

	Software Release Note for Sensor Management Computer (SMC) Core Components version 1.1.8		
	responsible engineer	M. Cookson	created 05/31/2012
	product name	SMC Core Components	last updated 06/05/2012
	software version	1.1.8	Status Final
	software part number	02617-12	
	document part number/rev	02619-12 Rev A	

TABLE OF CONTENTS

Overview.....	4
Changes in Version 1.1.8 (05/31/2012, P/N 02617-12).....	5
New Features.....	5
Bug Fixes	13
Dependencies	16
Open Issues.....	17
End-user Impacts	18
Support Impacts	18
Changes in Version 1.1.7 (04/16/2012, P/N 02617-11).....	18
New Features.....	18
Bug Fixes	18
Dependencies	18
Open Issues.....	19
End-user Impacts	19
Support Impacts	19
Changes in Version 1.1.6 (03/26/2012, P/N 02617-10).....	19
New Features.....	19
Bug Fixes	20
Dependencies	20
Open Issues.....	20
End-user Impacts	21
Support Impacts	21
Changes in Version 1.1.5 (02/03/2012, P/N 02617-09).....	21
New Features.....	21
Bug Fixes	21
Dependencies	22
Open Issues.....	22
End-user Impacts	22
Support Impacts	22

Changes in Version 1.1.4 (01/18/2012, P/N 02617-08).....	22
New Features.....	22
Bug Fixes	22
Dependencies	22
Open Issues.....	23
End-user Impacts	23
Support Impacts	23
Changes in Version 1.1.3 (12/15/2011, P/N 02617-07).....	23
New Features.....	23
Bug Fixes	23
Dependencies	23
Open Issues.....	23
End-user Impacts	23
Support Impacts	24
Changes in Version 1.1.2 (11/16/2011, P/N 02617-05).....	24
New Features.....	24
Bug Fixes	24
Dependencies	25
Open Issues.....	25
End-user Impacts	26
Support Impacts	26
Changes in Version 1.1.1 (11/11/2011, P/N 02617-04).....	26
New Features.....	26
Bug Fixes	26
Dependencies	26
Open Issues.....	27
End-user Impacts	27
Support Impacts	27
Changes in Version 1.1.0 (10/13/2011, P/N 02617-03).....	27
New Features.....	27
Bug Fixes	27
Dependencies	28
Open Issues.....	28
End-user Impacts	29
Support Impacts	30
Changes in Version 1.0.10 (8/19/2011, P/N 02617-02).....	30
New Features.....	30
Bug Fixes	31

Dependencies	31
Open Issues.....	32
End-user Impacts	33
Support Impacts	33
Changes in Version 1.0.9 (5/24/2011, P/N 02617-01)	33
New Features.....	33
Bug Fixes	33
Dependencies	33
Open Issues.....	33
End-user Impacts	33
Support Impacts	34
Installation/Upgrade Instructions	34
Setting up the SD Card	34
Working with the SMC and the Float C&C	35
Installing the Application	36
Changing Encryption Passphrases.....	38
Component Configuration	39
Aquatracka (Chelsea Technologies Group) Configuration	40
Commands	40
BenthosModem Configuration.....	42
Configuring the Client Program.....	42
Loading the Modem Firmware	42
Configuring Modem Parameters	43
C3 (Turner Designs) Configuration.....	44
Commands	44
Events and Telemetry	44
Common XML file keys to edit	46
DataRecorder Configuration	46
Common XML file keys to edit	46
Using the DataRecorder.....	46
Logger	47
LuaWrapper Configuration	48
Commands	48
Special Variables	49
CORBA Event Tables and Variable Definitions	51
Pre-defined Callback Functions	53
XML Configuration	54
Sample lua script.....	54

MAM Configuration (Datawell MOSE).....	56
Commands	57
Common XML file keys to edit	58
PayloadInterface Configuration.....	58
PPPHost Configuration.....	59
Common XML file keys to edit	59
Configuring the RUDICS Modem	59
Configure the dialup script	60
Configure the PPP connection script	60
Manual connection to Iridium network.....	60
Setting up a key file for scp/ssh (dbclient) connection without password	63
timelimit2.....	64
VemcoVR2C Configuration	64
Common XML file keys to edit	64
Using the VemcoVR2C.....	65
WetLabsSensorSimulator.....	65
Appendix 1 – Sample Application Configuration File	67
Appendix 2 – Sample Telemetry Configuration	69
History.....	74

Overview

The Sensor Management Computer (SMC) core components encompass the superset of standard applications, libraries, scripts and configuration files that might be loaded onto the embedded SMC Linux computer. These include:

COMPONENT	DESCRIPTION	VERSION
Aquatracka	Interface to Chelsea Aquatracka fluorimeter	1.1.8+
BenthosModem	Interface to Benthos acoustic modems (tested with version 8.2.2)	1.0.10+
busy	A utility that can be invoked from script to keep the CPU from going idle (busy on) or to turn that mode off (busy off)	1.1.8+
C3	Interface to Turner Designs C3 fluorometer	1.1.8+
catgps	This is a utility that can be used during board setup or troubleshooting to configure and show the output of the GPS module when the SMC software is not running.	1.1.8+
ConfigOctalUartLine	Configures an octal UART port electrical protocol (must be executed as root)	1.0.9+
ConfigurePower	Configures the power-saving features of the platform (must be executed as root)	1.0.9+
DataRecorder	A module capable of archiving received serial port data and making it available for extraction at runtime	1.1.0+
DemoSensor	An example sensor implementation	1.0.9+
EnableEthernet	Activates the Ethernet interface and establishes an IP address via DHCP	1.0.9+
EventManager	CORBA-based event subscription and distribution service	1.0.9+

COMPONENT	DESCRIPTION	VERSION
GPSTListener	Interface to GPS receiver	1.0.9+
InitOctalUartLines	Configures the electrical protocol for all octal UART ports (must be executed as root)	1.0.9+
InitPowerSwitches	Configures the power switches to their initial (off) state (must be executed as root)	1.0.9+
killsmc	Stops all SMC processes (must be executed as root)	1.0.9+
Logger	A utility program that accepts text from standard input and writes to a log file set with configurable maximum size	1.1.0+
LogListener	Listener for reported log events from the Float C&C	1.1.2+
LuaWrapper	Wrapper for a lua interpreter instance: allows execution of lua scripts	1.1.2+
MakeThumbnail	Creates a reduced resolution image of a picture.	1.1.8+
MakeZoom	Creates a cropped image from a picture.	1.1.8+
MAM	Interface to Datawell motion sensor	1.0.9+
omniNames	CORBA (omniORB) name server	1.0.9+
PayloadInterface	Implementation of Float C&C payload protocol, which exposes services registered via CORBA to the Float C&C and WGMS	1.0.9+
PayloadManager	A CORBA names registry that can be enumerated	1.0.9+
PPPHost	Manager of a point-to-point protocol connection (typically RUDICS) and executor of configured scripts while the internet connection is open	1.0.9+
ReportBoardVersion	Reports the version number of the SMC board ("ReportBoardVersion" or "ReportBoardVersion -v")	1.1.8+
ServerExec	A utility that contacts an application via its CORBA-registered name and invokes a command, returning any output.	1.0.9+
SetPowerSwitch	Turns one of the SMC power switches on or off (must be executed as root)	1.0.9+
TakePicture	Powers on an attached camera, sets zoom level, snaps a picture and downloads the image.	1.1.8+
timelimit2	Utility that limits runtime to a maximum time	1.1.2+
VemcoVR2C	Interface to the Vemco VR2C sensor	1.1.3+
WetLabsSensor	Interface to Wet Labs sensors like Eco Puck and CStar	1.0.10+
WetLabsSensorSimulator	Generate simulated Eco Puck samples over a pseudo serial port.	1.1.0+

Note that this distribution is intended for those who are comfortable with Linux and making configuration changes. Other installation kits for specific configurations are also available.

Changes in Version 1.1.8 (05/31/2012, P/N 02617-12)

New Features

SWSMAC-303 "Turner C3: initial implementation on SMC for data logging mode"

Description

Implement interface for the Turner C3 sensor on the SMC, as a part of the core software. The initial implementation requires controlling the power to the sensor via the on board power switches, receiving header and data samples, logging them on the SMC filesystem, and sending them as telemetry events whose format is similar to the existing interface implementation by the PIB.

Resolution

This initial implementation supports the following commands, whose syntax is similar to the existing PIB based interface:

- c3 -o Turns on the power switch associated with the sensor
- c3 -f Turns off the power switch associated with the sensor
- c3 -r Alias for c3 -o

c3 -x Alias for c3 -f

Received header and data samples are sent as telemetry (0x91) events formatted similar to the existing PIB based implementation.

The C3 application is a module capable of interfacing to, and receiving data, over a serial port, from a Turner C3 sensor. All communications and data received from the sensor is logged in the application's log file (~/.log/C3.log) and all data reported by the sensor (read from the serial port as ASCII strings) are also sent as telemetry events C3SampleHeader and C3SampleData of type C3Event.

The C3 application is started from: ~/roots/200_C3/

And is configured by the XML file: ~/roots/200_C3/C3-config.xml

The C3 instance is configured with the serverName C3, and assigned the boardId 32 (hexadecimal 20) and taskId 23 (hexadecimal 17).

Common XML file keys to edit:

<serialPortSettings> for C3, the default serial port is /dev/ttyLX4.

By default it is set to communicate with a baudrate of 9600, parity NONE, 1 stopbit, characterSize 8, and flowControl NONE. Ensure that the serial connector cables are plugged into the specified serial port on the SMC.

<powerOnVerb> and <powerOffVerb> for C3, the command string references power switch. Ensure that the power cable is plugged into the specified power switch on the SMC.

<defaultOutfile> for storing all header and data strings received from the C3, on the SMC filesystem. The default specified is c3data.txt in the local directory (/home/smc/roots/200_C3)

The C3Event type contains the following name-value pair items:

"C3EventName" = C3Event_C3SampleHeader or C3Event_C3SampleData

"Header" = header string received from sensor

"Date" = date value (format mm/dd/yyyy) in the sample data received from sensor

"Time" = time value (format hh:mm:ss) in the sample data received from sensor

"DataC1" = first data item in the sample data from sensor

"DataC2" = second data item in the sample data from sensor

"DataC3" = third data item in the sample data from sensor

"Pressure" = fourth data item in the sample data from sensor

"Temperature" = fifth data item in the sample data from sensor

"CompressedData" = all five data items packed into an array of 14 bytes as follows: first, second, third, and fourth data items represented as 3 bytes each where the first two bytes are the mantissa value and the third byte is the decimal value * 100, followed by the fifth data item * 100 represented as a two byte value.

The events sent as telemetry (0x91 message) to WGMS contain the following items:

C3SampleHeader = Latitude, Longitude, Timestamp, "Header" described above

C3SampleData = Latitude, Longitude, Timestamp, "CompressedData" described above

SWSMAC-217 "How to determine the revision of a SMC board through software"
--

Description

The SMC contains a PCA9554APW I2C expander that indicates the H/W revision of the SMC board. Upon reset, the PCA9554APW is in a state where its 8 input/output pins default to input mode with a weak pull-up resistor thus no additional mode settings are necessary.

The PCA9554APW can be read with the i2cget linux utility as follows:

i2cget -y 2 0x38 0

The return values from the command above can be interpreted as follows:

0x00 = SMC Revision A or B

0x01 = SMC Revision C

Resolution

I've added a new script called ReportBoardVersion, which can be invoked as:

(1) ReportBoardVersion

- if the underlying version is 0, report "Rev A or B"
- if the underlying version is 1, report "Rev C"
- otherwise, report the number and unknown; for example, "0x4A unknown"

(2) ReportBoardVersion -v

- always reports the number and the decoded version; for example, "0x00 Rev A or B" or "0x01 Rev C" or "0x4A unknown"

SWSMAC-232 "Datawell: new config option to average N spectra"

Description

Allow user to choose a different number for averaging of the spectrums. According to Datawell, larger average number yields a smoother spectrum.

Originally, the processing of the spectrum was based on 512 samples with averaging = 2 and 256 samples per spectrum.

Resolution

Added these new parameters in the MOSESensor-config.xml for flexible configuration.

```
<sampleNum>1792</sampleNum>
<sampleAverage>7</sampleAverage>
<spectrumNum>256</spectrumNum>
```

Note, spectrumNum must be a power of 2. $256 \times 7 = 1792$

The collecting duration must be long enough to collect enough samples.

```
<collectingDuration>20</collectingDuration>
```

With 2Hz data rate, 1792 needs about 15 minutes. Therefore, we have about 5 minutes buffer.

SWSMAC-233 "Aquatracka: create a set of the basic commands"

Description

Create a new Aquatracka project which begins with a set of the basic commands

Resolution

Implemented basic commands for Aquatracka:

```
AQUAT --start
AQUAT --stop
AQUAT -o | --output pathname
AQUAT --off
AQUAT --on
AQUAT --getconfig
AQUAT --header
AQUAT --pause
AQUAT --resume
AQUAT --verbose
AQUAT --noverbose
```

SWSMAC-234 "Aquatracka: create events for each data type"

Description

Create events for data samples and header info

Resolution

Added CORBA events to announce the Aquatracka header information and sample data:

1. The Header event contains a copy of the Aquatracka header data; the interval at which the header data is sent (headerInterval, minutes) and the details for the CORBA event name and type are configurable in the Aquatracka-config.xml file:

```
<headerInterval>60</headerInterval>
<headerEventSettings>
  <eventName>AquatrackaHeaderReport</eventName>
  <eventType>
```

```
<typeName>StringEvent</typeName>
<domainName>LiquidR-SMC</domainName>
</eventType>
</headerEventSettings>
```

2. The Data sample event contains a copy of the sample data from the Aquatracka sensor as text; this is sent out whenever a sample is received. The details for the CORBA event name and type are configurable in the Aquatracka-config.xml file:

```
<sampleEventSettings>
  <eventName>AquatrackaSampleReport</eventName>
  <eventType>
    <typeName>StringEvent</typeName>
    <domainName>LiquidR-SMC</domainName>
  </eventType>
</sampleEventSettings>
```

SWSMAC-252 "Aquatracka: xml for calibration constants"

Description

Without the built-in memory, Chelsea Technologies writes the calibration constants on the Certificate of Calibration. These constants are needed for the temperature and concentration calculation. The Calibration-config.xml is needed to store these constants on system.

Resolution

A new configuration file (Calibration-config.xml) can be placed on the device to set up calibration constants and the serial number. The format of the file is as follows:

```
<?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">
  <serial>Unknown-Serial</serial>
  <temperature1>0.0</temperature1>
  <temperature2>0.0</temperature2>
  <temperature3>0.0</temperature3>
  <fluoroRef1>0.0</fluoroRef1>
  <fluoroRef2>0.0</fluoroRef2>
  <fluoroRef3>0.0</fluoroRef3>
  <fluoroSig1>0.0</fluoroSig1>
  <fluoroSig2>0.0</fluoroSig2>
  <fluoroSig3>0.0</fluoroSig3>
</rootlevel>
</boost_serialization>
```

SWSMAC-255 "Aquatracka: Select and Export commands"

Description

Need the SELECT and EXPORT commands for filtering.

Resolution

The SELECT command directs AQUATRACKA program to select samples from the data stream for further processing:

```
SELECT [-t | --time | -s | --samples | -d | --distance] N
```

The EXPORT command saves the selected raw samples as indicated by the SELECT command:

```
EXPORT --count N | -o | --output export-file
```

If --count is provided, it sets the total number of samples to be exported to the file to N

The -o or --output options direct the exported samples the given file (export-file)

SWSMAC-256 "Datawell: autostart"

Description

Added new auto-start in the configuration

Resolution

Added the autoStart parameter in Aquatrack-config.xml. If autostart is true, Aquatracka will start to run after power on.

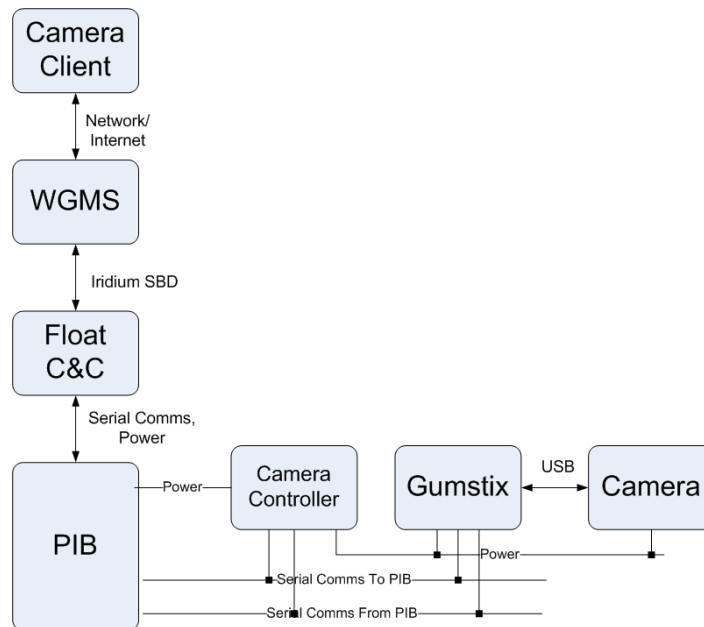
```
<autoStart>0</autoStart>
```

Note, autoStart should not be true if the MFG has not set the calibration constants in the Calibration-config.xml file.

SWSMAC-281 "Create scripts for SMC to allow Camera Client or WGMS user to take pictures"

Description

The existing still camera communication architecture is like this:



The communication use case is like this using file transfer:

01. Power on the payload

02. Power on the Gumstix/Camera using Set Parameter

03. To Gumstix: 3/9/2012 11:46:46 script.txt

```
gphoto2 --force-overwrite --set-config zoom=8 --capture-image-and-download
```

04. New File Available from Gumstix: 3/9/2012 11:53:30 000ab6c2.txt

Error ***

The zoom widget is not configurable.

Downloading 'IMG_0026.JPG' from folder '/store_00010001/DCIM/100CANON'...

No event yet, we'll try later. New file is in location

/store_00010001/DCIM/100CANON/IMG_0026.JPG on the camera

Saving file as IMG_0026.JPG

Deleting 'IMG_0026.JPG' from folder '/store_00010001/DCIM/100CANON'...

Deleting file /store_00010001/DCIM/100CANON/IMG_0026.JPG on the camera

05. To Gumstix: 3/9/2012 12:05:32 script.txt

```
convert -resize 5% IMG_0026.JPG IMG_0026Thumb.jpg; ls -l IMG_0026Thumb.jpg
```

06. New File Available from Gumstix: 3/9/2012 12:12:05 000ab6c8.txt

rw-rr- 8 root root 20026 Mar 9 20:11 IMG_0026Thumb.jpg

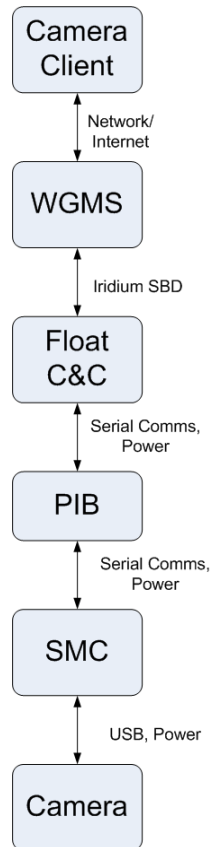
07: To Gumstix: 3/9/2012 12:12:15 script.txt

IMG_0026Thumb.jpg

08: New File Available from Gumstix: 3/9/2012 15:17:17 000ab6d8.jpg

(image)

When implementing on the SMC, the connection will be like this:



The important features to implement in priority are:

1. Send a single command to snap a picture with zoom level parameter. This would include powering the camera on and off and saving the full resolution image on the SD card, and that filename would be returned. Files downloaded from the camera are renamed in a format that the camera does not use (new files never overwrite old files)
2. Return any resulting file by name.
3. Resize a full-size image and optionally specify a section of the file, resulting in a new file.
4. (1) that includes the resolution of the desired thumbnail image and returns both the full res image name and the thumbnail name.
5. (4) that adds the option to send back the resulting thumbnail or full size image via separate file transfer. This assumes that the original command is sent via Write Console Script.
6. (3) that optionally sends back the resulting image via separate file transfer. This assumes that the original command is sent via Write Console Script.
7. Create a method for automatically managing the files on the SMC.

Resolution

The following scripts were created:

TakePicture -z zoom-level

The TakePicture script turns the camera on, sets camera parameters like flash and zoom, snaps the picture, downloads the resulting image and powers the camera off. The resulting image filename is chosen based on a date code to avoid accidentally overwriting existing files. A log of operations is returned; for example, the user executes:

```
TakePicture -z 3
```

The output of the command might be:

```
Turning Camera On
SetPowerSwitch: 0x11
Extracting Lens, Setting Flash Off, and Setting Zoom
Taking Picture...
Downloading 'IMG_0095.JPG' from folder '/store_00010001/DCIM/100CANON'...
Deleting 'IMG_0095.JPG' from folder '/store_00010001/DCIM/100CANON'...
/store_00010001/DCIM/100CANON/IMG_0095.JPG
Retracting Lens
Turning Camera Off
SetPowerSwitch: 0x00
Image at: /mnt/XDATA/pictures/IMG_20000101231818.JPG
Automation should look for a line starting with "Image at:" to determine where the resulting
file is stored.
```

Note that the "zoom-level" parameter is an argument to the Linux gphoto2 command and may be camera-specific.

MakeThumbnail *thumbnail-percent original-image resulting-image*

Creates a thumbnail (a reduced size representation of the full image) of "original-image" and stores it at "resulting-image"; the "thumbnail-percent" indicates how much reduction should be used. For example:

```
MakeThumbnail 5% /mnt/XDATA/pictures/IMG_20000101231818.JPG
/mnt/XDATA/pictures/IMG_20000101231818Thumb.JPG
```

The resulting output is a 'ls -l' of the resulting image file; for example:

```
-rw-r--r-- 1 root root 27079 Jan  1 23:20
/mnt/XDATA/pictures/IMG_20000101231818Thumb.JPG
```

MakeZoom *crop-area image-quality original-image resulting-image*

Crops and/or adjusts the quality of "original-image" and stores the result to "resulting-image". The "crop-area" parameter is a set of instructions to ImageMagick (see <http://www.imagemagick.org/script/command-line-processing.php#geometry>) that can translate or specify an area size within the source image; the "original-image" specifies how much compression is attempted (see <http://www.imagemagick.org/script/command-line-options.php?ImageMagick=5d3lqmf7u4tsagap2e3l9odp11#quality>). A value of 1 (minimum) uses the highest compression and lowest image quality; a value of 100 (maximum) uses the lowest compression and highest image quality. For example:

```
MakeZoom 100x100+1000+1000 30 /mnt/XDATA/pictures/IMG_20000101231818.JPG
/mnt/XDATA/pictures/IMG_20000101231818Stage1.JPG
```

In this case, a 100 by 100 pixel section of /mnt/XDATA/pictures/IMG_20000101231818.JPG starting at (x=1000,y=1000 pixels) offset within the picture with image quality 30 is written to /mnt/XDATA/pictures/IMG_20000101231818Stage1.JPG. The output of the command is a 'ls -l' of the resulting file; for example:

```
-rw-r--r-- 1 root root 11721 Jan  1 23:24
/mnt/XDATA/pictures/IMG_20000101231818Stage1.JPG
```

Note that these scripts are being included in the 1.1.8 release, but they will not work without a 6.0 kernel.

SWSMAC-288 "Datawell: handle gap in data, error flag in a sample, and missing received characters"

Description

The spectrum processing requires a good set of data. Any data which has time gap of more than 0.5 second, or has error flag of 1, or incomplete or missing characters is considered an error.

Resolution

Check all samples and keep track of the number of bad and good samples. When the number of errors reaches the error threshold, it disqualifies the set of data by clearing the data queue and start the queue from the beginning.

To allow user to set the error threshold, a new parameter called "errorThreshold" is added in the MOSESensor-config.xml.

Statistics of all samples are also added in the log for debugging purpose.

SWSMAC-290 "Suggest the upgraded form of a configuration XML automatically"

Description

When changes are made to a boost XML serialization schema, it is typical to provide the ability to load any old format as well as the new format and to save in the new format (only.) It would be nice if the new format of the XML were made immediately available rather than requiring extra steps to have it generated.

Resolution

The LoadSaveBoostXml module was modified to automatically drop an "upgraded" file indicating how the new schema would encode the same data that was originally read from an older format XML file. This should be backward compatible to all other callers, as long as they want the feature. It can be disabled with the appropriate parameter settings (in code)

SWSMAC-292 "Create a shortcut for outputting GPS data"

Description

It would be convenient if there were a script to configure the GPS port and output it to standard output. This would make it easier for people with limited Linux experience to test if GPS input is working.

Resolution

A new script (catgps) was added that configures the GPS port and executes a "cat" on it to continually send it standard output. For example:

catgps

SWSMAC-300 "Accommodate 1-second GPS"

Description

The SMC should accommodate 1-second GPS, which includes:

- continuing to support one event every 5 seconds as an event
- adding a new event that reports every second

This will reduce over-all load on the system since the EventManager now supports subscriptions (since 1.1.7) In addition:

- "GST" (accuracy) and "RMC" (recommended minimum) events are now reported; the ordering of the messages is different when the 1-second GPS firmware is installed vs. the legacy code. GPSTListener should support both scenarios.
- "RMC" reports magnetic declination, which would be useful to add to the GPSSample event
- "GST" reports standard deviation for latitude, longitude and altitude. The Float C&C computes RMS of these 3 values and uses it to determine whether to trust the sample. This would be a good feature.
- The location sentence reports geoidal separation and antenna altitude, but they were never added to the GPSSample event. It would be useful to have these values.

Resolution

The parsing features were added as specified in the request. The following new fields were added to GPSSample (the CORBA event that is broadcast):

- magnetic declination
- RMS accuracy
- elevation (antenna)

The configuration of the GPS event name has been augmented to include a corresponding seconds divisor, so it's possible to have separate events for 1 and 5 second sample and to add more events through XML configuration as needed:

```
<gpsEventSpecs class_id="3" tracking_level="0" version="0">
```

```

<count>2</count>
<item_version>1</item_version>
<item_class_id="4" tracking_level="0" version="1">
  <divisor>5</divisor>
  <divisorMode>0</divisorMode>
  <eventSettings class_id="5" tracking_level="0" version="0">
    <eventName>GPSSample</eventName>
    <eventType class_id="6" tracking_level="0" version="0">
      <typeName>GPSSample</typeName>
      <domainName>LiquidR-SMC</domainName>
    </eventType>
  </eventSettings>
</item>
<item>
  <divisor>1</divisor>
  <divisorMode>0</divisorMode>
  <eventSettings>
    <eventName>GPSSample1Sec</eventName>
    <eventType>
      <typeName>GPSSample</typeName>
      <domainName>LiquidR-SMC</domainName>
    </eventType>
  </eventSettings>
</item>
</gpsEventSpecs>

```

This configuration retains event name "GPSSample" once every 5 seconds and adds "GPSSample1Sec" every second.

In addition, the RMS accuracy threshold used to determine whether the GPSSample event is transmitted at all is configured through a new setting (rmsAccuracyThreshold, the maximum RMS value that will allow reporting.)

Bug Fixes

SWSMAC-238 "GPSTListener sets the hardware clock too frequently"

Description

GPSTListener runs the hardware clock verb to synchronize the hardware clock with the system time whenever it updates the time (whenever the current time is more than 0.100 seconds different from the time read from GPS) This causes problems when there is a long delay in writing to the hardware clock for whatever reason, including:

- unreliability of i2c comms
- high frequency of update

Once a delay is added, then processing times vary and trigger more frequent updates. As a result, the time is not stable and kernel error messages start being produced because the i2c bus is overloaded. I would suggest the following:

- make the delta time an XML-configurable item
- add a new setting for the minimum interval (seconds) between updating the hardware clock: a value of 0 would cause the hardware clock only to be updated on the first valid GPS reading
- requeue time setting and event generation to another thread to avoid slowing down serial parsing

Environment

SMC with GPSTListener configured and GPS connected, particularly with 1-second GPS

Root Cause

Unreliable communication with the hardware clock via i2c bus, GPSTListener triggers this problem by setting the hardware clock too frequently

Resolution

To address setting the hardware clock too frequently, new parameters were added to revision 4 of the GPSTListener settings:

deltaToSetTimeMS: configures the minimum delta in milliseconds between the OS time and the reported GPS time that will trigger the OS time to be set

deltaToSetHWClockMS: configures the minimum elapsed time in milliseconds that would allow the hardware clock to be set, assuming that the OS time was updated; a value of zero will allow the hardware clock to be set only the first time the OS time is updated.

Two new worker queues were added to decouple (1) setting the time and (2) sending out CORBA events from parsing the serial stream. The time-setting thread only evaluates the incoming time if there are no other waiting settings, so it won't get chronically behind.

SWSMAC-257 "Datawell: imprecise latitude and longitude info"

Description

Latitude and longitude information is missing floating numbers. Need to be more precise.

Root Cause

Coding error when converting latitude and longitude numbers, perhaps due to lack of understanding of requirements for latitude/longitude display resolution.

Resolution

Increased the number of digits of precision for latitude (setprecision(9)) and longitude (setprecision(10))

SWSMAC-262 "ServerExec command string containing one letter (A-Z or a-z) or beginning with one certain uppercase letter followed by a non letter character is being interpreted as not-a-string"

Description

The following commands will be interpreted as not-a-string (i.e. passed as a CORBA byte array rather than a CORBA string):

ServerExec C3 "A"
ServerExec ADCPSensor "z"
ServerExec C3 "C3"

The following will be interpreted normally (passed as a CORBA string):

ServerExec C3 "c3"
ServerExec C3 "A3"

Root Cause

Unexpected output from Linux 'file' command (small ASCII sentences are sometimes classified as something other than ASCII text)

Resolution

ServerExec previously relied on the DetermineFileExtension module to determine if it passed data as a string or as a byte array; this has been changed. CORBA C bindings do not allow nils to be passed within a string, so now ServerExec passes anything that does not have a nil in it as a string. Modules expecting SendCommand to receive a byte array should be coded to convert received strings into byte arrays.

The DetermineFileExtension module was modified to produce a .TXT extension for more cases. In addition, LZMA compressed files and tar balls now produce appropriate extensions.

SWSMAC-271 "Datawell: power manager turns MOSE on unintentionally"

Description

MAM's PowerManager automatically turns on the MOSE sensor although the user has purposely powered off the sensor and intended to leave it off.

Root Cause

The idea that the user might want to override automatic power management with "MOSE --off" was not an understood requirement (though it is a good idea, in retrospect)

Resolution

A new variable called pwrMgrAllowed was added to determine whether MAM updates power automatically. Initially it is set to "true", but if the user executes "MOSE --off", it is set to "false", so the power will no longer be updated automatically. "MOSE --on" sets the flag to "true" again, so automated power management is re-enabled.

SWSMAC-272 "RUDICS (pppd) connection does not detect loss of connection"

Description

The RUDICS (pppd) connection invoked by PPPHost has not made any provision through its configuration to detect that the connection drops. Some workarounds have been added like use of timelimit or timelimit2. Without this, the RUDICS interface can hang.

To test this directly with the smc software installed but not running:

1. Go to the /home/smc/roots/200_PPPhost directory
2. Enable power to the modem (typically, sudo SetPowerSwitch 1 1)
3. Wait for 20 seconds or so.
4. Run the pppd script in the background (./runppp &); note that this step can fail, in which case re-execute the runppp command.
5. Issue a command to verify connection to the internet is live (for example, sudo ping www.yahoo.com; cancel with ^C after some time)
6. Turn off power to the modem to simulate connectivity loss (sudo SetPowerSwitch 1 0)
7. Monitor the process listing occasionally with (ps -efH) and look for pppd. Note that the process does not time out or exit.

This experiment can be run using an executing system by watching the PPPHost log (tailf ~/log/PPPHost.log) and then turning the power off after the ppp connection is made. In some cases, an scp or dbclient/ssh comand will time out, but pppd does not detect loss of communication.

Environment

SMC with RUDICS connection via PPPHost, modem connected to SMC comms and power, antenna connected to modem with a clear view of the sky for Iridium connectivity

Root Cause

None of the provisions for detecting loss of communication were added to the pppd invocation.

Resolution

Modified the runppp script to include lcp-echo-interval (60) and lcp-echo-failure (3) terms, which causes pppd to send echo requests every minute and terminate the connection after 3 failures.

SWSMAC-276 "Aquatracka: allow to send a stop command while not running"

Description

Sending a command in a wrong state causes the Aquatracka program to hang.

Root Cause

Program design did not take into account that the sensor does not respond with a string if a "stop" command is sent when it is not running.

Resolution

To resolve the issue, if the sensor is not currently running when a "stop" command is issued, the program returns "Sensor already stopped"

In addition, the resume command simply returns "Sensor has not been paused" if the sensor is not in paused state; and the "pause" command simply returns "Sensor is not running" if the sensor is not in the run state.

SWSMAC-279 "GPSListener does not have a configuration to automatically start saving data"

Description

GPSListener has a "start" command (GPS START) that begins collecting raw GPS data; but there is no way to automatically start collecting on startup. If you want to do that, you have to modify the startup script to wait for a little after launching the program and then use ServerExec to call back and issue a GPS START. That's inconvenient. The XML configuration should include a way of automatically starting collection.

Root Cause

It was thought that Customers would be willing to modify scripts, but it doesn't seem so, in retrospect.

Resolution

A new command (GPS AUTOSTART ON or GPS AUTOSTART OFF) was added; the default state of AUTOSTART is OFF. This setting state is saved in an XML file (GPSListener-persistent-config.xml); this file updates a corresponding MD5 signature whenever its contents are changed, which verifies its integrity. The settings revert to default values if the file cannot be read and verified.

Another setting in the persistent configuration file is the default output filename (GPS DEFAULTOUTPUT filename-and-path), which defaults to "sensor.out"; users may wish to change it to the full path and name that they want to use.

Note that both of these commands can be executed via ServerExec or by targeting the appropriate address (typically 0x20F1) For example:

ServerExec GPSListener 'GPS DEFAULTOUTPUT /mnt/XDATA/MAM/GPSData.txt'

ServerExec GPSListener 'GPS AUTOSTART ON'

Note that enabling auto-start doesn't start the processing immediately: it starts logging automatically on the next system startup.

SWSMAC-289 "Datawell: data queue is missing every other sample"

Description

The high frequency data queue is missing every other sample. That causes the calculations of the spectrum and the wave dimensions to be incorrect.

Root Cause

Characterization of the sensor did not note that lines of output are bunched together in bursts rather than being sent as necessary. In addition, there was a logic error that assumed that bursts of output would include only one line.

Resolution

In function ReceiveSerialData, the received string is split into substrings (lines), and then each line of output is fed into ParseSample.

SWSMAC-291 "SMC script files missing documentation"

Description

SMC script files are missing inline documentation, which should explain usage of the script and how it works internally. The following scripts should be updated:

catgps
ConfigOctalUartLine
ConfigurePower
EnableEthernet
InitOctalUartLines
killsmc
SetPowerSwitch

Root Cause

Oversight

Resolution

Documentation was added to the files; for the scripts that require to be run as root, added a check for root user

Dependencies

This release expects to run on the kernel build 5.0.2 (P/N 02563-04), which adds support for encrypted filesystems.

If encryption is to be used, install the "encrypted startup" software (P/N 03209-XX) to receive the encryption key and unlock the encrypted filesystem. The program and its data are maintained on the encrypted partition.

The SMC is powered by a PIB and uses it for communication to the Float C&C. The PIB must be loaded with firmware version 2.0.335 (P/N 02639-01) or later. Note that some versions of PIB must be specifically configured with support for the SMC.

Support for Write Console Script was introduced in Float C&C release 1.5.826 (P/N 00655-05)

Open Issues

SWSMAC-239 "Ethernet activity interfering with received serial data"

Disposition:

This started off as a bug related to using Ethernet while receiving serial data, but it now seems to be a problem with the SMC kernel and/or the serial driver. It appears that if the CPU goes idle during serial receive, sometimes it doesn't exit idle mode before dropping bytes.

Workaround:

- Add "nohz=off" to the uboot settings. To do this:

1. Power up the SMC with the console cable installed
2. When Logic Loader starts up, it prints out status messages and then starts a count-down from 1-second: hit the Enter key to stop the count-down.
3. At the Logic Loader prompt, enter "printenv" (no quotes) and note the current value of the **bootargs** variable; for example:

```
=> printenv
baudrate=115200
loadaddr=0x81000000
rootfsaddr=0x81300000
consoledev=ttyS0
rootpath=/opt/nfs-exports/ltib-omap
ramdisksize=89000
kernelimage=uImage
nfsoptions=,wsize=1500,rsize=1500
nfsboot=setenv bootargs display=${display} console=${consoledev},${baudrate}
root=/dev/nfs rw nfsroot=${serverip}:${rootpath}${nfsoptions} ip=dhcp
${otherbootargs};tftpboot ${loadaddr} ${kernelimage};bootm ${loadaddr}
ramboot=setenv bootargs display=${display} console=${consoledev},${baudrate}
root=/dev/ram rw ramdisk_size=${ramdisksize} ${otherbootargs};tftpboot ${loadaddr}
${kernelimage};tftpboot ${rootfsaddr} rootfs.ext2.gz.uboot;bootm ${loadaddr}
${rootfsaddr}
xipboot=setenv bootargs display=${display} console=${consoledev},${baudrate}
root=/dev/ram rw ramdisk_size=${ramdisksize} ${otherbootargs};bootm ${loadaddr}
${rootfsaddr}
rootdevice=/dev/mtdblock4
rootfstype=yaffs
mtddbboot=setenv bootargs display=${display} console=${consoledev},${baudrate}
root=${rootdevice} rootfstype=${rootfstype} rw ${otherbootargs};bootm ${loadaddr}
sdmtddbboot=setenv bootargs display=${display} console=${consoledev},${baudrate}
root=${rootdevice} rootfstype=${rootfstype} rw ${otherbootargs};mmcinit;fatload
mmc0 ${loadaddr} ${kernelimage}; bootm ${loadaddr}
dieid#=70540024000000000040365f80c00e021
ethact=smc911x-0
display=15
bootcmd=run sdmtboot2
sdmtboot2=mmc init;fatload mmc 0 0x81000000 uImage;bootm 0x81000000
bootdelay=1
bootargs0=console=ttyS0,115200n8 display=15 ip=dhcp root=/dev/mmcbk0p2
rootfstype=ext3 rootdelay=1 rw
mtdids=nand0=nand
stdin=serial
stdout=serial
stderr=serial
ethaddr=41:00:00:00:95:06
bootargs=console=ttyS0,115200n8 display=15 root=/dev/mmcbk0p2 rootfstype=ext3
rootdelay=1 rw
```

```
Environment size: 1695/131068 bytes  
=>
```

4. Modify the **bootargs** variable to include the string "nohz=off":

```
=> setenv bootargs console=ttyS0,115200n8 display=15 nohz=off root=/dev/mmcblk0p2  
rootfstype=ext3 rootdelay=1 rw  
=>
```

5. Enter "printenv" again and confirm the new value of **bootargs**

6. Enter "saveenv" to save the data

7. Enter "reset" to restart the SMC

- Use faster baud rates when possible: this is not 100% effective, but it reduces the frequency of the problem (possibly because the CPU idles less.)

Planned Action:

LRI is investigating how to patch the kernel to address this problem.

End-user Impacts

Customers updating from earlier builds must re-format the SD card (or use a fresh one) and install the 5.0.2 kernel before installing this build.

Customers using encryption must either manually send a command to start the SMC application or use automation software to do so. WGMS 4.0.3 is scheduled to have this capability.

Support Impacts

For Customers using encryption, a 0x71 status message will report when the SMC is waiting for a decryption key; this will trigger creation of an Encryption Partition part in the parts tree (starting in WGMS 4.0.3.) The decryption command must be added to this part in order to automatically decrypt the partition and allow the SMC software to start. In the case of suspicious activity, this automated response should be disabled by removing the decryption command (i.e. the SMC will not be allowed to start on the next reboot.)

Changes in Version 1.1.7 (04/16/2012, P/N 02617-11)

New Features

N/A

Bug Fixes

SWSMAC-267 "MAM Directional Parameters never cleared"

Dependencies

This release expects to run on the kernel build 5.0.2 (P/N 02563-04), which adds support for encrypted filesystems.

If encryption is to be used, install the "encrypted startup" software (P/N 03209-XX) to receive the encryption key and unlock the encrypted filesystem. The program and its data are maintained on the encrypted partition.

The SMC is powered by a PIB and uses it for communication to the Float C&C. The PIB must be loaded with firmware version 2.0.335 (P/N 02639-01) or later. Note that some versions of PIB must be specifically configured with support for the SMC.

Support for Write Console Script was introduced in Float C&C release 1.5.826 (P/N 00655-05)

Open Issues

SWSMAC-238 "GPSTListener sets the hardware clock too frequently"

Disposition:

Setting the hardware clock via the i²c bus too frequently (especially with a fast GPS) can create a delay in updating the clock, which causes the time to be updated continually and generates kernel warning messages.

Workaround:

Reconfigure GPSTListener's hwClockVerb setting to be empty (don't update the clock)

Planned Action:

In the next release, the hardware clock will only be set the first time there is a valid GPS after SMC software startup, which allows for periodic synchronization of the date and time, and GPSTListener will be corrected so that the clock setting action does not create a delay.

End-user Impacts

Customers updating from earlier builds must re-format the SD card (or use a fresh one) and install the 5.0.2 kernel before installing this build.

Customers using encryption must either manually send a command to start the SMC application or use automation software to do so. WGMS 4.0.3 is scheduled to have this capability.

Support Impacts

For Customers using encryption, a 0x71 status message will report when the SMC is waiting for a decryption key; this will trigger creation of an Encryption Partition part in the parts tree (starting in WGMS 4.0.3.) The decryption command must be added to this part in order to automatically decrypt the partition and allow the SMC software to start. In the case of suspicious activity, this automated response should be disabled by removing the decryption command (i.e. the SMC will not be allowed to start on the next reboot.)

Changes in Version 1.1.6 (03/26/2012, P/N 02617-10)

New Features

SWSMAC-136 "Datawell: create MOSESimulator"

SWSMAC-172 "Datawell: report error in case the sensor doesn't receive enough samples"

SWSMAC-174 "LuaWrapper should allow the lua script to report and receive events generically"

SWSMAC-178 "ADCP: WHAM should generate a Telemetry event containing ADCP status information"

SWSMAC-197 "ADCP: WHAM should perform ADCP compass calibration on demand"

SWSMAC-200 "Datawell: create NDBC spectrum from Datawell's spectrum"

SWSMAC-201 "Datawell: new event for the raw data"

SWSMAC-203 "Create Simulator for DataWell"

SWSMAC-204 "Datawell: create a new event for the Datawell's spectrum"

SWSMAC-205 "Datawell: change the array param to vector"

SWSMAC-206 "Datawell: process data more efficiently according to the time interval"

SWSMAC-220 "Datawell: log data with GPS info"

SWSMAC-221 "Datawell: new output commands for processed data, datawell spectrum, and NDBC spectrum files"

Bug Fixes

SWSMAC-209 "Datawell: unintentional start up during power off"

SWSMAC-210 "Datawell: vector out of range exception"

SWSMAC-211 "Datawell: unable to send a telemetry at 0 minute of the hour"

SWSMAC-212 "Datawell: Every-minute thread caught exception"

SWSMAC-231 "EventManager does not filter events based on subscription"

SWSMAC-236 "Towfish GPS distribution has incorrect month and century"

SWSMAC-237 "Towfish GPS distribution should not be default on/enabled"

Dependencies

This release expects to run on the kernel build 5.0.2 (P/N 02563-04), which adds support for encrypted filesystems.

If encryption is to be used, install the "encrypted startup" software (P/N 03209-XX) to receive the encryption key and unlock the encrypted filesystem. The program and its data are maintained on the encrypted partition.

The SMC is powered by a PIB and uses it for communication to the Float C&C. The PIB must be loaded with firmware version 2.0.335 (P/N 02639-01) or later. Note that some versions of PIB must be specifically configured with support for the SMC.

Support for Write Console Script was introduced in Float C&C release 1.5.826 (P/N 00655-05)

Open Issues

SWSMAC-238 "GPSTListener sets the hardware clock too frequently"
--

Disposition:

Setting the hardware clock via the i²c bus too frequently (especially with a fast GPS) can create a delay in updating the clock, which causes the time to be updated continually and generates kernel warning messages.

Workaround:

Reconfigure GPSTListener's hwClockVerb setting to be empty (don't update the clock)

Planned Action:

In the next release, the hardware clock will only be set the first time there is a valid GPS after SMC software startup, which allows for periodic synchronization of the date and time, and GPSTListener will be corrected so that the clock setting action does not create a delay.

End-user Impacts

Customers updating from earlier builds must re-format the SD card (or use a fresh one) and install the 5.0.2 kernel before installing this build.

Customers using encryption must either manually send a command to start the SMC application or use automation software to do so. WGMS 4.0.3 is scheduled to have this capability.

The CORBA IDL for EventManager was modified to allow clients to subscribe to events rather than discarding unwanted events that are forwarded. EventManager's log now shows the names of the clients to which events are being distributed.

LuaWrapper now supports an additional callback method for generic events received (SMCReportEvent) and a new SMC command to send events (SMC:PublishEvent); see below.

MAM now supports additional commands to direct Datawell processed data, spectrum, and NDBC-style spectrum to files.

Support Impacts

For Customers using encryption, a 0x71 status message will report when the SMC is waiting for a decryption key; this will trigger creation of an Encryption Partition part in the parts tree (starting in WGMS 4.0.3.) The decryption command must be added to this part in order to automatically decrypt the partition and allow the SMC software to start. In the case of suspicious activity, this automated response should be disabled by removing the decryption command (i.e. the SMC will not be allowed to start on the next reboot.)

Changes in Version 1.1.5 (02/03/2012, P/N 02617-09)

New Features

SWSMAC-169 "Add GPS readings to LuaWrapper"

SWSMAC-171 "Create SMC monitor task"

SWSMAC-196 "Create a startup program to receive encryption key and mount encrypted partition"

Bug Fixes

SWSMAC-167 "LuaWrapper return data formatting is incomplete"

Dependencies

This release expects to run on the kernel build 5.0.2 (P/N 02563-04), which adds support for encrypted filesystems.

If encryption is to be used, install the “encrypted startup” software (P/N 03209-XX) to receive the encryption key and unlock the encrypted filesystem. The program and its data are maintained on the encrypted partition.

The SMC is powered by a PIB and uses it for communication to the Float C&C. The PIB must be loaded with firmware version 2.0.335 (P/N 02639-01) or later. Note that some versions of PIB must be specifically configured with support for the SMC.

Support for Write Console Script was introduced in Float C&C release 1.5.826 (P/N 00655-05)

Open Issues

N/A

End-user Impacts

Customers updating from earlier builds must re-format the SD card (or use a fresh one) and install the 5.0.2 kernel before installing this build.

Customers using encryption must either manually send a command to start the SMC application or use automation software to do so. WGMS 4.0.3 is scheduled to have this capability.

Support Impacts

For Customers using encryption, a 0x71 status message will report when the SMC is waiting for a decryption key; this will trigger creation of an Encryption Partition part in the parts tree (starting in WGMS 4.0.3.) The decryption command must be added to this part in order to automatically decrypt the partition and allow the SMC software to start. In the case of suspicious activity, this automated response should be disabled by removing the decryption command (i.e. the SMC will not be allowed to start on the next reboot.)

Changes in Version 1.1.4 (01/18/2012, P/N 02617-08)

New Features

N/A

Bug Fixes

SWSMAC-180 “Datawell: wave direction was off by 90 degree”

SWSMAC-192 “GPSListener does not accept differential reports”

Dependencies

This release expects to run on the kernel build 5.0.1 (P/N 02563-01)

The SMC is powered by a PIB and uses it for communication to the Float C&C. The PIB must be loaded with firmware version 2.0.335 (P/N 02639-01) or later. Note that some versions of PIB must be specifically configured with support for the SMC.

Support for Write Console Script was introduced in Float C&C release 1.5.826 (P/N 00655-05)

Open Issues

No change from the last version.

End-user Impacts

Customers updating from earlier builds must re-format the SD card (or use a fresh one) and install the 5.0.1 kernel before installing this build.

Support Impacts

No change from the last version.

Changes in Version 1.1.3 (12/15/2011, P/N 02617-07)

New Features

SWSMAC-173 "Interface Vemco VR2C sensor to report tagged marine life (demo implementation)"

Bug Fixes

n/a

Dependencies

This release expects to run on the kernel build 5.0.1 (P/N 02563-01)

The SMC is powered by a PIB and uses it for communication to the Float C&C. The PIB must be loaded with firmware version 2.0.335 (P/N 02639-01) or later. Note that some versions of PIB must be specifically configured with support for the SMC.

Support for Write Console Script was introduced in Float C&C release 1.5.826 (P/N 00655-05)

Tag detections sent as telemetry reports will require the VemcoVR2C Tag Report parsing in WGMS to properly display the formatted report.

Open Issues

No change from the last version.

End-user Impacts

This is an interim release targeted for a demo implementation of the Vemco VR2C sensor interface and Tag detection reporting: it is not recommended for general release.

Customers updating from earlier builds must re-format the SD card (or use a fresh one) and install the 5.0.1 kernel before installing this build.

SWSMAC-173 introduces the first implementation of an interface to the Vemco VR2C sensor. See below for notes.

Support Impacts

No change from the last version.

Changes in Version 1.1.2 (11/16/2011, P/N 02617-05)

New Features

SWSMAC-90 "Create a lua scripting wrapper with features to implement Sonardyne modem interface"

SWSMAC-96 "Datawell MOSE / MAM items left"

SWSMAC-154 "Set the hardware clock from GPSTListener"

SWSMAC-156 "Datawell: new command MOSE --run to run a script"

SWSMAC-157 "Datawell: rename command options"

Bug Fixes

SWSMAC-118 "Datawell: Missing MOSE --gpsstart | --gpsstop | --gpsoutput"

SWSMAC-123 "Datawell: nested MOSE --run scriptFile execute commands in wrong order"

SWSMAC-125 "BenthosModem module telemetry may be unnecessarily curtailed"

SWSMAC-127 "Datawell: MOSE --output filename doesn't work in script"

SWSMAC-129 "Datawell: MOSE --off doesn't turn power off"

SWSMAC-139 "Datawell: unnecessarily repeated error message in the response of the MOSE -run <script file>"

SWSMAC-140 "Datawell: imprecised error message description of the non-existing file"

SWSMAC-142 "Datawell: imprecised description of missing param values"

SWSMAC-143 "Datawell: missing command echo in the NAK response of mose --setconfig with extra param values"

SWSMAC-144 "Datawell: add a new parser to handle comments at the end of the command line"

SWSMAC-145 "PayloadInterface does not respect the 'no ack' flag for console messages"

SWSMAC-146 "Datawell: open the 2nd serial port only when run MOSE -gpsstart"

SWSMAC-147 "Datawell: 2 telemetry packets are sent at every top of the hour"

SWSMAC-148 "Continual unhandled PRCM interrupts reported on PCX-1 (SMC# V38591)"

SWSMAC-149 "BenthosModem hangs when BAM BREAK is issued with power off"

SWSMAC-150 "BenthosModem needs a way to ignore application commands when it is establishing range"

SWSMAC-151 "Datawell: MOSE --run should be able to execute without a script name"

SWSMAC-152 "Linux should use the RTC values when booting"

SWSMAC-153 "Standard timelimit utility does not kill processes by hierarchy, doesn't enforce time limits in some cases"

Dependencies

This release expects to run on the kernel build 5.0.1 (P/N 02563-01)

The SMC is powered by a PIB and uses it for communication to the Float C&C. The PIB must be loaded with firmware version 2.0.335 (P/N 02639-01) or later. Note that some versions of PIB must be specifically configured with support for the SMC.

Support for Write Console Script was introduced in Float C&C release 1.5.826 (P/N 00655-05)

Open Issues

SWSMAC-11 "Octal UART driver (lirixr16v798) does not sleep when all ports are closed"

Disposition:

This results in less optimal power consumption than desired, but it doesn't cause misoperation.

Planned Action:

Address in a future release.

SWSMAC-79 "BenthosModem interface does not allow for asynchronous command completion"

Disposition:

The current clients of the BenthosModem interface do not require or expect asynchronous command completion, so this is not considered to be a serious problem.

Planned Action:

Address in a future release.

SWSMAC-104 "WetLabsSensor does not retry after comm timeouts for the --start command"

Disposition:

This may cause misoperation occasionally, but it's still possible to reissue the command.

Workaround:

Reissue the start command if spurious communication error is suspected.

Planned Action:

Address in a future release.

SWSMAC-167 "LuaWrapper return data formatting is incomplete"
--

Disposition:

For the LuaWrapper process, the return data comes as XML, but the return formatting is inconsistent.

Workaround:

For data returned via file transfer (SBD or RUDICS), this is not an important feature.

Planned Action:

Address in the next release.

End-user Impacts

This is an interim release targeted for use by Sonardyne developers: it is not recommended for general release.

Customers updating from earlier builds must re-format the SD card (or use a fresh one) and install the 5.0.1 kernel before installing this build.

SWSMAC-90 introduces the first implementation of LuaWrapper. See below for notes.

SWSMAC-153 creates the **timelimit2** utility, which has a different syntax from **timelimit**. The release includes changes to PPPHost files to use this new utility, but server-side files should also be retrofitted. See below for notes on **timelimit2**.

Support Impacts

This is a required upgrade for SMC's without direct GPS feed, such as the towfish configuration.

Changes in Version 1.1.1 (11/11/2011, P/N 02617-04)

New Features

SWSMAC-164 "Add new functionality to distribute a GPS event in PayloadInterfaceController"

Bug Fixes

n/a

Dependencies

This release expects to run on the kernel build 5.0.1 (P/N 02563-01)

The SMC is powered by a PIB and uses it for communication to the Float C&C. The PIB must be loaded with firmware version 2.0.335 (P/N 02639-01) or later. Note that some versions of PIB must be specifically configured with support for the SMC.

Support for Write Console Script was introduced in Float C&C release 1.5.826 (P/N 00655-05)

Open Issues

No change from the last version.

End-user Impacts

This is a required upgrade for SMC's without direct GPS feed, such as the towfish configuration.

Customers updating from earlier builds must re-format the SD card (or use a fresh one) and install the 5.0.1 kernel before installing this build.

SWSMAC-164 changes the XML-based configuration of PayloadInterface process, which can be set to distribute GPS from the Float's polling message. In this case, it should also be configured to set the time based on the Float's polling message.

Support Impacts

This is a required upgrade for SMC's without direct GPS feed, such as the towfish configuration.

Customers updating from other builds should verify that the kernel is 5.0.1. If a kernel change is necessary, the SD card must be replaced with one formatted with the correct kernel or the existing card must be reformatted, and the software must be re-installed.

Changes in Version 1.1.0 (10/13/2011, P/N 02617-03)

New Features

SWSMAC-83 "Create new command for Wet Labs interface to enable/disable the verbose response"

SWSMAC-84 "Wet Labs interface should allow --start command to be tagged with timeout"

SWSMAC-89 "Add support for console message to SMC PayloadInterface"

SWSMAC-91 "Create simulator for Wet Labs sensors (ECO Puck and CStar)"

SWSMAC-101 "Create a serial port simulator for the Eco Puck"

SWSMAC-113 "Implement Data Recorder application on the SMAC"

SWSMAC-120 "Add XML load and save with MD5 verification (allows for reliable data persistence)"

Bug Fixes

SWSMAC-2 "Items missing from Linux build 1 for Zoom"

SWSMAC-60 "PPPHost and scp do not time out, hangs RUDICS communication"

SWSMAC-63 "Commands sent to SMAC process (camera client-XBee) result in PayloadInterface/StorageController exception."

SWSMAC-66 "PayloadInterface exception when sending command from CameraClient to 0x2012"

SWSMAC-71 "WetLabsSensor - multiple commands sent as a script are not being parsed and processed."

SWSMAC-87 "WetLabsSensor --getconfig following --setavg in script creates race condition"

SWSMAC-93 "Add new user command '--setconfig' and separate it from '--setavg' and '--setpkt' to WetLabsSensor"

SWSMAC-94 "WetLabsSensor response error with wrong command name"

SWSMAC-98 "WetLabsSensor user command's responses are not being logged"

SWSMAC-99 "WetLabsSensor stop command, when the sensor is already in stop state, returns incorrect response"

SWSMAC-100 "CSTAR telemetry report contained two samples, but the repeat count was 1"

SWSMAC-102 "SetPowerSwitch returns invalid result"

SWSMAC-103 "WetLabsSensor does not remember latest script file from --run <scriptFilename>"

SWSMAC-105 "WetLabsSensor --start command response has GPS data appended twice"

SWSMAC-108 "RUNSHELL does not return all of the expected data"

SWSMAC-109 "Pseudo-terminals not usable with SMC serial port interface class"

SWSMAC-110 "WetLabsSensor: --verbose off removes GPS info from samples in log files"

SWSMAC-114 "Limit log files and data output files offering options for file sets and rollover"

Dependencies

This release expects to run on the kernel build 5.0.1 (P/N 02563-01)

The SMC is powered by a PIB and uses it for communication to the Float C&C. The PIB must be loaded with firmware version 2.0.335 (P/N 02639-01) or later. Note that some versions of PIB must be specifically configured with support for the SMC.

Support for Write Console Script was introduced in Float C&C release 1.5.826 (P/N 00655-05)

Open Issues

SWSMAC-11 "Octal UART driver (lirixr16v798) does not sleep when all ports are closed"

Disposition:

This results in less optimal power consumption than desired, but it doesn't cause misoperation.

Planned Action:

Address in a future release.

SWSMAC-79 "BenthosModem interface does not allow for asynchronous command completion"

Disposition:

The current clients of the BenthosModem interface do not require or expect asynchronous command completion, so this is not considered to be a serious problem.

Planned Action:

Address in a future release.

SWSMAC-104 "WetLabsSensor does not retry after comm timeouts for the --start command"

Disposition:

This may cause misoperation occasionally, but it's still possible to reissue the command.

Workaround:

Reissue the start command if spurious communication error is suspected.

Planned Action:

Address in a future release.

SWSMAC-125 "BenthosModem module telemetry may be unnecessarily curtailed"

Disposition:

This is a serious reduction in functionality and should be addressed in a near-term release.

Planned Action:

Address in the next release.

End-user Impacts

This is a critical upgrade since it contains bug fixes that address crashing conditions.

Customers updating from earlier builds must re-format the SD card (or use a fresh one) and install the 5.0.1 kernel before installing this build.

SWSMAC-89 introduces the ability to send commands to SMC components using the WGMS Write Console Script message in place of the file transfer (File Write Init) method. This is a faster message exchange, but there are some limitations:

- The command sent to the SMC is contained in a single packet (~200 bytes) and is text.
- When commands are sent using function code 0, multiple responses to the message are allowed, but the number of packets is limited to 4 to avoid overwhelming the Float C&C's buffering capacity.
- Commands sent using function code 1 respond using a file transfer, which allows for a longer response.

Note that commands can still be sent via file transfer.

The fix for SWSMAC-60 relies on the standard **timelimit** utility program being used to control the runtime of scp or dbclient. Scripts used with the RUDICS interface or other Customer-provided scripts should be modified to make use of it and the timeouts should be configured appropriately for the Customer's use case. For example, **runscp** (part of the PPPHost configuration) has been modified as follows to limit scp to 10 minutes:

```
exec timelimit -pq -s 9 -t 600 scp $IDENT -P 10501  
vehicle@12.156.66.48:$DIRPART/runme workspace/runme
```

In this build, **timelimit** appears in the /home/smc/bin directory, but it will be included in the root filesystem build in a future release.

The fix for SWSMAC-114 introduces a new utility called Logger, which replaces use of the standard **tee** utility in the startup scripts. For example, the startup script for PayloadInterface has been modified as follows to keep up to 5 log files (including backups) of up to 10 Mb each:

Before:

```
#!/bin/sh  
PRG=PayloadInterface  
LOGFILE=~/.log/$PRG.log  
  
TEECMD="tee -a $LOGFILE"  
if [ -z "$1" ]; then TEECMD="$TEECMD > /dev/null" ; fi  
  
echo Starting $PRG  
ReRun 20 "$PRG 2>&1 | $TEECMD" &
```

After:

```
#!/bin/sh  
PRG=PayloadInterface  
  
echo Starting $PRG  
ReRun 20 "$PRG 2>&1 | Logger --work-dir=$HOME/log --prefix=\"$PRG\" --  
extension=log --max-file-size=10000000 --max-files=5 --keep-writes-together=1" &
```

Support Impacts

This is a critical upgrade since it contains bug fixes that address crashing conditions.

See notes above for the availability of the Write Console Script command and changes to RUDICS scripts.

Customers updating from other builds should verify that the kernel is 5.0.1. If a kernel change is necessary, the SD card must be replaced with one formatted with the correct kernel or the existing card must be reformatted, and the software must be re-installed.

Changes in Version 1.0.10 (8/19/2011, P/N 02617-02)

New Features

SWSMAC-42 "Develop Benthos modem interface (ADCP use case)"

SWSMAC-43 "Modify ADCP SMC module to interact with Benthos modem"

SWSMAC-49 "ECOSensor process should turn off power switch during shutdown"

SWSMAC-52 "Incorporate power switch into ADCP process"

SWSMAC-58 "Retry read in case the EcoPuck sensor is in sleep mode or fails communication"

SWSMAC-64 "Merge Eco Puck sensor and CStar sensor into WetLabs sensors"

SWSMAC-65 "Add capability of configuring debug level in XML file and the ability to specify loggers with different filter levels"

SWSMAC-67 "Add wave spectrum telemetry reports to MAM"

Bug Fixes

SWSMAC-6 "Error "instantiated from 'void boost::shared_ptr<...>' reported (Boost 1.41.0 issue)"

SWSMAC-7 "Zoom/SMC root filesystem build complains of "instantiated from 'void ...' from serialization methods (Boost 1.41.0)"

SWSMAC-37 "cron utility is not part of the SMC linux build"

SWSMAC-44 "Throwing CORBA exceptions may be causing process crashes"

SWSMAC-54 "populate_smc script gives errors and does not set permissions properly if there are spaces in filenames"

SWSMAC-59 "Reactivate throw CORBA exception in EcoPuck interface"

SWSMAC-61 "ADCP sensor commands file transfers failing with updated kernel build"

SWSMAC-68 "BenthosModem module crashes when executing 'BAM CMD AT\$BN100' on the target"

SWSMAC-69 "BenthosModem module does not adequately handle the output from AT\$BS or ATBS"

SWSMAC-70 "WetLabsSensor - Asynchronous command completion not working properly"

SWSMAC-72 "WetLabsSensor - responses to all commands in a script should be returned as a single concatenated response."

SWSMAC-73 "PayloadInterface telemetry GPS format is not compatible with other Float C&C reports"

SWSMAC-75 "For the ADCP Benthos Modem integration, do not correct ensembles for vehicle motion"

SWSMAC-76 "BenthosModem should (optionally) retry commands to a remote modem when there is no response"

SWSMAC-77 "BenthosModem interface should provide a simplified data recorder transfer upload process"

SWSMAC-78 "SMC software distribution does not reference licenses for included software"

SWSMAC-86 "BenthosModem and MAM processes use cron in a way that can lead to runaway process creation"

Dependencies

This release expects to run on the kernel build 5.0.1 (P/N 02563-01)

The SMC is powered by a PIB and uses it for communication to the Float C&C. The PIB must be loaded with firmware version 2.0.335 (P/N 02639-01) or later.

Open Issues

SWSMAC-11 "Octal UART driver (Irixr16v798) does not sleep when all ports are closed"

Disposition:

This results in less optimal power consumption than desired, but it doesn't cause misoperation.

Planned Action:

Address in a future release.

SWSMAC-60 "PPPHost and scp do not time out, hangs RUDICS communication"

Disposition:

The flavor of scp provided by the Linux dropbear package does not support a configurable timeout and does not time out appropriately on its own when the PPP session drops. As a result, scp (and ssh/dbclient) can hang and cause PPPHost to stop its periodic call-ins.

Workaround:

Monitor call-ins and restart the SMC if necessary by cycling power to the appropriate payload port.

Planned Action:

Develop a wrappers for scp and dbclient that support setting a timeout value and kill the process if the time limit is exceeded: next release.

SWSMAC-63 Commands sent to SMAC process (camera client-XBee) result in PayloadInterface/StorageController exception.

SWSMAC-66 "PayloadInterface exception when sending command from CameraClient to 0x2012"

Disposition:

Both SWSMAC-63 and SWSMAC-66 were caused by an incorrect implementation of asynchronous command completion in the Wet Labs (Eco Puck/CStar) interface. That implementation has been addressed, but the PayloadInterface process should be modified so that it doesn't have this vulnerability.

Planned Action:

Address in the next release

SWSMAC-71 "WetLabsSensor - multiple commands sent as a script are not being parsed and processed"

Disposition:

If command text is sent with multiple lines, they are not being broken up and processed as though they were a script file.

Workaround:

Send a script file and execute it with the `-run` command.

Planned Action:

Address in the next release

SWSMAC-79 "BenthosModem interface does not allow for asynchronous command completion"

Disposition:

The current clients of the BenthosModem interface do not require or expect asynchronous command completion, so this is not considered to be a serious problem.

Planned Action:

Address in a future release.

End-user Impacts

This is a critical upgrade since it contains bug fixes that address crashing conditions.

Customers updating from earlier builds must re-format the SD card (or use a fresh one) and install the 5.0.1 kernel before installing this build.

Support Impacts

This is a critical upgrade since it contains bug fixes that address crashing conditions.

Customers updating from earlier builds must re-format the SD card (or use a fresh one) and install the 5.0.1 kernel before installing this build.

Changes in Version 1.0.9 (5/24/2011, P/N 02617-01)

New Features

SWSMAC-48 "Add Datawell GPS logging to MAM"

Bug Fixes

SWSMAC-46 "omniNames log files can become corrupt and prevent SMC from starting"

Dependencies

This release expects to run on the kernel build 2.1.4 (P/N 02562-05)

The SMC is powered by a PIB and uses it for communication to the Float C&C. The PIB must be loaded with firmware version 2.0.335 (P/N 02639-01) or later.

Open Issues

SWSMAC-44 "Throwing CORBA exceptions may be causing process crashes"

End-user Impacts

This is a recommended upgrade since it contains bug fixes.

Support Impacts

This is a recommended upgrade since it contains bug fixes.

Installation/Upgrade Instructions

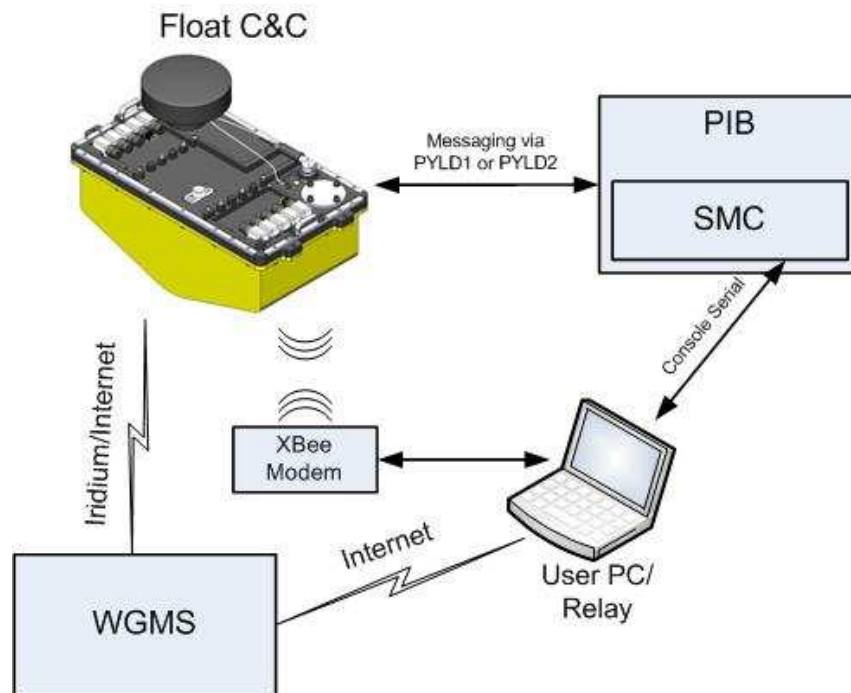


Figure 1 SMC interconnection with Float C&C and user PC (setup configuration)

WGMS (Wave Glider Management System) is a web-based application and database that provides a command and telemetry interface to vehicles. It communicates to vehicles via the Iridium network, but extends its reach via software relays.

The Float C&C is the "command and control" computer that supervises navigation and acts as a message router for commands and data for the vessel. It also implements the Iridium communication channel for access at sea, and allows for low-cost short range communication via XBee for development, setup and troubleshooting. The XBee relay (P/N 00775-XX) also exposes a terminal interface for debug message display and entering low-level commands.

The SMC implements additional communication channels, but for this setup case, communication will be via the Float C&C and XBee or via the SMC's USB-based console.

Setting up the SD Card

The SD card part (02574-XX) is created by setting up 3 partitions: BOOT, LINUX and XDATA:

- The BOOT partition contains instructions to the Logic Loader software (which is programmed on the CPU) as well as U-Boot and the Linux kernel.

- The LINUX partition contains the Linux root filesystem and any unencrypted SMC application files.
- The XDATA partition contains log files and resulting SMC data. XDATA can be configured as an encrypted filesystem, in which case the majority of the application is also loaded on XDATA.

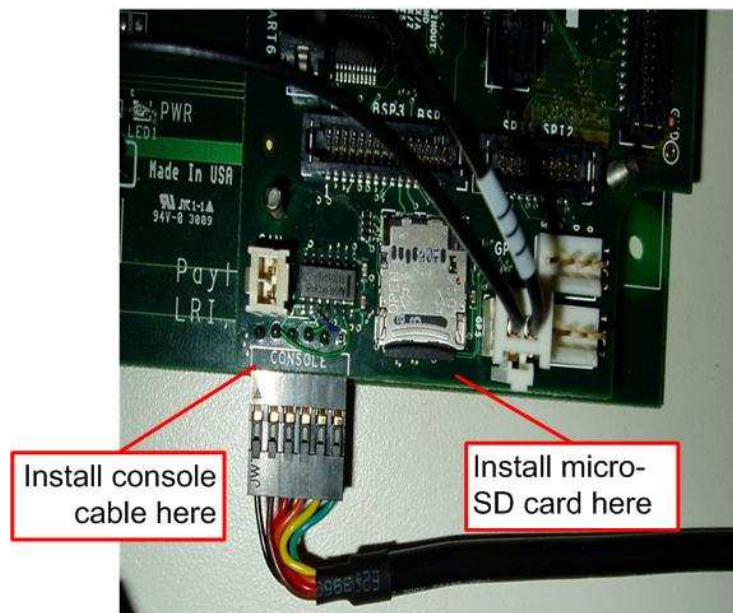
The decision of whether to encrypt the XDATA partition is typically made when creating the SD card. For systems requiring the security of an encrypted partition, care should be taken not to store anything sensitive – even temporarily – on the unencrypted (LINUX) partition.

Working with the SMC and the Float C&C

The PIB (Payload Interface Board) is a general purpose controller used to implement protocols to sensors or other computers. It has three (3) interface ports with their own power switches and serial ports. Special purpose interface boards (PCOMM's) are installed depending on the hardware protocol for each port: RS-232, RS-422 or RS-485.

The SMC can be mounted in place of one of the interface boards (two ports are consumed because of the size of the SMC.) To mount a SMC:

1. Ensure that the Float C&C payload port is powered off. This can be accomplished via the XBee Relay's debug console using command #29 (PYLD1) or #31 (PYLD2) or by issuing commands via WGMS.
2. Disconnect the PIB from the payload port (PYLD1 or PYLD2)
3. Install the SMC on PIB port J2; the PIB must be loaded with software version 2.0.335 (P/N 02639-01) or greater.
4. Reconnect the PIB to the appropriate payload port.



5. Interconnect the SMC with a PC using the USB console cable (P/N 02638)
6. Ensure that a micro-SD card with the correct kernel version has been installed in the SMC

7. Open the USB serial port provided by the console cable at 115,200 baud using a serial terminal like HyperTerminal or putty (Windows) or GkTerm, minicom or screen (Linux)
8. From WGMS, use the Set Parameter command to change the Payload Power On Enumeration Delay appropriately (110 seconds recommended.) This enables the SMC software to start up in time to respond to the Float C&C's enumeration request.
9. Enable power to the payload from the XBee Relay's debug console using command #28 (PYLD1) or #30 (PYLD2) or by issuing commands from WGMS. Momentarily, the LED's on the board turn on and messages from the Linux startup process begin to appear from the SMC console port. Finally, a Linux login prompt appears.
10. Watch the XBee Console for the response to the enumeration request. It should look something like this:

```
Description: Full PIB
Address: 0100 Major: 2 Minor: 1 Build: 368
Payload 1 Port

Description: SMC/Encrypted
Address: 2000 Major: 1 Minor: 1 Build: 5
Payload 1 Port
```

If you don't see a response from the SMC, one of the following may be true:

- There is no SMC software loaded.
- The PIB has not been configured to talk to the SMC at that particular address. See the user manual for the PIB for instructions on how to configure it.
- The wrong type or version of PIB software has been installed. In this case, copy the enumeration response and send it to LRI customer support for advice.
- The enumeration timeout is too short. Issuing the power on command again without powering it off first will restart the enumeration process; if the SMC then responds, try increasing the power on enumeration delay.

Note that it is important for the SMC to respond since the Float C&C uses enumeration responses to determine which addresses to poll. If the SMC's address is not polled, it cannot send any messages to the Float C&C.

Installing the Application

1. Before starting, put any software archives to be installed in a convenient place to be accessed. With the SD card inserted into a card reader and connected to Ubuntu:
 - For systems without an encrypted filesystem, copy the archives to a convenient location like /root

```
sudo cp tarball-filename /media/LINUX/root
sudo umount /media/*
```

- For systems using an encrypted filesystem, the archives should never be stored on the SD card unencrypted. In this case, mount the encrypted filesystem and copy the archive to the mount point

```
echo 'passphrase' | sudo cryptsetup create --cipher aes-  
cbc-essiv:sha256 --key-size 256 --hash sha256 secsd  
/dev/sdd3  
sudo mount /dev/mapper/secsd /mnt/secsd  
sudo cp tarball-filename /media/mapper/secsd  
sudo umount /mnt/secsd  
sudo cryptsetup remove secsd  
sudo umount /media/*
```

Note that the micro SD card device (/dev/sdd in my example) will vary depending on what other devices you have installed.

Be sure to wait for the umount command to complete before removing the micro SD card.

2. Insert the micro SD card into the SMC and power it up.
3. Log in as root
4. Set the password for the root account using **passwd**
5. Create the **smc** account (**useradd smc**)
6. Set the password for the **smc** account (**passwd smc**)
7. For systems using encryption, create the **smc0** account (**useradd smc0**) and set the password (**passwd smc0**)
8. For systems using encryption, mount the XDATA partition under /home/smc:

```
echo 'passphrase' | cryptsetup create --cipher aes-cbc-  
essiv:sha256 --key-size 256 --hash sha256 secsd  
/dev/mmcblk0p3  
  
mount /dev/mapper/secsd /home/smc
```
9. Install the application archive(s):

```
cd /  
tar xvpf /path-to-archive/tarball-filename
```
10. Exit the **root** user and log in as **smc**
11. For systems that do not use filesystem encryption, mount the XDATA partition and set it up:

```
sudo mount /dev/mmcblk0p3 /mnt/XDATA  
sudo bin/SetupXDATA
```
12. The startup script (/home/smc/startup) configures the transceivers for some of the serial ports for different hardware protocols. Change the line that with "InitOctalUartLines" to match the desired serial port configuration of ports /dev/ttyLX2-/dev/ttyLX7 For each port, enter 'rs232', 'rs422' or 'off'; there should be a total of six settings. For example, the following enable rs232 on /dev/ttyLX2 and /dev/ttyLX3, but /dev/ttyLX4-7 are not disabled:

```
sudo InitOctalUartLines rs232 rs232 off off off off
```

13. The `/home/smc/roots` directory contains a sub-directory for each application instance to be executed. By default, it contains one entry for each application, so the content must be customized for each shipping configuration.

```
smc@omap35x_torpedo:~/roots$ ls -l
total 76
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 001_omniNames
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 002_EventManager
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 050_PayloadManager
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 100_PayloadInterface
drwxr-xr-x 2 smc smc 4096 Aug 23 01:09 190_BenthosModem
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 200_GPSListener
drwxr-xr-x 2 smc smc 4096 Jan  1  2000 200_MAM
drwxr-xr-x 4 smc smc 4096 Aug 23 03:18 200_PPPHost
-rw-r--r-- 1 smc smc 1746 Jan  1  2000 AddressSummary.txt
```

Sample listing of configuration directories under `/home/smc/roots`

The applications are run in sorted order, and there is a three-digit prefix that is used by convention to establish run groupings:

- 001 is reserved for omniORB, which must run first so that applications can use CORBA services.
- 002 is reserved for EventManager, which must be available for applications that send or receive events.
- 050 is reserved for PayloadManager, which must be available for applications that register an enumerable name; since PayloadManager fires events to indicate when new names are registered, it must be after EventManager.

Other similar services can be registered in the 50-99 range.

- 100 is used by PayloadInterface, which implements the communication protocol with the Float C&C. Since some applications make use of PayloadInterface services, it makes sense for it to be available before they start.

Other similar services can be registered in the 100-199 range.

- 200 is used by most other applications. Some of these probably will not be used; services like GPSListener (which creates events from the GPS receiver) or PPPHost (which offers communication via RUDICS) will likely be required, depending on the hardware installed on the system.

By default, component directories contain a `“.disabled”` file that prevents them from starting. Delete this file to enable applications to start up. If you are installing a preconfigured archive, this will have already been done.

14. Additional component configuration steps are required for some applications (see below) The **startup** script loads the octal UART driver; it may be necessary to execute it and then run **sudo killsmc** to
15. Restart the system to run the default scenario (`sudo shutdown -r now`)

Changing Encryption Passphrases

Filesystem encryption works by performing all reads and writes using the passphrase provided: the data on the storage medium is actually changed. It's not like a password that lets you in the door: it's more like a filter. Therefore, changing the passphrase is easy. It means reading the entire

partition using the old passphrase to decrypt it, reformatting the partition using the new passphrase and then rewriting the partition data. For example:

1. Insert the micro SD card into to a reader connected to Ubuntu

2. Mount the old partition

```
echo 'old-passphrase' | sudo cryptsetup create --cipher  
aes-cbc-essiv:sha256 --key-size 256 --hash sha256 secsd  
/dev/sdd3
```

```
sudo mount /dev/mapper/secsd /mnt/secsd
```

3. Copy the partition data to a location elsewhere

```
pushd /mnt/secsd  
sudo tar cvf /path-to-other-location/save-archive .  
popd
```

4. Unmount the partition

```
sudo umount /mnt/secsd  
sudo cryptsetup remove secsd
```

5. Reformat the partition

```
echo 'new-passphrase' | sudo cryptsetup create --cipher  
aes-cbc-essiv:sha256 --key-size 256 --hash sha256 secsd  
/dev/sdd3
```

```
sudo mkfs.ext3 -L XDATA /dev/mapper/secsd
```

```
sudo tune2fs -c 0 /dev/mapper/secsd
```

6. Re-mount the partition

```
sudo mount /dev/mapper/secsd /mnt/secsd
```

7. Copy the partition data back:

```
pushd /mnt/secsd  
sudo tar xvf /path-to-other-location/save-archive  
popd
```

8. Unmount the partition again

```
sudo umount /mnt/secsd  
sudo cryptsetup remove secsd
```

Component Configuration

Each component or application instance to be started automatically has a corresponding subdirectory under /home/smc/roots. These directories include:

- A **startup** script that manages the startup process for the component.
- One or more XML-based configuration files specific to the executable.
- Other scripts used by the application.
- Data sub-directories or links to directories where data is stored.

The **startup** script is typically responsible for:

- Performing any initial cleanup from previous application runs.
- Executing the application, including any parameters to be passed.

- Setting up logging for applications that produce logs on standard output.
- Adding entries to the smc user's crontab to allow for cron-based callbacks.
- Performing any initial commands to configure behavior.

The XML configuration file (typically named *application-config.xml*) contains various settings like:

- Serial port name and parameters for serial-based sensors.
- CORBA parameters, like the name that the application will register and the names of other applications with which it will interact.
- Command verbs (what shell command to execute in a particular circumstance, like powering a sensor on or off)
- Event names for events that will be produced and ones that the application is looking for.
- Timing and retry limits

Most of these parameters will work without modification, but some exceptions are:

- Serial port names and references to electrical switches (these should be checked vs. the wiring of your system)
- The board address associated with each process, if the SMC is not configured to report as address 2000 (hex)
- References to external systems and directories that may vary by Customer or which must be unique per vehicle
- Different use cases that affect timing values
- Event naming: in one case, component A produces "Event1" and component B processes it; in another case, component A produces "Event1" that component C processes and merges with other data and produces "Event2", which component B consumes.
- Configuration of services used in common by multiple applications.

Aquatracka (Chelsea Technologies Group) Configuration

Aquatracka implements a CORBA command and event interface to a Chelsea Aquatracka fluorimeter. This section indicates what has changed and gives some setup guidance, but please consult the LRI Aquatracka Sensor User's Guide for additional detail.

Commands

Commands to Aquatracka are sent to process name "Aquatracka" with ServerExec or it is typically assigned an address of 0x2016 (the upper byte varies with the base address of the SMC.) The available commands are:

AQUAT --start

Wakes up the sensor and begins streaming data.

AQUAT --stop

Stops the sensor from streaming data (sensor is still powered on)

AQUAT --run *command-file*

Runs a set of commands in the given command file.

AQUAT [-o | --output] *output-file*

Directs raw output from the sensor to the given output file.

AQUAT --off

Switches off power to the sensor.

AQUAT --on

Switches on power to the sensor.

AQUAT --getconfig

Reads the configuration from the sensor.

AQUAT --header

Reads sensor header information (serial number and configuration constants.)

AQUAT --pause

In data collection mode, halts reporting of sensor data pending a resume command (power draw remains at data acquisition level.)

AQUAT --resume

In data collection paused mode, resumes reporting of sensor data.

AQUAT --verbose

Places the sensor in verbose reporting mode.

AQUAT --noverbose

Places the sensor in non-verbose reporting mode.

AQUAT --temperature

Polls for the temperature, which is reported in degrees (C) The sensor must have already been started.

AQUAT --concentration

Polls the Fluorophor concentration (gm/cc); the sensor must have already been started.

AQUAT --calibtemp

Turns off power to the sensor by executing the power off verb from the XML file.

AQUAT -calibref *n1 n2 n3*

Writes calibration reference values into Calibration-config.xml

AQUAT -calibsig *n1 n2 n3*

Writes signal calibration constants into Calibration-config.xml

AQUAT -calibserial *newvalue*

Writes sensor serial number into Calibration-config.xml

AQUAT -calibtemp *n1 n2 n3*

Writes temperature calibration constants into Calibration-config.xml

SELECT [-t | --time | -s | --samples | -d | --distance] *N*

Configures the way that samples are selected for export (see EXPORT command)

EXPORT [-count] [-o | --output *output-filename*]

Configures the name of the file where selected samples are written.

BenthosModem Configuration

BenthosModem implements a command interface and a pass-through data pseudo-port for use with Benthos acoustic modems. There are at least three use cases covered:

1. Acoustic range detection and communication test: this is initiated by the 'BAM START-RANGING *target-address ping-interval*' command (terminated by 'BAM STOP-RANGING') and results are reported via telemetry. This operation may be needed to locate the sea floor modem and calibrate communication parameters.
2. Pass-through mode: a client application sends and receives data via the local modem and the acoustic link to device connected to a remote (sea floor) modem. Data to be sent to the remote device are written to a pseudo port; data received from the remote device can be read from a pseudo port.
3. Collection of recorded data: a client program or script issues commands like 'BAM READDATARECORDER *target-address block-range*' to retrieve data stored in a remote (sea floor) modem's data recorder; this data has been passively-received and stored by a sea floor modem operating in low power mode.

Configuring the Client Program

For clients that operate in pass-through mode, verify that the pseudo-port name (typically /home/smc/roots/190_BenthosModem/pseudotty) is used in place of a serial port. In addition, the client program will need to invoke commands (for example, 'BAM CONNECT *target-address*') on the BenthosModem process in order to establish connection.

Loading the Modem Firmware

The BenthosModem process was written and tested with modem firmware version 8.2.2. Note that sea floor modems using the older generation of firmware (6.x.x) are incompatible with later generations (8.x.x), so the set should be upgraded together.

To read the current version of installed firmware:

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. Wait for the chirps from the modem to indicate that it has powered up; a will be displayed indicating the current firmware version:

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

Configuration BOM P/N 02690 lists the modem firmware upgrade kits that are available to use, or contact Teledyne Benthos for instructions and additional firmware.

Configuring 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, use the following script:

```
>AT&F
Factory Reset
```

```
user:5>@prompt=1
Prompt          | 1
>ATS13=0
OK
>ATS15=1
OK                                     (startup mode: 1=online, 2=data logger)
>ATS18=nnn
OK                                     (nnn = the address of the sea floor modem)
>@strictat=ena
Entering strictat mode
StrictAT        | Ena
>AT&W
Sregisters Stored
>
```

C3 (Turner Designs) Configuration

C3 is a command and event interface to a Turner C3 fluorometer; this is a re-implementation of the PIB-based C3 interface, and the commands are kept similar for continuity.

Commands

Commands to C3 are sent to process name "C3" with ServerExec or it is typically assigned an address of 0x2023 (the upper byte varies with the base address of the SMC.) The available commands are:

```
C3 -o
      Turns on the power switch associated with the sensor

C3 -f
      Turns off the power switch associated with the sensor

C3 -r
      Alias for C3 -o (turns on the power switch associated with the sensor)

C3 -x
      Alias for C3 -f (turns off the power switch associated with the sensor)
```

Events and Telemetry

The C3Event type contains the following name-value pair items:

- "C3EventName" = C3Event_C3SampleHeader or C3Event_C3SampleData
- "Header" = header string received from sensor
- "Date" = date value (format mm/dd/yyyy) in the sample data received from sensor
- "Time" = time value (format hh:mm:ss) in the sample data received from sensor
- "DataC1" = first data item in the sample data from sensor
- "DataC2" = second data item in the sample data from sensor
- "DataC3" = third data item in the sample data from sensor
- "Pressure" = fourth data item in the sample data from sensor
- "Temperature" = fifth data item in the sample data from sensor
- "CompressedData" = all five data items packed into an array of 14 bytes as follows:
first, second, third, and fourth data items represented as 3 bytes each where the first two bytes are the mantissa value and the third byte is the decimal value * 100, followed by the fifth data item * 100 represented as a two byte value.

The PayloadInterface-telemetry-config.xml entry for Turner C3 is:

```
<Element>
  <first boardID="0x20" taskID="0x17" />
  <second>
    <key boardID="0x20" taskID="0x17" />
    <payloadType value="0x00000060" />
    <eventNameToFormatCode>
      <Element>
        <first value="C3SampleHeader" />
        <second value="0" />
      </Element>
      <Element>
        <first value="C3SampleData" />
        <second value="1" />
      </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="FC0ASCIINormal" />
              <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="C3Event_Header" />
            </fields>
          </sampleHeaderFormat>
        </second>
      </Element>
      <Element>
        <first value="1" />
        <second maxSamples="7" 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="FC0BinaryFormat9" />
              <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="C3Event_CompressedData" />
            </fields>
          </sampleHeaderFormat>
        </second>
      </Element>
    </reportParams>
  </second>
</Element>
```

Common XML file keys to edit

<serialPortSettings> for C3, the default serial port is /dev/ttyLX4.

By default it is set to communicate with a baudrate of 9600, parity NONE, 1 stopbit, characterSize 8, and flowControl NONE. Ensure that the serial connector cables are plugged into the specified serial port on the SMC.

<powerOnVerb> and **<powerOffVerb>** for C3, the command string references power switch. Ensure that the power cable is plugged into the specified power switch on the SMC.

<defaultOutfile> for storing all header and data strings received from the C3, on the SMC filesystem. The default specified is c3data.txt in the local directory (/home/smc/roots/200_C3)

C3 is a command and event interface to a Turner C3 fluorometer; this is a re-implementation of the PIB-based C3 interface, and the commands are kept similar for continuity.

DataRecorder Configuration

The DataRecorder application is a module capable of archiving data received, over a serial port, in a data file on the SMC filesystem and making it available for extraction at runtime. The DataRecorder application also provides an interface for switching a pre-specified power switch (that the data sending device may be connected to), changing the communications settings of the serial port, changing the archive data file, and pause-resume recording of the data being received over the serial port. There are two instances of the DataRecorder installed on the SMC, designed to be used independently using separate serial ports and power switches. These are setup and configured in:

~/roots/200_DataRecorder1 and ~/roots/200_DataRecorder2

DataRecorder1 instance is configured with the serverName **DataRecorder1**, and assigned the boardId 32 (hexadecimal 20) and taskId 48 (hexadecimal 30).

DataRecorder2 instance is configured with the serverName **DataRecorder2**, and assigned the boardId 32 (hexadecimal 20) and taskId 49 (hexadecimal 31)..

Common XML file keys to edit

- **serialPortSettings** for DataRecorder1, the default serial port is /dev/ttyLX6 and for DataRecorder2 the default serial port is /dev/ttyLX7. By default, both are set to communicate with a baudrate of 9600, parity NONE, 1 stopbit, characterSize 8, and flowControl NONE. Ensure that the serial connector cables are plugged into the specified serial ports on the SMC.
- **outFileName** for DataRecorder1, the default archive file to record data received from the serial port is, **recorder1.out** in the default directory ~/roots/200_DataRecorder1. For DataRecorder2, the default archive file to record data received from the serial port is, **recorder2.out** in the default directory ~/roots/200_DataRecorder2.
- **powerOnVerb** and **powerOffVerb** for DataRecorder1, the command string references power switch 3 and for DataRecorder2 the command string references power switch 4. Ensure that the power cables are plugged into the specified power switches on the SMC.

Using the DataRecorder

The DataRecorder application supports the following commands (sent on the SMAC via ServerExec using the serverName of the target instance, or from shore side applications by specifying the boardId and taskId of the target instance):

START

setup recorder in resume (recording) mode and turn on the power switch.

STOP

turn off the power and set the recording mode to pause.

PAUSE

pause recording of data being received from the serial port (read & discard)

RESUME

continue recording data received from the serial port into the active output file.

OUTPUT <filename>

record subsequent data to the file <filename>.

ON

set power switch to ON

OFF

set power switch to OFF

SETPORT <baudrate> [<parity> [<characterSize> [<stopBits>]]]

change serial port's settings.

The above commands themselves are not case sensitive, however arguments such as <filename> are used as case-sensitive names.

Example: Output /mnt/XDATA/recorder1data.out

 OUTPUT /mnt/XDATA/Archive1.TXT

 output /home/smc/roots/200_DataRecorder2/rec2.bin

Logger

The Logger utility can be used in place of **tee** to write log files from process output, and it allows limitations to be placed on the size of log files produced. The following help is available by issuing **Logger --help**:

```
Logger: Logger { ... options ... }

Options can be specified as a collection of single-letter shortcuts
preceded by a dash and followed by tokens to use as arguments or as long-
form options preceded by double-dashes with arguments formatted as
assignments.  For example:

    Logger -wc /my-partition/workfiles BZIP2 --prefix=MYLOG --
    extension=.EXT --max-files 10 --max-file-size 1000000

The 'w' and 'c' short form options are assigned '/my-partition/workfiles'
and 'BZIP2'; the --prefix option is assigned 'MYLOG'; the --extension
option is assigned '.EXT'; the --max-files option is assigned '10'; and
the --max-file-size option is assigned '1000000'

The available options are as follows:
```

-a, --archive-dir	The directory where log archives are stored (defaults to work directory)
-c, --archive-compression	The type of archive compression to use: BZIP2, GZIP or NONE
-e, --extension	The extension used for each output file (before compression)
-f, --max-files	The maximum number of files (archives+work file) to retain
-g, --purge-strategy	The strategy to use when purging files: OLDEST or NONE
-k, --keep-writes-together	Allow lines to break across files (0) or keep writes/lines together (1)
-p, --prefix	The first part of the file name for each file in the set
-q, --max-bytes-queued	The size of the inter-thread write queue in bytes (larger value delays the logging client less)
-r, --max-retain-time	The maximum number of seconds to retain archives (0=no limit)
-s, --max-file-size	The maximum size of each file (before compression)
-w, --work-dir	The directory for the work file and default for archive files; starting directory if not specified

LuaWrapper Configuration

Commands

The following commands can be sent to LuaWrapper via ServerExec or by targeting its address with Write Console Script:

LWM -list

Lists the lua instance names and the file that was loaded into each instance.

LWM --loadlua *lua-script-file instance-name*

Creates a new lua instance named for *instance-name* and initializes with the script file indicated by *lua-script-file*

LWM --unloadlua *instance-name*

Unloads the lua instance previously-loaded as *instance-name*

LWM --run *command-file*

Loads the text lines from *command-file* and executes them as a set of LuaWrapper commands.

LWSCRIPT --invoke *instance-name function-name { ...argument-list... }*

Invokes the function named *function-name* on the script that is running in the previously-loaded instance, *instance-name*

Special Variables

LuaWrapper makes the following variables and meta-tables available to the script:

- SMC, which exposes the primary API to the script
- SMContext, which exposes an invocation-specific API to the script to control when the **LWSCRIPT --invoke** command returns
- A serial port meta-table that is returned by SMC:NewSerialPort
- A timer meta-table that is returned by SMC:NewTimer

The SMC meta-table is as follows:

SMC:Log(*log-level, message*)

Outputs the *message* string to the process log at a priority (*log-level*) which may be any of: "debug", "info", "notice", "warning", "error", "critical", "alert" or "emergency"

timsp-var = SMC:NewSerialPort(*config-ID*)

Creates an instance of a serial port object based on the selected configuration (*config-ID*, a string.) The returned data type is a serial port meta-table (see below) Received data is posted to the lua script via the **SMCReportSerialData** function, if it is defined.

timer-var = SMC:NewTimer(*timer-ID, duration*)

Creates an instance of a timer object with the given ID (*timer-ID*, a string) and *duration* (a number of milliseconds.) If the timer is not deleted before the timeout expires, the script's **SMCReportTimeout** function is invoked, if it is defined.

SMC:PublishDataSet(*event-name, event-type, event-domain, data-set, content*)

Publishes a data set event with event characteristics set by *event-name*, *event-type* and *event-domain* (all strings.) In addition, the data set name (*data-set*, a string) and the *content* of the data set (a string) are provided.

<i>event-name</i>	<i>event-type</i>	<i>event-domain</i>	<i>destination</i>
PPPHostNewFileAvailable	DataSetEvent	LiquidR-SMC	WGMS via Iridium SBD
PyIdNewFileAvailable	DataSetEvent	LiquidR-SMC	WGMS via Iridium SBD

uptime-seconds = SMC:SystemUptime()

Returns the number of seconds since the SMC was started as a floating point number with a resolution of 0.01 seconds.

SMC:PublishEvent(*event-table*)

Uses *event-table* to create a CORBA event and publish it to EventManager. See the CORBA Event Table section, below.

The SMCContext meta-table is as follows:

SMCContext:AppendOutput(*data*)

Appends *data* string to the output that will be reported from the command.

SMCContext:ReportComplete()

For a command that was previously reported as asynchronous (SMCContext:SetAsynch), this indicates that the command is now complete.

SMCContext:SetAsynch()

Indicates that the invocation (LWSCRIPT --invoke) corresponding to this context variable should not return until ReportComplete() is called. This allows a more complex state machine to be implemented as a result of the lua script function being called.

The serial port meta-table (returned from SMC:NewSerialPort) is as follows:

ID = *sp-var*:GetID()

Returns the serial port configuration ID string associated with this serial port.

sp-var:Transmit(*data*)

Transmits the *data* string over the serial port.

The timer meta-table (returned from SMC:NewTimer) is as follows:

ID = *timer-var*:GetID()

Returns the timer ID associated with the timer.

CORBA Event Tables and Variable Definitions

CORBA events are used to report significant occurrences and their data to other interested processes via the EventManager interface. Each event has:

- A **name**, which indicates what has occurred.
- A **type**, consisting of a **type name** and a **domain name**; these help to categorize the event report. By convention, the SMC software uses the event **type** to define how the accompanying data is formatted.
- A series of **name/value** pairs that carry data relative to the event report.

Each of the fields above is a string with the exception of the **value**; each value can take on one of the following CORBA types:

CORBA Type ID	Meaning	Example(s) in lua
BOOLEAN	Boolean value or flag	true or false
BOOLEAN_VECTOR	array of Boolean values or flags	{ { true, false, false, true } }
DOUBLE_FLOAT	double-precision floating point	<i>number</i> or 3.141592654
DOUBLE_FLOAT_VECTOR	array of double-precision floating point	{ { 1, 2, 3, 4 } }
INT16	16-bit signed integer	{ <i>number</i> , "INT16" } or { 8193, "INT16" }
INT16_VECTOR	array of 16-bit signed integers	{ { 1, 2, 3, 4 }, "INT16_VECTOR" }
INT32	32-bit signed integer	{ <i>number</i> , "INT32" } or { -1000000, "INT32" }
INT32_VECTOR	array of 32-bit signed integers	{ { 1, 2, 3, 4 }, "INT32_VECTOR" }
INT8	8-bit signed integer	{ <i>number</i> , "INT8" } or { 5, "INT8" } or { -5, "INT8" }
INT8_VECTOR	array of 8-bit signed integers	{ { 1, 2, 3, 4 }, "INT8_VECTOR" }
NULL	nil	Nil
SINGLE_FLOAT	single-precision floating point	{ <i>number</i> , "SINGLE_FLOAT" } or { 3.14, "SINGLE_FLOAT" }
SINGLE_FLOAT_VECTOR	array of single-precision floating point	{ { 1, 2, 3, 4 }, "SINGLE_VECTOR" }
STRING	character string	"Hello, world" or { 12, "STRING" }
STRING_VECTOR	array of character strings	{ { "Mercury", "Venus", "Earth", "Mars" } }
UINT16	16-bit unsigned integer	{ <i>number</i> , "UINT16" } or { 40000, "UINT16" }
UINT16_VECTOR	array of 16-bit unsigned integers	{ { 1, 2, 3, 4 }, "UINT16_VECTOR" }
UINT32	32-bit unsigned integer	{ <i>number</i> , "UINT32" } or { 1000000, "UINT32" }
UINT32_VECTOR	array of 32-bit unsigned integers	{ { 1, 2, 3, 4 }, "UINT32_VECTOR" }
UINT8	8-bit unsigned integer	{ <i>number</i> , "UINT8" } or { 12, "UINT8" }
UINT8_VECTOR	array of 8-bit unsigned integers	{ { 1, 2, 3, 4 }, "UINT8_VECTOR" }

The generic form of a CORBA event in lua table format is:

```
{ header =      { name = "event-name",
                  typename = "event-type-name",
                  domainname = "event-domain-name" },
  properties = { prop1name = prop1value,
                  prop2name = prop2value,
                  ...
                  propnname = propnvalue } }
```

For example, to publish an event:

```
local evt = { }

evt.header = { name = "LWTelem1",
               typename = "LWTelem1",
               domainname="LqrSMAC" }

evt.properties = { }
evt.properties["Telem-BoardID"] = { 32, "UINT8" }
evt.properties["Telem-TaskID"] = { 64, "UINT8" }
evt.properties["StringMessage"] = "Hello, world!"

SMC:PublishEvent(evt)
```

To receive an event and print out the fields to the console:

```
function SMCReportEvent(eventRx)

    print("SMCReportEvent name="..tostring(eventRx.header.name))
    print("header " ..tostring(eventRx.header))
    print("header.name " ..tostring(eventRx.header.name))
    print("header.typename " ..tostring(eventRx.header.typename))
    print("header.domainname " ..tostring(eventRx.header.domainname))

    print("properties " ..tostring(eventRx.properties))

    for k,v in pairs(eventRx.properties)
    do
        if type(v[1]) == "table" then
            local s2 = ""

            for k2,v2 in ipairs(v[1]) do
                if #s2 ~= 0 then
                    s2 = s2..","
                end
                s2 = s2..tostring(k2)..":"..tostring(v2)
            end

            print("properties["..k.."] ->
"..type(v[1]).."/"..s2.." " ..tostring(v[2]))
        else
            print("properties["..k.."] ->
"..type(v[1]).."/"..tostring(v[1]).." " ..tostring(v[2]))
        end
    end
end
```

Here's a simplified way of handling the typical GPS distribution event as a generic event:

```
function SMCReportEvent(eventRx)

    if tostring(eventRx.header.name) == "GPSSample" then

        local latitude = eventRx.properties["GPS-Latitude"][1]
        local longitude = eventRx.properties["GPS-Longitude"][1]

        if latitude and longitude then
            print("gps latitude="..tostring(latitude)..
longitude="..tostring(longitude))
        end
    end
```

end

Sometimes string data will be sent out as an array of bytes (UINT8_VECTOR.)

Byte arrays can be converted to string as follows:

```
local field = eventRx.properties["some-property"]
local data = field[1][1]

if type(data)=="table" and tostring(field[2][2]) == "UINT8_VECTOR" then
    data = string.char(unpack(data))
end
```

Pre-defined Callback Functions

LuaWrapper reserves the following script functions for call-backs:

function SMCReportEvent(eventRx)

This function is called whenever an event is received and is successfully converted to a lua table (see CORBA Event Tables and Variable Definitions, above.) Note that if the Event Transceiver settings in the XML have enabled the event filter and the received event is filtered out, SMCReportEvent not called.

function SMCReportSerialData(ID,data)

This function is called whenever serial data is received according to the configuration specified in the XML and referenced when opening the port:

ID is a string matching the configuration name passed to SMC:NewSerialPort;

data is a string containing the received data.

function SMCReportTimeout(ID)

This function is called whenever a timer expires:

ID is a string matching the timer ID passed to SMC:NewTimer

XML Configuration

The configuration file (LuaWrapper-config.xml), which must be located in the startup directory (typically /home/smc/roots/200_LuaWrapper), conforms to the format of other configuration files, but the controllerSettings section is of particular interest:

```
<controllerSettings class_id="13" tracking_level="0" version="1">
  <serialConfigMap class_id="14" tracking_level="0" version="0">
    <count>1</count>
    <item_version>0</item_version>
    <item class_id="15" tracking_level="0" version="0">
      <first>config1</first>
      <second class_id="16" tracking_level="0" version="1">
        <listenerSettings class_id="17" tracking_level="0" version="0">
          <maxReadSize>4096</maxReadSize>
          <intercharacterGapTimeoutMs>50</intercharacterGapTimeoutMs>
        </listenerSettings>
        <portSettings class_id="18" tracking_level="0" version="0">
          <portName>/dev/ttyLX2</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>
        </portSettings>
      </second>
    </item>
  </serialConfigMap>
</controllerSettings>
```

The **serialConfigMap** is a dictionary of settings (**second** section) for serial ports keyed by configuration name (**first** section); the configuration name is referenced by the SMC:NewSerialPort() function when opening a serial port. Various named configurations can be created in the XML file by repeating the **item** section and varying the configuration name. Note that **count** must be set to the number of **item** sections that will appear within the **serialConfigMap**.

Sample lua script

The following is a test case that uses some of the features above. Although the scripting interface is necessarily event-oriented to keep it single-threaded, this script shows how lua coroutines can be used to implement a top-down application that sends data to a serial port and listens for responses.

```
-- "Top-down" application module using co-routines to execute
-- sporadically on I/O and timing events

function Application()
    SMC:Log("debug", "Starting Application")
    port = SMC:NewSerialPort("config1")
```

```
    local firstTime, thisTime, delta, waitTime

    thisTime = SMC:SystemUptime()
    firstTime = thisTime-1

    while true
    do
        delta = thisTime-firstTime
        waitTime = delta-math.floor(delta)

        if (waitTime >= 0) and (waitTime < 1) then
            waitTime = math.ceil(1000*(1-waitTime))
        else
            waitTime = 1000
        end

        SMC:Log("debug", "Waiting "..tostring(waitTime))
        Wait(waitTime)
        SMC:Log("debug", "Back from waiting")

        local i
        local awaitResult

        i = 1
        awaitResult = false

        while (i <= 3) and (awaitResult == false)
        do
            port:Transmit("AT\r")
            awaitResult = Await("OK\r\n", 300)
            i = i+1
        end

        thisTime = SMC:SystemUptime()
    end
end

-- Waits for the given serial sequence to be received withint the given
-- number of milliseconds; returns true if successful.
function Await(sequence,timeout)

    SMC:Log("debug", "Await sequence="..sequence.." timeout="..timeout)

    local rxd, result

    awaitTimer = SMC:NewTimer("await", timeout)
    result = false

    while (awaitTimer ~= nil) and (result == false)
    do
        rxd = coroutine.yield()
        if (rxd == sequence) then result = true end
    end

    SMC:Log("debug", "Await rxd="..rxd.." at loop exit")

    awaitTimer = nil
    collectgarbage("collect")

    SMC:Log("debug", "Await:"..tostring(result))
    return result
end

-- Delays for the given number of milliseconds
function Wait(timeout)
```

```
        waitTimer = SMC:NewTimer("wait", timeout)

        while waitTimer ~= nil
        do
            coroutine.yield()
        end
    end

end

-- Callback routine for serial data received
function SMCReportSerialData(portName,data)

    SMC:Log("debug", "Recieved "..data.." on "..portName)

    if awaitTimer ~= nil then
        awaitTimer = nil
        collectgarbage("collect")
        coroutine.resume(appRtn, data)
    end

end

end

-- Callback routine for timer expirations
function SMCReportTimeout(timerName)

    SMC:Log("debug", "SMCReportTimeout timerName="..timerName)

    if timerName == "wait" then
        waitTimer = nil
        collectgarbage("collect")
        coroutine.resume(appRtn, nil)
    elseif timerName == "await" then
        awaitTimer = nil
        collectgarbage("collect")
        coroutine.resume(appRtn, nil)
    end

    SMC:Log("debug", "exiting SMCReportTimeout")

end

end

-- Initialize the module
SMC:Log("debug", "Starting test4")
waitTimer = nil
awaitTimer = nil

-- Launch the application on file load
appRtn = coroutine.create(function()
    Application()
end)
coroutine.resume(appRtn)

SMC:Log("debug", "test4 started")
```

MAM Configuration (Datawell MOSE)

MAM (MOSE Access Manager) interface a Datawell MOSE motion sensor and implements algorithms for computing wave spectrum and related summary values. This section indicates what has changed and gives some setup guidance, but please consult the LRI Datawell Sensor User's Guide and the MOSE documentation for additional detail.

Commands

Commands to MAM are sent to process name "MOSESensor" with ServerExec or it is typically assigned an address of 0x2022 (the upper byte varies with the base address of the SMC.) The available commands are:

MOSE --getconfig

Reads the configuration from the sensor by issuing command "\$PDTWL,CFG*"

MOSE --off

Turns off power to the sensor by executing the power off verb from the XML file.

MOSE --on

Turns on power to the sensor by executing the power on verb from the XML file.

MOSE -o *file-name*

Sets the name and path of the file to which raw samples are logged.

MOSE --output *file-name*

This is a synonym for MOSE -o

1.1.6 MOSE --procout *file-name*

Sets the name and path of the file to which wave summary data is logged; this includes wave height (hs), wave average (tz), dominant wave period (tp) and wave direction (dirp.)

MOSE --reportrate *new-rate*

Configures the number of minutes between spectrum calculations (1..60)

MOSE --reset

Resets the sensor by sending the "\$PDTWL,RST*" command.

MOSE --setconfig *LFSP LFLP HLFP SA-resist*

Sends configuration parameters to the sensor:

- **LFSP** is the low frequency, short period cut off: 1 = 10s, 2 = 30s
- **LFLP** is the low frequency, long period cut off: 1 = 300s, 2 = 600s, 3 = 1000s
- **HLFP** is high frequency, long period cut off: 1 = 30s, 2 = 60s, 3 = 100s
- **SA-resist**: 0 = off, 1 = ON

All four values may be specified, asterisks may be specified for unchanged parameters, or trailing parameters may be omitted if the values are unchanged.

1.1.6 MOSE --specout *file-name*

Sets the name and path of the file to which Datawell spectrum datasets are logged.

1.1.6 MOSE --spec2out *file-name*

Sets the name and path of the file to which NDBC-compatible wave spectrum data is logged.

MOSE --start

Begins logging data to files as configured.

MOSE --stop

Stops logging data to files.

MOSE --version

Reads the MOSE software version from the sensor by issuing the "\$PDTWL,VER*" command.

Common XML file keys to edit

- **controllerSettings → sampleNum** is the count of samples to include in a spectrum sample; it is typically set to 512 and must be set to a power of 2.
- **controllerSettings → seaStateInterval** is the number of minutes between spectrum calculations; it is typically set to 60, so the spectrum measurement is taken once per hour.
- **controllerSettings → includeBadSamples** can be set to a non-zero value for debugging, but should be deployed set to zero (no bad samples are included in calculations.)
- **controllerSettings → stabilizedDuration** is the number of minutes to allow for GPS acquisition. MAM may turn the the sensor off to save power, but it must allow some startup time.
- **controllerSettings → collectingDuration** is the number of minutes to allow for collection of samples (sampleNum); this time in addition to stabilizedDuration is the minimum "on" time that MAM budgets for a spectrum collection.

PayloadInterface Configuration

PayloadInterface implements the Payload protocol in order to communicate with the Float C&C and other devices on the Float and to interact with WGMS via file transfer and telemetry reports. It doesn't typically require any configuration, except to ensure that the required telemetry settings are merged into **PayloadInterface-telemetry-config.xml**. The shipping version contains the superset of the telemetry required by the core components; but if components are added, their telemetry formats need to be merged.

Starting in version 1.1.1, for situations where there is no direct GPS distribution (perhaps when the SMC is located in the sub), set **distributeGPS** to 1 and enter a command to set the date under **dateVerb** (sudo date -utc {mm}{dd}{HH}{MM}{yyyy}.{SS})

PPPHost Configuration

PPPHost establishes a Point-To-Point Protocol-based internet connection via an Iridium modem periodically or upon command.

Common XML file keys to edit

- **scpExecutorSettings → callInInterval** contains the number of seconds between automated call-ins (default 3600, or 1 hour). If this value is set to zero (0), connections are not made on a timed interval. However, the CallNow script (or executing "ServerExec PPPHost CALLNOW") causes a connection attempt to be made immediately.
- **scpExecutorSettings → scpVerb** contains the shell script command to execute just after the connection to the Internet is made. It is initially set to `./runscp 0002`, which causes a new sub-directory (`~/roots/200_PPPHost/workspace`) to be created and a "runme" script file to be copied from an LRI file server under the 0002 directory to the new workspace directory. The follow-up command is configured with the **runVerb** key.
- **scpExecutorSettings → runVerb** contains the shell script command to execute after successful execution of the **scpVerb** command. By default, it changes to the workspace directory and runs the "runme" file that was just copied.

Using this scheme, a shore-side server can input commands to be executed on call-in and vary the commands over time.

Configuring the RUDICS Modem

1. With the modem powered on and connected to an available serial port, establish a terminal session to the modem using HyperTerminal or putty (Windows) or GkTerm, minicom or screen (Linux). The default configuration should be able to operate at 19,200 baud, 8 data bits, 1 stop bit and no parity.
2. Reset the modem to factory default configuration by typing `AT&F0` (enter) and verify that the modem answers with `OK` or `0`.
3. Set the baud rate to a fixed 19,200 baud by typing `AT+IPR=6,0` (enter) and verify that the modem answers with `OK` or `0`.
4. Set 'verbose' mode by typing `AT&V1` (enter) and verify that the modem answers with `OK`.
5. Set echo off by typing `AT&E0` (enter) and verify that the modem answers with `OK`.
6. Save the configuration first typing `AT&W0` (enter) and then `AT&W1` (enter), verifying that the modem answer with `OK` after each command.
7. Cycle power to the modem and then type `AT+IPR?` (enter), verifying that it is still configured as set above (6,0).
8. Read the IMEI number from the modem by typing `AT+CGSN` (enter); if desired, record the resulting number for reference and compare with billing or provisioning records.
9. Read the ICC ID (Integrated Circuit Card Identification) or SIM card number by typing `AT+CICCID` (enter); the response should be a 19 or 20 digit number.

Various errors may be returned if the SIM card is not properly installed: a SIM card is required for RUDICS. If desired, record the SIM card number for reference and compare with billing or provisioning records.

Configure the dialup script

The PPP daemon uses a dialer script to interact with the modem and establish the phone call. I used the chat program and the following script (/home/smc/roots/200_PPPhost/chat-iridium) to establish connection:

```
ABORT 'BUSY'
ABORT 'NO CARRIER'
ABORT 'ERROR'
ABORT 'NO DIALTONE'
ABORT 'NO ANSWER'
'' AT
OK AT+CGSN
OK AT+CICCID
OK AT+CSQF
OK AT+CREG?
OK AT+CPIN?
OK AT+CBST=71,0,1
OK ATDT008816999999
CONNECT \d\c
```

Correct the telephone number to match the RUDICS gateway number provided by the Iridium vendor for use with this modem.

Configure the PPP connection script

The PPP connection script (/home/smc/200_PPPhost/runppp) uses the dialer script (above) to contact the Iridium gateway and then implements the Point-To-Point Protocol over the established connection to connect to the Internet. It contains the serial port name to be used when establishing the connection (typically /dev/ttyLX0). Edit this script to change the port name if necessary.

Manual connection to Iridium network

The following script is normally executed by PPPHost during connection, but it can be executed manually to set the system up via a terminal program. Note that the modem must be connected to an antenna with an unobstructed view of the sky.

1. With the modem connected the SMC (assuming port /dev/ttyLX0 and electrical switch 1), turn on power to the modem:

```
smc@omap35x_torpedo:~$ sudo SetPowerSwitch 1 1
SetPowerSwitch: 0x11
```

2. Enable RTS/CTS flow control by typing AT&K3 and verify that the modem answers with OK
3. Configure the modem to hang up on DTR transition from ON to OFF by typing AT&D2 (enter) and verify that the modem answers with OK
4. Prevent the modem from answering incoming calls by AT+SBDMTA=0 (enter) and verify that the modem answers with OK
5. Disable the SBD ring indicator by typing AT+SBDMTA=0 (enter) and verify that the modem answers with OK

6. Stop automated signal strength reports by typing AT+CIER=0,0,0 (enter) and verify that the modem answers with OK
7. Enable automated network registration reports by typing AT+CREG=2 (enter) and verify that the modem answers with OK
8. Get an immediate registration report by typing AT+CREG? and verify that the modem answers with a string formatted like +CREG:002,001,"891","ffff" followed by OK

The status number, which is the second number in the response, '001' in the example above should be interpreted as follows:

- 0 (not registered, not searching for new operator) or 3 (registration denied) indicate failure
- 1 (registered, home network) or 5 (registered, roaming) indicate success
- 2 (not registered, searching for new operator) or 4 (unknown/initial state) are normal waiting values: continue to monitor automated reports. Note that the Iridium network has some gaps that can delay registration for 5-10 minutes.

In case of failure, verify antenna connection or check modem provisioning with Iridium.

9. Disable automated registration reports by typing AT+CREG=0 (enter) and verify that the modem answers with OK
10. Get the status of the SIM card by typing AT+CPIN? (enter) and verify that the modem responds with +CPIN:SIM PIN2 message followed by OK. Other responses indicate that the SIM card is not usable and troubleshooting is required.
11. From the Linux command prompt, run the PPP connection script (./runppp &) and wait for the connection to be made.

```
smc@omap35x_torpedo:~/roots/200_PPPhost$ ./runppp &
[1] 361
smc@omap35x_torpedo:~/roots/200_PPPhost$ abort on (BUSY)
abort on (NO CARRIER)
abort on (ERROR)
abort on (NO DIALTONE)
abort on (NO ANSWER)
send (AT^M)
expect (OK)
^M
OK
-- got it

send (AT+CGSN^M)
expect (OK)
^M
^M
300025019999999^M
^M
OK
-- got it

send (AT+CICCID^M)
expect (OK)
^M
^M
898899999999999999^M
^M
```

```
OK
-- got it

send (AT+CSQF^M)
expect (OK)
^M
^M
+CSQF:5^M
^M
OK
-- got it

send (AT+CREG?^M)
expect (OK)
^M
^M
+CREG:000,001^M
^M
OK
-- got it

send (AT+CPIN?^M)
expect (OK)
^M
^M
+CPIN:SIM PIN2^M
^M
OK
-- got it

send (AT+CBST=71,0,1^M)
expect (OK)
^M
^M
OK
-- got it

send (ATDT008816999999^M)
expect (CONNECT)
^M
^M
CONNECT
-- got it

send (\d)
Serial connection established.
using channel 1
Using interface ppp0
Connect: ppp0 <--> /dev/ttyLX0
.
.
.
... A large number of lines omitted ...
.
.
.
local IP address 192.168.13.51
remote IP address 192.168.13.254
```

12. Test the connection by pinging a well-known internet server; for example, sudo ping www.google.com

```
smc@omap35x_torpedo:~/roots/200_PPPhost$ sudo ping www.google.com
PING www.l.google.com (74.125.153.104) 56(84) bytes of data.
64 bytes from ty-in-f104.1e100.net (74.125.153.104): icmp_seq=1 ttl=47 time=1543 ms
64 bytes from ty-in-f104.1e100.net (74.125.153.104): icmp_seq=2 ttl=47 time=1520 ms
64 bytes from ty-in-f104.1e100.net (74.125.153.104): icmp_seq=3 ttl=47 time=1522 ms
64 bytes from ty-in-f104.1e100.net (74.125.153.104): icmp_seq=4 ttl=47 time=1601 ms
^C
--- www.l.google.com ping statistics ---
```

5 packets transmitted, 4 received, 20% packet loss, time 8888ms
 rtt min/avg/max/mdev = 1520.877/1547.182/1601.591/32.714 ms, pipe 2

- When you are done with the connection, get a process listing (`ps -efH`) and look for the `pppd` (PPP daemon); kill the process:

```
smc@omap35x_torpedo:~/roots/200_PPPhost$ ps -efH
UID      PID  PPID  C  STIME TTY          TIME CMD
root         2      0  0  05:43 ?           00:00:00 [kthreadd]
.
.
... a large number of lines omitted ...
.
smc        317      1  0  05:43 ttyDefault 00:00:00 -sh
smc        361     317  0  05:46 ttyDefault 00:00:00 /bin/sh ./runppp
root       362     361  0  05:46 ttyS0      00:00:00 pppd /dev/ttyLX0 19200 crt
smc        374     317  0  05:47 ttyDefault 00:00:00 ps -efH
smc@omap35x_torpedo:~/roots/200_PPPhost$ sudo kill 362
Terminating on signal 15
smc@omap35x_torpedo:~/roots/200_PPPhost$ Connect time 1.0 minutes.
Sent 772 bytes, received 1024 bytes.
sent [LCP TermReq id=0x5 "User request"]
rcvd [LCP TermAck id=0x5]
Connection terminated.
```

- Turn off power to the modem:

```
smc@omap35x_torpedo:~/roots/200_PPPhost$ sudo SetPowerSwitch 1 0
SetPowerSwitch: 0x00
```

Setting up a key file for scp/ssh (dbclient) connection without password

The SMC provides the option to transfer files using `scp` and execute scripts using an embedded systems flavor of `ssh` called `dbclient`. The following procedure for creating an identity key file can be used to avoid using passwords:

- From the `/home/smc/200_PPPhost` directory, generate a public/private key pair as follows:

```
(RSA key pair) dropbearkey -t rsa -f keyfile | grep ssh-rsa > keyfile.pub
(DSS key pair) dropbearkey -t dss -f keyfile | grep ssh-dss > keyfile.pub
```

By convention, the name of the vehicle is used for *keyfile*. For example, if my vehicle is called "stingray", the resulting private key file is called "stingray" and the resulting public key file is called "stingray.pub"

- Make a connection to the Internet manually (see above)
- Use `scp` to copy the public key file (e.g. "stingray.pub") to the target account's ".ssh" directory; for example:

```
scp stingray.pub stingrayuser@bigserver.com:.ssh
```

The first time that the server is contacted, `scp` prompts with a warning message like "Host 'bigserver.com' is not in the trusted hosts file" and ask for permission to proceed. Verify that you want to proceed, which adds the entry into the `known_hosts` file. Subsequent script executions will not be asked.

Enter the password for the target account when prompted.

- On the target server (e.g. `bigserver.com`) and account (`stingrayuser`), replace `~/.ssh/authorized_keys` with the new public key file; or if multiple vehicles are

calling in, concatenate the files into `~/.ssh/authorized_keys` so that any of them can connect.

5. Verify ssh/dbclient access without a password:

```
dbclient -i stingray stingrayuser@bigserver.com
```

6. Verify scp access without a password:

```
ls > testfile.txt
scp -i stingray testfile.txt stingrayuser@bigserver.com:.
```

timelimit2

The standard **timelimit** utility limits the clock runtime of a process to a given number of seconds; for example:

```
exec timelimit -pq -s 9 -t 600 scp $IDENT -P 10501
vehicle@12.156.66.48:$DIRPART/runme workspace/runme
```

In this case, the “scp” command is limited to 600 seconds, and then is killed if it is not complete yet. Some issues with this were found, though, when the command creates a child task. In this configuration, **timelimit** only kills the top level process: the children continue to run.

To resolve this, **timelimit2** uses some classes already in use in the SMC framework classes to recursively kill the children as well as the parent. The syntax is as follows:

```
timelimit2 -timeout=ttt [--signal=sss] [--debug] ...command-line...
```

The timeout value (*ttt*) is the maximum number of seconds to allow the process to run; the optional signal value (*sss*) is the signal number to use when killing processes, which defaults to 9 (SIGKILL)

VemcoVR2C Configuration

The VemcoVR2C application is a module capable of interfacing to, initializing, and receiving data, over a serial port, from a Vemco VR2C sensor. All communications and data received from the sensor is logged in the application’s log file `~/log/VemcoVR2C.log` and all Tag detections reported by the sensor (read from the serial port as ASCII strings) are also sent as telemetry events **VR2CTagReport**. The VemcoVR2C application is started from:

```
~/roots/200_VemcoVR2C/
```

And is configured by the XML file:

```
~/roots/200_VemcoVR2C/VemcoVR2C-config.xml
```

The VemcoVR2C instance is configured with the serverName **VR2C**, and assigned the boardId 32 (hexadecimal 20) and taskId 21 (hexadecimal 15).

Common XML file keys to edit

- **serialPortSettings** for VemcoVR2C, the default serial port is `/dev/ttyLX4`. By default it is set to communicate with a baudrate of 9600, parity NONE, 1 stopbit, characterSize 8, and flowControl NONE. Ensure that the serial connector cables are plugged into the specified serial port on the SMC.

- **powerOnVerb** and **powerOffVerb** for VemcoVR2C, the command string references power switch. Ensure that the power cable is plugged into the specified power switch on the SMC.
- **VR2CserialNumber** this is used to identify responses from the sensor, and is specified as a six digit number. For example **450006**
- **VR2CcmdPrefix** this string is used to construct the command to be sent to the sensor and includes special characters, the serial number and a checksum, as per a fixed format. For example: ***450006.0#15,**
- **VR2CwakeupRetries** number of times to re-try the wakeup sequence on the sensor if no response is received. Default setting is **1**.
- **VR2CwakeupDelay** time, in milliseconds, after which a response to the wakeup command should be received. Default setting is **150**.
- **VR2CresponseDelay** time, in milliseconds, after which a response to any command (except wakeup) should be received. Default setting is **2000**.
- **TagPrefix** an ASCII string depicting the start of a valid tag detection field in the message received from the sensor. Default setting is **A69-**.
- **ReportChokeMethod** a number specifying what method, if any, to use for regulating the frequency of reporting tag detection samples as telemetry events. A value of 0 indicates no regulation. A value of 1 indicates a minimum gap (delay) between successive events as defined below. All tag detection samples will be reported, but with the minimum delay between successive reports, if the samples arrive more frequently. Default setting is **1**.
- **ReportDelaySeconds** time, in seconds, for the gap/delay between successive telemetry events of tag detection samples if ReportChokeMethod 1 is selected. Default setting is **60**.
- **ReportDelaySeconds** time, in seconds, for the gap/delay between successive telemetry events of tag detection samples if ReportChokeMethod 1 is selected. Default setting is **60**.

Using the VemcoVR2C

The VemcoVR2C application, for this demo implementation, performs a power On of the sensor, initializes it as per a set study sequence for real time monitoring of tag detections, and then monitors all serial data received from the sensor for tag detections which are packaged and sent as telemetry events.

WetLabsSensorSimulator

The utility `~/bin/WetLabsSensorSimulator` is designed to generate simulated Eco Puck samples, provide configuration and menu responses to sensor commands, and emulate the sensor's start and stop states when responding to these commands. The commands are received and simulated responses sent over a pseudo serial port, specified when starting the utility.

In order use the simulated communications, the WetLabsSensor application must be configured for communicating with an Eco Puck sensor (configured in `~/roots/200_EcoSensor`) and using the pseudo serial port specified when invoking the WetLabsSensorSimulator.

Example:

```
WetLabsSensorSimulator /home/smc/roots/200_EcoSensor/pseudo
```

Appendix 1 - Sample Application Configuration File

The following is a sample application configuration file listing taken from the GPS Listener process (GPSListener-config.xml):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE boost_serialization>
<boost_serialization signature="serialization::archive" version="7">
<rootlevel class_id="0" tracking_level="0" version="2">
  <loggerSettings class_id="1" tracking_level="0" version="0">
    <logLevel>LT_WARNING</logLevel>
    <processName>GPSListener</processName>
    <isFileDes>1</isFileDes>
    <fileDes>1</fileDes>
    <fileName></fileName>
    <pipeName></pipeName>
  </loggerSettings>
  <serialPortSettings class_id="2" tracking_level="0" version="0">
    <portName>/dev/ttyS1</portName>
    <baudRate>4800</baudRate>
    <parity>NONE</parity>
    <characterSize>8</characterSize>
    <stopBits>1</stopBits>
    <flowControl>NONE</flowControl>
    <rts>1</rts>
    <dtr>1</dtr>
    <breakTimeoutMultiplier>1</breakTimeoutMultiplier>
  </serialPortSettings>
  <maxString>1000</maxString>
  <readTimeoutMs>50</readTimeoutMs>
  <gpsEventName>GPSSample</gpsEventName>
  <gpsEventType>GPSSample</gpsEventType>
  <gpsEventDomain>LiquidR-SMC</gpsEventDomain>
  <serverName>GPSListener</serverName>
  <serverKind></serverKind>
  <dateSetDirectly>1</dateSetDirectly>
  <dateVerb>sudo date --utc {mm}{dd}{HH}{MM}{yyyy}.{SS}</dateVerb>
  <defaultOutputFileName>/mnt/XDATA/GPS/sensor.out</defaultOutputFileName>
  <eventTransceiverSettings class_id="3" tracking_level="0" version="2">
    <eventServiceName>EventManager</eventServiceName>
    <eventServiceKind></eventServiceKind>
    <receiveEnable>0</receiveEnable>
    <aliasToEvent class_id="4" tracking_level="0" version="0">
      <count>0</count>
      <item_version>0</item_version>
    </aliasToEvent>
    <eventFilterEnabled>0</eventFilterEnabled>
    <eventNameFilter class_id="6" tracking_level="0" version="0">
      <count>0</count>
      <item_version>0</item_version>
    </eventNameFilter>
    <eventTypeFilter class_id="7" tracking_level="0" version="0">
      <count>0</count>
      <item_version>0</item_version>
    </eventTypeFilter>
    <displayReceivedEvents>0</displayReceivedEvents>
    <fullyDescribeReceivedEvents>0</fullyDescribeReceivedEvents>
    <doNotDisplayList>
      <count>0</count>
      <item_version>0</item_version>
    </doNotDisplayList>
  </eventTransceiverSettings>
  <pmClientSettings class_id="9" 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="10" tracking_level="0" version="0">
  <boardID>32</boardID>
  <taskID>241</taskID>
  <description>GPS event source and time updater</description>
  <ior>(This will be overwritten)</ior>
  <readableName>GPSListener</readableName>
</payloadDescription>
</pmClientSettings>
</rootlevel>
</boost_serialization>
```

Appendix 2 - Sample Telemetry Configuration

The following listing is a sample configuration for the telemetry reported by PayloadInterface (PayloadInterface-telemetry-config.xml):

```
<?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" />

```

```

LengthToFollow" />
    <Element libField="Raw" eventField="Telem-
    <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" />
            </fields>
          </sampleHeaderFormat>
        </second>
      </Element>
    </reportParams>
  </second>
</Element>

```

```

        <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>
<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>
    <Element>
        <first boardID="0x20" taskID="0x15" />
        <second>
            <key boardID="0x20" taskID="0x15" />
            <payloadType value="0x0000007F" />
            <eventNameToFormatCode>
                <Element>
                    <first value="VR2CTagReport" />
                    <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="FC0ASCIINormal" />
                        <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="StringEvent-String"
/>
                        </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="FC0ASCIINormal" />
      <value content="CA_LITERAL">
        <literal type="UINT8" value="0x10" />
      </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>

```

History

date	changes made	author
8/10/2011	Initial version	Mike Cookson
8/22/2011	Additions for 1.0.10 release	Mike Cookson
10/13/2011	Additions for 1.1.0 release	Mike Cookson
11/15/2011	Additions for 1.1.1 release	Viet Tran
11/18/2011	Additions for 1.1.2 release	Mike Cookson
12/15/2011	Additions for 1.1.3 release	Raj Jha
01/18/2012	Additions for 1.1.4 release	Viet Tran
02/03/2012	Additions for 1.1.5 release	Viet Tran
02/27/2012	Modification of 1.1.5 release notes	Mike Cookson
04/03/2012	Additions for 1.1.6 release	Mike Cookson
04/16/2012	Additions for 1.1.7 release	Mike Cookson
05/31/2012	Additions for 1.1.8 release	Mike Cookson