Seaglider Rev E Software Notes

rev 1.1, 17-Jun-2024

1. Critical differences with Rev B and/or HII Rev E

1. There are no motor retries in Rev E. The *_AD_RATE parameters have been removed.
2. Pressure sensor software gain is a supervisor setting (hw/super/setup from the main menu). $AD7714Ch0Gain has been removed.
3. science and targets file formats have changed (from older versions of Rev B)
4. tcm2mat file format has changed (also applies to earlier HII supplied versions of Rev E firmware)
5. Firmware update procedure has changed.
6. XMODEM has been replaced by YMODEM in most use cases. Raw protocol is more robust and should be the default for file transfers via Iridium ($PROTOCOL,9).
7. The interpretation of $COMM_SEQ has changed.
8. $LOITER_N_DIVE must be set to activate loiters. Setting $T_LOITER alone is not sufficient.
9. The bathy command has been removed from epdos. Use the equivalent menu option modes/bathy/verify (menu modes/bathy/verify from epdos or a batch file).
10. The bits of $NOCOMM_ACTION are different when compared to HII Rev E firmware.
11. The raw protocol has changed. Latest firmware requires the installation of rawrcv2 (from basestation3 sources) and linking or copying rawrcv2 to rawrcvb when using $PROTOCOL,9.

2. Firmware updating

Rev E firmware is updated via an onboard bootloader, either via serial port or via file on the microSD card. When (re)booting, the glider will pause at a ‘#’ prompt for approximately 1 second. Pressing enter at this prompt will activate the bootloader menu. If no key is pressed, regular glider startup proceeds. From the bootloader, you can load new firmware over serial via YMODEM, backup the current firmware to a file on microSD card, or restore the firmware from a previous backup.

New firmware can also be loaded by copying the firmware file (typically glider.bin) to a file named boot.bin on the microSD card. When the glider (re)boots, the bootloader detects the presence of this file, automatically loads the firmware to flash, and deletes the file. If you have access to the motherboard, you can load the file to the microSD card by powering the glider off, removing the card from the glider, inserting the card into a PC, copying the file, re-inserting the card into the motherboard and powering up the glider. If the motherboard is not accessible, you can load boot.bin to the card using file transfer commands from pdos (e.g., “yr” and then send boot.bin using YMODEM. Once boot.bin is transferred, reboot to flash the firmware.

3. Auto parameters

The latest Rev E firmware includes onboard auto-tuning of pitch gain, pitch center and VBD center. Tuning is controlled by new parameters, $C_VBD_AUTO_MAX, $C_VBD_AUTO_DELTA, $C_PITCH_AUTO_MAX, $C_PITCH_AUTO_DELTA, $PITCH_GAIN_AUTO_MAX, and $PITCH_GAIN_AUTO_DELTA. The algorithm is similar to tuning via regression plots and the classic “half the distance” rule. For each parameter, the glider computes onboard regression values for centers and pitch gain and applies a correction according to the DELTA value (0-1), with the correction not to exceed the MAX value. Typical values might be 0.5 for a DELTA and 100 (counts) for

$C_VBD_AUTO_MAX or $C_PITCH_AUTO_MAX or 2 (deg/cm) for $PITCH_GAIN_AUTO_MAX. When using auto tuning with VBD, it is important that $RHO and $MASS be set correctly. Sea-Bird coefficients also must be set correctly when using an SBE CT sail.

Roll centers adjusted via $ROLL_ADJ_GAIN and $ROLL_ADJ_DBAND are now carried over from dive-to-dive.

4. epdos commands and automatic batch file execution

Rev E implements its own epdos subsystem. You can enter epdos at any time by typing “pdos” at the menu prompt. Individual epdos commands can also be executed at the menu prompt by prepending with ! (e.g., !dir).

4.1 System commands

menu *menuPath args …*

Execute a command from the menu system. *menuPath* is a / separated list of menu, sub-menu and menu options as described by the [name] strings displayed in the menu system.

```
menu hw/vbd/ad pos=2000
```

for example will move the VBD to 2000 counts. This is equivalent to navigating to the hardware menu (hw), VBD sub-menu (vbd) and choosing the AD option to manually move the VBD control.

quit

Exit the epdos system and return to the menu system.

pdos [*/f*]

Exit the glider program. Reboots the glider firmware. The /f flag forces the exit (skips the confirmation question.

reboot

Sets $RELAUNCH to indicate an epdos commanded reboot and restarts the glider program.

clock [*mm/dd/yyyy HH:MM:SS*]

Read or optionally set the main real-time clock. This does not set the supervisor RTC (use hw/super/setrtc from the menu) or any peripheral clocks.

flash backup *a-f*

Save currently flashed firmware to a file, backup_x.bin on microSD card, where x is one of a-f.

sysclk *[speed] [hse]*

Display system clock information. If speed is specified, set CPU operating clock to speed. Specify hse to use the high-speed external oscillator as the system clock (this is almost always what you want). Use with care. An incorrect speed specification could leave the glider in an inoperative state.

scuttle *confirmation*

Scuttle the glider. This pitches the glider full forward and opens the valve.

wipe *confirmation*

Wipe the microSD card by filling the entire card with a random sequence of bytes.

4.2 Scripting commands

stroke

Stroke the watchdog.

time *command args* ...

Run *command* with *args*. Report the execution time.

repeat c*ount command args* …

Run *command* with *args count* times. *command* can be a batch file.

abort_if 0|1

(only useful inside a batch file) Stop batch file execution if the previous command returned False (0) or True (1). Not all commands return useful values.

delay *milliseconds*

Pause execution for *milliseconds*. Does not enter low power sleep.

sleep *seconds*

Enter low power sleep for *seconds*.

print *arg1 [arg2…] [> | >> dest]*

Print (echo) strings to screen (and capture file) or file *dest*. Arguments are expanded using a similar syntax as logdev and serdev serial strings. See argument expansion below.

4.3 File transfer commands

yr

Receive files via YMODEM.

ys *file*

Send *file* via YMODEM. In online mode (during Iridium call, via pdoscmds.bat for example), *file* must be a single file. From the console to a computer connected via serial port, *file* can be glob (e.g., sg*dz.a) to send multiple files.

raw_r *file*

Receive *file* via raw protocol.

raw_s *file*

Send *file* via raw protocol.

4.4 File utility and shell-like commands

gzip *originalFile compressedFile*

Compress a file.

gunzip *compressedFile uncompressedFile*

Uncompress a file.

tar *x|t[v][z] tarFile*

Extract (x) or display contents (t) of *tarFile*.

tar *c[v][z] tarFile glob [pathglob]*

Create a tar file. Specify z to create a gzip compressed tar file. v to operate verbosely. *path* is optional. If specified it will be pre-prended to the the file pattern *glob*. *path* can itself be a glob. For example, “tar cvz eng.tgz sg0*dz.a dv00*”, will search all directories matching dv00* for files matching sg0*dz.a and tar them into a single file.

md5 *file*

Compute and display MD5 hash of *file*.

mkdir *directory*

Create *directory* on the microSD card.

ren *oldName newName*

Rename a file.

mv *glob1* [*glob2* …] *directory*

mv is a directory aware version of rename, similar to the Unix mv command. Multiple glob glob patterns/filenames can be specified. The destination directory must exist. If there are just two arguments and the second argument is not an existing directory than mv behaves exactly like rename.

cp *file1 file2*

Copy a file.

del [/v] *glob1* [*glob2…*]

Delete files from microSD card.

rm *glob [path]*

Recursively delete files matching *glob*. Path is optional. If specified, it is the starting point (highest level) in the directory tree for the search for matching files.

dir *[glob1] [glob2...]*

Show directory entries for files matching a list of glob patterns. If no patterns are given, shows the entire root directory.

ls *[glob] [path]*

Recursively show directory entries for files matching *glob*, with optional starting *path*. If no arguments are given, lists the entire directory tree.

dirfile *file [glob] [path]*

Same as ls but results are saved to *file* rather than output to screen.

usage *[glob1] [glob2...]*

Count files and sum sizes for files matching a list of glob patterns. If not patterns are given, shows the entire root directory.

cat *glob1* [*glob2 …*] [> | >> *dest*]

Cat (output) files to console or optionally redirected to a file. Use > to overwrite *dest*. Use >> to append to *dest*.

trunc *file length*

Truncate *file* to *length* bytes.

grep /s *string1* [/s *string2 …*] [*/b linesBefore*] [*/a linesAfter*] *[/v] [/c] [/or] glob1 [glob2...]*

Search files matching a list of glob patterns line-by-line for matching string patterns. Files can be gzipped or plain text. Use /b and /a to control how many lines of context are printed before and after matched lines. /c specifies that only a count of matches will be returned. /v inverts the search - only non-matching lines are reported (or counted). By default multiple search conditions are and-ed together (i.e., a line in a file matches only if all strings are found on that line). Use */or* to change this to logical or.

xargs *pathspec filespec [commands] [args]*

Similar to a unix pipeline consisting of find piped to xargs, builds a list of files by separately globbing paths and filenames. Searches for directories matching *pathspec*, and then within each matched directory searches for files matching *filespec*. With no additional arguments a list of matched files is printed. If *command* is specified (possibly with its own arguments) the list of files is appended as additional arguments. Valid commands are any that take a list of files or glob patterns as the final arguments (dir, usage, del, grep, cat (without redirection)). "xargs dv001? sg*kz.a grep /s ,C," for example searches dives 10-19 capture files for the string ,C,.

4.5 Capture system commands

capvec [*subsystem level dest*]

With no arguments, prints the current capture vector information. With arguments, sets debugging verbosity/level (DEBUG, NORMAL, CRITICAL) and destination (BOTH, FILE, SCREEN, NONE) for *subsystem*.

capflush

Flush memory buffered capture contents to disk.

capclose *file [capFile]*

Close the current working capture contents (or *capFile* if specified) to *file* and start a new empty capture file. In online mode (during an Iridium call via pdoscmds.bat), the capture file has already been moved to a temporary name to store the output of the commands in pdoscmds.bat. To close and reset the “regular” capture file (in which all of the offline results have been stored), specify thisdive.kap as *capFile*.

4.6 Glider control and data file commands

target [*targetName*]

Re-read targets file. With no arguments, displays current targets. With an argument, sets *targetName* as the active target.

science

Re-read and report science file contents.

resend_dive *[/l | /d | /c] diveNumber [fragment1] [fragment2...]*

Queue log (/l), eng (/d), capture (/c), or all (no specifier) files from *diveNumber* for re-transmission. A list of specific fragments to resend is optional. If not provided, all fragments will be sent.

split *file*

Split *file* into fragments according to $N_FILEKB.

resend_dive */f file [fragment1] [fragment2…]*

Queue logger file *file* for re-transmission. A list of specific fragments to resend is optional. If not provided, all fragments will be sent.

4.7 Low-level filesystem and microSD commands

sdinfo

Report microSD card information (serial number, manufacturer info, capacity, etc.)

sdfree *[/v]*

Report microSD card free space. With the /v option, report verbose information about the filesystem (open files, etc.)

mem

Display memory (RAM) status.

format *confirmation*

Format the microSD card filesystem.

walkdsk *[/v]*

Recursively walk the entire filesystem to look for potential errors. /v turns on verbose reporting.

checkdsk *[/v]*

Recursively walk the entire filesystem and attempt to open and close every file.

fsck *[/v]*

Recursively walk the entire filesystem, opening and close every file. Tries to delete any file that cannot be opened.

4.8 Low-level nonvolatile memory commands

initnv *page*

Query the status of of nonvolatile storage page *page*. Page 0 is parameter memory. Page 1 is utility memory.

dumpnv *page*

Display low-level information about storage page *page*.

formatnv *page confirmation*

Format (erases all existing data) page. *confirmation* must be YES.

readnv *var*

Read value of utility variable *var*.

writenv *var value*

Write *value* to utility variable *var*.

4.9 Exec system commands

eval [*conditionName*]

If no conditionName is specified, evaluate all currently set conditions. If the condition evaluates true, execute the corresponding set number or pdos command.

clear *conditionName*

Delete conditionName from the list of active conditions.

condition name=*conditionName* [set=*setNumber* | pdos=*command*] expresssion=”*conditionalExpression*” init=”*expression*” [init=”*expression*” ...]

Define a new condition and add it to the list of active conditions.

exec [*setNumber*]

If no setNumber is given, show all active conditions. If setNumber is given, execute it.

4.10 Automatic batch file execution

While deployed, Rev E executes special batch files (if present) at multiple points in the state machine. Recognized names (and their point of execution) are: boot.bat, launch.bat, selftest.bat, connect.bat (Iridium connection), disconnect.bat, finish.bat (subsurface finish), dive.bat (dive start), climb.bat, apogee.bat, loiter.bat, escape.bat (only at start of escape), recovery.bat (every call cycle in recovery).

4.11 Argument expansion

Arguments to some commands are expanded using a similar syntax as logdev and serdev serial strings. Generally this is useful in batch files for specifying filenames that contain a changing dive number or timestamp, or in a print command to cause the value of a parameter or other accessible variable to be written into the capture file. Available expansions are:

%r = CR (13)
%n = NL (10)
%% = %
%e = esc (27)
%#xx = send hex byte xx - must specify both characters
%d = dive number (as 04d)
%g = glider ID (as 03d)
%T = temperature (deg C)
%D = depth (m)
%H = target magnetic heading
%V = soundspeed (m/s)
%W = vertical velocity (m/s)

%t = seconds into dive(s)
%p = pitch
%r = roll
%h = current heading
%H = desired heading
%G = GC flags
%l = estimated lat and lon (.5f,.5f)
%{} = interpret characters inside {} as strftime string
%($PARAM) = parameter value (%f for float parameters, %d for integer parameters)

5. tcm2mat

The compass calibration file for Rev E is named tcm2mat.cal. It is now a “first class” file in that it will be automatically downloaded by the glider from the basestation if it is present on the basestation. The file specifies pitch and roll calibration coefficients (rarely used), hard and soft iron for all compasses, control values for the onboard calibration algorithm (see $COMPASS_USE below), and control values for pressure filtering. Values are specified in typical key=value pairs, one per line. Values not specified explicitly are set to defaults. A typical file might only have hard0 and soft0 lines for example.

Valid keys are (Values in [] indicate choose one of 0, 1, or 2 to specify which compass this applies to. Almost always only 0 in general usage.):

- roll[0,1,2] = "0,1,0,0" roll coefficients: offset, scale, cos scale, sin scale
- pitch[0,1,2] = "0,1,0,0" pitch coefficients: offset, scale, cos scale, sin scale
- hard[0,1,2] = "0,0,0" hard iron coefficients
- soft[0,1,2] = "1,0,0,0,1,0,0,0,1" soft iron coefficients
- min-quality = 0.2 minimum circularity threshold before accepting a calibration result for operational use
- min-cover = 18 minimum coverage threshold (in tens of degrees) before accepting a calibration result for operational use
- Wf = 0.1 weight for 3D residuals relative to 1.0 for 2D (horizontal) residuals
- tail = 3 number of dives to run the calibration
- maxpts = 2000 maximum number of data points to include in calibration calculations
- pfilter = 3 number of points to use in pressure filter. Valid values are 1-3. 1=no filtering, 2=2-pt average, 3=3-pt median

Example:

```
pitch0="0 1 0 0"
roll0="0 1 0 0"
soft0="1.00 -0.033 -0.19 -0.095 1.08 0.167 -0.013 -0.028 1.15"
hard0="-467 -862 -739"
```

6. $COMPASS_USE and automatic compass calibration

As before, $COMPASS_USE controls the collection, reporting and use of magnetic data from the compass. It now also controls the onboard compass calibration algorithm. For normal operations without onboard calibration, it should have at least bit 2 (=4) set which trusts inputs from the compass, uses and reports fully calibrated values, and reports raw magnetic data in the engineering file for shoreside tracking and calculation of magnetic calibrations. Add 16384, 32768, and 65536 as appropriate for onboard soft and hard iron use. Bits 0,1,3-13 are rarely set. Consult the Parameter Reference Manual for complete usage.

- Bit 2 (4) Offboard cal mode (include mag data in engineering file):
  - 0 = do not report mag values in data file
  - 1 = report mag values in data file for use in onboard or offboard cal
- Bit 13 (8192) whether to use motherboard accelerometer for compass
  - 0 = use bit 11 to determine active compass
  - 1 = use motor controller (Rev E) or motherboard (Rev F) accelerometer for compass
- Bit 14 (16384, 0x4000) run post-dive calibration for soft and hard iron
  - 0 = do not run
  - 1 = run calibration according to control values in tcm2mat.cal
- Bit 15 (32768, 0x8000) prefer onboard calibration result over tcm2mat.cal file when reloading coefficients
  - 0 = always load coefficient from tcm2mat.cal
  - 1 = load coefficients from tcm2mat.aut if it exists
- Bit 16 (65536, 0x10000) use results from latest onboard calibration and save to tcm2mat.aut if quality and coverage requirements are met.
  - 0 = do not apply results
  - 1 = use and save calibration result if quality and circularity minima are met and lower than previously applied result.

The calibration algorithm is controlled by additional parameters that can be set in tcm2mat.cal:

- min-quality= Set the threshold for the circularity of the resulting calibrated points. Circularity is measured as the RMS deviation from a circle normalized by the calculated mean field strength (circle radius). Zero is a perfect score. Default is 0.2. Only used when bit 15 is set to use the calculated results as the current calibration.
- min-cover= Set the threshold for minimum coverage of the calibrated points, in tens of degrees. A value of 36 requires that data be collected around all compass points for example. Default is 18. Only used when bit 15 is set to use the calculated results as the current calibration.
- Wf= Set the weight for the 3D component of the residual (compared to 1.0 for horizontal) in the second step of the calibration (with soft iron). Default 0.1.
- tail= Set the number of dives over which to run the calibration. Three (default) uses the current dive plus two previous dives.
- maxpts= Sets the maximum number of points to use in the calibration. Points will be evenly selected from the dive data specified by the tail value. Default 2000.

Some useful typical values might be:

- start the mission at 0x1c004 to run auto calibration every dive and accept good or improved calibrations as active

- maybe change to 0xc004 to keep running the calibration algorithm to report results in the log file, but do not accept any further changes to the applied calibration. If the coefficients are re-read for any reason (a reboot for example), use the saved result from the onboard calibration.
- use 0x4004 to run the onboard calibration and report results in log files, but always use the coefficients from tcm2mat.cal.

If the calibration algorithm generates a new calibration for operational use, those results are written to tcm2mat.aut. Results from the calibration algorithm if run (bit 14) are reported in the $MAGCAL line of the log file. The current hard and soft iron values used during the dive are always reported in the $IRON line in the log file.

7. science file

The format of the Rev E science follows the targets format with each line defining a depth bin (the bottom of the bin) followed by a series of key=value pairs. For example

100	seconds=5.0	gc=180	sensors=112	profiles=333	dives=111	compass=1	pressure=1	
500	seconds=10.0	gc=300	sensors=122	profiles=331	dives=112	compass=1	pressure=1	timeout=10
1000	seconds=30.0	gc=300	sensors=120	profiles=311	dives=122	compass=1	pressure=1	timeout=12
loiter	seconds=60.0	gc=300	sensors=100		dives=122	compass=5	pressure=1	timeout=8
bottom limit of depth bin	base sampling interval in seconds	guidance and control interval	interval for each configured sensor (list of multipliers of seconds)	which profile (dive=1, climb=2, both=3) for each sensor (list of profile flags)	which dives for each sensor (1=every, 2=every other, etc.) (list of dive moduli)	interval for compass sampling (seconds multiplier)	interval for pressure sampling (seconds multiplier)	master timeout (seconds) for all sampling (default=10 if not specified)

Other than the depth bin which must be first, fields can appear in any order and must be tagged with their field name. The seconds, sensors, and gc fields are required; profiles, dives, compass, pressure, and timeout are optional and will default to every dive, every profile and always sample compass and pressure if not otherwise specified.

An optional bin, loiter, controls sampling during loitering at any apogee depth.

Each depth bin can also have a batch=file.bat attached to specify an arbitrary batch file to be executed when first starting sampling at that bin.

8. targets file

The format of the targets file remains the same relative to later versions of Rev B and all earlier Rev E software, but several new fields are present. One target is listed per line, and the target name must be listed first. The order of the other fields does not matter. Comment lines can be included, preceded by a /.

```
SEVEN          lat=4807.0      lon=-12223.0      radius=200      goto=SIX
SIX            lat=4806.0      lon=-12222.0      radius=200      goto=FIVE
FIVE           lat=4805.0      lon=-12221.0      radius=200      goto=EIGHT
FOUR           lat=4804.0      lon=-12220.0      radius=200      goto=EIGHT
```

KAYAKPT	lat=4808.0	lon=-12223.0	radius=100	goto=KAYAKPT
Target name - this can be any string of numbers and/or letters, without whitespace.	Latitude, in +/- ddmm.m; positive North	Longitude, in +/- dddmm.m; positive East	Radius, in meters, within which the glider determines it has reached the target.	Next target

Above is a typical targets file. Each target must contain all of lat, lon, radius, goto or head. radius values can be positive or negative. When negative, the target is achieved when the distance to the target is greater than the specified value. This is typically used for changing behaviors via the exec system when trying to stay within some watch circle or fenced region. Other fields include:

escape=KAYAKPT

Under $USE_ICE, the escape field specifies which target to move to if the glider would otherwise enter recovery, but the error condition is such that the glider can still navigate (voltage cutoffs, current cutoffs, battery exhausted, motor timeouts, fileystem errors). The escape field must specify a named target in the file and can vary for each named target. One possible use is to have the standard targets along a cyclical survey route all point to a single escape target that then points (through goto) to a series of targets that define an entire route to a convenient recovery location. The glider must have a recent navigation fix to escape by target. When a fix is available, a valid escape target overrides $ESCAPE_HEADING. When no fix is available (escaping due to $FIX_MISSING_TIMEOUT) $ESCAPE_HEADING is used to determine the escape route.

depth=100

Specifying a non-zero value for depth on a target means that target is achieved once the vehicle crosses a bathymetric contour. If the value is positive the target is achieved when crossing that contour from deep to shallow. When negative, the target is achieved by moving across that contour from shallow to deep. The glider compares the target depth against the depth last found by altimetry or by $T_NO_W on dive.

finish=90

Establishes a finish line through the target, perpendicular to the direction specified. The target is considered achieved when the difference between the bearing to the target and the finish direction is greater than 90 (or less than -90) degrees. Example 1: finish=90 specifies a north-south finish line drawn through the target; the target is achieved when the glider is east of the line. Example 2: finish=180 specifies an east-west finish line; target is achieved when glider is south of the line. If finish is -1 or no specification is made, no finish line will be tested.

head=180

The head field specifies that the glider should fly a fixed heading rather than point towards a fixed target. This is intended to be used with depth or timeout fields to define how the target is achieved. If the value is -1 or no value is specified then lat and lon must be specified. If lat, lon and head are all specified, heading is ignored.

exec=2

Execute control file set 2 on target completion.

timeout=3.0

Length of time (in days) that the glider should try to achieve this target. If the timeout is exceeded the glider will proceed to the target named by timeout-goto. If timeout-goto is not specified then the regular goto target will be used. If timeout is zero or no specification is made, no timeout applies.

dives=10

Length of time in dives that the glider should try to achieve this target. Dives are counted beginning at the dive the target becomes active. If the dives is exceeded the glider will proceed to the target named by timeout-goto (or goto). If dives is zero or no specification is made, no timeout applies.

timeout-goto=FOUR

On a timeout, goto FOUR rather than the target specified by the regular goto value.

timeout-exec=3

Execute control set 3 only on a timeout (either time or dives). On timeout, the set specified by exec will not be executed if timeout-exec also exists.

src=ans0

Specifies that this target is a possibly mobile navigation source that will be updating its lat and lon. The specified name must be a valid source identifier in a rafos.dat file.

src-timeout=3

Length of time in days to give up on this target if no updated position information is received.

src-timeout-goto=ans1

goto target when src-timeout is exceeded. If not specified, the goto target will be used.

src-timeout-exec=11

Control set to execute on src-timeout.

slow-progress=m,R,N,radius

Parameters used in determining whether the glider should complete this target due to inadequate progress or failure to keep up.

slow-progress-goto=FOUR

goto target when slow-progress is triggered. If not specified, the goto target will be used.

slow-progress-exec=11

Control set to execute on slow-progress.

fence-lat=7305.6

Center point latitude of the geographic fence.

fence-lon=-14504.5

Center point longitude of the geographic fence.

fence-radius=-20000

Radius (meters) of the fenced area. For negative values, target is achieved when the glider position is greater than this distance from the fence enter. For positive values, target is achieved when the glider position is inside this radius from fence center. Geographic fencing allows for watch circle like behaviors centered on points other than the actual target point. This might be useful when you want to follow a mobile navigation source, but only while the source remains within some boundary.

fence-goto=FOUR

goto target when a fence is triggered. If not specified, the goto target will be used.

fence-exec=21

The control set to execute on when target is completed due to a fence trigger.

goto and -goto fields my contain the specifiers CLOSE and CLOSEABS to indicate that the next target will be determined by proximity to the glider's current location.

- goto=CLOSE specifies that the next target will be the closest target to the current location, excluding the current target.
- goto=CLOSEABS specifies that the next target will be the closest target to the current location. The current target will be included in the search.
- goto=CLOSE:mooring*
- goto=CLOSEABS:mooring* adding :search specifies that the search for the closest target will be limited to those targets with names matching the search specifier.

9. Loiter

New parameters are available to more finely control behavior during apogee loiters. As before, $T_LOITER controls the loiter time. $T_MISSION and $T_DIVE should be set independently of the loiter, i.e., for the dive and climb without consideration of the loiter. It is no longer necessary to increase $T_MISSION to account for loiter time.

$LOITER_D_BOTTOM	Depth (Meters) below which to always pump during loiter. [default 0]
$LOITER_D_TOP	Depth (meters) above which to always bleed during loiter. Use a negative value to specify no pumps above this depth during loiter (if for example you do not want to bleed, but if the glider is oscillating around this depth due to internal waves or drifts above this depth, but then starts coming down again, do not pump to try to stem that downward velocity if above a negative value of $LOITER_D_TOP). [default 0]
$LOITER_DBDW	$DBDW value to use during loiters. Often this will be a small value to avoid making overly large corrections while trying to hold depth. [default 0]
$LOITER_N_DIVE	An integer value that determines which dives the glider will loiter at apogee. For values greater than or equal 1, the rule is that when the remainder of $DIVE divided by $LOITER_N_DIVE is zero, the glider will loiter at apogee. Other dives will proceed directly from dive to climb as normal. For negative values less than -1, this logic is reversed, and values of $DIVE divisible by $LOITER_N_DIVE will not loiter; all others will loiter. A value of 0 disables loitering. [default 0]
$LOITER_W_DBAND	Deadband (cm/s) around 0.0 for when to adjust buoyancy during loiters. [default 2]

When loitering, the loiter state begins after the glider reaches apogee depth and pumps to neutral buoyancy (as defined by C_VBD). During loiter, the glider monitors pressure to try to maintain zero vertical velocity. If the vertical velocity exceeds the deadband, the glider will pump (if sinking) or bleed (if rising and above $D_NO_BLEED) according to $LOITER_DBDW. If the glider sinks below $LOITER_D_BOTTOM, the glider will always pump (regardless of how slowly it is descending). Likewise, if the glider rises above $LOITER_D_TOP, the glider will bleed (if above $D_NO_BLEED).

Bit 1 of the experimental features parameter $OPTIONS can be set to pump to in situ neutral density rather than $C_VBD (e.g., $OPTIONS,2).

10. exec system

The exec system is the glider's onboard mission scripting mechanism. It uses a decision tree architecture to copy standard control files (cmdfile, science, targets, etc.) into place autonomously. It is generally used when the glider will not be able to reach the surface for extended periods (due to ice for example). There are two key concepts in the system – *set numbers*, referring to a specific set of control files to be executed together, and *conditions*, referring to the specific conditions under which a given *set number* is to be activated. Exec sets can also be triggered by targets (exec=, timeout-exec=, etc.) Sets are all contained in a single file: exec.dat. The list of active conditions is maintained by the glider in a special batch file, conditions.dat.

10.1 exec.dat

exec.dat is the central file for the numbered command sets that are triggered (executed) upon conditional evaluation or via any of the target triggers (exec=, fence-exec=, timeout-exec=, src-timeout-exec=, slow-progress-exec=). It is the gliders onboard basestation – the glider looks to exec.dat for new control files just like it would normally download files from the basestation. The conditions and targets control when those files are present to be "downloaded".

Whitespace matters in this file. Each file to be executed as part of a set begins with the filename (cmdfile, targets, science, scicon, pdoscmds) on a line by itself with no leading whitespace (first column), followed by a series of set numbers delimited by periods. For example, cmdfile.3.11.911 is the cmdfile for sets 3, 11, and 911. Set numbers are arbitrary (except for 911 and 912). You can associate them with a dive (as in a couple of the sets in the example below), but any such association is purely for the pilot's own sense of organization.

The filename is followed with lines that are to become the operating control file when a matching set number is executed for any reason. File contents must start with leading whitespace (must be indented). The file ends with a blank line.

Any time a file is changed (cmdfile.201 -> cmdfile for example), the existing file is copied with a .pre extension. This previous version can be referenced by the include mechanism.

Files can be included in the exec file by reference. #cmdfile.pre for example will copy the contents of cmdfile.pre into the exec.dat file. Rather than copying the same sets of parameters for loiter dives or profiling dives that are common to many different sets you can use #cmdfile.loiter in the exec.dat to reference a file with that common set of parameters. .pre generation is automatic. Other included files must exist on the disk.

There are two hardcoded sequence numbers, 911 and 912. 911 is exec'd on starting escape for any reason if position is known. 912 is executed if position is not known ($FIX_MISSING_TIMEOUT). Including 911 and 912 sets in exec.dat is optional.

10.1.1 exec.dat example

```
/ requires "condition name=dive1 set=1 expression="$DIVE >= 1" " be set at launch
cmdfile.1
 $GO

pdoscmds.1
 condition name=dive3 set=3 expression="$DIVE >= 3"

/ switch to deeper dives after dive 3 (4 onwards)
cmdfile.3
 $D_TGT,600
 $T_MISSION,400
 $T_DIVE,200
 $GO

pdoscmds.3
 condition name=dive11 set=11 expression="$DIVE >= 11"

/ switch to full depth dives at dive 11
cmdfile.11
 $D_TGT,990
 $T_MISSION,720
 $T_DIVE,360
 $MAX_BUOY,120
 $GO

pdoscmds.11
 condition name=dive16 set=16 expression="$DIVE >= 16"

cmdfile.16
 $PITCH_ADJ_GAIN,0
 $ROLL_ADJ_GAIN,0
 $COMPASS_USE,49156
 $C_VBD_AUTO_DELTA,0
 $C_VBD_AUTO_MAX,0
 $C_PITCH_AUTO_DELTA,0
 $C_PITCH_AUTO_MAX,0
 $GO

/ turn off some of the auto stuff and keep deep dives
/ we'll also use these dives to run to A in a 911 escape (w/fix)
cmdfile.20.911
 $PITCH_ADJ_GAIN,0
 $COMPASS_USE,49156
 $C_VBD_AUTO_DELTA,0
 $C_VBD_AUTO_MAX,0
 $T_MISSION,720
 $T_DIVE,360
 $D_TGT,990
 $LOITER_N_DIVE,0
 $GO

/ loiter and hold at A - also the loiter dives in an escape w/o fix
cmdfile.21.912
 $T_LOITER,86400
 $LOITER_N_DIVE,1
 $D_TGT,500
 $T_DIVE,180
 $T_MISSION,360
 $LOITER_D_TOP,400
 $LOITER_D_BOTTOM,700
 $LOITER_DBDW,400
 $LOITER_W_DBAND,2
 $D_NO_BLEED,450
 $GO

targets.21
 1_21 lat=7434.3 lon=-14540.0 radius=-20000 goto=1_22 exec=22

/ return to center if outside watch circle
/ also the condition to hold if we run low on energy,
/ requires condition name=powerSave set=22 expression="Ahr24V > 300"
targets.22
 1_22 lat=7434.3 lon=-14540.0 radius=4000 goto=1_22 exec=21

cmdfile.22
 $T_DIVE,360
 $D_TGT,990
 $T_MISSION,720
 $LOITER_N_DIVE,0
 $GO

/ seasonal stuff

/ requires condition name=summer set=101 expression="time() > epoch(2024,6,1,0,0,0)"
cmdfile.101
 $SURFACE_URGENCY,10
 $SURFACE_URGENCY_TRY,10
 $SURFACE_URGENCY_FORCE,20
 $GO

science.101
 30.0 gc=300.0 seconds=10.0 sensors=111 compass=1 pressure=1
 1050.0 gc=360.0 seconds=30.0 sensors=111 compass=1 pressure=1
 loiter gc=600.0 seconds=120.0 sensors=111 compass=1 pressure=1
```

```
/ requires condition name=winter set=102 expression="time() > epoch(2023,10,1,0,0,0)"
cmdfile.102
 $SURFACE_URGENCY,0
 $SURFACE_URGENCY_TRY,0
 $SURFACE_URGENCY_FORCE,0
 $COMM_SEQ,9
 $UPLOAD_DIVES_MAX,10
 $GO

science.102
 30.0 gc=300.0 seconds=10.0 sensors=100 compass=1 pressure=1
 1050.0 gc=360.0 seconds=30.0 sensors=100 compass=1 pressure=1
 loiter gc=600.0 seconds=120.0 sensors=100 compass=1 pressure=1

/contingencies

/ change to shallow loiters if no fixes for 2 days
/ requires condition name=noFix expression="time() > fixTime + 86400*2" set=201
cmdfile.201
 $T_LOITER,21600
 $LOITER_N_DIVE,1
 $D_TGT,200
 $T_DIVE,70
 $T_MISSION,140
 $LOITER_D_TOP,100
 $LOITER_D_BOTTOM,250
 $LOITER_DBDW,400
 $LOITER_W_DBAND,2
 $GO

pdoscmds.201
 condition name=oneDive set=202 expression="$DIVE > A{%.0f}" init="A{$DIVE}"

/ revert to previous dive state after 1 shallow loiter dive
/ reset the no fixes timeout
cmdfile.202
 #cmdfile.pre

pdoscmds.202
 condition name=noFix expression="time() > fixTime + 86400*2" set=201

/ 912 is escape w/o fix - cmdfile is day long loiters as above
targets.912
 run_912 head=240 lat=0 lon=0 goto=run_912
```

10.2 exec conditions

To set a condition from pdos

condition name=IDENTIFIER set=N expression=CONDITIONAL init=INITIALIZER init=INITIALIZER ...

IDENTIFIER is a text string used to refer to the condition (for clearing generally) and for logging when executed. N is a set number referencing control sets in the exec.dat file

CONDITIONAL is any conditional numeric expression that can reference variables, standard math functions, other functions, and “constants” that are evaluated at the time the condition command is executed to enable baseline or delta values

INITIALIZERs are mathematical expressions that are evaluated at the time the condition is installed (not when the condition is tested and possibly executed) and then substituted back into the CONDITIONAL expression to determine the expression that will be evaluated and tested at each check of the exec system.

Examples:

condition name=August set=101 expression="time() > epoch(2021,8,1,0,0,0)"

will execute set 101 (which might contain science file changes for summer sampling for example) at the first check that occurs after 01-Aug-2021 at midnight.

condition name=oneDive set=20 expression="$DIVE >= A{%.0f}" init="A{$DIVE + 1}"

when this condition statement is run, the init= portion is evaluated to fill in the variable A that is referenced in the expression. Once evaluated to the current dive number + 1, that result is then submitted into the text of the expression using the printf specifier %.0f (no decimal places) at position A in that expression. The resulting expression is then stored with the substitution in place. If $DIVE is 10 when this condition statement is run, the resulting condition that is stored will be

condition name=oneDive set=20 expression="$DIVE >= 11"

*condition name=deepDive set=30 expression="time() > TP2{%.0f}" init="TP2{time() + 86400*2}"*

will execute set 30 two days after the condition statement is run.

*condition name=goneEast set=201 expression="lon > BOUNDARY{%.5f} || time() > TIMEOUT{%.0f}" init="BOUNDARY{lon + 2}" init="TIMEOUT{time() + 86400*20}"*

will execute set 201 after the glider moves two degrees east of its location when the statement is run or after 20 days, whichever comes first.

Conditions are evaluated after the point where the phone call is normally made, before new control files are parsed and calculations for the next dive are completed. See epdos commands above for additional commands (exec, eval, clear) useful in managing exec conditions.

10.2.1 exec conditions reference

Permissible variables in expression= and init= conditional expressions are:

  $PARAM (any parameter, but not arbitrary log results)
  min10V
  min24V
  Ahr10V
  Ahr24V
  fixTime
  lat
  lon
  nocomm
  network
  xxTimeouts (where xx is compass, pressure, sc, pm, ct, etc.)
  humidity
  internalP

Operators:
  standard operators: +,-,*,/,%,()
  binary operators: <<,>>,|,&,~
  logical operators: >,<,&&,||,==,!=,<=,>=,!
  if-then-else operator: a ? b : c

Math functions:
  sin, cos, tan, sinh, cosh, tanh
  pow, exp, log, log10,
  sqrt, hypot, ceil, fmod,
  fabs

Symbolic constants:
  pi

Physical oceanography functions:
  salinity(C, T, P) (psu)
  potentemp(S, T, P, RefP) (degC)
  soundspeed(S, T, P) (m/s)
  density(C, T, P) (kg/m^3)
  potendens(C, T, P, RefP) (kg/m^3)

  C=mMho/cm, T=degC, P=dbar, S=psu

Other functions:
  distance(lat0, lon0, lat1, lon1) (meters)
  time() (RTC time since epoch)
  epoch(yyyy, mm, dd, HH, MM, SS) (converts calendar time to epoch time)

All math is done in double precision, regardless of the source variable type. Variables that are int, long or float on the glider are cast to double before expressions are evaluated and expression evaluations are returned as doubles. All printf specifiers for constants in expressions should be based on %f.