RTI ADCP/DVL

USER GUIDE

Table of Contents

Introduction 4
Chapter 2 System Overview 5
Chapter 3 Instrument Care 6
3.1 Guidelines to Instrument Care 6
Chapter 4 ADCP/DVL Power and Serial Communications 7
4.1 Deck Box Diagram 7
4.2 Pigtail Cable Configuration Diagram 9
4.3 End Cap Connector 10
4.4 ADCP/DVL Operations with Serial Communications 11
4.5 Self-Contained ADCP 12
Chapter 5 User Commands and Data Formats 14
5.1 User Commands (ASCII) 14
5.2 User Command Examples 24
5.3 NMEA Data 26
5.4 Binary Data 27
Chapter 6 System Checkout and Diagnostics 28
6.1 System Bench Test 28
6.2 Diagnostics 30
Appendix A 31
A1 Coordinate systems 31
A2 GPS Compass 33
Appendix B 34
B1 Payload Matrix Contents 34
B2 C# Binary Decoding Example 42
B2 C# Checksum Calculation 47
B3 C# Binary Data Packet 48
B4 System Serial Number and Subsystem Codes 49
B5 System Status Word 51
Appendix C 52
C1 DA 150 Outline Drawings 52
C2 DA 150 Close up Hardware 53

C3 DP 300/600 Outline Drawings 54
C4 DP 300/600 Close up Hardware 55
C5 Dual Frequency DP 300/1200 Outline Drawing 56
C6 300 KHZ SC 6000M ASSY DWG 57
C6 SC 600 COASTAL ASSY DWG 58
Appendix D 59
D1 SELF-CONTAINED ENDCAP WIRE DIAGRAM 59
D2 SELF-CONTAINED WIRE HARNESS DIAGRAM 60
D3 SELF-CONTAINED ENDCAP BATTERY ASSY 61

Introduction

Thank you for purchasing the Rowe Technologies Inc. (RTI) ADCP/DVL. This User Guide explains important features and considerations while owning and operating this instrument.

Please spend time in reading and noting the many aspects of this User Guide that relates to your applications.

The RTI ADCP/DVL is capable of DVL Bottom and Water track operations or it can be configured to current profile with or without bottom track enabled. Included in this instrument are a water temperature sensor and a minimum 2 gigabyte secure digital (SD) memory card. The installed compass/attitude sensor is used primarily for Earth referenced current profiling.

Chapter 2 System Overview

Chapter 3 Instrument Care

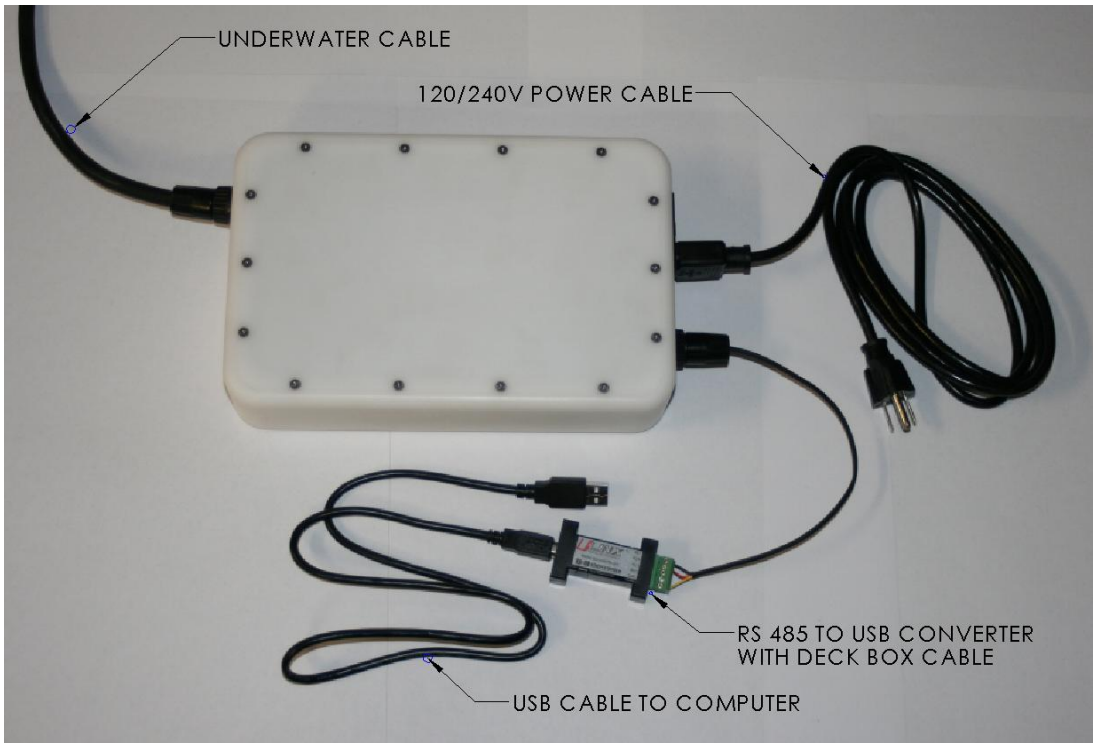
3.1 Guidelines to Instrument Care

Your RTI ADCP/DVL is a robust instrument designed for prolonged use. Please consider the following guidelines to prolong the life of your instrument, decrease the risk of damage and continue its factory tested performance:

1. Please do not open the instrument housing or the deck box enclosure unless you have contacted service at RTI. There is no regular field maintenance to perform on these units.
2. Handle with care; dropping the unit or severe impact could damage the transducer elements, the water tight integrity of the instrument housing and the internal electronics of the system.
3. Avoid leaving the instrument and the deck box in direct sun light for prolong periods of time. If they do need to be placed in direct sun light consider a cover. Try not to store or transport these items in extreme temperatures. When the instrument and deck box are not in use please place them back in the original shipping container.
4. Keep the instrument and the deck box clean and clear of dirt, oils and any chemicals. Dirt may contaminate the o-ring seals and the electrical connection of the underwater connector.
5. The “red” transducer face is made of a special urethane. This urethane has acoustic properties necessary for the instrument to perform. This urethane is also softer than the other exterior plastic that makes up the white housing and the black transducer mounting head. The transducer faces must be given extra care to avoid punctures, cuts, direct sun light, any chemical contact, discoloration from oils and contaminates in the water. The transducer face is soft enough to sustain impressions from anything left on it or if it is placed against or on top of anything.
6. Your instrument has an internal magnetic flux gate compass. The compass may not work correctly if the instrument is near a steel structure or magnetic field.
7. Please do not “ping” your instrument in air for any prolong amounts of time, this could cause damage. Please use a container of water for lab testing.

Chapter 4 ADCP/DVL Power and Serial Communications

4.1 Deck Box Diagram



When RS 485 is active a green indicator light located on the side of the deck box will flash.

The power receptacle has both an external switch to activate power and external access to 2 fuses. Please check the condition of the fuses first if the deck box does not power up. The power receptacle has a lighted switch to indicate power is on.

WIRE DIAGRAM FOR DECKBOX RS 232 / RS 485 CABLE ASSEMBLY

PIN #	SIGNAL	WIRE COLOR
1	RS 232 IN	BLUE
2	RS 232 OUT	WHITE
3	N/C	
4	DATA COM	BLACK
5	RS 485 (A-)	ORANGE
6	RS 485 (B+)	GREEN

6 PIN DECKBOX CONNECTOR

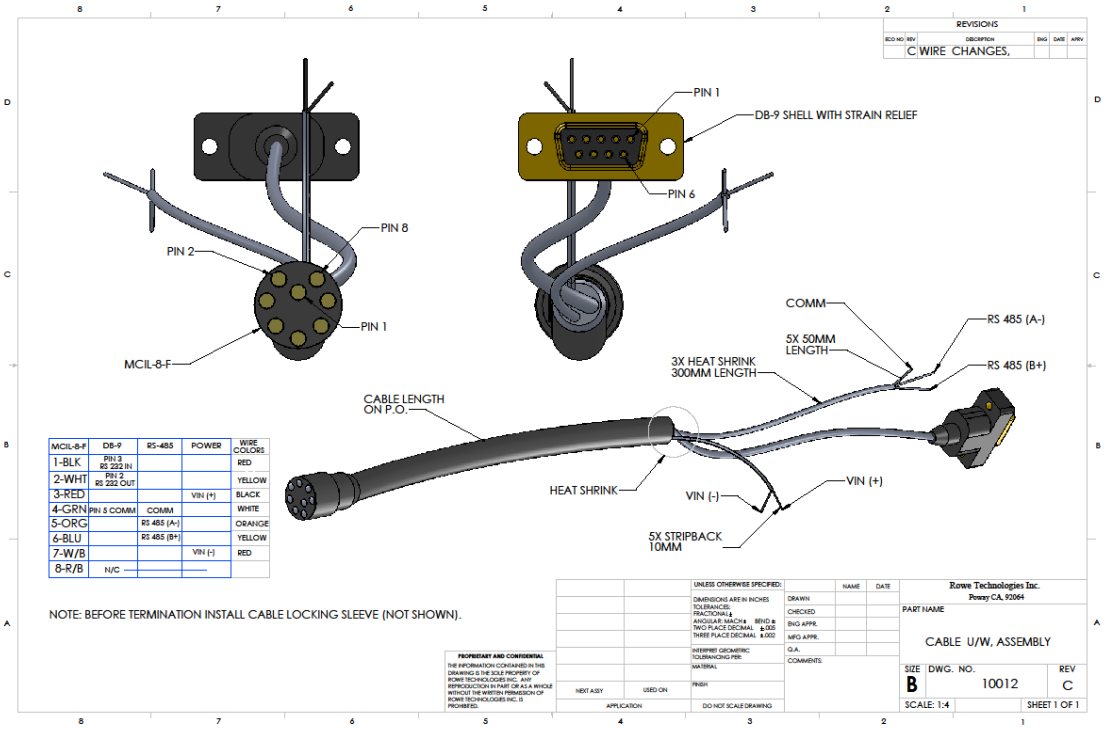
PIN #	SIGNAL	WIRE COLOR
1	RS 485 (A-)	ORANGE
2	RS 485 (B+)	GREEN
3	N/C	
4	N/C	
5	DATA COM	BLACK

RS 485 /USB CONNECTOR

PIN #	SIGNAL	WIRE COLOR
1	N/C	
2	RS 232 IN	BLUE
3	RS 232 OUT	WHITE
4	N/C	
5	DATA COM	BLACK
6	N/C	
7	N/C	
8	N/C	
9	N/C	

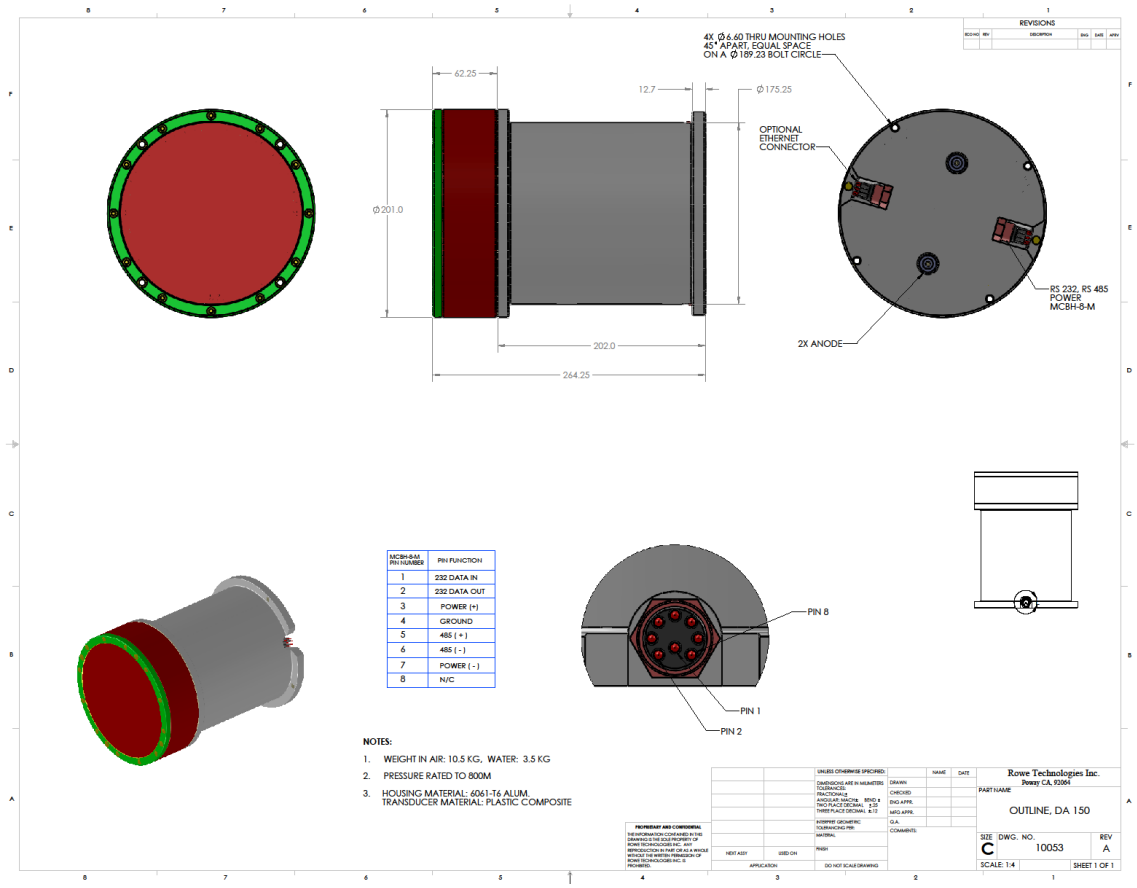
RS 232 (DB- 9) CONNECTOR

4.2 Pigtail Cable Configuration Diagram



MCIL-8-F	DB-9	RS-485	POWER	WIRE COLORS
1-BLK	PIN 3 RS 232 IN			RED
2-WHT	PIN 2 RS 232 OUT			YELLOW
3-RED			VIN (+)	BLACK
4-GRN	PIN 5 COMM	COMM		WHITE
5-ORG		RS 485 (A-)		ORANGE
6-BLU		RS 485 (B+)		YELLOW
7-W/B			VIN (-)	RED
8-R/B	N/C			

4.3 End Cap Connector



MCBH-8-M PIN NUMBER	PIN FUNCTION
1	232 DATA IN
2	232 DATA OUT
3	POWER (+)
4	GROUND
5	485 (+)
6	485 (-)
7	POWER (-)
8	N/C

4.4 ADCP/DVL Operations with Serial Communications

1. Serial Communications
    a. Electrical Interface
        i. RS-232 (standard)
            1. 8 bit, No Parity, 115200 Baud (can go higher for short distances)
            2. Full Duplex (3 wire, shares return with 485)
            3. Minimal noise immunity
            4. Short distances (100ft)
        ii. RS-485 (standard)
            1. 9 bit, Multi-drop Mode Address/Data Parity, up to 921600 Baud
            2. Half Duplex (3 wire, shares return with 232)
            3. Good noise immunity
            4. Long distances (4000ft)
        iii. RS-422 (precludes 232 and 485)
            1. 8 bit, No Parity, up to 921600 Baud
            2. Full Duplex (requires all 5 communication wires)
            3. Good noise immunity
            4. Long distances (4000ft)
    b. Protocol
        i. ASCII commands are terminated with a carriage return <CR>. Using the example "START<CR>" the firmware inputs each ASCII character one at a time checking for a 0x0d (13) or <CR>. When the <CR> is detected the firmware converts the most resent characters in the input buffer into a string which is then compared to a list of valid commands. The command is echoed (sent back out the data port) to the sender followed by either a 0x06 <ACK> or a 0x15 <NAK>. The <ACK> indicates to the sender that command is valid and will be acted upon. A <NAK> indicates to the sender that the command is not valid and no action will be taken. In the example "START<CR>" the system would recognize the command as valid and echo START<ACK> and then proceed to collect profile data.
2. Windows based PC RS232 Setup
    a. On a Windows based PC install a RS232 to USB convertor.
    b. Note communication port associated with the convertor.
    c. Connect the DB9 RS232 connector to the USB to RS232 convertor.
    d. Connect the 8 pin SEACON MCIL-8-F underwater connector to the ADCP/DVL instrument housing.
    e. Connect the 2 power wires to an appropriate DC power source.
    f. Turn on the DC power source.
3. Windows based PC RS485 Setup
    a. On a Windows based PC install the R485 device driver and the RS485 to USB convertor.
    b. Note communication port associated with the convertor.

c. Connect the RS485 terminal block to RS485 convertor.
d. Connect the 8 pin SEACON MCIL-8-F underwater connector to the ADCP/DVL instrument housing.
e. Connect the 2 power wires to an appropriate DC power source.
f. Turn on the DC power source.

4. Windows based PC RS485 with Deck box Setup
   a. Install the R485 device driver and the RS485 to USB convertor.
   b. Note communication port associated with the convertor.
   c. Connect the RS485 terminal block to RS485 convertor.
   d. Connect the 4 or 5 pin Switch craft Conxall connector to the RTI deck box.
   e. Connect the 8 pin Switch craft Conxall connector to the RTI deck box.
   f. Connect the 8 pin SEACON MCIL-8-F underwater connector to the DP300 instrument housing.
   g. Connect the 3 pin AC cord to the RTI deck box.
   h. Plug the AC power cord into a conveniently located AC outlet (90 - 230 VAC).
   i. On the RTI deck box turn the power switch located near the AC power cord to ON.
   j. Green LED near the power switch should light up.
   k. The data indicator light on the RTI deck box should illuminate briefly then go out after power is applied to the ADCP.
5. Windows based software setup:
   a. Using the communication port from above refer to the DP-Pro software user guide to enable communications between the ADCP/DVL and the PC serial port.
   b. DP-Pro software can be used to send commands to the DVL.

4.5 Self-Contained ADCP

1. Internal battery pack(s). Internal packs reside within the same pressure housing as the Doppler electronics.
   a. Single 445 W-hr Alkaline pack, RTI PN 10172 Rev E
      i. Refer to Rowe Technologies document number DOC0021 for a description of how to replace/install a single internal battery pack. Also see Appendix D1 through D3.
      ii. After electrically connecting the battery to the system electronics, using the available J7 or J8 connector on the SC wire harness, the system is powered. Additionally, depending on the last state of the ADCP before it was powered down, the ADCP may start pinging. To prevent the battery from discharging prematurely, an external DC supply, such as the 36 VDC unit included with the ADCP, must be connected to an underwater cable connected to the end cap connector.
      iii. The input voltage wire (see Appendix D1 and D2) on the end-cap contains a diode in series with it to prevent DC current flow between an external power supply and the internal battery. The battery pack also has a protection diode

internal to the pack. The DC voltage of a fresh 19 cell pack is approximately 32 Volts.

iv. The battery pack has 5 Amp fuse to protect against sort circuit failures. If you need to change the fuse we recommend that you use:

Littelfuse Inc 0AGW005.V

b. Compass Calibration
   i. The battery pack must be degaussed prior to installation.
   ii. The internal compass must be recalibrated anytime the internal battery pack is changed.

Chapter 5 User Commands and Data Formats

5.1 User Commands (ASCII)

1. Help Commands
    a. **Help**<CR>, **H**<CR>, or **?**<CR> Returns a list all of the available commands.
2. Configuration Commands
    a. Mode:
        i. **CDVL**<CR> Enable DVL Mode. The system, when started, will output NMEA formatted data. Bottom track and a single water tracking cell are supported in this mode.
        ii. **CPROFILE**<CR> Enable Profiling mode. The system, when started, will output binary or text formatted data. Bottom track and multi cell water profiling supported in this mode.
    b. Water Profiling: (used in Profiling mode)

        Note: To control subsystems other than 0 add [c] to the command string, where c is the subsystem number.

        i. **CWPON n**<CR> Water Profile Pings On. Enables or disables water profile pings.
            1. n = 0. Disables water profiling.
                a. When the application requires bottom tracking only disabling profiling allows for more bottom pings per second.
            2. n = 1. Enables water profiling.
        ii. **CWPBB 1, 2**<CR> Water Profile Broad Band. Sets water profile coded pulse transmission and lag.
            1. Transmit Pulse Type and Processing
                a. 0 = Narrowband.
                    i. Provides long range profiles at the expense of variance.
                    ii. Not recommended for use with bin size less than the default bin size.
                b. 1 = Broadband.
                    i. Typically 15% less range than narrow band but has greatly reduced variance (depending on lag length).
                    ii. Used in conjunction with CWPBP for small bins.
                c. 2 = Un-coded pulse-to-pulse (no ambiguity resolver).
                    i. Provides ultra low variance for small bin sizes. Non-coded has slightly higher variance than the coded transmit without the annoying autocorrelation side peaks.
                d. 3 = Broadband pulse-to-pulse (no ambiguity resolver).
                    i. Provides ultra low variance for small bin sizes.

e. 4 = Non Coded Broadband pulse-to-pulse (no ambiguity resolver).
    i. Provides ultra low variance for small bin sizes.
f. 5 = Broadband with ambiguity resolver ping.
    i. Used in conjunction with CWPBP averaging.
g. 6 = Broadband pulse to pulse with pulse to pulse ambiguity resolver ping.
    i. Used in conjunction with CWPAP.

2. Lag length in vertical meters (m).
    a. Not used with Narrowband.
    b. A longer lag will have lower variance and a lower ambiguity velocity.

iii. **CWPST 1, 2, 3**<CR> Water Profile Screening Thresholds.
1. Correlation Threshold (0.00 to 1.00).
    a. Used for screening profile beams. A beam with a correlation value less than the threshold will be flagged bad and not included in the bin average. Nominal beam correlation values are dependent on the pulse coding, the number of repeated codes, and whether not the pulse-to-pulse processing is being used. For example:
        i. The pulse-to-pulse nominal correlation is 1.00. A correlation value of 0.50 occurs when the signal is equal to the noise (SNR = 1 or 0dB).
        ii. Broad band correlation is dependent on the number of repeated code sequences in the transmission. If 5 repeats are transmitted the nominal correlation will be 4/5 or 0.80. A correlation value of 0.4, in this case, indicates a signal to noise ratio is 1.
2. Q Velocity Threshold (m/s)
    a. Used for screening transformed profile bins. A bin with a, absolute Q velocity that is higher than the Q threshold will be flagged as bad. Beam coordinate velocity data is not affected.
3. V Velocity Threshold (m/s)
    a. Used for screening transformed profile bins. A bin with a, absolute Vertical velocity that is higher than the V threshold will be flagged as bad. Beam coordinate velocity data is not affected.

iv. **CWPBL n.nn**<CR> Water Profile Blank (meters). n.nn = 0 to 100. Sets the vertical range from the face of the transducer to the first sample of the first bin.

v. **CWPBS n.nn**<CR> Water Profile Bin Size (meters). n.nn sets the vertical bin size.

vi. **CWPX n.nn**<CR> Water Profile Transmit (meters). n.nn sets the vertical transmit size. A value of 0.00 (default) will cause the system to set transmit to the same length as the bin size.

vii. **CWPBN n**<CR> Water Profile Bin N. n = 0 to 200 sets the number bins that will be processed and output.

viii. **CWPP n**<CR> Water Profile Pings. n = 0 to 10,000 sets the number of pings that will be averaged together during the ensemble.

ix. **CWPBP 1, 2**<CR> Water Base Pings (used when CWPBB = 0 or 1). This command should be used when small, non pulse to pulse, bin sizes are required for profiling. The Base Pings are averaged together as a complex number which allows for better correlation screening and averaging.
    1. Pings a value of 2 to 100 sets the number of pings that will be averaged together during each CWPP ping. A value of 0 or 1 disables Base Ping averaging.
    2. Time Between Base Pings (seconds). Normally it is a small number i.e. 0.01 for the 1200 kHz system.

x. **CWPTBP n.nn**<CR> Water Profile Time Between Pings. n.nn = 0.00 to 86400.00 seconds (24 hours) sets the time between the last ping, regardless of ping type, and the next profile ping.

xi. **CWPAP 1, 2, 3, 4, 5**<CR> Water Ambiguity Ping (used when CWPBB = 5). Pulse to pulse ping and processing is used for the ambiguity resolver therefore (Blank + Bin Size) < Lag.
    1. 0 to 100 sets the number of pings that will be averaged together.
    2. Lag (meters) sets the length of the lag.
    3. Blank (meters) sets the starting position of the bin.
    4. Bin Size (meters).
    5. Time Between Ambiguity Pings (seconds).

c. Ensemble: (used in Profiling mode)

i. **CEI HH:MM:SS.hh**<CR> Ensemble Interval. Sets the time interval that system will output the averaged profile/bottom track data.

> Note: all digits including the space following CEI and the separators must be part of the command or the system will reject it.

ii. **CBI HH:MM:SS.hh,n,m**<CR> Burst Interval. Used when a precise short time interval is required between ensembles followed by a period of sleep.
    1. HH:MM:SS.hh sets the time interval between a series of ensembles.
    2. n sets the number of ensembles that are output during each burst. The time between each ensemble is controlled by the CEI command.

3. M sets the Burst Pair flag. 0 = false, 1 = true; If Burst Pair is set to 1 the next sub system will be interleaved (alternating pings) with the current sub system during the burst. The CBI commands for the pair should be set to the same value(s).

   Note: Burst sampling typically requires precise timing intervals. If serial data output is enabled during burst sampling the user needs to ensure that each data transfer completes before the next ensemble is output. If the length of the data transfer exceeds the time between samples the data will bottleneck and the desired sample rate will not be maintained. Self-contained ADCP users can disable the serial data output by sending the command **CEOUTPUT 0**<CR>. Of course you need to enable the internal data recording **CERECORD 1**<CR>. During burst sampling, internal data recording only occurs at the end of the burst or when the STOP<CR> command is received by the ADCP. Another option, when real-time data is required, is to increase the data output rate by using the C485B 921600<CR> or C422B 921600<CR> command. Be aware that the RS232 connection may not reliably support the higher data rates.

iii. **CEPO ccccccccccccccc**<CR> Ensemble Ping Order. Sets the order in which the various subsystems will be pinged. Note: a space and at least one subsystem code must follow CEPO or the system will reject the command. For example:
   1. CEPO 23<CR> will ping subsystem 2 followed by sub system 3.
   2. CEPO 32<CR> will ping subsystem 3 followed by sub system 2.
   3. CEPO 22<CR> will ping subsystem 2 followed by sub system 2 using a different setup (i.e. bin size).

iv. **CETFP YYYY/MM/DD,HH:mm:SS.hh**<CR> Ensemble Time of First Ping. Sets the time that the system will awaken and start pinging.

   Note: all digits including the space following CETFP and the separators must be part of the command or the system will reject the command.

v. **CERECORD n,m**<CR> n=Ensemble Recording 0=disable, 1=enable. m=SinglePing Recording 0=disable, 1=enable.
   1. When ensemble recording is enabled and the ADCP is started (START<CR>) the firmware searches for the next available file number to record to on the SD card. The ensemble file name starts with the letter “A” followed by a 7 digit number and ending with the extension “.ens”. For example: The first ensemble file will be named “A0000001.ens”. During deployment as each ensemble is completed the data is

appended to the current file. The 7 digit number following the "A" is incremented each time the system is (re)started or when the file size exceeds 16Mbytes bytes.
Note: Internal ensemble data recording during burst sampling only occurs at the end of the burst.

2. When single recording is enabled and the ADCP is started (START<CR>) the firmware searches for the next available file number to record to on the SD card. The single ping file name starts with the letter "S" followed by a 7 digit number and ending with the extension ".ens". For example: The first single ping file will be named "S0000001.ens". During deployment as each ping is completed the data is appended to the current file. The 7 digit number following the "S" is incremented each time the system is (re)started or when the file size exceeds 16Mbytes bytes. Each ping, whether bottom track or profile, is considered to be a single ping.

   Note: No error/ threshold screening or coordinate transformation is performed on the data contained in a single ping file.

vi. **CEOUTPUT n**<CR> Ensemble output type.
   1. n = 0 disables serial output. Saves battery energy when recording data to the SD card during a self-contained deployment by reducing extra on time of the system due to data transfer.
   2. n = 1 enables the standard binary output data protocol to be sent out the serial port.
   3. n =2 enables an ASCII text serial output that is dumb terminal compatible.

d. Bottom Tracking (used in profiling and DVL mode):
   Note: To control subsystems other than 0 add [c] to the command string, where c is the subsystem number.

i. **CBTON n**<CR> Bottom Track ON. Enables or disables bottom track pings.
   1. n = 0 disable bottom tracking. Allows for more water profile pings per second and saves battery energy during self-contained deployments.
   2. n = 1 enable bottom tracking. When enabled a bottom track ping occurs once per ensemble. Or, when profiling is enabled, a bottom track ping occurs at the beginning of the ensemble and then after every 10 profile pings in the ensemble. If there are less than 10 profile pings per ensemble the bottom track ping will only occur once at the beginning of the ensemble.

ii. **CBTBB mode, p to p lag, LR depth**<CR> Bottom Track BB control.
   1. Mode

a. n = 0 Narrowband long range.
b. n = 1 Broadband coded transmit.
c. n = 2 Broadband non-coded transmit.
d. n = 3 NA.
e. n = 4 Broadband non-code pulse to pulse.
f. n = 5 NA.
g. n = 6 NA.
h. n = 7 Auto Switch between n = 0, n = 2, and n = 4.

2. Pulse-to-Pulse Lag(m)
    a. Lag length in vertical meters. When enabled bottom track will use pulse-to-pulse transmit and processing at depths less than ½ the lag length. Allows for near bottom ultra low variance velocity measurements.
3. Long Range depth (m).
    a. The range in meters beyond which the bottom track will switch to narrowband long range processing when n = 7.

iii. **CBTST 1, 2, 3**<CR> Bottom Track Screening Thresholds.
1. Correlation Threshold (0.00 to 1.00).
    a. Used for screening beam data. A beam with a correlation value less than the threshold will be flagged bad and not included in the average. Nominal correlation for bottom tracking is 1.
2. Q Velocity Threshold (m/s)
    a. Used for screening transformed bottom track velocities. An absolute Q velocity that is higher than the Q threshold will be flagged as bad. Beam coordinate velocity data is not affected.
3. V Velocity Threshold (m/s)
    a. Used for screening transformed bottom track velocities. An absolute Vertical velocity that is higher than the V threshold will be flagged as bad. Beam coordinate velocity data is not affected.

iv. **CBTT a, b, c, d**<CR> Bottom Track Thresholds. Where:
a. SNR(dB) shallow detection threshold.
b. Depth(m) at which the bottom track switches from using the shallow to the deep SNR.
c. SNR(dB) deep detection threshold.
d. Depth(m) at which the bottom track switches from low to high gain receive.
e.

v. **CBTBL n.nn**<CR> Bottom Track Blank (meters). n.nn = 0 to 10. Sets the vertical distance from the face of the transducer at which the bottom detection algorithm begins searching for the bottom.

vi. **CBTMX n.nn**<CR> Bottom Track Max Depth (meters). n = 5 to 10000. Sets the maximum range over which the bottom track algorithm will search for the bottom. A large value will slow acquisition time.

vii. **CBTTBP n.nn**<CR> Bottom Track Time Between Pings. n.nn = 0.00 to 86400.00 seconds (24 hours) sets the time between the last ping, regardless of ping type, and the next bottom ping.

e. **Water Tracking:** (used in DVL mode)

Note: To control subsystems other than 0 add [c] to the command string, where c is the subsystem number.

i. **CWTON n**<CR> Water Track On. Enables or disables water track pings. n = 0 to 1, 0 = disable, 1 = enable.

ii. **CWTBB n**<CR> Water Track Broadband. Enables or disables water track broadband processing. n = 0 to 1, 0 = disable, 1 = enable.

iii. **CWTBL n.nn**<CR> Water Track Blank (meters). n.nn = 0 to 100. Sets the vertical range from the face of the transducer to the first sample of the bin.

iv. **CWTBS n.nn**<CR> Water Track Bin Size (meters). n.nn sets the vertical bin size.

v. **CWTTBP n.nn**<CR> Water Track Time Between Pings. n.nn = 0.00 to 86400 sets the time between bottom pings.

f. Environmental (used in all modes):

i. **CWSSC 1, 2, 3, 4** <CR> Water Speed of Sound Control. 0 = command, 1 = sensor, 2 = internal calculation.

1. Water Temperature source
2. Transducer Depth source
3. Salinity source
4. Speed of Sound source

ii. **CWS n.nn**<CR> Water Salinity (ppt). Used in the water speed of sound calculation.

iii. **CWT n.nn**<CR> Water Temperature (degrees). Used in the water speed of sound calculation if the temperature sensor is not available.

iv. **CTD n.nn**<CR> Transducer Depth (meters). Used in the water speed of sound calculation.

v. **CWSS n.nn**<CR> Water Speed of Sound (meters per second). Used when the speed of sound source is set to sensor or command i.e. CWSSCx,x,x,0<CR> or CWSSCx,x,x,1<CR> . During standby and pinging the firmware monitors the communication ports for changes in CWSS and for the NMEA string $AML,SVM,1481.772,SN,131196*08. The $AML string is used on the Ocean Server platform for speed of sound input.

vi. **CHO n.nn**<CR> Heading Offset(+-180 deg). Added to the compass output prior to heading being used within the system.

vii. **CHS n**<CR> Heading Source. Select the heading source for ENU transformations. n = 1 for internal (PNI) compass, n = 2 for GPS HDT string via either serial port.

viii. **CVSF n.nn**<CR> = Velocity Scale Factor. This scale factor is applied to all velocity measurement data. New Velocity = CVSF * Velocity.

ix. **CPZ**<CR> Zero Pressure Sensor. Sets the current pressure reading to the zero point (if pressure sensor is installed).

g. Data Communications:

i. **CEMAC**<CR> temporarily enables Ethernet communication. This command is typically sent via one of the serial ports.

1. One of two responses occurs on the serial port after the command is accepted by the ADCP.

```
MAC 02:ff:fe:fd:fc:fb
IP  192.168.1.130
Link OK

OR

MAC 02:ff:fe:fd:fc:fb
IP  192.168.1.130
No Link
```

ii. IP n.n.n.n<CR> sets a new Ethernet address in the ADCP. The ADCP response to the IP command will be the same as CEMAC.

iii. **C232B n**<CR> RS232 bps. n = 2400, 4800, 9600, 19200, 38400, 115200, 230400, 460800, 921600.

iv. **C485B n**<CR> RS485 bps. n = 2400, 4800, 9600, 19200, 38400, 115200, 230400,460800, 921600.

v. **C422B n**<CR> RS422 bps. n = 2400, 4800, 9600, 19200, 38400, 115200, 230400,460800, 921600.

h. System Configuration:

i. **CLOAD**<CR> Loads the file “SYSCONF.BIN”, contained on the SD card, into the system configuration.

ii. **CSAVE**<CR>= Saves a copy of the system configuration to the file “SYSCONF.BIN” on the SD card.

iii. **CSHOW**<CR> Causes the system to display/output the system configuration

```
CSHOW<CR>
CSHOW
DP1200 System Configuration:
  Mode Profile
  Sim 0
  CEI 00:00:01.00
  CEPO 22
  CETFP 0000/00/00,00:00:00.00
  CERECORD 0
  CEOUTPUT 1
  CWPON[0] 1 [1] 1
  CWPBB[0] 3,0.300 [1] 1,0.050
```

```
CWPST[0] 0.400,1.000,1.000 [1] 0.400,1.000,1.000
CWPBL[0] 0.10 [1] 0.10
CWPBS[0] 0.04 [1] 1.00
CWPX [0] 0.00 [1] 0.00
CWPBN[1] 20 [1] 40
CWPP[0] 1 [1] 1
CWPBP[0] 10,0.00 [1] 1,0.02
CWPTBP[0] 0.13 [1] 0.13
CBTON[0] 1 [1] 0
CBTBB[0] 0, 0.000, 30.00 [1] 0, 0.000, 0.00
CBTST[0] 0.900,1.000,1.000 [1] 0.000,0.000,0.000
CBTBL[0] 0.05 [1] 0.00
CBTMX[0] 75.00 [1] 0.00
CBTTBP[0] 0.05 [1] 0.00
CBTT[0] 15.0,25.0,5.0,2.0 [1] 0.0,0.0,0.0,0.0
CWTON[0] 0 [1] 0
CWTBB[0] 0 [1] 0
CWTBL[0] 2.00 [1] 0.00
CWTBS[0] 2.00 [1] 0.00
CWTTBP[0] 0.13 [1] 0.00
CWS 0.00
CWT 15.00
CTD 0.00
CWSS 1500.00
CHO 0.00
CHS 1
CVSF 1.000
C232B 115200
C485B 115200
C422B 115200
```

iv. **CDEFAULT**<CR> = Restores the system configuration to factory defaults

Note: Actual default values will vary depending on system type.

Note: Default values are set for all hardware configured subsystems. When using the CEPO command defaults for repeated subsystems are set to zero (0). You need to set all of the commands when using a single subsystem for multiple setups.

3. Firmware Commands
   a. Information:
      i. **FMSHOW**<CR> Show Firmware Version and creation date.

```
FMSHOW
Copyright (c) 2009-2012 Rowe Technologies Inc. All rights reserved.
System Firmware Version: 00.02.05 Mar 21 2012 06:44:56
```

   b. Update: (obsolete in version 0.2.0. The newer system boots from the SD card)
      i. **FMCopyB**<CR> Copy the boot firmware “rtiboota.bin” from the secure digital storage to internal Flash.
      ii. **FMCopyS**<CR> Copy the system firmware “rtisys.bin” from the secure digital storage to internal Flash.

4. System Deployment Commands:
   a. System Control:
      i. **START**<CR> Start pinging continuously. Once started the system will only respond the STOP, STIME, and CSHOW commands.

ii. **STOP**<CR> Stop pinging and return to the command input mode.
iii. **SLEEP**<CR> Power down system.

b. Geo Position:
i. **SPOS**<CR> Display system geo position
ii. **SPOS** Lat(deg),Lon(deg),Depth(m)<CR> Set system geo position and water depth

c. Real Time Clock:
i. **STIME**<CR> Display system time.
ii. **STIME YYYY/MM/DD,HH:MM:SS**<CR> Set system Time.

Note: all digits including the space following STIME and the separators must be part of the command or the system will reject it.

5. Data Storage Commands:
a. File Transfer:
i. **DSXRfilename.abc**<CR> Data Storage XMODEM Receive (upload). This command is used to transfer a file, via the serial communication link, from an external device to the SD card contained within the RTI system. File names are limited to a maximum of 8 characters before the extension.
ii. **DSXTfilename.abc**<CR> Data Storage XMODEM Transmit (download). This command is used to transfer a file, via the serial communication link, from the SD card contained within the RTI system to an external device. File names are limited to a maximum of 8 characters before the extension.

b. Secure Digital:
i. **DSFORMAT**<CR> Data Storage Format. Stores the system files in FLASH then completely erases the SD card after which rewrites the system files to the SD card.
ii. **DSDIR**<CR> Data Storage Directory. Show a directory of stored files.
iii. **DSSHOW**<CR> Data Storage Show. Shows SD card capacity and usage.

6. Diagnostic Commands:
a. Compass:
i. **DIAGCPT**<CR> Diagnostic Compass Pass Thru. Allows an external device to connect directly to the internal compass via a serial communication link. To disconnect the pass-thru the external device must send 16 consecutive X's (XXXXXXXXXXXXXXXX) to the system. The pass thru allows external software to directly access and calibrate the compass. The compass Baud rate is 38400 and should never be changed. The ADCP default Baud rate is 115200 which can be changed if desired.
ii. **DIAGSAMP**<CR> Diagnostic Ping. See an example of this command in the System Bench test procedure.

5.2 User Command Examples

Example 1: The following is optimized for a 150 kHz DVL. This list of commands will instruct a RTI DVL to collect a broadband bottom track and 16 meter broadband water bin.

1. CDEFAULT (Clear out all previously entered commands)
2. CDVL (Place the system in the DVL mode)
3. CWTON1 (Turns on the water track bin)
4. CWTBS16 (Selects a 16 meter bin for water track)
5. CWTTBP 1 (Sets 1 second between water track pings)
6. CWTBB 1 (Selects Broadband water track processing)
7. CBTON 1 (Turns Bottom track on)
8. CBTBB 1 (Selects Broadband bottom track processing)
9. CSAVE (Save the command setup to the SD card for power up reset)
10. START (Begin pinging and output NMEA data to serial port)

Example 2: The following is optimized for a 300 kHz self-contained ADCP. The following list of commands will instruct a RTI ADCP to collect a 120 meter Broadband profile with bottom track disabled.

1. CDEFAULT (Clears out all previously entered commands)
2. CERECORD 1 (turn on internal SD recording)
3. CEOUTPUT 0 (turn off serial communication ensemble output)
4. CPROFILE (Places the DP300 in the profile mode)
5. CWPBS 4 (Sets the bin size to 4 meters)
6. CWPBN 30 (Selects 30 bins for processing and output)
7. CEI 01:00:00.00 (Sets a 1 hour averaging interval)
8. CWPTBP 720 (Sets 12 minutes between profile pings)
9. CWPBB 1, 1.0 (Selects Broadband profile processing with a 1 meter lag)
10. CWPP 5 (Sets a 5 pings per ensemble)
11. CBTON 0 (Turns Bottom track off)
12. CSAVE (Save the command setup to the SD card for power up reset)
13. START(Begin pinging and output Binary data to serial port)

Example 3: The following is optimized for a dual frequency 300/1200 kHz 20 degree piston ADCP. The following list of commands will instruct the 300 kHz RTI ADCP to collect a 120 meter Narrowband profile with bottom track enabled and the 1200 kHz system to collect a 5 meter Broadband profile. The system SN is 01420000000000000000000000000015

1. CDEFAULT (Clears out all previously entered commands)
2. CPROFILE (Places the DP300/1200 in the profile mode)
3. CEI 00:15:00.00 (Sets a 15 minute ensemble output rate)
4. CEPO 10 (select subsystem 1 to ping first followed by 0)
5. CWPBS[0] 4 (Sets the bin size to 4 meters)
6. CWPBN[0] 30 (Selects 30 bins for processing and output)
7. CWPP[0] 10 (Sets a 10 profile pings per ensemble)
8. CWPTBP[0] 0.5 (Sets 0.5 second between profile pings)
9. CWPBB[0] 0,X (Selects Narrowband profile processing)

10. CBTON[0] 1 (Turns Bottom track on)
11. CBTBB[0] 1 (Selects Broadband bottom track processing)
12. CWPBS[1] 0.5 (Sets the bin size to 0.5 meters)
13. CWPBN[1] 10 (Selects 10 bins for processing and output)
14. CWPP[1] 10 (Sets a 10 ping per ensemble)
15. CWPTBP[1] 0.25 (Sets 0.25 second between profile pings)
16. CWPBB[1] 1,0.1 (Selects Broadband profile processing with a 0.1 meter lag)
17. CBTON[1] 0 (Turns Bottom track off)
18. CSAVE (Save the command setup to the SD card for power up reset)
19. START(Begin pinging and output Binary data to serial port)

Example 4: The following is optimized for a 600 kHz self-contained ADCP. The following list of commands will instruct a RTI ADCP to collect a 2 bins of Broadband velocity data with bottom track disabled.

1. CDEFAULT (Clears out all previously entered commands)
2. CERECORD 1 (turn on internal SD recording)
3. CEOUTPUT 0 (turn off serial communication ensemble output)
4. CPROFILE (Places the DP600 in the profile mode)
5. CWPBS 0.5 (Sets the bin size to 0.5 meter)
6. CWPBN 2 (Selects 2 bins for processing and output)
7. CEI 00:00:00.25 (Sets a 0.25 second ensemble interval)
8. CBI 00:15:00.00,1024 (Sets a 15 minute burst interval containing 1024 ensembles)
9. CWPP 1 (Sets 1 profile ping per ensemble)
10. CWPBB 1,0.25 (Selects Broadband profile processing with a 0.25 meter lag)
11. CBTON 0 (Turns Bottom track off)
12. CSAVE (Save the command setup to the SD card for power up reset)
13. START(Begin pinging and save data to the SD card)

5.3 NMEA Data

1. The standard RTI DVL output is a NMEA type string that starts with a $ and ends with a linefeed. The data strings are RS232/RS485 serial ASCII at 115200 Baud, 8 bit, No Parity.
2. NMEA strings have the following format:
    a. Leading dollar sign ($).
    b. Identity string, followed by a comma (,).
    c. Data fields delimited by commas (final data field has no trailing comma).
    d. An asterisk (*).
    e. Two digit hexadecimal checksum. Checksum is computed from all characters between the leading $ and trailing * characters. Value of the checksum is the exclusive-or (XOR) of those characters.
    f. Carriage return followed by a line feed <CR><LF>
3. The RTI proprietary strings:

```
$PRTI01,ssssssshh,nnnnnn,TTTT,XXXX,YYYY,ZZZZ,DDDD,xxxx,yyyy,zzzz,dddd,ABCD*FF<CR><LF>
Where:
PRTI01 = Identity string.
sssssshh = Start time of this sample in hundreds of seconds since power up or user reset.
nnnnnn = Sample number
TTTT = Temperature in hundreds of degrees Celsius.(measured at the transducer)
XXXX = Bottom track X velocity component mm/s.    (-99999 indicates no valid velocity)
YYYY = Bottom track Y velocity component mm/s.    (-99999 indicates no valid velocity)
ZZZZ = Bottom track Z velocity component mm/s.    (-99999 indicates no valid velocity)
DDDD = Depth below transducer in mm. (range to the bottom in front of the transducer, 0 = no
detection)
xxxx = Water mass X velocity component mm/s.      (-99999 indicates no valid velocity)
yyyy = Water mass Y velocity component mm/s.      (-99999 indicates no valid velocity)
zzzz = Water mass Z velocity component mm/s.      (-99999 indicates no valid velocity)
dddd = Depth of water mass measurement in mm.  (position of the bin in front of the transducer)
ABCD = Built in test and status bits in hexadecimal (0000 = OK).

$PRTI02,ssssssshh,nnnnnn,TTTT,EEEE,NNNN,UUUU,DDDD,eeee,nnnn,uuuu,dddd,ABCD*FF<CR><LF>
Where:
PRTI02 = Identity string.
sssssshh = Start time of this sample in hundreds of seconds since power up or user reset.
nnnnnn = Sample number
TTTT = Temperature in hundreds of degrees Celsius. (measured at the transducer)
EEEE = Bottom track East velocity component mm/s.  (-99999 indicates no valid velocity)
NNNN = Bottom track North velocity component mm/s. (-99999 indicates no valid velocity)
UUUU = Bottom track Up velocity component mm/s.    (-99999 indicates no valid velocity)
DDDD = Depth below transducer in mm. (range to the bottom in front of the transducer, 0 = no
detection)
eeee = Water mass East velocity component mm/s.    (-99999 indicates no valid velocity)
nnnn = Water mass North velocity component mm/s.   (-99999 indicates no valid velocity)
uuuu = Water mass UP velocity component mm/s.      (-99999 indicates no valid velocity)
dddd = Depth of water mass measurement in mm.   (position of the bin in front of the transducer)
ABCD = Built in test and status bits in hexadecimal (0000 = OK). See appendix B5 for description.

$PRTI30,h.hhh,p.ppp,r.rrr*15
Where:
PRTI30 = Identity string.
h.hhh = Heading used during the bottom track ping.
p.ppp = Pitch used during the bottom track ping.
r.rrr = Roll used during the bottom track ping.

$PRTI31,h.hhh,p.ppp,r.rrr*15
Where:
PRTI31 = Identity string.
h.hhh = Heading used during the water mass ping.
p.ppp = Pitch used during the water mass ping.
r.rrr = Roll used during the water mass ping.
```

5.4 Binary Data

1. Version 4 MAT-File Format (MATLAB compatible)
    a. A MAT-file may contain one or more matrices. The matrices are written sequentially to a file, with the bytes forming a continuous stream. Each matrix starts with a fixed-length 20-byte header that contains information describing certain attributes of the Matrix. The 20-byte header consists of five long (4-byte) integers.
    b. Velocity, Amplitude, and Statistical data are contained in 50 x 4 arrays (bin x beam).
    c. Ancillary data such as Time, Heading, Pitch, Roll and Temperature are contained in a 1 x n array. Where n is the number of columns.
2. Ensemble Binary Data Packet (3 sections)
    a. 32 byte Header
        i. 16 bytes containing 0x80
        ii. 4 byte Ensemble number
        iii. 4 byte Ensemble number ones compliment
        iv. 4 byte payload size (in bytes)
        v. 4 byte payload size ones compliment
    b. Payload
        i. Version 4 MAT-File Format (see matlab4.pdf for details)
            1. A MAT-file may contain one or more matrices. The matrices are written sequentially to a file, with the bytes forming a continuous stream. Each matrix starts with a fixed-length 20-byte header that contains information describing certain attributes of the Matrix. The 20-byte header consists of five long (4-byte) integers.
        ii. Velocity, Amplitude, and Statistical data are contained in bins x 4 arrays (row x column).
        iii. Ancillary data such as Time, Heading, Pitch, Roll and Temperature are contained in an n x 1 array. Where n is the number of rows.

    c. CRC-16 Checksum (see appendix B for example)
        i. 4 byte checksum of all the bytes in the payload
        ii. First 2 bytes are always 0
        iii. The second 2 bytes contain the CRC-16 value
            1. CCITT 16 bit algorithm (X^16 + X^12 + X^5 + 1)
            2. CRC seed = 0
3. Defined RTI matrix names in payload (see Appendix B for details)
4. Binary Ensemble decoding (see Appendix B for example)

Chapter 6 System Checkout and Diagnostics

6.1 System Bench Test

1. Power On Test:
    a. Apply power to the DVL/ADCP via the deck box and observe that the wakeup message contains no error messages.
    b. Possible error messages are:
        i. Hardware Timeout!
            1. Indicates that a hardware problem occurred during power on. Refer to the diagnostic procedure to troubleshoot this problem.
2. PNI Prime Magnetic Compass Test:
    a. Enable the compass diagnostic mode by sending the command: DIAGCPT<CR>.
    b. Start the PNI software “StudioPrime_V1p7.exe”.
    c. On the Connection tab establish communication with the compass. The Computer Baud Rate should 115200. Do not change the Module Baud Rate.
    d. On the Configuration tab:
        i. Mounting Options should be Standard or Standard 180 depending on system type.
        ii. Acquisition Settings should be Push.
        iii. Calibration Settings should be set to Mag Only Calibration and Automatic Sampling should not be checked.
        iv. If you have changed any settings click Save.
    e. On the Calibration tab: (refer to section 6.2 in the Prime user manual for the complete procedure).
        i. Click Take Sample each time the DA150 is positioned per 6.2.1 in the Prime user manual.
        ii. Verify that the Mag Score and the Accel Score are within the Prime manual limits.
        iii. Click Save Current User Cal when finished.
    f. On the Test tab:
        i. Click Go.
            1. Orient the DVL/ADCP so that the beams are facing down or up and beam 0 is pointed to the North.
            2. Verify that the heading is correct for the direction in which beam 0 is facing.
            3. Verify that the helicopter is upside down if the system is facing downward.
            4. Raise beam 0 higher than beam 1 and verify that the pitch is positive.
            5. Raise beam 1 above beam 0 and verify that the pitch is negative.
            6. Raise beam 2 above beam 3 and verify that roll is positive. Note: ±180 degree is the same as 0 degrees for a downward facing system.
            7. Raise beam 3 above beam 2 and verify that the Roll is negative.
            8. Click Stop.
    g. Exit the Studio Prime software. To reestablish communication with the DVL/ADCP send a string of 16 X’s to the system XXXXXXXXXXXXXXXX<CR>. Or cycle the DVL/ADCP power.
3. Low Signal Level Test:

a. Using a dumb terminal program send the command DIAGSAMP<CR>. Note: this test must run in RF "clean" environment.
b. Observe that the following data are displayed:

```
DIAGSAMP

Auto Gain:
amp     20.00,   30.00,   30.00,   30.00
Hz    xxxx.xx, xxxx.xx, xxxx.xx, xxxx.xx
cor      0.10,    0.09,    0.09,    0.13

Water      22.45 degrees
BackPlane  28.87 degrees
Pressure  0.000000 bar (1Pa = 10^-5 bar)
```

c. "amp" should be between 20 and 40.
d. All of the "cor" values should be less than 0.35.
e. Water degrees should match within 1 degree of an independent measurement of the ambient water or air temperature where the DVL/ADCP is located.
f. Backplane degrees should be within 4 degrees of ambient but will be higher if the system has been recently pinging.

4. High Signal Level Test:
   a. Connect a signal generator with 10vpp sine wave output to a hoop of wire that is laying flat on the face of the transducer.
   b. Set the frequency of the signal generator to 1000 Hz above the operating frequency of the DVL/ADCP.
      150 kHz = 155640 Hz
      300 kHz = 311281 Hz
      600 kHz = 622562 Hz
   c. Using a dumb terminal program send the command DIAGSAMP<CR>.
   d. Observe that the following data are displayed:

```
DIAGSAMP

Auto Gain:
amp    110.00,  110.00,  110.00,  110.00
Hz    1000.00, 1000.00, 1000.00, 1000.00
cor      0.99,    0.99,    1.00,    0.99

Water      22.45 degrees
BackPlane  28.87 degrees
Pressure  0.000000 bar (1Pa = 10^-5 bar)
```

   e. "amp" should be significantly higher than in the Low Level Test.
   f. All of the "Hz" values should be near 1000.
   g. All of the "cor" values should be near 1.00.
5. Power Test:
   a. Make sure the DVL/ADCP transducer is submerged in water.
   b. Measure the AC Power going into the deck box. It should be less than 3 watts.
   c. Send the following commands to the DVL/ADCP:
      i. CBTON1<CR>
      ii. CWTON0<CR>
      iii. CWPON0<CR>
      iv. ENGBTDEEP<CR>
         1. Verify the response is:

ENGBTDEEP
BT DEEP ON

2. This command configures the system for worst case long duration ping.

d. Start the DVL/ADCP pinging by sending START<CR>.
e. Measure the Peak AC Power. It should be greater than 100 watts and less than 1000 Watts.
f. Send STOP<CR> to stop the DVL/ADCP pinging.

6.2 Diagnostics

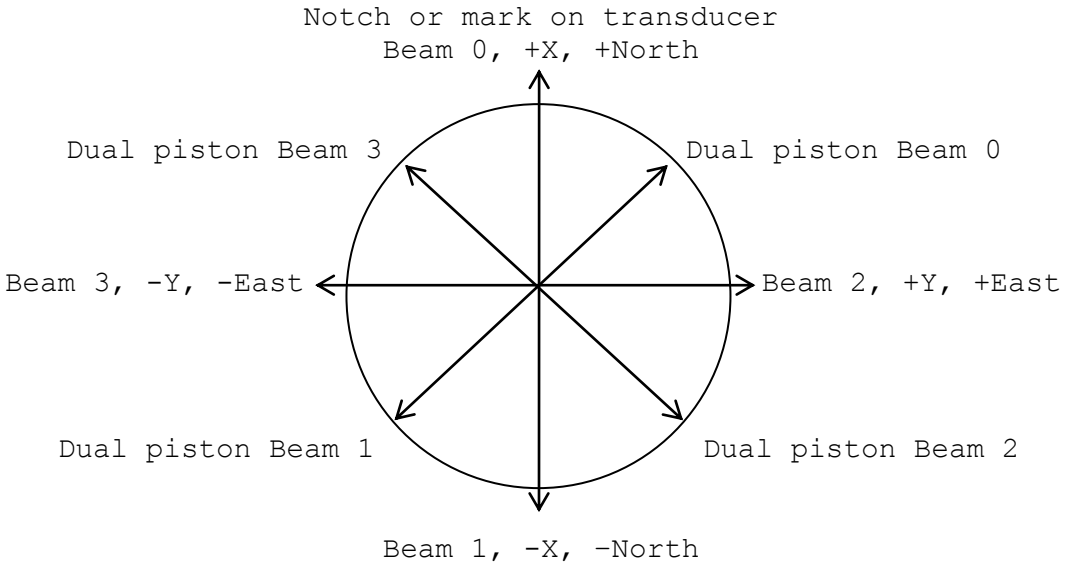
Appendix A

A1 Coordinate systems

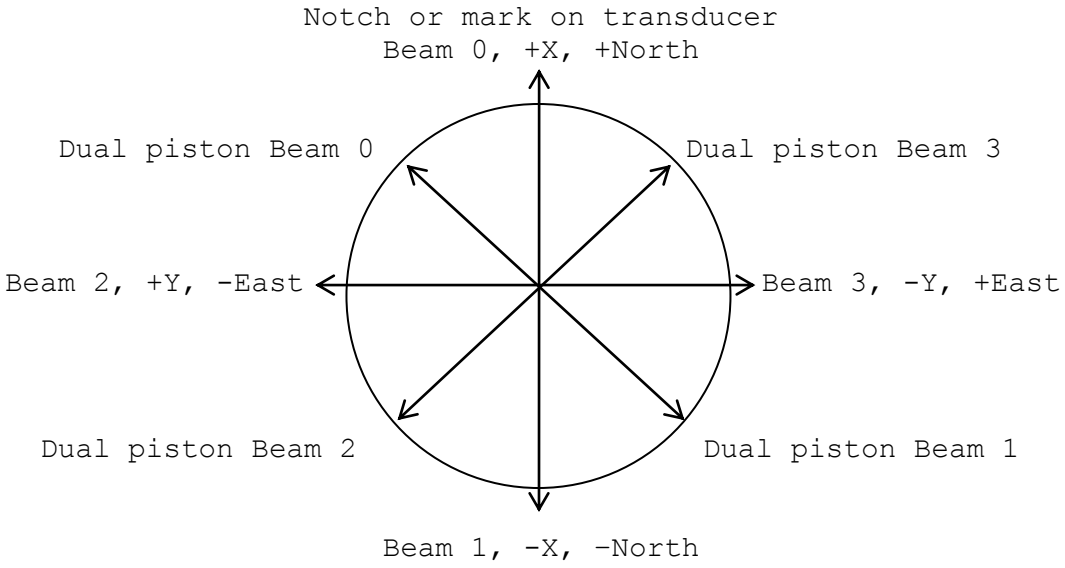
RTI uses three different coordinate systems.

1. Beam
    a. The velocity vector is in the direction each beam points.
    b. Beams are located on azimuth angles of 0, 90, 180, and 270 degrees around transducer.
    c. Beam elevation angles can be 15, 20, or 30 degrees from vertical.
    d. A positive velocity measurement occurs when a beam and an object move closer together.
    e. Beam velocity data is useful for beam diagnostics.
    f. Beam velocity data may require further processing to be useful for navigation or scientific work.
2. Instrument
    a. Beam velocities are rotated to a 3 axis velocity vector X, Y, and Z. The X axis is along a line drawn between beams 0 and 1. The Y axis is along a line drawn between beams 3 and 2. The Z axis is orthogonal to both the X and Y axes.
    b. The instrument coordinate system follows the Right Hand Rule convention.
3. Earth
    a. With the addition of a Heading, Pitch, Roll (HPR) sensor the beam velocities are "bin mapped" to a common horizontal plane and then rotated to the Earth referenced velocity vector East, North, and Up (ENU).

Note: Transformed bottom and water track velocity data are reported as transducer motion. Transformed water profiling bin velocities are reported as water motion. In other words, bottom track velocities have an opposite sign from profile bins if the measured velocity is only due to transducer motion.



Downward facing Transducer (face away from observer)



Upward facing Transducer (face towards observer)

A2 GPS Compass

When using a GPS compass to rotate from the ADCP/DVL XYZ coordinate system to the ENU system use the following:

1. Align beam 0 with the GPS compass North. Beam 0 is located by a small notch on the side of the transducer.
2. Extract the GPS heading from the NMEA HDT string.
3. Process the ADCP XYZ velocity data using the following C# code:

```
double theta = GPS_HDT / 180.0 * Math.PI; //convert from degrees to radians
double CosT = Math.Cos(theta);
double SinT = Math.Sin(theta);
double NorthVelocity = VelocityX * CosT - VelocityY * SinT;
double EastVElocity  = VelocityX * SinT + VelocityY * CosT;
```

4. Repeat the processing for all bins and bottom velocities.

Appendix B

B1 Payload Matrix Contents

1. **E000001**
    a. Single Precision Floating Point
    b. Bins x Beams
        1. Beam Velocity in m/s
            a. Contains the Beam coordinate velocity profile data as measured along each beam. The data is useful for diagnostic purposes and for when the user wants to perform their own transformation. It is arranged beam 0 to 3 for all bins.

2. **E000002**
    a. Single Precision Floating Point
    b. Bins x Beams
        1. Instrument Velocity in m/s
            a. Contains the Instrument coordinate velocity profile. It is arranged X, Y, Z, Q for all bins where:

```
X =  (BmVel[1] - BmVel[0])/(2*sine(BeamAngle))
Y =  (BmVel[3] - BmVel[2])/(2*sine(BeamAngle))
Z = -(BmVel[0] + BmVel[1] + BmVel[2] + BmVel[3])/(4*cosine(BeamAngle))
Q =  (BmVel[0] + BmVel[1] - BmVel[2] - BmVel[3])/(4)
```

3. **E000003**
    a. Single Precision Floating Point
    b. Bins x Beams
        i. Earth Velocity in m/s
            1. Contains the Earth coordinate velocity profile. It is arranged E, N, U, Q for all bins where:

```
East  = X * (SH * CP) - Y * (CH * CR + SH * SR * SP) + Z * (CH * SR - SH * CR * SP);
North = X * (CH * CP) + Y * (SH * CR - CH * SR * SP) - Z * (SH * SR + CH * SP * CR);
Up    = X * (SP)      + Y * (SR * CP)                + Z * (CP * CR);
Q     = (BmVel[0] + BmVel[1] - BmVel[2] - BmVel[3])/(4)
```

Note: X, Y, Z are “bin mapped” prior to the Earth transformation. To bin map the beam data use the following code:

```
CP  = cosine(pitch)
CR  = cosine(roll)
SP  = sine(pitch)
SR  = sine(roll)
SH  = cosine(heading)
CH  = sine(roll)
CBA = cosine(BeamAngle)
SBA = sine(BeamAngle)
```

```
BeamAngle = 15, 20, or 30 degrees

//generate range scale factor for each beam
RS[0] = fabs(CBA / (CP * CR * CBA + SP * SBA));
RS[1] = fabs(CBA / (CP * CR * CBA - SP * SBA));
RS[2] = fabs(CBA / (CP * CR * CBA + SR * CP * SBA));
RS[3] = fabs(CBA / (CP * CR * CBA - SR * CP * SBA));

//generate the map
for(j=0;j<Bins;j++)
{
    for(i=0;i<4;i++)//beams
            BinMap[i][j] = (int)(j * RS[i]);
}

//map the velocity, correlation, and amplitude data
for(j=0;j<Bins;j++)
{
    for(i=0;i<4;i++)
    {
        if(BinMap[i][j] < Bins && BinMap[i][j] >= 0)
        {
            mvel[i][j] = Velocity[i][BinMap[i][j]];   // matrix E000001
            mamp[i][j] = Amplitude[i][BinMap[i][j]];  // matrix E000004
            mcor[i][j] = Correlation[i][BinMap[i][j]];// matrix E000005
        }
        else
        {
            RWP->mvel[i][j] = 0.0;
            RWP->mamp[i][j] = 0.0;
            RWP->mcor[i][j] = 0.0;
       }
    }
}

//transform mapped beam data to XYZ. Repeat for each bin
X =  (mvel[1] - mvel[0])/(2*sine(BeamAngle))
Y =  (mvel[3] - mvel[2])/(2*sine(BeamAngle))
Z = -(mvel[0] + mvel[1] + mvel[2] + mvel[3])/(4*cosine(BeamAngle))
Q =  (mvel[0] + mvel[1] - mvel[2] - mvel[3])/(4)
```

4. **E000004**
    a. Single Precision Floating Point
    b. Bins x Beams
        i. Amplitude in dB
            1. Contains the Beam Amplitude profile. It is arranged beam 0 to 3 for all bins.

5. **E000005**
    a. Single Precision Floating Point
    b. Bins x Beams
        i. Correlation where 1.0 = 100% and 0.5 = 50% correlation.
            1. Contains the Beam Correlation profile. It is arranged beam 0 to 3 for all bins.

6. **E000006**
    a. 32 bit Integer
    b. beams x bins
            1. Good Beam Pings

a. Contains the number of good pings for each bin/beam in the beam velocity data matrix E000001. It is arranged beam 0 to 3 for all bins.

7. **E000007**
   a. 32 bit Integer
   b. Bins x Beams
      1. Good Earth Pings
         a. Contains the number of good pings for each bin/beam in the ENUQ velocity data matrix E000003. It is arranged NG3, NG3, NG3, NG4 for each bin. Where: NG3 = the Number of Good 3 beam solutions and NG4 = the Number of Good 4 beam solutions in the bin.

Note: RTI systems can use 3 or 4 beams to solve for the Earth coordinate system. If 4 beam velocities are available then all 4 are used. If only 3 beams are available, perhaps due to a beam failure or a signal fade, the 4th beam velocity is calculated from the other 3 before calling the Earth transformation function (shown above). To calculate a 3 beam solution it is assumed that the beams are configured as a 4 beam Janus transducer and that the Q velocity on average is equal to 0.0 m/s.

Since

```
Q = (BM0 + BM1 – Bm2 – BM3) = 0.0
```

The "missing" beam solution is:

```
BM0 = -BM1 + BM2 + BM3;
BM1 = -BM0 + BM2 + BM3;
BM2 =  BM0 + BM1 - BM3;
BM3 =  BM0 + BM1 - BM2;
```

8. **E000008**
   a. 32 bit Integer
   b. 22 x 1
      i. Ensemble data
         1. Ensemble Number
         2. Bins
         3. Beams
         4. Pings in Ensemble (desired)
         5. Ping Count (actual)
         6. Status (see appendix B5 for description)
         7. Year
         8. Month
         9. Day
         10. Hour
         11. Minute
         12. Seconds
         13. 1/100 Seconds
         14. System Serial Number A (see appendix B4 for a description of the SN)

15. System Serial Number B
16. System Serial Number C
17. System Serial Number D
18. System Serial Number E
19. System Serial Number F
20. System Serial Number G
21. System Serial Number H
22. System Type and Firmware Version
    a. Most significant byte contains a code that describes the hardware Subsystem for this data set. (See description of this code in appendix B4).
    b. Next byte contains the Major firmware version.
    c. Next byte contains the Minor firmware version.
    d. Least significant byte contains the firmware Revision.
23. Subsystem Configuration
    a. Least significant byte contains a number that identifies which command setup is being used for the current subsystem. Use this byte along with the System Type byte to separate the ensembles for each setup/subsystem.

9. **E000009**
    a. Single Precision Floating Point
    b. 13 x 1
        i. Ancillary data (Contains additional Ensemble data)
            1. Range of First Bin from the transducer in meters
            2. Bin Size in meters
            3. First Profile Ping Time in seconds
            4. Last Profile Ping Time in seconds
            5. Heading in degrees
            6. Pitch in degrees
            7. Roll degrees
            8. Water Temperature in degrees (used in SOS)
            9. System Temperature in degrees
            10. Salinity in parts per thousand (ppt) (used in SOS)
            11. Pressure in bar
            12. Depth in meters (used in SOS)
            13. Speed Of Sound in m/s

18. **E000010**
    a. Single Precision Floating Point
    b. 54 x 1
        i. Bottom Track data
            1. First Bottom Track Ping Time in seconds
            2. Last Bottom Track Ping Time in seconds
            3. Heading in degrees
            4. Pitch in degrees
            5. Roll in degrees
            6. Water Temperature in degrees (used in SOS)
            7. System Temperature in degrees
            8. Salinity in parts per thousand (ppt) (used SOS)

9. Pressure in bar
10. Depth in meters (used in SOS)
11. Speed Of Sound in m/s
12. Status (final value is logical OR of each bit below)
    i. Good Status 0x0000
        a. Water Track 3 Beam solution 0x0001 (DVL only)
            i. 3 of 4 beams have a valid signal
        b. Bottom Track 3 Beam Solution 0x0002
            i. 3 of 4 beams located the bottom
        c. Bottom Track Hold (not searching yet) 0x0004
            i. Holding the search to last known depth
        d. Bottom Track Searching 0x0008
            i. Actively searching for the bottom
        e. Hardware Timeout 0x8000
            i. The hardware did not respond to the ping request
13. Number of Beams
14. Ping Count
15. Range_0 (Beam 0 vertical range in meters)
16. Range_1 (Beam 1 vertical range in meters)
17. Range_2 (Beam 2 vertical range in meters)
18. Range_3 (Beam 3 vertical range in meters)
19. SNR_0 (Beam 0 Signal to Noise in dB)
20. SNR_1 (Beam 1 Signal to Noise in dB)
21. SNR_2 (Beam 2 Signal to Noise in dB)
22. SNR_3 (Beam 3 Signal to Noise in dB)
23. Amplitude_0 (Beam 0 Bottom Amplitude in dB)
24. Amplitude_1 (Beam 1 Bottom Amplitude in dB)
25. Amplitude_2 (Beam 2 Bottom Amplitude in dB)
26. Amplitude_3 (Beam 3 Bottom Amplitude in dB)
27. Correlation_0 (Beam 0 Correlation (0.5 = 50%))
28. Correlation_1 (Beam 1 Correlation (0.5 = 50%))
29. Correlation_2 (Beam 2 Correlation (0.5 = 50%))
30. Correlation_3 (Beam 3 Correlation (0.5 = 50%))
31. Velocity_0 (Beam 0 Velocity in m/s)
32. Velocity_1 (Beam 1 Velocity in m/s)
33. Velocity_2 (Beam 2 Velocity in m/s)
34. Velocity_3 (Beam 3 Velocity in m/s)
35. BeamN_0 (Beam 0 Number of pings averaged)
36. BeamN_1 (Beam 1 Number of pings averaged)
37. BeamN_2 (Beam 2 Number of pings averaged)
38. BeamN_3 (Beam 3 Number of pings averaged)
39. Instrument_0 (X velocity in m/s)
40. Instrument_1 (Y velocity in m/s)
41. Instrument_2 (Z velocity in m/s)
42. Instrument_3 (Q velocity in m/s)
43. XfrmN_0 (Number of 3 beam solutions averaged)
44. XfrmN_1 (Number of 3 beam solutions averaged)
45. XfrmN_2 (Number of 3 beam solutions averaged)

46. XfrmN_3 (Number of 4 beam solutions averaged)
47. Earth_0 (East velocity in m/s)
48. Earth_1 (North velocity in m/s)
49. Earth_2 (Up velocity in m/s)
50. Earth_3 (Q velocity in m/s)
51. EarthN_0 (Number of 3 beam solutions averaged)
52. EarthN_1 (Number of 3 beam solutions averaged)
53. EarthN_2 (Number of 3 beam solutions averaged)
54. EarthN_3 (Number of 4 beam solutions averaged)

19. **E000011**
    a. 8 bit Byte
    b. n x 1
        i. External NMEA ASCII collected during ping
            1. Contains captured serial NMEA data that arrived during the ensemble. One 8 bit byte per ASCII character. Maximum number of bytes = 8192.

20. **E000012**
    a. Single Precision Floating Point
    b. 23 x 1
        i. Profile Engineering data contains data that describes the Profile ping setup.
            1. PrePingVel[0]
            2. PrePingVel[1]
            3. PrePingVel[2]
            4. PrePingVel[3]
            5. PrePingCor[0]
            6. PrePingCor[1]
            7. PrePingCor[2]
            8. PrePingCor[3]
            9. PrePingAmp[0]
            10. PrePingAmp[1]
            11. PrePingAmp[2]
            12. PrePingAmp[3]
            13. SamplesPerSecond
            14. SystemFreqHz
            15. LagSamples
            16. CPCE
            17. NCE
            18. RepeatN
            19. PrePingGap
            20. PrePingNCE
            21. PrePingRepeatN
            22. PrePingLagSamples
            23. TRhighGain
21. **E000013**
    a. Single Precision Floating Point
    b. 30 x 1
        i. Bottom Track Engineering data contains data that describes the bottom track ping setup.
            1. SamplesPerSecond
            2. SystemFreqHz
            3. LagSamples
            4. CPCE
            5. NCE
            6. RepeatN
            7. AmbHz[0]
            8. AmbHz[1]
            9. AmbHz[2]
            10. AmbHz[3]
            11. AmbVel[0]
            12. AmbVel[1]
            13. AmbVel[2]
            14. AmbVel[3]
            15. AmbAmp[0]
            16. AmbAmp[1]

17. AmbAmp[2]
18. AmbAmp[3]
19. AmbCor[0]
20. AmbCor[1]
21. AmbCor[2]
22. AmbCor[3]
23. AmbSNR[0]
24. AmbSNR[1]
25. AmbSNR[2]
26. AmbSNR[3]
27. LagUsed[0]
28. LagUsed[1]
29. LagUsed[2]
30. LagUsed[3]

B2 C# Binary Decoding Example

```
int PacketPointer = 0;

int ByteArrayToInt(byte[] packet)
{
    ByteArrayToNumber.A = packet[PacketPointer++];
    ByteArrayToNumber.B = packet[PacketPointer++];
    ByteArrayToNumber.C = packet[PacketPointer++];
    ByteArrayToNumber.D = packet[PacketPointer++];

    return ByteArrayToNumber.Int;
}
string ByteArrayToString(byte[] packet, int len)
{
    string s = "";
    int i;
    for (i = 0; i < len; i++)
    {
        s += (char)packet[PacketPointer++];
    }
    return s;
}
float ByteArrayToFloat(byte[] packet)
{
    ByteArrayToNumber.A = packet[PacketPointer++];
    ByteArrayToNumber.B = packet[PacketPointer++];
    ByteArrayToNumber.C = packet[PacketPointer++];
    ByteArrayToNumber.D = packet[PacketPointer++];
    return ByteArrayToNumber.Float;
}

string VelocityID     = "E000001\0";
string InstrumentID   = "E000002\0";
string EarthID        = "E000003\0";
string AmplitudeID    = "E000004\0";
string CorrelationID  = "E000005\0";
string BeamNID        = "E000006\0";
string XfrmNID        = "E000007\0";
string EnsembleDataID = "E000008\0";
string AncillaryID    = "E000009\0";
string BottomTrackID  = "E000010\0";
string NMEAID         = "E000011\0";

void DecodeEnsemble(byte[] packet, ArrayClass m)
{
    int i = 0;
    int SizeCount = 0;
    int ArrayCount = 0;

    PacketPointer = HDRLEN;
    m.VelocityAvailable = false;
    m.InstrumentAvailable = false;
    m.EarthAvailable = false;
    m.AmplitudeAvailable = false;
    m.CorrelationAvailable = false;
    m.BeamNAvailable = false;
    m.XfrmNAvailable = false;
    m.EnsembleDataAvailable = false;
    m.AncillaryAvailable = false;
    m.BottomTrackAvailable = false;
    m.NmeaAvailable = false;

    for (i = 0; i < MaxArray; i++)
    {
        m.Type[i]    = ByteArrayToInt(packet);
        m.Bins[i]    = ByteArrayToInt(packet);
```

```
m.Beams[i]   = ByteArrayToInt(packet);
m.Imag[i]    = ByteArrayToInt(packet);
m.NameLen[i] = ByteArrayToInt(packet);
m.Name[i]    = ByteArrayToString(packet, 8);

ArrayCount = m.Bins[i] * m.Beams[i];
SizeCount = PacketPointer;

if (VelocityID.Equals(m.Name[i], StringComparison.Ordinal))
{
    m.VelocityAvailable = true;
    for (int beam = 0; beam < m.Beams[i]; beam++)
    {
        for (int bin = 0; bin < m.Bins[i]; bin++)
        {
            m.Velocity[beam, bin] = ByteArrayToFloat(packet);
        }
    }
}
else
{
if (InstrumentID.Equals(m.Name[i], StringComparison.Ordinal))
{
    m.InstrumentAvailable = true;

    for (int beam = 0; beam < m.Beams[i]; beam++)
    {
        for (int bin = 0; bin < m.Bins[i]; bin++)
        {
            m.Instrument[beam, bin] = ByteArrayToFloat(packet);
        }
    }
}
else
{
if (EarthID.Equals(m.Name[i], StringComparison.Ordinal))
{
    m.EarthAvailable = true;
    for (int beam = 0; beam < m.Beams[i]; beam++)
    {
        for (int bin = 0; bin < m.Bins[i]; bin++)
        {
            m.Earth[beam, bin] = ByteArrayToFloat(packet);
        }
    }
}
else
{
if (AmplitudeID.Equals(m.Name[i], StringComparison.Ordinal))
{
    m.AmplitudeAvailable = true;
    for (int beam = 0; beam < m.Beams[i]; beam++)
    {
        for (int bin = 0; bin < m.Bins[i]; bin++)
        {
            m.Amplitude[beam, bin] = ByteArrayToFloat(packet);
        }
    }
}
else
{
if (CorrelationID.Equals(m.Name[i], StringComparison.Ordinal))
{
    m.CorrelationAvailable = true;
    for (int beam = 0; beam < m.Beams[i]; beam++)
    {
        for (int bin = 0; bin < m.Bins[i]; bin++)
        {
            m.Correlation[beam, bin] = ByteArrayToFloat(packet);
        }
    }
```

```
}
else
{
if (BeamNID.Equals(m.Name[i], StringComparison.Ordinal))
{
    m.BeamNAvailable = true;
    for (int beam = 0; beam < m.Beams[i]; beam++)
    {
        for (int bin = 0; bin < m.Bins[i]; bin++)
        {
            m.BeamN[beam, bin] = ByteArrayToInt(packet);
        }
    }
}
else
{
if (XfrmNID.Equals(m.Name[i], StringComparison.Ordinal))
{
    m.XfrmNAvailable = true;
    for (int beam = 0; beam < m.Beams[i]; beam++)
    {
        for (int bin = 0; bin < m.Bins[i]; bin++)
        {
            m.XfrmN[beam, bin] = ByteArrayToInt(packet);
        }
    }
}
else
{
if (EnsembleDataID.Equals(m.Name[i], StringComparison.Ordinal))
{
    m.EnsembleDataAvailable = true;
    m.E_EnsembleNumber = ByteArrayToInt(packet);
    m.E_Cells = ByteArrayToInt(packet);
    m.E_Beams = ByteArrayToInt(packet);
    m.E_PingsInEnsemble = ByteArrayToInt(packet);
    m.E_PingCount = ByteArrayToInt(packet);
    m.E_Status = ByteArrayToInt(packet);
    m.E_Year = ByteArrayToInt(packet);
    m.E_Month = ByteArrayToInt(packet);
    m.E_Day = ByteArrayToInt(packet);
    m.E_Hour = ByteArrayToInt(packet);
    m.E_Minute = ByteArrayToInt(packet);
    m.E_Second = ByteArrayToInt(packet);
    m.E_Hsec = ByteArrayToInt(packet);
    for (i =0; i<32; i++)
    {
      if (packet[PacketPointer] > 31 && packet[PacketPointer] < 127)
        m.E_SN_Buffer[i] = packet[PacketPointer];
      else
        m.E_SN_Buffer[i] = 63;
      PacketPointer++;
    }
    m.E_SN_Buffer[i] = 0;
    for (i = 0; i < 4; i++)
    {
      m.E_FW_Vers[i] = packet[PacketPointer];
      PacketPointer++;
    }
}
else
{
if (AncillaryID.Equals(m.Name[i], StringComparison.Ordinal))
{
    m.AncillaryAvailable = true;
    m.A_FirstCellDepth = ByteArrayToFloat(packet);
    m.A_CellSize = ByteArrayToFloat(packet);
    m.A_FirstPingSeconds = ByteArrayToFloat(packet);
    m.A_LastPingSeconds = ByteArrayToFloat(packet);

    m.A_Heading = ByteArrayToFloat(packet);
```

```
        m.A_Pitch = ByteArrayToFloat(packet);
        m.A_Roll = ByteArrayToFloat(packet);
        m.A_WaterTemperature = ByteArrayToFloat(packet);
        m.A_BoardTemperature = ByteArrayToFloat(packet);
        m.A_Salinity = ByteArrayToFloat(packet);
        m.A_Pressure = ByteArrayToFloat(packet);
        m.A_Depth = ByteArrayToFloat(packet);
        m.A_SpeedOfSound = ByteArrayToFloat(packet);
    }
    else
    {
    if (BottomTrackID.Equals(m.Name[i], StringComparison.Ordinal))
    {
        m.BottomTrackAvailable = true;
        m.B_FirstPingSeconds = ByteArrayToFloat(packet);
        m.B_LastPingSeconds = ByteArrayToFloat(packet);

        m.B_Heading = ByteArrayToFloat(packet);
        m.B_Pitch = ByteArrayToFloat(packet);
        m.B_Roll = ByteArrayToFloat(packet);
        m.B_WaterTemperature = ByteArrayToFloat(packet);
        m.B_BoardTemperature = ByteArrayToFloat(packet);
        m.B_Salinity = ByteArrayToFloat(packet);
        m.B_Pressure = ByteArrayToFloat(packet);
        m.B_Depth = ByteArrayToFloat(packet);
        m.B_SpeedOfSound = ByteArrayToFloat(packet);

        m.B_Status = ByteArrayToFloat(packet);
        m.B_Beams = ByteArrayToFloat(packet);
        m.B_PingCount = ByteArrayToFloat(packet);

        for (int beam = 0; beam < m.B_Beams; beam++)
            m.B_Range[beam] = ByteArrayToFloat(packet);
        for (int beam = 0; beam < m.B_Beams; beam++)
            m.B_SNR[beam] = ByteArrayToFloat(packet);
        for (int beam = 0; beam < m.B_Beams; beam++)
            m.B_Amplitude[beam] = ByteArrayToFloat(packet);
        for (int beam = 0; beam < m.B_Beams; beam++)
            m.B_Correlation[beam] = ByteArrayToFloat(packet);
        for (int beam = 0; beam < m.B_Beams; beam++)
            m.B_Velocity[beam] = ByteArrayToFloat(packet);
        for (int beam = 0; beam < m.B_Beams; beam++)
            m.B_BeamN[beam] = ByteArrayToFloat(packet);

        for (int beam = 0; beam < m.B_Beams; beam++)
            m.B_Instrument[beam] = ByteArrayToFloat(packet);
        for (int beam = 0; beam < m.B_Beams; beam++)
            m.B_XfrmN[beam] = ByteArrayToFloat(packet);
        for (int beam = 0; beam < m.B_Beams; beam++)
            m.B_Earth[beam] = ByteArrayToFloat(packet);
        for (int beam = 0; beam < m.B_Beams; beam++)
            m.B_EarthN[beam] = ByteArrayToFloat(packet);
    }
    else
    {
    if (NMEAID.Equals(m.Name[i], StringComparison.Ordinal))
    {
        m.NmeaAvailable = true;
        i=0;
        while(packet[PacketPointer] != 0)
        {
            m.NMEA_Buffer[i++] = packet[PacketPointer++];
            if (i >= 8192)
                break;
        }
        for (int end = i; end < 8192; end++)
        {
            m.NMEA_Buffer[end] = 0;
        }
    }
    }
```

```
        }
        }
        }
        }
        }
        }
        }
        }
        }
        //correct for changes in array length
        SizeCount = (PacketPointer - SizeCount) / 4;

        if (SizeCount != ArrayCount)
        {
            PacketPointer += 4 * (ArrayCount - SizeCount);
        }
        if (PacketPointer + 4 >= PacketSize)
            break;//no more data
    }
    nArray = i;
    if (m.E_Cells < 1)
        m.E_Cells = 1;
```

B2 C# Checksum Calculation

The following C# code calculates the checksum for the RTI ensemble:

```
const int HDRLEN = 32;
long csum = 0;

long EnsembleSize  = DataBuff [24];
      EnsembleSize += DataBuff [25] << 8;
      EnsembleSize += DataBuff [26] << 16;
      EnsembleSize += DataBuff [27] << 24;

 //CCITT 16 bit algorithm (X^16 + X^12 + X^5 + 1)

 ushort crc = 0; //seed = 0
 for (i = HDRLEN; i < EnsembleSize + HDRLEN; i++)
 {
    crc = (ushort)((byte)(crc >> 8) | (crc << 8));
    crc ^= DataBuff[i];
    crc ^= (byte)((crc & 0xff) >> 4);
    crc ^= (ushort)((crc << 8) << 4);
    crc ^= (ushort)(((crc & 0xff) << 4) << 1);
 }
 ushort csum = crc;
```

B3 C# Binary Data Packet

1. Ensemble Binary Data Packet
    a. Consists of 3 sections
        i. 32 byte Header
            1. 16 bytes containing 0x80
            2. 4 byte Ensemble number
            3. 4 byte Ensemble number ones compliment
            4. 4 byte payload size (in bytes)
            5. 4 byte payload size ones compliment
        ii. Payload
            1. Version 4 MAT-File Format (see matlab4.pdf for details)
                a. A MAT-file may contain one or more matrices. The matrices are written sequentially to a file, with the bytes forming a continuous stream. Each matrix starts with a fixed-length 20-byte header that contains information describing certain attributes of the Matrix. The 20-byte header consists of five long (4-byte) integers.
            2. Velocity, Amplitude, and Statistical data are contained in bins x 4 arrays (row x column).
            3. Ancillary data such as Time, Heading, Pitch, Roll and Temperature are contained in an n x 1 array. Where n is the number of rows.
        iii. CRC-16 Checksum
            1. 4 byte checksum of all the bytes in the payload
            2. First 2 bytes are always 0
            3. The second 2 bytes contain the CRC-16 value
            4. CCITT 16 bit algorithm (X^16 + X^12 + X^5 + 1)
            5. CRC seed = 0

B4 System Serial Number and Subsystem Codes

1. The system serial number is 32 bytes in length
    a. SNXX0123456789ABCDEabcdefghi123456
        i. The first 2 bytes after the SN (XX) are combined to form an integer number. This number describes the base electronics hardware architecture.
        ii. The next 15 bytes (0 - E) represent the 15 possible different sub systems.
            1. Each of the 15 digits 0 through E contains a code (0 - Z) that describes a subsystem. See list below for a description of the codes.
            2. Subsystem references are made by using the corresponding character 0 – F. For example: the command CWPP[3] 10<CR> sets the number of water track pings for subsystem 3 to 10 pings. Subsystem 3 is the 4th subsystem contained in the SN.
        iii. The next 9 bytes (a – i) are spares.
        iv. The last 6 digits (1 - 6) are the system serial number.
    b. Example 1:
        i. SN013000000000000000000000000000001
            1. A 600 kHz 4 beam 20 degree piston, serial number 1
    c. Example 2:
        i. SN013400000000000000000000000000001
            1. Dual frequency 600/300 kHz 4 beam 20 degree piston, serial number 1
    d. Example 3:
        i. SN01K000000000000000000000000000001
            1. A 150 kHz 4 beam 30 degree array, serial number 1
    e. Example 4:
        i. SN01KBXV00000000000000000000000001
            1. A dual 150/600 kHz 4 beam 30 degree array
            2. A dual 75/300 kHz 4 beam 15 degree array
            3. Serial number 1

```
Subsystem Codes:
code  description
0     spare
1     2 MHz 4 beam 20 degree piston
2     1.2 MHz
3     600 kHz
4     300 kHz
5     2 MHz 4 beam 20 degree piston, 45 degree heading offset
6     1.2 MHz
7     600 kHz
8     300 kHz
9     2 MHz vertical beam piston
A     1.2 MHz
B     600 kHz
C     300 kHz
D     150 kHz
E      75 kHz
F      38 kHz
G      20 kHz
H     spare
I     600 kHz 4 beam 30 deg array
J     300 kHz
K     150 kHz
L      75 kHz
M      38 kHz
N      20 kHz
```

```
O     600 kHz 4 beam 15 deg array
P     300 kHz
Q     150 kHz
R      75 kHz
S      38 kHz
T      20 kHz
U     600 kHz 1 beam 0 deg array
V     300 kHz
W     150 kHz
X      75 kHz
Y      38 kHz
Z      20 kHz
a     spare
b     spare
      etc...
```

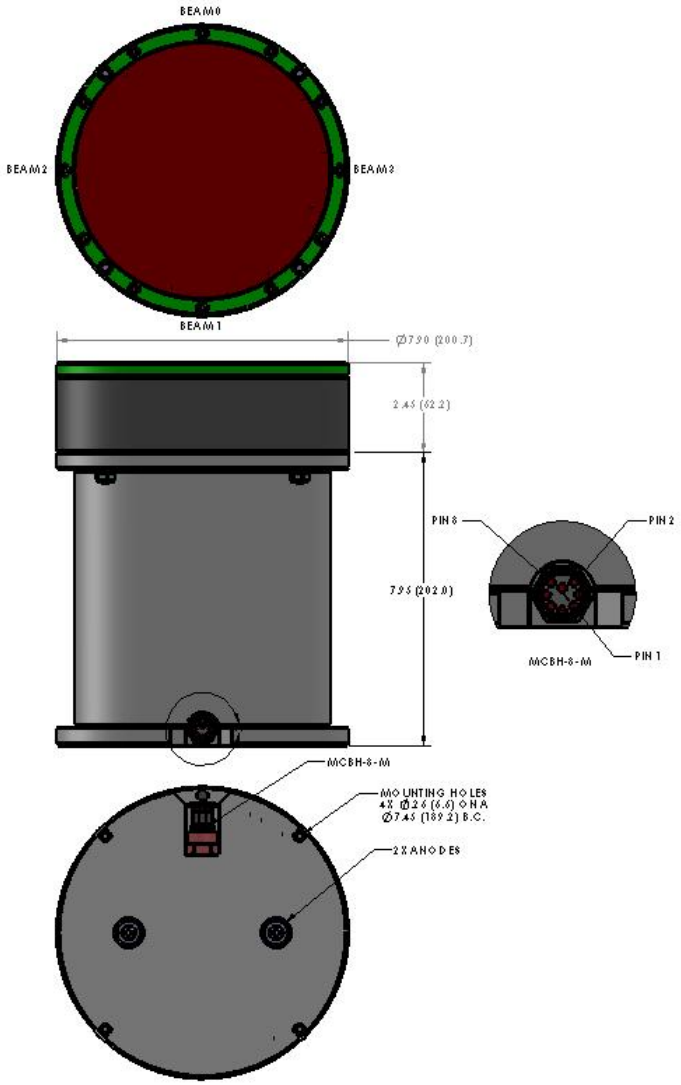
B5 System Status Word

The System Status word is output with both the NMEA and Binary data structures. The final value is the logical OR of each bit shown below:

1. 0x0000 Good Status
2. 0x0001 Water track (NMEA output only) 3 beam solution
    a. Indicates the water track velocity output contains a 3 beam solution.
3. 0x0002 Bottom track (NMEA output only) 3 beam solution
    a. Indicates the bottom track velocity output contains a 3 beam solution.
4. 0x0004 Bottom track hold
    a. Indicates bottom track did not detect the bottom but is still using the last good detection as an estimate of the bottom location.
5. 0x0008 Bottom Track Search
    a. Indicates bottom track is actively changing the ping settings to attempt bottom detection.
6. 0x0010 Bottom Track Proof
    a. Indicates bottom track is waiting for the next valid bottom track ping before allowing velocity to be output.
7. 0x0100 Heading sensor Error
8. 0x0200 Pressure Sensor Error
9. 0x0400 Power Down failure
    a. System power did not shut off between ensembles.
10. 0x0800 Non Volatile data error
    a. Non volatile memory storage checksum failed.
11. 0x1000 Real Time Clock (RTC) error
    a. The RTC did not respond or the time data value contained illegal values i.e. month = 13.
12. 0x2000 Temperature sensor Error
    a. The temperature sensor ADC did not respond or the temperature value was out of range (-30 to 70 C).
13. 0x4000 Receiver data error
    a. The receiver did not output the expected amount of data.
14. 0x8000 Receiver Timeout
    a. The receiver hardware did not respond to the ping request.

Appendix C

C1 DA 150 Outline Drawings



Weight in Air: 9.0 kg

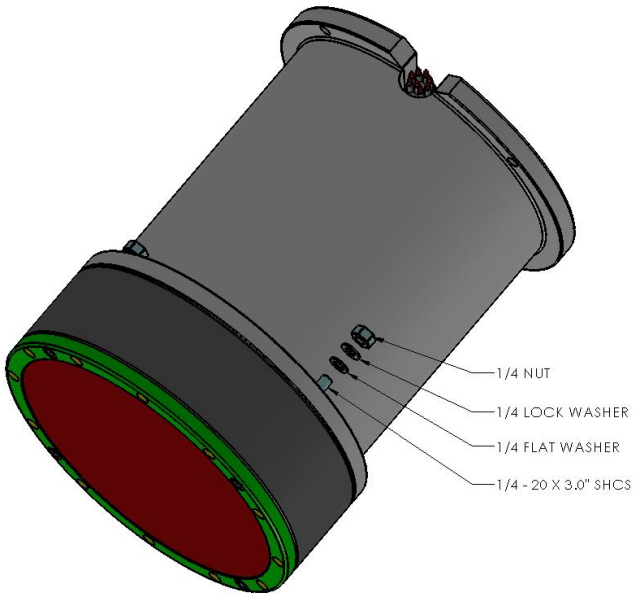
Weight in Water: 1.5 kg

Housing Material: Copolymer Acetal & 6061 T6 ALUMINUM

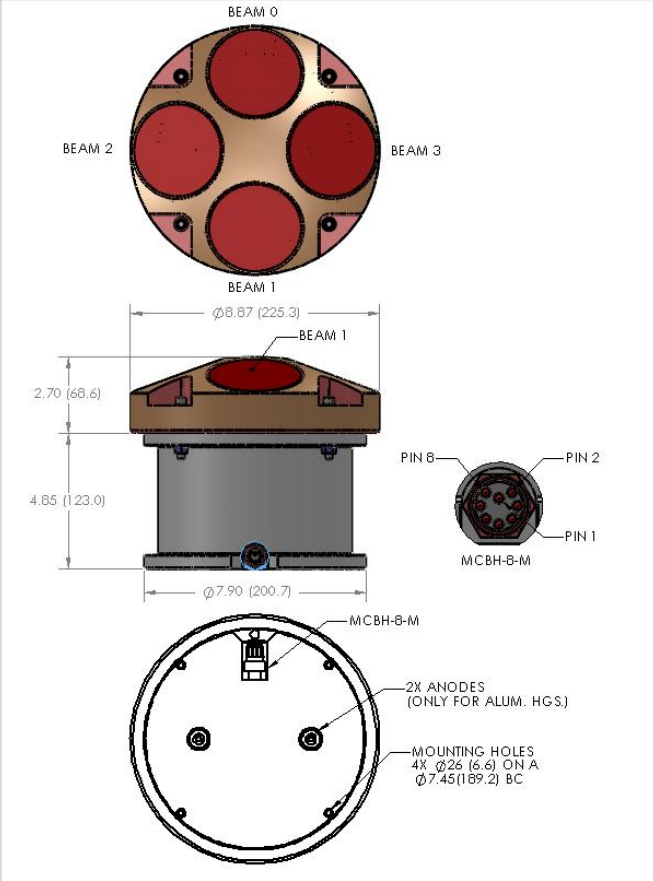
Transducer Face: Urethane

Pressure Rating : 800m

C2 DA 150 Close up Hardware



C3 DP 300/600 Outline Drawings



Weight in Air: 7.5 kg

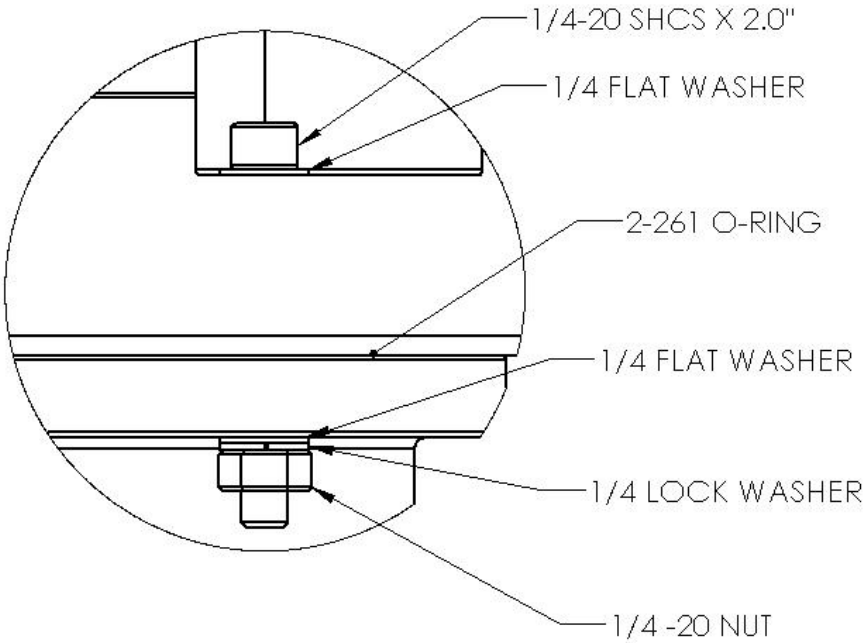
Weight in Water: 1.5 kg

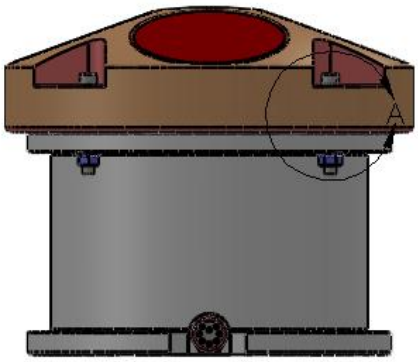
Housing Material: Copolymer Acetal

Transducer Face: Urethane

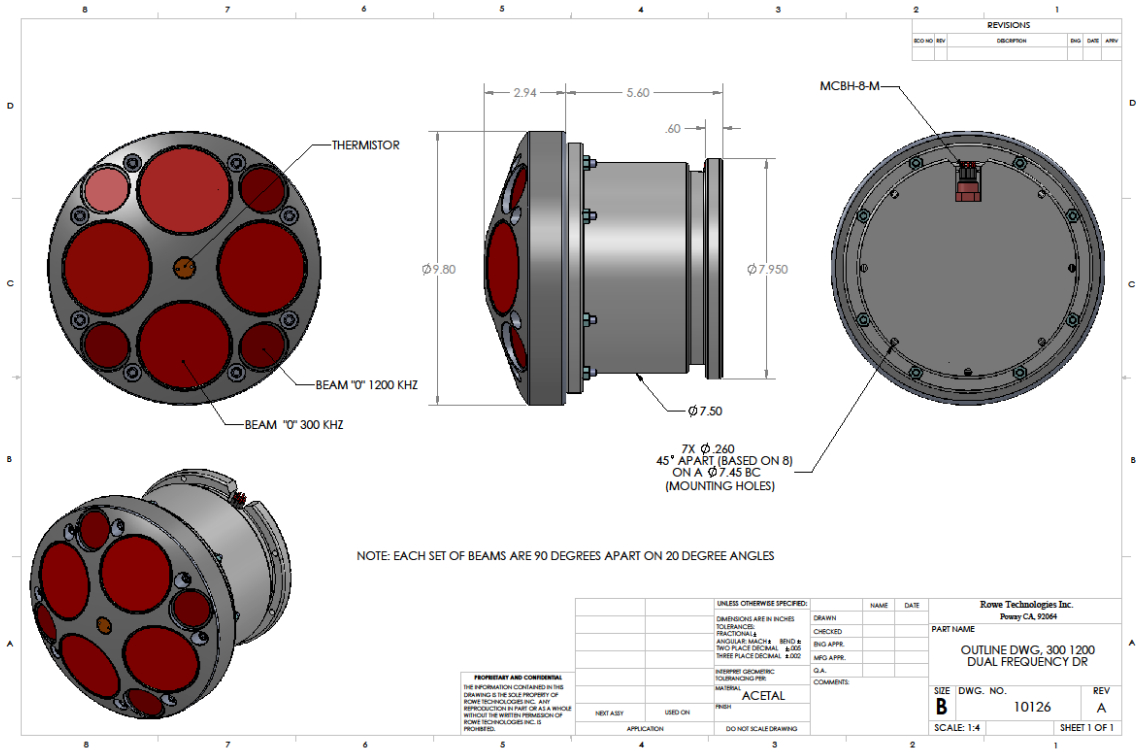
Pressure Rating : 200m

C4 DP 300/600 Close up Hardware

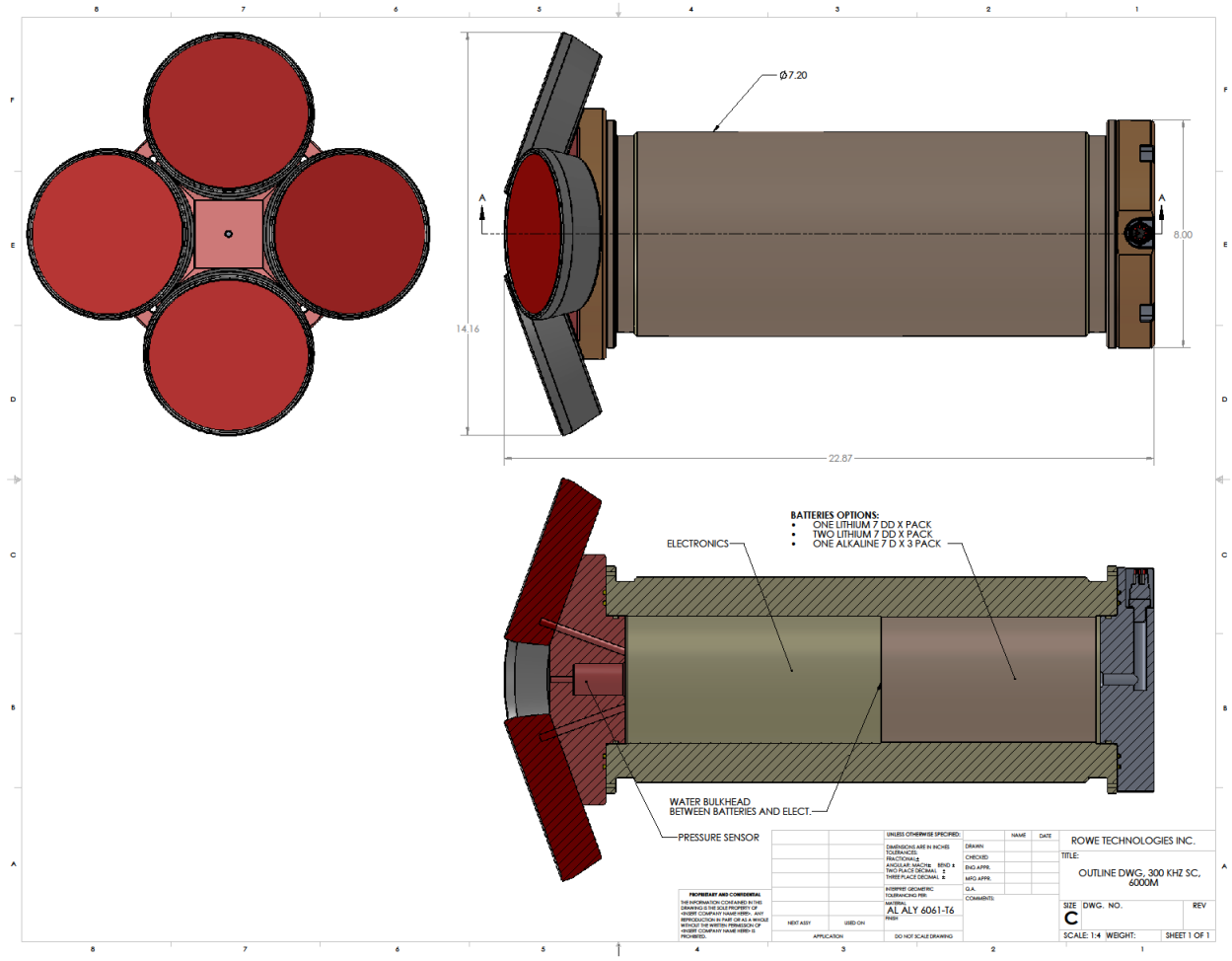




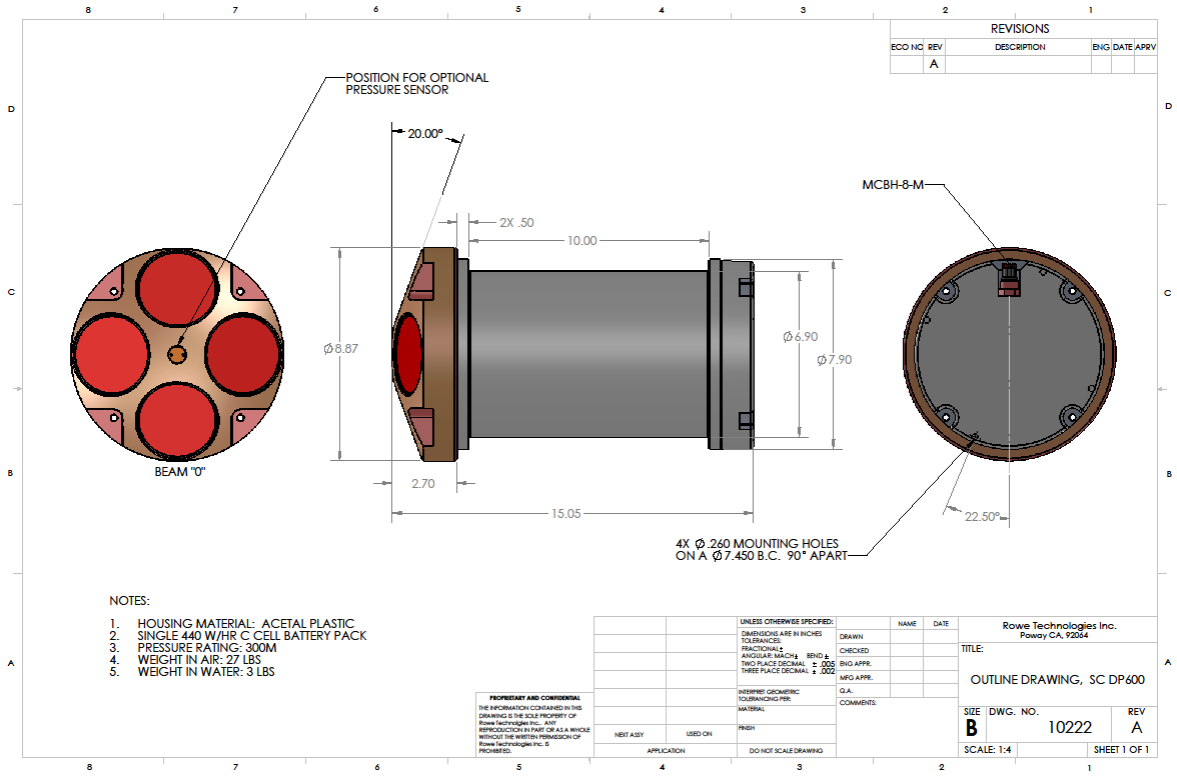
C5 Dual Frequency DP 300/1200 Outline Drawing



C6 300 KHZ SC 6000M ASSY DWG

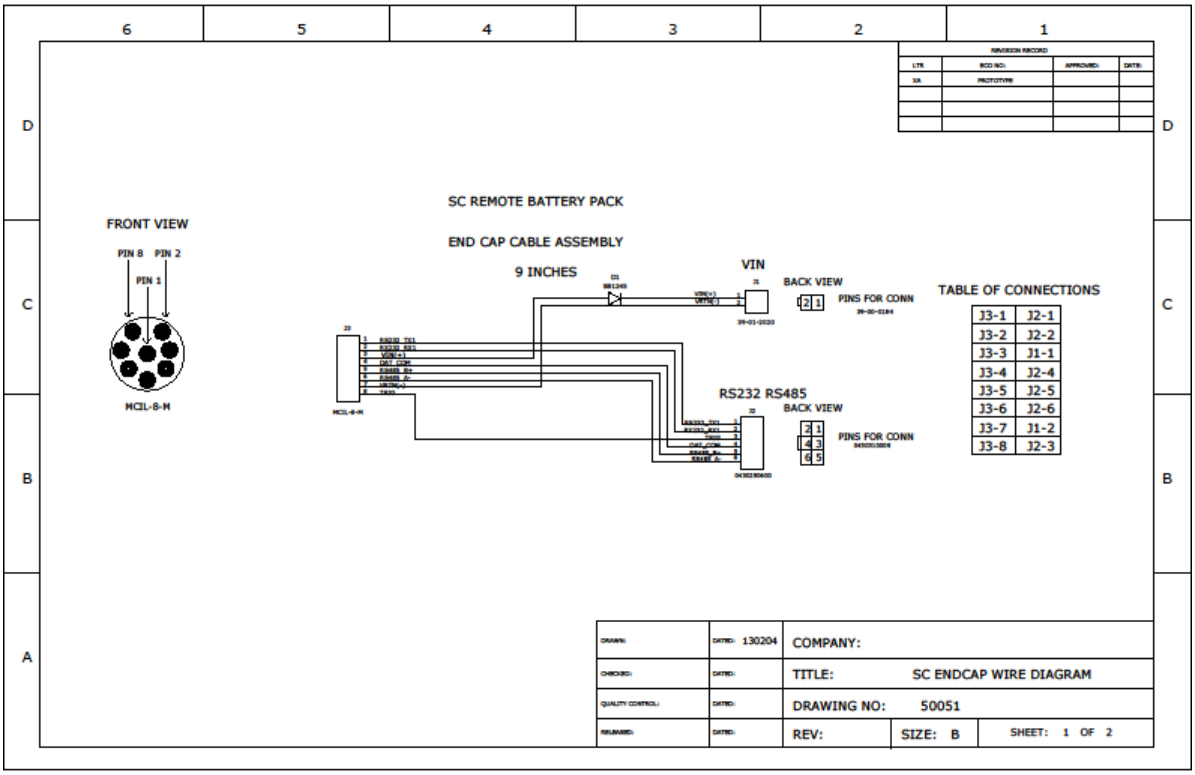


C6 SC 600 COASTAL ASSY DWG

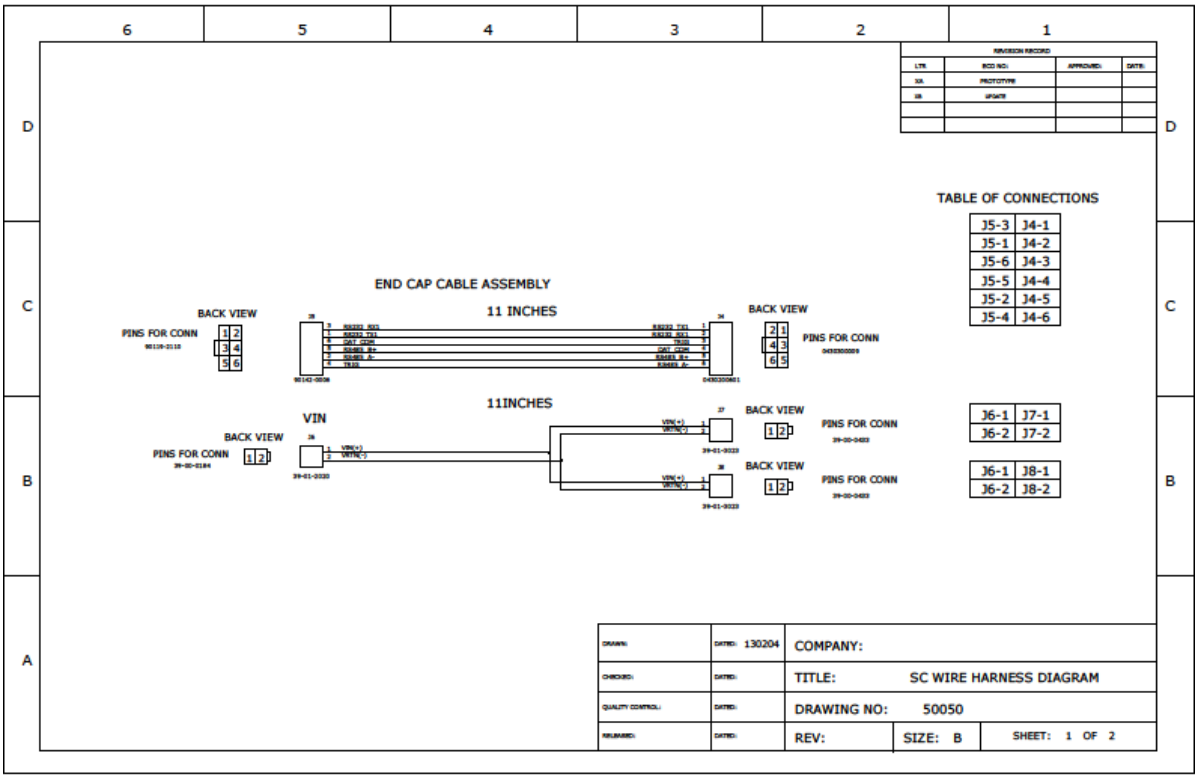


Appendix D

D1 SELF-CONTAINED ENDCAP WIRE DIAGRAM



D2 SELF-CONTAINED WIRE HARNESS DIAGRAM



D3 SELF-CONTAINED ENDCAP BATTERY ASSY