Wave Glider Sensor Integration User Guide

For Teledyne Benthos Acoustic Modem

P/N 02731-02

Change Record

Date	Description of change
8/24/2011	Initial version

Table of Contents

1. System Overview

1.1 Product Overview

The Liquid Robotics integration of Teledyne Benthos acoustic modems provides a mobile, long endurance platform for transmitting and receiving data from sea floor sensors that incorporate the Teledyne Benthos communication solution.

The modem interface software, called BenthosModem, resides on the LRI Sensor Management Computer (SMC). The access manager (BAM) manages the communication session with the float-mounted modem and can act as a conduit for local applications to interact with sea floor resident sensors. The command set is accessible via CORBA messages, and pass-through data is sent via a pseudo port.

The Linux-based Sensor Management Computer (SMC) can host applications written in C, C++, Java and a variety of scripting languages. Data can be sent to shore using standard Wave Glider communications channels as well as payload-resident modems. The BAM application software may be reconfigured dynamically to reflect changing mission objectives and constraints.

This manual primarily focuses on the LRI integration specific aspects of the product, such as mechanical integration and the command set available for controlling BAM. It assumes a working familiarity with the features and operation of Benthos modems.

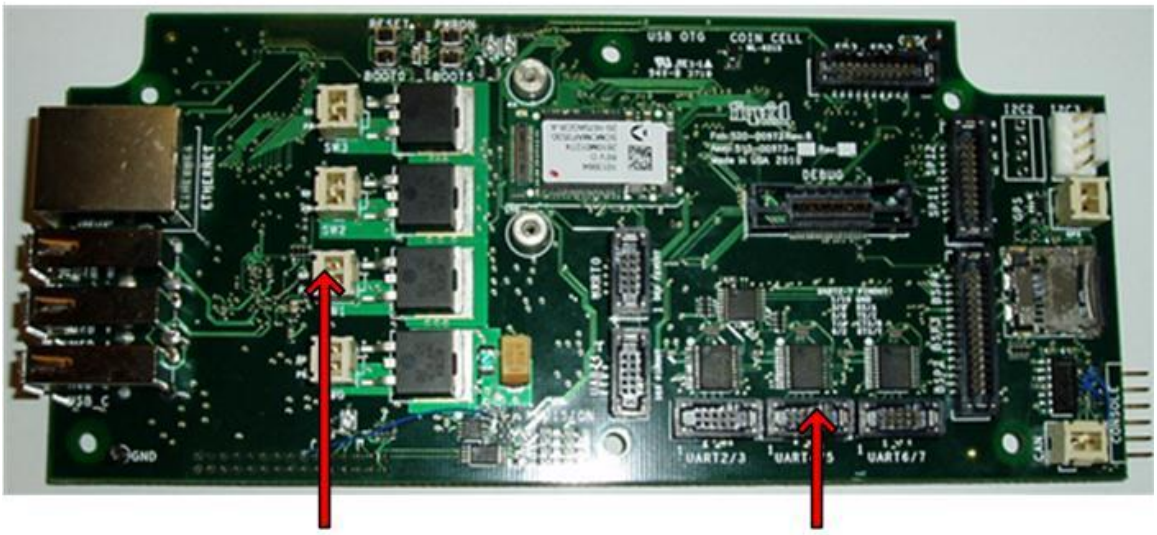
1.2 Mechanical Architecture

The modem may be mounted in a variety of locations on the Wave Glider. The most common locations are through the hull of the float or in the Liquid Robotics configurable tow body. The tow body integration is shown. Float integration is dependent on considerations beyond the scope of this document, such as other sensors mounted on the Wave Glider's float.

1.3 Electronic Architecture

The modem is controlled by software running on the Sensor Management Computer. Power is provided by a high-voltage board and is switched by one of the power output ports on the SMC; the RS-232 connection to the float-based modem is provided by one of the serial ports on the SMC. The sea-floor based modem and sensors typically operate on battery power.

Sensor Management Computer



Cables are included with the product if the integration is performed by Liquid Robotics.

1.4 Software Architecture

Broadly, data flows through the paths as shown in the image Modem Control and Data Flow and Data Flow.

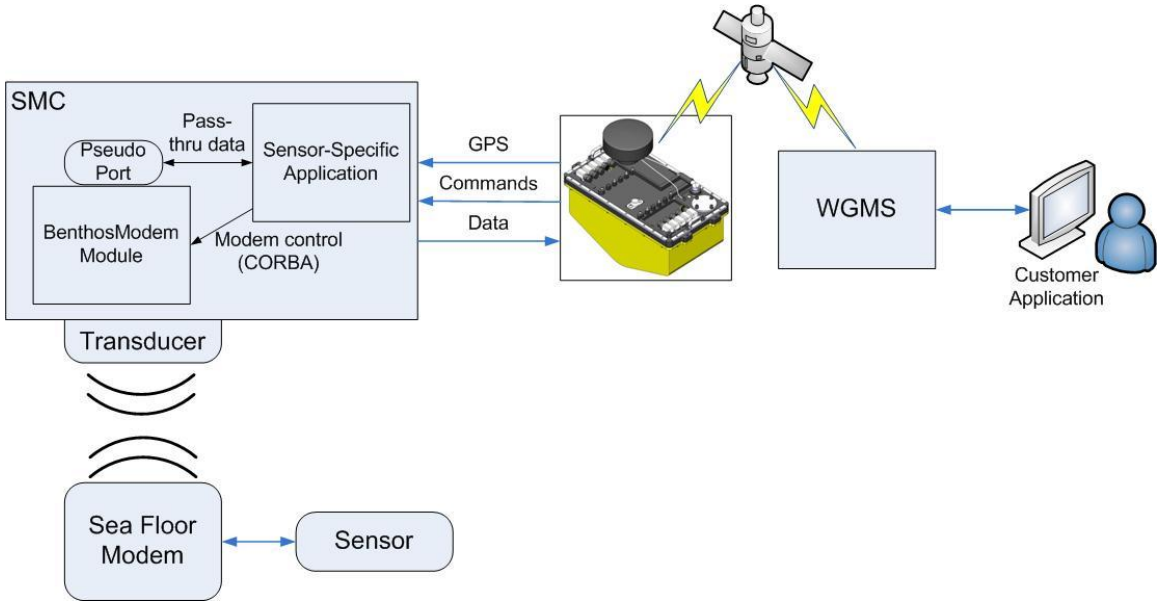


Figure 1 Modem Control and Data Flow

The BenthosModem module exposes the ICommandable interface via CORBA and accepts commands control power to the modem and to issue modem control commands. Sensor-specific commands and responses, as well as asynchronous reports from the sensor, are passed via a pseudo-port. Once connection is established with the sea floor modem, applications that work when directly connected typically work via this pseudo port, as long as they can be configured to work with the additional latency.

Starting from the right, the user plans his use of the modem and the interconnected sensor using appropriate resources, such as the Benthos and sensor manufacturer documentation. He then uses WGMS to send commands to the BenthosModem module and sensor-specific application programs running on the SMC. These can be individual commands or a file of commands (a script). BenthosModem module commands control modem power, test range and communication quality with the target sea floor modem or interact with the sea floor sensor. Commands also can be issued from a shell script running on the SMC, using the LRI ServerExec command.

In some cases, there is no sensor-specific application on the SMC. Sensor data is simply recorded by the sea floor modem and is periodically collected and returned to WGMS.

To facilitate location of the sea floor modem, the BenthosModem module can be placed into a mode where it continually performs communication quality tests (ATY) and range checking (ATR) with a target modem and returns it to shore as payload telemetry. The user navigates and tunes parameters to achieve optimum performance before attempting to transfer data.

1.5 Related Documents

- Teledyne Benthos ATM-900 Series Acoustic Telemetry Modems, User's Manual, P/N M-270-26, Rev. A
- WET Labs ECO Three-measurement Sensor User's Guide, Revision F, 7 May 2009
- Interface Control Description, version 2.00, Liquid Robotics P/N 030-00026-02
- Sensor Management Computer Programmer's Guide, LRI, ??
- Wave Glider Electrical Interfaces Guide, Liquid Robotics P/N 030-01676

1.6 Glossary

BAM	Benthos Access Manager (i.e., the BenthosModem process on the SMC)
C&C	The Wave Glider's Command and Control module.
Board Address	A 16-bit number with the upper 8 bits allocated as a Board ID and the lower 8 bits allocated as the Task ID. The Board Address uniquely identifies an entity that might send or receive commands and data. A unique Board ID is assigned to each computer that might be addressed with commands or might report data. A computer-unique Task ID is assigned to each addressable entity on each board.
CORBA	Common Object Request Broker Architecture, an inter-process communication mechanism standardized by OMG and used on the SMC
LRI	Liquid Robotics, Incorporated
SC	Storage Controller; an area with an associated Board Address where WGMS can storage or retrieve files
SMC	Sensor Management Computer – an embedded Linux application processor that is resident in a payload bay.
WGMS	Wave Glider Management System – the LRI shore-side control application.

2. Getting Started with the Benthos Modem

2.1 Mission Planning

The combination of the Wave Glider sensor management system with an acoustic modem supports a large number of possible missions. The mission planner must balance mission objectives and environmental constraints when developing the detailed mission plan.

However, one of the key advantages of the LRI integration is that it is easy to modify a mission at any time, including after launch. The core mechanism for controlling the modem is to send BenthosModem module scripts to execute. These scripts can be executed from the SMC file system or can be "real-time", sent from shore. It may be advisable to prepare for likely alternatives prior to launch so that you can test and pre-load the scripts on-shore. It is also possible to load new scripts onto a Wave Glider via the Iridium and other communications links.

The details of the commands are provided in later sections of this manual.

Key considerations include:

- Power budget – Acoustic modems draw a relatively high amount of power while they are in operation; this usage must be included when assessing the total power budget of the Wave Glider.
- Sea floor modem power budget – Special consideration must be given to the battery life of the sea floor modem in various scenarios. Dynamic tuning of operational parameters can reduce power draw and communication quality.

- Testing scripts – Acoustic modems are very flexible tools, capable of supporting thousands of different data collection and reporting schemes. You should confirm that the scripts you develop work as you intended, prior to running them at sea.
- Onboard storage – The SMC generally ships with a 16 GB Secure Digital flash file system, of which approximately 12GB is available for user data. You should estimate how much data your application will generate and configure the system accordingly. Managing file system space and deleting unneeded files is a user responsibility.

Typical use cases are:

- Replacement for locally attached sensor – In this case, a process or script interfaces a sensor attached to a sea floor modem via the pseudo port; connection and modem direction commands may be handled externally. This is a scenario that is easy to implement, but which does not optimize the power used by the sea floor modem.
- Periodically collecting data accumulated in logger – In this case, the sea floor modem remains in low power state, collecting data spontaneously emitted from an attached sensor. When the Wave Glider is in range, it wakes up the sea floor modem long enough to collect accumulated data and perhaps to purge the data, and then the sea floor modem is put back into low power state. This scenario is more optimal for the sea floor modem power budget, but it the processing is completely different than the locally attached sensor case.
- Navigation and optimization of communication – In this mode, the Wave Glider sends ATR (range measurement) and ATY (communication quality) messages and maneuvers and tunes power levels in order to optimize communication; this process may be done with an automated script or manually.

2.2 Setting Up a Benthos Modem for a Mission

2.2.1 Verifying and downloading Benthos firmware

Modems containing different generations of modem firmware are incompatible. The BenthosModem process was developed for use with version 8.2.2; sea floor modems should also be configured with a compatible 8.x.x version of firmware.

To read the version of firmware in use on the modem, cycle power to the modem and read the output at the configured baud rate (9600 baud, by default):

```
Teledyne Benthos ATM-900 Series
LF Frequency Band
Standard version 8.2.2
Aug 18 2011 19:26:49
```

Various firmware sets are available for installation directly from Teledyne Benthos or as listed in configuration BOM P/N 02690 To update the firmware:

a. Connect a terminal program with XModem 1K available to the modem and set the baud rate to 115,200, 8 data bits, 1 stop bit and no parity.

b. Cycle power to the modem and while the modem is booting press the ESC (escape) key.

c. You will see the boot loader come up with something like below:

```
Boot Loader version 1.x
user:1>
```

— or —

```
Boot Loader version 3.x
 (FFS-enabled)
user:1>
```

d. At the user prompt type: (one of the following)

`setpriv root` <enter> (For Boot Loader version 1.x)
Enter password: `neptune`

— or —

`setpriv update` <enter> (For Boot Loader version 3.x)

e. At the prompt type: (one of the following)

`update 0x200000` <enter> (For Boot Loader version 1.x)

— or —

`update boot` <enter> (For Boot Loader version 3.x)

f. Follow on-screen instructions and upload the included "S-270-66....rom" file using 1K-Xmodem.

g. When complete the message "Verifying image... OK" or "Finished programming, verifying... OK." will be displayed.

h. Power down the modem by removing power or switching the unit off.

i. Re-enter the boot loader by following steps a-c

j. At the user prompt type: `setpriv update` <enter>

k. At the update prompt type: `update diag` <enter>

l. Follow on-screen instructions and upload the "S-270-67....rom" file using 1K-Xmodem.

m. When complete the message "Verifying image... OK" will be displayed.

n. Power down the modem by removing power or switching the unit off.

o. Re-enter the boot loader by following steps a-c

p. At the user promt type: update <enter>

q. Follow on-screen instructions and upload the "S-270-68....rom" file using 1K-Xmodem.

r. When complete the message "Verifying image... OK" will be displayed.

s. Power down the modem by removing power or switching the unit off.

t. Connect a PC running a terminal program to the modem serial port at 9,600 baud set to 8 data bits, non-parity and 1 stop bit.

u. Reapply power to the modem.

v. Enter command mode on the modem by typing +++

w. At the user prompt type: `VERSION` <enter> or `ATI` <enter>

x. Verify the new version of code is loaded.

2.2.2 Configuring the modem parameters

Once the modem's firmware is set, the operational parameters should be set as follows:

1. Connect the modem serial port to an available PC serial port or verify connection to the SMC
2. Open the USB serial port provided by the console cable at 9600 baud using a serial terminal like HyperTerminal or putty (Windows) or GkTerm, minicom or screen (Linux)
3. Disable hardware handshaking on the terminal program
4. Enable power to the modem; for example, on the SMC: `sudo SetPowerSwitch 2 1` (modify the switch number as necessary)
5. After the startup banner is displayed, the modem either responds with a command prompt or a 'CONNECT' message. 'CONNECT' indicates that modem is in pass-through mode: any characters typed will be transmitted. To get back into command mode, type +++ (three plus signs) quickly.
6. At the command prompt, issue the following commands for a top side modem:

```
>AT&F

Factory Reset
user:5>@prompt=1
Prompt          | 1
>ATS8=2
OK
>ATS10=255
OK
>ATS13=3
OK
>ATS15=0
OK
>@strictat=ena
Entering strictat mode
StrictAT        | Ena
>AT&W
Sregisters Stored
>
```

For sea floor modems, the following script is suggested:

```
>AT&F

Factory Reset
user:5>@prompt=1
Prompt          | 1
>ATS13=0
OK
>ATS15=1                    (startup mode: 1=online, 2=data logger)
```

```
OK
>ATS18=nnn                    (nnn = the address of the sea floor modem)
OK
>setpriv factory
Password:  (enter password nobska)
Privilege level changed to factory
Ok
```

For SR-50::

```
>@RlsType=1
RlsType          | 1 (SmRel)
```

For SM-75:

```
>@RlsType=2
RlsType          | 2 (SmMdm)
>@RlsCode=nnnnn                (nnnnn = the serial number of the modem)
RlsCode          | nnnnn
>cfg store
Configuration stored.
>@strictat=ena
Entering strictat mode
StrictAT         | Ena
>AT&W
Sregisters Stored
>
```

2.3 Retrieving BenthosModem Data

<data comes back in two forms – file transfer from the SMC and as LRI messages. This section describes the pros and cons of each approach as well as the nuts and bolts of user access to the data.>

2.4 Maintenance and Service

TRDI software and documentation, included in the shipment, describe TRDI-recommended testing and maintenance procedures.

Maintenance and service for the ECO Puck itself is provided by TRDI. For testing purposes, it may be necessary to connect the ECO Puck directly to a Windows PC, using cables supplied by TRDI and included in the shipment.

3. Deployment Checklist

- ☐ Prepare mission plan
- ☐ Confirm that enough mass storage is available on the SMC file system
- ☐ Implement and test any new BenthosModem scripts
- ☐ Implement and test any new sensor-specific scripts via the BenthosModem pseudo-port as necessary
- ☐ Using WGMS, confirm that the modem can connect to a remote modem test fixture

4. Wave Glider recovery and storage – protecting the instrument from damage, shipping crate considerations…

5. Troubleshooting

- It won't turn on
- It returns unreasonable values in testing
- It stopped reporting when at sea
- …

6. Mechanical and Electrical Integration

- Mechanical and Electrical Setup – how the sensor is physically mounted and electrically plumbed.
- How to remove it
- How to install it

7. Software Installation

- Software Installation

8. BenthosModem Module Commands

Commands to the BenthosModem module can be sent individually or in a list; they can also be sent interactively from other programs via CORBA or via shell script using the LRI ServerExec application.

8.1 Command Acknowledgements

Command results (outputs from the modem) are returned as the command result; if no data is returned, a successful result is indicated with "OK"

In the case where multiple commands are given in a single execution, command results are prefixed by the command number in parenthesis. For example, consider a script as follows:

```
BAM ON
BAM CONNECT 100
```

A successful result is indicated as:

```
(1)   OK
(2)   OK
```

Unsuccessful results are indicated by thrown CORBA exceptions, which can be caught easily by languages like C++ or Java. For ServerExec or WGMS (PayloadInterface) commands, the exception is described in the result:

(PayloadInterface)
```
Caught CORBA exception UnavailableInPowerOffState
with message='The modem is not powered up' while
forwarding command to remote server (sc=0x20F3)
```

(ServerExec)
```
Caught CORBA exception BadParameter with
parameterName='command' and message='Too many
arguments for BAM CONNECT'
```

Typical exceptions are as follows:

Exception Name	Description
Aborted	The operation was aborted before it completed, possibly by the user
BadParameter	One or more command parameters are incorrect, missing, could not be parsed or could not be evaluated
CommFailure	There was a communication failure with the modem: no response or unexpected response.
NotReady	The module is not in the appropriate state to handle the command
UnavailableInPowerOffState	The command expects the modem to already be powered on, but it is powered off; or the command attempted to power the modem on, but it failed to do so.

8.2 BAM Command

These are commands to the Benthos Access Manager:

BAM BREAK *milliseconds*

BAM CMD *command-message*

BAM CONNECT [*address*]

BAM CONNECTON [*address*]

BAM CONNECTSTATUS

BAM DISCONNECT

BAM DISCONNECTOFF

BAM OFF

BAM ON

BAM READDATALOGGER *address range*

BAM READRANGE *address*

BAM START-RANGING *address delay-minutes*

BAM STOP_RANGING

8.2.1 BAM BREAK *milliseconds*

Summary

BAM BREAK initiates a break sequence on the connected remote modem, which is held for *milliseconds* on the local modem.

Semantics

If connected, a Linux break is initiated for the indicated duration; this causes the modem to request a break on the remote modem.

Notes

The number of milliseconds does not result in an equivalent length on the remote modem for the tested modem.

Samples

Command:

```
ServerExec BenthosModem 'BAM BREAK 500'
```

Response:

```
OK
```

8.2.2 BAM CMD *command-message*

Summary

BAM CMD sends the given command message to the modem and waits for it to execute, returns the resulting message.

Semantics

If the modem is in 'online' mode, '+++' is sent to transition to command mode; the message is then sent. In most cases, end of the command is delineated by a ">"; when this arrives, the data received during execution is returned. If no data was received, the return string is "OK"

Notes

n/a

Samples

Command:

```
ServerExec BenthosModem 'BAM CMD ATS?'
```

Response:

```
Local Sregisters
S00=082  S01=000  S02=000  S03=002  S04=005  S05=000  S06=001
S07=015  S08=020  S09=000  S10=255  S11=000  S12=000  S13=001
S14=000  S15=000  S16=001  S17=001  S18=000  S19=000  S20=000
```

Command:

```
ServerExec BenthosModem 'BAM CMD ATR100'
```

Response:

```
Range 0 to 100 : 0.0 m
```

8.2.3 BAM CONNECT [*address*]

Summary

BAM CONNECT attempts to connect to the remote modem with the given address; if no address is given, it re-connects.

If the command is successful, the modem will be in online mode so that pass through data received on the pseudo-port are sent to the remote modem for transmission to an

attached device, and received data from the remote modem are transmitted to the pseudo-port for processing by an attached application.

Semantics

If an address is given, the remote address is set (ATS14=*address*) and then a connection attempt is made using “ATD*address*”; otherwise, ‘ATO’ is issued to go online.

Notes

In some configurations, the power must be off for a certain amount of time to ensure that the modem is really off before it can be turned on again; and turning the modem on may incur a delay while capacitors charge. The command will not return until the applicable constraints are satisfied.

Samples

Command:

```
ServerExec BenthosModem ‘BAM CONNECT 100’
```

Response:

```
OK
```

8.2.4 BAM CONNECTON [*address*]

Summary

BAM CONNECTON is the equivalent of BAM CONNECT, but it also powers the modem on if it’s powered off.

If the command is successful, the modem will be in online mode so that pass through data received on the pseudo-port are sent to the remote modem for transmission to an attached device, and received data from the remote modem are transmitted to the pseudo-port for processing by an attached application.

Semantics

If the modem is powered off, the power-on sequence is initiated, which enables the power switch and waits for a sign-on. Once that is received, the command proceeds as with BAM CONNECT. That is, if an address is given, the remote address is set (ATS14=address) and then a connection attempt is made using “ATDaddress”; otherwise, ‘ATO’ is issued to go online.

Notes

n/a

Samples

Command:

```
ServerExec BenthosModem ‘BAM CONNECTON 100’
```

Response:

```
OK
```

8.2.5 BAM CONNECTSTATUS

Summary

BAM CONNECTSTATUS tests the connection status and returns it.

Semantics

If the modem is powered off, ‘OFF’ is returned.

If the modem is in command mode, ‘DISCONNECTED’ is returned.

Otherwise, the modem connected to a remote modem, so ‘ATS14?’ is issued to get the remote address and this return value is used to format the return message as ‘CONNECTED *address*’

Notes

n/a

Samples

Command:

```
ServerExec BenthosModem ‘BAM CONNECTSTATUS’
```

Response:

```
CONNECTED 100
```

8.2.6 BAM DISCONNECT

Summary

BAM DISCONNECT exits from ‘online’ or connected mode back to command mode and hangs up the connection.

Semantics

If the module is currently in online mode, the ‘+++’ sequence is sent to go to command mode. Once in command mode, the ‘ATH’ command is issued to hang up.

Notes

Note that hanging up the local modem puts the remote modem into low power state. In practice, the modem does not respond to ‘ATH’ with a simple ‘>’; instead, it reports the new state as ‘Off-line’

In some configurations, the power must be off for a certain amount of time to ensure that the modem is really off before it can be turned on again; and turning the modem on may incur a delay while capacitors charge. The command will not return until the applicable constraints are satisfied.

Samples

Command:

```
ServerExec BenthosModem ‘BAM DISCONNECT’
```

Response:

```
Off-line
```

8.2.7 BAM DISCONNECTOFF

Summary

BAM DISCONNECTOFF is like BAM DISCONNECT, which means that it exits online mode and hangs up the modem; but it also turns the modem off.

Semantics

If the module is currently in online mode, the '+++' sequence is sent to go to command mode. Once in command mode, the 'ATH' command is issued to hang up. The power is then switched off.

Notes

Note that hanging up the local modem puts the remote modem into low power state. In practice, the modem does not respond to 'ATH' with a simple '>'; instead, it reports the new state as 'Off-line'

In some configurations, the power must be off for a certain amount of time to ensure that the modem is really off before it can be turned on again; and turning the modem on may incur a delay while capacitors charge. The command will not return until the applicable constraints are satisfied.

Samples

Command:

```
ServerExec BenthosModem 'BAM DISCONNECTOFF'
```

Response:

```
Off-line
```

8.2.8 BAM OFF

Summary

BAM OFF switches power to the modem off.

Semantics

The power switch is simply turned off.

Notes

In some configurations, the power must be off for a certain amount of time to ensure that the modem is really off before it can be turned on again; and turning the modem on may incur a delay while capacitors charge. The command will not return until the applicable constraints are satisfied.

Samples

Command:

```
ServerExec BenthosModem 'BAM OFF'
```

Response:

```
OK
```

8.2.9 BAM ON

Summary

BAM ON switches power to the modem on and waits for the modem to emit its power-on messages.

Semantics

The power is switched on and the module waits for the power-on messages followed by either the command mode indicator ('>') or the on-line indicator ('CONNECT *nnnnn* bits/sec'.) If the expected indicators are not received, the power-up is considered to have failed, and power is switched off again.

Notes

In some configurations, the power must be off for a certain amount of time to ensure that the modem is really off before it can be turned on again; and turning the modem on may incur a delay while capacitors charge. The command will not return until the applicable constraints are satisfied.

Samples

Command:

```
ServerExec BenthosModem 'BAM ON'
```

Response:

```
OK
```

8.2.10 BAM READDATALOGGER *address range*

Summary

Reads a range of pages from the data logger at the given address; the resulting data is concatenated when it is returned. The format of "*range*" is:

range	**Description**
nn	Reads a single page (*nn*)
nn-	Reads all pages starting with *nn*
-nn	Reads all pages ending with *nn*
mm-nn	Reads pages in the range of *mm* to *nn*

Semantics

This is the equivalent of issuing AT$BS*target*,*page* for each page in the range.

Notes

n/a

Samples

Command:

```
BAM READDATALOGGER 100 0-
```

Response:

Responds with the raw binary concatenation of the pages read.

8.2.11 BAM READRANGE *address*

Summary

Reads the range and runs a communication test at various baud rates to a remote modem indicated by *address*.

Semantics

This is the equivalent of issuing a script of the following commands:

```
BAM CONNECT address
BAM CMD ATYaddress
BAM CMD ATRaddress
```

The results are returned prefixed with command numbers.

Notes

n/a

Samples

Command:

```
BAM READRANGE 100
```

Response:

```
(1) OK
(2) TX time:02:16:43.9765
(2) Rx Time:02:16:46.0744
(3) TX time:02:16:47.6265
(3) Rx Time:02:16:49.7256
(3) $02400 bits/sec 1 of 4
(3) Acquisition stats SNR:36.6 MPD:00.2 SPD:+00.0
(3) Header stats CRC:Pass SNR:23.9 CCERR:013
(3) MOD:08 ERR:000 SNR:19.0 AGC:29 SPD:+00.0
(3) >Rx Time:02:16:51.9004
(3) $01200 bits/sec 1 of 4, Rate 1/2 CC
(3) Acquisition stats SNR:36.0 MPD:00.3 SPD:+00.0
(3) Header stats CRC:Pass SNR:22.9 CCERR:013
(3) MOD:07 ERR:000 SNR:19.9 AGC:28 SPD:+00.0 CCERR:013
(3) Rx Time:02:16:54.9501
(3) $01066 bits/sec 1 of 4, Rate 1/2 CC, 3.12ms MGP
(3) Acquisition stats SNR:36.1 MPD:00.3 SPD:+00.0
(3) Header stats CRC:Pass SNR:21.7 CCERR:014
(3) MOD:06 ERR:000 SNR:20.1 AGC:30 SPD:+00.0 CCERR:013
(3) Rx Time:02:16:58.2217
(3) $00800 bits/sec 1 of 4, Rate 1/2 CC, 12.5ms MGP
(3) Acquisition stats SNR:36.0 MPD:00.2 SPD:+00.0
(3) Header stats CRC:Pass SNR:19.1 CCERR:013
(3) MOD:05 ERR:000 SNR:21.3 AGC:30 SPD:+00.0 CCERR:013
(3) Rx Time:02:17:02.1588
(3) $00600 bits/sec 1 of 4, Rate 1/2 CC, 25ms MGP
(3) Acquisition stats SNR:36.4 MPD:00.2 SPD:+00.0
```

```
(3) Header stats CRC:Pass SNR:23.6 CCERR:014
(3) MOD:04 ERR:000 SNR:20.1 AGC:30 SPD:+00.0 CCERR:013
(3) Rx Time:02:17:06.9849
(3) $00300 bits/sec 1 of 4, repeated twice, Rate 1/2 CC, 25ms MGP
(3) Acquisition stats SNR:36.4 MPD:00.2 SPD:+00.0
(3) Header stats CRC:Pass SNR:22.9 CCERR:013
(3) MOD:03 ERR:000 SNR:22.1 AGC:31 SPD:+00.0 CCERR:013
(4) TX time:02:17:27.2812
(4) Rx Time:02:17:29.6751
(4) Range 0 to 100 : 0.0 m  (Round-trip  0.1 ms) speed  0.0 m/s
(5) OK
```

8.2.12 BAM START-RANGING *address delay-minutes*

Summary

BAM START-RANGING enters a mode in which the equivalent of READ-RANGE *address* is executed at intervals of *delay-minutes*, sending the results back as telemetry. All other commands are blocked until STOP-RANGING command is received.

Semantics

As with READRANGE, this is the equivalent of the following:

```
BAM CMD ATS13=3
BAM CONNECT address
BAM CMD ATYaddress
BAM CMD ATRaddress
BAM CMD ATS13=1
```

The resulting data is returned as a type 0x91 structure as follows:

FORMAT CODE 0 – RANGE TELEMETRY RESULTS (ATY/ATR)

Field Name	Bytes Used	Byte #	Format
PayloadType	4	0-3	125 (Benthos Modem)
Board Id	1	4	Unsigned 8-bit integer
Task Id	1	5	Unsigned 8-bit integer
Sensor Data Format	1	6 (bits 0-3)	Unsigned 4-bit integer – 0
Default Parser	1	6 (bit 4)	0=Binary/1=ASCII – 0
Delivery Priority	1	6 (bits 5-7)	Unsigned 3-bit integer – 0
Repeat Count	1	7	Unsigned 8-bit integer – 1
Latitude	4	8-11	Latitude
Longitude	4	12-15	Longitude
Time	4	16-19	Unix time_t
Distance to Target	4	20-23	Unsigned 32-bit integer, tenths of meters

Records of the following format representing results for different baud rates, 6 sets (150 bytes):

Baud Rate	2	0-1	Signed 16-bit integer
Acquisition - SNR	2	2-3	Signed 16-bit integer

Acquisition - MPD	2	4-5	Signed 16-bit integer
Acquisition - SPD	2	6-7	Signed 16-bit integer
Header - CRC	1	8	Signed 8-bit integer (0=fail, 1=pass)
Header - SNR	2	9-10	Signed 16-bit integer
Header – CCERR	2	11-12	Signed 16-bit integer (-1 if not reported)
Data - MOD	2	13-14	Signed 16-bit integer
Data - ERR	2	15-16	Signed 16-bit integer
Data - SNR	2	17-18	Signed 16-bit integer
Data - AGC	2	19-20	Signed 16-bit integer
Data - SPD	2	21-22	Signed 16-bit integer
Data – CCERR	2	23-24	Signed 16-bit integer

Notes

With Iridium communication, it's not advisable to request telemetry at intervals of less than 2 minutes as this tends to tie up the communication channel back to shore.

While in this range finding/communication testing telemetry mode, all other commands (exception START-RANGING and STOP-RANGING) report NotReady exceptions.

Samples

Command:

```
ServerExec BenthosModem 'BAM START-RANGING 100 2'
```

Response:

```
OK
```

8.2.13 BAM STOP-RANGING

Summary

BAM STOP-RANGING exits range finding/communication testing telemetry mode (allows other commands to execute.)

Semantics

An internal flag that indicates that the telemetry mode is in effect is cleared.

Notes

The range finding/communication testing telemetry mode is entered with the BAM START-RANGING command.

Samples

Command:

```
ServerExec BenthosModem 'BAM STOP-RANGING'
```

Response:

OK

9. Configuration

SMC service processes are configurable via:

- A “startup” script that is executed to launch the process and optionally issue initializing commands
- An XML-based configuration file, typically named as *ProcessName*-config.xml
- Additional script and data files used by the process

These files are stored in an instance-specific directory under /home/smc/roots; the BenthosModem configuration is typically stored under /home/smc/roots/190_BenthosModem:

```
smc@omap35x_torpedo:~/roots/190_BenthosModem$ ls -l
-rw-r--r-- 1 mike mike 4199 2011-08-22 18:33 BenthosModem-config.xml
-rwxr-xr-x 1 mike mike  211 2011-08-22 19:00 startup
```

Note that some files in other directories can also have an impact. For example, the format of telemetry being returned to shore via the PayloadInterface process and the Float C&C’s Iridium SBD modem is determined by entries in PayloadInterface-config.xml and PayloadInterface-telemetry-config.xml, both of which are located under the 100_PayloadInterface directory.

The most likely items that will need to be changed are:

- **serverName** if there is more than one instance of the process (server names must be unique)
- **loggerSettings→logLevel** if the verbosity of the log output should be changed
- **loggerSettings→processName** to match **serverName**
- **controllerSettings→powerOnVerb** and **controllerSettings→powerOffVerb** if the power switch assignment changes
- **controllerSettings→symlinkName** if there is more than one instance of the process or the pseudo port is moved
- **serialPortSettings→portName** if there is more than one instance of the process or the serial port assignment changes
- **payloadManagerClientSettings→payloadDescription→boardID** to match the payload board ID, typically if more than one SMC is installed
- **payloadManagerClientSettings→payloadDescription→taskID** if there is more than once instance of the process (task ID must be unique)
- **payloadManagerClientSettings→payloadDescription→readableName** to match **serverName**

9.1 Configuration values in BenthosModem-config.xml

Tag Name	Description
serverName	The CORBA name by which the process instance is known; the first parameter to ServerExec
serverKind	A CORBA type parameter that is not typically used on the SMC
loggerSettings	Logger class settings that control the type and verbosity of logging
shutdownEvent	Event settings that describe which event will be received to indicate that the system is about to shut down, typically as a result of a power off command to PayloadInterface
controllerSettings	BenthosModem Controller class settings, the most Benthos modem-specific parameters

Tag Name	Description
serialPortSettings	SerialPort class configuration which configures the baud rate and other parameters used when communicating with the local modem
eventTransceiverSettings	EventTransceiver class settings, which control the CORBA events that are received and distributed
payloadManagerClientSettings	PayloadManagerClient settings, which describe how the process advertises itself to other processes

9.1.1 Benthos Modem Controller class settings

Tag Name	Description
defaultConnectAddress	Reserved for future use
powerOnVerb	The shell command to execute to switch the modem power on
powerOffVerb	The shell command to execute to switch the modem power off
symlinkName	The path where the pseudo port will be linked after it is created
modemSerialListenerSettings	SerialListener class settings that configure how the modem serial port is read and how events are fired to the application
pseudoPortSerialListenerSettings	SerialListener class settings that configure how the pseudo port is read and how events are fired to the application
powerOffTimeoutMS	The number of milliseconds to hold in power off state before allowing the power to be switched on again; this is to ensure that the modem is really off before powering it on
powerOnTimeoutMS	The maximum number of milliseconds to wait for the power on message once the power to the modem is switched on
commandTimeoutMS	The maximum number of milliseconds to wait for a response to a transmitted command.
plusTimeGapMS	The minimum number of milliseconds to impose between the last transmission and sending '+++' to change from Data mode to Command mode. This time gap enables the modem to distinguish data to be transmitted to the remote modem from a mode switch.
dataModeTimeoutMS	Reserved for future use
intervalTimeoutMS	Reserved for future use
maxNumSavedLines	The maximum number of result lines to return from a command.
rangeTelemEvent	Event class settings that define the name of the event that is sent to report the results from the START-RANGING command.
retryNoResponse	Determines whether a 'No response' message from the modem results in a retry (1) or if 'No response' is included with the command response (0)
maxTries	The maximum number of transmissions of a command to make before giving up and failing the command.

9.1.2 Event settings

Tag Name	Description
eventName	The name of the event
eventType→typeName	The type component of the event type (typically corresponds to the specific event class)
eventType→domainName	The domain component of the event type (typically corresponds to an industry or a company)

9.1.3 EventTransceiver settings

Tag Name	Description
eventServiceName	The name of the service to contact when sending or subscribing to events (typically, EventManager)
eventServiceKind	The kind of service to contact when sending or subscribing to events
receiveEnable	Determines whether the process is listening for incoming events (1) or not (0)
aliasToEvent	A list of events to rename before processing
eventFilterEnabled	Determines whether the event filters are applied (1) or not (0)
eventNameFilter	A list of event names that can be received when the event filter is enabled; if the list is empty, it doesn't act as a filter, even if the event filter is enabled.
eventTypeFilter	A list of event types that can be received when the event filter is enabled; if the list is empty, it doesn't act as a filter, even if the event filter is enabled.
displayReceivedEvents	Determines whether incoming events are logged (1) or not (0); events that are filtered out are not displayed in any case

Tag Name	Description
doNotDisplayList	A list of event names for which the event is not displayed in any case

9.1.4 Logger class settings

Tag Name	Description
logLevel	The lowest level of log outputs that are actually output; possible values are, in order of priority: LT_EMERGENCY LT_ALERT LT_CRITICAL LT_ERROR LT_WARNING LT_NOTICE LT_INFO LT_DEBUG
processName	The process name to include on each line of the log; this is useful primarily when logs from several processes are streamed together.
isFileDes	Determines whether logging to a file descriptor (1) or using the file name (0)
fileDes	When isFileDes=1, the file descriptor to log to (1=standard output)
fileName	Reserved for future use
pipeName	When isFileDes=0, the name of the pipe to which to stream

9.1.5 PayloadDescription class settings

Tag Name	Description
boardID	The board ID associated with the client (used by PayloadInterface to contact the service)
taskID	The task ID associated with the client (used by PayloadInterface to contact the service)
description	A description of the service for use when reporting which services are running
ior	The CORBA instance string that allows the running service to be contacted directly; the content of this item will be replaced by a valid runtime value.
readableName	The CORBA instance name that can be used to get the IOR

9.1.6 PayloadManagerClient class settings

Tag Name	Description
serviceName	The name of the process that registers process parameters (i.e. PayloadManager)
serviceKind	The kind of the process that registers process parameters
deviceAddedEventName	The name of the event that advertises new devices when they start
deviceChangedEventName	The name of the event that advertises when a device registration has been modified
deviceRemovedEventName	The name of the event that advertises when a device has been deleted
commonEventType	The type name to use for all device change events
commonEventDomain	The domain name to use for all device change events
payloadDescription	PayloadDescription class settings for the BenthosModem

9.1.7 SerialListener class settings

Tag Name	Description
maxReadSize	The maximum number of bytes to read from the serial port in a given operation (should be set larger than the typical return)
intercharacterGapTimeoutMs	The number of milliseconds between characters that will cause whatever bytes have been received to be transmitted, ideally resulting in a single buffer of bytes for each output

9.1.8 SerialPort class settings

Tag Name	Description

Tag Name	Description
portName	The name of the device node (port) to open
baudRate	The baud rate at which to communicate; only certain discrete settings are accepted: 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800
parity	The parity bits to expect/generate for each byte; valid settings are: EVEN NONE ODD
characterSize	The number of bits to expect/generate for each byte (5…8)
stopBits	The number of stop bits to generate for each transmitted byte (1…2)
flowControl	The type of flow control to be used; valid settings are: HARDWARE NONE XON/XOFF
rts	The initial setting of the RTS line (0…1)
dtr	The initial setting of the DTR line (0…1)
breakTimeoutMultiplier	A multiplier to apply when holding a break signal for a given period of time, which accounts for differences in implementations of Linux

9.2 Example listing of BenthosModem-config.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="9">
<rootlevel class_id="0" tracking_level="0" version="1">
        <serverName>BenthosModem</serverName>
        <serverKind></serverKind>
        <loggerSettings class_id="1" tracking_level="0" version="0">
                <logLevel>LT_INFO</logLevel>
                <processName>BenthosModem</processName>
                <isFileDes>1</isFileDes>
                <fileDes>1</fileDes>
                <fileName></fileName>
                <pipeName></pipeName>
        </loggerSettings>
        <shutdownEvent class_id="2" tracking_level="0" version="0">
                <eventName>Shutdown</eventName>
                <eventType class_id="3" tracking_level="0" version="0">
                        <typeName>EventBase</typeName>
                        <domainName>LiquidR-SMC</domainName>
                </eventType>
        </shutdownEvent>
        <controllerSettings class_id="4" tracking_level="0" version="1">
                <defaultConnectAddress>0</defaultConnectAddress>
                <powerOnVerb>sudo SetPowerSwitch 2 1</powerOnVerb>
                <powerOffVerb>sudo SetPowerSwitch 2 0</powerOffVerb>
                <symlinkName>/home/smc/roots/190_BenthosModem/pseudotty</symlinkName>
                <modemSerialListenerSettings class_id="4" tracking_level="0" version="0">
                        <maxReadSize>65536</maxReadSize>
                        <intercharacterGapTimeoutMs>10</intercharacterGapTimeoutMs>
                </modemSerialListenerSettings>
                <pseudoPortSerialListenerSettings>
                        <maxReadSize>100</maxReadSize>
                        <intercharacterGapTimeoutMs>10</intercharacterGapTimeoutMs>
                </pseudoPortSerialListenerSettings>
                <powerOffTimeoutMS>15000</powerOffTimeoutMS>
                <powerOnTimeoutMS>60000</powerOnTimeoutMS>
                <commandTimeoutMS>10000</commandTimeoutMS>
                <plusTimeGapMS>1500</plusTimeGapMS>
                <dataModeTimeoutMS>1000</dataModeTimeoutMS>
                <intervalTimeoutMS>10000</intervalTimeoutMS>
                <maxNumSavedLines>50</maxNumSavedLines>
                <rangeTelemEvent>
```

```
                        <eventName>BenthosModemRangeTelemetry</eventName>
                        <eventType>
                                <typeName>BenthosModemRangeTelemetryEvent</typeName>
                                <domainName>LiquidR-SMC</domainName>
                        </eventType>
                </rangeTelemEvent>
                <retryNoResponse>1</retryNoResponse>
                <maxTries>5</maxTries>
        </controllerSettings>
        <serialPortSettings class_id="6" tracking_level="0" version="0">
                <portName>/dev/ttyLX3</portName>
                <baudRate>9600</baudRate>
                <parity>NONE</parity>
                <characterSize>8</characterSize>
                <stopBits>1</stopBits>
                <flowControl>NONE</flowControl>
                <rts>1</rts>
                <dtr>1</dtr>
                <breakTimeoutMultiplier>1</breakTimeoutMultiplier>
        </serialPortSettings>
        <eventTransceiverSettings class_id="7" tracking_level="0" version="2">
                <eventServiceName>EventManager</eventServiceName>
                <eventServiceKind></eventServiceKind>
                <receiveEnable>1</receiveEnable>
                <aliasToEvent class_id="8" tracking_level="0" version="0">
                        <count>0</count>
                        <item_version>0</item_version>
                </aliasToEvent>
                <eventFilterEnabled>1</eventFilterEnabled>
                <eventNameFilter class_id="10" tracking_level="0" version="0">
                        <count>1</count>
                        <item_version>0</item_version>
                        <item>Shutdown</item>
                </eventNameFilter>
                <eventTypeFilter class_id="11" tracking_level="0" version="0">
                        <count>0</count>
                        <item_version>0</item_version>
                </eventTypeFilter>
                <displayReceivedEvents>1</displayReceivedEvents>
                <fullyDescribeReceivedEvents>0</fullyDescribeReceivedEvents>
                <doNotDisplayList>
                        <count>0</count>
                        <item_version>0</item_version>
                </doNotDisplayList>
        </eventTransceiverSettings>
        <payloadManagerClientSettings class_id="12" tracking_level="0" version="0">
                <serviceName>PayloadManager</serviceName>
                <serviceKind></serviceKind>
                <deviceAddedEventName>PayloadDeviceAdded</deviceAddedEventName>
                <deviceChangedEventName>PayloadDeviceChanged</deviceChangedEventName>
                <deviceRemovedEventName>PayloadDeviceRemoved</deviceRemovedEventName>
                <commonEventType>PayloadRegistrationEvent</commonEventType>
                <commonEventDomain>LiquidR-SMC</commonEventDomain>
                <payloadDescription class_id="13" tracking_level="0" version="0">
                        <boardID>32</boardID>
                        <taskID>243</taskID>
                        <description>Benthos acoustic modem interface</description>
                        <ior>(This will be overwritten)</ior>
                        <readableName>BenthosModem</readableName>
                </payloadDescription>
        </payloadManagerClientSettings>
</rootlevel>
</boost_serialization>
```

9.3 Telemetry event configuration in PayloadInterface-telemetry-config.xml

Telemetry message are sent asynchronously based on events received by PayloadInterface and are formatted according to PayloadInterface-telemetry-config.xml, an example of which is shown below. The section directly related to the Benthos

Modem is highlighted. Note that board ID and task ID changes for the Benthos Modem must be reflected in this telemetry configuration file, as well.

```
<?xml version="1.0"?>
<root>
   <publishers>
      <Element>
         <first boardID="0x20" taskID="0xF0" />
         <second>
            <key boardID="0x20" taskID="0xF0" />
            <payloadType value="0x0000007B" />
            <eventNameToFormatCode>
               <Element>
                  <first value="MYTELEMEVENT" />
                  <second value="0" />
               </Element>
            </eventNameToFormatCode>
            <reportParams>
               <Element>
                  <first value="0" />
                  <second maxSamples="2" targetAddress="2">
                     <reportHeaderFormat>
                        <fields>
                           <Element libField="Header91" />
                           <Element libField="Raw" eventField="Telem-LengthToFollow" />
                           <Element libField="Raw" eventField="Telem-PayloadType" />
                           <Element libField="Raw" eventField="Telem-BoardID" />
                           <Element libField="Raw" eventField="Telem-TaskID" />
                           <Element libField="FC0BinaryNormal" />
                           <Element libField="Raw" eventField="Telem-NumSamples" />
                        </fields>
                     </reportHeaderFormat>
                     <sampleHeaderFormat>
                        <fields>
                           <Element libField="Raw" eventField="Telem-Latitude" />
                           <Element libField="Raw" eventField="Telem-Longitude" />
                           <Element libField="Raw" eventField="Telem-TimeSec" />
                           <Element libField="DummySample" />
                        </fields>
                     </sampleHeaderFormat>
                  </second>
               </Element>
            </reportParams>
         </second>
      </Element>
      <Element>
         <first boardID="0x20" taskID="0xF3" />
         <second>
            <key boardID="0x20" taskID="0xF3" />
            <payloadType value="0x0000007D" />
            <eventNameToFormatCode>
               <Element>
                  <first value="BenthosModemRangeTelemetry" />
                  <second value="0" />
               </Element>
            </eventNameToFormatCode>
            <reportParams>
               <Element>
                  <first value="0" />
                  <second maxSamples="1" targetAddress="2">
                     <reportHeaderFormat>
                        <fields>
                           <Element libField="Header91" />
                           <Element libField="Raw" eventField="Telem-LengthToFollow" />
                           <Element libField="Raw" eventField="Telem-PayloadType" />
                           <Element libField="Raw" eventField="Telem-BoardID" />
                           <Element libField="Raw" eventField="Telem-TaskID" />
                           <Element libField="FC0BinaryNormal" />
                           <Element libField="RawINT8" eventField="Telem-NumSamples" />
                        </fields>
                     </reportHeaderFormat>
```

```
                  <sampleHeaderFormat>
                     <fields>
                        <Element libField="Raw" eventField="Telem-Latitude" />
                        <Element libField="Raw" eventField="Telem-Longitude" />
                        <Element libField="Raw" eventField="Telem-TimeSec" />
                        <Element libField="Raw" eventField="ATRSample" />
                        <Element libField="Raw" eventField="ATXSample0" />
                        <Element libField="Raw" eventField="ATXSample1" />
                        <Element libField="Raw" eventField="ATXSample2" />
                        <Element libField="Raw" eventField="ATXSample3" />
                        <Element libField="Raw" eventField="ATXSample4" />
                        <Element libField="Raw" eventField="ATXSample5" />
                     </fields>
                  </sampleHeaderFormat>
               </second>
            </Element>
         </reportParams>
      </second>
   </Element>
   <Element>
      <first boardID="0x20" taskID="0x12" />
      <second>
         <key boardID="0x20" taskID="0x12" />
         <payloadType value="0x0000007B" />
         <eventNameToFormatCode>
            <Element>
               <first value="EcoPuckSample" />
               <second value="0" />
            </Element>
         </eventNameToFormatCode>
         <reportParams>
            <Element>
               <first value="0" />
               <second maxSamples="2" targetAddress="2">
                  <reportHeaderFormat>
                     <fields>
                        <Element libField="Header91" />
                        <Element libField="Raw" eventField="Telem-LengthToFollow" />
                        <Element libField="Raw" eventField="Telem-PayloadType" />
                        <Element libField="Raw" eventField="Telem-BoardID" />
                        <Element libField="Raw" eventField="Telem-TaskID" />
                        <Element libField="FC0BinaryNormal" />
                        <Element libField="RawINT8" eventField="Telem-NumSamples" />
                     </fields>
                  </reportHeaderFormat>
                  <sampleHeaderFormat>
                     <fields>
                        <Element libField="Raw" eventField="Telem-Latitude" />
                        <Element libField="Raw" eventField="Telem-Longitude" />
                        <Element libField="Raw" eventField="Telem-TimeSec" />
                        <Element libField="Raw" eventField="col1" />
                        <Element libField="Raw" eventField="col2" />
                        <Element libField="Raw" eventField="col3" />
                        <Element libField="Raw" eventField="col4" />
                        <Element libField="Raw" eventField="col5" />
                        <Element libField="Raw" eventField="col6" />
                        <Element libField="Raw" eventField="col7" />
                        <Element libField="Raw" eventField="col8" />
                        <Element libField="Raw" eventField="col9" />
                     </fields>
                  </sampleHeaderFormat>
               </second>
            </Element>
         </reportParams>
      </second>
   </Element>
    <Element>
      <first boardID="0x20" taskID="0x13" />
      <second>
         <key boardID="0x20" taskID="0x13" />
         <payloadType value="0x0000007C" />
```

```
         <eventNameToFormatCode>
            <Element>
               <first value="CStarSample" />
               <second value="0" />
            </Element>
         </eventNameToFormatCode>
         <reportParams>
            <Element>
               <first value="0" />
               <second maxSamples="2" targetAddress="2">
                  <reportHeaderFormat>
                     <fields>
                        <Element libField="Header91" />
                        <Element libField="Raw" eventField="Telem-LengthToFollow" />
                        <Element libField="Raw" eventField="Telem-PayloadType" />
                        <Element libField="Raw" eventField="Telem-BoardID" />
                        <Element libField="Raw" eventField="Telem-TaskID" />
                        <Element libField="FC0BinaryNormal" />
                        <Element libField="RawINT8" eventField="Telem-NumSamples" />
                     </fields>
                  </reportHeaderFormat>
                  <sampleHeaderFormat>
                     <fields>
                        <Element libField="Raw" eventField="Telem-Latitude" />
                        <Element libField="Raw" eventField="Telem-Longitude" />
                        <Element libField="Raw" eventField="Telem-TimeSec" />
                        <Element libField="Raw" eventField="col1" />
                        <Element libField="Raw" eventField="col2" />
                        <Element libField="Raw" eventField="col3" />
                        <Element libField="Raw" eventField="col4" />
                        <Element libField="Raw" eventField="col5" />
                        <Element libField="Raw" eventField="col6" />
                     </fields>
                  </sampleHeaderFormat>
               </second>
            </Element>
         </reportParams>
      </second>
   </Element>
   <Element>
      <first boardID="0x20" taskID="0x22" />
      <second>
         <key boardID="0x20" taskID="0x22" />
         <payloadType value="0x0000007E" />
         <eventNameToFormatCode>
            <Element>
               <first value="MOSESampleComplete" />
               <second value="0" />
            </Element>
         </eventNameToFormatCode>
         <reportParams>
            <Element>
               <first value="0" />
               <second maxSamples="1" targetAddress="2">
                  <reportHeaderFormat>
                     <fields>
                        <Element libField="Header91" />
                        <Element libField="Raw" eventField="Telem-LengthToFollow" />
                        <Element libField="Raw" eventField="Telem-PayloadType" />
                        <Element libField="Raw" eventField="Telem-BoardID" />
                        <Element libField="Raw" eventField="Telem-TaskID" />
                        <Element libField="FC0BinaryNormal" />
                        <Element libField="RawINT8" eventField="Telem-NumSamples" />
                     </fields>
                  </reportHeaderFormat>
                  <sampleHeaderFormat>
                     <fields>
                        <ELement libField="Raw" eventField="Telem-Latitude" />
                        <ELement libField="Raw" eventField="Telem-Longitude" />
                        <ELement libField="Raw" eventField="Telem-TimeSec" />
                        <Element libField="Raw" eventField="MOSECompleteEvent_Hs" />
```

```
                        <Element libField="Raw" eventField="MOSECompleteEvent_Tav" />
                        <Element libField="Raw" eventField="MOSECompleteEvent_Tp" />
                        <Element libField="Raw" eventField="MOSECompleteEvent_Dirp" />
                     </fields>
                  </sampleHeaderFormat>
               </second>
            </Element>
         </reportParams>
      </second>
   </Element>
</publishers>
<fieldLibrary>
   <lib>
      <Element>
         <key value="Raw" />
         <value content="CA_RAW" />
      </Element>
      <Element>
         <key value="DummySample" />
         <value content="CA_LITERAL">
            <literal type="STRING" value="[D]" />
         </value>
      </Element>
      <Element>
         <key value="Header91" />
         <value content="CA_LITERAL">
            <literal type="UINT8" value="0x91" />
         </value>
      </Element>
      <Element>
         <key value="FC0BinaryNormal" />
         <value content="CA_LITERAL">
            <literal type="UINT8" value="0x00" />
         </value>
      </Element>
      <Element>
         <key value="RawINT8" />
         <value content="CA_LENGTH1" />
      </Element>
   </lib>
</fieldLibrary>
</root>
```

10. Support