1 iAcommsDriver

1.1 Configuration Parameters

variable	type	purpose
PortName	String	Serial port name
ID	integer	acomms ID for addressing packets
PSK_minipackets	boolean	Use fsk or psk mini packets
Community (global)	string	get vehicle name

1.2 Subscriptions

variable	type	purpose
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

1.3 Publications

variable	type	purpose
ACOMMS_RECEIVED_DATA	Binary string	Data received in a transmission
ACOMMS_RECEIVED_DATA_HEX	String	Received data in hex format
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

1.4 Basic Usage

1.4.1 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.

1.4.2 Transmitting

Set the destination and rate in advance using the ACOMMS_TRANSMIT_RATE and ACOMMS_TRANSMIT_DEST variables. Rate 100 is used for mini-packets and a destination of 0 is used for broadcasts. Initiate a transmission by posting to 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 (see micromodem documentation). The driver will post a hex translation of the transmitted data 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 when transmitting.

1.4.3 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. Note that there will be no indication in the received data if individual frames were lost – currently you must check ACOMMS_RECEIVED_ALL or ACOMMS_RECEIVED_SIMPLE.

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.

1.5 Message Formats

ACOMMS_RECEIVED_ALL is created by calling the DebugString() method on the ModemTransmission protobuf. Line endings are replaced with the placeholder “<|>”.

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”).

1.6 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.

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”