MEX Software Manual

Release 6.14/1.0

Embention Sistemas Inteligentes, S.A.

2026-02-10

Contents

Scope of Changes

- Version 1.0
  - Added:
    - First version issued

Software applications

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.

Veronte Link

Veronte Link establishes 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.

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.

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.

> **Important**
>
> 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 interest

Lists of Variables

This section shows all the variables employed by **MEX**.

BIT Variables

ID	Name	Description
0	Always fail	This signal is always fail - 0 for fail, 1 for OK
1	Always OK	This signal is always OK - 0 for fail, 1 for OK
5	Power error	Power supply state
6	File system error	System file manager
7	System error	This bit checks whether the system is running properly. 0 for system error, 1 for system OK
8	Memory Allocation	RAM allocation - 0 for trying to use more than available memory, 1 for running
9	PDI error	PDI files - Dependent on PDI Error Source (UVar 50) • 0 for wrong PDI configuration: if PDI Error Source > 0 • 1 for running OK: if PDI Error Source == 0
10	CIO Low or C2 Error	Bits 400 and 401 are recommended instead - 0 for

ID	Name	Description
		failed, 1 for OK **Warning** Deprecated variable
12	System power up bit error	Power up - 0 for error, 1 for OK
13	Reset and write disabled	Reset and non-operation PDI writes are allowed - 0 for disabled, 1 for enabled
16	Stack core 1 usage FAIL	0 for stack overflow, 1 for OK
53	Sensor-Internal Magnetometer (LIS3MDL)	Internal LIS3MDL magnetometer - 0 for disabled, 1 for enabled
60	Sensor-External I2C device 0	External communication I2C of device 0
65	SCI A Transmitting (Sara)	Serial Communication Interface - sara transmission
66	SCI A Receiving (Sara)	Serial Communication Interface - sara reception. 0 for not receiving, 1 for receiving
67	SCI B Transmitting (Radio)	Serial Communication Interface - radio transmission
68		

ID	Name	Description
	SCI B Receiving (Radio)	Serial Communication Interface - radio reception. 0 for not receiving, 1 for receiving
69	SCI C Transmitting (RS485)	Serial Communication Interface - RS485 transmission
70	SCI C Receiving (RS485)	Serial Communication Interface - RS485 reception. 0 for not receiving, 1 for receiving
73	CAN A ERROR	CAN A state - 0 for error, 1 for OK
74	CAN B ERROR	CAN B state - 0 for error, 1 for OK
75	CAN A warning	CAN A state - 0 for warning, 1 for OK
76	CAN B warning	CAN B state - 0 for warning, 1 for OK
96-98	SCI A-C receiving error	SCI A to C - 0 for error in this port (invalid format or configuration), 1 for OK
102-103	CAN A-B receiving	CAN A to B communication - 0 for not receiving, 1 for receiving
104-105	Stick PPM 0-1 not detected	Stick PPM 0-1 - 0 for not detecting, 1 for detecting
108-109	Stick PPM 2-3 not detected	Stick PPM 2-3 - 0 for not detecting, 1 for detecting

ID	Name	Description
111-112	CAN A-B transmitting	CAN signals A to B - 0 for not transmitting, 1 for transmitting
120-123	Pulse 0-3 not detected	Pulse 0 to 3 detection - 0 for pulse not detected, 1 for detected
329	3.3V power source	0 for error, 1 for OK
330	Jetibox COMM Error	0 for error with Jetibox communications, 1 for Jetibox communication OK
400	C1 Low Frequency	Low priority thread frequency • 0 for error → Low priority thread running frequency < 10 Hz • 1 for OK → Low priority thread running frequency → 10 Hz
402	Acquisition step missed	• 0 for Acquisition step missed → High priority thread frequency fluctuation is higher than permitted (1%) • 1 for Acquisition Task OK (High priority thread frequency fluctuation is under set limits (1%)
403	CIO Hi Overload warning	High priority thread overload • 0 for Acquisition Task overload → Acquisition Task Maximum CPU Ratio > 90%

ID	Name	Description
		- 1 for Acquisition Task usage OK → Acquisition Task Maximum CPU Ratio ≤ 90% **Note** Non-recoverable variable
800-807	PWM 0-7 GPIO Off	PWM GPIO 0-7 communication State - 0 for Off, 1 for On
816-819	EQEP_A-I (GPIO17-20) Off	Input/Output State - 0 for Off, 1 for On
1010-1019	Custom msg 0-9 Rx Error	Custom message timeout - 0 for error, 1 for OK
1200-1209	User BIT 00-09 Error	User bit 00 to 09 - 0 for error, 1 for OK

Real Variables (RVar) - 32 Bits

ID	Name	Units/ Values	Description
50	CAN-A Tx Rate	pkts/s	CAN-A transmission packet rate
51	CAN-B Tx Rate	pkts/s	CAN-B transmission packet rate
52	CAN-A Tx skip Rate	pkts/s	CAN-A messages delayed because no mailbox is available for sending

ID	Name	Units/ Values	Description
53	CAN-B Tx skip Rate	pkts/s	CAN-B messages delayed because no mailbox is available for sending
300	Relative Timestamp	s	Time spent since power-on of the system
313	Magnetometer - X Body Axis	T	Magnetometer measurement for X axis **Warning** Deprecated variable
314	Magnetometer - Y Body Axis	T	Magnetometer measurement for Y axis **Warning** Deprecated variable
315	Magnetometer - Z Body Axis	T	Magnetometer measurement for Z axis **Warning** Deprecated variable
322		T	

ID	Name	Units/ Values	Description
	Internal LIS3MDL Magnetometer Raw X in SI		**Internal LIS3MDL Magnetometer** raw measurement for X axis
323	Internal LIS3MDL Magnetometer Raw Y in SI	T	**Internal LIS3MDL Magnetometer** raw measurement for Y axis
324	Internal LIS3MDL Magnetometer Raw Z in SI	T	**Internal LIS3MDL Magnetometer** raw measurement for Z axis
325	Internal LIS3MDL Magnetometer Temperature	K	**Internal LIS3MDL Magnetometer** temperature
700-703	RPM 0-3	rad/s	Angular speed associated to pulse captured 0-3
800-805	PWM 0-5	custom type	Pulse Width Modulation signal 0 to 5
1100-1104	Lidar 0-4 Distance	m	Configurable variables for Lidar distances 0 to 4
1320	CEX/MEX ADC 3.3V Input 0	V	MEX ADC 3.3 V input 0

ID	Name	Units/ Values	Description
1322-1323	CEX/MEX ADC 5.0V Input 0-1	V	MEX ADC 5.0 V inputs 0 and 1
1324	CEX/MEX ADC 12.0V Input 0	V	MEX ADC 12.0 V input 0
1326	CEX/MEX ADC 36.0V Input 0	V	MEX ADC 36.0 V input 0
1328-1329	CEX/MEX ADC vIn 0-1	V	MEX External power supplies 0 and 1
1330	PCB Temperature	K	MEX PCB Temperature (from ADC input)
1331	ADC HW Version	V	Hardware version of MEX ADC
1450-1453	Captured Pulse 0-3	customType	Input values from pulses
3100-3119	User Variable 00-19 (Real - 32 Bits)	customType	Free variables for the user to use

Integer Variables (UVar) - 16 Bits

ID	Name	Description
50	PDI Error Source	Index for PDI error source identification. For further information, consult the List of PDI

ID	Name	Description
		errors section of the **1x Software Manual**
51	Operator error source	Index for operation error source identification
54	4XV Veronte CAP	Current Autopilot 1x selected
90	Version Major	Major software version
91	Version Minor	Minor software version
92	Version Revision	Revision software version
95	UAV Address	UAV address
450	CAN-A Tx errors	CAN A communication errors in transmission
451	CAN-A Rx errors	CAN A communication errors in reception
452	CAN-B Tx errors	CAN B communication errors in transmission
453	CAN-B Rx errors	CAN B communication errors in reception
454-455	CAN to Serial 0-1 frames dropped	Lost messages during CAN to Serial transformations
495-496	Global configuration state (crc) of files-memory	Global configuration state (crc) of files-memory

ID	Name	Description
	(Higher-Lower 16 bits)	
497	Config manager status (flash / sd / maintenance mode)	Configuration manager status
498-499	Global configuration state (crc) of files-memory	Global configuration state (crc) of files-memory
600	PPM channel 0 output	MEX PPM channel output
620	Jetibox max successfully parsed message	Maximum Jetibox messages successfully parsed
1000-1019	User Variable 00-19 (Unsigned Integer - 16 bits)	Free variables for user

List of Addresses

Every Embention device communicate with other devices/tools using its address through VCP.

The following list contains all these addresses:

Address	Recognized as	Description
0	Dummy for pdi builders	Dummy for pdi builder

Address	Recognized as	Description
1	Cloud	**Veronte Cloud address**
2	Vlink	Address used by **Veronte Link** app to communicate with Veronte units
2-3	App + Address	Veronte applications addresses. **App 2** is the one used by default by Veronte applications, although App 3 is also available
255-511	App dynamic + Address	**Dynamic addresses** for Veronte applications
998	Broadcast	To **all devices** on a network
999	Address unknown	This address can be used for a device that **does not** have a **valid address** configured
1000-1777	1x v4.0 + Address	Specific address of an **Autopilot 1x** with **hardware version 4.0**
1778-3999	1x v4.5 + Address	Specific address of an **Autopilot 1x** with **hardware version 4.5**
4000-17999	1x v4.8 + Address	Specific address of an **Autopilot 1x** with **hardware version 4.8**
18000-19899	1x BCS + Address	Specific address of a **BCS** unit

Address	Recognized as	Description
19900-19999	1x v4.7. For internal use only + Address	Specific address of an **Autopilot 1x** with **hardware version 4.7**
20000-21999	Smart Can Isolator + Address	Specific address of a **Smart Can Isolator** unit
30000-31999	MC01 + Address	Specific address of a **MC01** unit
32000-34999	MC24 motor controller + Address	Specific address of a **MC24** unit
35000-39999	MC110 motor controller + Address	Specific address of a **MC110** unit
40000-41999	CEX + Address	Specific address of a **CEX** with **hardware version 1.2**
42000-43999	MEX + Address	Specific address of a **MEX** unit
44000-49999	CEX2 + Address	Specific address of a **CEX** with hardware **version 2.0**
50000-51089	Arbiter v1.0 + Address	Specific address of an **Arbiter** with **hardware version 1.0**
51090-51999	Arbiter v1.2 + Address	Specific address of an **Arbiter** with **hardware version 1.2**
52000-59999	Arbiter v1.8 + Address	Specific address of an **Arbiter** with **hardware version 1.8**

Address	Recognized as	Description
60000-65535	Reserved + Address	Reserved addresses
65536-69631	Virtual v4.0 + Address	Specific address of a **Virtual Autopilot 1x** with **hardware version 4.0**
69632-73727	Virtual v4.5 + Address	Specific address of a **Virtual Autopilot 1x** with **hardware version 4.5**
73728-77823	Virtual v4.8 + Address	Specific address of a **Virtual Autopilot 1x** with **hardware version 4.8**

CAN Bus protocol

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

On the one hand, there are some specific messages that are already configured internally in the system so no configuration in **MEX** is required for them. Consequently, the device that is intended to communicate with **MEX** must correctly configure those messages. This message configuration is detailed below in the Application processor section.

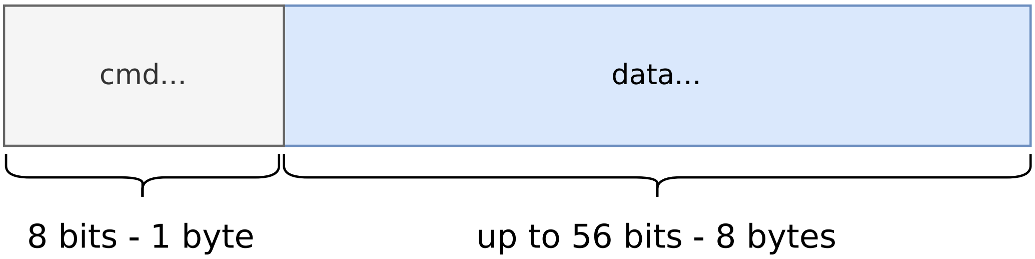
On the other hand, it is also detailed the structure of other messages that are not contemplated by default in the **MEX** configuration but that can be carried out for communication and operation with other devices.

Application processor

This is the configuration of specific messages that must perform any device to communicate with **MEX**.

> ⓘ **Note**
>
> No configuration of these messages is required in **MEX**, as it is already internally configured to process messages configured in this way.
>
> For these messages to be processed correctly, they must be received by the 'Consumer' **Application processor**.

MEX Communication Protocol over CAN bus is defined as follows:



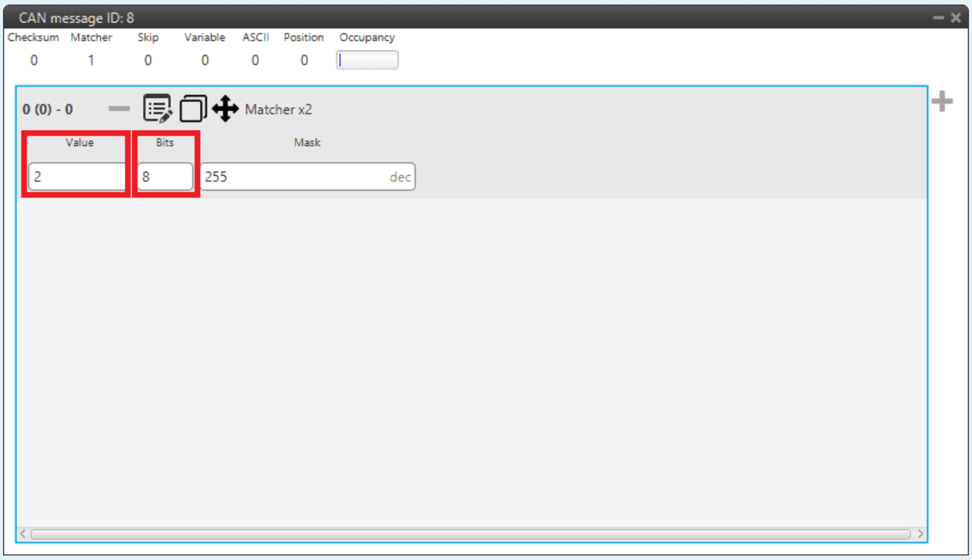
CAN messages structure

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	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

Type	Value	Description
t_stick_sel	17	Command for Stick selection
t_mcu_tm1	18	MCU telemetry data 1
t_mcu_tm2	19	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 | |
>
> 0 (0) - 0 Matcher x2
>
> | Value | Bits | Mask |
> |---|---|---|
> | 2 | 8 | 255 dec |
>
> **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.

1. **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.

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)

Type	Value	Bits	Description
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

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

Type	Value	Bits	Description
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

Type	Value	Bits	Description
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

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

Type	Value	Bits	Description
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

Type	Value	Bits	Description
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

Type	Value	Bits	Description
data (pwm2)	-	12	PWM value for sub-id 18
data (pwm3)	-	12	PWM value for sub-id 19

A complete example of how to command PWMs from **Veronte Autopilot 1x** and read them into **MEX** can be consulted in the Commanding/Reading PWMs - Integration examples section of the **MEX PDI Builder** user manual.

MCU telemetry

From MEX

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

- **Message 1**:

Type	Value	Bits	Description
cmd (t_mcu_tm1)	18	8	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

Type	Value	Bits	Description
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	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]

To MEX

The telemetry sent to **MEX** must be configured as follows:

Type	Value	Bytes	Description
cmd (t_mcu_cmd)	10	1	MCU battery command

Type	Value	Bytes	Description
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.

Scorpion Tribunus ESC Telemetry

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]

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

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

Type	Value	Bits	Description
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

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

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

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	Command for Stick selection
data (sysaddr)	-	1	address 0
data (sysaddr)	-	1	address 1

MEX as external magnetometer

In this section it is explained how to configure the **MEX** magnetometer to be used as an external magnetometer for another device, either via CAN or serial.

CAN

The CAN messages sent by **MEX** must have the following structure:

- **CAN Id**: It can be in standard frame format (11-bits) or in extended frame format (29-bits). The CAN Id frame format will depend on the CAN protocol supported by the receiving device.
- **Variables**: The **MEX** variables associated with the magnetic data are:
    - ID **313**: **Magnetometer - X Body Axis**
    - ID **314**: **Magnetometer - Y Body Axis**
    - ID **315**: **Magnetometer - Z Body Axis**

In addition, users must configure the sending **period** and **endianness** of these messages, as well as the **baudrate** of the CAN bus.

> ⓘ **Note**
>
> Detailed information on **how to build CAN messages** can be consulted in the Custom Messages types - Input/Output section of the **1x PDI Builder** user manual.

The device receiving this information, must be configured properly so that it matches what has been configured in **MEX**.

Serial

The serial messages sent by **MEX** must have the following structure:

- **Variables**: The **MEX** variables associated with the magnetic data are:
    - ID **313**: **Magnetometer - X Body Axis**
    - ID **314**: **Magnetometer - Y Body Axis**
    - ID **315**: **Magnetometer - Z Body Axis**

- **Checksum**: It is useful to include a checksum to verify that the message is sent and received correctly.

In addition, users must configure the sending **period** of these messages, as well as the **baudrate** of the serial port (RS232/RS485).

> ⓘ **Note**
>
> Detailed information on **how to build CAN messages** can be consulted in the Custom Messages types - Input/Output section of the **1x PDI Builder** user manual.

The device receiving this information, must be configured properly so that it matches what has been configured in **MEX**.

Communication with MEX

MEX can also be used to receive data through a communication protocol and transmit it through another.

Serial reception and CAN transmission

- For **serial reception**, follow the steps below:

1. Set the **baudrate** of the serial port used, RS232 or RS485.
2. In order to communicate with the device sending the data, serial messages must be built to match the serial protocol specified by that device.

   > ⓘ **Note**
   >
   > Normally, it contains the variables to be stored in **MEX** and a checksum to verify that the message is sent and received correctly. Please refer to the list of variables in this manual to see the variables available in **MEX**.

3. Configure the **timeout** of the messages. Remember that it has to be higher than the message sending period specified in the sending device.
4. Set the **time to idle** for these messages.

- For **CAN transmission**, follow the steps below:

1. Build a CAN message with the following structure:
    - **CAN Id**: It can be in standard frame format (11-bits) or in extended frame format (29-bits). The CAN Id frame format will depend on the CAN protocol supported by the receiving device.
    - **Variables**: The variables in which the information received via serial has been stored must be set.

      Please refer to the list of variables in this manual to see the variables available in **MEX**.
2. Configure the sending **period** and **endianness** of these messages, as well as the **baudrate** of the CAN bus.

The device receiving this information, must be configured properly so that it matches what has been configured in **MEX**.

CAN reception and serial transmission

- For **CAN reception**, follow the steps below:

1. In order to communicate with the device sending the data, the CAN messages must be built to match the CAN protocol specified by that device.

    > ⓘ **Note**
    >
    > Normally, it is composed of a CAN Id and variables to be stored in **MEX**. Please refer to the list of variables in this manual to see the variables available in **MEX**.

2. Configure the sending **period** and **endianness** of these messages, as well as the **baudrate** of the CAN bus.

- For **serial transmission**, follow the steps below:

1. Build a serial message with the variables received via CAN:
    - **Variables**: The variables in which the information received via CAN has been stored must be set.

      Please refer to the list of variables in this manual to see the variables available in **MEX**.
    - **Checksum**: It is useful to include a checksum to verify that the message is sent and received correctly.
2. Configure the sending **period** of these messages, as well as the **baudrate** of the serial port (RS232/RS485).

The device receiving this information, must be configured properly so that it matches what has been configured in **MEX**.

Firmware Changelog

This section presents the changes between firmware versions of Veronte MEX.

6.14.61

This section presents the changes between the previous firmware version of Veronte MEX, **v.6.12.84**, and this firmware version, **v.6.14.61**.

Improved

- CAN Custom RX bits