



EvoLogics GmbH  
Ackerstr. 76  
13355 Berlin, Germany  
Tel.: +49 30 4679 862 - 0  
Fax: +49 30 4679 862 - 01  
Email: [support@evologics.de](mailto:support@evologics.de)

# **EvoLogics Software Defined Modems & USBL Reference Guide**

Document version 1.0

August 2018

2000 - 2018 EvoLogics GmbH. All rights reserved.

Printed in Germany.

EvoLogics GmbH, Ackerstrasse 76, 13355 Berlin, Germany

**Trademarks:** Trademarks and service marks of EvoLogics GmbH (EvoLogics) contained in this document are attributed to EvoLogics with the appropriate symbol. For queries regarding EvoLogics trademarks, contact us at the address shown above.

All other trademarks are the property of their respective holders.

**Restricted Print Permission:** This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from EvoLogics. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

- The publication may be used solely for personal, informational, and noncommercial purposes;
- The publication may not be modified in any way;
- Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and
- EvoLogics reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from EvoLogics.

**Disclaimer:** Information in this publication is subject to change without notice and does not represent a commitment on the part of EvoLogics. The information contained herein is the proprietary and confidential information of EvoLogics or its licensors, and is supplied subject to, and may be used only by EvoLogics customer in accordance with, a written agreement between EvoLogics and its customer. Except as may be explicitly set forth in such agreement, EvoLogics does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. EvoLogics does not warrant that use of such information will not infringe any third party rights, nor does EvoLogics assume any liability for damages or costs of any kind that may result from use of such information.

# Contents

<b>1</b>	<b>Overview</b>	<b>3</b>
<b>2</b>	<b>SDM Protocol</b>	<b>4</b>
2.1	Interface description . . . . .	4
2.1.1	Access SDM PHY . . . . .	4
2.1.2	SDM Frame Format . . . . .	4
<b>3</b>	<b>Using SDM Shell</b>	<b>8</b>
<b>4</b>	<b>SDM with JANUS</b>	<b>10</b>

# 1 Overview

EvoLogics **Software Defined modems** allow to:

- configure **SDM PHY** - an API for the physical layer of the modem
- upload a custom reference signal to **SDM PHY**;
- transmit a custom signal form;
- detect and receive a signal (after SDM PHY detects a signal, matching to the user-specified reference).

A brief description of the API follows.

## **2 SDM Protocol**

### **2.1 Interface description**

#### **2.1.1 Access SDM PHY**

Default socket:

<IP Address>:4200

Here IP Address is the IP address of your modem, listed in its Factory Certificate.  
Make sure you use the correct address, if you changed the factory setting.

#### **2.1.2 SDM Frame Format**

The SDM frame format is:

<0x80,0x00,0x7f,0xff,0x00,0x00,0x00,0x00>,<cmd:8>,<param:24>,<len:32>,  
<data:len\*16>

here all numbers are little-endian.

**To PHY** See commands to PHY below:

Command	Cmd Code	Parameter	Length	Description
STOP	0	0	0	ongoing TX: stop transmitting until FPGABUSY EDGE ongoing RX: stop receiving until FPGABUSY EDGE otherwise: send SDM_CMD_STOP command
TX	1	0	0	ongoing TX: return BUSY,TX and stop ongoing data transmission until FPGABUSY EDGE ongoing RX: return BUSY,RX and stop ongoing data reception until FPGABUSY EDGE otherwise: transmission of <code>len</code> 16-bit samples, transmission begins immediately (TX_ON and SDM_CMD_TX) <code>len</code> must be divisible by 1024, otherwise undefined behavior
RX	2	<code>&gt;=0</code>	0	ongoing TX: return BUSY,TX and stop ongoing data transmission until FPGABUSY EDGE ongoing RX: redefine parameters and continue to RX otherwise: send SDM_CMD_RX command <code>len</code> parameter: length of 16-bit samples to be received if <code>len</code> equals 0 then 16-bit samples will be received till command SDM_CMD_STOP
REF	3	0	<code>&gt;0</code>	ongoing TX or RX: return BUSY and stop ongoing process until FPGABUSY EDGE otherwise: send SDM_CMD_REF
CONFIG	4	<code>thr:16</code> <code>gain:1</code> <code>srclevel:7</code>	0/1	If <code>len==1</code> , data contains preamplifier gain: <code>preamp_gain:4,reserved:12</code> . Preamplifier gain is described in table 2.1.2 below. Example: by <code>thr:350</code> , <code>gain: 1</code> , source level: 2, param value is 0x5e,0x01,0x82
USBL_CONFIG	5	<code>preamp_gain:4</code> <code>sample_rate:3</code> <code>reserved:17</code>	4	<code>write_delay:32</code> in samples, 65535 max <code>data.length:32</code> in samples, 1024, 2048...51200 Sample rate is described in table 2.1.2 below
USBL_RX	6	<code>data_len:16</code> <code>channel:3</code> <code>reserved:5</code>	0	
SYSTIME	7	0	0	

**From PHY** See responses from PHY below:

Command	Cmd Code	Parameter	Length	Description
STOP	0	0	0	report about STOP to higher level
RX	2	0	>=0	start data reception, prepare RX frame header, len contains parameters, predefined by higher layer
USBL_RX	0	0	0	report about STOP to higher level
SYSTIME	0	0	0	report about STOP to higher level

NOTE: by switching to PHY, STOP is generated.

**Preamplifier gain** See the values described below:

Value	Description
0	auto
1	0 dB
2	+6 dB
3	+12 dB
4	+18 dB
5	+24 dB
6	+30 dB
7	+36 dB
8	+42 dB
9	+48 dB
10	+54 dB
11	+60 dB
12	+66 dB
13	+72 dB

**USBL sample rate** See the values described below:

Value	Description
0	250 kHz, default
1	1000 kHz
2	500 kHz
3	250 kHz
4	125 kHz
5	62.5 kHz
6	31.25 kHz

**Source level** See the values described below:

Value	Description
00	max
01	7510 5011 min

**Gain** See the values described below:

Value	Description
0	normal
1	low, -20 dB

**SDM sublayer reports** See reports from SDM sublayer below:

Command	Cmd Code	Parameter	Length	Description
REPORT	255	0	0	stop rejected (not in SDM mode)
		1	len	transmission stopped after len samples (FPGABUSY EDGE)
		2	len	reception stopped after len samples (FPGABUSY EDGE)
		3	0	reference update failed
		3	len	reference update finished
		4	0	config failed
		5	0	usbl config failed
		5	1	usbl config accepted
		7	0	system time request failed
		254	len	garbage dropped
		255	code	unknown command code
BUSY	254	1 or 2	0	in case of race conditions



### 3 Using SDM Shell

See the notes below on using the `sdmsh` SDM shell.

- Get sources:

```
$ git clone https://github.com/EvoLogics/sdmsh.git
```

```
Cloning into 'sdmsh'...
remote: Counting objects: 15, done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 15 (delta 1), reused 15 (delta 1), pack-reused 0
Unpacking objects: 100% (15/15), done.
```

- Go to `sdmsh`:

```
$ cd sdmsh
```

- You need to install the `readline` developer package. In `debian/ubuntu`:

```
$ sudo apt-get install libreadline-dev
```

- Compile:

```
$ make
```

- Do not forget to put S2C modem into PHY state:

```
$ nc 192.168.0.127 9200
```

```
ATP
```

```
or
```

```
+++ATP
```

- You will get a response:

```
INITIATION PHY
```

- Next:

```
$ ./sdmsh 192.168.0.127 4200
```

```
127> help
config      -      Config SDM command.
                  Usage: config <threshold> <gain> <source level> [<preamp_gain>]
usbl_config-      Config SDM USBL command.
                  Usage: usbl_config <delay> <samples> <gain> <sample_rate>
stop        -      Stop SDM command.
                  Usage: stop
ref         -      Update reference signal.
                  Usage: ref [<number of samples>] [<driver>:]<params>
tx          -      Send signal.
                  Usage: tx [<number of samples>] [<driver>:]<parameter>
rx          -      Receive signal [0 is inf].
                  Usage: rx <number of samples> [<driver>:]<params> [[<driver>:]<params>]
usbl_rx     -      Receive signal from USBL channel.
                  Usage: usbl_rx <channel> <number of samples> [<driver>:]<params>
systime     -      Request systime.
                  Usage: systime
usleep      -      Delay in usec.
                  Usage: usleep <usec>
help        -      This help
                  Usage: help [command]
history     -      Display history. Optional [number-lines] by default is 10.
                  Usage: history [number-lines]

<driver> is:
    ascii:<filename> or file extension ".dat" or ".txt".
    This is default driver File format: float (-1.0 .. 1.0) or
    short interger as text line, one value per line

    raw:<filename> or file extension ".raw", ".bin" or ".dmp".
    Binary format: int16_t per value

    tcp:<connect|listen>:<ip>:<port>.
    Opens TCP socket to send or receive data, int16_t per value
```

## Supported drivers <drv>

- ascii:<filename>  
This is default driver.  
File format is float (-1.0 .. 1.0) or short integer as text line, one value per line
- raw:<filename>  
Binary format: int16\_t per value
- tcp:<connect|listen>:<ip>:<port>  
Opens TCP socket to send or receive data, int16\_t per value

**rx** command supports up to 2 sinks.

## sys\_time

```
127> sys_time
current_time = 1374704890, tx_time = 1374200191, rx_time = 4294950616
127>
```

- `current_time` - current system time 32-bit counter
- `tx_time` - 32-bit time counter from beginning of last data transmission
- `rx_time` - 32-bit time counter from beginning of last data reception

## 4 SDM with JANUS

**What is JANUS** JANUS is a simple, robust signaling method for underwater communications. It has been developed at the Centre for Maritime Research and Experimentation (CMRE) with the collaboration of academia, industry, and government with the intention that it should be freely distributed and available to all.

There are 2 JANUS implementations, available at <http://www.januswiki.com/tiki-index.php> in C and Matlab. It is required to register on the site to obtain the source code.

`sdmsh` can be used with `janus-c` implementation from [www.januswiki.com](http://www.januswiki.com) with an additional patch set.

- Please contact EvoLogics to obtain the patch set for `janus-c`.

**Using `sdmsh` with JANUS** In order to follow the reference instructions below, you need to unpack the `janus-c` (version 3.0.1) implementation, apply a patch set from EvoLogics and compile the `janus-c`.

- Create a `janus-c` configuration for signal transmission:

```
cat > txcfg.ini << EOF
--pset-file etc/parameter_sets.csv
--pset-id 1
--stream-driver tcp
--stream-driver-args listen:127.0.0.1:9999
--stream-fs 62500
--stream-format S16
EOF
```

- Create a `janus-c` configuration for signal reception:

```
%cat > rxcfg.ini << EOF
%--pset-file etc/parameter_sets.csv
%--pset-id 1
%--stream-driver tcp
%--stream-driver-args listen:127.0.0.1:9998
%--stream-fs 62500
%--stream-format S16
%EOF
```

- Next,

```
$ sdmsh/> echo "config 30 0 3 0" | ./sdmsh 211 -f -
```

```
%tx cmd CONFIG: 1 samples
%rx cmd REPORT:  CONFIG done
```

- \$ janus-c/> ./janus-rx --config-file rxcfg.ini

Here janus-rx is waiting for the TCP client to start signal reception.

Use the --rx-once option, if you need to stop running janus-rx after a successful demodulation.

- \$ sdmsh/> echo "rx 0 tcp:connect:127.0.0.1:9998 raw:rx.raw" | ./sdmsh 211 -f -

```
tx cmd RX      : 0 samples
rx cmd RX      :
rcv 270328 samples
```

At this point sdmsh is storing the data in rx.raw and passes it to janus-rx, janus-rx performs the JANUS signal processing.

- \$ janus-c/> ./janus-tx --config-file txcfg.ini

Here janus-tx is waiting for the TCP client to start signal transmission.

- \$ sdmsh/> echo "tx 102400 tcp:connect:127.0.0.1:9999" | ./sdmsh 163 -f -

After running the command, JANUS signal transmission starts using sdmsh.