Protocol

Water Linked DVL protocol

This document describes the Water Linked DVL protocols (serial and ethernet).

Terminology

- DVL - Doppler Velocity Log - Hydro-acoustic unit which uses acoustic beams to measure distance to bottom surface and the velocity which the unit is moving across the surface.
- ACK - Acknowledgement. The command issued was successful.
- NAK - Negative acknowledgement. The command issued failed.

Version

This document describes protocol version 2.0.x (major.minor.patch)

The protocol versioning follows semantic versioning in that:

- MAJOR version increments represent incompatible API changes
- MINOR version increments represent added functionality in a backwards-compatible manner
- PATCH version increments represent backwards-compatible bug fixes

Serial Protocol

Overview

The serial communication format is 115200 8-N-1 (no hardware flow control).

Packets sent to and received from the DVL start with a `w` and end with LF or CR+LF. The packet format is:

Start byte	Direction	Command	Options (0 to many)	Checksum	End byte

`w`	`c` or `r`	`x`	`,`
			`[option]`
				`*xx`	`\n` or
					`\r\n`

Direction is command (`c`) for commands issued to the DVL and the DVL replies with direction set to response (`r`). The commands can be sent as a string or entered one char at a time from a terminal.

The protocol can support Water Linked DVLs with different feature sets. To support any Water Linked DVL the connection procedure is to:

- Get protocol version. Verify that the major version number is 2.
- Get product detail. Verify product type is dvl.

> **Note**
>
> Checksum is optional when sending commands to the DVL. The DVL always returns a checksum. The checksum algorithm is CRC-8 and it is formatted as a hexadecimal number using 2 lower-case charaters (ex: `*c3`).

Commands

Commands in the table are shown **without** the checksum for readability.

Command	Description	Response	Description
`wcv`	Get protocol version	`wrv,` *[major],[minor],[patch]*	Protocol version. eg: `wrv,2.1.0`
`wcw`	Get product detail	`wrw,` *[name],[version],[chipID],[IP address]*	Where type is dvl, name is product name, version is software version, chip ID is the chip ID and *optionally* the IP address if connected to DHCP server: eg: `wrw,dvl-a50,1.4.0,0xfedcba98765432` or `wrw,dvl-a50,1.4.0,0xfedcba98765432,10.11.12.140`
		`wrx,` *[details below]*	Velocities measured. See details below
		`wr?`	Malformed request: Response when packet cannot be understood
		`wr!`	Malformed request: Packet does not match the given checksum

Velocity report

Velocity report is outputted after each measurement has been completed. The expected update rate varies depending on the altitude and will be in the range is from 2-26 Hz. The X, Y, Z axis are oriented according to the marking on the DVL.

The velocities measured response is on the following format: `wrx,` *[time],[vx],[vy],[vz],[fom],[altitude],[valid],[status]*

Variable	Description
time	Milliseconds since last velocity report (ms)
vx	Measured velocity in x direction (m/s)
vy	Measured velocity in y direction (m/s)
vz	Measured velocity in z direction (m/s)
fom	Figure of merit, a measure of the accuracy of the measured velocities (m/s)
altitude	Measured altitude to the bottom (m)
valid	If valid is "y" the DVL has lock on the bottom and the altitude and velocities are valid (y/n)
status	0 for normal operation, 1 for high temperature warning

Example where velocities are valid:

```
wrx,112.83,0.007,0.017,0.006,0.000,0.93,y,0*d2
wrx,140.43,0.008,0.021,0.012,0.000,0.92,y,0*b7
wrx,118.47,0.009,0.020,0.013,0.000,0.92,y,0*54
```

Example where velocity and altitude is not valid and high temperature warning is given:

```
wrx,1075.51,0.000,0.000,0.000,2.707,-1.00,n,1*04
wrx,1249.29,0.000,0.000,0.000,2.707,-1.00,n,1*6a
wrx,1164.94,0.000,0.000,0.000,2.707,-1.00,n,1*39
```

Transducer report

Transducer report is outputted after each measurement has been completed. The expected update rate varies depending on the altitude and will be in the range is from 2-26 Hz.

The distances measured form each transducer is on the following format: `wrt,` *[dist_1],[dist_2],[dist_3],[dist_4]*

Variable	Description
dist_1	Measured distance to bottom from trancduser 1 (m)
dist_2	Measured distance to bottom from trancduser 2 (m)
dist_3	Measured distance to bottom from trancduser 3 (m)
dist_4	Measured distance to bottom from trancduser 4 (m)

Example where velocities are valid:

```
wrt,15.00,15.20,14.90,14.20*b1
wrt,14.90,15.10,14.80,14.10*ac
```

Example where distance is not valid on transducer 4:

```
wrt,14.90,15.10,14.80,-1.00*53
wrt,15.00,15.20,14.90,-1.00*71
```

Checksum

The checksum algorithm is CRC-8 (Polynomal: 0x07, Init: 0x00, RefIn/RefOut: false, XorOut: 0x00, Check: 0xf4). Checksum is formatted as a hexadecimal number using 2 lower-case charaters (ex: `*c3`).

Compatible implementations:

- Python 3: crcmod `crcmod.predefined.mkPredefinedCrcFun("crc-8")`
- Golang: github.com/sigurn/crc8 `crc8.MakeTable(crc8.CRC8)`

Example for how to verify checksum using Python 3 and crcmod:

```
crc = crcmod.predefined.mkPredefinedCrcFun("crc-8")
sentence = b"wrx,1164.94,0.000,0.000,0.000,2.707,-1.00,n,1*39"
data, checksum = sentence.split(b"*")

if crc(data) == int(checksum, 16):
    print("CRC valid")
else:
    print("CRC invalid")
```

Ethernet protocol (TCP)

Overview

The DVL supports sending velocity updates using the Transmission Control Protocol (TCP). The DVL runs a TCP server on port 16171.

Each packet sent contains a velocity report from the DVL on JSON format.

Velocity report

Velocity report is outputted after each measurement has been completed. The expected update rate varies depending on the altitude. The X, Y, Z axis are oriented according to the DVL. The messages are delimited by newline.

Variable	Description
time	Milliseconds since last velocity report (ms)
vx	Measured velocity in x direction (m/s)
vy	Measured velocity in y direction (m/s)
vz	Measured velocity in z direction (m/s)
fom	Figure of merit, a measure of the accuracy of the measured velocities (m/s)
altitude	Measured altitude to the bottom (m)
velocity_valid	If valid is true the DVL has lock on the bottom and the altitude and velocities are valid (true/false)
status	Reports if there are any issues with the DVL. 0 means no errors
format	Format type and version for the velocity report
transducers	Is a list containing information from each transducer: [id, velocity, distance, rssi, nsd, beam_valid]

Example of TCP report. (indented for readability)

```
{
  "time": 170.52674865722656,
  "vx": -0.00563613697886467,
  "vy": -0.007631152402609587,
  "vz": -0.007641898933798075,
  "fom": 0.001959984190762043,
  "altitude": 0.6173566579818726,
  "transducers": [
    {
      "id": 0,
      "velocity": -0.007625679485499859,
      "distance": 0.6769760251045227,
      "rssi": 38.66838836669922,
      "nsd": 18.295578002929688,
      "beam_valid": true
    },
    {
      "id": 1,
      "velocity": -0.0034413286484777927,
      "distance": 0.6769760251045227,
      "rssi": 35.403541564941406,
      "nsd": 19.518909454345703,
      "beam_valid": true
    },
    {
      "id": 2,
      "velocity": -0.006717036943882704,
      "distance": 0.6653040051460266,
      "rssi": 41.03888702392578,
      "nsd": 20.25017738342285,
      "beam_valid": true
    },
    {
      "id": 3,
      "velocity": -0.01045388076454401,
      "distance": 0.6536320447921753,
      "rssi": 31.09071922302246,
      "nsd": 17.366933822631836,
      "beam_valid": true
    }
  ],
  "velocity_valid": true,
  "status": 0,
  "format": "json_v1"
}
```