CEX Software manual

Release 6.8

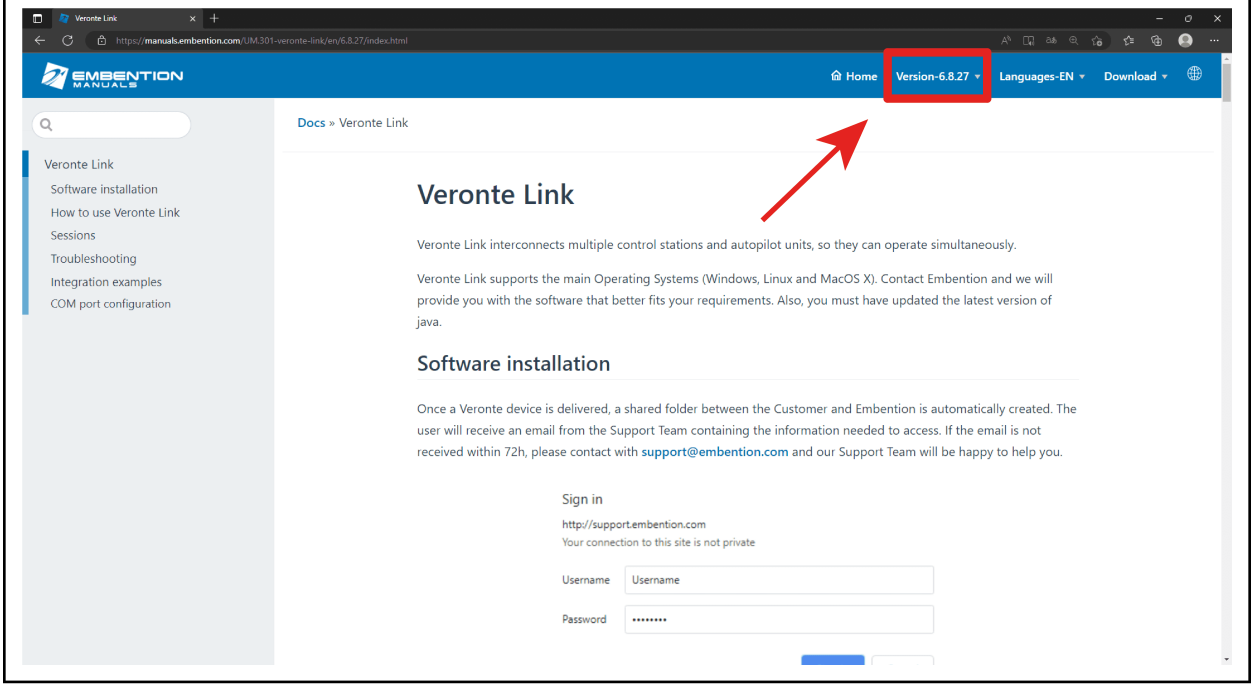
Embention

2023-08-21

CONTENTS

Once the **CEX** has been connected, it is necessary to install Veronte Link to connect it to the computer and CEX PDI Builder to configure it.

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.



CHAPTER

ONE

SOFTWARE APPLICATIONS

1.1 Veronte Link

Veronte Link is the HUB that manages all Veronte and Embention devices, it stablishes communication between a computer and any Veronte product by creating a VCP bridge. **Veronte Link** allows to use multiple control stations and devices to be interconnected, operating simultaneously. It 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 CEX PDI Builder

CEX PDI Builder is the main configuration tool to adapt a **CEX** to be suitable for a specific system, being its main goal to expand the available communication protocols. **CEX PDI Builder** includes:

- Communications: Through general purpose CAN bus, inputs and outputs and PWMs.
- Stick control signal management: Compatible with **Stick Expander**, Futaba, Jeti, FrSky and TBS. It includes custom configuration for other sticks.
- Arbitration: **CEX** is able to send PWM signals using arbitration in the same way **Veronte Autopilot 4x** does.

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

Note: By default, **CEX** has not any configuration. In consequence, **CEX** 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 **CEX 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).

CHAPTER

TWO

CAN BUS PROTOCOL

This section defines the CEX communication protocol.

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

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

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

CEX Communication Protocol via 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:

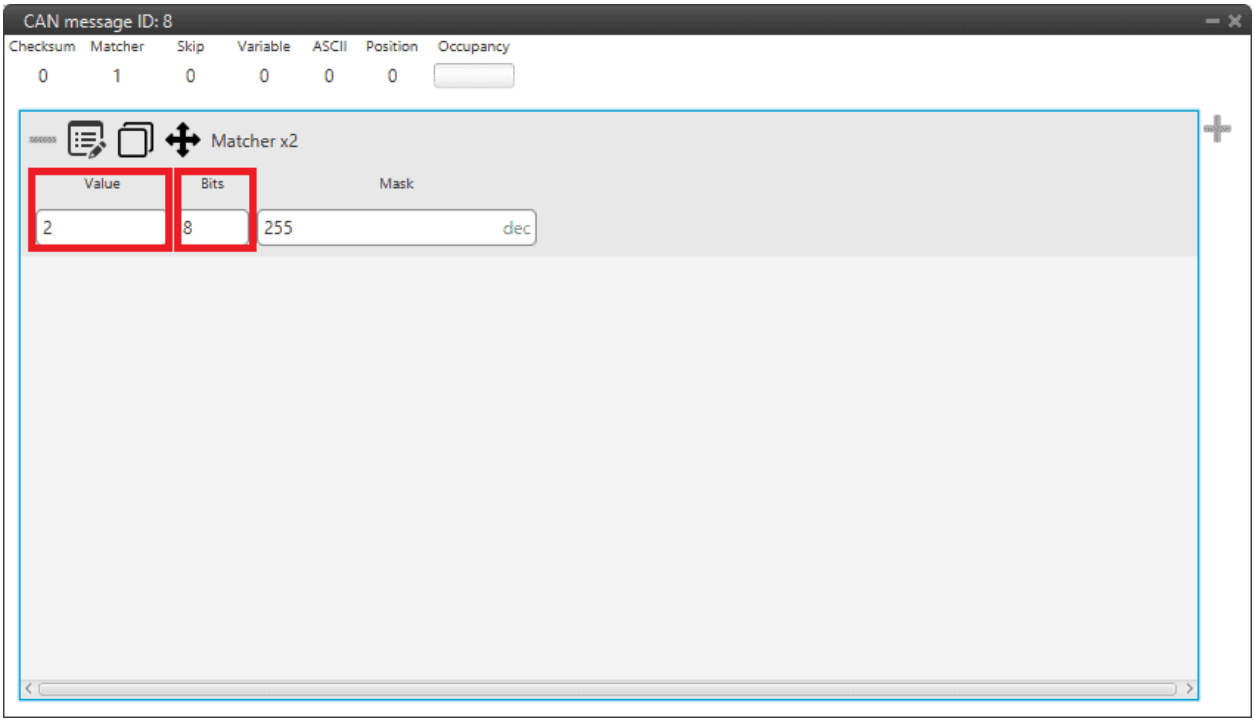


Fig. 1: **Message Type example**

- **Value**: **2**. This is because it is the value for the message for PWMs 0 to 3 (it is **indifferent to the PWM number**).
- **Bits**: **8**. This is 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*.

2.1 CEX Status

CEX status message is composed as follows:

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

2.2 Arbitration

CEX Arbitration Status message is composed as follows:

- **Message 1**: Sent when "**Send status**" is enabled

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

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

Type	Value	Bytes	Description
cmd (t_arbitration)	0	1	Arbitration message
data	-	1	Autopilot ID [0, 3]
data	-	4 (32 bits)	Autopilot score as Float

2.3 Command PWMs

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

That allows to control up to four PWMs using the same message if that 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	Bytes	Description
cmd (t_pwm_0_3_set)	2	1	PWMs 0 to 3
data (pwm0)	-	12 bits	PWM value for sub-id 0
data (pwm1)	-	12 bits	PWM value for sub-id 1
data (pwm2)	-	12 bits	PWM value for sub-id 2
data (pwm3)	-	12 bits	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	Bytes	Description
cmd (t_pwm_4_7_set)	3	1	PWMs 4 to 7
data (pwm0)	-	12 bits	PWM value for sub-id 4
data (pwm1)	-	12 bits	PWM value for sub-id 5
data (pwm2)	-	12 bits	PWM value for sub-id 6
data (pwm3)	-	12 bits	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	Bytes	Description
cmd (t_pwm_8_11_set)	11	1	PWMs 8 to 11
data (pwm0)	-	12 bits	PWM value for sub-id 8
data (pwm1)	-	12 bits	PWM value for sub-id 9
data (pwm2)	-	12 bits	PWM value for sub-id 10
data (pwm3)	-	12 bits	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	Bytes	Description
cmd (t_pwm_12_15_set)	12	1	PWMs 12 to 15
data (pwm0)	-	12 bits	PWM value for sub-id 12
data (pwm1)	-	12 bits	PWM value for sub-id 13
data (pwm2)	-	12 bits	PWM value for sub-id 14
data (pwm3)	-	12 bits	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	Bytes	Description
cmd (t_pwm_16_19_set)	13	1	PWMs 16 to 19
data (pwm0)	-	12 bits	PWM value for sub-id 16
data (pwm1)	-	12 bits	PWM value for sub-id 17
data (pwm2)	-	12 bits	PWM value for sub-id 18
data (pwm3)	-	12 bits	PWM value for sub-id 19

2.4 Lift MCU telemetry

2.4.1 CEX to 1x

The telemetry sent by CEX via CAN Bus is composed by:

- **Message 1**:

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

2.4.2 1x to CEX

The telemetry sent from 1x to CEX 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 CEX will use the SUB-id of the PWM associated to the “Scorpion Tribunus”/PWM ID to identify the value to be used.

2.5 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]

2.6 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

2.7 Jeti BEC Telemetry

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

- **Message 1**:

Type	Value	Bytes	Description
cmd (t_bec_tm1)	7	1	Jeti BEC telemetry data 1
data	-	2	Device ID
data	-	12 bits	Input voltage in the range [0, 70] Volts
data	-	12 bits	Output voltage in the range [0, 70] Volts
data	-	12 bits	Temperature in the range [0, 575] Kelvin

- **Message 2**:

Type	Value	Bytes	Description
cmd (t_bec_tm2)	8	1	Jeti BEC telemetry data 2
data	-	2	Device ID
data	-	12 bits	Current in the range [0, 100.0] Amps

2.8 Jeti Temperature Sensor Telemetry

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

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

2.9 Set Maintenance Mode Command

This command will configure the CEX in maintence mode, setting its configuration in a way that communications can work through 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

2.10 Stick Selection Command

This command is used to **enable or disable the CEX PPM reader**. If the **address** received matches the CEX's one, CEX 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