1 iAcommsDriver

1.1 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.

1.2 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.
in_sim	Boolean	false	Configure for simulation use
Community (global)	String		Used to set vehicle name.

1.3 Subscriptions

Shaded cells are deprecated -prefer use of ACOMMS_TRANSMIT.

Variable	Type	Description
ACOMMS_TRANSMIT	Binary string	Unified data transmission message provided by HoverAcomms library
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
ACOMMS_TRANSMITTED_REMOTE	Binary string	Simulation only – record of transmissions by all instances of iAcommsDriver

1.4 Publications

Shaded cells are deprecated – prefer use of ACOMMS_RECEIVED and ACOMMS_TRANSMITTED.

Variable	Type	Description
ACOMMS_RECEIVED_DATA	Binary string	Data received in a transmission

ACOMMS_RECEIVED	Binary string	Unified data received message provided by HoverAComms library
ACOMMS_TRANSMITTED	Binary string	Unified data transmitted message provided by HoverAcomms library
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

1.5 Basic Usage

Use of the new transmitting and receiving methods is preferred, but both old and new should work. All publications, both new and old format, will be made regardless of the method used to initiate a transmission.

1.5.1 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.

1.5.2 Transmitting – new version

Use the AcommsTransmission class provided by the HoverAcomms library to construct a complete transmission request included rate, destination, and data. The serialized version of this message should be posted to ACOMMS_TRANSMIT in binary format. If the driver is ready the message will be sent immediately and the details of the transmission will be published using the same AcommsTransmission class to ACOMMS_TRANSMITTED (binary) and ACOMMS_RECEIVED_ALL (non-binary). If the driver is not ready then the transmission request will be ignored and a warning issued.

1.5.3 Transmitting – Deprecated version

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.

1.5.4 Receiving – new version

All information for each reception is published as a single message to ACOMMS_RECEIVED using the AcommsReception class defined in the HoverAcomms library. Use of this class and its included member functions should simplify applications that used to subscribe to multiple variables for acomms reception information.

1.5.5 Receiving - deprecated version

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.

Comment [J1]: not actually sure if this would be 2,3 or 1,2. need to check if 0 or 1 indexed.

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.

1.6 Message Formats

The new message formats used for the variables ACOMMS_TRANSMIT, ACOMMS_RECEIVED, and ACOMMS_TRANSMITTED are all defined in the HoverAcomms library.

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").

1.7 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.

1.8 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.

Comment [J2]: Fix this someday..

1.9 Typical bridging setup

When using pAcommMonitor, which is usually run on the shoreside computer, ACOMMS_RECEIVED and ACOMMS_TRANSMITTED should be bridged from all vehicles to the shoreside.

If running the acomms driver in simulation mode then ACOMMS_TRANSMITTED should also be bridge from the shoreside to all vehicles under the alias ACOMMS_TRANSMITTED_REMOTE.

Comment [J3]: Simulation mode needs work to deal with apparent issues with getting binary variables echoed via the shoreside

2 Lib_acomms_messages

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

2.1 SIMPLIFIED_RECEIVE_INFO

2.1.1 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

2.1.2 Format

“vehicle_name,%s:source,%d:rate,%d:num_frames,%d:num_good_frames,%d:num_bad_frames,%d”

2.2 SIMPLIFIED_TRANSMIT_INFO

2.2.1 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

2.2.2 Format

“vehicle_name,%s:rate,%d:dest,%d:num_frames,%d”

3 uPokeDBHex

3.1 Brief Overview

Comment [J4]: Command line help needs updating

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.

3.2 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.

4 iHoverKayak

4.1 Brief Overview

Interface with the low level control running on the Arduino.

4.2 Configuration

Variable	Type	Default	Description
`PORT_NAME`	`String`	`/dev/ttyO0`	Serial port name (not required since all use the hardware /dev/ttyO0)
`BAUD_RATE`	`Integer`	`115200`	Default 115200 – no need to change
`INVERT_RUDDER`	`Boolean`	`False`	

RUDDER_OFFSET	Double	0	
RADIO_WAIT_TIME	Integer	120	time in seconds to wait for confirmation when switching radio power before switching back

4.3 Subscriptions

Variable	Type	Description
DESIRED_THRUST	double	Thrust output (-100 to 100)
DESIRED_RUDDER	double	Rudder angle (-90 to 90, but further limited by Arduino)
RADIO_POWER	String	“freewave” or something else

4.4 Publications

Variable	Type	Description
VOLTAGE	double	Battery voltage as measured by motor driver
CPU_BOX_TEMP	Double	Temperature in degrees Celsius of cpu box
ROBOTEQ_HEATSINK_TEMP	double	
ROBOTEQ_INTERNAL_TEMP	double	
ROBOTEQ_BATTERY_CURRENT	double	
ROBOTEQ_MOTOR_CURRENT	double	
CPU_BOX_CURRENT	double	
ARDUINO_THRUST	double	Actual output thrust by motor driver
ARDUINO_RUDDER	double	Currently just the set point

Comment [J5]: At some point arduino should get a crude model of rudder position

4.5 Radio Power Switching

Radio power is switched immediately when posting to RADIO_POWER. If the same value is not posted again to RADIO_POWER within RADIO_WAIT_TIME seconds then the driver will switch back.

Comment [J6]: Future versions should also include some automatic switching in case of comms loss by monitoring a heartbeat variable provided by the shore