Kingfisher USV NMEA Interface

Revision History

Version	*Date*	*Description*	Author
0.0	April 29, 2013	Initial Draft	Alon Yaari
0.1	July 4, 2013	• Filled out TODO text sections. • Added reference frame section. • Removed messages not implemented for July 2013 delivery. • Applied CPR template. • Clarify missing checksum format.	Mike Purvis

Contents

1.0 Background

Previous generation Kingfisher craft supported an in-house serial protocol called Clearpath Communications Protocol. By building an interface to CCP, it was possible to have MOOS (and other software stacks) completely replace Clearpath's in-house ROS-based software. However, this was non-ideal for several reasons:

- Limited ability to share code and functionality between MOOS and ROS.
- Alternative systems required their own set of hardware peripheral drivers.

The new generation of Kingfisher has exacerbated both of these issues, as Kingfisher now includes much more core autonomy functionality out of the box (implemented as ROS nodes), and also includes a more extensive set of basic peripherals—with even more in the planning stages.

It may have been possible to use a direct MOOS-ROS bridge, which would allow the two systems to coexist and interoperate. However the intention was to create a separation such that a user of either system would not have to have any significant training or knowledge on the operation of the other. The interface should be defined in such a way that it is independent of implementation details on either side. For example, a decision by Clearpath to migrate ROS topics between namespaces should not then necessitate recompilation of MOOS binaries or editing of mission files.

MOOS has existing support for a set of NMEA sentences which are used to control the Bluefin UUV. MIT and Clearpath Robotics have collaborated to extend and modify that set of sentences for use with Kingfisher. This limits the new development work required to support this interface from the MOOS side.

2.0 System Definitions

This section defines key concepts and terms for the remainder of the protocol document.

2.1 Vehicle

Where this document refers to the "vehicle" it means the computer system that interfaces with payload computing. In other words, payloads or external users view the "vehicle" as a black box entity that manages the vehicle's systems and control surfaces and communicates through a full-duplex streaming computer interface.

2.2 Payload

Where this document refers to the "payload" it means the computer system that interfaces with the vehicle. The "payload" could be a system running on the same physical hardware as the vehicle, a separate computing unit plugged into the vehicle through a payload port, or a remotely-located system communicating or a network. In other words, the “payload” is a black box entity which requests vehicle status and attempts to send control commands to the vehicle through a full-duplex streaming computer interface.

2.3 Interface

This section defines the specifics of the interface through which the payload and vehicle communicate.

- The defined interface assumes a full-duplex data stream, which could include but is not limited to a serial port, network socket, or pseudo-tty.
- As of the current revision, the only transport offered is a TCP socket between payload and vehicle.
- The vehicle provides a TCP server listening on port 29500
- The vehicle supports multiple clients that can connect and disconnect at any time.
- As of the current revision, any client may send any command to the vehicle. Future revisions may include role definitions that restrict clients and commands.

2.4 Sentences

Sentences are the atomic units of communication to be used in transmitting information over the interface.

- Each message sent TO or FROM the vehicle is in the form of a sentence.
- Sentences follow the NMEA standard for sentences.
- Sentences sent FROM the vehicle will always include a checksum.
- Sentences sent TO the vehicle may or may not include a checksum.
- All characters in the sentence must be printable ASCII characters between `0x20` and `0x07F`, inclusive
- Comma '`,`' (`0x2C`) may only be included as a delimiter between defined fields, never as data content
- Asterisk '`*`' (`0x2A`) may only be included as the final character after the last data element, never as data content
- Message types and content are case sensitive (e.g., `AAAAA` is not the same as `aaaaa`)
- Nodata can be indicated with a blank field between two commas or comma and asterisk (e.g., `$ABCDEF,,*HH`)

2.5 Sentence Format

Following is an example of expected format for NMEA sentences in the protocol.

```
tens position:      0000000000111111111122222222  2  2
ones position:      0123456789012345678901234567  8  9
example NMEA:       $ABCDEF,data1,data2,dataN*76 cr lf
```

Position	Description
00	Sentences always begin with '`$`' (`0x24`)
01 to 06	The second through sixth characters, inclusive, consist of five upper-case letters `A` (`0x41`) to Z (`0x5A`) that define the message type
07	The seventh character is always a comma
08 to 24	The body of the sentence is comma-separated data elements.
25	The last data element is always followed by a '`*`' (`0x2A`) and not a comma
26 to 27	On messages from the vehicle, the two characters following the '`*`' are always the checksum, in capitalized hexadecimal. On messages to the vehicle, these may be omitted.
28 to 29	The last two characters are always a carriage return (`0x0D`) followed by a linefeed (`0x0A`)

On messages to the vehicle, checksum characters are optional. When checksum is omitted, the carriage return and linefeed must immediately follow the * character.

2.6 Sentence Checksum

- XOR of all characters between the '`$`' and the '`*`', non-inclusive.
- Value to be presented as two-digit capitalized hexadecimal value.
- See appendix for sample checksum generation code.

2.7 Timekeeping

- Every sentence sent from the vehicle has a timestamp as the first field. Time is “GPS time”.
- Timestamp is accurate to the millisecond of when the message was created (does not take network latency into account).
- Sentences may contain additional time stamps relevant to data encoded in the message.

2.8 Reference Frames

- Messages which report latitude and longitude use the WGS84 reference coordinate system.
- Messages which report vehicle acceleration and depths use the sea vehicle convention[1], which is:
    - X-axis is positive toward the bow.
    - Y-axis is positive toward the starboard deck.
    - Z-axis is positive downward.
- Similarly, reports of vehicle attitude and motion use the right-hand rule about those axes:
    - Roll is positive when the port deck is elevated (or rising).
    - Pitch is positive when the bow is elevated (or rising).
    - Heading and yaw are positive as the vehicle rotates clockwise from north (zero).

[1] http://en.wikipedia.org/wiki/File:RPY_angles_of_ships.png

3.0 Messages FROM vehicle

Messages sent FROM the vehicle have codes that start with "`CP`"

Navigation Update

- Current pose and position as estimated by a filter running on the vehicle.
- Takes GPS, digital compass, and possibly IMU into account

`$CPNVG,1,2,3,4,5,6,7,8,9,10,11,12*HH<cr><lf>`

1	[Timestamp]	Timestamp of the sentence
2	[Lat_NMEA]	Calculated latitude, in NMEA format
3	[LatNS_NMEA]	Hemisphere (N or S) of latitude
4	[Lon_NMEA]	Calculated longitude, in NMEA format
5	[LonEW_NMEA]	Hemisphere (E or W) of longitude
6	[PosQual]	Quality of position estimate (no GPS = 0, otherwise = 1)
7	[AltBottom]	Altitude in meters from bottom, for surface vehicles this field is blank
8	[DepthTop]	Depth in meters from top, for surface vehicles this field is blank
9	[Heading]	Direction of travel in degrees clockwise from true north
10	[Roll]	Degrees of roll
11	[Pitch]	Degrees of pitch
12	[NavTimestamp]	Timestamp for time this pose/position was calculated. If blank, use [Timestamp].

Example: `$BFNVG,081025.987,4221.81092,N,07106.84603,W,1,,,203.1,-3.4,4.5,081025.980*5B`

Velocity and Rate Update

- Current velocity and change in pose as estimated by a filter running on the vehicle

`$CPNVR,1,2,3,4,5,6,7*HH<cr><lf>`

1	[Timestamp]	Timestamp of the sentence
2	[Vel_East]	East component of vehicle transit velocity
3	[Vel_North]	North component of vehicle transit velocity
4	[Vel_Down]	Vertical component of vehicle transit velocity
5	[Rate_Pitch]	Deg/s of pitch rate
6	[Rate_Roll]	Deg/s of roll rate
7	[Rate_Yaw]	Deg/s of yaw rate

Raw Compass Data

- Data is read from the compass sensor by the vehicle and repackaged into this sentence

`$CPRCM,1,2,3,4,5,6*HH<cr><lf>`

1	[Timestamp]	Timestamp of the sentence
2	[ID_Compass]	Unique ID number of the compass being reported on
3	[Heading]	Raw reading from compass for degrees clockwise from true north
4	[Pitch]	Raw reading from compass for degrees of pitch
5	[Roll]	Raw reading from compass for degrees of roll
6	[NavTimestamp]	Timestamp for time compass reported this data. If blank, use [Timestamp]

Battery Voltage

- One sentence is published for every bank of cells

`$CPRBS,1,2,3,4,5,6*HH<cr><lf>`

1	[Timestamp]	Timestamp of the sentence.
2	[ID_Battery]	Unique ID number of the battery being reported on.
3	[V_Batt_Stack]	Voltage of the battery bank.
4	[V_Batt_Min]	Lowest voltage read from cells in the bank.
5	[V_Batt_Max]	Highest voltage read from cells in the bank.
6	[TemperatureC]	Temperature in Celsius.

Raw IMU Data

- Data is read from the IMU by the vehicle and repackaged into this sentence

```
$CPIMU,1,2,3,4,5,6,7,8*HH<cr><lf>
```

1	[Timestamp]	Timestamp of the sentence.
2	[AngularRate_X]	Angular rate about the X axis in deg/s, right-hand rule
3	[AngularRate_Y]	Angular rate about the Y axis in deg/s, right-hand rule
4	[AngularRate_Z]	Angular rate about the Z axis in deg/s, right-hand rule
5	[Accel_X]	Acceleration along the X axis in m/s^2, forward positive
6	[Accel_Y]	Acceleration along the X axis in m/s^2, forward positive
7	[Accel_Y]	Acceleration along the X axis in m/s^2, forward positive
8	[NavTimestamp]	Timestamp for time compass reported this data. If blank, use [Timestamp]

Raw GPS NMEA Sentences

```
$GPRMC
$GPGGA
```

Others may be included, depending on configuration of GPS receiver.

4.0 Messages TO vehicle

Messages sent TO the vehicle have codes that start with "`PY`"

Command Desired Thrust

- Unique to Clearpath
- Payload manages a PID to request a desired throttle percentage and rudder angle.
- Vehicle adjusts thrust to percent of throttle operating range
- Vehicle attempts actual/equivalent rudder angle, caps angle at max rudder angle.

`$PYDEP,1,2*HH<cr><lf>`

1	[DesYawRate]	Desired rate of yaw, in radians/sec.
2	[DesThrustPct]	Desired percent of thrust, -100 to 100. Stopped = 0, positive thrust = forward motion.

Command Vehicle Helm

- Unique to Clearpath
- Payload commands a desired heading and speed
- Vehicle PID controls vehicle turn rate to achieve desired heading relative to true north
- Vehicle PID controls thrust to maintain desired speed

`$PYDEV,1,2*HH<cr><lf>`

1	[DesHeading]	Desired heading relative to true north, 0 to 359.
2	[DesSpeed]	Desired speed over ground, 0.0 and positive real numbers.

Command Vehicle Motors

- Unique to Clearpath differential-drive vehicle.
- Payload commands percent thrust for each motor.
- Vehicle adjusts thrust to percent of each motor's throttle operating range.

`$PYDIR,1,2*H<cr><lf>`

1	[DesThrustPct_L]	Desired percent of thrust for the portside motor, -100 to 100. Stopped = 0, positive thrust = forward motion.
2	[DesThrustPct_R]	Desired percent of thrust for the starboard motor, -100 to 100. Stopped = 0, positive thrust = forward motion.

APPENDIX A: Unimplemented Bluefin Sentences

The Bluefin Standard Payload Interface includes many sentences that are not applicable to the Clearpath wire protocol (as of this Rev A writing). For completeness, they are described here.

Unimplemented Bluefin Sentences FROM the Vehicle

$BFMSC Payload Mission Command

Sent from the vehicle when executing vehicle-level user-defined behaviors.

$BFBDL Begin Data Logging

Sent from the vehicle when begining the lean-in trackline.

$BFSDL Stop Data Logging

Sent from the vehicle after the lead-out trackline has completed

$BFTOP Topside Message

Delivery of a sentence sent from the topside.

$BFDVT Begin/End DVL External Triggering

If a DVL unit is onboard, this sentence indicates the payload should start or stop external DVL triggering.

$BFTEL Telemetry Status

Sent from the vehicle to provide acoustic modem status.

$BFSVS Sound Velocity

If a sound velocity sensor is onboard, this sentence provides raw output from the sensor.

$BFRDP Raw Depth Sensor Data

If a depth sensor is onboard, this sentence provides raw output from the sensor.

$BFRVL Raw Vehicle Speed

If a tailcone/water speed sensor is onboard, this sentence provides raw output from the sensor.

$BFMBS Begin New Behavior

Sent from the vehicle when a new behavior initiates on the vehicle-side controller.

$BFMBE End New Behavior

Sent from the vehicle when a new behavior completes on the vehicle-side controller.

$BFMIS Mission Status

Sent from the vehicle when a mission starts or stops.

$BFERC Elevator and Rudder Data

On vehicles with a tailcone, reports the currently commanded and actual measurements of elevator and rudder position.

$BFDVL Raw DVL Data

If a DVL sensor is onboard, this sentence provides raw output from the sensor.

$BFCTD Raw CTD Sensor Data

If a CTD sensor is onboard, this sentence provides raw output from the sensor.

$BFRNV Relative Navigation Position

Only used by a specific Bluefin vehicle, not applicable to Clearpath.

$BFPIT Pitch Servo Positions

Only used by a specific Bluefin vehicle, not applicable to Clearpath.

Unimplemented Bluefin Sentences TO the vehcile

$BPTOP Request to Send Data Topside

Not implemented by Bluefin or anyone else.

$BPDVR Request to Change DVL Triggering Method

Only used when a DVL is present onboard.

APPENDIX B: Data Types

Uppercase denotes fixed-width (e.g., 'HH' checksum), lowercase denotes variable width (e.g., .mmm in LatNMEA).

[Acceleration] D.opt

Acceleration in the specified direction, m/s^2

`D`	Integer whole number representing meters per second squared in the stated direction

[AltBottom] dD.opt

Altitude in meters above the bottom of the water body. Always blank for surface vehicles.

`dD`	Whole integer number of meters from the bottom (value is 0 when value between -1 and 1, non-inclusive)
`.opt`	Optional variable-precision fraction of the value.

[AngularRate] sD.opt

`s`	Optional sign character, '`+`' (`0x2B`) or not included for positive, '`-`' (`0x2D`) for negative values
`D`	Integer whole number representing degrees per second of body rotation (includes 0 if value between -1.0 and 1.0, non-inclusive).
`.opt`	Optional variable-precision fraction of the value.

Rate of rotation for the stated body degree of freedom, in degrees per second. For [Rate_Pitch], positive values when nose is rising. For [Rate_Roll], positive values when starboard side is rising. For [Rate_Yaw], positive values when vehicle is rotating clockwise.

[Checksum] HH

`HH`	Hexadecimal digits `00` through `FF` (includes leading `0` if less than `0xA`)

See [HH] for details.

[DepthTop] dD.opt

`dD`	Whole integer number of meters below the water surface (`0` digit is present when value between -1 and 1, non-inclusive)
`.opt`	Optional variable-precision fraction of the value.

Depth in meters below the water's surface. Always blank for surface vehicles.

[DesHeading] dD

`dD`	Positive whole integer number of degrees, at least one digit and no leading zeros

Zero degrees is true north. Degrees increase clockwise from true north.

[DesSpeed] dD.opt

`dD`	Positive whole integer number representing speed (`0` digit is present when value less than 1.0).
`.opt`	Optional variable-precision fraction of the value.

Minimum speed is 0.0. Desired speed can be any number but will (obviously) be capped by the maximum speed the vehicle is capable of achieving.

[DesRudderAngle] sdD

`s`	Optional sign character, '`+`' (`0x2B`) or not included for positive, '`-`' (`0x2D`) for negative values
`dD`	Whole integer representing desired rudder angle -90 to 90 inclusive (at least one digit, no leading zeros)

For a differential drive, rudder angle is a theoretical equivalent. Rudder of 0 is centered, meaning vehicle should travel with no change in yaw. Positive rudder means rudder turns toward starboard, negative rudder turns the rudder toward port. Vehicle will not transit unless desired thrust is non-zero.

[DesThrustPct] sdD

`s`	Optional sign character, '+' (0x2B) or not included for positive, '-' (0x2D) for negative values
`dD`	Whole integer representing desired rudder angle (at least one digit, no leading zeros)

Desired percent of thrust. Thrust of 0 is stopped, regardless of rudder angle.

[Heading] dD.opt

`dD`	Whole integer number of degrees clockwise from true north (value is 0 when value less than 1.0)
`.opt`	Optional variable-precision fraction of the value.

[HH] HH

`HH`	Hexadecimal digits 00 through FF (includes leading 0 if less than 0xA)

The standard NMEA checksum is an 8-bit result of XOR'ing all the characters between '$' and '*' non-inclusive. When represented in hexadecimal, the high-order digit is shown on the left and the low-order on the right.

[ID] D

`D`	Whole integer ID number. Must have at least one digit

ID number is unique within the domain (e.g., non-repeating ID number for each digital compass starting with 1, another set of non-repeating numbers starting with 1 for GPSes).

[ID_Battery]

See [ID] for details. Unique set of integer IDs for onboard batteries, consecutively numbered starting at 1.

[ID_Compass]

See [ID] for details. Unique set of integer IDs for onboard compasses, consecutively numbered starting at 1.

[Lat_NMEA] Type Name:

`DD`	Integer whole number of degrees 00 to 90 (includes leading 0 if less than 10)
`MM`	Integer whole number of minutes (includes leading 0 if less than 10)
`Fff`	Fractional part of the minutes, given with a variable number of digits for the precision

Latitude, in the NMEA standard publishing format.Always 4 digits, then a decimal point, then a variable number of digits, depending on available precision (minimum of one digit to the right of the decimal point). Hemisphere of latitude is provided in the separate field LatNS_NMEA.

[LatNS_NMEA] C

`C`	Either 'N' (0x4E) or 'S' (0x53)

Hemisphere for the Lat_NMEA value.'N' for northern denotes positive values, 'S' for southern denotes negative values.

[LogCode] CCC

`CCC`	Three-letter code for the message identifier for the message name sent from the vehicle.

Message can be "ALL". The thee-letter code is created by removiong the "CP" of the five-letter message ID.

[Lon_NMEA] DDDMM.Fff

`DDD`	Integer whole number of degrees 00 to 180 (includes leading 0 if less than 10)
`MM`	Integer whole number of minutes (includes leading 0 if less than 10)
`Fff`	Fractional part of the minutes, given with a variable number of digits for the precision

Longitude, in the NMEA standard publishing format.Always 4 digits, then a decimal point, then a variable number of digits, depending on available precision (minimum of one digit to the right of the decimal point). Hemisphere of longitude is provided in the separate field LonEW_NMEA.

[LonEW_NMEA] C

C	Either 'E' (0x45)or 'W' (0x57)

Hemisphere for the Lon_NMEA value. 'E' for eastern notes positive values, 'W' for western denotes negative values.

[NavTimestamp]Identical to [Timestamp]

Timestamp of the computed navigation provided in the same sentence as this value. If blank, the [Timestamp] field of the message can be used instead.

[OnOff] cCC

cCC	Either "ON" or "OFF"

This code must be in all uppercase.

[OpenString] cC

cC	String of any length (minimum 1 character)

String may include almost any letter, number, and most punctuation between 0x21 and 0x7F, inclusive. String may NOT include comma (0x2C), space (0x20), or asterisk (0x2A).

[Pitch] sdD.opt

s	Optional sign character, '+' (0x2B) or not included for positive, '-' (0x2D) for negative values
dD	Integer whole number of degrees of pitch -90 to 90 (includes 0 if value between -1.0 and 1.0, non-inclusive)
.opt	Optional variable-precision fraction of the value.

Degrees of pitch reported for the vehicle body. Positive numbers indicate nose above the horizontal.

[QualPos] C

C	Either '0' (0x30) or '1' (0x31)

Quality of position estimate. If position solution includes the GPS, value is '1', otherwise '0'.

[Rate_Pitch]

[Rate_Roll]

[Rate_Yaw]

See definition for [AngularRate].

[Roll] sdD.opt

s	Optional sign character, '+' (0x2B) or not included for positive, '-' (0x2D) for negative values
dD	Integer whole number of degrees of roll -90 to 90 (includes 0 if value between -1.0 and 1.0, non-inclusive).
.opt	Optional variable-precision fraction of the value.

Degrees of roll reported for the vehicle body. Positive numbers indicate starboard side above the horizontal.

[StatusFlag] D

D	"1" or "0"

Single-digit status flag. "0" indicates failed; "1" indicates everything is OK.

[Vel_Down]

See definition for [Velocity].

[Vel_East]

See definition for [Velocity].

[Vel_North]

See definition for [Velocity].

Type Name: [Velocity] sD.opt

`s`	Optional sign character, '`+`' (`0x2B`) or not included for positive, '`-`' (`0x2D`) for negative values
`D`	Integer whole number representing meters per second of travel (includes `0` if value between -1.0 and 1.0, non-inclusive).
`.opt`	Optional variable-precision fraction of the value.

Speed in the stated compass direction of travel, in meters per second. Negative value indicates travel away from the stated compass direction. Velocity is stated as being for East [Vel_East], North [Vel_North], or Down[Vel_Down]. For example, [Vel_East] = -2.5 indicates 2.5 meters per second toward the west.

[Voltage] D.opt

`D`	Integer whole number representing voltage.
`.opt`	Optional variable-precision fraction of the value.

Voltage of the specified item.

[V_Batt_Min]

See [Voltage] for details. Minimum cell voltage in the specified set.

[V_Batt_Max]

See [Voltage] for details. Maximum cell voltage in the specified set.

[V_Batt_Stack]

See [Voltage] for details. Describes voltage measurement of a single set of batteries.

[Timestamp] HHMMSS.FF

`HH`	Integer whole number of the hour `00` to `23` (includes leading `0` if less than 10)
`MM`	Integer whole number of the minutes `00`to `59` (includes leading `0` if less than 10)
`SS`	Integer whole number of the seconds `00`to `59`(includes leading `0`if less than 10)
`FF`	Fractional part of the seconds, always two digits to show miliseconds

Timestamp, in the NMEA standard publishing format. Precision is always to the millisecond (two digits to the right of the decimal point). If source time is precise only to the second, the millisecond is given as '`00`'.

APPENDIX C: NMEA Checksum Example (C++)

```
// SentenceChecksum
//      Input:  Pointer to a character array containing the sentence
//     Output:  True if a checksum could be calculated on a properly-structured sentence
//              False if input was improperly formed
//      Note:   First char must be $, last char must be *
boolSentenceChecksum(char* sentence, unsigned int&cSum)
{
  intlen = strlen(sentence);
  if (len < 10)
    return false;
  if (sentence[0] != '$')  // Sentence must start with '$' or it is malformed
    return false;
  if (sentence[len 1] != '*') // Sentence must end with '*' or it is malformed
    return false;
  cSum = 0;
  for (int i = 1; i <len 1; i++)
    cSum ^= sentence[i];
  returncSum;
}
```