

Starting with Spray code version 0608, the navigation structure is comprised of a waypoint list which a route list then references to specify the desired route. Before deployment, a waypoint file and the initial route file are generated and downloaded to the Spray. These files are then transferred to the UNIX ground station. These files can be modified (or new ones defined) as the desired route changes; the perl script **sbd_route.pl** uses these files to create the correct shore commands to send to Spray.

The waypoint file name is not restricted, but the present protocol is 'NNNN_YY.WPT' where NNNN is the experiment name (i.e. ASAP,HB06, or LINE90), and YY is a revision number for this file. An example file is shown here for the CalCOFI Line 90 experiment (file name **LINE90_001.WPT**):

```
HOME      33.4580 -117.7900  1   Emergency recovery waypoint
Deploy    33.4400 -117.7260  1   desired deploy location
DP_1     33.435  -117.7530  1   head ~1.6 NM offshore
DP_2     33.470  -117.782   1   head ~2.6 NM N, to ~L90
90.28    33.485  -117.768   1   Inshore L90
90.Cat   33.257  -118.260   3   L90 abeam of Catalina
90.Clm   33.100  -118.660   3   L90 abeam of San Clemente
90.100   31.085  -122.662   3   L90 SW (outer) end-point
END
```

The waypoint name is the first 8 characters, followed by the latitude and longitude, in decimal degrees. The fourth column is an optional watch-circle radius [km]. At the end of the line are optional comments.

The route file name is not restricted, but the present protocol has the suffix being '.rte'. The corresponding file to the above waypoint file is file **L90_013_1.rte** where 013 is the Spray serial number, and 1 is the revision number:

```
LINE90_001.WPT      =file with the waypoint list.
   6  1  2  1      =#pts in route, next pt, end-action, direction
DP_1      0 1 0    =waypt label, arrival detect-mode, range, approach_angle
DP_2      1 0 0    gets us N up to L90
90.28     1 0 0    inshore point on line 90
90.Cat    1 0 0    L90 abeam Catalina
90.Clm    1 0 0    L90 abeam San Clemente
90.100    1 0 0    SW endpt of L90
```

The first line has the waypoint file name to reference, with an optional comment.

The second line has four parameters: **NR NEW END DIR** <comment>

1. **NR** = The number of waypoints in the route list (max value=19).
2. **NEW** = The route entry index to presently head for.
3. **END** = The action to take when the end-of-route is reached: 0=go HOME; 1=repeat the route; 2=reverse the route; 3=stay stationed at the end-of-route.
4. **DIR** = The direction to proceed through the route list (1=forward, -1=reverse).

The second line is then proceeded by **NR** lines of information for each route entry. Each line has up to five entries:

1. Eight-character name (must match a name in the waypoint list file, but is case-insensitive).
2. Arrival-detection mode: 0=range only, 1=range+finish line, 2=finish line only.
3. Watch circle radius[km]: 0=auto-range (based on depth of last-dive).
4. Approach Angle to the waypoint (is used to define the finish-line): 0 = auto-compute (approach angle = angle between Spray and the waypoint at the time the Spray switches to using that waypoint).
5. Any desired comment to help explain the route.

Downloading the Route to the Spray, using the UNIX ground station

`/home/gS/waypt` = directory with the waypt and route files plus the perl script.

`sbd_route.pl`, takes the route file and constructs the output message for the Spray (the main bulk of the code is in `sbd_route.c`).

FIRST-TIME SET-UP ON YOUR SYSTEM:

1. Modify `sbd_route.c` for the appropriate machine (UNIX or PC) plus directory structure names. A directory is required for the waypoint and route files, for the queue file for the SBD email message, and for the configuration `.cfg` file.
2. Modify the parameter 'CFG', dependent upon whether the `.cfg` files are regularly updated (thus we know the present configuration of Spray's waypoint list: if so, set `CFG=1`), or to force the program to always download a waypoint list to match the desired route (set `CFG=0`).
3. Modify `HAND_EDIT` to 0 to not allow hand-editing, else 1 to allow it (more flexible, but easier for the person to get into trouble).
4. Compile `sbd_route.c`.
5. Verify that the `sbd_route.pl` has the right directory structure for your machine.

Type in `./sbd_route.pl` to execute the perl script.

It will query for the Spray serial number and then execute the code from `sbd_route.c`. It will ask for the waypoint file name, and then query for any user changes. It is advisable that the file be edited correctly for the desired settings first, but here is the list of what the user can edit (if `sbd_route.c` was compiled with `HAND_EDIT = 1`):

1. The new waypoint to head for (see definition for *NEW* above).
2. *END* =End-of-Route Action (what to do when the last waypoint is reached). This will provide you with a list of options.
 - i. 0 = go to the HOME waypoint.
 - ii. 1 = repeat the route from the beginning.
 - iii. 2 = reverse the route (retrace the route in the opposite direction).
 - iv. 3 = stay stationed at the last waypoint.
 - v. 4 = ABORT when the waypoint is reached (not recommended).

As one example for options 1 and 2, if there are three waypoints in the route, repeating the route (option 1) will result in the pattern 1-2-3 -1-2 -3 (etc., i.e. doing a box pattern). Reversing the route (option 2) will result in the pattern 1-2-3-2-1-2-3 (etc., i.e. tracing the same route in the opposite direction).
3. Change the 'route direction' *DIR*, where the choices are 1=Forward, -1 = Reverse. This refers to the present direction that we're going through the route. If the next entry is #2, and the route direction is forward, after reaching #2, we'll head for entry #3. If the route direction is reverse, after reaching #2, we'll head for entry #1.
4. Change any or all of the watch-circle radii. These should already be set in the route file, but the user can edit here if desired. NOTE, the radius needs to be entered as an integer value, units=km; radius=0 = auto-ranging (based on depth of the last dive).
5. Change any or all of the waypoint arrival-detection values. This determines what parameters the Spray will use to determine if we're at the waypoint, and should already have been set in the route file.

- i. *Detect* = 0 : arrival is based on range only. The Spray has only arrived at the waypoint if it is within the watch-circle.
 - ii. *Detect* = 1 : arrival is based on range OR the 'finish line.' If the Spray is within the watch-circle radius OR has 'crossed the finish line' then the Spray has arrived at the waypoint. The finish line is perpendicular to the approach angle, intersecting the waypoint that we're headed for.
 - iii. *Detect* = 2 : arrival is just based on crossing the finish line.
6. The user is given the option to repeat all of the above.
7. The code then formulates the output command sequence to send the new route settings, and prints this out to the screen.
8. The user is asked if this command list should be written to the sbd queue file. If YES, the commands are written to a file, which is then attached to an outgoing email to the Spray. If NO, commands are NOT sent and no action is taken.

Route and Waypoint Structures as defined in SprayWaypoint List Structure (up to 19 waypoints, excluding the HOME setting):

waypt[*iw*].*lat* = latitude (decimal degrees) for index *iw* .
waypt[*iw*].*lon* = longitude (decimal degrees).
waypt[*iw*].*valid* = flag to indicate that this waypoint has been set.

Basic Route Structure:

nr = # of waypoints in the route (up to 19 entries allowed).
now = route entry index that we're now heading for ($1 \leq now \leq nr$).
dir = direction to head through the route.
 1 = forward : increment the route index *ir*,
 -1 = reverse: decrement the route index.
end = action to take when a route end-point (start or end) is reached.
 Endpoints are: start-of-route: *now* = 1 if *dir* < 0
 end-of-route: *now* = *nr* if *dir* > 0
 0 = Go to the HOME waypoint (*waypt*[0]).
 1 = step through sequentially in the direction '*dir*'.
 if *dir* > 0 and *now* = *nr*, the next waypoint will be = 1.
 if *dir* < 0 and *now* = 1, the next waypoint will be = *nr* ..
 end = 1 will result in a 'box' survey.
 2 = re-trace the route whenever the start or end is reached.
 if *now* = 1 and *dir* < 0 (reached the start-of-route),
 set *now* = 2, *dir* = 1 (proceed forward thru the route).
 if *now* = *nr* and *dir* > 0 (reached the end-of-route),
 set *now* = *nr* - 1, *dir* = -1. (do the route in reverse).
 3 = stay stationed when a route end-point is reached (start or end):
 if *dir* < 0, then will stay stationed at waypoint 1.
 if *dir* > 0, then will stay stationed at waypoint *nr*.

Route List Structure: Four arrays define each route entry (index range= [1..nr]) .

rlist[] = list of *nr* waypoint indices, of the desired order for the route.
 example: if *rlist* = [3 4 2 1], *now* = 2, then *rlist*[*now*] = 4 = *iw* .
detect[*now*] = detect-mode to use to decide if we're at the waypoint:
 0 = range only (use only a watch circle of radius *circle*).
 1 = range or 'finish line'.
 2 = 'finish line' only.
circle[] = watch circle radius [km] to determine if we're at the waypoint.
 circle[*now*] = 0 = auto-scaling, based on last dive depth.
blist[] = approach-bearing [degrees TRUE] to the next waypoint; this is used to
 define the finish line angle. If 0, then it automatically computed.

Examples of stepping through a route = *rlist* = [1 2 3], *nr* = 3, *now* = 1:

end = 1, *dir* = 1 : 1-2-3- 1-2-3 etc.
end = 1, *dir* = -1: 1-3-2- 1-3-2 etc.
end = 2: 1-2-3- 2-1 -2-3 etc. *dir* is toggled when an end-point is reached.
 sign of *dir* ++ -- ++ etc. (toggles at start-of-route and end-of-route).