KONGSBERG

Seaglider Logdev

Users Guide

Kongsberg Underwater Technology, Inc.
19210 33rd Avenue West, WA 98036-4707 Lynnwood United States of America
Telephone +1 425 712 1107 Telefax +1 425 712 1197 http://www.km.kongsberg.com
Enterprise number 760687050

Revisions

Rev.	Written by		Checked by		Approved by	
	Date	Sign.	Date	Sign.	Date	Sign.
A	2/11/2014	JEF	2/14/2014	WRR		
B	2/28/2014	JEF				
C	5/19/2014	JEF	5/19/2014	WRR		
D	8/4/2014	WRR				
E	1/16/2015	WRR				
F	10/30/2015	WRR				
G	08/11/2016	ZRM				

Document history

Rev. D Added more information regarding the `$xx_UPLOADMAX` parameter, and noted that the value of the "prefix" parameter must be lower-case.

Rev. E Updated after logdev branch merged with trunk and 66.11.R3 released.

Rev. F Remove TBD re %c line substitution. %c outputs 1 or 2, not a or b
Also change all "predive" to "pre-dive"

Rev. G Added log-cmd, log-resp, log<1-3>, _RECORDABOVE negative # feature.

The information contained in this document may be subject to change at a later date (due, for instance, to availability of components). Notice will be given only in case such a change is deemed to be of any consequence for customers. Kongsberg Maritime AS shall not be liable for incidental or consequential damages in connection with the furnishing, performance, or use of this document.

Table of contents

1 OVERVIEW

This document describes Seaglider's *logdev* interface.

1.1 Logger Devices

A logger device is a sensor or instrument with built-in, on-board memory. Once the device is told to start logging, it will collect samples continuously, storing the data internally, until it is told to stop logging. At a later point in time the data stored on the device is off-loaded for subsequent processing.

Logger devices are thus quite different from simple serial sensors such as Wetlabs that the Seaglider supports via the *serdev* interface. Seaglider typically powers on these *serdev* sensors, takes a single sample, and then powers them off according to the sampling schedule defined in the Seaglider *science* file.

1.2 Logdev

The Seaglider *logdev* interface provides software support for logger devices on Seaglider. It allows the user to add a new logging sensor to the Seaglider without changing the Seaglider code base. An ASCII configuration file (.cnf file) allows the specifics of device configuration and access to be described to the Seaglider run-time software without requiring source code changes.

2 CNF FILE

This section introduces the various fields in a CNF file and how they work.

2.1 CNF File Naming

Note that the CNF file must conform to the DOS 8.3 file naming standard. The file name must be in lowercase. By convention, CNF files use the '.cnf' extension. File names that exceed the 8.3 file naming convention will fail to load successfully on the Seaglider.

2.2 Required Fields

The logdev .cnf file has required fields and optional fields. The eight required fields are used to specify how the Seaglider should interact with the device. They also provide information needed to integrate the logger device into the Seaglider's hardware configuration, menu system, battery usage calculations, etc.

name=	Name of the logger device. This name will appear in the Seaglider logger device configuration menus. Can be mixed-case. Example: name=PAR
baud=	Baud rate to use for serial communications with the device. Typically one of the following values: 2400, 4800, 9600, 19200, 38400, 57600. Example: baud=9600
warmup=	Warm up time (milliseconds) between device power on and spot sample. Integer value. Typical values include 10, 50, 200, 500, 2000, etc. If your device comes up and outputs a prompt, consider using powerup-timeout instead. If powerup-timeout is specified, it will override the warmup value. Example: warmup=300
timeout=	Specifies how long to wait for a response from the device when the device is already powered on (milliseconds). Integer value. Note – use powerup-timeout to specify how long to wait for a device to power on and become ready to process commands. Example: timeout=2000

voltage=	Voltage (10 or 24) required by the device. This value is used when calculating battery usage. Example: voltage=10
current=	Current (amps) drawn by the device when powered on. This value is used when calculating battery usage. Example: current=0.5
cmdprefix=	Four-character prefix of the form '$xx_' that *cmdfile* parameters related to the device will have. The Seaglider software recognizes four parameters for each logdev device, and finds them based on the cmdprefix value. The four parameters are $xx_RECORDABOVE, $xx_PROFILE, $xx_XMITPROFILE, and $xx_UPLOADMAX (the $xx_ portion of these parameter is replaced by the cmdprefix value). Example: cmdprefix=$SH_ The $SH_ cmdprefix causes Seaglider to look for $SH_RECORDABOVE, $SH_PROFILE, $SH_XMIT_PROFILE, and $SH_UPLOADMAX. Note: the $SC_ cmdprefix is reserved for the Seaglider's optional science controller CPU.
prefix=	Two-character prefix to use when data files are transferred from the device and stored on Seaglider's compact flash. These characters must be in lower case. Example: prefix=la will result in data files named la0001az.x (The remaining characters of the data file name encode the dive number, whether the data is from a dive or a climb, whether it is compressed, and whether the data should be transferred upon surfacing).
prompt=	The prompt string output by your device. Logdev will wait for this string when processing some commands. For example, the 'download' command will wait for the prompt string while reading and saving data output by the logger sensor. Example: prompt=S>

There are some related, non-required, fields:

datatype=	Type of data produced by the logger sensor. Specify 'z' for gzipped data, or 'u' for uncompressed data. The default is 'u'. Example: datatype=u

powerup-timeout=	When the Seaglider powers on the device, it will wait this long (ms) for the device to power on and output the prompt string (if your device does not output a prompt string use warmup= instead). If powerup-timeout is not specified, Seaglider will check if a warmup= time has been specified, and will wait for that many ms and then output the wakeup string, if any. If neither is specified then Seaglider assumes the device is immediately available for use.

2.3 Command String Substitution

The logdev .cnf file allows command strings to be sent to the logger device. These strings tell the device to do something – take a spot sample, start sampling, stop sampling, etc. Before sending the command string to the device, logdev first expands special character sequences in the command string according to well-defined rules. The following substitutions are supported:

%1 .. %9	Introduce a small delay to give the logger device a chance to process the preceding command in the command string. A %1 causes a 10 millisecond delay, a %2 causes a 20 millisecond delay, etc., up to %9 which causes a 90 millisecond delay. Example: wakeup=%r%9%r%[prompt>]
%b	Break (often used to get device's attention). Example: wakeup=%b%r
%r	<CR> Carriage Return – ASCII 13.
%n	<LF> Line Feed – ASCII 10.
%e	<ESC> Escape – ASCII 27.
%$xx	(Available in version 66.12) Send hex byte xx - must include both characters (00 - ff)
%p	Wait for the device to output a prompt. Ignore device output until the prompt is seen. The prompt to wait for is defined by the .cnf file 'prompt=' field. The amount of time to wait is given by the .cnf file 'timeout=' field.

%[]	Wait for an arbitrary sequence of characters. Characters between the brackets can include any ASCII character. To wait for %, use %%. To wait for], use %]. Use %r and %n to wait for CR and NL, respectively. Use %$xx to wait for an arbitrary hex byte. Several commands use the string gather while waiting for this sequence as a string result to be parsed (metadata and log-cmd for example).
%%	%
%i	Turn the power on
%o	Turn the power off
%g	Insert Seaglider ID (three digits, with leading zeroes if required)
%d	Dive number (four digits, with leading zeroes if required)
%f	Seaglider's local filename string without extension (ex. tm0001az)
%m	Maximum size of file to upload. The $xx_UPLOADMAX cmdfile parameter specifies the value of this substitution string. The units (bytes, lines, samples) of this string must be specified by the logger.
%l (lower-case letter l)	Last GPS position, formatted as a $GPRMC NMEA string.
%s	Sequential start number
%c	Cast number. Expands to 1 (dive) or 2 (climb). This is the sequential start number within a single dive cycle.
%a	Profile indicator letter – 'a' for dive, 'b' for climb. This can be useful when constructing filenames to match glider local names, e.g. if 'prefix=xy' and 'datatype=u', then %f could also be written as xy%d%au.x
%F	Flush pending input from the logger device.
%T	Replace with the last temperature value read (degrees C), formatted as a floating point number with three decimal places.
%D	Replace with last depth value read (meters), formatted as a floating point number with two decimal places.
%V	Replace with last calculated value of speed of sound (meters/sec), formatted as a floating point number with one decimal place.
%W	Replace with last vertical velocity calculated (meters/sec), formatted as a floating point number with four decimal places.
%t	Replace with number of seconds Seaglider is into this dive (seconds), formatted as a floating point number with three decimal places.

%{}	Interpret the characters inside {} as a strftime() string. For more information on the strftime() function please refer to http://man7.org/linux/man-pages/man3/strftime.3.html, or do an internet search for 'strftime'.
%X, %Y, %Z	Executes scripts as explained below in Script Files

2.4 Command Strings

Command strings are basically the commands you would type to your logger sensor if you were connected to it via e.g. minicom or hyperterminal. The logdev interface allows you to specify the command strings that will be sent to your device during various phases of the Seaglider dive profile. The command strings are stored in the logdev .cnf file.

Command string substitution occurs before the command is sent to the device, allowing you to provide the exact character sequences required by your sensor. The logger device's documentation should describe the exact format of the command strings it expects to receive. Using command string substitution you can ensure that the proper carriage control specifiers (%r, %n) are used. Command string substitution characters are described in the preceding section.

Note that a command string containing spaces must be enclosed in quotes. For example, your command string might look like this:

sample="do sample%r%[s>]"

The following table describes the command strings that are supported:

wakeup=	Character sequence to send (when power is on) in order to get a prompt . Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: wakeup="wake up%r"

sample=	The command your logger sensor needs to receive in order to acquire and report a single spot sample. Normally not invoked while your sensor is busy with continuous sampling. Typically only invoked from the Seaglider menu system via the 'hw/loggers/<devName>/sample' menu choice. Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: sample=sample%r%[prompt>] The %[] wait sequence at the end of the command is important. See section on 'Wait Sequence vs. Wait Prompt'.
start=	The command your logger sensor needs to receive in order to start continuous sampling. The Seaglider will monitor the conditions you specified with $xx_RECORDABOVE and $xx_PROFILE and invoke this command when those conditions are first met (replace $xx_ with the 'cmdprefix=' value). Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: start="start %f compress%r%p"
stop=	The command your logger sensor needs to receive in order to stop continuous sampling once it has been started. The Seaglider will monitor the conditions you specified with $xx_RECORDABOVE and $xx_PROFILE and invoke this command when those conditions are no longer met (replace $xx_ with the 'cmdprefix=' value). Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: stop=stop%r%p
status=	If this command is specified, Seaglider will send it to your logging sensor at each sample interval (typically every 5-6 seconds). This can be used to pass status information from Seaglider to the logger sensor (e.g. %t, %T, %D, %V, %W). Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: status=depth:%D%r

xmodem=	Use this command if your logging sensor provides an xmodem interface to download files requested by filename. If this command is specified, Seaglider will send it to your logging sensor after a dive. This command should cause your logger sensor to SEND the named file to Seaglider via xmodem; Seaglider will invoke the corresponding xmodem RECEIVE and will store the file on Seaglider's compact flash. Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: xmodem="xs %f %m%r%p" Note – at this time, the xmodem command does not support inclusion of data received from the .cnf file 'metadata' command.
download=	Use this command if your logging sensor outputs its collected data by writing ASCII output to the serial port. If this command is specified, Seaglider will send it to your logging sensor after a dive. Depending on the value of $xx_PROFILE, this command may be sent 0, 1, or 2 times. The command you specify should cause your logger device to download ASCII data to Seaglider via serial dump. Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: download="cat %f%r" Note – Internally, the Seaglider code for 'download' will capture sensor output to a file until it sees the prompt string defined by the 'prompt=' field in the .cnf file. Therefore, the string you specify for the 'download' command should *NOT* end in a %[] sequence intended to wait for the sensors prompt, nor should it end in a %p.
metadata=	If this command is specified, Seaglider will send it to your logging sensor just before sending the 'download=' command to download data. The command can specify the %[] sequence to capture output which will be included at the head of the resultant data file. Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. A maximum of 4094 bytes of captured metadata output is supported. Example: metadata=%F%r%pUH%F%r%[->] In this example, the UH command is sent to the logger device, and all output that results, up until the -> prompt is received, will be written at the start of the data file. The %[] wait sequence at the end of the command is important. See section on 'Wait Sequence vs. Wait Prompt'.

cleanup=	If this command is specified, Seaglider will send it to your logging sensor just after the glider has been launched. You can use this command to cleanup logger device memory before the next series of dives. Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: cleanup=”del la*.x%r%p”
pre-dive =	If this command is specified, Seaglider will send it to your logging sensor before it starts each dive. You can use this command to reset state prior to a dive, to cleanup any per-dive temporary files, etc. Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: pre-dive=%r%pRESET%r%p
clock-read=	Specify the command needed to read the clock on the logger sensor. This command is only invoked from the hw/loggers/<yourDev>/clock menu option. The output is not interpreted by logdev, but will be displayed on the screen for the user to review. Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: clock-read=”%r%pGETTM%r%[s>]” The %[] wait sequence at the end of the command is important. See section on ‘Wait Sequence vs. Wait Prompt’.
clock-set=	Specify the command needed to set the clock on your logger sensor. Use the %{} substitution sequence to specify the order of the date and time components required by your device. The %{} sequence is interpreted using the strftime() function. Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: clock-set=%F%r%pDATETIME=%{%m%d%Y%H%M%S}%r%p
clock-sync=	Specify when the clock-set command should be sent to your logger sensor. This is a comma-separated list of points at which the clock-set command is sent. Choices are **gps1** (after GPS1 fix), **gps2** (after GPS2 fix), and **start** (before every ‘start’ command). Default is **gps2**.

sleep=	Specifies the command to execute to put logger sensor into a lower-power 'sleep' state. Used by the post-* commands. Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: sleep=%r%pZZZ%r
post-clock=	Power management policy to apply after a clock-sync operation. Must be one of: **on**, **off**, or **sleep**. On means leave logger sensor on. Off means turn logger sensor off. Sleep means send the 'sleep=' command. Default is **off.**
post-transfer=	Power management policy to apply after a file transfer operation. Must be one of: **on**, **off**, or **sleep**. Default is **off.**
post-stop=	Power management policy to apply after stopping sampling. Must be one of: **on**, **off**, or **sleep**. Default is **off.**
pre-off=	Allows a command to be sent to the logger sensor just before the logger sensor power is turned off. Normally not needed for typical marine devices. Useful when logdev is talking to a separate single-board computer with a connected device – in this case, the pre-off command can be used to tell the application and o/s on the single-board computer to exit before the power to the device is removed. Maximum of 255 bytes. If the command string contains spaces the entire string must be enclosed in quotes. Example: pre-off=%r%pEXIT%r%9%9%9%9%9%9%9

2.5 Additional Parameters

The CNF file provides a way for commands string to include values from up to three Seaglider cmdfile parameters.

param-x=	The name of a cmdfile parameter. This parameter's value will be sent when the sequence %x is found in a command and expanded.
param-y=	The name of a cmdfile parameter. This parameter's value will be sent when the sequence %y is found in a command and expanded.
param-z=	The name of a cmdfile parameter. This parameter's value will be sent when the sequence %z is found in a command and expanded.

For example, if param-x=RATE, and cmdprefix=$MD_, then the cmdfile parameter $MD_RATE is being referenced. If the cmdfile contains $MD_RATE,1.2 then when %x is encountered in a command the value 1.2 is sent to the device.

2.6 Script Files

The CNF file provides a way to execute a script of commands from a separate file. This feature can be used to allow complex, multi-command interactions with a device.

script-x=	Name of a script file. If %X is found in a command, then the commands found in the script-x file are interpreted and then sent to the device. The script is expected to exist on the Seaglider compact flash card.
script-y=	Name of a script file. If %Y is found in a command, then the commands found in the script-y file are interpreted and then sent to the device. The script is expected to exist on the Seaglider compact flash card.
script-z=	Name of a script file. If %Z is found in a command, then the commands found in the script-z file are interpreted and then sent to the device. The script is expected to exist on the Seaglider compact flash card.

The lines in the script-x, script-y, and script-z files are fully interpreted using the command string substitution mechanism described above.

2.6.1 Auto-Download of Script Files

The .cnf file supports automatic download of potentially updated copies of the script files above from the base station when the Seaglider is at the surface.

download-script-x=	Set to 1 to download script-x from basestation when at surface. Default is 0,meaning do not download script-x when at surface.
download-script-y=	Set to 1 to download script-x from basestation when at surface. Default is 0,meaning do not download script-y when at surface.
download-script-z=	Set to 1 to download script-x from basestation when at surface. Default is 0,meaning do not download script-z when at surface.

This feature can be used by the pilot to control the logger device during a mission. For example, a device's 'start=' command might become:

```
start=%X
script-x=mystart.bat
download-script-x=1
```

where the contents of mystart.bat are:

```
%r%pvalue1=17.0%r
%pvalue2=0.05%r
%pstart%r
```

Now the pilot can change the values of the parameters 'value1' and 'value2' being used by the device by putting an updated version of the file mystart.bat on the base station. When the Seaglider surfaces and calls in, the updated file will be downloaded to the glider and stored on the compact flash card. The next time logdev tells the logger to start sampling, the new values of those parameters will be used.

2.6.2 Post-Dive Log Values (Available in version 66.12)

Provides the ability to get specific data from a logger added to the log file at the end of a dive e.g. free disk space.

log-cmd=	Command to send to collect post-dive log results. Results will be reported in the log file as a string using the parameter name given by log-0 (below) unless a log-resp value is also given.
log-resp=	Specifies how the results from the log-cmd should be parsed to break out up to 3 floating point pseudo-parameters to be reported in the log file as log-0, log-1, and log-2.
log-0=	Name of pseudo-parameter to report in log file.
log-1=	Name of pseudo-parameter to report in log file.
log-2=	Name of pseudo-parameter to report in log file.

Example:

prefix = XX
log-cmd = freediskspace%r%n
log-resp = %0 kB available. %1 files
log-0 = FREE
log-1 = FILES

will put the following in the log file

$XXFREE,<some value>
$XXFILES,<some value>

If no log-resp is provided the results would be:

$XXFREE,n kB available. m files

3 CMDFILE PARAMETERS

This section describes the Seaglider cmdfile parameters that are used by the logdev interface.

3.1 Per-Logger Parameters

All logdev devices recognize their own versions of these four parameters:

- $xx_RECORDABOVE – when Seaglider is above this depth, the logger is run. When Seaglider drops below this depth, the logger is stopped. A negative value indicates that the logger should be run BELOW this depth (i.e., -200 means run logger below 200 m, 200 means run logger above 200 m). A value of 0 means never record.
- $xx_PROFILE – a value between 0 and 3 (inclusive) that specifies when the logger should run. 0=none, 1= dive, 2= climb, 3=both dive and climb.
- $xx_XMITPROFILE – indicates which data should be transmitted from the Seaglider to the base station via Iridium. 0=none, 1=dive, 2=climb, 3=both dive and climb.
- %xx_RECORDAPOGEE - indicates whether or not to continue recording dive data until the end of a loiter period specified by T_LOITER. 0 = stop recording when the glider goes neutral at apogee. 1 = continue recording after the glider goes neutral at apogee state until the end of T_LOITER.
- $xx_UPLOADMAX – lets you specify a size limit for uploaded files. Applied when the glider requests a file from the device via the .cnf file xmodem= or download= commands. The parameter value is sent to the logger via the '%m' command string substitution operator. The Seaglider does not process this value itself, it is strictly for use by your logger. Limiting the file size this way can be used to reduce the size of data files transferred from the sensor to Seaglider compact flash, and from Seaglider compact flash to shore over the Iridium link. The procedure for using this parameter is:
  1. write a data transfer program for your logger that has a command-line option for maximum file size.
  2. add a 'xmodem=' or 'download=' command string to the logger's .cnf file that invokes the logger's data transfer program and include the '%m' substitution operator.
  3. add the $xx_UPLOADMAX parameter to the cmdfile in the glider's home directory on the basestation and specify the desired maximum file size.

The ‘$xx_’ prefix is defined by the *cmdprefix* field in the .cnf file.

For example, if the .cnf file for a given logger device specifies ‘cmdprefix=$MD_’, then logdev will use the values of the cmdfile parameters $MD_RECORDABOVE, $MD_PROFILE, $MD_XMITPROFILE, $MD_RECORDAPOGEE, and $MD_UPLOADMAX to control data collection and data upload for that device.

The Seaglider stores copies of these parameters in a file called ‘loggers.cmd’ on the Seaglider compact flash when a cmdfile containing the parameters is received from the base station.

You can change the values of these parameters from the Seaglider menu system during indoor testing when the Seaglider is unable to connect to the base station. At any menu system prompt, enter a textual parameter definition, e.g. ‘$PH_PROFILE,0’. Be aware that once Seaglider can connect to the basestation settings made at the command line are subject to being overridden by values in the cmdfile.

3.2 $LOGGERS

Seaglider supports up to four logger devices. The $LOGGERS parameter in the cmdfile is used to indicate which logger devices will be activated during self tests or dives. This parameter must be set properly in order to enable your logger. The value specified for this parameter is an integer value between 0 and 15. The value is treated as a bit mask; bits that are set indicate logger devices that should be enabled. The bits correspond to logger device slot numbers; these are displayed by the hw/config/show command). The following table shows the effect of the various values of the $LOGGERS parameter. A ‘y’ in this table indicates a slot that is enabled, while a blank indicates a slot that is not enabled. Thus, a $LOGGERS value of 0 (the default) indicates that no logger devices are in use, and a $LOGGERS value of 3 indicates that the logger devices in slots 1 and 2 are enabled.

$LOGGERS value	Logger device in Slot 4 enabled	Logger device in Slot 3 enabled	Logger device in Slot 2 enabled	Logger device in Slot 1 enabled
0				
1				y
2			y	
3			y	y
4		y		
5		y		y
6		y	y	
7		y	y	y
8	y			
9	y			y

10	y		y	
11	y		y	y
12	y	y		
13	y	y		y
14	y	y	y	
15	y	y	y	y

3.3 $CAPUPLOAD

The $CAPUPLOAD parameter controls whether or not the capture file is uploaded from the Seaglider to the basestation. During testing of your logger device you probably want to set this parameter to 1 so that the capture file does get uploaded. When testing is complete and the glider deployed on a mission, this parameter is typically set to 0 in order to reduce the glider's time on the surface when uploading data over a satellite link.

4 GETTING STARTED WITH A NEW DEVICE

Suppose your team has a new logging sensor that needs to be integrated onto your Seaglider. Where do you begin?

4.1 Connection

The first step is to get the device connected to Seaglider so you can experiment with it. Once connected, you'll be able to send commands to the device via a serial connection and see the device respond to your commands.

Start by locating the user documentation for the device (printed manual, online manual, etc.). Make sure the device will operate on either 10V or 24V power. Make sure the device is a typical serial device and not a frequency mode device (these are rare).

Then work with your team to connect the logger to your Seaglider. Someone on the team should know which end-cap ports are routed to which aft terminal board connections (if not, obtaining this knowledge may require partial disassembly of your vehicle). The following table might prove helpful (fill in the end-cap port and sensor names yourself):

End-cap port	Aft term board label	Port number	Nominally	Voltage	Freq. Mode?	Sensor Name
	TPU 6/7	3	"Optics 1"	10		
	TPU 8/9	4	"O2"	10	Yes	
	MUX0	5	"Optics 2"	10		
	MUX1	6	"aux 1"	10 or 24		
	MUX2	7	"aux 2"	10 or 24		

Connect the new logging sensor to the end-cap port using an IE55 cable. Cables are typically provided by the sensor manufacturer. If not, you will have to procure one.

Remember both the port number and 'nominally' name; you will need these when configuring a device that uses your .cnf file.

4.2 Write Simple CNF File

Once the device is connected you need to tell Seaglider a little bit about the device so that Seaglider will know how to access it. You will need to write a very simple CNF file to do this.

Use a unix or linux system to create the CNF file so that it will have the right line-ending characters for the Seaglider. If you must use Windows for this then be sure to use an editor that understands linux line-endings, such as WordPad, TextPad, or gedit.

Your simple CNF file needs to have just eight lines – one for each of the required fields in a logdev cnf file. Here is an example:

```
name=MYDEV
prefix=md
cmdprefix=$MD_
baud=9600
voltage=10
current=.018
warmup=0
timeout=400
```

The name field is the name of your new device. This name will appear in the Seaglider menu system when you are working with the device. Replace MYDEV with a name that makes sense for your new logger.

The prefix field is a two character prefix that will be used when data from the device is downloaded into the Seaglider's compact flash. Replace 'md' with two characters related to your new logger.

The cmdprefix is a four-character string in the form $xx_. Each logger device is controlled by four cmdfile parameters, and this string is the prefix for those parameters. Replace the MD here with capital letters used in your prefix field.

The baud field gives the baud rate of the logger device. If the baud rate of your new logger is something other than 9600 then change this field as needed.

The voltage field is the voltage required by the logger device. This should be either 10 or 24.

The current field would come from the device documentation. This value is used for estimating the battery power consumed by the device. Change this value to match your device documentation.

The timeout and fields also come from the device documentation. Timeout is the number of milliseconds to wait for a response from the device. Warmup is the number of milliseconds to wait between power-up and the first spot sample. Warmup can be 0, but timeout should be a non-zero value.

Save the file as 'mydev.cnf' (again, replace 'mydev' with something resembling the name field in your .cnf) and exit the text editor.

4.3 Load CNF and Configure Device

Now you need to transfer your file to the Seaglider, strip it, load it into the logger library, and configure a device that uses the CNF file. Appendices A and B cover the required steps in detail. Once these steps have been completed you should be able to access the logger using the Seaglider menu system as described in the next section.

5 LOGGER DEVICE TEST MENU

5.1 Accessing the Logger Device Test Menu

With your sensor configured you can now try to use the Seaglider menu system to access the new device.

Start by returning to the Seaglider main menu:

```
------ Main Menu ------
   1 [param   ] Parameters and configuration
   2 [hw      ] Hardware tests and monitoring
   3 [modes   ] Test operation modes and files
   4 [pdos    ] PicoDOS commands (and exit)
   5 [launch  ] Pre-launch
Enter selection (1-5,CR): 2
```

Make choice 2 (hw). You should now see the Hardware menu:

```
------ Hardware Menu ------
*Motors and VBD
   1 [pitch   ] Pitch control
   2 [roll    ] Roll control
   3 [vbd     ] VBD control
   4 [pressure] Pressure sensor
   5 [compass ] Compass (tcm2)
   6 [gps     ] GPS
   7 [modem   ] Modem (xmodem mode)
   8 [intpress] Internal pressure
   9 [altim   ] Altimeter
  10 [sensors ] Sensors
  11 [loggers ] Loggers
*Other
  12 [batt    ] Batteries and fuel gauges
  13 [lowlevel] Low-level hardware (IO,A-D,CF)
  14 [misc    ] Miscellaneous (travel, timeouts, date/time)
  15 [develop ] Developer tests
  CR)  Return to previous
Enter selection (1-15,CR): 11
```

Make choice 11 (loggers) to do hardware testing on loggers. If you have configured a logger device, it should be displayed on the Loggers menu:

```
------ Loggers menu ------
   1 [nw      ] NEW
  CR)  Return to previous
Enter selection (1-1,CR): 1
```

Enter the number corresponding to the logger device you configured. In this example the device is named 'NEW' and the prefix is 'nw', so choice 1 is entered.

You should then see the Logger device test menu:

```
------ Logger device test menu ------
  1 [on      ] Turn on
  2 [off     ] Turn off
  3 [selftest] Selftest
  4 [sample  ] Report a sample
  5 [syncclk ] Synchronize device clock to TT8
  6 [clock   ] Read device clock
  7 [file    ] Get file from device
  8 [action  ] Execute logger action
  9 [config  ] Show configuration
 10 [edit    ] Edit configuration
 11 [direct  ] Direct comms
 12 [capture ] Capture direct output
 13 [command ] Send command
 CR)  Return to previous
Enter selection (1-13,CR):
```

This is the menu you will use most often while developing and testing your CNF file.

5.2 Quick Device Test

When you are first trying to integrate a new sensor, powering it on and trying to communicate with it directly is the typical first task.

To power your sensor on, use menu choice 1:

```
Enter selection (1-14,CR): 1
265.846,HNEW,N,turning NEW on
```

Now you can try using *direct comms* to see if you can communicate with the device. At the menu prompt make choice 11. Then hit the <CR> key a few times and see if you get your device's prompt:

```
Enter selection (1-14,CR): 11
272.585,HNEW,N,RX = 10, TX = 11
272.677,SUSR,N,Starting breakable loop - Ctrl-Q to end

S>
S>
```

In this example, the 'S>' represents the prompt output by the device (so, we were successful).

In direct mode you should be able to see output from your device, and you should be able to type in commands to the device. Note that depending on the device, you may need to turn on local echo in your terminal emulator in order to see the commands that you type.

To escape from direct comms and return to the menu system, use <CTRL>Q. This information is output when you first enter direct comms, but after entering commands and viewing output the message can scroll off the visible part of the display.

Remember to turn on the power before trying to use direct comms to talk to your sensor. Also, remember to turn off the power before exiting the logger device test menu.

5.3 On/Off

These commands turn the power to the logger device on or off.

5.4 Selftest Command

This command is NOT a quick test for determining if the device is powered on and responding to commands. Use the 'sample' command for that.

This command exercises all of the various command strings in the CNF as if the Seaglider were doing a Sea Launch or Test Launch. It is thus best used once the CNF file command strings are near completion.

This command will power on the device and set the time. It will send the 'start sampling' command and pretend to do the 'descent' phase of a dive for 12 intervals. Next it will stop logging as if the Seaglider reached apogee. As the 'ascent' phase starts logdev will start sampling again and continue to sample for 12 intervals before stopping logging at the simulated surface. Files will be transferred from the logger device to the Seaglider file system via the xmodem= or download= commands.

5.5 Sample Command

The sample command (menu choice 4) is a good way to test that your sensor is alive and well and responding to commands. This option causes logdev to send the sample= command string from the .CNF file to the logger device, and to output the response.

Before sending the sample= command, logdev will check to see if the device is powered on. If not, logdev will turn on the power. If powerup_timeout= was specified then logdev will wait that long for the device to output the prompt string (prompt=). If powerup_timeout= was not specified, then logdev will wait warmup_time= msecs and then will output the wakeup= string, if any.

After outputting any response from the device logdev will power off the device if it was not on when the command was issued.

5.6 Syncclk and Clock Commands

The syncclk command (menu choice 5) and clock command (menu choice 6) can be used to test the .cnf file command strings used to set the date and time on your logger device, and to read the current time from your device. Like the sample command, these commands will turn on the device power if it is not on when the command is invoked.

5.7 File Command

The file command (menu choice X) can be used to exercise the commands that download data from the logger device to the Seaglider's compact flash card.

The command takes arguments and therefore cannot be invoked by entering the number at the 'Enter selection' prompt.

Instead, enter the command as follows:

Enter selection (1-14,CR): **file name=pa0004au**

The value for the 'name=' argument is the name of the file to download from the logger.

If the .cnf file contains a download= command, then the Seaglider will attempt to download ascii data from the sensor and store it in the specified file. It will first send the metadata= command to the logger, if one was specified (logdev will insert the output from the metadata= command at the start of the downloaded data file). After that it will receive data from the logger and store the data in the output file on Seaglider compact flash. When the prompt string is detected logdev will stop expecting data from the logger.

If the .cnf file contains an xmodem= command instead of the download= command, then the command will be sent to the logger and logdev will wait for data to arrive from the logger over the serial port in the xmodem protocol. Logdev will store the data that arrives in the output file on Seaglider compact flash.

The metadata= command is not supported for xmodem= downloads – if specified, it will be ignored.

5.8 Action – Test Start= and Stop= Command Strings

Use the 'action' option on the 'logger device test menu' to test your CNF file command strings for starting and stopping logging. Make sure the logger device has been turned on before sending these commands.

After entering the number for the action command menu choice, you will be prompted for the action to perform. Enter '2' to start logging, or '3' to stop logging. No other actions are supported at this time.

Here is an example that shows how to test the CNF file command that starts logging:

```
Enter selection (1-14,CR): 8
Action? [2] 2
977.730,HPAM,N,sample start 2
979.087,HPAM,N,sending [05,14,2014,13,14,05]
979.150,HPAM,N,sleeping 114183 ticks, sync will be 0x0d
982.162,HPAM,N,sampling started pa0001au.a
982.245,SUSR,N,Logger action 2 returned 0
```

To test the CNF file command that stops logging:

```
Enter selection (1-14,CR): 8
Action? [2] 3
992.082,HPAM,N,sampling stopped
992.151,SUSR,N,Logger action 3 returned 0
```

5.9 Show/Edit Configuration

The ‘config’ command will list out the contents of the currently-loaded .cnf file.

The ‘edit’ command is not implemented.

5.10 Direct Communications

Think of this as ‘minicom’, ‘teraterm’, or ‘hyperterminal’ mode. It enabled two-way communication between the Seaglider menu system and the logger device. The device’s prompt will be displayed on the screen, commands to the device can be entered, and output from the device will be displayed.

This menu option can be used to alter the device settings, test commands, etc.

5.11 Capture Mode

Not used.

5.12 Command Mode

Another useful option on the ‘logger device test menu’ is the ‘command’ option. Using this option you can quickly test a command sequence before including it in the .CNF file (make sure the logger device has been turned on before sending it a command).

Enter this command by typing it at the ‘Enter selection’ prompt when the logger device test menu is displayed. For example:

```
------ Logger device test menu ------
     1 [on      ] Turn on
     2 [off     ] Turn off
     3 [selftest] Selftest
     4 [sample  ] Report a sample
     5 [syncclk ] Synchronize device clock to TT8
     6 [clock   ] Read device clock
     7 [file    ] Get file from device
     8 [action  ] Execute logger action
     9 [config  ] Show configuration
    10 [edit    ] Edit configuration
    11 [direct  ] Direct comms
    12 [capture ] Capture direct output
    13 [command ] Send command
    CR)  Return to previous
Enter selection (1-13,CR): command string="SMP%r%[pam> ]"
```

In this example the ‘command’ menu option is invoked by name. The command sequence to be executed is specified via the required argument named ‘string’. If the command sequence to be sent to the logger contains white space then the command sequence must be enclosed in double quotes.

If your .cnf file contains a ‘wakeup=’ sequence, logdev will send it to your device before sending the specified command.

Notice that the command sequence in this example contains a ‘wait sequence’ (%[…]). If you want logdev to wait for your logger to output a response, then you must include a wait sequence in your command string. Without the wait sequence, any output from the logger will be ignored (not displayed). For more information refer to the section on Wait Sequence vs. Wait Prompt.

6 EXTENDING THE SIMPLE CNF FILE

Now that you are familiar with the logger device test menu, you can use it to test out commands to help extend the functionality of your CNF file.

Note that each time you update the CNF and want to test your changes, you will need to reload the CNF file. To reload the CNF file follow these steps:

1. remove the device (Appendix C)
2. unload the CNF file (Appendix D)
3. load the updated CNF file (Appendix A)
4. reconfigure the device (Appendix B)

Appendix E provides a quick summary of all the steps involved. You'll be able to do this without referring to the instructions after the first handful of CNF file edits.

6.1 Prompt, Datatype

Enter the prompt string that must be output by the device before a serial command can be entered. This is probably the same string that the device output when you were using 'direct' mode to talk to the device. To add the prompt string to the CNF file you would enter a line of this form:

prompt=S>

(obviously replacing 'S>' with the string output by the device you are using).

Also enter the datatype field which indicates whether the datafiles produced by the device are compressed ('z') or uncompressed ('u').

datatype=u

6.2 Wakeup Command

When you turned the device power on and entered direct mode, what (exactly) did you have to do to make that prompt appear? The 'wakeup' string should include that information. Indicate that you typed a <CR> by adding a '%r' to the wakeup command string. If you had to wait a bit add %9 to the command string. A combination of waiting and hitting <CR> that works for the Sea-Bird GPCTD device looks like this:

wakeup=%9%9%r%9%9%9%9%9%F%r%p

(wait, wait, send <CR>, wait, wait, wait, wait, wait, flush input from device, send <CR>, then wait for the device to output it's prompt string)

6.3 Sample Command

The sample command is invoked from the hw/loggers/<device>/sample menu command. When this menu choice is selected, the device is powered on and the wakeup command, followed by the sample command, are sent to the device.

Once 'wakeup' and 'sample' are in your CNF file you can enable CNF debugging and observe the bytes being sent in both directions (see chapter on Debugging the CNF File).

The GPCTD device's sample command follows:

```
sample=%r%pOutputFormat=1%r%pTS%r%[S>]
```

Translation: to get a sample, 'send <CR>, wait for the prompt, send the string 'OutputFormat=1<CR> ', wait for the prompt, send the string 'TS<CR>', and then use %[S>] to wait for the prompt.' The TS command causes the device to output one sample. The sample is output in decimal format due to the OutputFormat=1 command. Using %[S>] to wait for the prompt causes the data output by the device to be returned for processing instead of being ignored (as it would be if %p was used to wait for the prompt). Note that both TS and OutputFormat are commands that are specific to the GPCTD sensor, as is the 'S>' prompt.

The 'Wait Sequence vs. Wait Prompt' chapter provides a discussion about when to use %p to wait for the prompt, and when to use %[].

6.4 Start and Stop Sampling Commands

These commands are sent from Seaglider to your logger device to tell your device to start or stop sampling. Send these commands from Direct Mode so you can verify that the commands are sent properly, received by the logger, and responded to as expected. After logging for a while, stop logging and verify that the device collected data. Reproduce the commands and responses in your CNF file.

The GPCTD device's start and stop commands follow:

```
start=%r%pStart%F%r%p
stop=%r%pStop%F%r%p
```

Translation – to start sampling, send <CR>, wait for the prompt, send 'Start', flush the input buffer, send <CR> , then wait for the prompt. To stop sampling, do the same thing except send Stop.

6.5 Metadata Command

The metadata command is sent to your sensor during the downloading of sensor data. The intent of this command is to collect data from the sensor that will be included at the start of the resultant data file. The metadata might describe the rest of the data included in the file.

An example metadata command follows:

```
metadata=%F%r%pUH%F%r%[S>]
```

This command requests that the serial port input buffer be flushed. It then sends a <CR> to the device. Next it waits for the device's prompt sequence. Once it gets the prompt, it sends 'UH', a command which causes this particular sensor to output all its header records (the header records include cast numbers, date, time, number of samples, etc.). The final %[S>] causes the output to be captured until the prompt sequence 'S>' is seen. The metadata is captured by logdev and added to the start of the data file that is being generated by the data download mechanism.

TIP: be careful when ending your metadata command with %p. The %p will cause logdev to **drop** any bytes output by the device while waiting for the prompt sequence. If you are sending the metadata command you are probably trying to capture and **keep** the bytes output by the device. In most cases, ending your metadata command with a %[] wait sequence explicitly containing the prompt sequence is a better choice. In the above example the device's prompt is 'S>'. By using the %[S>] wait sequence here, logdev is able to capture and keep the output data instead of dropping it.

This is perhaps more fully explained in the chapter on 'Wait Sequence vs. Wait Prompt'.

The CNF debug mechanism can be very useful in finding issues with your metadata command. When logdev's debug output shows the bytes received from your device it will append the string "(drop)" to the byte if it will not make it into your data file. See the chapter on 'Debugging Your CNF Command Strings'.

6.6 Download Command

This command should cause your sensor to start downloading ASCII data to Seaglider over the serial connection. Seaglider will keep reading bytes from your sensor until it sees the prompt string specified by the 'prompt=' string in the .cnf file.

Here is an example of the download string:

download=%F%r%pOutputFormat=0%F%r%pUC%c%r

The command means ‘flush input buffer, send <CR>, wait for prompt, send ‘OutputFormat=0’, flush input buffer, send <CR>, wait for prompt, send ‘UC’, send either 1 (for dive) or 2 (for climb), send <CR>’. After this, Seaglider will read output from the sensor and write it to the output file until either a.) the device outputs it’s prompt, or b.) the read times out.

6.6.1 Download - Important Note

Internally, the code for download will capture sensor output to a file until it sees the prompt string defined in the .cnf file. Your ‘download’ command should *NOT* end in a %[] sequence designed to wait for the sensor to output the prompt string. Similarly, your ‘download’ command should *NOT* end in a %p.

6.7 Cleanup Command

Your CNF file can include a *cleanup* command to cleanup temporary files from the logger device that might have accumulated the last time the glider was used. If a cleanup command is supplied it will be invoked when the glider is first launched.

cleanup=%r%pResetLogging%F%r%[confirm%r%n]ResetLogging%F%r%p

Be particularly conservative and careful with the cleanup= command. We all know that it is difficult, time-consuming, and expensive to collect data in the marine environment. By trying to keep your device ‘clean’ you also run the risk of losing data you meant to keep. Use this command with care as data, once deleted, cannot be recovered.

7 DEBUGGING THE CNF FILE

Sometimes it is hard to figure out why your command string isn't working the way you expected. Maybe your command string isn't causing the proper sequence of bytes to be sent to the device. Maybe the device is returning something not accounted for in your %[] wait sequence. Maybe you can talk to the device just fine with Hyperterminal or minicom, but it ignores the Seaglider.

CNF debug support is turned on and off via the 'debug' menu option on the logger device test menu. When the debug option is turned on, the Seaglider executable will generate additional log message output that shows the bytes that it sends to the logger device, as well as the bytes received from the device. Non-printing characters are output in decimal.

The debug feature is for use in debugging Seaglider's interactions with your device. It is for bench-top testing and test launches only. *This is enforced by a runtime check that prohibits the use of this image for sea launches.*

Note: CNF debug logging should only be used during bench testing and deck dives. It is not available for sea-testing because the additional load of the many extra log messages could adversely affect Seaglider's flight control software.

The additional output enabled by this feature appears on the Seaglider menu system console like other log messages, and thus also ends up in the Seaglider capture file.

7.1 Turn Debug Mode On

In the following, the GPCTD device is selected from the hw/loggers menu:

```
------ Loggers menu ------
     1 [pc      ] GPCTD
    CR)  Return to previous
Enter selection (1-1,CR): 1
```

The 'debug' menu option is at the bottom of the logger device test menu:

```
------ Logger device test menu ------
     1 [on      ] Turn on
     2 [off     ] Turn off
     3 [selftest] Selftest
     4 [sample  ] Report a sample
     5 [syncclk ] Synchronize device clock to TT8
     6 [clock   ] Read device clock
     7 [file    ] Get file from device
     8 [action  ] Execute logger action
```

```
 9 [config  ] Show configuration
10 [edit    ] Edit configuration
11 [direct  ] Direct comms
12 [capture ] Capture direct output
13 [command ] Send command
14 [debug   ] CNF Debug
CR)  Return to previous
Enter selection (1-14,CR):
```

To turn the debug support on, enter ‘debug state=1’ at the logger device test menu prompt:

```
Enter selection (1-14,CR): debug state=1
```

7.2 Example Debug Output

The CNF debug output for the GPCTD device’s ‘wakeup’ command string is as follows:

```
1419.662,HGPCTD,N,open port
1419.701,HGPCTD,N,powering on
1419.741,HGPCTD,N,turning GPCTD on
1419.788,HGPCTD,N,wait warmup time
1420.236,HGPCTD,N,sending wakeup command
1420.291,HGPCTD,N,delay 90 ms
1420.424,HGPCTD,N,delay 90 ms
1420.556,HGPCTD,N,send '<CR>'
1420.596,HGPCTD,N,delay 90 ms
1420.729,HGPCTD,N,delay 90 ms
1420.862,HGPCTD,N,delay 90 ms
1420.995,HGPCTD,N,delay 90 ms
1421.128,HGPCTD,N,delay 90 ms
1421.260,HGPCTD,N,flush port input buffer
1421.315,HGPCTD,N,send '<CR>'
1421.355,HGPCTD,N,wait 5 secs for sequence [S>]
1421.418,HGPCTD,N,recv '<CR>'
1421.458,HGPCTD,N,recv '<LF>'
1421.498,HGPCTD,N,recv 'S'
1421.535,HGPCTD,N,recv '>'
1421.572,HGPCTD,N,got expected sequence [S>]
1421.630,HGPCTD,N,back from sending wakeup command
```

The .cnf debug output shows you each character sent in both directions over the serial link. The above output shows that Seaglider receives four bytes from the GPCTD: <CR>, <LF>, ‘S’, and ‘>’. In general, the display of the bytes received from the device can be extremely useful in helping you to provide valid wait sequences (%[]) within your command strings.

7.3 Turn Debug Mode Off

To turn the debug support off, enter ‘debug state=0’ at the logger device test menu prompt:

```
Enter selection (1-14,CR): debug state=0
```

8 WAIT SEQUENCE (%[]) VS. WAIT PROMPT (%P)

You may have noticed that the .cnf syntax gives you two ways to wait for the device to output a prompt. You can use %p, or you can use a %[] sequence whose contents specifies the prompt (for GPCTD, %[S>]). How are these two different?

Basically, you can use %p to wait for the prompt whenever you are NOT interested in the characters returned by the device while waiting. You must use a %[] wait sequence if the data output by the device is important to further processing.

The distinction is small, and perhaps best shown by example.

In the following example, the 'sample=' command string for the GPCTD device will be used. The 'TS' near the end of the string asks the device to output a sample as described in the preceding section. First the results of (incorrectly) trying to use %p to wait for the the sample output will be shown. Then the results of using %[] to correctly capture the output will be provided.

8.1 Using %p

What happens if we replace the %[S>] at the end of the GPCTD 'sample=' command with a %p?

The resultant command string looks like this:

```
sample=%r%pOutputFormat=1%r%pTS%r%p
```

After reloading the .cnf file and reconfiguring the device, when we try to get a sample (using hw/loggers/<device>/sample), we don't get what we expect:

```
Enter selection (1-14,CR): 4
930.029,HGPCTD,N,=r
```

We were supposed to get four floating point values, but we only got "=r".

To see what went wrong, type "debug state=1" at the /hw/loggers/<device> menu prompt, and then choose the sample option again. An excerpt from the resultant output showing the transmission of the sample command string follows:

```
941.892,HGPCTD,N,send '<CR>'
941.931,HGPCTD,N,wait for prompt
941.976,HGPCTD,N,recv '<CR>' (drop)
942.023,HGPCTD,N,recv '<LF>' (drop)
942.071,HGPCTD,N,recv 'S'    (match)
```

```
942.119,HGPCTD,N,recv '>'    (match)
942.167,HGPCTD,N,(got the prompt)
942.212,HGPCTD,N,send 'O'
942.248,HGPCTD,N,send 'u'
942.283,HGPCTD,N,send 't'
942.318,HGPCTD,N,send 'p'
942.354,HGPCTD,N,send 'u'
942.389,HGPCTD,N,send 't'
942.425,HGPCTD,N,send 'F'
942.460,HGPCTD,N,send 'o'
942.496,HGPCTD,N,send 'r'
942.531,HGPCTD,N,send 'm'
942.567,HGPCTD,N,send 'a'
942.602,HGPCTD,N,send 't'
942.638,HGPCTD,N,send '='
942.673,HGPCTD,N,send '1'
942.709,HGPCTD,N,send '<CR>'
942.748,HGPCTD,N,wait for prompt
942.793,HGPCTD,N,recv 'O'    (drop)
942.840,HGPCTD,N,recv 'u'    (drop)
942.888,HGPCTD,N,recv 't'    (drop)
942.935,HGPCTD,N,recv 'p'    (drop)
942.983,HGPCTD,N,recv 'u'    (drop)
943.030,HGPCTD,N,recv 't'    (drop)
943.078,HGPCTD,N,recv 'F'    (drop)
943.125,HGPCTD,N,recv 'o'    (drop)
943.172,HGPCTD,N,recv 'r'    (drop)
943.220,HGPCTD,N,recv 'm'    (drop)
943.267,HGPCTD,N,recv 'a'    (drop)
943.315,HGPCTD,N,recv 't'    (drop)
943.362,HGPCTD,N,recv '='    (drop)
943.410,HGPCTD,N,recv '1'    (drop)
943.457,HGPCTD,N,recv '<CR>' (drop)
943.504,HGPCTD,N,recv '<LF>' (drop)
943.552,HGPCTD,N,recv 'S'    (match)
943.600,HGPCTD,N,recv '>'    (match)
943.649,HGPCTD,N,(got the prompt)
943.885,HGPCTD,N,send 'T'
943.921,HGPCTD,N,send 'S'
943.956,HGPCTD,N,send '<CR>'
943.995,HGPCTD,N,wait for prompt
944.041,HGPCTD,N,recv 'T'    (drop)
944.088,HGPCTD,N,recv 'S'    (match)
944.137,HGPCTD,N,recv '<CR>' (drop)
944.184,HGPCTD,N,recv '<LF>' (drop)
946.488,HGPCTD,N,recv ' '    (drop)
946.536,HGPCTD,N,recv ' '    (drop)
946.583,HGPCTD,N,recv ' '    (drop)
946.631,HGPCTD,N,recv '-'    (drop)
946.678,HGPCTD,N,recv '0'    (drop)
```

```
946.726,HGPCTD,N,recv '.'    (drop)
946.773,HGPCTD,N,recv '1'    (drop)
946.820,HGPCTD,N,recv '7'    (drop)
946.868,HGPCTD,N,recv ','    (drop)
946.915,HGPCTD,N,recv ' '    (drop)
946.963,HGPCTD,N,recv '2'    (drop)
947.010,HGPCTD,N,recv '3'    (drop)
947.057,HGPCTD,N,recv '.'    (drop)
947.105,HGPCTD,N,recv '0'    (drop)
947.152,HGPCTD,N,recv '1'    (drop)
947.200,HGPCTD,N,recv '7'    (drop)
947.247,HGPCTD,N,recv '5'    (drop)
947.295,HGPCTD,N,recv ','    (drop)
947.342,HGPCTD,N,recv ' '    (drop)
947.389,HGPCTD,N,recv '0'    (drop)
947.437,HGPCTD,N,recv '.'    (drop)
947.484,HGPCTD,N,recv '0'    (drop)
947.532,HGPCTD,N,recv '0'    (drop)
947.579,HGPCTD,N,recv '0'    (drop)
947.627,HGPCTD,N,recv '0'    (drop)
947.674,HGPCTD,N,recv '6'    (drop)
947.721,HGPCTD,N,recv ','    (drop)
947.769,HGPCTD,N,recv ' '    (drop)
947.816,HGPCTD,N,recv ' '    (drop)
947.864,HGPCTD,N,recv ' '    (drop)
947.911,HGPCTD,N,recv ' '    (drop)
947.959,HGPCTD,N,recv '0'    (drop)
948.006,HGPCTD,N,recv '.'    (drop)
948.053,HGPCTD,N,recv '0'    (drop)
948.101,HGPCTD,N,recv '0'    (drop)
948.148,HGPCTD,N,recv '<CR>' (drop)
948.195,HGPCTD,N,recv '<LF>' (drop)
948.243,HGPCTD,N,recv 'S'    (match)
948.291,HGPCTD,N,recv '>'    (match)
948.340,HGPCTD,N,(got the prompt)
```

After the ‘TS’ command was issued, the %p sequence caused Seaglider to wait for the prompt. All the numeric data received in response to the ‘TS’ was dropped (ignored) by Seaglider (note the ‘(drop)’ text in the output) while waiting for the prompt – the buffer that was supposed to receive the numeric data remained empty. Thus no data was available to use when outputting the sample results.

8.2 Using %[]

Let’s try again using a %[] wait sequence at the end of the ‘sample=’ command string:

The sample command now looks like this:

```
sample=%r%pOutputFormat=1%r%pTS%r%[S>]
```

The output from the sample command (with debug turned off) is now the sequence of four floating point values that we expect:

```
Enter selection (1-14,CR): 4
4014.541,HGPCTD,N,TS
   -0.18, 23.1885, 0.00006,    0.00
```

After turning on debug mode the excerpted ‘sample=’ output is as follows:

```
4023.698,HGPCTD,N,send '<CR>'
4023.738,HGPCTD,N,wait for prompt
4023.785,HGPCTD,N,recv '<CR>' (drop)
4023.833,HGPCTD,N,recv '<LF>' (drop)
4023.882,HGPCTD,N,recv 'S'    (match)
4023.931,HGPCTD,N,recv '>'    (match)
4023.981,HGPCTD,N,(got the prompt)
4024.027,HGPCTD,N,send 'O'
4024.063,HGPCTD,N,send 'u'
4024.100,HGPCTD,N,send 't'
4024.137,HGPCTD,N,send 'p'
4024.173,HGPCTD,N,send 'u'
4024.210,HGPCTD,N,send 't'
4024.246,HGPCTD,N,send 'F'
4024.283,HGPCTD,N,send 'o'
4024.320,HGPCTD,N,send 'r'
4024.356,HGPCTD,N,send 'm'
4024.393,HGPCTD,N,send 'a'
4024.429,HGPCTD,N,send 't'
4024.466,HGPCTD,N,send '='
4024.503,HGPCTD,N,send '1'
4024.539,HGPCTD,N,send '<CR>'
4024.579,HGPCTD,N,wait for prompt
4024.626,HGPCTD,N,recv 'O'    (drop)
4024.675,HGPCTD,N,recv 'u'    (drop)
4024.723,HGPCTD,N,recv 't'    (drop)
4024.772,HGPCTD,N,recv 'p'    (drop)
4024.820,HGPCTD,N,recv 'u'    (drop)
4024.869,HGPCTD,N,recv 't'    (drop)
4024.918,HGPCTD,N,recv 'F'    (drop)
4024.966,HGPCTD,N,recv 'o'    (drop)
4025.015,HGPCTD,N,recv 'r'    (drop)
4025.063,HGPCTD,N,recv 'm'    (drop)
4025.112,HGPCTD,N,recv 'a'    (drop)
4025.160,HGPCTD,N,recv 't'    (drop)
4025.209,HGPCTD,N,recv '='    (drop)
4025.258,HGPCTD,N,recv '1'    (drop)
4025.306,HGPCTD,N,recv '<CR>' (drop)
4025.354,HGPCTD,N,recv '<LF>' (drop)
4025.403,HGPCTD,N,recv 'S'    (match)
```

```
4025.752,HGPCTD,N,recv '>'    (match)
4025.802,HGPCTD,N,(got the prompt)
4025.848,HGPCTD,N,send 'T'
4025.884,HGPCTD,N,send 'S'
4025.921,HGPCTD,N,send '<CR>'
4025.962,HGPCTD,N,wait 5 secs for sequence [S>]
4026.025,HGPCTD,N,recv 'T'
4026.068,HGPCTD,N,recv 'S'    (match)
4026.118,HGPCTD,N,recv '<CR>'
4026.160,HGPCTD,N,recv '<LF>'
4028.454,HGPCTD,N,recv ' '
4028.497,HGPCTD,N,recv ' '
4028.539,HGPCTD,N,recv ' '
4028.582,HGPCTD,N,recv '-'
4028.625,HGPCTD,N,recv '0'
4028.668,HGPCTD,N,recv '.'
4028.710,HGPCTD,N,recv '1'
4028.753,HGPCTD,N,recv '8'
4028.796,HGPCTD,N,recv ','
4028.839,HGPCTD,N,recv ' '
4028.881,HGPCTD,N,recv '2'
4028.924,HGPCTD,N,recv '3'
4028.967,HGPCTD,N,recv '.'
4029.010,HGPCTD,N,recv '1'
4029.053,HGPCTD,N,recv '8'
4029.095,HGPCTD,N,recv '9'
4029.138,HGPCTD,N,recv '5'
4029.181,HGPCTD,N,recv ','
4029.224,HGPCTD,N,recv ' '
4029.266,HGPCTD,N,recv '0'
4029.309,HGPCTD,N,recv '.'
4029.352,HGPCTD,N,recv '0'
4029.395,HGPCTD,N,recv '0'
4029.437,HGPCTD,N,recv '0'
4029.480,HGPCTD,N,recv '0'
4029.523,HGPCTD,N,recv '6'
4029.566,HGPCTD,N,recv ','
4029.608,HGPCTD,N,recv ' '
4029.651,HGPCTD,N,recv ' '
4029.694,HGPCTD,N,recv ' '
4029.737,HGPCTD,N,recv ' '
4029.779,HGPCTD,N,recv '0'
4029.822,HGPCTD,N,recv '.'
4029.865,HGPCTD,N,recv '0'
4029.908,HGPCTD,N,recv '0'
4029.950,HGPCTD,N,recv '<CR>'
4029.993,HGPCTD,N,recv '<LF>'
4030.035,HGPCTD,N,recv 'S'    (match)
4030.085,HGPCTD,N,recv '>'    (match)
4030.135,HGPCTD,N,got expected sequence [S>]
```

```
4030.193,HGPCTD,N,close port
4030.232,HGPCTD,N,turning GPCTD off
4030.300,HGPCTD,N,TS
   -0.18, 23.1895, 0.00006,    0.00
```

Note that after the ‘TS’ command is sent to the device, the numeric output that returns is no longer marked as dropped. Waiting for a wait sequence causes the output data to be stored in the output buffer instead of ignored. The received data is then available for output as floating point numbers.

8.3 Important Points

Use %[] to wait for the device prompt when the device output is needed for display or for capture into a file. Use %p to wait for the device prompt when all output up until the device prompt is seen is unimportant and can be dropped. Use the .cnf debugger to help discover why you aren’t getting the output you think you should be getting.

The end of the sample command and the end of the metadata command are two places where you will usually need to use %[] instead of %p to wait for the prompt.

9 EXAMPLE – SEABIRD GPCTD

In this section the .cnf file required to support the Sea-Bird Electronics, Inc. Glider Payload CTD (GPCTD) is presented and discussed.

The .cnf file for the GPCTD is called payload.cnf. The contents are as follows:

```
name=GPCTD
prefix=pc
timeout=5000
powerup-timeout=0
baud=9600
warmup=400
voltage=10
current=0.018
cmdprefix=$PC_
prompt=S>
datatype=u
wakeup=%9%9%r%9%9%9%9%9%F%r%p
sample=%r%pOutputFormat=1%r%pTS%r%[S>]
start=%r%pInterval=%x%r%pStart%F%r%p
stop=%r%pStop%F%r%9%p%9Stop%r%p
clock-set=%F%r%pDATETIME=%{%m%d%Y%H%M%S}%r%p
clock-read=%F%r%pDS%F%r%[vMain]
clock-sync=gps2
metadata=%F%r%pUH%F%r%[S>]
download=%F%r%pOutputFormat=0%F%r%pUC%c%r
cleanup=%r%pResetLogging%F%r%[confirm%r%n]ResetLogging%F%r%p
pre-dive=%r%pResetLogging%F%r%[confirm%r%n]ResetLogging%F%r%p
post-stop=off
post-clock=off
post-transfer=off
param-x=INTERVAL
```

Notice that the string substitution characters are used frequently throughout the file. It makes the file look a bit complicated at first glance, but if you patiently work your way through the various sequences you'll see that its really fairly simple.

The 'name=' field provides a name that will appear in the Seaglider menu system, so make sure to use something that is close to the identity and purpose of your sensor so that it is easy to identify.

The 'prefix=' field gives the two character prefix that will be assigned to data files associated with your sensor. Try to choose something that is both a reminder of your new sensor, and something that is not in use elsewhere on your Seaglider.

The baud, voltage, current, and warmup, and prompt values come directly from the documentation provided with your sensor. If necessary, contact the sensor manufacturer directly to get any of these values that are not apparent from the documentation.

The 'cmdprefix=' value is typically derived from the 'prefix=' field so that the data file prefixes and the cmdfile parameters have similar values.

The 'wakeup=' string has lots of %9 sequences in it. The %9 just means to wait for 90 milliseconds. The %r means to send a <CR> to the device. If you've worked with serial port based sensors before then you probably remember setting up a minicom/teraterm/hyperterm connection to a device, powering on the device, and then hitting <CR> and waiting (and repeating) until you got the device to output its prompt. That is exactly what the wakeup string is designed to accomplish.

The 'sample=' string is the command that is intended to get your sensor to output a line of data so that you know it is working. In the above 'sample=' string, the command that causes data to be output is the 'TS' command. The OutputFormat command just causes the TS command to output its data in a human-readable way. Note how the %[] wait sequence is used at the end of this command string so that the data will be returned to the logdev instead of being ignored by logdev.

The 'start=' string provides the command that causes the sensor to go into continuous sampling mode. The start command for this particular sensor is interesting because it shows an example of using a cmdfile parameter in a substitution sequence. The %x sequence means that the value for the param-x value should be provided to the sensor. Here cmdprefix=$PC_, and param-x=INTERVAL, so the %x is replaced by the current value of the $PC_INTERVAL cmdfile parameter. The pilot can thus change a cmdfile parameter to easily affect the sampling interval for the GPCTD sensor.

The 'stop=' string gives the command required to stop continuous sampling by the sensor. The documentation for the GPCTD says that you may need to give the 'stop' command more than once to get the sensor to stop collecting data, so the command string specifies 'stop' more than once. The second stop is usually not required but does no harm – when CNF debug mode is enabled, you can usually see a warning to the effect of 'stop command is only valid when logging has been started'.

The 'clock-set=' string shows how the %{} sequence is used to potentially reorder the various fields of the date and time to meet the needs of various sensors. Note that the clock-set command will be sent as specified by the clock-sync command. Here, the clock-set command will be sent whenever the gps2 device gets a fix.

The ‘download=’ command is used to tell the sensor to send the collected output to Seaglider over the serial port. Seaglider will receive the data and write it into a file on the Seaglider compact flash card. Before starting the download Seaglider will issue the metadata command if it has been provided. Any data returned by the sensor will be written at the start of the resultant data file. Note that the metadata command ends in the wait sequence %[S>]. Logdev will wait for metadata output from the sensor until the specified prompt string is seen, or until the read times out, or until the (4094 byte) output buffer fills up. Occurrence of any of these conditions terminates the wait for metadata.

The download= command does not end in the wait sequence, nor in a %p sequence to wait for the prompt. Logdev will keep receiving 4094 byte data buffers from the sensor and writing them to the output file until it sees the prompt string (or until the read times out).

The cleanup= command sends ‘ResetLogging’ to the device, then waits for a sequence that ends in ‘confirm<CR><LF>’, and then confirms by sending ‘ResetLogging’ again.

The pre-dive= command is invoked before the Seaglider starts each dive. Since the Seaglider will ask the sensor to download its data after each dive, the data files on the sensor are no longer needed. Removing them before the next dive starts can be accomplished by including the command to remove them in the pre-dive command.

As you write the .cnf file for your device, start out simply and don’t try to use every possible feature immediately. Make small incremental changes. Work on one command string at a time. Use the CNF debug feature. Do lots and lots of testing before going to sea. Capture the output from the menu system window, take the time to analyze it, and learn something from each test run.

10 REMAINING TASKS

Once your logger sensor is producing data files, and the data files are being transferred from the sensor to the Seaglider, the files should be uploaded to the base station according to the $xx_XMITPROFILE cmdfile parameter.

10.1 Test Launch

After testing your .cnf file as much as possible, run repeated test dives (in the Seaglider menu system, select 'launch' and then 'test') and watch the messages that are output. Alternatively, set $CAPUPLOAD,1 so that the capture file is uploaded to the base station, and then review it in a text editor. Review the output and look for error messages indicating problems with your .cnf file commands.

10.2 File Uploads

If you specified a download= or xmodem= command then Seaglider should ask your logger device to send over its data after the dive (depending on your $XX_XMITPROFILE and $XX_UPLOADMAX parameters). You should be able to observe this in the menu system output or in the capture file. If data is transferred from your sensor to the Seaglider, then it should also be transferred to the base station. Look for files that start with the prefix= value from your .cnf file.

10.3 Base Station

When the Seaglider transfers your device's data files to the base station, the raw data files will appear with the two-character prefix you gave in the 'prefix' field of your .cnf file. If your prefix was 'md', you might find files like 'md0001az.x00' and 'md0001b.x00'.

The next step is to get the Seaglider base station code to reassemble the data file fragments into a single file. The single file will have a name like 'md0001a.x' or 'md0001b.r'.

To get the basestation to reassemble your data file fragments, you need to copy your device's CNF file into the /usr/local/basestation/Sensors/ directory. You also have to add an entry to the file /usr/local/basestation/Sensors/.sensors. Review some of the other entries and then add one for your device. Your entry should include 'process_data_files'.

The basestation tries to collect all data related to a dive into a single file in netCDF format. If you want to include your logger's data in the netCDF file you will probably need to specify a python script sensor extension instead of just using your .cnf file. The python script must be placed in /usr/local/basestation/Sensors, and the .sensors file must name your script and the functions to be used. Use the payload_ext.py .sensors entry and the payload_ext.py file as examples. Your setup should specify both 'init_logger' and 'process_data_files'.

The details of these tasks are beyond the scope of this document. KUTI will try to address these tasks in a future document.

10.4 Calibration Data

Plotting of your data may require the presence of per-sensor calibration values. You may have to add these calibration values to the sg_calib_constants.m file on the base station. You may also need to take additional steps to get the calibration values into the .nc file. These tasks are beyond the scope of this document. KUTI will try to address these tasks in a future document.

10.5 Plotting

Once the logger data is in a .eng file or netCDF file and time scale information is available, the data can be plotted using matlab or another tool. The specifics of this are outside the scope of this document. See future document.

APPENDIX A: LOAD CNF FILE

This section describes how to transfer a CNF file onto the Seaglider, and how to load the CNF file into the glider's logger library.

A.1 Check for Existing Filename

Power on the Seaglider and wait until the main menu is displayed.

Before transferring your CNF file to the Seaglider you need to make sure that the file you are about to transfer does not already exist on Seaglider's compact flash. If the file already exists the file transfer command will fail.

To check if the file exists, type **pdos** at the main menu prompt to exit to the picoDos prompt.

Use the 'dir' command to check if the file exists:

```
picoDos>> dir mydev.cnf
```

Note that picoDOS always lists the directory contents using upper case.

If the file is found, you can rename it:

```
picoDos>> ren mydev.cnf mydev.bck
```

or you can simply remove it:

```
picoDos>> del mydev.cnf
```

A.2 Transfer CNF file to Seaglider

To transfer the file use the 'xr' command:

```
picoDos>> xr mydev.cnf
```

After entering this command, use the terminal emulator's 'file send' feature to send the CNF file to the Seaglider using xmodem. The exact mechanism for this depends on the terminal program you are using.

If you are using minicom, enter <CTRL>A<SHIFT>Z to get to the main minicom menu. Enter 'S' to send a file, and choose xmodem from the resulting pop-up menu. When the file browser appears use it to browse your PC's file system to locate the CNF file you wish to transfer.

A.3 Strip CNF file

Next, strip pad bytes from the end of the CNF file you just transferred:

```
picoDos>> strip1a mydev.cnf
```

The CNF file is now ready for use. Return to the Seaglider software:

picoDos>>quit

You should now be back within the Seaglider menu system.

A.4 Load CNF into Logger Library

With the Seaglider 'Main Menu' displayed, you should see:

```
------ Main Menu ------
   1 [param   ] Parameters and configuration
   2 [hw      ] Hardware tests and monitoring
   3 [modes   ] Test operation modes and files
   4 [pdos    ] PicoDOS commands (and exit)
   5 [launch  ] Pre-launch
Enter selection (1-5,CR): 1
```

Make choice 1 (param).
You should now see the 'Edit parameters' menu:

```
------ Edit parameters ------
*Flight control and mission definition
   1 [basic   ] Basic mission and glider parameters
   2 [dive    ] Dive parameters
   3 [flight  ] Flight parameters
   4 [surface ] Surface parameters
   5 [rafos   ] RAFOS parameters
   6 [password] Set/show glider login password
   7 [telnum  ] Set/show basestation phone number
   8 [altnum  ] Set/show basestation alternate phone number
*Pitch, roll, VBD
   9 [pitch   ] Pitch parameters
  10 [roll    ] Roll parameters
  11 [vbd     ] VBD parameters
*Sensors and peripherals
  12 [config  ] Hardware configuration parameters
  13 [pressure] Pressure (external) parameters
  14 [intpress] Pressure (internal) parameters
  15 [compass ] Compass parameters
  16 [altim   ] Altimetry parameters
  17 [seabird ] Seabird CT calibration
  18 [power   ] Power parameters
*Utility
  19 [all     ] Edit all parameters
  20 [validate] Validate parameters
```

```
   21 [details ] Show parameter details
   22 [show    ] Show changed parameters
   23 [clear   ] Clear changed parameters
   24 [save    ] Save parameters by name to file
   25 [dump    ] Dump parameters to screen
   26 [load    ] Load parameters from file
   27 [reset   ] Reset to defaults
   CR)  Return to previous
Enter selection (1-27,CR): 12
```

Make choice 12 (config).

You should now see the 'Edit hardware configuration parameters' menu:

```
------ Edit hardware configuration parameters ------
    1 [show    ] Show current hardware configuration
    2 [motherbo] Configure motherboard
    3 [compass ] Configure compass
    4 [compass2] Configure spare compass
    5 [phone   ] Configure phone
    6 [gps     ] Configure GPS
    7 [xpdr    ] Configure XPDR
    8 [sensor  ] Configure sensor
    9 [logger  ] Configure logger sensor
   10 [seradd  ] Add device cnf to hardware library
   11 [serdel  ] Delete device cnf from hardware library
   12 [serlib  ] List device cnf files in hardware library
   13 [logadd  ] Add logger cnf to hardware library
   14 [logdel  ] Delete logger cnf from hardware library
   15 [loglib  ] List logger cnf files in hardware library
   16 [pressure] Configure external pressure sensor
   17 [param   ] Edit parameters directly
   CR)  Return to previous
Enter selection (1-17,CR): 15
```

Before loading the CNF file, we should check to see if any loglib slots are available. Use choice 15 (loglib) list the contents of the library. The output might look like this:

```
15.071,SSENSOR,N,Current user logger files:
15.166,SSENSOR,N,  0: [empty]
15.241,SSENSOR,N,  1: [empty]
```

The above output indicates that both locations in loglib are available.

Now use choice 13 (logadd) on the 'Edit hardware configuration parameters' menu to add your configuration file:

Enter selection (1-17,CR): **13**
9.752,SSENSOR,N,Current user logger files:
9.845,SSENSOR,N, 0: [empty]
9.919,SSENSOR,N, 1: [empty]

User logger number (0-1,CR): **0**
.cnf file for new logger: **new.cnf** **(Note: use lowercase here)**
17.798,SSENSOR,N,reading cnf new.cnf
18.017,SSENSOR,N,assigned SBE_CT to sensor slot 1 (p = 2 [index 0; port 2])
18.205,SSENSOR,N,assigned SBE_O2 to sensor slot 2 (p = 20 [index 1; port 4])
18.398,SSENSOR,N,assigned WL_BB2FLVMG to sensor slot 3 (p = 35 [index 2; port 3)
19.030,SPOWER,N,Iridium_during_init saved current=0.10300
19.200,SPOWER,N,Iridium_during_connect saved current=0.16000
19.374,SPOWER,N,Iridium_during_xfer saved current=0.22300
19.545,SPOWER,N,GPS saved current=0.02696
19.694,SPOWER,N,TT8 saved current=0.01496
19.842,SPOWER,N,TT8_Active saved current=0.01421
20.005,SPOWER,N,TT8_Sampling saved current=0.03743
20.164,SPOWER,N,TT8_CF8 saved current=0.04717
20.319,SPOWER,N,TT8_Kalman saved current=0.05915
20.480,SPOWER,N,Analog_circuits saved current=0.01200
20.646,SPOWER,N,Compass saved current=0.01573

The command output should show that your new CNF file was read. If the CNF file was valid then no errors should be displayed.

Note – you **must** specify the .cnf file name using lower case.

APPENDIX B: CONFIGURE DEVICE

Once your CNF file has been loaded into the loglib you can add a device of that type to the Seaglider configuration.

B.1 Check Existing Configuration

Before adding your new logger device you should review the current Seaglider sensor configuration to see which logger slots are available.

From the main menu, choose ‘param’ followed by ‘config’ to get to the ‘Edit hardware configuration parameters’ menu:

```
------ Edit hardware configuration parameters ------
   1 [show    ] Show current hardware configuration
   2 [motherbo] Configure motherboard
   3 [compass ] Configure compass
   4 [compass2] Configure spare compass
   5 [phone   ] Configure phone
   6 [gps     ] Configure GPS
   7 [xpdr    ] Configure XPDR
   8 [sensor  ] Configure sensor
   9 [logger  ] Configure logger sensor
  10 [seradd  ] Add device cnf to hardware library
  11 [serdel  ] Delete device cnf from hardware library
  12 [serlib  ] List device cnf files in hardware library
  13 [logadd  ] Add logger cnf to hardware library
  14 [logdel  ] Delete logger cnf from hardware library
  15 [loglib  ] List logger cnf files in hardware library
  16 [pressure] Configure external pressure sensor
  17 [param   ] Edit parameters directly
  CR)  Return to previous
Enter selection (1-17,CR): 1
```

Make choice 1 to show the current sensor configuration. Your output might look something like this:

```
346.630,SUSR,N,Sensor in slot 1 is SBE_CT on port 2, TPU04/TPU05, nominally 'CT'
346.796,SUSR,N,Sensor in slot 2 is SBE_O2 on port 4, TPU08/TPU09, nominally 'O2'
346.962,SUSR,N,Sensor in slot 3 is WL_BB2FLVMG on port 3, TPU06/TPU07, nominall'
347.126,SUSR,N,Sensor in slot 4 is not installed
347.231,SUSR,N,Sensor in slot 5 is not installed
347.337,SUSR,N,Sensor in slot 6 is not installed
347.445,SUSR,N,Logger Sensor in logger slot 1 is not installed
347.573,SUSR,N,Logger Sensor in logger slot 2 is not installed
```

```
347.700,SUSR,N,Logger Sensor in logger slot 3 is not installed
347.827,SUSR,N,Logger Sensor in logger slot 4 is not installed
347.948,SUSR,N,Motherboard is Rev.B.1
348.034,SUSR,N,Phone is NullModem
348.125,SUSR,N,    Phone is not configured for flow control and carrier detect
348.279,SUSR,N,    Phone supply is battery 2 with modeled current measurement
348.418,SUSR,N,GPS is Synthetic GPS w/o PPS
348.512,SUSR,N,Compass is TCM2-50
348.595,SUSR,N,    Spare compass is not installed
348.697,SUSR,N,XPDR is AAE_955
348.809,SUSR,N,External pressure sensor gain (1.000000) consistent with synthetic
```

Near the middle of the above output the configured Logger Sensors are displayed. In this case there are no logger sensors configured. Thus, all logger sensor slots are available.

B.2 Know What Port to Use

Before proceeding you should know the port and 'nominally' name where your new logger device will be connected. For example, 'port 6, nominally aux 1'. See section 4.1 for additional information.

B.3 Add New Logger Device

Now that you know what logger slots are free and what port your sensor will be connected to, you can configure the device. From the main menu, use 'param' and then 'config' to get to the 'Edit hardware configuration parameters' menu:

```
------ Edit hardware configuration parameters ------
   1 [show    ] Show current hardware configuration
   2 [motherbo] Configure motherboard
   3 [compass ] Configure compass
   4 [compass2] Configure spare compass
   5 [phone   ] Configure phone
   6 [gps     ] Configure GPS
   7 [xpdr    ] Configure XPDR
   8 [sensor  ] Configure sensor
   9 [logger  ] Configure logger sensor
  10 [seradd  ] Add device cnf to hardware library
  11 [serdel  ] Delete device cnf from hardware library
  12 [serlib  ] List device cnf files in hardware library
  13 [logadd  ] Add logger cnf to hardware library
  14 [logdel  ] Delete logger cnf from hardware library
  15 [loglib  ] List logger cnf files in hardware library
```

```
   16 [pressure] Configure external pressure sensor
   17 [param   ] Edit parameters directly
   CR)  Return to previous
Enter selection (1-17,CR): 9
```

To add a logger device use choice 9 (logger). You will see output similar to what follows. Here the new logger sensor's device name is 'NEW' (the device name comes from the 'name' field in the .cnf file). It is being added to logger slot 1 (always use the lowest free slot number). It is being installed on port 6:

```
Configure device in slot (1-4,CR): 1
81.929,SUSR,N,
Configuring attached logger sensor:
  1) SciCon
  2) NEW
  3) not installed

Enter selection: 2
  0) port 0, TPU-1/TPU-1, nominally 'null'
  1) port 1, TPU02/TPU03, nominally 'compass'
  2) port 2, TPU04/TPU05, nominally 'CT' IN USE by SBE_CT
  3) port 3, TPU06/TPU07, nominally 'Optics 1' IN USE by WL_BB2FLVMG
  4) port 4, TPU08/TPU09, nominally 'O2' IN USE by SBE_O2
  5) port 5, TPU10/TPU11 (mux channel 0), nominally 'Optics 2'
  6) port 6, TPU10/TPU11 (mux channel 1), nominally 'aux 1'
  7) port 7, TPU10/TPU11 (mux channel 2), nominally 'aux 2'
  8) port 8, TPU10/TPU11 (mux channel 3), nominally 'altimeter'

Enter selection: 6
p = 134, port_idx = 6, dev_idx = 8
90.560,HTT8,N,Updating parameter $LOGGERDEVICE1 to 134
90.716,SSENSOR,N,assigned NEW to logger sensor slot 1 (p = 134)
```

Verify that the configuration was updated by choosing option 1 (show) from the 'Edit hardware configuration parameters' menu to show the current configuration:

```
Enter selection (1-17,CR): 1
191.511,SUSR,N,Sensor in slot 1 is SBE_CT on port 2, TPU04/TPU05, nominally 'CT'
191.677,SUSR,N,Sensor in slot 2 is SBE_O2 on port 4, TPU08/TPU09, nominally 'O2'
191.843,SUSR,N,Sensor in slot 3 is WL_BB2FLVMG on port 3, TPU06/TPU07,
  nominally 'Optics 1'
192.007,SUSR,N,Sensor in slot 4 is not installed
192.113,SUSR,N,Sensor in slot 5 is not installed
192.218,SUSR,N,Sensor in slot 6 is not installed
192.349,SUSR,N,Logger sensor in logger slot 1 is NEW on port 6, TPU10/TPU11,
   (mux channel 1), nominally 'aux 1'
192.541,SUSR,N,Logger Sensor in logger slot 2 is not installed
```

192.668,SUSR,N,Logger Sensor in logger slot 3 is not installed
192.796,SUSR,N,Logger Sensor in logger slot 4 is not installed
192.917,SUSR,N,Motherboard is Rev.B.1
193.002,SUSR,N,Phone is NullModem
193.094,SUSR,N, Phone is not configured for flow control and carrier detect
193.247,SUSR,N, Phone supply is battery 2 with modeled current measurement
193.387,SUSR,N,GPS is Synthetic GPS w/o PPS
193.481,SUSR,N,Compass is TCM2-50
193.564,SUSR,N, Spare compass is not installed
193.666,SUSR,N,XPDR is AAE_955
193.778,SUSR,N,External pressure sensor gain (1.000000) consistent with synthetic

Notice that the logger slot 1 is now occupied by a logger device called NEW.

If your new device name does not appear in the list on the 'Configuring attached logger sensor' menu, you will have to do some investigation. First check if your .cnf file is listed as being included in the logger library (on the 'Edit hardware configuration parameters' menu, choose the 'loglib' option). If not, then perhaps the 'logadd' step failed. If the .cnf file is listed in the loglib output, then make sure you specified the .cnf filename on the 'logadd' step using lower case (if you use upper case, the .cnf file gets loaded into the library but does not appear on the list of logger devices that you can configure). If necessary, use 'logdel' to delete the .cnf file from the logger library, and then repeat the 'logadd' step using lowercase for the .cnf file name.

B.4 Save Changes

Finally, save your changes. Exit the 'Edit hardware configuration parameters' menu and answer 'Y' when asked if you want to re-initialize the hardware configuration:

Enter selection (1-17,CR):
Re-initialize hardware configuration [N] **y**
139.305,SSENSOR,N,reading cnf new.cnf
139.526,SSENSOR,N,assigned SBE_CT to sensor slot 1 (p = 2 [index 0; port 2])
139.716,SSENSOR,N,assigned SBE_O2 to sensor slot 2 (p = 20 [index 1; port 4])
139.910,SSENSOR,N,assigned WL_BB2FLVMG to sensor slot 3 (p = 35 [index 2; port)
140.112,SSENSOR,N,assigned NEW to logger sensor slot 1 (p = 134)
140.413,SSENSOR,N,Error - could not open loggers.cmd to load logger parameters
140.835,HTT8,N,Writing NVRAM...done.
149.568,SPOWER,N,Iridium_during_init saved current=0.10300
149.740,SPOWER,N,Iridium_during_connect saved current=0.16000
149.915,SPOWER,N,Iridium_during_xfer saved current=0.22300
150.086,SPOWER,N,GPS saved current=0.02696
150.237,SPOWER,N,TT8 saved current=0.01496
150.387,SPOWER,N,TT8_Active saved current=0.01421
150.548,SPOWER,N,TT8_Sampling saved current=0.03743
150.710,SPOWER,N,TT8_CF8 saved current=0.04717
150.867,SPOWER,N,TT8_Kalman saved current=0.05915

```
151.029,SPOWER,N,Analog_circuits saved current=0.01200
151.197,SPOWER,N,Compass saved current=0.01573
```

Notice the error about not being able to open loggers.cmd to load the logger parameters. This error will be displayed when you configure the first logger device on your system. It will go away when the Seaglider connects to the base station and pulls over a cmdfile containing the parameters specific to your new logger device (e.g., $xx_PROFILE, $xx_RECORDABOVE, etc.).

APPENDIX C: REMOVE DEVICE

If you need to change what port a device uses, or update it's CNF file, or simply remove the device from the Seaglider, you must first remove the device from the Seaglider's configuration.

To do this, start at the main menu and choose 'param' followed by 'config'. You should now see the 'Edit hardware configuration parameters' menu:

```
------ Edit hardware configuration parameters ------
   1 [show    ] Show current hardware configuration
   2 [motherbo] Configure motherboard
   3 [compass ] Configure compass
   4 [compass2] Configure spare compass
   5 [phone   ] Configure phone
   6 [gps     ] Configure GPS
   7 [xpdr    ] Configure XPDR
   8 [sensor  ] Configure sensor
   9 [logger  ] Configure logger sensor
  10 [seradd  ] Add device cnf to hardware library
  11 [serdel  ] Delete device cnf from hardware library
  12 [serlib  ] List device cnf files in hardware library
  13 [logadd  ] Add logger cnf to hardware library
  14 [logdel  ] Delete logger cnf from hardware library
  15 [loglib  ] List logger cnf files in hardware library
  16 [pressure] Configure external pressure sensor
  17 [param   ] Edit parameters directly
  CR)  Return to previous
Enter selection (1-17,CR): 1
```

Devices are removed by specifying the slot number they current occupy. To see what devices are installed in what slots, use option 1 (show). The output might look like the following:

```
8.087,SUSR,N,Sensor in slot 1 is SBE_CT on port 2, TPU04/TPU05, nominally 'CT'
8.250,SUSR,N,Sensor in slot 2 is SBE_O2 on port 4, TPU08/TPU09, nominally 'O2'
8.414,SUSR,N,Sensor in slot 3 is WL_BB2FLVMG on port 3, TPU06/TPU07, nominally '
8.575,SUSR,N,Sensor in slot 4 is not installed
8.677,SUSR,N,Sensor in slot 5 is not installed
8.780,SUSR,N,Sensor in slot 6 is not installed
8.908,SUSR,N,Logger sensor in logger slot 1 is GPCTD on port 6, TPU10/TPU11 (mu'
9.100,SUSR,N,Logger Sensor in logger slot 2 is not installed
9.225,SUSR,N,Logger Sensor in logger slot 3 is not installed
9.349,SUSR,N,Logger Sensor in logger slot 4 is not installed
9.467,SUSR,N,Motherboard is Rev.B.1
9.550,SUSR,N,Phone is NullModem
```

```
9.639,SUSR,N,    Phone is not configured for flow control and carrier detect
9.789,SUSR,N,    Phone supply is battery 2 with modeled current measurement
9.926,SUSR,N,GPS is Synthetic GPS w/o PPS
10.017,SUSR,N,Compass is TCM2-50
10.099,SUSR,N,    Spare compass is not installed
10.199,SUSR,N,XPDR is AAE_955
10.310,SUSR,N,External pressure sensor gain (1.000000) consistent with synthetic
```

To remove a logger device from the configuration use choice 9 (logger). When prompted, enter the slot number for the logger device to configure. Then enter choice 3 (not installed) to uninstall the device:

```
Configure device in slot (1-4,CR): 1
20.222,SUSR,N,Current logger sensor in slot 1 is GPCTD on port 6, TPU10/TPU11
  (mux channel 1), nominally 'aux 1'
20.412,SUSR,N,
Configuring attached logger sensor:
  1) SciCon
  2) GPCTD
  3) not installed

Enter selection: 3
p = -1, port_idx = -1, dev_idx = -1
29.608,HTT8,N,Updating parameter $LOGGERDEVICE1 to -1
```

The logger device has now been removed from the system. Be sure to save the changes to the configuration by exiting the current menu (by entering <CR>) and then responding 'Y' when prompted to 'Re-initialize hardware configuration'.

APPENDIX D: UNLOAD CNF FILE

If you need to update a CNF file or free up the loglib slot it uses, then you must unload the CNF file from loglib. Before doing this you should be sure that any devices using the CNF file have been removed from the configuration (see preceding section).

To remove the CNF file from loglib, start at the main menu and choose ‘param’ followed by ‘config’. You should now see the ‘Edit hardware configuration parameters’ menu. Enter choice 15 (loglib) to list the logger CNF files stored in loglib. Enter choice 14 (logdel) to remove one of the files from loglib. When done, exit the ‘hardware configuration parameters’ menu and respond ‘Y’ when prompted to ‘Re-initialize hardware configuration’.

APPENDIX E: RELOADING THE CNF FILE

Each time you edit the CNF file and want to test the changes, you need to remove the device (Appendix C), unload the CNF file (Appendix D), load the updated CNF file (Appendix A), and reconfigure the device (Appendix B).

Here is a quick summary of the steps involved:

1. Power on Seaglider.
2. At main menu enter choice 1 to go to 'param' mode.
3. At 'Edit parameters' menu enter choice 12 to go to 'config' mode.
4. At 'Edit hardware' menu enter choice 9 to go to (un)configure a 'logger'.
5. Enter the slot number where the logger is installed (or enter <CR> to return to 'Edit hardware' menu and choose choice 1 to show the configuration and see the logger slots that are in use).
6. After entering the logger slot number, choose 'not installed' to uninstall it.
7. At 'Edit hardware' menu enter choice 14 (logdel).
8. Choose the number corresponding to the .cnf file you want to remove from loglib.
9. At 'Edit hardware' menu hit <CR> to exit the menu.
10. At the 'Re-initialize hardware' prompt enter 'Y'.
11. Return to the main menu.
12. Enter 'pdos'.
13. In picodos, rename or delete the .cnf file you want to update.
14. Now you can reinstall the updated .cnf file
15. In picodos, use 'xr new.cnf' file to receive the new file. Use the minicom command menu to send over that file from the PC.
16. In picodos, 'strip1a new.cnf'
17. In picodos, enter 'main' to restart the glider software and menu system.
18. Use menu system param/config/logadd to add the updated cnf file to the loglib library. Watch to make sure no errors are reported when the device is parsed.
19. Use menu system param/config/logger to configure a device using the new cnf file.
20. Finally, re-initialize the hardware [Y].