# iAcommsDriver

## Brief Overview

iAcommsDriver is an interface for the WHOI uModem to allow both complete access to modem statistics and facilitate easy transmission and reception of data using the modem. Toby Schneider’s Goby libraries are used for communication with the modem. Most statistics and received data are published to more than one MOOS variable for easier use by other applications and thorough logging.

## Configuration Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable** | **Type** | **Default** | **Description** |
| PortName | String |  | Serial port name |
| ID | Integer |  | ID used by this node for filling in source in uModem packets. |
| PSK\_minipackets | Boolean | False | Use fsk or psk encoding for mini packets. |
| enable\_ranging | Boolean | False | Enable synchronization to PPS for one-way ranging. |
| show\_range\_pulses | Boolean | True | Publish range pulses for visualization in pMarineViewer. |
| Community (global) | String |  | Used to set vehicle name. |

## Subscriptions

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Description** |
| ACOMMS\_TRANSMIT\_DATA | String | Passing ascii data to driver for transmission |
| ACOMMS\_TRANSMIT\_DATA\_BINARY | Binary string | Passing binary data to driver for transmission |
| ACOMMS\_TRANSMIT\_RATE | Double | Integer rate |
| ACOMMS\_TRANSMIT\_DEST | Double | Integer ID of destination (0 for broadcast) |
| NAV\_X, NAV\_Y | Double | Used for posting of range pulses |
| LOGGER\_DIRECTORY | String | to put log file into same directory as pLogger |

## Publications

|  |  |  |
| --- | --- | --- |
| **Variable** | **Type** | **Description** |
| ACOMMS\_RECEIVED\_DATA | Binary string | Data received in a transmission |
| ACOMMS\_RECEIVED\_DATA\_HEX | String | Received data in hex format |
| ACOMMS\_BAD\_FRAMES | String | Comma delimited list of bad frames |
| ACOMMS\_TRANSMITTED\_DATA\_HEX | String | Transmitted data in hex format |
| ACOMMS\_RECEIVED\_ALL | String | DebugString of received ModemTransmission protobuf |
| ACOMMS\_RECEIVED\_SIMPLE | String | Brief summary of reception |
| ACOMMS\_TRANSMIT\_SIMPLE | String | Brief summary of transmission |
| ACOMMS\_DRIVER\_STATUS | String | status of driver, updated every 5 seconds |
| ACOMMS\_DRIVER\_WARNING | String | For debugging information |
| VIEW\_RANGE\_PULSE | String | Posting of range pulses on transmission or reception |
| ACOMMS\_IMPULSE\_RESPONSE | String | Raw CAIRE message from modem |
| ACOMMS\_SNR\_OUT, ACOMMS\_SNR\_IN, ACOMMS\_DQR | Double | Data picked from ACOMMS\_RECEIVED\_ALL for ease of access by other applications |

## Basic Usage

### Driver status

The driver will publish its current status to ACOMMS\_DRIVER\_STATUS at least once every 5 seconds. Status can be “transmitting”, “receiving”, “ready”, or “not running” (only occurs at startup). Transmission requests will be ignored if the driver is not ready.

### Transmitting

The transmission rate is set using the ACOMMS\_TRANSMIT\_RATE variable. See the uModem documentation for a complete listing of possible rates and the size of their data payloads. 13-bit mini-packets can be transmitted by setting rate 100. See section 1.7 for more details on sending mini-packets. Transmit destination is set using the ACOMMS\_TRANSMIT\_DEST variable. For now, only use the default value of 0 (broadcast).

Transmission is initiated when data is posted to either ACOMMS\_TRANSMIT\_DATA or ACOMMS\_TRANSMIT\_DATA\_BINARY. You must use the binary variable if your data contains the byte 0x00. Data will automatically be packaged into frames according to the set rate and truncated if necessary. The driver will post a hex translation of the transmitted data (post truncation) to ACOMMS\_TRANSMITTED\_DATA\_HEX and a brief summary of the transmission information will be posted to ACOMMS\_TRANSMIT\_SIMPLE. A yellow range pulse is posted emanating from the transmitter’s location if range pulses are enabled.

### Receiving

All receptions should be accompanied by a posting to ACOMMS\_RECEIVED\_ALL containing all receive information, including statistics. A brief summary will be posted to ACOMMS\_RECEIVED\_SIMPLE. If data was received, it will be posted to ACOMMS\_RECEIVED\_DATA as binary and ACOMMS\_RECEIVED\_DATA\_HEX as a hex translation. Multiple frames will be concatenated together before publication. A comma delimited string of the indices of bad frames is published to ACOMMS\_BAD\_FRAMES, but no placeholder is included with the received data publication.

As a simple example we explore a hypothetical transmission (note this is not an actual micromodem transmission type). Consider a packet consisting of 4 frames sized 2 bytes each. On the transmitter we post to ACOMMS\_TRANSMIT\_DATA the string “abcdefghi”. The string is truncated and split into frames to be transmitted: “ab”, “cd”, “ef”, and “gh”. The middle two frames are lost. On the receiver the string “abgh” is published to ACOMMS\_RECEIVED\_DATA and “2,3” is published to ACOMMS\_BAD\_FRAMES to indicate that the 2nd and 3rd frames were lost.

If there are no bad frames an empty string will be published to ACOMMS\_BAD\_FRAMES. A posting of “-1” indicates that no frames were received.

The raw impulse response message from the modem is caught and posted to ACOMMS\_IMPULSE\_RESPONSE, primarily for logging purposes. Individual statistics can be posted as their own variables for ease of use. Currently snr\_in, snr\_out, and dqr are posted individually.

## Message Formats

ACOMMS\_RECEIVED\_ALL is created by calling the DebugString() method on the ModemTransmission protobuf. Line endings are replaced with the placeholder “<|>”. The simple acomms parser source code can be used as reference for decoding this and other goby protobuf structures.

ACOMMS\_TRANSMIT\_SIMPLE and ACOMMS\_RECEIVED\_SIMPLE are defined in lib\_acomms\_messages.

Hex formatted messages use colon delimiters between bytes. For example the phrase “Hello world” would be posted as “48:65:6c:6c:6f:20:77:6f:72:6c:64”. Hex values less than 10 will be posted using one digit instead of two (e.g. “61:0:61”).

## Minipackets (rate 100)

Minipackets can carry 13 bits of information passed in two bytes. The micromodem will always perform a bitwise and with 0x1f on the first byte. If only a single byte is passed to the driver for transmission, it will be packed with 0x00 in the first position. See the following examples:

acomms\_transmit\_data\_binary --> acomms\_received\_data  
a) 0x6161 --> 0x0161  
b) 0x0061 --> 0x0061  
c) 0x6100 --> 0x0100  
d) 0x61 --> 0x0061

ACOMMS\_TRANSMITTED\_DATA\_HEX can be used to check the data actually being transmitted in a minipacket.

## Logging

The driver writes a separate “goby log” in the same folder used by pLogger for the MOOS logs. This log includes all of the raw nmea sentences exchanged between the goby uModem driver and the uModem hardware. File names are goby\_logX.txt where X is an integer that is incremented as needed if the driver is restarted. Because iAcommsDriver depends on a publication for pLogger to determine the logging directory, it cannot be run before pLogger is started.

# Lib\_acomms\_messages

Library used for passing acomms related messages containing multiple pieces of information.

## SIMPLIFIED\_RECEIVE\_INFO

### Fields

|  |  |  |
| --- | --- | --- |
| **field** | **type** | **description** |
| Vehicle name | String | Name of the vehicle that sent the transmission |
| Source | Integer | Source id of the transmitter |
| Rate | Integer | Transmission rate (100 for mini) |
| Num frames | Integer | Total number of expected frames |
| Num good frames | Integer | Number of frames correctly received |
| Num bad frames | Integer | Number of frames with errors |

### Format

“vehicle\_name,%s:source,%d:rate,%d:num\_frames,%d:num\_good\_frames,%d:num\_bad\_frames,%d”

## SIMPLIFIED\_TRANSMIT\_INFO

### Fields

|  |  |  |
| --- | --- | --- |
| **field** | **type** | **description** |
| Vehicle name | String | Name of the vehicle that sent the transmission |
| Rate | Integer | Transmission rate (100 for mini) |
| Dest | integer | Destination ID (0 for broadcast) |
| Num frames | Integer | Total number of frames sent |

### Format

“vehicle\_name,%s:rate,%d:dest,%d:num\_frames,%d”

# uPokeDBHex

## Brief Overview

uPokeDBHex is essentially the same as uPokeDB, except that it works for binary strings instead of normal strings. It cannot be used to poke normal strings, but it will still display their contents albeit in hex notation.

## Usage

Exactly the same as uPokeDB for doubles. When poking binary strings, use hex notation with colons to separate bytes.

uPokeDBHex ACOMMS\_TRANSMIT\_DATA\_BINARY=”68:65:6c:6c:6f”

The value of string and binary string variables will also be displayed in hex format after being poked.