MEX Software Manual

Release 6.10

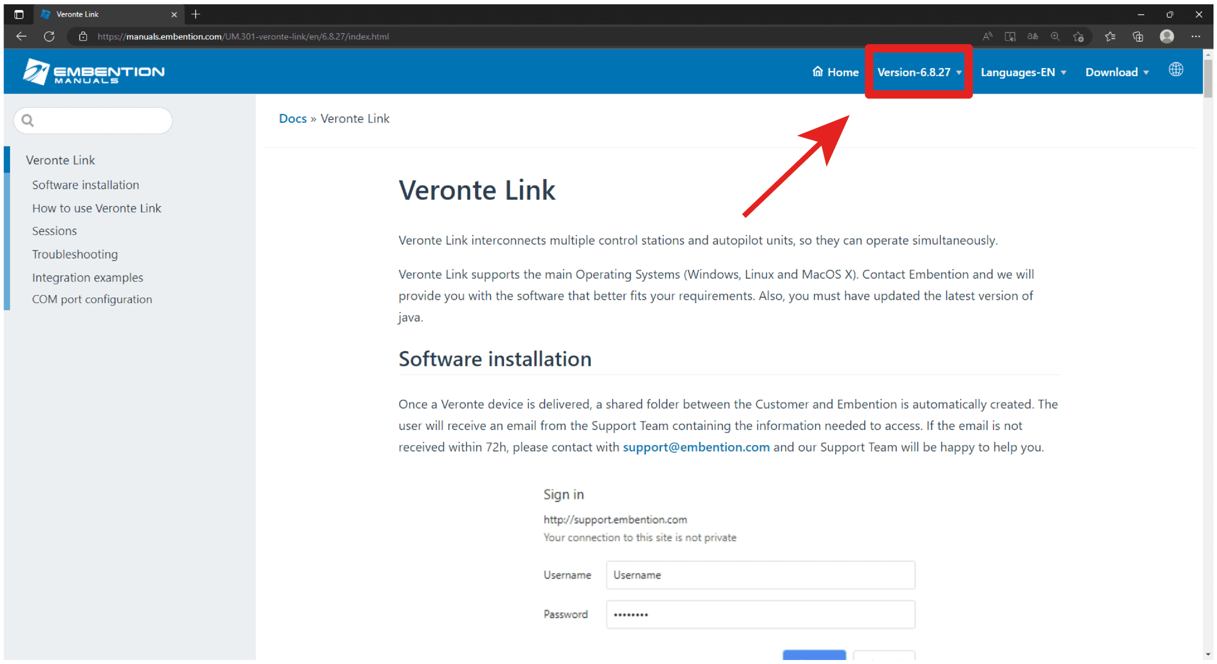
Embention

2023-08-21

CONTENTS

First of all, Veronte Link is required to connect a **MEX** to a computer. Then, it can be configured with MEX PDI Builder and calibrated with MEX PDI Calibration.

Warning: Select your version before reading any user manual for software. The following image shows where to select a version from any Embention user manual.



Veronte Link

Veronte Link interconnects multiple control stations and autopilot units, so they can operate simultaneously.

Veronte Link supports the main Operating Systems (Windows, Linux and MacOS X). Contact Embention and we will provide you with the software that better fits your requirements. Also, you must have updated the latest version of java.

Software installation

Once a Veronte device is delivered, a shared folder between the Customer and Embention is automatically created. The user will receive an email from the Support Team containing the information needed to access. If the email is not received within 72h, please contact with support@embention.com and our Support Team will be happy to help you.

Sign in

http://support.embention.com

Your connection to this site is not private

Username: Username

Password: ••••••••

CHAPTER ONE

SOFTWARE APPLICATIONS

1.1 Veronte Link

Veronte Link stablishes communication between a computer and any Veronte product by creating a VCP bridge. It allows to use multiple control stations and devices to be interconnected, operating simultaneously. **Veronte Link** also includes a post-flight viewer, to reproduce all recorded data from previous flights and generate plots and reports.

Read the user manual for Veronte Link for more information.

1.2 MEX PDI Builder

MEX PDI Builder is the main configuration tool to adapt a **MEX** to a specific vehicle, including user-defined commnication protocols. It includes:

- Telemetry: real-time onboard UAV metrics, such as sensors, actuators and control states.
- Communications: through general purpose inputs and outputs, PWMs and CAN channels.
- Stick control signal management: compatible with **Stick Expander**, Futaba, Jeti, FrSky and TBS. It includes custom configuration for other sticks.
- Arbitration: **MEX** is able to send PWM signals using arbitration in the same way **Veronte Autopilot 4x** does.

Read the user manual for MEX PDI Builder for more details.

1.3 MEX PDI Calibration

MEX PDI Calibration is a straightforward application employed to calibrate the magnetometer embedded in **MEX**. It is recommended to use the **MEX PDI Calibration** the first time and every time **MEX** is employed at a different region, since the magnetic field of the Earth may change.

For more details, read the user manual for MEX PDI Calibration.

Note: By default, **MEX** has not any configuration. In consequence, **MEX** will be in maintenance mode and **Veronte Link** will show the **Loaded with Error** status. Nonetheless, it is possible to load a new configuration with **MEX PDI Builder**; since the maintenance mode allows to connect a computer and load any configuration, with any connection (USB, RS-232, RS-485 or CAN).

LISTS OF VARIABLES

This section shows the variables employed by **MEX**. Nonetheless, **MEX** share variables with **Autopilot 1x**. To read these variables, read the List of variables of **1x Software Manual**.

BIT Variables

ID	Name	Description
330	Jetibox COMM Error	0 for error with Jetibox communiations, 1 for Jetibox OK

Real Variables (RVar) - 32 Bits

ID	Name	Units/Values	Description
322	Internal Magnetometer Raw X in SI	T	Magnetometer raw measurement for X axis
323	Internal Magnetometer Raw Y in SI	T	Magnetometer raw measurement for Y axis
324	Internal Magnetometer Raw Z in SI	T	Magnetometer raw measurement for Z axis
325	Internal Magnetometer Temperature	K	Temperature measured by the internal magnetometer

CHAPTER

THREE

CAN BUS PROTOCOL

This section defines the **MEX** communication protocol.

This is the configuration of messages that must be performed with **Veronte Autopilot 1x** to communicate with **MEX**.

Note: No configuration of these messages is required in **MEX**, as **MEX** is already internally configured to "understand" messages configured in this way.

Warning: For these messages sent from the **Autopilot 1x** to be processed correctly, they must be received by the 'Consumer' **Application processor**.

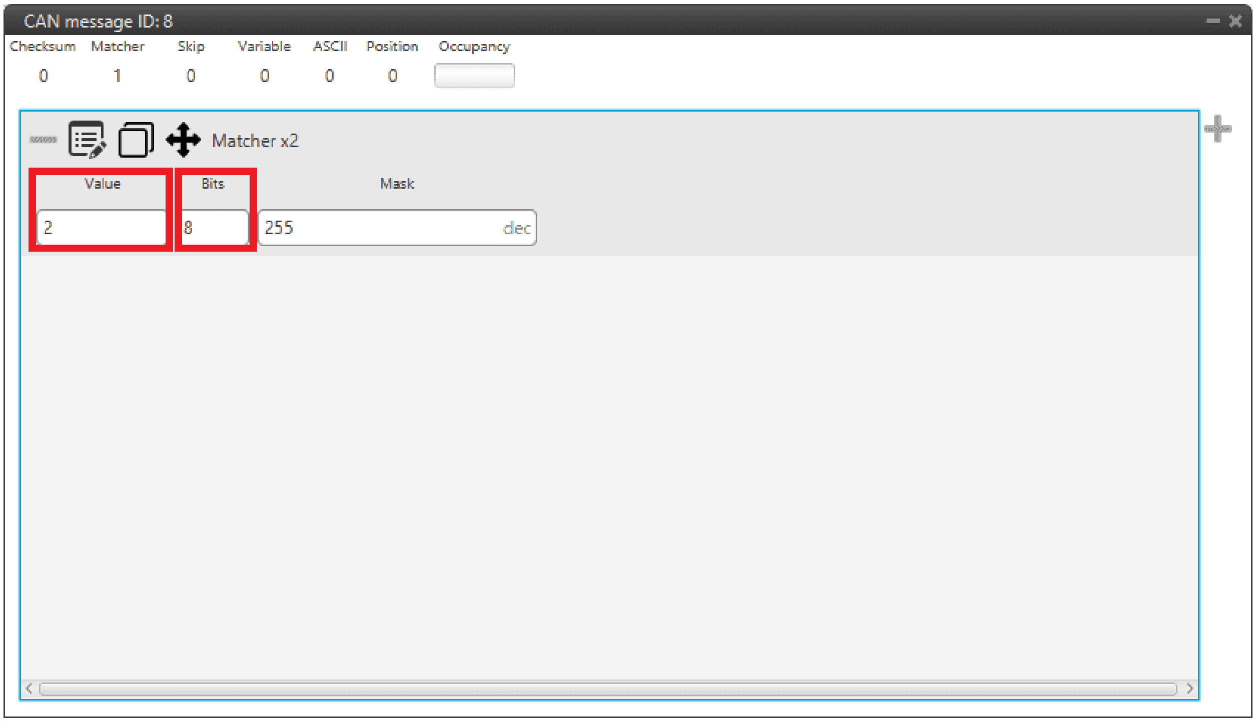
MEX Communication Protocol over CAN bus is defined as follows:

1. **cmd (8 bits - 1 byte):** first byte refers to the **Message Type**.

   Messages Type are defined as follows:

Type	Value	Description
t_arbitration	0	Arbitration message
t_version	1	Version request / response
t_pwm_0_3_set	2	PWMs 0 to 3
t_pwm_4_7_set	3	PWMs 4 to 7
	4	Reserved
t_esc_tm	5	Scorpion Tribunus ESC telemetry data
t_esc_tm2	6	Jeti ESC telemetry data
t_bec_tm1	7	Jeti BEC telemetry data 1
t_bec_tm2	8	Jeti BEC telemetry data 2
t_temp_tm	9	Jeti Temperature sensor telemetry data
t_mcu_cmd	10	Lift MCU battery command
t_pwm_8_11_set	11	PWMs 8 to 11
t_pwm_12_15_set	12	PWMs 12 to 15
t_pwm_16_19_set	13	PWMs 16 to 19
	14	Reserved
	15	Reserved
t_cmd_maint	16	Command to go to Maintenance Mode
t_stick_sel	17	Command for Stick selection
t_mcu_tm1	18	Lift MCU telemetry data 1
t_mcu_tm2	19	Lift MCU telemetry data 2

Note: All these *Message Type* are defined as a "Matcher" in the CAN custom messages configuration. For example, for PWMs 0-3, the *Message Type* will be configured as follows:



CAN message ID: 8

Checksum	Matcher	Skip	Variable	ASCII	Position	Occupancy
0	1	0	0	0	0	

Matcher x2

Value	Bits	Mask
2	8	255 dec

Fig. 1: **Message Type example**

- **Value**: **2**, since it is the value for the message for PWMs 0 to 3 (it is **indifferent to the PWM number**).
- **Bits**: **8**, because the *Message Type* is an 8-bit message.

2. **data (up to 56 bits - 8 bytes)**: The following bytes refer to the **Message data** .

Next sections decribe each one of the possible messages with an example. The following examples include complete messages, so each beginning corresponds to *Message Type*.

3.1 MEX Status

MEX status message is composed as follows:

Type	Value	Bits	Description
cmd (t_version)	1	8	Version request / response
data	-	8	Version - Major
data	-	8	Version - Minor
data	-	8	Version - Revision
data (sysaddr)	-	8	Serial number - address 0
data (sysaddr)	-	8	Serial number - address 1
data	-	1	System Error bit (ID 7)
data (MEX status)	-	1	System power up bit error bit (ID 12)
data (MEX status)	-	1	PDI error bit (ID 9)
data (MEX status)	-	1	Memory Allocationbit (ID 8)
data (MEX status)	-	1	File system error bit (ID 6)
data (MEX status)	-	1	CAN A ERROR bit (ID 73)
data (MEX status)	-	1	CAN B ERROR bit (ID 74)
data (MEX status)	-	1	false
data (MEX status)	-	1	Arbiter enabled
data (MEX status)	-	1	Arbiter status

3.2 Arbitration

MEX Arbitration Status message is composed as follows:

- **Message 1**: Sent when “**Send status**” is enabled

Type	Value	Bits	Description
cmd (t_arbitration)	0	8	Arbitration message
Flag	255 ([0xFF])	8	Status Flag
CAP	-	7	Active Autopilot (Current)
data	-	1	Arbitrating
data	-	1	AP0 Alive
data	-	1	AP1 Alive
data	-	1	AP2 Alive
data	-	1	AP3 Alive (External)
data	-	1	AP0 Ready
data	-	1	AP1 Ready
data	-	1	AP2 Ready
data	-	1	AP3 Ready (External)
data (MEX status)	-	1	System bit error (ID 7)
data (MEX status)	-	1	System power up bit error (ID 12)
data (MEX status)	-	1	PDI bit error (ID 9)
data (MEX status)	-	1	Memory Allocation bit (ID 8)
data (MEX status)	-	1	File system bit error (ID 6)
data (MEX status)	-	1	CAN A bit error (ID 73)
data (MEX status)	-	1	CAN B bit error (ID 74)
data (MEX status)	-	1	false
data (MEX status)	-	1	Arbiter enabled
data (MEX status)	-	1	Arbiter status

- **Message 2** (One for each **Veronte Autopilot 1x**): Sent when "**Send score**" is enabled

Type	Value	Bits	Description
cmd (t_arbitration)	0	8	Arbitration message
data	-	8	Autopilot ID [0, 3]
data	-	32 (4 bytes)	Autopilot score as Float

3.3 Command PWMs

Each PWM in **MEX** has to be associated to a Sub Id that indicates which CAN Bus message's PWM is listening to.

That allows to control up to four PWMs using the same message if it is desired. Each message is composed by 4 PWMs maximum.

- PWMs from 0 to 3 are sent in a message that includes 4 PWMs coded as 12-bit integers:

Type	Value	Bits	Description
cmd (t_pwm_0_3_set)	2	8	PWMs 0 to 3
data (pwm0)	-	12	PWM value for sub-id 0
data (pwm1)	-	12	PWM value for sub-id 1
data (pwm2)	-	12	PWM value for sub-id 2
data (pwm3)	-	12	PWM value for sub-id 3

- PWMs from 4 to 7 are sent in a message that includes 4 PWMs coded as 12-bit integers:

Type	Value	Bits	Description
cmd (t_pwm_4_7_set)	3	8	PWMs 4 to 7
data (pwm0)	-	12	PWM value for sub-id 4
data (pwm1)	-	12	PWM value for sub-id 5
data (pwm2)	-	12	PWM value for sub-id 6
data (pwm3)	-	12	PWM value for sub-id 7

- PWMs from 8 to 11 are sent in a message that includes 4 PWMs coded as 12-bit integers:

Type	Value	Bits	Description
cmd (t_pwm_8_11_set)	11	8	PWMs 8 to 11
data (pwm0)	-	12	PWM value for sub-id 8
data (pwm1)	-	12	PWM value for sub-id 9
data (pwm2)	-	12	PWM value for sub-id 10
data (pwm3)	-	12	PWM value for sub-id 11

- PWMs from 12 to 15 are sent in a message that includes 4 PWMs coded as 12-bit integers:

Type	Value	Bits	Description
cmd (t_pwm_12_15_set)	12	8	PWMs 12 to 15
data (pwm0)	-	12	PWM value for sub-id 12
data (pwm1)	-	12	PWM value for sub-id 13
data (pwm2)	-	12	PWM value for sub-id 14
data (pwm3)	-	12	PWM value for sub-id 15

- PWMs from 16 to 19 are sent in a message that includes 4 PWMs coded as 12-bit integers:

Type	Value	Bits	Description
cmd (t_pwm_16_19_set)	13	8	PWMs 16 to 19
data (pwm0)	-	12	PWM value for sub-id 16
data (pwm1)	-	12	PWM value for sub-id 17
data (pwm2)	-	12	PWM value for sub-id 18
data (pwm3)	-	12	PWM value for sub-id 19

3.4 Lift MCU telemetry

3.4.1 MEX to Autopilot 1x

The telemetry sent by **MEX** through CAN Bus is composed by:

- **Message 1**:

Type	Value	Bits	Description
cmd (t_mcu_tm1)	18	8	Lift MCU telemetry data 1
data	-	8	Battery Serial Number [0]
data	-	8	Battery Serial Number [1]
data	-	8	Battery Temperature (as received from MCU)
data	-	8	Low Cell Voltage (as received from MCU)
	-	4	Reserved (Zeros)
data (Status Bit)	-	1	PWM receiving Ok
data (Status Bit)	-	1	CAN PWM receiving Ok
data (Status Bit)	-	1	CAN B receiving
data (Status Bit)	-	1	CAN A receiving

- **Message 2**:

Type	Value	Bytes	Description
cmd (t_mcu_tm2)	19	1	Lift MCU telemetry data 2
data	-	1	Battery Serial Number [2]
data	-	1	Battery Serial Number [3]
data	-	1	Battery Serial Number [4]
data	-	1	Battery Serial Number [5]
data	-	1	Battery Serial Number [6]
data	-	1	Battery Serial Number [7]

3.4.2 Autopilot 1x to MEX

The telemetry sent from **Autopilot 1x** to **MEX** must be configured as follows:

Type	Value	Bytes	Description
cmd (t_mcu_cmd)	10	1	Lift MCU battery command
data	-	1	SUB-id A
data	-	1	LED Value A
data	-	1	SUB-id B
data	-	1	LED Value B
data	-	1	SUB-id C
data	-	1	LED Value C

Each **MEX** will use the SUB-id of the PWM associated to the “Scorpion Tribunus”/PWM ID to identify the value to be used.

3.5 Scorpion Tribunus ESC Telemetry (Lift)

The telemetry read from the Scorpion ESC is sent as:

Type	Value	Bytes	Description
cmd (t_esc_tm)	5	1	Scorpion Tribunus ESC telemetry data
data	-	1	Input voltage in range [0, 85]
data	-	1	Temperature in Celsius
data	-	1	Error Flags from the ESC
data	-	1	Current in Amps [0, 255]
data	-	1	Consumption in mAmps [0, 25500]
data	-	1	RPMs [0, 25500]
data	-	1	Throttle as percentage*2 [0, 200]

3.6 JetiTM ESC Telemetry

The telemetry read from Jeti-TM compatible ESCs is sent as:

Type	Value	Bytes	Description
cmd (t_esc_tm2)	6	1	Jeti ESC telemetry data
data	-	1	Throttle value [0, 200]
data	-	2	Current RPMs
data	-	10 bits	Input voltage in the range [0, 70] Volts
data	-	10 bits	Temperature in the range [0, 575] Kelvin
data	-	12 bits	Current in the range [0, 400.0] Amps

3.7 Jeti BEC Telemetry

The telemetry read from Jeti BEC will be sent in 2 different messages:

- **Message 1**:

Type	Value	Bits	Description
cmd (t_bec_tm1)	7	8	Jeti BEC telemetry data 1
data	-	16	Device ID
data	-	12	Input voltage in the range [0, 70] Volts
data	-	12	Output voltage in the range [0, 70] Volts
data	-	12	Temperature in the range [0, 575] Kelvin

- **Message 2**:

Type	Value	Bits	Description
cmd (t_bec_tm2)	8	8	Jeti BEC telemetry data 2
data	-	16	Device ID
data	-	12	Current in range [0, 100.0] Amps

3.8 Jeti Temperature Sensor Telemetry

The telemetry read from a Jeti Temperature sensor will be sent as:

Type	Value	Bits	Description
cmd (t_temp_tm)	9	8	Jeti Temperature sensor telemetry data
data	-	16	Device ID
data	-	12	Measured temperature 1 in the range [0, 750] Kelvin
data	-	12	Measured temperature 2 in the range [0, 750] Kelvin

3.9 Set Maintenance Mode Command

This command will configure the **MEX** in maintenance mode, setting its configuration in a way that communications can work over SCI-A, SCI-B or Serial-to-CAN configured as:

- **SCI-A** and **SCI-B**: 115200 bauds, 8 data bits, 1 stop, no parity.
- **Serial to CAN**:
  - TX Id: 1301
  - RX Id: 1301

The format of the command is:

Type	Value	Bytes	Description
cmd (t_cmd_maint)	16	1	Command to go to Maintenance Mode

3.10 Stick Selection Command

This command is used to **enable or disable the MEX PPM reader**. If **address** received matches the **MEX**'s one, MEX PPM reader will be enabled, otherwise it will be disabled.

The format of the command is:

Type	Value	Bytes	Description
cmd (t_stick_sel)	17	1	Jeti Temperature sensor telemetry data
data (sysaddr)	-	1	address 0
data (sysaddr)	-	1	address 1