        SAL
        start - end                     F       1
```

As a result of these instructions, flag_data will:

For the MET data file

- flags column 5 (air temperature) of the met file after Nov 29, (day 333) 1998 at 14:51:00 as bad, inserting 1e33 for the data values and 5 for the data quality code .
- flags all wind data (columns 1 through 4 of the met file) from the beginning until Nov 22 (day 326) at 11:30:00 as bad (inserting 1e33 and quality code 5).
- adds 3.8 to the wind direction and recomputes U and V with this rotated direction (automatically).

For the RAIN data file

- It replaces 1 sample (at 20:14:34) of rain data with the string 22.43.

For the sw radiation file RAD

- It does a linear interpolation of the radiation data (both columns - radiation and standard deviation) over a 44 minute period.

For the barometric pressure BARO file

- It changes the quality code from the default, 2, to 4 for all columns of the barometric pressure file (there is only one column).

For the subsurface temperature TEMP file

- It flags the daytime module temperatures for the 3 modules below the buoy.

For the salinity SAL file

- It lowers the quality index of the temperature sensor in columns 5 and deeper. It then flags all samples for the first column of salinity (and automatically flags the conductivity and density files identically).

From this example, note that the flag###x.txt file is divided into sections for each type of data: MET, RAIN, RAD, LWR, BARO, TEMP, PRES, SAL. Each section modifies one file type except for SAL, which flags all three of COND, SAL and DENS files. Each line flags for a given date range with the method specified. Method possibilities are:

- F : Flag the data point with the bad value flag, 1e33 (*and set quality code to 0?)*
- I : interpolates linearly with time from the data point before the flagged period to the sample after the flagged period. Uses first (or last) good value if you flag to the beginning or end, respectively, of the file.
- "xxxx" : Replaces the value in the file with xxxx. Sometimes useful for fixing rain problems with siphons. This is a direct string substitution.
- +4.8, -3, /4.7, *.99 : does the given math operation to the column. If done to wind speed or direction (met file, columns 3 or 4), U and V are recalculated as well. Flag_data can accept simple first or second order equations (mx+b or ax^2+bx+c) as flags, with the "x" variables represented by data columns or a counter. This is useful for offsetting a column of data by subtracting some sort of (1st or 2nd order) fit to it. This met flag, for example: "-0.669468085136+(-20698.427382592596*(col0/1e12)+41442.069732808952) 5" uses the date column (col0) to subtract a linear fit from the AT column, to eliminate a drift in the AT data. (Note that if you make the huge dates manageable with col0/1e12, you need to maintain the decimal precision of all of the fit coefficients to 12 places in order for flag_data to give you the correct results.) Another example is "-(7.63/6692)*$xn 6", which applies a linear fit to the RH column by using a counter ($xn) that spans the dates covered by this flag, with "1" for the first date. Other examples: "1000+3*(col0/1e12)**2+3*(col0/1e12)", "**$xn-100 1", "-(col0/1e12)**$xn 1", "*(-1)+500-(col3-col4) 6", "-(col1) 11". (Note that in the last example, parentheses are used to avoid having flag_data choke on a "--#" evaluation in the event that the data in column 1 are negative.)
- DAY : flags during "daytime". Finds local noon from the location in the header of the file, and flags data for 6 hours on either side. Should leave 1/2 of a day's data points. Rarely used now.
- Q#: Sets the Quality Index code to the value specified in #. The code is applied to the data corresponding to the quality index (Q) column number (NOT the data column number). In the case of met data, there are four Q columns, corresponding to wind speed, wind direction, air temperature, and relative humidity. So in the flag.txt file, for example, to change wind speed quality to "4" from the default quality of "2", enter a line of the form: start - end Q4 1. This will cause the first of the four Q columns (corresponding to the speed data in column 3) to be set to 4.

- Flags (including Q alterations) are weighted, with sorting by weights, so in the "unlikely" event that data need to be reprocessed and new flags need to be assigned, the user can prioritize old versus new flags by adding "w#" (with higher numbers having greater weights) at the end of a flagging line (e.g. start - end f 1..4 w23). The order of the flags listed in flag###X.txt does not matter because flag_data sorts flags before applying them to the data. Flags are sorted first by weight, then by starting date, so all "w0" flags will be listed from earliest to latest, followed by all "w1" flags, then "w2", etc. A "w9" flag with starting date 20030010000 will be listed before a "w10" flag with starting date 20023180000. Flags are applied to the data in the order of lowest-weighted to highest-weighted. Any flags that don't have a weight listed (such as the ones in flag###X.txt files created before May 2003), are assigned a default "w0" and applied first. Older "F" flags are removed if they coincide with newer math or interpolation flags. The assumption behind this is that the user (or program, such as process_rainspikes) did not realize a 1e35 flag existed when assigning an "I" or a math operation to it (or to within 1 point of it, in the case of interpolation). For all other flag types, whenever a newer flag coincides with an older flag (e.g. "xxxx" replacement of a 1e35 flag, or a math operation on a previously interpolated value, etc....), it is assumed the user (or program) knew about the old flag when applying the new one.

Note again that output file names are the same as the input files, but with `.flg` replacing the `.ram` extension.

If you are flagging any files at any time, be **sure** you run whichever of the following processes are relevant afterwards. This always includes `davg`. Freshly flagged rain files need `rain_desiphon`, `rain_smooth`, and `rain_rate` run as well. And then check to make sure the finished files are as you expect.

10. Complete processing of salinity data

See section *****.

11. Compute daily averages

Once the data files are cleaned up as best they can be, cd to the working directory on tao04 (typically /taodata/recovered/[site/[pm###]/working/) and create the daily average files containing the means for each day between 00:00 - 23:59:59 GMT. To create these files, type "davg" and reply to the responses.

The davg program takes the `.flg` and `.rat` files produces daily averages from the high resolution (10 min average) data in. In addition to calculating daily means (00:00 - 23:59:59) and the statistics listed below, it rearranges the columns so that in the output `.davg` files each quality and source column is immediately after the column(s) it refers to. This is to make them more similar to the daily average files produced in real-time from the Argos transmissions. If more than half the high-resolution samples were flagged as bad, the daily average will be replaced with 1e36. All files produced have the `.davg` extension. Create and save hardcopy plots in the deployment processing documentation.

12. Compare RAM delayed mode data with real-time daily averages

Use the Matlab siteplot program to plot the daily average data derived from RAM vs. those transmitted in real time. Take actions required to resolve discrepancies.

13. Complete the archiving steps

This is NDBC specific. It may include the following steps:

- *Clean up the working directories. Delete any unnecessary files, duplicates, old files, temporary files, files created by your text editor or anything of the sort. It is* MUCH *easier for you to clean these up than for someone else to try to figure out what they all are a couple of years from now.*
- *Notify Jing Zhou and Don Biven to create the netcdf files and put them in the right place*
- *Load daily averages into SiteData database with* `RAMdbload`
- *Write summary report*
- *Compress files*
- *Document completion of mooring processing (how?)*

Raw dump files (.DAT and/or .BIN files)

```
                                       |
                                       |
                        processA2Mod | processA2Tube
                                       |
                                      \ /
                     Preliminary processed data (.RAM)
                                       |
                                       |
                              flag_data |
                                       |
                                      \ /
 MATLAB plotting <---------- Trimmed and flagged data (.FLG)
     plottube                          |
     plotmod                           |
                                      / \
       / \                           /   \
        |                 --<-rain----     ---others->--
        |                /                              \
        |               |                                |
        |  rain_desiphon |                               |
        |               |                                |
        |              \ /                               |
```

```
             \------ Accumulation file (.ACM)                      |
             |                  |                                  |
             |                  |                                  |
             |      rain_smooth |                                  |
             |                  |                                  |
             |                 \ /                                 |
             |    Filtered accumulation (.FLT)                     |
             |                  |                                  |
             |                  |                                  |
             |        rain_rate |                                  |
             |                  |                                  |
             |                 \ /                                 |
             \----- Rain rate file (.RAT)                          |
                                |                                  |
                                |                                  |
                                 \                                /
                                  \                              /
                                   ----rain->--     -<-others----
                                               \   /
                                                \ /
                                                 |
                                            davg |
                                                 |
                                                \ /
                                     Daily averaged files (.DAVG)
                                                 |
                                       RAMdbload |
                                                 |
                                                \ /
                              SiteData SQL database of daily averages
```

Figure 2. Flow chart outlining the steps in processing TAO mooring data and the file name extension conventions.

Summary of Data Codes:

```
Data Quality Codes
------------------

    0 - No Sensor, or Datum Missing.

    1 - Highest Quality.  Pre/post-deployment calibrations agree to within
        sensor specifications.  In most cases, only pre-deployment
calibrations
        have been applied.

    2 - Default Quality.  Pre-deployment calibrations applied.  Default value
        for sensors presently deployed and for sensors which were either not
        recovered or not calibratable when recovered.

    3 - Adjusted Data.  Pre/post calibrations differ, or original data do not
```

```
        agree with other data sources (e.g., other in situ data or climatology),
        or original data are noisy.  Data have been adjusted in an attempt to
        reduce the error.

    4 - Lower Quality.  Pre/post calibrations differ, or data do not agree with
        other data sources (e.g., other in situ data or climatology), or data are
        noisy.  Data could not be confidently adjusted to correct for error.

    5 - Sensor or Tube Failed


Position Quality Codes
----------------------

    8 - Has moved from deployed location

    9 - Daily drift speed exceeds .1 kts (0.05 m/s)


Data Mode Codes
---------------
    0 - No Sensor, No Data
    1 - Real Time (Telemetered Mode)
    2 - Derived from Real Time
    3 - Temporally Interpolated from Real Time
    4 -
    5 - Recovered from Instrument RAM (Delayed Mode)
    6 - Derived from RAM
    7 - Temporally Interpolated from RAM
    8 -
    9 -


Data Flag Values

------------------
 1E30 - Threshold test value- anything greater than this is flagged
 1E33 - Flagged data (manual)
 1E34 - Value computes out of range (automatic)
 1E35 - No data / missing / initialize all output
 1E36 - Insufficient samples for averaging (only for averaged data)


Data Origin (Sensor ID) Codes
-----------------------------

    0 - No Sensor
    1 - Temperature (Cable)
    2 - Pressure
    3 - Wind (RMY)
    4 - Conductivity (FSI)
```

```
 5 - Self-Calibrating Temperature
 6 - Air Temperature
 7 - SST
 8 - RH
 9 - Pressure/Temp
10 - Radiation
12 - Rain
13 - (9-bit NOPP Wind)
14 - NextGen Conductivity
15 - NextGen Temperature
19 - NextGen Pressure
22 - Rain (PIC board update)
24 - NextGen Conductivity (Module software version 5.03+)
33 - Gill Windsonic anemometer
40 - SonTek Argonaut Current Meter
50 - Longwave Radiation
51 - Longwave Radiation Thermistor
60 - Barometric Pressure
69 - Means of conductivities from more than one sensor type
70 - Seacat Conductivity
71 - Microcat Conductivity
72 - Seacat Temperature
73 - Microcat Temperature
74 - Seacat Pressure
75 - Microcat Pressure
93 - Handar Wind (KEO)

99 - Unknown
```

2.3.1 Salinity Data Measurement

2.3.1.1 Description: The salinity measurements are taken using 3 different module types. The original type was a Seacat SBE 16 produced by Sea Bird Electronics. The Seacat has a resolution of 0.0001 S m^{-1}, a total range of 0-6 S m^{-1} and a measured range of 3-6 S m^{-1}, and an accuracy of ±0.02 psu. This sensor is being replaced by the Microcat SBE 37 produced by Sea Bird Electronics due to the Seacat 16 no longer being produced. The Microcat was used instead of the upgraded Seacat 16 plus because it has the same specifications as the original Seacat 16 except for the resolution is 0.00001 S m^{-1}, so no evaluation of sensor performance had to be performed. The third module is an ATLAS module utilizing the Sea Bird cells which has a resolution of 0.002 S m^{-1}, a total range of 0-6 S m^{-1} and a measured range of 3-6 S m^{-1}, and the accuracy is under evaluation, **reference 4**. The outer casings of the sensor modules are not being used. They are custom fabricated by PMEL just the electronic packages are being used.

The sampling rate of the salinity measurement referenced in table 2-1, indicates the intervals at which the measurements are taken. The data that are received by the NDBC contains the previous day's daily mean, **reference 2**.

Table 2-1, Salinity Sampling Information

Measurement	Sample rate	Sample Period	Sample Time	Data Recorded	Transmitted Data
Subsurface Temperature	1 per 10 minute	instantaneous	0000,0010… begin time	Every 10 Minutes	Daily mean

Salinity measuring instruments (i.e., conductivity and temperature measuring instruments) recovered in working condition are returned for post-deployment calibration before being reused on future deployments. After post-deployment calibrations are made, the resultant coefficients are compared to the pre-deployment coefficients. A set of output values is computed by application of the calibration equation using pre-deployment coefficients to a set of input values. Input values are chosen so that the output values range over normal environmental conditions. A second set of output values is generated by application of the calibration equation using post-deployment coefficients to the same set of input values. Sensor drift is calculated by subtracting the first set of output values from the second set of output values. The sensors are then assigned quality codes based on drift.

Some salinity time series had a few positive spikes or periods of unusual high frequency variability. These spikes were most common at 1-m or 3-m time series and were distinct from negative salinity spikes due to rainfall. These spiky salinity periods were often correlated with strong daytime heating and high frequency variability in the temperature time series such as on November 3, 14, and 18, 1992, at 0°, 156°E (Fig. 9). On 19 November, when similar heating occurred, but without the high frequency variability, salinity variability was relatively small.

Note that in the example shown the spikes are limited to less than 5 m depth. The strong diurnal heating was associated with days with low wind speed, during which a thin layer of surface water was heated in the presence of weak mixing. The high frequency variability in the temperature time series was perhaps due to small gusts of wind overturning the heated layer for short periods. A physical process generating positive salinity spikes of 0.4 psu or more exhibited here is unlikely. Using observed temperature and wind values in the bulk formula for latent heat flux, evaporative increases in salinity due to enhanced SST should have been much less than 0.1 psu. In addition, no source of such high salinity water was available immediately below the surface. It is presumed that the source of much of this salinity noise was a mismatch in the response times of the temperature and conductivity sensors. The Seacat temperature time constant is 0.5 s, while the conductivity time constant is a function of the cell flush rate. Anti-fouling plugs and low wind speed (perhaps associated with calm seas and a low surface-current speed) would both tend to decrease the flush rate. Under these circumstances, the conductivity time constant would exceed 1 s (Nordeen Larson, Sea-Bird Electronics, Inc., personal communication). These salinity spikes were corrected by PMEL researchers on a caseby-case basis. Identification and characterization of periods with erroneous spikes was aided by graphical computer software which displayed the temperature and salinity of the noisy instrument and, when available, rainfall and/or the salinity of an adjacent instrument located a few meters below the first. Methods of salinity correction included substitution of the adjacent instrument's data, linear interpolation (in time), or smoothing. Substitution was used when salinity at both levels was nearly identical during the nights 20 before and after the day with noisy salinity. When salinity at both levels differed at night, or when no adjacent time series was available, then a subjective choice of when and how to interpolate or smooth was made. When spiky data were identified, but no satisfactory correction could be made, the data were flagged as bad. Salinity data that were modified in these ways are identified by their quality index in the data archives. In some cases, surface salinity data were considered suspicious due to spikiness, but because no corroborating data (rainfall or nearby salinity) were available, no correction was attempted, **reference 4**.

2.3.1.2 Delayed Mode (RAM) Procedures: The following procedures and recommended QC activity all assume that basic delayed mode Tube and Module data processing and general delayed mode QC, paragraph 3.3, steps have been completed. This includes running the "flag data" program as many times as needed to handle data processing errors and creating the temperature plots discussed in these procedures.

- Lots of problems here, usually. Salinities often drift over the deployment period, sometimes show sharp increases or decreases and are often noisy. Common problems include:

- Diurnal changes in salinity caused by solar heating of the module (module temperature goes up, while water temperature and conductivity remain constant, causing calculated salinity to decrease).

- The conductivity cell responds more quickly to changes of water properties does the temperature sensor, so if a water mass with different conductivity and temperature (but the same or different salinity) moves in, the measured conductivity changes quickly, while the measured temperature changes more slowly, causing a spike in salinity (due to the mismatch) until the temperature and conductivity sensors equilibrate. These events are not usually flagged.

- When the batteries get low in a module, the conductivity starts getting "dropouts", i.e. low salinity values that gradually get lower and lower, and at some point no valid values are recorded. This happens several months before the module batteries get so low that the module stops logging. These events are usually flagged after the first bad data point.
- Salinity values are calculated from measured conductivity and temperature data using the method of Fofonoff and Millard (1983), **reference 9**. Surface salinity records are plotted and examined for periods of spiky data caused by response time differences between conductivity and temperature sensors. The identified spiky periods are flagged. Conductivity values from all depths are adjusted for sensor calibration drift by linearly interpolating over time between values calculated from the pre-deployment calibration coefficients and those derived from the post-deployment calibration coefficients.
- A thirteen point Hanning filter is applied to the high-resolution (ten minute interval) conductivity and temperature data. A filtered value is calculated at any point for which seven of the thirteen input points are available. The missing points are handled by dropping their weights from the calculation, rather than by adjusting the length of the filter. Salinity values are recalculated from the filtered data and sub-sampled to hourly intervals.
- The drift-corrected salinities are checked for continuity across deployments. In addition, for those deployments which had multiple depths instrumented with conductivity sensors, the records are compared to one another and checked for unusual density inversions indicating uncorrected drift of one or more instruments. If uncorrected drift is found, an attempt is made to identify the sensor at fault and adjust its data based on differences with data from adjacent depths during unstratified conditions. The procedures used to identify and adjust problematic data are similar to those described in Freitag et al (1999), **reference 4,** and used to correct Seacat salinity data.
- Delayed mode daily salinity and density values are calculated by taking the mean of the available hourly values for the day. If there are fewer than 12 hourly values available, a daily mean value is not computed.
- **Getting Started**: To begin, you will need access to the Next Generation Deployment Log for the mooring that you are processing. An example of one of these logs is shown in Appendix A (need to get). A listing of Data Directories, and a brief description of File Naming strategies are presented in subsequent Sections (3.3.1.5.4 and 3.3.1.5.5, respectively). In the hypothetical example that follows “999” refers to the deployment number and an “m” or “s” in front of “999” refers, respectively, to a Module or Seacat data source. An “a” or “b”, etc. after “999” refers to different segments of the mooring.
    - Make a new sub-directory named pm999 under /home/summer4/data/mod_sals. Set the environmental variables for the new

directory ($ms999) and for the ram directory ($rm999) on drumlin for the deployment. Create a moor.dat file in the new directory, which is an ascii file that contains some basic information about the site – nominal lat and lon, date range, etc. from the Next Generation Deployment Log for this mooring. (nxgo, nxmoor).

- Copy the pm999a.cal file from drumlin to $ms999. Check in $rm999 for a pm999a.cal file before assuming the file in /home/drumlin/data/recovered/60s60w/pm999 should be used. Access the mysql database and substitute post-cal coefficients for each conductivity pre-cal. Create pm999_notes.doc and write the pre and post cal conductivity coefficients in them, and any desired notes about the calibrations. (nxgo, get_cals)
- Copy (and bunzip2) the Module dump files and the flag files from drumlin from $rm999 to $ms999. (nxgo, bzget, bzlg)
- Run processA2Mod on the Module dump files using the updated post calibration file. *.ram files will be generated. The temp999a.ram file should be identical to the $rm999/temp999a.ram file. (nxgo, processA2Mod)
- Run flag_data. *.flg files will be generated. (nxgo, flag_data)
- Interpolate over time the pre-cal conductivity (on drumlin) and the post-cal conductivity (on summer). Recalculate salinity and density. Make notes in the pm999_notes.doc file about the size of the salinity drifts. *.dft files will be output. (nxgo, interp_cond)
- Run a 13-point Hanning filter on temperature and conductivity and sub-sample to hourly values. *.flt files will be generated. (nxgo, hanning_qc)
- Make matlab time series plots of all 10-minute temperatures on one page, all densities, and all salinities. (plotwhat)
- Make time series plots of the differences in filtered temperature, salinity, and density between different depths (Modules). (set_diff_plt, uniplt2k)
- Make continuity plots (first 3 days of the mooring and last 3 days of the previous mooring; last 3 days of the mooring and first 3 days of the following mooring). (set_end_plt, uniplt2k)
- Get ascii files of CTD data for the site from http://epic.pmel.noaa.gov/epic/ewb_selprof.htm, the EPIC web page. Plot a profile of the CTD data with the closest 5 hours of filtered mooring data over-plotted. (pl_ctd_nx3)
- After examining all the plots decide if additional linear drift corrections should and can be made based on CTD values. Check if there are any sudden drops or jumps in time series and decide if this needs to be corrected. If so,

make the corrections. *.adj files will be created. Make notes about the corrections in pm999_notes.doc. (mk_linear_adj, rego, adj_data)

- Filter the adjusted data. Remake any desired plots. (rego, hanning_qc, plotwhat, uniplt2k for difference plots and continuity plots, pl_ctd_nx3)
- Decide if information from adjacent Modules can be used to further correct data from a Module where some of the data are suspect. If a bias is within the mixed layer of adjacent Modules, the bias can be corrected. If a correction is to be made, find all points where the nearby values are within a specified density difference range, and calculate the salinity differences of the selected Module with the mean of the nearby Modules and an estimated salinity based on linear interpolation. Plot the differences. (nx_mix_out, uniplt2k)
- Fit a line to the differences and plot. (tim_fit_nx, uniplt2k)
- adj999a.txt (adj_data input) and run tim_fit_nx, and adj_data. Rego processes the data with adjustments. (tim_fit_nx, rego, adj_data)
- Note that the following QC codes (1-4) are used to designate salinity data quality and are assigned at this time as part of running adj_data.

  1: Drift <0.04, no other adjustments

  2: Only one calibration

  3: Drift >0.04, any other adjustments

  4: Set manually, the data are obviously bad and QC code is not 3

  (i.e., spikiness (often time a mismatch with temperature data)
- Iterate over steps 13 to 16 until you come up with what appears to be a best fit. A best fit is purely subjective. Document adjustments in pm999_notes.doc.
- Create the hourly (*.hry) files. If there are Seacat data, decide whether the Seacat data, the nx data, or a combination of the two will be used. If a combination will be used, you will need to create a rfm999a.txt file. Use reformat_sd to help create the rfrm999a.txt file. (reformat_sd)
- Login on drumlin. Add information from pm999_notes.doc on summer to the pm999a.sum file on drumlin. Copy the hourly files from summer to drumlin. If there isn't an actual file of combined Seacat and nx data, make a symbolic link pointing from salc999a.hry to salm999a.hry (Module data) or sals999a.hry (Seacat data). Make daily averages from the hourly files; output will be sal(m,c,s)999a.davg. (cpnotes, cphry, mklnk, davg)
- On summer, for salinity and density for each depth, overplot data from sal999a.flt (not really necessary), salc999a.hry, salc999a.davg, and ht999a.avg (real-time daily data). Check carefully for any spots where there are gaps in

the ram data, but not in the real-time data. Decide if the real-time data should be flagged for those points. (uniplt2k)

- If there aren't any real-time data that need to be flagged, then on drumlin, load the data into the mysql database and check that the loaded data look ok. (RAMdbload)
- Use the web page based RAM UPDATES FORM to advise (Dai?) that there are new data to serve, and to inform (Dan?) of any real-time data that need to be flagged. If there are real-time data to be flagged, the ram data will be loaded (by Dan?) into the mysql database in a routine to flag real-time data.

2.3.2 Ocean Currents/ADCP Data Measurements

2.3.2.1 Description: The ATLAS current measurements are taken using an Argonaut-MD doppler current meter produced by Sontek. The Argonaut-MD has a resolution of 0.1 cm s^{-1} and 0.1°C, a range of 0-600 cm s^{-1}, acoustic frequency of 1.5 Mhz, and an accuracy of $\pm$ 5 cm s^{-1} and $\pm$ 5°C. A vane with an aspect ratio of 2.5:1 was attached to the Argonaut-MD an was found to have acceptable dampening performance yet was small enough to have minimal impact on mooring deployment operations and low potential for being fouled by fishing nets or lines. The low-cost, all-plastic vanes are fixed to the existing clamps which attach the instrument on the mooring wire. The vane causes a significant reduction in instrument motion as evidenced by a reduction in the standard deviation of compass heading and tilt from the instrument. The differences between the Argonaut-MD with a vane and the ADCP current speed are small and comparable with the mean difference between the 45 m vaned Argonaut-MD and the nearby ADCP of -2.4 cm s^{-1} (sign implies Argonaut-MD > ADCP) and the RMS difference between daily mean data was 5.3 cm s^{-1}. The use of these vanes is now standard practice on all equatorial TAO moorings deployed with Argonaut-MD current meters. At present Argonaut-MDs mounted with the vane have been deployed and recovered from five moorings, and in most cases the velocity data are comparable to the ADCP currents with no significant bias toward lower values. Argonaut-MD and ADCP data compare most favorably at more shallow (25 m to 80 m) depths. Somewhat larger differences (up to 15 cm s^{-1} in the mean) have been found for some observations at 120 m. While the reason for this has not been determined at this time, a possible source may be lower signal strength deeper in the water column. In the eastern equatorial Pacific, 120 m is below the biologically productive surface mixed layer, and thus the density of acoustic targets is lower. Signal strength reported by instruments at 120 m is typically 20% to 40% lower than from those at shallower depths. Increasing the signal strength of the Argonaut-MD may provide better data at this depth, **reference 7**.

The RD Instruments narrow band Acoustic Doppler Current Profiler (ADCP) is a four-beam system that transmits a high-frequency acoustic pulse, measures the Doppler shift in the backscattered acoustic energy as a function of time, and computes the beam-direction velocity component as a function of range. Using compass measurements, the range-gated ADCP beam velocities are then converted into vertical profiles of zonal and meridional velocity. The ADCP has a resolution of 0.1cm s^{-1} and 0.006°C, a range of 0-256 cm s^{-1}, accuracy of $\pm$ 5 cm s^{-1} and $\pm$2.5°C, **reference 8**.

The sampling rate of the ATLAS current measurement is referenced in table 3-1 and 3-2, indicates the intervals at which the measurements are taken. The data that are received by the NDBC contains a daily mean transmitted to service Argos on the Sontek and no data transmissions from the ADCP, **reference 2**.

See table 2-2 for Sontek sampling information and table 2-3 for the ADCP sampling information .

Table 2-2, ATLAS Sontek Current Sampling Information

Measurement	Sample rate	Sample Period	Sample Time	Data Recorded in Memory	Transmitted Data
Current velocity	1-hz	2-3 min	2359-0001 … begin time	10 Minutes	Daily mean

Table 2-3 ADCP Sampling Information

Measurement	Sample rate	Sample Period	Sample Time	Data Recorded in Memory	Transmitted Data
Water velocity profile. Narrowband 150kHz ADCP	0.333 hz	15-min	0000-0015 … begin time	Hourly (Time stamp at beginning of average)	None

The ADCP instruments are a stand alone system located near the ATLAS moorings. There are currently four ADCP instruments maintained by NDBC. The Sontek instruments are mounted on the ATLAS moorings near the ADCP instruments except at one location. The following table 2-5 indicates the current locations of the Sontek and ADCP instruments.

Table 2-4, ADCP and Sontek Locations and Depth

Location	ADCP	Depths	Sontek	Depths
0N 110W	X	250m	X	7m, 12m, 27m, 47m, 82m, 122m
0N 140W	X	270m	X	12m, 27m, 47m, 82m, 122m
2N 140W			X	7m, 12m, 17m, 22m, 27m
0N 170W	X	300m	X	12m, 52m, 102m, 152m
0N 165E	X	280m	X	12m, 52m, 102m, 202m

2.3.2.2 Delayed Mode (RAM) Procedures: The following procedures and recommended QC activity all assume that basic delayed mode Tube and Module data processing and general delayed mode QC, paragraph 2.3, steps have been completed. This includes running the "flag data" program as many times as needed to handle data processing errors and creating the temperature plots discussed in these procedures.

- **Set environment on Linux machine**
  - Set the following environments on the Linux machine:

    setenv bcprog ~OCRD/plimpton/ndbc/prog

    setenv bcs ~OCRD/plimpton/ndbc/sontek

    setenv bcsraw ~OCRD/plimpton/ndbc/sontek/raw

    setenv bcsplots ~OCRD/plimpton/ndbc/sontek/plots

    setenv bcsproc ~OCRD/plimpton/ndbc/sontek/proc

- **Deployment and Recovery log sheets – Determine true start/stop times/dates**
  - Note the anchor drop time/date. Take a start time/date for data as being approximately two hours after anchor drop time. In this example:

    anchor drop is at 3 26 23 5 04 (hours minutes day month year) so use 5 26 23 5 04.

  - Note the release time/date. Use an end time/date as being the preceeding 10 minute interval BEFORE the release time. In this example:

    release time is at 16 53 12 9 04 so use 16 50 12 9 04.

  - The Sontek's measurement is made over a three (3) minute period centered at 10 minute intervals on the 10's (00, 10, 20, 30, 40, 50). For example the Sontek starts pinging at 8 minutes 30 seconds after the hour. So add 90 seconds to data to bring it to center on the 10's.
  - Magnetic declination for this site is taken as 9 degrees east so rotate this amount.
  - Process raw Sontek.dat files
  - Raw Sontek files have filenames like: D138001.dat, where the serial number of this instrument is ARG138.
  - First few lines of this raw file:

```
2004 05 15 17 08 30   -16.5    -7.9   -19.3   5.5   2.5   3.1  42  45  44 100   99.9  -0.8   1.6
6.7  1.4  1.0  20.89     0.0000    0.000   9.8
2.6   2.0

2004 05 15 17 18 30     4.5   -14.5    -9.7   5.2   3.1   3.3  42  45  43 100   99.1  -0.4   1.6
6.4  1.2  1.0  21.00     0.0000    0.000   9.8
3.0   2.0
```

```
2004 05 15 17 28 30   -9.7  -14.5  -18.1  4.7  2.9  2.9 42 45 43 100 101.9 -0.4  1.6
6.3 1.2 0.9 21.10   0.0000   0.000  9.8
3.2  2.0

2004 05 15 17 38 30  -16.5  -11.5  -22.2  5.5  2.5  3.0 42 45 43 100  99.7 -0.4  1.6
6.4 1.3 1.0 21.18   0.0000   0.000  9.8
1.9  2.0
```

- Use the program *con_sontek5.f* . This program will adjust the time by 90 seconds, rotate the data by 9 degrees and correspond the depth of the Sontek with its serial number by user input from the deployment and recovery logs.

```
con_sontek.in file:

'bcsraw/D138001.dat'  'bcsraw/pm438egr.005' 'TAO pm438 ARG138  1'
5. 10. 9. 90

'bcsraw/D69001.dat'   'bcsraw/pm438egr.010' 'TAO pm438 ARG069  1'
10. 10. 9. 90

'bcsraw/D101001.dat'  'bcsraw/pm438egr.025' 'TAO pm438 ARG101  1'
25. 10. 9. 90

'bcsraw/D117001.dat'  'bcsraw/pm438egr.045' 'TAO pm438 ARG117  1'
45. 10. 9. 90

'bcsraw/D129001.dat'  'bcsraw/pm438egr.080' 'TAO pm438 ARG129  1'
80. 10. 9. 90

'bcsraw/D204001.dat'  'bcsraw/pm438egr.120' 'TAO pm438 ARG204  1'
120. 10. 9. 90

$exit

$bcprog/con_sontek5 < $bcsproc/con_sontek.in

 adds 90sec to data time and have seconds in time

fnami,fnamo,heads,depth,delt,cvar,incsec
```

- Run the program:

```
$bcprog/con_sontek5 < $bcproc/con_sontek.in
```

- This produces the output files:

  $bcsraw/pm438egr.005, $bcsraw/pm438egr.010, $bcsraw/pm438egr.025, $bcsraw/pm438egr.045, $bcsraw/pm438egr.080, $bcsraw/pm438egr.120

- The first few lines of $bcsraw/pm438egr.005:

```
TAO pm438 ARG138 1  10.0   5.0 Argonaut

 1710 0 15 52004   -17.5    -5.2    18.3 253.4 20.89   0.00  -19.3   5.5   2.5
3.1  42  45  44 100  99.9  -0.8   1.6   6.7   1.4   1.0   0.0   9.8    1
```

```
1720 0 15 52004    2.2   -15.0    15.2 171.8 21.00   0.00   -9.7   5.2   3.1
3.3  42  45  43 100  99.1  -0.4   1.6   6.4   1.2   1.0   0.0   9.8     2

1730 0 15 52004   -11.8   -12.8    17.4 222.8 21.10   0.00  -18.1   4.7   2.9
2.9  42  45  43 100 101.9  -0.4   1.6   6.3   1.2   0.9   0.0   9.8     3

1740 0 15 52004   -18.1    -8.8    20.1 244.1 21.18   0.00  -22.2   5.5   2.5
3.0  42  45  43 100  99.7  -0.4   1.6   6.4   1.3   1.0   0.0   9.8     4
```

- Note the start and end times of each output file:

$bcsraw/pm438egr.005
1710 0 15 5 2004
1940 0 13 9 2004

$bcsraw/pm438egr.010
1550 0 14 5 2004
1950 0 13 9 2004

$bcsraw/pm438egr.025
1610 0 14 5 2004
2020 0 13 9 2004

$bcsraw/pm438egr.045
21 0 0 14 5 2004
2030 0 13 9 2004

$bcsraw/pm438egr.080
16 0 0 14 5 2004
20 0 0 13 9 2004

$bcsraw/pm438egr.120
0 0 0 28 4 2004
2010 0 13 9 2004

- Plot the 10 minute data:
- Use Plimpton's plotting software

$bcsplots/plsontek_5m.indat
pm438_005m_1.gif, pm438_005m_2.gif, pm438_005m_3.gif,
pm438_005m_4.gif

$bcsplots/plsontek_10m.indat
pm438_010m_1.gif, pm438_010m_2.gif, pm438_010m_3.gif,
pm438_010m_4.gif

$bcsplots/plsontek_25m.indat
pm438_025m_1.gif, pm438_025m_2.gif, pm438_025m_3.gif,
pm438_025m_4.gif

$bcsplots/plsontek_45m.indat
pm438_045m_1.gif, pm438_045m_2.gif, pm438_045m_3.gif, pm438_045m_4.gif

$bcsplots/plsontek_80m.indat
pm438_080m_1.gif, pm438_080m_2.gif, pm438_080m_3.gif, pm438_080m_4.gif

$bcsplots/plsontek_120m.indat
pm438_120m_1.gif, pm438_120m_2.gif, pm438_120m_3.gif, pm438_120m_4.gif

- First plot is a stack of zonal velocity, meridional velocity, speed, direction, temperature and vertical velocity.
- Second plot is U deviation, V deviation, W deviation, Beam 1 intensity, Beam 2 intensity, Beam 3 intensity and percent good pings.
- Third plot is magnetic direction, pitch, roll and battery
- Fourth plot is compass deviation, pitch deviation and roll deviation.

- **Rotate vertical velocity**
  - Include the vertical velocity in the total speed. Recompute u and v from this total speed but keep the direction unchanged. The new file keeps the old vertical velocity. Note: The Sontek records the pitch and roll of the instrument but does not use this in computing u, v w. This was recommended by the manufacturer to help with correcting data errors caused by instrument accelerations. Later fins were added to the instrument to help reduce rotational motins which did improve data quality. The manufacturer continues to recommend that pitch and roll not be used.

```
$bcprog/rot_vert_to_speed < $bcsproc/rot_vert_to_speed.in
```

makes in $bcsraw
pm438egr_pr.005, pm438egr_pr.010, pm438egr_pr.025, pm438egr_pr.045, pm438egr_pr.080, pm438egr_pr.120

- **Convert the data to ram format**
  - Convert the data to ram format so the rest of the processing will be along module procedures.

```
$bcprog/convert_sontek_to_ram_2 < $bcsproc/convert_sontek_to_ram.in
```

- **Make flag data with no edits for comparing with ADCP data**
  - Using standard RAM programs, run flag_data from Port01/pmel/dist/processing/delayed_processing/nxram/bin

```
/usr/local/nxram/bin/flag_data $bcsraw/curr438a.ram
$bcsraw/flag438a_noedits.txt
```

makes curr438a.flg

- copy curr438a.flg to curr438a_noedits.flg ! so there is a copy for you to see

Contents of $bcsraw/flag438a_noedits.txt file:

- **Make plots of ADCP/Sontek comparison**
  - Use Plimpton's plotting software and programs:

    $bcsplots/cmp_sontek_ADCP.indat
    $bcsplots/reg_sontek_ADCP.indat

  - makes: $bcsplots:

    cmp_005m.gif cmp_025m.gif cmp_080m.gif
    cmp_010m.gif cmp_045m.gif cmp_120m.gif
    reg_005m_spdir.gif reg_025m_spdir.gif reg_080m_spdir.gif
    reg_010m_spdir.gif reg_045m_spdir.gif reg_120m_spdir.gif
    reg_005m_uv.gif reg_025m_uv.gif reg_080m_uv.gif
    reg_010m_uv.gif reg_045m_uv.gif reg_120m_uv.gif

  - Note that the instruments do not measure direction well at low speeds. For these regressions exclude directions where the speed for either instrument was less than 5 cm/s. This sometimes results in n: for direction less than n: for u, v, and speed.

- **Edit Sontek data**
  - At 80m there are two intensity drops in beam 3. The second intensity drop coincides with an increase in the ADCP - Sontek velocity difference. Set Sontek data bad beginning 2004236200000
  - There are spikes in the 120m data from 2004246220000 to 2004246222000 for beam 1 intensity, u deviation, u, sp, vert vel, and compass direction. (pm438_120m_1.gif, pm438_120m_2.gif, pm438_120m_3.gif, pm438_120m_4.gif). Set the data bad for this time period
  - Note: On 245 2004 (1Sep04), the 120m D204 Sontek appeared to slip on the wire and rotate ~100 degrees. (pm438_120m_3.gif) This did not appear to cause error in the data. Likewise on 200 2004 (18Jul04), the 80m D129 Sontek appeared to slip ~60 degrees on the wire with no error in data. (pm438_080m_3.gif)
  - No apparent fishing pulls from pressure data or Sonja's editing of temperature. The Q4 values set for the temperature at 500m does not apply to shallower Sonteks.

- **Run flag_data and davg again**
  - /usr/local/nxram/bin/flag_data $bcsraw/curr438a.ram $bcsraw/flag438a_testedits.txt

- makes curr438a.flg
- copy curr438a.flg to curr438a_testedits.flg ! so there is a copy for you to see
- /usr/local/nxram/bin/davg $bcsraw/curr438a.flg
- makes curr438a.davg
- copy curr438a.davg to curr438a_testedits.davg ! so there is a copy for you to see
- Contents of the bcsraw/flag438a_testedits.txt file:

```
CURR
2004144053000 - end Q2 1..24  # has 6 depths
2004236200000 - end f 17..20  # inten drop for 80m, set bad 1e33
qual=52004246220000 - 2004246222000 f 21..24   # glitch
vels,inten,deviation,compdir
```

- **Remake ADCP/Sontek comparison plots for 80m and 120m**
  - Use Plimpton's plotting software and programs

    $bcsplots/cmp_sontek_ADCP.indat
    $bcsplots/reg_sontek_ADCP.indat

  - makes:

    $bcsplots:
    cmp_080m_testedits.gif   cmp_120m_testedits.gif
    reg_080m_spdir_testedits.gif reg_120m_spdir_testedits.gif

    reg_080m_uv_testedits.gif   reg_120m_uv_testedits.gif

  - Note that the apparent lack of agreement in the September 2004 meridional velocity at 120m may well be due to ocean variability and there is no reason to set the Sontek data bad.
  - Note also that the Sontek tends to underestimate the speed at 120m where scatterers are less abundant. This is typical of the 3mHz Sontek.
  - Sonteks when compared with ADCP data (25m, 45m, 80m (edited), 120m (edited)) all have mean differences less than 5cm/s and 5degrees and rms of differences less than 10 cm/s and 10 degrees, so set the quality of the data at these depths to Q1. (I am still determining the exact cutoffs for quality 1 data, but this is a good guess. Leave the 10m and 25m data that were compared with extrapolated ADCP data as Q2 since differences probably due to ADCP extrapolation rather than problems with Sonteks. The short distance needed to extrapolate for the 25m extrapolated ADCP data allowed me to be able to assign the 25m Sontek data as Q1.
  - At this point I do not generally adjust the Sonteks according to the Sontek compass calibration. However, I keep track of differences with ADCP and

Sontek calibrations to verify quality of future use of Sontek compass calibration, and may reprocess these data after I have confidence in the Sontek compass calibrations.

- **Direction Comparison: CA09 and PM438**
  - 1st column is the mean direction difference between the adcp with compass calibration applied and the original Sontek direction (see $bcplots/pltreg_all_pm438a.gif and pltreg_all_pm438b.gif), then the pre and post calibrations for the Sontek compass are listed, then the difference between the ADCP and the Sontek with the mean pre and post Sontek calibration removed. Don't consider the 5m and 10m data as ADCP is extrapolated.

	ADCP-Sontek	Sontek calib		ADCP-calibrated Sontek	
		pre	post		
5m D138	-3.92	0.2	0.0	extrap data	
10m D69	-1.14	-0.3	-1.0	extrap data	
25m D101	1.62	-0.1	0.2	1.67	
45m D117	3.47	-1.0	-2.0	1.97	high residual roll error
80m D129	.78	-0.7	-0.2	.33	
120m D204	1.24	-0.3	-1.7	.24	

  - Use of the Sontek compass calibration would appear to improve the adcp/sontek direction comparison for this deployment.

- **Do final edits and set qualities**
  - /usr/local/nxram/bin/flag_data $bcsraw/curr438a.ram $bcsraw/flag438a.txt makes curr438a.flg
  - /usr/local/nxram/bin/davg $bcsraw/curr438a.flg makes curr438a.davg
  - Contents of the $bcsraw/flag438a.txt file:

```
CURR
2004144053000 - end Q1 5..12  # 25m,45m,80m,120m compare well with
adcp
2004236200000 - end f 17..20  # inten drop for 80m, set bad 1e33 qual=5
2004246220000 - 2004246222000 f 21..24 # glitch in
vels,inten,dev,compassdir
```

- **Make summary file to be added to ram summary file**
  - $bcprog/make_ram_sum_list < $bcsproc/make_ram_sum_list.in

    makes $bcsraw/pm438a_sontek.sum

- copy into ram summary file: pm438a.sum

2.3.2.3 ADCP Delayed Mode (RAM) Procedures: The processing of delayed mode ADCP data is presented in this section. It is intended to be a step by step guide for processing one deployment from a single instrument. The steps described here use a specific deployment data set. The same methods can be used and adapted for other deployment files.

- **Set environment on Linux machine**
    - Set the following environments on the Linux machine:

      setenv bcprog ~OCRD/plimpton/ndbc/prog

      setenv bcadcp ~OCRD/plimpton/ndbc/adcp

      setenv bclogadcp ~OCRD/plimpton/ndbc/adcp/logadcp

      setenv bcraw ~OCRD/plimpton/ndbc/adcp/raw

      setenv bcplots ~OCRD/plimpton/ndbc/adcp/plots

      setenv bcproc ~OCRD/plimpton/ndbc/adcp/proc

      setenv bcsen ~OCRD/plimpton/ndbc/adcp/sensors

- **Deployment and Recovery log sheets – Determine true start/stop times/dates**
    - Note the anchor drop time/date. Take a start time/date for data as being at the top of the hour approximately two hours after anchor drop time.
    - Note the release time/date. Use an end time/date as being at the top of the hour on the preceding hour BEFORE the release time for the ADCP. For the pressure record which is a 10 minute sampling interval, use the most recent time prior to the release time.

- **Process raw ADCP binary files**
    - Transfer the raw ADCP binary files (ie: 140w.001 through 140w.022), to the directory on the pc that contains the RDI program logadcp. Also bring in the companion files 140wblk1.dat (created when extracting raw data from the instrument) and extract.log (this file contains status of extracting adcp data from eprom).
    - From command prompt in the directory with the logadcp program, type:

      c:\RDIprograms\logadcp

      enter data filename: 140w.001

      hit c <cr> key for conversion of file

      enter the ouput filename prefix ca09 <cr>

exit program. This will produce the ascii files (for this example: ca 09.001 to ca09.075).

Transfer these ASCII files to the Linux machine in the $bcraw directory

- Create the header, velocity, amplitude, % good and convert log ASCII files
- Go to the bcraw directory and run rdadcpascii.f program to convert ca09.001 to ca09.075 to ascii files: ca09.head, ca09.vel, ca09.amp, ca09.good, ca09_conv.log.
- Note: user input in BOLD

$bcprog/rdadcpadcii

Program RDIADCPASCII

Enter input file name: **CA09.000**

LLead, LVel,Lamp,LPGood,LStatus? (T/F) **t t t t f**

Stem of output files: ? **ca09**

Log file name: **ca09_conv.log**

Header Output filename: **ca09.head**

Velocity Output filename: **ca09.vel**

Amplitude Output filename: **ca09.amp**

% Good Output filename: **ca09.good**

Starting year? **03**

Starting and ending ensemble numbers? **1 9990** (Note: Just enter 1 to a large number to ensure getting all the ensembles)

Minutes to add to the time word? **0**

There are 44 bins available

Enter starting and ending bin numbers **1 44**

Input file name: CA09.000

Ensemble No. 1, Time 8 24 0 0 3

Ensemble No. 127, Time 8 29 6 0 3

Input file name: CA09.075

Ensemble No. 9244, Time 8 12 3 0 4

This ends the program. In the example above, the program reports that data were collected from 0000 24 Aug 03 to 0300 12 Sep 04.

Files created: $bcraw/ca09.amp, $bcraw/ca09.good, $bcraw/ca09.head, $bcraw/ca09vel, and $bcraw/ca09_conv.log

- Determine line numbers of the start/stop times.
- Go to $bcraw and open the ca09.head file. Locate the line number that has the start time as determined in section 3B above. Do the same for the stop time. In this example (ca09), the starting line is 649 and the ending line is 9237.
- Now enter these line numbers in the file: ext_head3.in

```
'bcraw/ca09.head' 'bcraw/ca09h.tilt' '(1x,2i2,1x,3i2,2f6.1,2(f6.2,f6.1),4f6.1,i6)'
'TAO CA9' 60. 649 9237
10 18 19 20 21 22 23 24 25 26 27  250 # $bcprog:adcp_ext_head3
head,sdhead,temp,vhi,xcur,vlo,pitch,sdpitch,roll,sdroll
```

- Create the pitch, roll and temperature files
- This program extracts data from header file output of adcp_conv and outputs in standard time series format.
- Go to $bcprog and run the program adcp_ext_head3.f
- $bcprog/adcp_ext_head3 < $bcprog/ext_head3.in
- This program uses data from the file: ca09.head.
- The program creates the ca09h.tilt file in the raw directory.
- **Here are the first few lines of the ca09h.tilt file:**

```
TAO CA9          60.  250.
 0 0 20 9 3 179.1  3.0 10.63  34.8  1.60  10.8   0.1   0.1   0.0   0.2     1
 1 0 20 9 3 194.6  1.0 10.91  35.3  1.60  10.8   0.1   0.1   0.0   0.2     2
 2 0 20 9 3 197.5  2.0 10.99  35.0  1.60  10.8   0.1   0.1   0.0   0.2     3
 3 0 20 9 3 202.6  2.0 11.00  34.8  1.60  10.8   0.1   0.1   0.0   0.2     4
 4 0 20 9 3 181.6  2.0 10.30  34.8  1.59  10.8   0.1   0.1   0.0   0.2     5
 5 0 20 9 3 166.0  2.0 10.08  35.0  1.59  10.9   0.1   0.1   0.0   0.1     6
```

- which has the following format:

```
hr m da mo yr head  sdhead temp   vhi   xcur   vlo  pitch sdpitch roll sdroll
 0 0  20  9   3  179.1   3.0     10.63  34.8  1.60  10.8   0.1   0.1      0.0   0.2  1
```

- Process temperature, pressure and salinity data
- Process the MicroCat dat first

  The raw MicroCat data is in the sensors directory: $bcsen/MC15.DAT

  This file has many corrupted records so manually extract the good data into the following files:

  MC15pep1.DAT

MC15pep2.DAT
MC15pep3.DAT
MC15pep4.DAT
MC15pep5.DAT

- Edit the $bcsen/mcread.in file to reflect these five (5) files

```
'bcsen/MC15pep1.DAT' 'bcsen/ca09egr1.300mc' '  CA09 '  300. 3. 2
 0  0  20 9 2003
20 30  11 9 2004     ! $bcprog/mcread < mcread.in
$exit

'bcsen/MC15pep2.DAT' 'bcsen/ca09egr2.300mc' '  CA09 '  300. 3. 2
 0  0  20 9 2003
20 30  11 9 2004     ! $bcprog/mcread < mcread.in
$exit

'bcsen/MC15pep3.DAT' 'bcsen/ca09egr3.300mc' '  CA09 '  300. 3. 2
 0  0  20 9 2003
20 30  11 9 2004     ! $bcprog/mcread < mcread.in
$exit
'bcsen/MC15pep4.DAT' 'bcsen/ca09egr4.300mc' '  CA09 '  300. 3. 2
 0  0  20 9 2003
20 30  11 9 2004     ! $bcprog/mcread < mcread.in
$exit

'bcsen/MC15pep5.DAT' 'bcsen/ca09egr5.300mc' '  CA09 '  300. 3. 2
 0  0  20 9 2003
20 30  11 9 2004
$exit
```

- Then run mcread.f on this .in file $bcprog/mcread < $bcsen/mcread.in
- This produces the following files in the $bcsen directory: ca09egr1.300mc, ca09egr2.300mc, ca09egr3.300mc, ca09egr4.300mc, ca09egr5.300mc
- Now run the program filgap999d5.f which will fill the time gaps with -999's in the output file: ca09egr.300mc.
- $bcprog/filgap999d5 < $bcsen/filgap999d5.in Makes $bcsen/ca09egr.300mc
- Processing the TP data:

  Use Sonya Noor's processing techniques and programs to convert $bcsen/11145.BIN to temp009c.ram and pres009c.ram

- Regression plots of temperature and pressure for MicroCat and TP comparison

  Use Plimpton's plotting programs

$bcplots/regtpmc.indat makes $bcplots/regtpmc_temp.gif and $bcplots/regtpmc_press.gif

These plots show that the MicroCat and TP sensors agreed to within 0.36m for pressure and to 0.01 degrees for temperature. In this example, since the MicroCat is a partial record, use the TP data for temperature and pressure to process the ADCP.

- **Average the 10 minute TP data to 30 minute data.**

  When averaging over the three (3) 10 minute values, require two (2) good values to make an average. Want to use 30 minute averaged data since the ADCP data is collected over a 15 minute interval from the top of the hour to 15 minutes after the hour. This program makes a *.date file in old time/date format rather than julian day time/date format.

  $bcprog/scalv4_ram2 < $bcsen/avtp_ca9.in

  This creates the following files:$bcsen/pres009chlf.ram, bcsen/temp009chlf.ram, $bcsen/pres009chlf.date, and $bcsen/temp009chlf.date

- **Make regression plot of ADCP temperature from $bcraw/ca09.tilt versus TP temperature from $bcsen/temp009chlf.ram.**

  Use Plimpton's plotting programs

  $bcplots/tempreg.indat

  This makes $$bcplots/tempreg.gif

  The plot shows that the ADCP and TP temperatures differ by 0.72 degrees.

- **Convert pressure file to depth**

  Using input parameters: latitude and -2.1m (depth of pressure sensor below the ADCP transducer from log sheets and line numbers).

  $bcprog/convert_pres_to_depth < $bcsen/convert_pres_to_depth.in

  This creates the file $bcsen/pres009chlf_meters_tdepth.date which is the depth of the ADCP transducer from the TP pressure every half hour. This depth will be compared with the depth obtained from the ADCP target strength.

- Use Plimpton's plotting programs to compute statistics, histograms and spectrum of pressure and temperature from TP data. Note that average pressure from TP is 293.59 with minimum of 289.53 and a maximum of 338.60

$bcplots/no program provided in proc.notes

This produces $bcplots/TPpres_hist.gif and $bcplots/TPtemp_hist.gif

Run the program on the five (5) good MicroCat data files for salinity

$bcplots/no program provided in proc.notes

This produces $bcplots/MCsalin_hist1.gif, $bcplots/MCsalin_hist2.gif, $bcplots/MCsalin_hist3.gif, $bcplots/MCsalin_hist4.gif, $bcplots/MCsalin_hist5.gif

Note that the average salinity is about 34.86 ppt.

- Convert echo amplitude to approximate target strength (so that surface reflection is a maximum for the deployment)
- Open $bcraw/ca09.amp file and determine the maximum echo amplitude for each of the four beams. For ca9 Plimpton used: beam1 = 130, beam2 = 121, beam3 = 126 and beam4 = 148. Enter these values in the target_strength3.f program for Eref() and compile.
- $bcprog/target_strength3 < $bcproc/target_strength_ca9.in
- Converts $bcraw/ca09.amp to $bcraw/ca09.tar_str
- Estimate the bin number that would intersect the surface for the min and max depths. (The first bin is 1 bin + blank =~ 13m and pressure sensor is 2m below transducer) (1 bin = 9m, blank = 4.34m, pressure sensor 2m below transducer =~15m)
- From step above calculating statistics on the TP pressure ($bcplots/TPpres_hist.gif) the mean depth is 293.59m, minimum depth is 289.53m and maximum depth is 338.60m.
- Make estimate:
- 339-15=324 bin 36 with 9m bins
- 289-15=274 bin 30 with 9m bins
- Set the bin range for polfit_to_tgstrnew.in as 27 to 40 (just outside the min/max bin range of estimate) and the depth range as 257 to 355. NOTE: This depth range does not correspond to bins 27 to 40.
- Do polynomial fit to determine the precise depth of maximum target strength.
- This program uses the nominal depth of bin of maximum target strength to compute actual depth of maximum target strength
- $bcprog/polfit_to_tgstrnew < $bcproc/polfit_to_tgstrnew.in

- Plot of transducer depth from target strength versus transducer depth from TP pressure sensor
- Use Plimpton's plotting software and command files.
- $bcplots/regtptg.indat
- This creates $bcplots/regtptg.gif
- Edit out points where surface reflection is too small, use 25 db
- $bcprog/ed_tgstr < $bcproc/edit_tgr_strnew.in
- This converts $bcraw/polfit_to_tgstrnew2.dat to $bcraw/polfit_to_tgstrnew2.dat
- Plot transducer depth from target strength edited versus transducer depth from TP pressure sensor
- Use Plimpton's plotting software and command files.
- $bcplots/regtptg_ed.indat
- This creates $bcplots/regtptg_ed.gif
- See that difference between transducer depth from target strength edited and transducer depth from TP sensor is -4.57. Will use this in remap programs.
- Adjust velocities for sound velocity
- Use mean salinity of 34.86 from MicroCat (step G 7 above). Use 30 minute temperatures from TP to compute sound velocity at transducer. Every other 30 minute temperature average is used since ADCP data are collected in first 15 minutes of the hour.
- $bcprog/fix_surfsndvel < fix_surfsndvel_ca9.in
- Creates $bcraw/ca09_tc.vel
- No rotation of CA09 data are required to correct for compass pre/post calibrations (pre compass calibration was -.10, post calibration was 0.40). If a rotation was required use the following command.
- $bcprog/rotate_adcp < rotate_ca8.in
- Compute the mean sigmat between the surface and the mean transducer depth

- Using historical ctd file (mean_ctd.140w)

  $bcprog/compute_sigmat_2 < $bcproc/sigmat.in

  For 292dbmean sigma_t = 25.9239

  Average density = 1.02593

- Map profile data to standard depths
- File format for **$bcproc/remap_data_ca9.in file**
- Input:

  nArrays new_ymin new_ydel new_numbins outfile theta nom_bw nom_pulse nom_blnk vel_or_amp tlat mean_tdepth(z) depth_adj sidelobe_deepening:

  narrays = 1,

  new_ymin=new start depth,

  new_ydel=new delta depth,

  new_numbins=new number of depth bins,

  outfile,

  theta=20degrees,

  nom_bin=nominal bin width=8,

  nom_pulse=nominal pulse length=8,

  nom_blnk=nominal blanking=4,

  whether you are processing a velocity or and amplitude file (amplitude signal is collected in the last 1/4 of the pulse),

  tlat=latitude,

  mean transducer depth in meters (not db) adjusted for 2.0m difference between the transducer and the pressure sensor = 290m,

  depth_adj = depth adjustment determined from the regression between the target strength transducer depth and the pressure sensor transducer depth = 4.57,

  amount to deepen sidelobe cutoff depth (=2. if adjust depth)

  depthoffset,density,itype,transdepth/1stgooddepth_outfilename,actual_depth_outfilename:

  depth difference between transducer and pressure sensor = 2.0 from log sheets, average density computed above,

itype = 1 means do not set data bad if shallower than sidelobe cutoff and itype = 0 means set data bad if shallower than sidelobe cutoff,

name of outfile to output transducer depth,

shallowest good depth for profile calculated from (trans_depth*(1-cos(theta))+bin_width +sidelobe adj) and 1st good mapped depth,

name of outfile to output actual depth,u,v,err or actual depth,

beam1 amp,

beam2 amp,

beam4 amp

next lines are standard read_array inputs

the num_ctd_depths num_ctd_var & ctd_file_name nskip ctd_ydel are the same for all deployments which have max depths less than 500m.

- This is the contents of the file **$bcproc/remap_data_ca9.in**

1 -115. 5. 87 'bcraw/ca09_m115_5_adjdepth.amp' 20. 8. 8. 4. 'amp' 0.0 290 4.57 2.

2.0 1.02593 1 'junk.out' 'bcraw/ca09_m115_5_actd_adjdepth.amp'

2 00 00 20 9 03 20 00 11 9 04 0 0 .041666667

'bcraw/ca09.amp' T T T T F F 1 44

7 00 00 20 9 03 20 00 11 9 04 1 1 .041666667

'bcsen/pres009chlf.ram' '(4x,i2,i3,2i2,2x,f9.2)' 5 300 -99.99

500 2

'bcproc/mean_snd.140w' 1 1.

$ exit

$bcprog/remap_subdata_adjustdepth < $bcproc/remap_data_ca9.in

mapped amp from -115 to 315, no sidelobe cutoff

1 -115. 5. 87 'bcraw/ca09_m115_5_adjdepth.vel' 20. 8. 8. 4. 'vel' 0.0 290 4.57 2.

2.0 1.02593 1 'junk.out' 'bcraw/ca09_m115_5_actd_adjdepth.vel'

2 00 00 20 9 03 20 00 11 9 04 0 0 .041666667

'bcraw/ca09_tc.vel' T T T T F F 1 44

7 00 00 20 9 03 20 00 11 9 04 1 1 .041666667

```
'bcsen/pres009chlf.ram' '(4x,i2,i3,2i2,2x,f9.2)' 5  300 -99.99
500 2
'bcproc/mean_snd.140w' 1 1.
$ exit
$bcprog/remap_subdata_adjustdepth < $bcproc/remap_data_ca9.in
mapped vel from -115 to 315, no sidelobe cutoff
1 10. 5. 62  'bcraw/ca09_10_5_adjdepth.vel' 20. 8. 8. 4. 'vel' 0.0 290 4.57 2.
2.0 1.02593 2 'bcraw/ca09_1stdepth_adjdepth.out' 'junk.out'
2  00 00 20 9 03 20 00 11 9 04 0 0 .041666667
'bcraw/ca09_tc.vel' T T T T F F 1 44
7  00 00 20 9 03 20 00 11 9 04 1 1 .041666667
'bcsen/pres009chlf.ram' '(4x,i2,i3,2i2,2x,f9.2)' 5  300 -99.99
500 2
'bcproc/mean_snd.140w' 1 1.
$ exit
$bcprog/remap_subdata_adjustdepth < $bcproc/remap_data_ca9.in
mapped vel from 10 to 315, with sidelobe cutoff
1 10. 5. 62  'bcraw/ca09_10_5_adjdepth.amp' 20. 8. 8. 4. 'amp' 0.0 290 4.57 2.
2.0 1.02593 2 'bcraw/ca09_1stdepth_adjdepth_amp.out' 'junk.out'
2  00 00 20 9 03 20 00 11 9 04 0 0 .041666667
'bcraw/ca09.amp' T T T T F F 1 44
7  00 00 20 9 03 20 00 11 9 04 1 1 .041666667
'bcsen/pres009chlf.ram' '(4x,i2,i3,2i2,2x,f9.2)' 5  300 -99.99
500 2
'bcproc/mean_snd.140w' 1 1.
$ exit
$bcprog/remap_subdata_adjustdepth < $bcproc/remap_data_ca9.in
mapped amp from 10 to 315, with sidelobe cutoff
```

- This program **(remap_subdata_adjustdepth.f)** is run four times as described below:
- First run:

  First map all the depths, even those above the surface and do not set data bad for sidelobe cutoff:

  $bcprog/remap_subdata_adjustdepth < $bcproc/remap_data_ca9.in

  Maps velocity (vel) from -115 to 315 with no sidelobe cutoff.

  BOLD is user input.

  Make daily average:

  $bcprog/adcp_avg3

  Program ADCP_avg3

  Enter input filename: **bcraw/ca09_m115_5_adjdepth.vel**

  Enter output filename: **bcraw/ca09d_m115_5_adjdepth.vel**

  Starting and ending ensemble dates? **0 00 20 9 03 23 00 10 9 04**

  Number of ensembles over which to average? **24**

  Minimum good per average? **12**

  Bad data flag? **5120**

  Make contour plots to make sure everything looks ok

  Use Plimptons plotting software

  $bcplots/cc_remap_vel.indat

  Makes: $bcplots/uvcontour.gif
- Second run:

  $bcprog/remap_subdata_adjustdepth < $bcproc/remap_data_ca9.in

  Maps amplitude (amp) from -115 to 315 with no sidelobe cutoff.

  Make daily average:

  $bcprog/adcp_avg3

  Program ADCP_avg3

  Enter input filename: **bcraw/ca09_m115_5_adjdepth.amp**

  Enter output filename: **bcraw/ca09d_m115_5_adjdepth.amp**

  Starting and ending ensemble dates? **0 00 20 9 03 23 00 10 9 04**

  Number of ensembles over which to average? **24**

Minimum good per average? **12**

Bad data flag? **5120**

Make contour plots to make sure everything looks ok

Use Plimptons plotting software

$bcplots/cc_remap_amp.indat

Makes: $bcplots/beam1.gif beam2.gif beam3.gif beam4.gif

- Next map the depths beginning at a 10m depth with 5 meter delta depth and set data bad if shallower than sidelobe cutoff depth:

- Third run:

$bcprog/remap_subdata_adjustdepth < $bcproc/remap_data_ca9.in

Maps velocity (vel) from 10 to 315 with sidelobe cutoff.

Make daily average:

$bcprog/adcp_avg3

Program ADCP_avg3

Enter input filename: **bcraw/ca09_10_5_adjdepth.vel**

Enter output filename: **bcraw/ca09d_10_5_adjdepth.vel**

Starting and ending ensemble dates? **0 00 20 9 03 23 00 10 9 04**

Number of ensembles over which to average? **24**

Minimum good per average? **12**

Bad data flag? **5120**

- Fourth run:

$bcprog/remap_subdata_adjustdepth < $bcproc/remap_data_ca9.in

Maps amplitude (amp) from 10 to 315 with sidelobe cutoff.

Make daily average:

$bcprog/adcp_avg3

Program ADCP_avg3

Enter input filename: **bcraw/ca09_10_5_adjdepth.amp**

Enter output filename: **bcraw/ca09d_10_5_adjdepth.amp**

Starting and ending ensemble dates? **0 00 20 9 03 23 00 10 9 04**

Number of ensembles over which to average? **24**

Minimum good per average? **12**

Bad data flag? **5120**

- Make sample hourly profile plots

- Make sample hourly profile plots throughout the deployment to make sure that sidelobe cutoff depth is deep enough and profiles of velocity and amplitude look right with respect to the surface. Overplot the mapped data (dashed) (bc/ca09_10_5_adjdepth.vel) on the unmapped data (line) (bcraw/ca09_m115_5_actd_adjdepth.vel)

Use Plimptons plotting software: uniplt plotting program

$bcplots/pl_prof.indat

Creates $bcplots/prof1.gif prof2.gif prof3.gif as samples

- Make mapped extrapolated ADCP velocity file

- Make mapped ADCP velocity file that is extrapolated to shallower depths to inter-compare with Sontek Argonaut velocities

- Can put in the number of bins to extrapolate – use 4 bins to get 5m value most of the time with depths of this deployment:

$bcprog/remap_subdata_extrapolate_set < $bcproc/remap_extrapolate.in

Creates $bcraw/ca09_5_5_adjdepth_ext4.vel

Make daily average:

$bcprog/adcp_avg3

Program ADCP_avg3

Enter input filename: **bcraw/ca09_5_5_adjdepth_ext4.vel**

Enter output filename: **bcraw/ca09d_5_5_adjdepth_ext4.vel**

Starting and ending ensemble dates? **0 00 20 9 03 23 00 10 9 04**

Number of ensembles over which to average? **24**

Minimum good per average? **12**

Bad data flag? **5120**

- Make regression plots

- Make regression plots to make sure that the ADCP data agrees with the Sontek Argonaut data (deployment pm395 and pm438) to 5 cm/s in speed and 5 degrees in direction. Exclude directions when the daily speed is less than 5.0 cm/s for either deployment.

  Use Plimptons plotting software: uniplt plotting program

  $bcplots/pltreg_all_pm438.indat

  Creates: $bcplots/pltreg_all_pm438a.gif and $bcplots/pltreg_all_pm438b.gif

- Convert ADCP data to hourly and daily

- Convert format of the ADCP data (hourly and daily) to give to DQA for TAO web posting.

  $bcprog/rdipak_mem < $bcproc/pakrdi140in

  Creates $bcraw/ca9h.adcp and $bcraw/ca9d.adcp

- These files go to the TAO web master for posting on the web page.

- This completes the ADCP data QA/QC processing.

2.3.3 Ocean Subsurface Pressure Measurement

2.3.3.1 Description: The subsurface pressure measurement is taken utilizing a Paine: 211-30-660-01 transducer. It has a resolution of 0.03 psi, a measured range of 400-800 psi and a full range of 0-1000 psi. The transducer has an accuracy of ± 0.25% at the full scale of 1000 psi and the measurement is taken at the 300meters and 500meters depths on all the ATLAS moorings. There are no pressure instruments on the ADCP.

It is extremely rare that module pressures are flagged. The pressure sensors are fairly stable and have perhaps never needed flagging. If a module record ends early due to battery failure, the tail end of the data may be random numbers, and need flagging. However, the module pressures do provide you useful information about the mooring motion, and allow you to know when the modules (and hence their temperature readings) are not at the correct depth. To repeat, module pressures are never flagged, but they are used to flag temperatures sometimes.

The pressure sensors record vertical motion of the mooring. When something pushes (or pulls) on the buoy, the mooring is pulled out of the vertical, and the depths of all the modules decrease. No flagging is performed to account for the "natural" motions of the buoys, which usually have periods of longer than a day and amplitudes on the order of 10 meters at 500 meters. However, temperature quality codes are changed for temperatures that correspond to pressures falling outside the normal expected range. Pressures outside this range comprise less than 1% of nominally good data. The normal range for the 300m pressures is 293db - 305db, while the range for 500m pressures is 488db - 508db. A quality code of 4 (Q4) is set for 500m temperatures when the 500m pressures fall outside this normal range, and Q4 is set for temperatures between 300m and just below the SST if 300m pressures fall outside their normal range (or if there are no 300m data and the 500m pressures fall outside their normal range). Range-checking and flagging is automated via the "find_low_press" perl script (see below). Inverse catenary's moorings have much more natural vertical motion than taut moorings, and may drift up and down tens of meters.

The sampling rate of the subsurface pressure referenced in table 2-5, indicates the intervals at which the measurements are taken. The data that are received by the NDBC contains the previous day's daily mean and the latest 2 minute mean of the current day at the top of the most recent hour at the time of transmission.

Table 2-5, Subsurface Pressure Sampling Information

Measurement	Sample rate	Sample Period	Sample Time	Data Recorded in Memory	Transmitted Data
Subsurface Pressure	1 per 10 minute	instantaneous	0000,0010… begin time	10 Minutes	Daily mean

2.3.3.2 Delayed Mode (RAM) Procedures: Problems arise when a ship suddenly yanks hard on a buoy. This is presumed to be due to fishing activities. The 500m pressure can decrease by 50 or more decibars, and noticeable temperature spikes can show up in the module data at quite shallow depths. These events tend to be less than one day long, with very sharp changes in pressure. If one of these occurs, all the temperatures, conductivities, salinity's and densities from modules below the SST* are flagged from the first decreased pressure time point until one half hour after the last decreased pressure time point. This is done to prevent biasing the data from a given depth with data collected at a depth with a substantially different temperature. The extra half-hour is to allow the modules to re-equilibrate their temperatures to their "native" depth.

A perl program called `find_low_press` can perform this flagging automatically. In addition, the program automatically sets the Q4 flags for the temperatures corresponding to out-of-range pressures. To run the program:

- Type `find_low_press` in the directory containing the relevant pressure data.
- The program will prompt you for the pressure flag file. Enter the pressure file in the current working directory containing points to be flagged (presXXXX.flg).
- The next prompt can be ignored (press return) unless you've run into a "wuzzle": Occasionally you may encounter pressure data that are offset because a kink ("wuzzle") shortened the nilspin and prevented the modules below the wuzzle from reaching their planned depths. In such cases, the average temperature gradient between the deepest 2 modules is used to interpolate the temperature at 300m and to extrapolate the 500m temperature. The actual temperatures at these depths are adjusted by using the flag file. For example, if there was a knot in the nilspin between the 300 and 500m modules, with 500m pressures having a median of 490db, and the 300m and 500m temperatures had means of 11.23 and 8.10 Celsius, respectively, then a linear gradient of [8.10-11.23]/[490-300] = -0.01647 ° C/m would be assumed and the 500m temperatures would be shifted by -0.165 ° C). In these cases, the pressure offset needs to be entered into find_low_press so that the program's range boundaries can be offset accordingly. Hence the following two prompts: "If the 300m pressures are offset by some (+ or -) number of db, please enter that number: [0]". "If the 500m pressures are offset by some (+ or -) number of db, please enter that number: [0]".
- After these prompts, the program will perform the pressure range checks and set Q4 codes where necessary, writing the resulting changes into a pressure.txt file in the current directory.
- Next it will ask, do you want to flag vandalism spikes? (N for no, anything else is a yes).
- A "yes" response will result in the prompt: "Please enter the acceptable variation of pressures from their nominal 300db and 500db means (default:

8db)”. This is the maximum variation of any given pressure from the mean that will not induce flagging.

- The subsequent prompt asks for the maximum jump between adjacent pressure values that will not induce flagging: “Please enter a maximum jump size”. Adjacent pressures differing by more than this value will be considered 'vandalism events'.

- Pressure jumps greater than either the jump-interval or than 1 deviation from the nominal (300db or 500db) mean will register as "vandalism events", and the resulting flags will be listed in pressure.txt. Copy and paste flags (including quality code changes) from pressure.txt into flag###X.txt before running flag_data to perform the actual flagging. Check the flag list before applying it, and delete flags as appropriate. In particular, there may be extraneous flags included if you are using the default criteria for finding vandalism spikes, since these criteria are set to be stringent in order not to miss any potential vandalism events. The vandalism flags will have "w1" weights assigned, so that flag_data will perform them after the Q4 flagging is completed. This insures that a flagged value will not end up with a quality code of 4 (instead of the correct 5 for "failure"). Note: On some moorings processed in the past, only temperatures below the mixed layer (determined from a time-temperature plot of the event) were flagged, but due to the subjectivity of this approach, it was altered so that now all modules below the SST are flagged.

2.3.4 Ocean Subsurface Temperature Measurement

2.3.4.1 Description: The subsurface temperature (T) measurements are taken using 3 different sensor types. The original type was a Seacat SBE 16 produced by Sea Bird Electronics. The Seacat has a resolution of .001°C, a total range of -5 to 35°C and a measured range of 1-31°C, and an accuracy of ± .003°C. This sensor is being replaced by the Microcat SBE 37 produced by Sea Bird Electronics due to the Seacat 16 no longer being produced. The Microcat was used instead of the upgraded Seacat 16 plus because it has the same specifications as the original Seacat 16, so no evaluation of sensor performance had to be performed, **reference 4**. The nine NX ATLAS moorings are utilizing a Thermistor 46006 produced by Yellow Springs Instruments. It currently undergoing evaluations but current specifications is a resolution of .001°C, total range of 0 to 40°C with a measured range of 6-32°C, and an accuracy of ±.02°C, **reference 4 and reference 5**. The outer casings of the sensor modules are not being used. They are custom fabricated by PMEL just the electronic packages are being used.

The sampling rate of the subsurface temperature referenced in table 2-6, indicates the intervals at which the measurements are taken. The data that are received by the NDBC contains the previous day's daily mean and the latest 2 minute mean of the current day at the top of the most recent hour at the time of transmission, **reference 2**.

Table 2-6, Subsurface Temperature Sampling Information

Measurement	Sample rate	Sample Period	Sample Time	Data Recorded in Memory	Transmitted Data
Subsurface Temperature	1 per 10 minute	instantaneous	0000,0010… begin time	10 Minutes	Daily mean

The tables 2-7 below outlines the depths at which the subsurface temperatures are taken based on the longitude of that area of the array.

Table 2-7, Subsurface Temperature Measurement Depths

Pacific Ocean	SST	T1	T2	T3	T4	T5	T6	T7	T8	TP9	TP10
Depths (m) east of 155°W	1	20	40	60	80	100	120	140	180	300	500
Depths (m) at and west of 155°W	1	25	50	75	100	125	150	200	250	300	500

2.3.4.2 Delayed Mode Procedures: The following procedures and recommended QC activity all assume that basic delayed mode Tube and Module data processing and general delayed mode QC, paragraph 3.3, steps have been completed. This includes running the “flag data” program as many times as needed to handle data processing errors and creating the temperature plots discussed in these procedures.

- Temperature data are not usually problematic. If the buffers become garbled, you will have bad temperatures; this often occurs right before the module stops logging due to dead batteries. Sometimes a calibration will be slightly off, and the "difference plots" will show negative values in the mixed layer. Other times temperature inversions are real, usually not for more than a few days at a time. Shallow modules (but not usually SST, which is shaded by the buoy) often show solar heating during the daytime, which is worth noting.

- A moderately common problem is for the deployment or recovery time in the ***PM###X.CAL*** file to be wrong. Ideally, these times are "anchor drop" and the first time anything was done to the buoy on recovery. The latter could be the release fire, or could be a small boat visit to unplug sensors. The ***flag_data*** program truncates the ***.flg*** files at 2 hours after the deployment time, and right at the recovery time. The 2 hours **should** allow the mooring to settle and module temperatures to equilibrate before data are used. However, sometimes this isn't enough time, particularly for inverse catenary’s (slack) moorings, and you might need to adjust the time. The recovery time is more often incorrect, for a variety of reasons. To look for either case, look closely at the ends of the plots: temperature spikes and pressure dips are usually due to incorrect dates.

- An important thing to do is to look at the ***plotmod.m*** plot at the beginning and end of each deployment. Check to see that the temperatures all go to "normal" (i.e. ~25 at the surface, ~10 at 500m) temperatures at the time they are supposed to have been deployed. Sometimes clocks are set wrong before they are deployed and 1 or more modules might start on the wrong day. A more common problem is for the recovery date to not match the time stamps from one or more modules. See below. If either of these is the case, there are a variety of programs on LINUX in ***/pmel/dist/processing/delayed_process/nxram/bin/*** that can fix it, mostly the ***shift_days*** programs.

2.3.5 Sea Surface Temperature Measurement

2.3.5.1 Description: The sea surface temperature (SST) measurements are taken using 3 different sensor types. The original type was a Seacat SBE 16 produced by Sea Bird Electronics. The Seacat has a resolution of .001°C, a total range of -5 to 35°C and a measured range of 1-31°C, and an accuracy of + .003°C. This sensor is being replaced by the Microcat SBE 37 produced by Sea Bird Electronics due to the Seacat 16 no longer being produced. The Microcat was used instead of the upgraded Seacat 16 plus because it has the same specifications as the original Seacat 16, so no evaluation of sensor performance had to be performed, **reference 4**. The nine NX ATLAS moorings are utilizing a Thermistor 46006 produced by Yellow Springs Instruments. It currently undergoing evaluations but current specifications is a resolution of .001°C, total range of 0 to 40°C with a measured range of 6-32°C, and an accuracy of + .02°C, **reference 4 and reference 5**. The outer casings of the sensor modules are not being used. They are custom fabricated by PMEL just the electronic packages are being used.

The sampling rate of the sea surface temperature referenced in table 2-8, indicates the intervals at which the measurements are taken. The data that are received by the NDBC contains the previous day's daily mean and the latest 2 minute mean of the current day at the top of the most recent hour at the time of transmission, **reference 2**.

Table 2-8, Sea Surface Temperature Sampling Information

Measurement	Sample rate	Sample Period	Sample Time	Data Recorded in Memory	Transmitted Data
Sea Surface Temperature	1 per 10 minute	instantaneous	0000,0010… begin time	10 Minutes	Daily mean

2.3.5.2 Delayed Mode (RAM) Procedures: The following procedures and recommended QC activity all assume that basic delayed mode Tube and Module data processing and general delayed mode QC, paragraph 3.3, steps have been completed. This includes running the "flag data" program as many times as needed to handle data processing errors and creating the temperature plots discussed in these procedures.

- Temperature data are not usually problematic. If the buffers become garbled, you will have bad temperatures; this often occurs right before the module stops logging due to dead batteries. Sometimes a calibration will be slightly off, and the "difference plots" will show negative values in the mixed layer. Other times temperature inversions are real, usually not for more than a few days at a time. Shallow modules (but not usually SST, which is shaded by the buoy) often show solar heating during the daytime, which is worth noting.

- A moderately common problem is for the deployment or recovery time in the ***PM###X.CAL*** file to be wrong. Ideally, these times are "anchor drop" and the first time anything was done to the buoy on recovery. The latter could be the release fire, or could be a small boat visit to unplug sensors. The ***flag_data*** program truncates the ***.flg*** files at 2 hours after the deployment time, and right at the recovery time. The 2 hours **should** allow the mooring to settle and module temperatures to equilibrate before data are used. However, sometimes this isn't enough time, particularly for inverse catenary's (slack) moorings, and you might need to adjust the time. The recovery time is more often incorrect, for a variety of reasons. To look for either case, look closely at the ends of the plots: temperature spikes and pressure dips are usually due to incorrect dates.
- An important thing to do is to look at the ***plotmod.m*** plot at the beginning and end of each deployment. Check to see that the temperatures all go to "normal" (i.e. ~25 at the surface, ~10 at 500m) temperatures at the time they are supposed to have been deployed. Sometimes clocks are set wrong before they are deployed and 1 or more modules might start on the wrong day. A more common problem is for the recovery date to not match the time stamps from one or more modules. See below. If either of these is the case, there are a variety of programs on LINUX in ***/pmel/dist/processing/delayed_process/nxram/bin/*** that can fix it, mostly the ***shift_days*** programs.

2.3.6 Air Temperature Measurement

2.3.6.1 Description: The air temperature (AT) measurement is taken using a Model MP101A combined humidity and air temperature probes manufactured by Rotronic Instrument Corporation of Huntington, NY, mounted on all 55 ATLAS moorings. The sensor utilizes a Pt100 RTD (Resistance Temperature Detector) with specifications of $\pm$ 0.2^{o} C accuracy, 0.4 ^{o} C drift/year, and $\pm$ 0.1^{o} C repeatability. The air temperature sensor is pre-calibrated with the equation $AT\ (^{o}C) = B_{at}V_{at}$ where *V= sensor voltage (-0.4 to +0.6), which equals an output of -40^{o}C to 60^{o}C, and B= calibration coefficients with a nominal value of 100.* The sensor is mounted on the ATLAS mooring at a height of 2.5 meters above ground level (AGL), **reference 1**.

The sampling rate of the air temperature, referenced in table 2-9, indicates the intervals at which the measurements are taken. The data that is received by the NDBC contains the previous days daily mean and the latest 2 minute mean of the current day at the top of the most recent hour at the time of transmission, **reference 2**.

Table 2-9, Air Temperature Sampling Information

Measurement	Sample rate	Sample Period	Sample Time	Data Recorded in Memory	Transmitted Data
Air Temperature	2 – hz	2 minutes	2359-0001 begin time	10 Minutes	Daily mean and 2 minute mean from top of most recent hour

2.3.6.2 Delayed Mode Procedures: The following procedures and recommended QC activity all assume that basic delayed mode Tube and Module data processing and general delayed mode QC, paragraph 3.3, steps have been completed. This includes running the “flag data” program as many times as needed to handle data processing errors and creating the temperature plots discussed in these procedures.

- Air temperature and humidity sensors are both built into the same physical housing and sometimes fail simultaneously. Both also have similar failure modes. There are, of course, physical sensor failures for which the good data end abruptly. You will probably need to flag all values after a failure like this. Sometimes these sensors have sharp spikes or dips which need to be flagged. But sometimes humidity, particularly, can have real dips over short periods (days). These sensors can also slowly drift up or down. It can be very difficult to determine if and when this happens. One thing that helps sometimes is to look at plots of the real-time data with the climatologic means superimposed. Often temperature and humidity fail at the same time.

2.3.7 Wind Measurement

2.3.7.1 Description: The wind measurement is taken using a R.M. Young/05103 Propeller and Vane and EG&G/63764 or KVH/LP101-5 fluxgate compass mounted on all 55 ATLAS moorings. The sensor utilizes a helicoid anemometer with propeller specifications of 4 blades 18 cm diameter X 30 cm pitch, max speed of 60 m/s, threshold of 1 m/sand accuracy of + 0.3 m/s. The vane specifications are a 10 kilo ohm precision potentiometer, range of 0-355o, threshold of 1.1 m/s; accuracy $\pm 3^{o}$. The compass has a range of 0-360 o and an accuracy of 5^{o}. The wind speed is determined by the square root of (U2 + V2) and the wind direction is the plot intersection of “U” and “V”. The sensor is mounted on the ATLAS mooring at a height of 4 meters above ground level (AGL), **reference 3**.

The sampling rate of the wind sensor referenced in table 2-10, indicates the intervals at which the measurements are taken. The data that is received by the NDBC contains the previous days daily mean and the latest 2 minute mean of the current day at the top of the most recent hour at the time of transmission, **reference 2**.

Table 2-10, Wind Sampling Information

Measurement	Sample rate	Sample Period	Sample Time	Data Recorded in Memory	Transmitted Data
Wind	2 – hz	2 minutes	2359-0001 begin time	10 Minutes	Daily mean and 2 minute mean from top of most recent hour

2.3.7.2 Delayed Mode Procedures: The following procedures and recommended QC activity all assume that basic delayed mode Tube and Module data processing and general delayed mode QC, paragraph 3.3, steps have been completed. This includes running the “flag data” program as many times as needed to handle data processing errors and creating the temperature plots discussed in these procedures.

- The first page (speed, direction and "stick" plot) is generally the most useful. Look for the data ending early, gaps, obvious bad values like wind speeds of 50 m/s, direction constant, direction random etc. At gaps (caused by tube resets or sensor failure), be sure to look at the .ram file and/or plot it to see what times need to be flagged. Winds are not "self-flagging", i.e. if the sensor fails or is unplugged, values generally go to zero or some small value. As you don't really know what the winds are, you need to flag all those values. Be sure to check the last few lines of the met###X.flg file for this problem.

- Winds can go bad for two reasons which are not at all obvious. The tube can lose its compass, or the vane reading for the direction can become fixed. In either case, the plot of winds will not necessarily show any problems. If the compass fails, the vane reading still changes as the buoy rotates or wind changes direction. And if the vane fails, the compass still changes as the buoy rotates. So if only one fails, the wind directions still vary over time. For newer wind data (e.g. tube versions 2.24, 4.05), you can find out if either of these problems occurred by checking the .ram files, which will have additional columns containing the compass, vane and other data (see a more detailed description below). These columns are removed from the files by the flag_data program, and so are not present in the .FLG files or at any later stages of processing. For older wind data, there are two places to find out if either of these problems occurred. One is to look at the 11th column of hex data in the wt###X_r.dat file produced by the RAM processing to match the formatting of real-time data. The other is to check the 11th column of hex data in real-time itself, under the TAO internal home page, Data/Data monitoring files/'check latest transmissions, (The address is http://port01/monitor-dyn.html). Simply enter the ptt number (located at the top of the cal and doc files) and a large enough number of 'days previous' to span the deployment you want to check, then select the 'wind/humidity/conductivity' data type. The right 2 hex characters are a spot vane reading, and the left 2 hex characters are a spot compass reading. If either doesn't change from day to day (often becoming FF or 00), that sensor has failed. This problem should be noted in the pm###.doc file, which is why it's important to read that document. If this occurs, the U, V and DIR columns (1, 2 and 4) of the met###X.ram file should be flagged.
- Another unusual problem for winds is that the tube can lose its "compass table". This table corrects the compass directions calculated for different orientations. If the tube is not using it, the wind directions could be substantially off. This also should be noted in the pm###.doc file, but is recorded only in the 13th column of the real-time data. If this column is "0000", things are ok. If not, the compass table wasn't being used and the wind's directions may need to be flagged.
- Enhanced RAM files from newer tube versions: Compass and vane tracking can be done more precisely in recent tube software versions (e.g. versions 2.24, 4.05) by looking at the met###X.ram file, which contains 10 minute values of:
- **CO**, the most recent 1-sec compass reading. The output number has been calibrated; i.e. it is in degrees, not counts. To convert to counts, multiply by 256/360. If these are not changing, or zero, it's likely something was wrong with the compass.
- **VA**, the most recent 1-sec anemometer vane reading. The output number has been calibrated; i.e. it is in degrees, not counts. To convert to counts,

multiply by 256/360. If these are not changing or zero, it's likely something was wrong with the vane potentiometer

- **SP**, the most recent 1-sec anemometer speed reading. The output number has been calibrated; i.e. it is in m/s, not counts. If these go to zero for an extended period, there is a problem with the anemometer.
- **FLAG**, a diagnostic status word. Should be 0 if the tube has had no problems. If the "compass table" stored on board the tube to correct the compass readings for improved accuracy has been lost, the 8th bit will be set. If that is the only problem, the value will be 128 (80 hex). If other bits are set, the value could be some other non-zero value.

2.3.8 Barometric Pressure Measurement

2.3.8.1 Description: The barometric pressure measurement is taken by a Paroscientific: MET1-2 pressure transducer. The transducers specifications are a resolution of 0.1hPa, range of 800-1100hPa, and an accuracy of + 0.01% of reading. The sensor is mounted 3 meters AGL on the ATLAS Mooring and located at 3 locations longitudinally at 2S 110W, 0 110W, and 2N 110W. Figure 2.3.8.1 shows where the sensor is mounted on the mooring, reference 1.

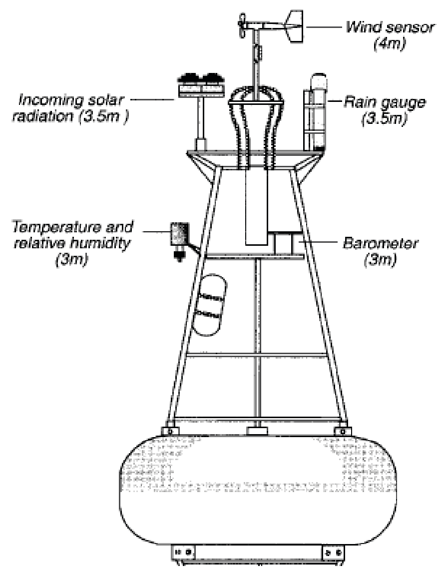


Figure 2.3.8.1 ATLAS Mooring Meteorological Sensor Configuration

The sampling rate of the barometric sensor is listed in table 2-11. The data that is received by the NDBC contains the previous days daily mean and the latest 2 minute mean of the current day at the top of the most recent hour at the time of transmission, reference 2.

Table 2-11 Barometric Pressure Sampling Rate

Measurement	Sample rate	Sample Period	Sample Time	Data Recorded in Memory	Transmitted Data
Barometric Pressure	2 – hz	2 minutes	2359-0001 begin time	Hourly Recording	Daily mean and 2 minute mean from top of most recent hour

2.3.8.2 Delayed Mode Procedures: The following procedures and recommended QC activity all assume that basic delayed mode Tube and Module data processing and general delayed mode QC, paragraph 3.3, steps have been completed. This includes running the “flag data” program as many times as needed to handle data processing errors and creating the temperature plots discussed in these procedures.

- Barometric sensor measurements are fairly trouble free and require no specific delayed mode QC processes outside the general check procedures.

2.3.9 Rainfall Measurement

2.3.9.1 Description: The rainfall measurement is taken with a R.M. Young: 50203-34 capacitance rain gauge mounted on 28 TAO ATLAS moorings. Sensor specifications are a resolution of 0.2 mm/hr, range of 0-50 mm, and an accuracy of + 0.4 mm on 10 minute filtered data. The measuring tube has a maximum capacity of 500mL, equivalent to 50 mm of rain between siphon events. The volume of water in the tube is determined by capacitance. A stainless steel capacitive probe covered with mylar is located at the center of the measuring tube. The capacitance is measured between the steel probe and the water, with the mylar acting as the dielectric between these two ‘‘plates’’ of the capacitor. An outer stainless steel probe surrounding the inner mylar coated probe closes the circuit. As the water level rises, increasing the surface area in touch with the mylar, the capacitance increases. A PMEL-designed circuit converts the capacitance to frequency, which is then averaged over 1-min intervals and output as a digital signal (counts). The conversion from counts to volume is given by the equation V= a/N + b where V (in mL) is the volume in the measuring tube, N is the number of counts, and a (in mL-counts) and b (in mL) are calibration coefficients to be determined through a least squares regression of V on N21. Mean coefficients for over 100 sensor calibrations are 5 6.5 3 107 and b 5 21100. The 1-min volume samples are stored on board the mooring while at sea, and are available for post processing on recovery. While the mooring is deployed, the daily mean and standard deviation of rain rate, and the daily percent time raining, are sent to PMEL in real time via Service Argos, Inc. Figure 2.3.9.1 shows where the sensor is mounted and a breakdown of the components, **reference 6**.

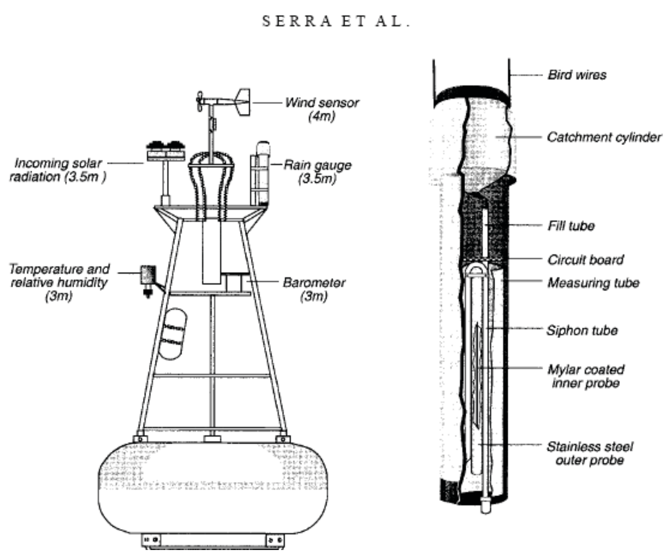


Figure 2.3.9.1 ATLAS Mooring and Rain Gauge Configuration

The rain gauge accuracy is based on controlled environmental data, but deployment has indicated that there are several factors which can contribute to causing errors in the measurement value. These are:

- Dry period noise

- Volume dependency
- Temperature sensitivity
- Evaporation and sea spray
- Wind

The sampling rate of the rainfall measurement is outlined in table 2.12.

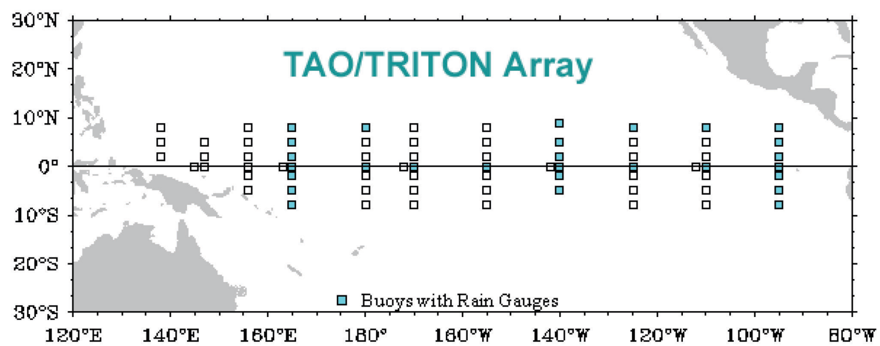
Table 2.12 Rainfall Sampling Rate

Rain rate	1-hz	1 min	0000-0001, 0001-0002,...	1 min	Daily mean, standard deviation, and percent time raining

To reduce instrumental noise, internally recorded 1-minute rain accumulation values are smoothed with a 16 minute Hanning filter upon recovery. These smoothed data are then differenced at 10-minute internals and converted to rain rates in mm/hr. The resultant rain rate values are centered at times coincident with other 10-minute data.

Buoys that measure Rainfall are in table 2-13.

Table 2-13 TAO Moorings that Measure Rainfall



2.3.9.2 Delayed Mode Check Procedures: Rainfall data are collected using a RM Young rain gauge, and recorded internally at a 1-min sample rate. The RM Young rain gauge consists of a 500 ml catchment cylinder which, when full, empties automatically via a siphon tube. Data from a 3-min period centered near siphon events are ignored. Occasional random spikes, which typically occur during periods of rapid rain accumulation, or immediately preceding or following siphon events, are eliminated manually.

Rain rates computed from first differences of 1-min accumulations are often noisy because of the sensitivity of rate calculations to noise in accumulations over short time scales. To reduce this noise, 1-min accumulations are filtered with a 16-point Hanning filter, and rates are computed at 10-min intervals. Residual noise in the filtered time series may include occasional spurious negative rain rates, but these rarely exceed a few mm hr. Serra et al (2001) [1] estimate the overall accuracy of 10-min data to be 0.3 mm hr on average.

The following procedures and recommended QC activity all assume that basic delayed mode Tube and Module data processing and general delayed mode QC, paragraph 3.3, steps has been completed. This includes running the "flag data" program as many times as needed to handle data processing errors and creating the temperature plots discussed in these procedures.

- In addition to the rain rate/accumulation plots you print after running plottube, you should also load the raw volume data (rain###X.flg) into Matlab (e.g., type [x,y,h]=a2load('/home/drumlin/data/nxram/pm181/rain181a.flg');), and plot and print them out, so this plot can be included with the other plots in the data notebook. Before filing the plot, examine it for abnormally rapid rain dips, siphons that exceed 550ml, and any spikes that occur away from siphon events (spikes at the leading edges of rain events are fairly common). It is useful to create an additional plot containing subplots which are 'blow-ups' of regions of suspicious-looking rain data. This aids in locating the dates and timings of any points that need to be hand-flagged. Such dates can be confirmed and pinpointed by browsing the rain###X.ram (or .flg) file itself.
- Keep in mind that while you flag the 1 minute data (RAIN###X.RAM), the final rain product is the 10 minute RAIN###X.RAT file. So many problems which appear in the 1 minute rates will go away in the 10 minute data, thanks to the smoothing algorithm implemented with rain_smooth. However, you should still fix any obvious problems in the 1 minute data.
- Generally, rain rates below about -3 mm/hr or above about 200 mm/hr (again in the 10 minute data), are assumed incorrect and fixed. One needs to examine in detail (by plotting or looking at the data file directly) the 1 minute data for the period in question and flag the points which are causing a problem. Often there is a large (or small) single point which is easily removed. Other times the volume dips for a half hour or so, then comes back to it's original volume. Sometimes a siphon event is improperly handled by rain_desiphon and resulting rates are very large - you'd have to understand the details of how the program works to know exactly what it will do. Some periods are excessively noisy, often when the gauge is full. There isn't anything to do for these, but flag them if they're bad enough. Often gauges are found at the end of the deployment to have their top sections (including funnel) missing, or tilted over at an angle. If either of these is the case (which should be described in the FileMaker database recovery log), you need to try to figure out when it happened. Often it is assumed that physical damage like this occurs at the same time as a ship visit is observed in the module pressure, or when a wind vane or radiometer is broken as well. If the funnel assembly is broken off, you can get a general idea of when as no rain should be recorded after that point. But one generally doesn't know exactly when this sort of vandalism occurs. If

all you have is no rain recorded after a certain point, you can use the last rain event as the last time you know it was working.

- Evaporation: When examining the .flg data, create 'blow-up' plots of any regions where rapid evaporation occurs. Use these to measure and calculate the evaporation rates (in ml/day, mm/day, and mm/hour). Higher rates can occur over very short time-periods. In general, the maximum reasonable evaporation rate is around 2 to 3 mm/day. Cross-compare evaporation data with related data i.e. SWR, wind, RH, AT, to help verify its validity. Often, invalid drops appear unnatural, with large vertical drops and squared-off profiles, but sometimes variations caused by the pic board electronics are gradual enough that they cannot be distinguished from real data. Note any suspicious evaporation rates (and dates) in the summary file.
- Diurnal fluctuations: When examining the .flg data, check for small diurnal oscillations in the data, which are usually most apparent in the 'rain-less' time periods. If they occur, check the version of the pic board used by the rain gauge (this can be found by entering the gauge's number into the 'rain' section of the mysql database) then cross-compare these data with the air temperature data. The electronics on version 3 or older pic boards were unstable outside the range of 14-38 degrees Celsius, leading to bad data that included oscillations and noise. If the oscillations in your data occur at temperatures outside the normal operating range, and the pic board used was version 3 or older, make a note in the summary file but otherwise don't worry since nothing can be done about it (except for a pic board upgrade). Version 3.5 (and higher) boards should compensate for temperature instability. Notify Dave Zimmerman if the oscillations in your data are large (>10ml) and either occur within the normal operating temperature range (if the pic version was 3 or older), or at any temperature (if the pic version was 3.5 or newer). Note the problem (along with the pic board version) in the summary file.
- A diagnostic parameter for the rain data is in the RAIN###X.RAM file from recently deployed tubes. N is the number of PIC board samples collected by the tube during the last sampling period. Nominally 60 (1 Hz for 60 sec), during ARGOS transmit window’s, the number will be around 50. If the tube could not talk to the PIC board, the number will be zero.

2.3.10 Relative Humidity Measurement

2.3.10.1 Description: The relative humidity (RH) measurement is taken using a Model MP101A combined humidity and air temperature probes manufactured by Rotronic Instrument Corporation of Huntington, NY, mounted on all 55 ATLAS moorings. The sensor utilizes a C94 Rotronic Hygromer thin film capacitive sensor protected beneath a Teflon foam filter cap with specifications of + 1.0% accuracy, less than 1.0% drift/year, and + 0.3% repeatability. The relative humidity sensor is pre-calibrated with the equation RH (%) = BrhVrh where V= sensor voltage 0 to 1), which equals an output of 0% to 100%, and B= calibration coefficients with a nominal value of 100. The sensor is mounted on the ATLAS mooring at a height of 2.5 meters above ground level (AGL), **reference 1**.

The sampling rate of the relative humidity referenced in table 2-14, indicates the intervals at which the measurements are taken. The data that is received by the NDBC contains the previous days daily mean and the latest 2 minute mean of the current day at the top of the most recent hour at the time of transmission, **reference 2**.

Table 2-14, Relative Humidity Sampling Information

Measurement	Sample rate	Sample Period	Sample Time	Data Recorded in Memory	Transmitted Data
Relative Humidity	2 – hz	2 minutes	2359-0001 begin time	10 Minutes	Daily mean and 2 minute mean from top of most recent hour

2.3.10.2 Delayed Mode Procedures: The following procedures and recommended QC activity all assume that basic delayed mode Tube and Module data processing and general delayed mode QC, paragraph 3.3, steps has been completed. This includes running the "flag data" program as many times as needed to handle data processing errors and creating the temperature plots discussed in these procedures.

- Air temperature and humidity sensors are both built into the same physical housing and sometimes fail simultaneously. Both also have similar failure modes. There are, of course, physical sensor failures for which the good data end abruptly. You will probably need to flag all values after a failure like this. Sometimes these sensors have sharp spikes or dips which need to be flagged. But sometimes humidity, particularly, can have real dips over short periods (days). These sensors can also slowly drift up or down. It can be very difficult to determine if and when this happens. One thing that helps sometimes is to look at plots of the real-time data with the climatologic means superimposed. Often temperature and humidity fail at the same time.

2.3.11 Shortwave Radiation Measurement

2.3.11.1 Description: The shortwave radiation measurement is taken by an Eppley Precision Spectral Pyranometer Model PSP with a Delrin Case. The pyranometer's specifications are a resolution of 0.4 W m-2, range of 0-1600 W m-2, measured range of 200-1000 W m-2, and an accuracy of + 1%. The sensor is mounted 3.5 meters ASL on the ATLAS Mooring and located at 11 locations longitudinally at 0110W, 0125W, 0140W, 0170W, 8N165E, 5N165E, 2N165E, 0165E, 2S165E, 5S165E, and 8S165E. Figure 2.3.11.1 shows where the sensor is mounted on the mooring, **reference 1**.

Shortwave radiation data is encoded in three adjacent 16 bit words. Prior to ATLAS software version 2.24 and 4.06, daily average values were made using samples from 12 hours centered on local noon. Later versions compute daily average from 24 hourly.

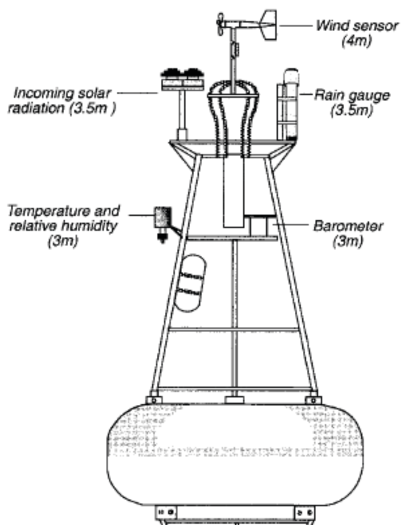


Figure 2.3.11.1 ATLAS Mooring Meteorological Sensor Configuration

The sampling rate of the shortwave radiation sensor is listed in table 2-15. The data that is received by the NDBC contains the previous day's daily mean, **reference 2**.

Table 2-15 Shortwave Radiation Sampling Rate

Measurement	Sample rate	Sample Period	Sample Time	Data Recorded in Memory	Transmitted Data
Shortwave Radiation	2 – hz	2 minutes	2359-0001 begin time	10 Min.	Daily mean

3.3.11.5 Delayed Mode Procedures: The following procedures and recommended QC activity all assume that basic delayed mode Tube and Module data processing and general delayed mode QC, paragraph 3.3, steps has been completed. This includes running the "flag data" program as many times as needed to handle data processing errors and creating the temperature plots discussed in these procedures.

- These sensors usually fail completely if they fail. As usual, look for gaps, ending early, etc. Occasionally you will get spurious high values, or non-zero nighttime values. Looking at the standard deviations of shortwave radiation (the second column in RAD###X.FLG) can often help find problematic periods if something has gone wrong. You can load the file into MATLAB with [x,y,h]=a2load('/home/disk_tao1/atlasrt/atlas/pm###/ram/rad###X.flg'); and the see when the standard deviation is greater than 200 with newatlas(x(find(y:,2)>200)). If you are processing a more recent tube RAD###X.RAM will contain a column, N, of the number of PIC board samples collected by the tube during the last sampling period is stored. Nominally 120 (1 Hz for 120 sec), during ARGOS transmit window's, the number will be around 107-108. If the tube couldn't communicate with the PIC board, the number will be zero.

3. Documentation

3.1 Automated Documents

Status.txt: Currently, several programs such as a2k_recover, a2k_deploy, a2k_swap, and a2k_ship, input text statements automatically into this file. The information can be viewed from the monitor-static webpage by selecting the file status.txt.

3.2 Manual Documents

Status.txt: To input manual entries, enter the command system by right clicking the mouse and selecting the command window. Type in "cd /taodata/realtime" and press enter. Type in "vi status.txt" and press enter. Select the "I" key on the keyboard then use the keyboard arrows to move the prompt to the point you want to enter an entry or change an existing entry. Type in the entry and then press the "Esc" key. Press "shift" and the ":" key. At the colon prompt type "wq" and press enter. This completes the process. Exit the command system window.

4. Non-QC/Analysis Procedures

4.1 Data Archive

4.1.1 Sitedb Directory: The history of all measurement information for each TAO site is automatically maintained by daily cronjobs within the sitedb file directory. The files are broken out by site ID directories.

4.2 Account Coordination

4.2.1 Service Argos Account: The information from the buoys is transmitted using satellite transmitters that connect to the service Argos satellite network. The account has been pre-established and should not require any renewal.

Appendix A

A.1 List of Figures

A.2 List of Tables

Appendix B

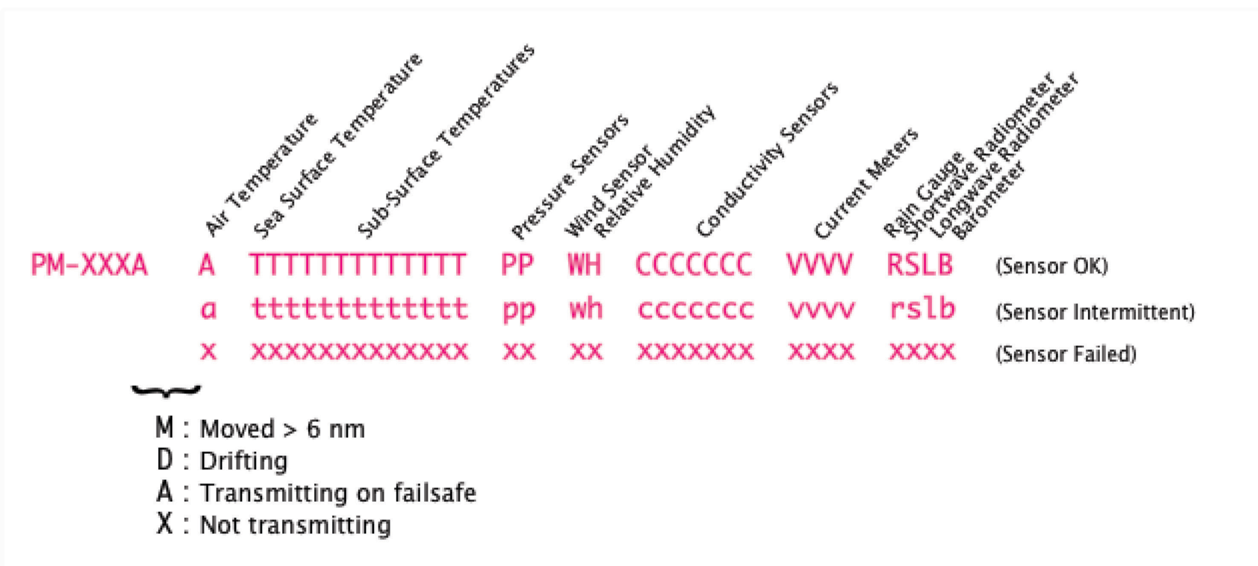
B-1 Data Quality/Position Codes Table

Indices Number	Rating	Description
0	**No Sensor or Datum Missing**	Used when there is no sensor measurement data or missing data for a particular sensor and it is not known if it is an intermittent issue or permanent.
1	**Highest Quality**	Pre/post-deployment calibrations agree to within sensor specifications. In most cases, only pre-deployment calibrations have been applied.
2	**Default Quality**	Pre-deployment calibrations only or post-deployment calibrations only applied. Default value for sensors presently deployed and for sensors which were not recovered or not able to be calibrated when recovered, or for which pre-deployment calibrations have been determined to be invalid.
3	**Adjusted Data**	Pre/post calibrations differ, or original data do not agree with other data sources (e.g., other in situ data or climatology), or original data are noisy. Data have been adjusted in an attempt to reduce the error.
4	**Lowest Quality**	Pre/post calibrations differ, or data do not agree with other data sources (e.g., other in situ data or climatology), or data are noisy. Data could not be confidently adjusted to correct for error.
5	**Sensor or Tube Failed**	Used when there is known tube or sensor failure that is preventing measurement information from being collected.
8	**Buoy Moved**	Used when the buoy has moved significantly away from it's originally deployed location.
9	**High Buoy Drift Speed**	Used when the buoy daily drift speed exceeds .1 knots (.05 m/s).

1 B-2 Data Mode Codes Table

Code No.	Description
0	No Sensor, No Data
1	Real-time (Tele-metered Mode)
2	Derived from Real-time
3	Temporarily Interpolated from Real Time
5	Recovered from Instrument RAM (Delayed-Mode)
6	Derived from RAM
7	Temporarily Interpolated from RAM

2 B-3 Status Key



3

1 B-4 Data Flag Values Table

Value	Description
1E30	Threshold test value-anything greater than this is flagged
1E33	Flagged Data (Manual)
1E34	Value computes out of range (automatic)
1E35	No data / Missing / Initialize all output
1E36	Insufficient samples for averaging (only for averaged data)

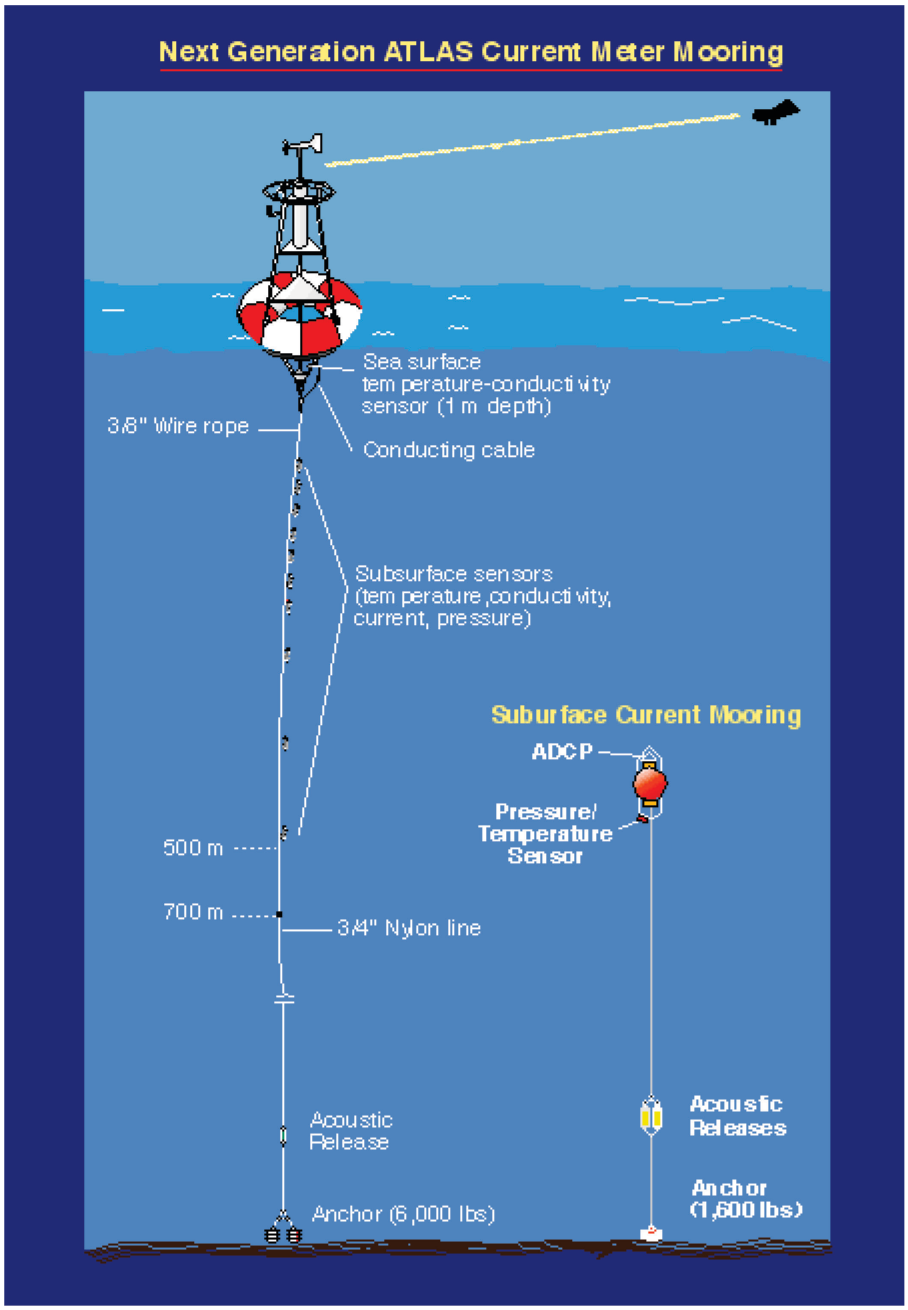
1 B-5 Data Origin (Sensor ID) Codes

ID No.	Description
0	No Sensor
1	Temperature (Cable)
2	Pressure
3	Wind (RMY)
4	Conductivity (FSI)
5	Self-Calibrating Temperature
6	Air Temperature
7	Sea Surface Temperature
8	Relative Humidity
9	Pressure/Temperature
10	Radiation
12	Rain
13	(9 bit NOPP Wind)
14	Next Generation Conductivity
15	Next Generation Temperature
19	Next Generation Pressure
22	Rain (PIC board update)
24	Next Generation Conductivity (Module software version 5.03+)
33	Gill Windsonic anemometer
40	Sontek Argonaut Current Meter

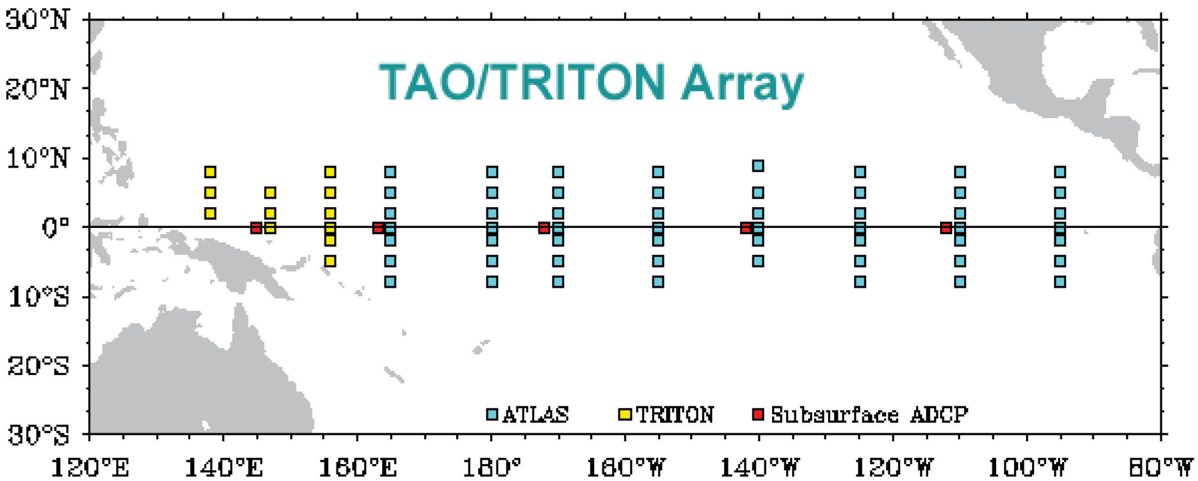
ID No.	Description
50	Longwave Radiation
51	Longwave Radiation Thermistor
60	Barometric Pressure
69	Means of conductivity from more than one sensor type
70	Seacat Conductivity
71	Microcat Conductivity
72	Seacat Temperature
73	Microcat Temperature
74	Seacat Pressure
75	Microcat Temperature
93	Handar Wind
99	Unknown

Appendix C

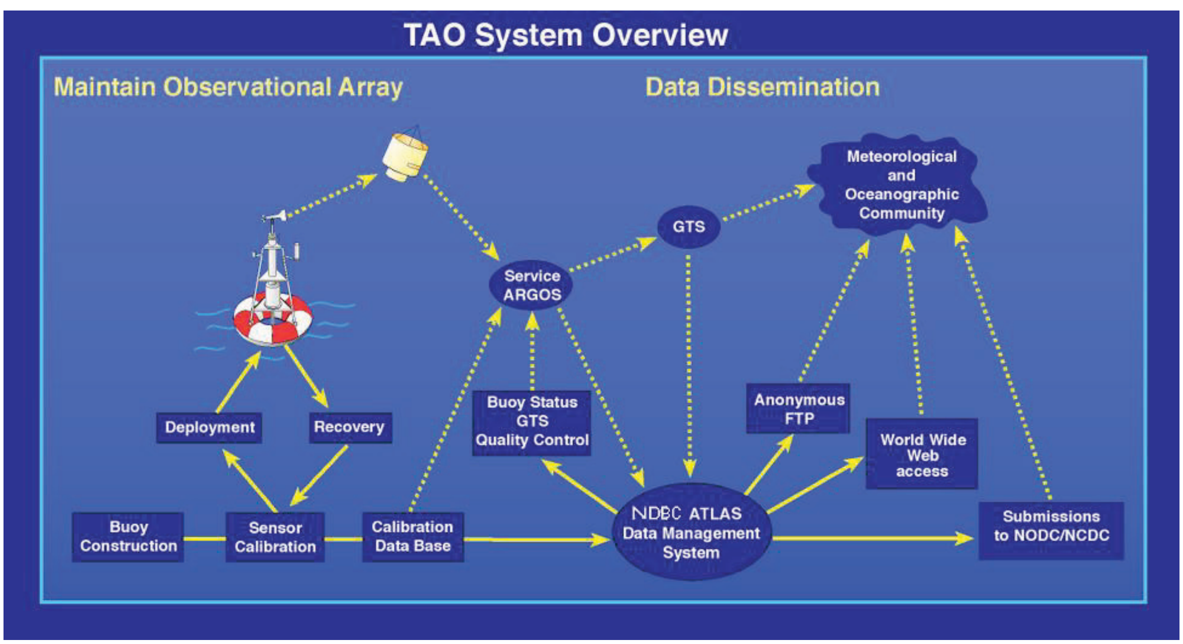
C-1 ATLAS Mooring



C-2 TAO/TRITON Array



C-3 TAO System Overview



Appendix D

D-1 Calibration File Description

Calibration File Description

Structure and Format of Calibration Files, *.cal

Version of 07/04/05

Calibration files contain 6 header lines and a variable number of sensor calibration lines (maximum of 50). Maximum record length of a line is 150 characters. The type of mooring and its type of sensor calibration lines is determined by the value of field 2 on line 6. The mooring type is either 1 or 2; see following documentation for differences in reading the resultant sensor calibration lines. The total number of sensor calibration lines beginning with line 7 are given in line 6, first field (maximum of 50). Note that not every calibration file contains information in all of the fields described below, so any program reading the cal files needs to handle missing fields.

The information in this document has been deduced from example calibration files, from the routines atlascal.c and atlascal.h, and from the document "Atlas Sensor Data and Calibration.htm" (located on the I drive.) For more details on how the conversions into engineering values are done, see the latter document.

Note that the first character of a line may be a blank, a "+", a "-", or an "@". The programs that use the calibration file must interpret this character and use it properly. Blank means standard processing. "+" means intermittent data from the sensor INTERMITTANT_DATA_FLAG). "-" means sensor is bad and generally software uses it to turn off near real time processing of that sensor (FAILED_SENSOR_FLAG). "@" means the sensor is not to be processed in near real time (DELAYED_MODE_FLAG). Supposedly the symbol # stands for HEADER_IDENTIFIER, but this has not been observed in use.

VERSION 1: MOORING TYPE 1 ("NXGEN") (1 IN LINE 6, FIELD 2)

```
Lines 1 -6          Calibration header lines
Lines 7 - end   Sensor information and calibration coefficients (maximum number allowed is 50)
```

<u>Line 1</u>

Examples

```
 6882   38 980260000 980321740                                   WMO
NO. 13010
  6882   38 980260000 980321740 000000000 000000000     0000000  WMO
 NO. 13010
```

columns	Contents
1-5	ptt id number (%5d)
6-10	buoy(mooring) id number (%5d)
7-20	ptt assign time as YYDDDHHMM (%10c)
21-30	deploy time as YYDDDHHMM (%10c)
31-40	recover time as YYDDDHHMM (%10c)
41-50	ptt deassign time as YYDDDHHMM (%10c)
55-62	ptt clock correction as stttttt where s is a sign (%c%7c)
72-77	wmo number

Line 2

Examples

```
-DEPLOYED AT  0  0.8N, 110  2.6W ON GP1-98-KA  s.sss  dddd
98112
-DEPLOYED AT 18 05.9N, 115 35.8E ON WP2-97-OR
98065 98065
```

columns	Contents
1	drift flag (blank or -)
23-33	deployment position (character string)
37-45	deploy_cruise_id (character string)
48-52	mooring_scope (%5.3f)
55-58	mooring_depth (%4d)
70-74	mooring_drift_start_time (YYDDD)
76-80	mooring_out_of_range_time (YYDDD)

Line 3

Examples

```
-TUBE 413T , SB   6, IO   6, WND   6,MB    0,CPU    0,RN   0,RD  36
98224
-TUBE 147P , SB   0, IO   0, WND   0
98201 98206
```

columns	Contents
1	tube flag character (blank or -)
7-9	tube serial number (%3d)
10	tube type character (%c) (valid are N, T, G, S)
16-19	sb board id
24-28	io board id
33-36	wind board id

If the tube type character is N, T, G, S the next 4 ids may be present:

37-44	mother board id
45-53	cpu id
54-60	rain board id
61-67	radiation board id
70-74	buoy-on failsafe time (%5c)
76-80	end transmit time (%5c)

Line 4

Examples

```
 AIR   218, SST11576, WIND 22535, CABLE 041, RAIN 681, EPPLEY 32173
 AIR 25570, SST 8402, WIND 11674, CABLE 141
```

columns	Contents
5-11	air sensor serial number (%6d – ignores the trailing ,)
15-21	sst sensor serial number (%*c%5d)(ignores the trailing ,)
27-32	wind sensor serial number (%6d)
40-44	cable serial number (%4d)
50-53	rain sensor serial number (%4d)
62-67	radiation sensor serial number (%6d)

Line 5

Examples

```
 TUBE SOFTWARE VER 2.16, SENSOR VER 0.00
SM:2M232/SM:5A464.
```

```
 TUBE SOFTWARE VER 2.16, SENSOR VER 0.00                        SITE
ID: 15N118ESM:2M232/SM:5A464.

  TUBE SOFTWARE VER 4.07, SENSOR VER 0.00                        SITE
 ID: 2N95W
```

columns	Contents
19-23	tube software (firmware) version (%5f)
36-40	sensor version (%5f)
	(pressure sensor id)
73-80	site id (%8c)

The use of the rest of the line (shaded) is not clear; however, it is likely that this information applies only to older moorings. The code in atlascal.c reads

```
		if (buoy->which_atlas > IS_STD)
			sscanf( lineread, " TUBE SOFTWARE VER%5f, SENSOR VER%5f",&(buoy->tube_firmware_vers), &(buoy->sensor_vers));
		else
			strncpy( buoy->press_sensor_id, lineread, 40 );
			if ( latlon2site(buoy->deploy_long, buoy->deploy_lat, tempstring3, &wmocheck) )
			return 1;

		if ( chptr = strstr(lineread, "SITE ID:") )
		{
			sscanf( chptr+8, "%8c", tempstring1 );
			for ( i = 0, j = 0; i < 8; i++ )
			{
				if ( (*(tempstring1+i) > 0x2F) && (*(tempstring1+i) < 0x5B) )
					*(tempstring2+j++) = *(tempstring1+i);
			}
			strcpy( buoy->site_id, tempstring2 );
			if ( 0 != strcmp(tempstring3, tempstring2) )
				fprintf( stderr,
				"***\n*** Warning: Site ID in Calibration File does not match database table TAO_cal.Site\n***\n" );
		}
		else
			strcpy( buoy->site_id, tempstring3 );
			if ( (buoy->deploy_time != MISSING_TIME_VALUE) && (wmocheck != buoy->wmo_number) )
			fprintf( stderr,
			"***\n*** Warning: WMO number in TAO_cal.Site does not match Calibration File\n***\n" );
```

Line 6

Example

```
   18          1
```

columns	Contents

1-5 number total sensors (%5d) Number of sensor calibration lines following (max of 50)

6-10 mooring type (%10d) – allowed values 1 and 2. If type 2 mooring (“NEW NXGEN”), read an additional field from the end of the sensor records to specify the buffer and location within the buffer for that sensor data.)

One line for each of the number of total sensors follows as lines 7 to end.

Structure of each sensor line is

Field 1 flag value: blank means OK, + means intermittent and – means bad and the telemetry software will not report the data. @ appears to mean do not process in near real time, only delayed mode

Field 2 sensor type id

Field 3 sensor serial number

Field 4 sensor type or sensor depth

Field 5 coef a

Field 6 coef b

Field 7 coef c

Field 8 coef d (not always used)

Field 9 coef e (not always used)

Field 10 buffermap – only for Mooring Type 2

Field 11 or 10 flagdate1

Field 12 or 11 flagdate2

Field 13 or 12 flagdate3

Field 14 or 13 flagdate4

Example: Type 1 Mooring (“NEXT GENERATION”)

```
Line #   Data Fields
1     770  481 042720000 043120043
WMO NO. 32320
2   -DEPLOYED AT  1 59.7N,  95 18.1W ON GP6-04-RB  0.986  3118
05012
3    TUBE 639T , SB4041, IO5206, WND1095,MB    0,CPU    0,RN   0,RD
4    AIR 40323, SST11743, WIND 38172, CABLE 481, RAIN 1049,
5    TUBE SOFTWARE VER 4.07, SENSOR VER 0.00
SITE ID: 2N95W
6      18          1
7   - 640323 AIR -2.8744390E-01  1.0076860E+02  0.0000000E+00 0.00005
0.020571 05012
8    1511743 SST  1.0102380E-03  5.5933720E-04  1.7053430E-06  19.71
9    1511996T 20  1.0105850E-03  5.5988830E-04  1.7434660E-06  19.62
10   1512047T 40  1.0074740E-03  5.5992800E-04  1.7295580E-06  19.73
11   1512049T 60  1.0145310E-03  5.5745330E-04  1.7647320E-06  19.75
12   1512048T 80  1.0135970E-03  5.5796680E-04  1.7312330E-06  19.84
13   1512011T100  1.0115070E-03  5.5963270E-04  1.7138630E-06  19.75
14   1512013T120  1.0129030E-03  5.5855580E-04  1.7475220E-06  19.70
15   1512017T140  1.0125160E-03  5.5823220E-04  1.7541020E-06  19.65
16   1512025T180  1.0161300E-03  5.5792620E-04  1.7278090E-06  19.76
17   1512530T300  1.0168650E-03  5.5864860E-04  1.7397160E-06  19.73
18   1512679T500  1.0139460E-03  5.5822220E-04  1.7217310E-06  19.74
19   1993989P300 -1.4252880E+02  2.8037770E-02  1.7864870E-02
20   1997326P500 -1.5257790E+02  2.7286690E-02  7.0119840E-03
21   - 338172WIND  1.0000000E+00  7.0000000E+00  0.0000000E+00 -.13166
.1953154 04318
22   - 840323  RH  9.2915510E-02  1.0031670E+00  0.0000000E+00 -0.0055
0.994190 05012
23   24 1551C  1  1.3970610E+00 -1.1716660E-02  8.9807230E-04 -10.072
24   22 1049RAIN -3.8259000E+02  1.1578690E+12  0.0000000E+00
```

Sensor type for the sensor calibration record is determined from the sensor type id. Sensor types and example calibration file records as of the preparation of this document are:

<u>Sensor Type ID 6: Air Temperature (AT,at, AIR)</u>

Example

```
        ID             A[0]            A[1]            A[2]
A[3]           A[4]
     6 58358 AIR -0.3097342E+00  0.1019228E+03  0.0000000E+00 -
0.2515173E-03  0.2060663E-01
```

<u>Sensor Type ID 8: Relative Humidity (RH/rh)</u>

Example

```
      ID             A[0]            A[1]            A[2]            A[3]
A[4]
  8 58358  RH -0.4318653E+01  0.1081028E+01  0.0000000E+00
0.2635982E-02  0.9945107E+00
```

<u>Sensor Type ID 15: Seawater Temperature (SST, sst, Tn)</u>

Some type of calibration temperature is sometimes observed in Field 7 (see highlighted below). Takes up a 16 character field. It has been observed used in the "a2k" module. Exactly what it is is not clear. It is also not clear if possibly a second 16 character field may also be used at times in the conversion to temperature, although no T calibration record with anything in that field has yet been observed.)

Example

```
     ID             A[0]            A[1]            A[2]         TC
  15 12852 T300  0.1022229E-02  0.5562479E-03  0.1784249E-05  19.71
```

<u>Sensor Type ID 19: Pressure (Pn)</u>

Example

```
      ID             A[0]            A[1]            A[2]
 19 93932 P300 -0.1446640E+03  0.2817997E-01  0.7349011E-01
```

<u>Sensor Type ID 14/24: Seawater Conductivity (Cn)</u>

A temperature value must accompany a conductivity value

Example

```
     ID             A[0]            A[1]            A[2]
 15 12235 T 20  1.0118000E-03  5.5828030E-04  1.7345480E-06
 24  1031 C 20  1.6058240E+00 -5.1618390E-03  5.8238900E-04 -
1.0537530E+01
```

Sensor Type ID 3: Wind Speed and Direction (UV, uv), RM Young (mechanical) Wind Sensor
Sensor Type ID 33: Wind Speed and Direction (UV, uv), Gill Sonic Wind Sensor
Sensor Type ID 93: Wind Speed and Direction (UV, uv), Vaisala Sonic Wind Sensor

Example

```
    ID              A[0]            A[1]            A[2]            A[3]
A[4]
  3 16363 WIND  1.0000000E+00  6.0000000E+00  0.0000000E+00
9.0389900E-02  1.9556140E-01
```

Sensor Type ID 12/22: Rainfall (RNA RNB RNC RAIN)

Example

```
     ID             A[0]            A[1]            A[2]
 22  682 RAIN -4.0857000E+02  1.2008800E+12  0.0000000E+00
```

Sensor Type ID 10: Shortwave Radiation (RAD RADA RADB RADC)

Example

```
       ID             A[0]            A[1]            A[2]
 10 32178 RAD  1.0000000E+00  0.0000000E+00  4.3100000E-01
```

Sensor Type ID 50/51: Longwave Radiation (LWR LTD LTC LWRA LWRB LWRC)

Example

```
      ID              A[0]            A[1]            A[2]            A[3]
A[4]
 50 32219 LWR  4.1200000E+00  3.5000000E+00  0.0000000E+00
2.1220990E+01  4.2561960E-01
 51 32219 LTD  1.0287420E-03  5.5083304E-04  1.9063668E-06 -
4.9982970E+04  2.5009370E+08
 51 32219 LTC  1.0287420E-03  5.5083304E-04  1.9063668E-06 -
4.9990990E+04  2.5012450E+08
```

Sensor Type ID 40: Current Velocity (UnVn WnNn)

Example

```
     ID               A[0]            A[1]            A[2]
 40  196 V 10  2.0000000E+00  6.0000000E+00  1.0000000E+00
```

Sensor Type ID 60: Barometric Pressure (BARO, BP, bp)

Example

```
      ID              A[0]            A[1]            A[2]
 60 81192 BARO  1.0000000E+00  0.0000000E+00  0.0000000E+00
```

TYPE 2 MOORING ("NEW NEXT GENERATION") (2 IN LINE 6, FIELD 2)

If the mooring is "new Nxtgen type 2" (line 6, field 2 in header), an additional field is read from the end of each sensor calibration record that specifies the buffer and location within buffer for that sensor data. This apparently has been added to allow more flexibility in the order in which sensor lines are put into the file (for example, not requiring lines to be in order of increasing depth). Otherwise the contents and formats of the lines are the same as for Type 1 moorings.

Example file

<u>Line #</u>   <u>Data Fields</u>

```
1       994  508 050800000 051050240
WMO NO. 32316
2      DEPLOYED AT  2 02.9N, 110 02.1W ON GP2-05-KA  0.984  3768
3      TUBE 625S , SB4027, IO5133, WND1061,MB    0,CPU    0,RN   0,RD
4      AIR 59312, SST12516, WIND 55616, CABLE 508,
5      TUBE SOFTWARE VER 4.07, SENSOR VER 0.00
SITE ID: 2N110W
6        17         2
7       659312 AIR -1.8136520E-01  1.0134560E+02  0.0000000E+00 -
3.0866050E-04  2.0582970E-02  002
8      1512516 SST  1.0089040E-03  5.5930580E-04  1.7271290E-06
19.81                     004
9      1512810T 20  1.0055850E-03  5.6172770E-04  1.6632900E-06
19.73                     006
10     1512813T 40  1.0214050E-03  5.5737900E-04  1.7532030E-06
19.70                     008
11     1512814T 60  1.0150730E-03  5.5990830E-04  1.6982810E-06
19.69                     010
12     1512783T 80  1.0103710E-03  5.6058820E-04  1.7003550E-06
19.68                     012
13     1512785T100  1.0160010E-03  5.5663930E-04  1.7393220E-06
19.79                     014
14     1512786T120  1.0067690E-03  5.6190110E-04  1.6740600E-06
19.75                     016
15     1512788T140  1.0197220E-03  5.5607390E-04  1.7568160E-06
19.78                     018
16     1512789T180  1.0136370E-03  5.5800980E-04  1.7272330E-06
19.85                     020
```

```
17     1512997T300  1.0034980E-03  5.5906670E-04  1.6993910E-06
19.85                022
18     1512998T500  1.0101380E-03  5.5858880E-04  1.7168270E-06
19.67                024
19     1997348P300 -1.5749790E+02  2.6810740E-02  2.2323370E-01
026
20     1993988P500 -1.3879050E+02  2.7169930E-02  4.0132360E-02
028
21      355616WIND  1.0000000E+00  9.0000000E+00  0.0000000E+00
1.1024150E-01  1.9338820E-01  034
22      859312  RH -6.5182190E+00  1.1130730E+00  0.0000000E+00
9.4332720E-04  1.0001070E+00  042
23     6083480BARO  1.0000000E+00  0.0000000E+00  0.0000000E+00
092
```

Lines 1 – 6 are the same as Type 1 Moorings

Lines 17 – end (maximum of 50 sensors) The subsequent sensor calibration lines are the same as for Type 1 Moorings except for the addition of the 4 character-wide field at the end specifying the "buffermap" that gives the buffer and location within the buffer for that sensor data.

Some Miscellaneous and Diagnostic Data

This information was described in the document "Atlas Sensor Data and Calibration.htm". It implied this information was obtainable from a *.cal file, but I don't understand how. Or even if this information REALLY is stored in a *.cal file.

co/va

instantaneous compass value (high order 8 bits)
instaneous wind vane value (low order 8 bits)

Evaluating a time series of compass and vane values can indicate abnormalities (constant or 0 values) that are not readily apparent viewing wind vector or direction values.
When combined with the "spba" data word (see below), wind speed,direction, and vector components U-V can be computed for the last instantaneous sample taken.

- ***Conversion***
- d1 = mod((N1+N2),256)
- direction = 180 + (A[1] + A[2] * (d1 / 255.0 * 360.0))
- **<u>Example</u>**

- Raw Data: 2A61 (N1 = 42, N2 = 97 decimal)
-
- Calibration Record:
- ID A[0] A[1] A[2] A[3] A[4]
- 3 16363 WIND 0.0000000E+00 6.0000000E+00 1.0000000E+00 9.0389900E-02 1.9556140E-01
-
- Raw Compass = N1 = 42
- Raw Vane = N2 = 97
-
- d1 = mod((42+97), 256)
- = 139
-
- direction(degT) = 180 + (6.0000000E+00 + 1.0000000E+00 * (139 / 255.0 * 360.0))
- = 22.2

sp/ba

instantaneous wind speed (high order 8 bits)
logic battery voltage (low order 8 bits)

- ***Conversion***
- Speed(m/s) = A[3] + A[4] * N1
- BatteryVolts = N2 * (21.25/256.0)
- **Example**
- Raw Data: 2099 (N1 = 32, N2 = 153 decimal)
-
- Calibration Record:
- ID A[0] A[1] A[2] A[3] A[4]
- 3 16363 WIND 0.0000000E+00 6.0000000E+00 1.0000000E+00 9.0389900E-02 1.9556140E-01
-

engr

count of tube resets since powered on (high order 8 bits)
compass correction table flag (low order 8 bits)

If the low order 8-bit word is 80 hexadecimal (128 decimal), the onboard programming for calibration of compass output has been lost. This condition results in reduced quality of wind data averaged by the onboard realtime system.

The high order 8 bit-value is a cummulative counter for tube resets.

lp/xb

inductive loop impedance (high order 8 bits)
transmit battery voltage (low order 8 bits)

- ***Conversion***
- LoopImpedance = 1700/N1 - 15
- BatteryVolts = N2 * (15.25/256.0)
- **Example**
- Raw Data: 4EAA (N1 = 78, N2 = 170 decimal)
-
- loop impedance = 1700 / 78 - 15
- = 6.8
-
- transmit battery voltage = 170 * (15.25/256.0)
- = 10.1

MF

mode flag

The mode flag word has a value of 0 or 1. Mode 0 is normal sampling mode for deployed buoys. Mode 1 indicates fast-sampling test mode.

SWRn

number of SWR samples in daily average

D-2 Rain Rate Processing

"RAIN RATE PROCESSING"

This documentation was extracted verbatim from PMEL file "rainspikes.html."

Use "process_rainspikes" after initially spawning the rain processing via processA2Tube. The first (truncated) rain~.flg file is used as the initial input by the script, which successively runs and reruns remove_rainspikes_ram, flag_data, rain_desiphon, remove_rainspikes_acm, and interpolate_rainspikes_flg to produce numerous flag~.txt and rain~.flg files. It outputs a cleaned rain~.flg file that may be subsequently examined and re-flagged by hand, after which rain_desiphon, rain_smooth and rain_rate need to be rerun to create the final cleaned .rat file.

In order to avoid the lengthy process of adding new flags to the flag~.txt file and re-running flag_data on both the new and old flags multiple times, process_rainspikes copies each new rain~.flg file to rain~.ram and then flag_data is run using a flag~.txt file containing just the new flags. The copying is accomplished using rain_copyflg2ram, a perl script that insures the quality and source codes in the .flg file are removed from the header and data as it is copied to a .ram file.

To prevent siphons from being flagged and therefore missed by the subsequent running of rain_desiphon, a rain~.log file is read in and siphons are tracked by the remove_rainspikes_acm program, the central program run by the script. This insures that the points immediately surrounding each siphon will not be flagged. As each iteration of the flagging process re-runs rain_desiphon, which produces a new log file, the remainder of the iteration then uses this new log file for locating siphons. The first log file used is the one created by remove_rainspikes_ram; this is used instead of the original log file because its siphons are more likely to be valid due to the prior removal of all spikes above 600ml.

Copies of the original rain~.ram, rain~.flg and rain~.log files are automatically created at the beginning of the script, since these files are overwritten numerous times during the course of the run. The original files have an "_orig" extension added to their filenames.

To run the script, type `process_rainspikes`, then enter inputs to the following prompts. Examples and recommended values appear in brackets:

- This script assumes that the rain file to be cleaned is located in a subdirectory of /home/drumlin/data/nxram/. Please enter the deployment number and letter for this subdirectory and rain file: [e.g. 250a]
- Please enter the desired number of rain-cleaning iterations [3]:
- Please enter the threshold (in ml) below which rain volume differences will be ignored [10]:

- Please enter the number of points (minutes) for averaging to the left [16]:
- Please enter the number of points (minutes) for averaging to the right [16]:
- Please enter the threshold (in ml) for rain-spike warnings sent out after the data have been cleaned [540]:

After process_rainspikes has finished running, you can check rainspikes.log (in the working directory) to find out whether any errors occurred during processing. The most current rain data will be contained in the rain~.flg file, while a listing of all the flags used in the successive iterations can be found in flag~.txt_all.

Following are the post-processing steps to take:

1) As you plot and examine rain~.flg, add any new flags to the existing flag~.txt file, which is a copy of the final flag~.txt_interpolated flag list. Check and manually flag any remaining anomalies in the data, including any events & siphons that may not have been captured, or any spikes that may have been smoothed and captured as siphons.

While adding any new flag, be careful to check it against any conflicting flags that may already be listed in the existing file. For example, if there is already an 'i' flag to interpolate from 2002161103532 - 2002161105632, and you want to add an 'f' flag to delete 2002161103332 - 2002161104032, flag_data will produce an error because it will sort the starting dates and perform your flag first, then hiccup when it tries to interpolate from a 1e33 value. It is a good idea to manually check any spike that reaches close to the 600ml upper limit, since at times the processing program can 'catch' a segment of noise by interpolating between enough reasonable-looking values to generate an event that appears real; an unusually high 'event' is one indicator that this may have occurred. Another, less likely but equally serious problem to look for is the possibility of a missed event. Occasionally an event profile falls off gradually enough (<350ml drop) that rain_desiphon cannot detect it and so it isn't included in the rain~.log file. As a result, process_rainspikes doesn't know about the siphon and sees it as a large drop, which it then flags. Usually the length of real events prevents the entire event from being obliterated by the processing, so signs to look for are smaller than normal peaks, and shallower than normal drops. If you do run into such a problem, the data need to be reinstated from the original ram file; copy and paste the event from rain~.ram_orig into the latest rain~.ram file before flagging, and make sure that the flag listed in flag~.txt_all is erased (it is also a good idea to include a comment about the removed flag, either in the flag~.txt or the pm~.sum file). Always compare questionable data points in the rain~.flg file with the original data in rain~.ram_orig.

2) Run flag_data on the latest rain~.ram file, using the flag list in flag~.txt. (Note that the latest .ram file is a copy of the latest rain~.flg_despiked# file, containing cleaned data created prior to the interpolation that produced the latest rain~.flg file. This is important because the user may need to edit or remove interpolation flags manually before the final flag_data is run; at this point, the interpolation is re-run along with any new user-set flags.)

3) After this final flagging, re-run the desiphon, smooth, rain rate and davg programs.

4) Append the new manual flags in flag~.txt to the full flag list: flag~.txt_all. Rename the full list to flag~.txt, and delete all of the other (partial) flag~.txt* files (there will be one flag~.txt_despiked# file for each iteration that occurred).

Following is more background information about the programs included in the process_rainspikes script...

Rain-Spike Removal Programs:

remove_rainspikes_ram:

This reads a noisy rain~.ram file and generates lists of flags to be added to flag~.txt for subsequent running of the flag_data program. The flags take into account only values exceeding a 600ml threshold. This is used to initially weed out rain spikes prior to running remove_rainspikes_acm; large stretches of spikes in particular would otherwise cause the averaging calculations in remove_rainspikes_acm to run amok.

remove_rainspikes_acm:

This program reads a noisy rain~.acm file and generates lists of flags to be added to flag~.txt for subsequent running of the flag_data program. Inputs include a flagging threshold and two 'number of included points' values; one for the average calculated to the left of each point, and one for the median calculated to the right of each point. A data point is flagged (output to the flag~.txt file) if it jumps more than the user-set flagging threshold from the previous point without being a part of natural accumulation or of a siphon event. Subsequent points are also flagged if they meet various criteria (e.g., if they continue in the same direction, if they are farther than the threshold from an average of non-flagged points taken before the jump, or if they exceed the median of points following the jump). Flagging after a 'square wave' drop in the data is automatically truncated after 11 points, to prevent run-away flagging. The 11-point flag enables the square drop to be ignored by the subsequent smoothing program, so it won't appear as a negative rate.

Following is a layout of the program's logic as it differences consecutive points (skipping any invalid 1e3* values inbetween):

1) A 'jump' (or drop) is found if the difference between consecutive valid points is larger than the user-entered threshold.

2) After a jump is output to a flag~.txt file, it is cleared; only one jump (or a summation of multiple jumps) is stored in memory at any time. A limited history recording jump versus non-jump points is maintained in order to provide a reasonable left average (needed for the logic

described below), and also to check for permanent (square-wave) drops that would otherwise cause run-away flagging. The history is currently set at 5 (with each point an average of the user-set number of points [16] chosen for the left average). Three arrays take care of this: xavg stores the valid values on the left, avg stores each average of the previous [16] points, and xreal keeps track of the jumps that are real versus the ones that are summed across several points (so that after-jump values can be tracked and checked for square waves). Note that the 11-point flagging used to erase square-wave drops can be subsequently undone if a user-chosen "siphonthresh" value is set too low while running interpolate_rainspikes_flg; this will cause the drop to be filled and thus subsequently seen as a negative rate.

3) If there was a jump (or drop), then:

- If the one before was also a jump, in the same direction, then add the new one to the stored jump.
- elseIf the one before was a jump in the other direction, check the average of points to the left:
    - If there was a drop followed by a jump, flag the drop & store the jump
    - elseIf there was a jump followed by a drop, look at the median of points to the right:
        - If the median is invalid due to bad data, flag both the jump and drop (conservative).
        - elseIf the right median exceeded the left average (accumulation), then:
            - If the jump fell below the right median, keep the jump, flag the drop.
            - elseIf the drop fell between the left average & right median, keep the drop & flag the jump. (Dissect the jump to exclude points that fell below the right median).
        - elseIf there was no accumulation (to within 1 threshold), then keep the drop & flag the jump.
        - elseIf the accumulation was large and negative, warn user of potential missed siphon.
        - elseIf the accumulation was small and negative, store the jump for a square_wave_check (see below *).
        - else flag both the jump and drop.
    - elseIf there was a drop followed by a 'drop' (from the left average), then flag both drops.
    - else There was a jump followed by a drop back to the left average (to within 1 threshold):
        - If the median is invalid due to bad data, flag the jump (conservative).
        - elseIf there was accumulation, keep the jump, flag the drop.
        - elseIf there was no accumulation (or the jump exceeded any accumulation), keep drop & flag jump. (Dissect the jump to exclude points that fell below the right median).

- else The one before was not a jump, so this is the first point in a jump: store the jump.

4) elseIf there was no jump, but the previous point had jumped, then:

- If the non-jump continued in the same direction as the jump, store it too.
- elseIf the non-jump moved in a new direction, look at the left average:
    - If both were 'drops' (>threshold) below the left average, then:
        - If the non-jump dropped further, flag both the jump and non-jump.
        - elseIf the non-jump fell close to the jump, store the drop in case it continues.
    - If both were 'jumps' (>threshold) above the left average, then add the non-jump to the stored jump.
    - elseIf the non-jump was close to (within a threshold of) the left average, check the right median:
        - If the median was invalid due to bad data, flag the jump (conservative).
        - elseIf there was accumulation, keep the jump, flag the drop.
        - elseIf there was no accumulation (or the jump exceeded any accumulation), keep the drop & flag the jump. (Dissect the jump to exclude points that fell below the right median).
- * Check: In case the previous point wasn't a real jump (but a summed one spanning many points), do a square_wave_check: If a jump was followed by 11 non-jump pts, flag it and clear it to avoid runaway flagging. If there was accumulation, simply clear the jump (don't flag it).

5) else There was no jump, nor was there a jump at the previous point. Do a square_wave_check (as described above *). This will insure that any jump currently stored and expanding via the logic stated above will not result in run-away flagging.

interpolate_rainspikes_flg:

This is run last, in order to fill sections of flagged data that might otherwise result in rain accumulation and rates being missed by the rain_desiphon and rain_rate programs. The program first seeks the first unflagged value in the rain~.flg file to use as a starting value (in case subsequent points are flagged), then searches for unflagged values throughout the rest of the file, storing their locations and noting (flagged) gaps between them. If a gap spans more than one point, then the number of flags in the previous 16 points and the number of flags (if any) between the current value and the next good value to the right are both stored, and the following logic checks are performed:

1) If the rain accumulation between the point before and the point following the gap is >=5ml, then:

- If there are more than 7 flags in the previous 16-point interval, interpolation is applied to the gap.
- elseIf there are fewer than 7 flags on the left, but more than 7 flags on the right, and the accumulation between the current point and the point following the gap to the right is less than 5ml (i.e. the right-hand gap will not be filled), then interpolation is applied to the current gap. Otherwise, the flags to the right might cause the 10-minute data spanning this point (produced by the 16-point Hanning filter during rain_smooth) to be flagged, and the subsequent differencing in rain_rate will be unable to see the accumulation and capture a rate.

2) If the gap falls across a siphon:

- If the gap spans more than 7 points, the program fills it with the post-siphon value (the first point to the right of the gap) to insure that the siphon drop will be seen by subsequent running of rain_desiphon.
- elseIf there are fewer than 7 flags on the left, but there is a gap to the right that spans more than 7 points and whose accumulation is less than 5ml (so it won't be filled), then the current gap is filled with the post-siphon value to help prevent the siphon from being missed (although this may no longer be needed because of the below check #3).

3) If the gap falls in a post-siphon region, spanning at least the first 4 points after a siphon, then interpolation is applied to the gap to help prevent the siphon from being missed by rain_desiphon. If the gap spans a drop or zero accumulation, then instead of being interpolated, the gap is filled with the lower of the two values.

4) If the difference across the gap is a drop smaller in magnitude than the -50ml siphon threshold, and the gap itself spans fewer than 11 points (i.e. rain_smooth won't flag the gap and so the drop will be picked up by rain_rate as a large negative rate), then a warning is printed out to the user. No interpolation is performed, since this may be a real siphon event with an odd profile that was missed by rain_desiphon (isn't in the log file).