4x Software Manual

Release 6.12

Embention

2024-04-25

CONTENTS

1 Software applications **3**
- 1.1 Veronte Link . . . 4
- 1.2 1x Software Manual . . . 4
- 1.3 1x PDI Builder . . . 4
- 1.4 4x PDI Builder . . . 4

2 Lists of interest **5**
- 2.1 Activation System Error bits . . . 5
- 2.2 Lists of variables . . . 5
  - 2.2.1 Bit Variables . . . 6
  - 2.2.2 Real Variables (RVar) - 32 Bits . . . 9
  - 2.2.3 Integer Variables (UVar) - 16 Bits . . . 9
- 2.3 List of PDI errors . . . 11

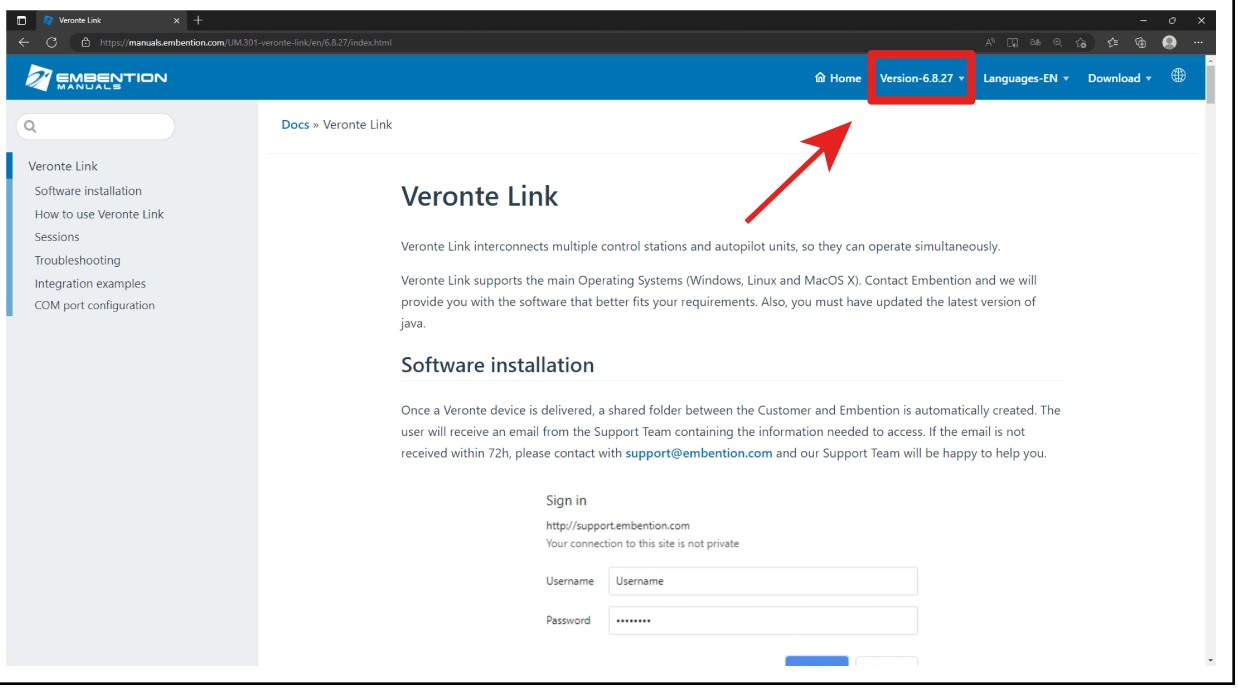
3 CAN Bus protocol **13**
- 3.1 Arbitration messages . . . 14
  - 3.1.1 Status Message . . . 14
  - 3.1.2 Score Message . . . 15
  - 3.1.3 Ready Message . . . 16
  - 3.1.4 Arbitration Message . . . 16

4 Status management **17**
- 4.1 Arbiter boot . . . 17
- 4.2 Status message . . . 17
- 4.3 Ready status . . . 17
- 4.4 Alive status . . . 19
- 4.5 Maintenance Mode . . . 20

In this manual the user can consult a brief description of all the applications created and designed to work together with the **Veronte Autopilot 4x**.

In addition, links are available to access the manuals for each application.

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

Due to the elements present in the **Autopilot 4x** system (Arbiters and Autopilots 1x), the software applications with which 4x works along are the ones listed below.

The following diagram summarizes the operation regarding these applications:

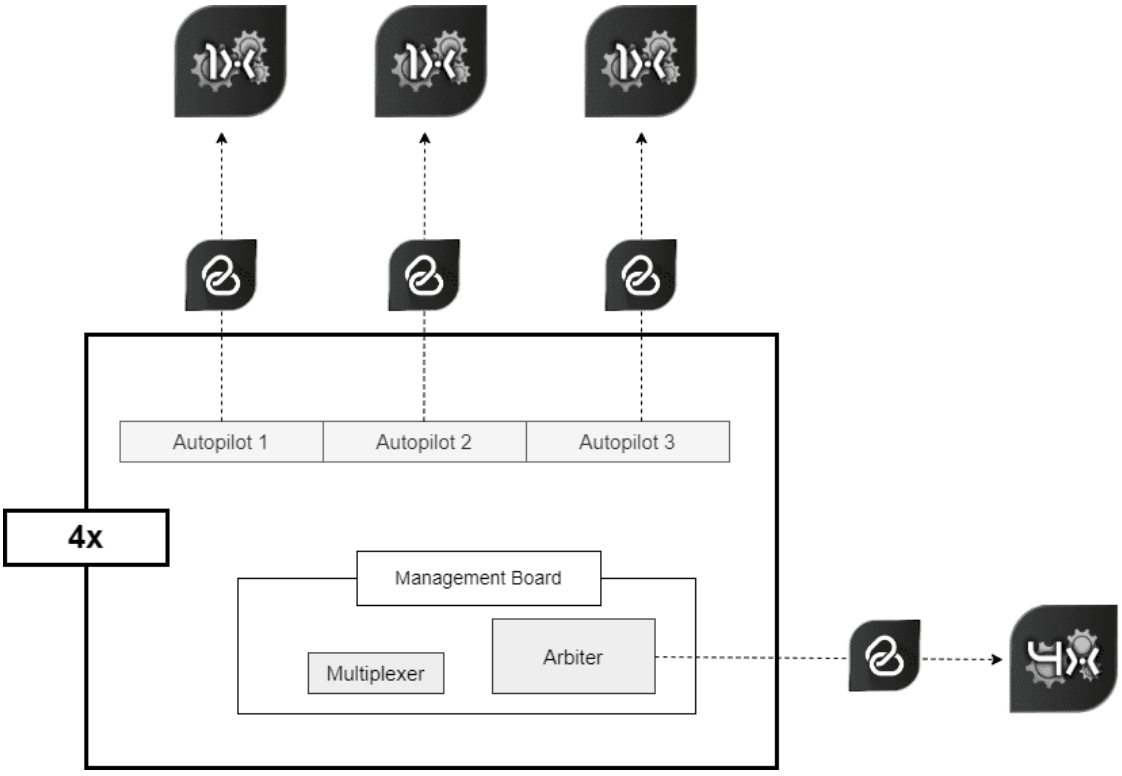


Fig. 1: **4x application diagram**

1.1 Veronte Link

Each inner **Autopilot 1x** and the **Arbiter** must be configured individually. First of all, a connection to a computer is required employing **Veronte Link**. All devices can be connected one by one or simultaneously, since **Veronte Link** is able to manage multiple connections.

For more information, read the user manual for Veronte Link.

1.2 1x Software Manual

To operate individually any **Veronte Autopilot 1x**, read the 1x Software Manual, since all software applications are applicable to **Autopilot 4x**.

1.3 1x PDI Builder

Each internal **Autopilot 1x** must be configured with **1x PDI Builder**, it allows to adapt the autopilot to a specific vehicle, including user-defined commnication protocols. **1x PDI Builder** includes:

- Telemetry: real-time onboard UAV metrics, such as sensors, actuators and control states.
- Configuration: edit vehicle settings, such as servo trim, interface/port management and modes.
- Automations: actions that are automatically executed when a set of configured conditions are accomplished.
- Block Programs: **Veronte Autopilot 1x** can be programmed with a friendly-user programming language.

For more information, read the user manual for 1x PDI Builder.

1.4 4x PDI Builder

4x PDI Builder allows to configure the **Arbiter** communications, including CAN buses and input/output signals. It also configures the criterion to select the autopilot which controls the aircraft.

For more information, read the user manual for 4x PDI Builder.

CHAPTER

TWO

LISTS OF INTEREST

This section contains all the lists with information of interest for the user.

2.1 Activation System Error bits

The **System Error** variable is indicated by bit number 7. This bit checks whether the system is running properly. If one of certain malfunctions occur, the **System Error** will be set as 0 and the FTS will be activated. Othwerwise, if everything is OK, it will remain as 1.

> **Warning:** This bit works different for **Autopilot 1x** and **Arbiter**. This explanation is for **Arbiter**.

The **System Error** will be triggered and remain as 0 if one of the following unwanted events happens:

- An error occurred with **System power up** according to *bit 12*.
- RAM allocation is in error state due to try using more memory than available, this is indicated with a 0 on *bit 8*.
- CAN A bus is not working, hence *bit 73* is set as 0.
- CAN B bus is not working, hence *bit 74* is set as 0.
- One of the internal voltages is not in range, i.e. bit 5 of one of the Autopilots 1x is set as 0.
- There is not any autopilot alive.
- Low priority task frequency is not correct, i.e. *bit 400* is set as 0.
- Acquisition task frequency is not correct, i.e. *bit 402* is set as 0.

2.2 Lists of variables

This section shows the variables employed by **Veronte Autopilot 4x**, both variables of the Arbiter and of Autopilots 1x.

A suitable configuration of the Autopilots 1x is key for ensuring proper communication and operation of the Autopilot 4x. Consider the structure of *Arbitration messages*, and consult the Lists of variables section of **1x Software Manual** for further information.

2.2.1 Bit Variables

Important: Variables marked with "*" are stored in **Autopilot 1x**. If communication is configured accordingly, they represent information transmitted from the **Arbiter**.

ID	Name	Description
6	File System Error	System file manager - 0 for error, 1 for running
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 - 0 for wrong PDI configuration, 1 for running OK
10	CIO Low or C2 Error **Warning: Deprecated variable**	CIO low or C2 failed. Bits 400 and 401 are recommended instead
11	4X CAN failed	Fail of CAN communication between 1x and Arbiter - 0 for fail, 1 for communication OK This bit is initialized with 1. When it is set as 0, it cannot recover until 1x is restarted
12	System Power up BIT Error	Power up - 0 for error, 1 for OK
16	Stack C1 usage FAIL	0 for stack overflow of core 1, 1 for OK
20*	4XV System	System Error bit (bit 7) from the Arbiter
21*	4XV System Power up BIT	Autopilot 4x power up - 0 for fail, 1 for running OK
22*	4XV PDI	PDI files for Autopilot 4x - 0 for wrong PDI configuration, 1 for OK
23*	4XV Memory allocation	RAM allocation - 0 for trying to use more than available memory, 1 for OK
24*	4XV CAN-A BUS OFF	Autopilot 4x CAN A bus - 0 for error, 1 for running OK
25*	4XV CAN-B BUS OFF	Autopilot 4x CAN B bus - 0 for error, 1 for running OK
26*	4XV C1 arbiter	Main Task of CP1 in Autopilot 4x - 0 for error, 1 for running OK
27*	4XV Acquisition arbiter	Autopilot 4x acquisition task in real time - 0 for error, 1 for OK

continues on next page

Table 1 – continued from previous page

ID	Name	Description
28*	4XV Power A	State of power supply for Autopilot 4x - 0 for error, 1 for OK
29*	4XV not in maintenance mode	1 for NOT in mantenance mode - 0 for maintenance mode
30*	4XV Alive 0	Indicates whether Autopilot 1x number 0 is sending status messages or not - 0 for dead, 1 for alive
31*	4XV Alive 1	Indicates whether Autopilot 1x number 1 is sending status messages or not - 0 for dead, 1 for alive
32*	4XV Alive 2	Indicates whether Autopilot 1x number 2 is sending status messages or not - 0 for dead, 1 for alive
33*	4XV Alive 3 external	Indicates whether external Autopilot is sending status messages or not - 0 for dead, 1 for alive
34*	4XV Ready 0	Inner Autopilot 1x number 0 state - 0 for not ready, 1 for ready
35*	4XV Ready 1	Inner Autopilot 1x number 1 state - 0 for not ready, 1 for ready
36*	4XV Ready 2	Inner Autopilot 1x number 2 state - 0 for not ready, 1 for ready
37*	4XV Ready 3 external	External Autopilot 1x state - 0 for not ready, 1 for ready
38*	4XV Arbitrating	Arbiter state - 0 for not ready, 1 for ready
39*	4XV File Open Error	System file manager state - 0 for error, 1 for running OK
40*	4XV PDI version not compatible	PDI files state - 0 for not compatible with current version, 1 for compatible
41*	4XV Stack usage FAIL	0 for memory overflow allocated for local variables, 1 for OK
42*	4XV PWM1 GPIO Off	GPIO/PWM 1 Value to read - 0 for OFF, 1 for ON
43*	4XV PWM2 GPIO Off	GPIO/PWM 2 Value to read - 0 for OFF, 1 for ON
44*	4XV PWM3 GPIO Off	GPIO/PWM 3 Value to read - 0 for OFF, 1 for ON
45*	4XV PWM4 GPIO Off	GPIO/PWM 4 Value to read - 0 for OFF, 1 for ON
46*	4XV PWM5 GPIO Off	GPIO/PWM 5 Value to read - 0 for OFF, 1 for ON
47	4XV Watchdog Error. (v4.7+)	*For version 4.7 or higher* - 0 for watchdog signal is not read correctly, 1 for OK
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

continues on next page

Table 1 – continued from previous page

ID	Name	Description
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
117	Main Power Error	Main power supply A. It will be 0 fot indicating error state
124	4XV Vcc for Arbiter CPU Error	Power state of CPU Arbiter (Based on *RVar 1360*) - 0 for error, 1 for OK
125	4XV Vcc-A Error	State of redundant power supply A (Based on *RVar 1361*) - 0 for error, 1 for OK
126	4XV Vcc-B Error	State of redundant power supply B (Based on *RVar 1362*)- 0 for error, 1 for OK
127	4XV Vcc-0 Error	Power supply for inner Autopilot 1x number 0 (Based on *RVar 1363*) - 0 for error, 1 for OK
128	4XV Vcc-1 Error	Power supply for inner Autopilot 1x number 1 (Based on *RVar 1364*) - 0 for error, 1 for OK
129	4XV Vcc-2 Error	Power supply for inner Autopilot 1x number 2 (Based on *RVar 1365*) - 0 for error, 1 for OK
183	4X Selected	4x Veronte Autopilot selected - 0 when this AP is not the selected AP, 1 when this AP is the selected one
230-293	4XV Custom msg 0-63 Rx Error	Custom message of Arbiter - 0 for timeout, 1 for OK
400	C1 Low Frequency	Low priority tasks frequency • 0 for error → Running frequency < 10 Hz • 1 for OK → Running frequency > 10 Hz
402	Acquisition step missed	• 0 for Acquisition step missed → C1 hi frequency fluctuation is higher than permited (1%) • 1 for Acquisition step OK → C1 hi frequency fluctuation is under set limits (1%)
800-805	PWM 0-5 GPIO Off	PWM GPIO 0-5 communication Sate - 0 for Off, 1 for On
1200-1209	User BIT 00-09 Error	User bit 0 to 9 - 0 for error, 1 for OK

2.2.2 Real Variables (RVar) - 32 Bits

Important: Variables marked with "*" are stored in **Autopilot 1x**. If communication is configured accordingly, they represent information transmitted from the **Arbiter**.

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
53	CAN-B Tx Skip Rate	pkts/s	CAN-B messages delayed because no mailbox is available for sending
300	Relative Timestamp	s	CAN-A messages delayed because no mailbox is available for sending
1350-1356	4XV ADC0-6 Converted Value	V	4XV ADC0-6 Converted Values
1357-1358	4XV Internal ADC7-8 Converted Value	V	4XV Internal ADC7-8 Converted Values
1359	4XV Internal ADC 9 Converted Value	V	4XV Internal Arbiter identifier (A or B)
1360	4XV Vcc for arbiter	V	Vcc for arbiter
1361	4XV Vcc-A 3.3V		Vcc-A 3.3V
1362	4XV Vcc-B 3.3V		Vcc-B 3.3V
1363	4XV Vcc-0		Vcc-0
1364	4XV Vcc-1		Vcc-1
1365	4XV Vcc-2		Vcc-2
1366*	4XV Autopilot 0 Score	customType	Score of Autopilot 1x number 0
1367*	4XV Autopilot 1 Score	customType	Score of Autopilot 1x number 1
1368*	4XV Autopilot 2 Score	customType	Score of Autopilot 1x number 2
1369*	4XV Autopilot External Score	Decimal	Score of external Autopilot 1x

2.2.3 Integer Variables (UVar) - 16 Bits

Important: Variables marked with "*" are stored in **Autopilot 1x**. If communication is configured accordingly, they represent information transmitted from the **Arbiter**.

ID	Name	Description
2	Internal ADC 0	Internal ADC pin **Note:** Variable for internal use
3-7	ADC 0-4	Direct reading of ADC pins

continues on next page

Table 2 – continued from previous page

ID	Name	Description
8-17	Internal ADC 1-10	Internal ADC pins **Note:** Variables for internal use
53	4XV Veronte ID	ID of the Autopilot 1x for the redundant configuration (0 - 3)
54	4XV Veronte CAP	Current Autopilot 1x selected - If the Autopilot has version 4.7 or higher, it is obtained from MUX readings, otherwise it is copied from RVar 55 (in this table)
55*	4XV Veronte selected	Autopilot 1x selected
56*	4XV Config manager status (flash / sd / safe mode)	Configuration manager state (flash, sd or safe)
57*	4XV File system status	State error for DFS2 file system
58*	4XV CAN to Serial 0 frames dropped	Lost messages during transformations CAN to Serial 0
59*	4XV CAN to Serial 1 frames dropped	Lost messages during transformations CAN to Serial 1
60-69*	4XV Internal ADC 0-9	4XV Internal ADC pind 0-9
70*	4XV VCC arbiter	4XV VCC for arbiter
71*	4XV VCC A	4XV VCC A
72*	4XV VCC B	4XV VCC B
73*	4XV VCC 0	4XV VCC 0
74*	4XV VCC 1	4XV VCC 1
75*	4XV VCC 2	4XV VCC 2
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	Global configuration state (crc) of files (Higher 16 bits)	Global configuration state (crc) of files
496	Global configuration state (crc) of memory (Higher 16 bits)	Global configuration state (crc) of memory
498	Global configuration state (crc) of files	Global configuration state (crc) of files

continues on next page

Table 2 – continued from previous page

ID	Name	Description
499	Global configuration state (crc) of memory	Global configuration state (crc) of memory

2.3 List of PDI errors

This section shows the errors that can be displayed by **Veronte Autopilot 4x**. The rest of errors can be read in the List of PDI errors section of **1x Software Manual**.

Code	Nº	Explanation
pdi_arbitration	10000	Error ID for Arbitration cfg
pdi_arbitration_can	10001	Error ID for Arbitration_can cfg
pdi_arbitration_can1	10002	Error ID for Arbitration_can cfg
pdi_arb_cfg0	10003	Error ID for Arb cfg preferred ap oor
pdi_arb_cfg1	10004	Error ID for Arb cfg method oor(out of range)
pdi_arb_cfg2	10005	Error ID for Arb cfg tmin oor
pdi_arb_cfg3	10006	Error ID for Arb cfg hysteresis oor
pdi_arb_init_time	10011	Error ID for Arbiter Power Init Time less than 0
pdi_arb_varcfg	10013	Incorrect arbiter variable configuration

CHAPTER

THREE

CAN BUS PROTOCOL

CAN message structure is defined by two main parts: **cmd** and **data**.

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:

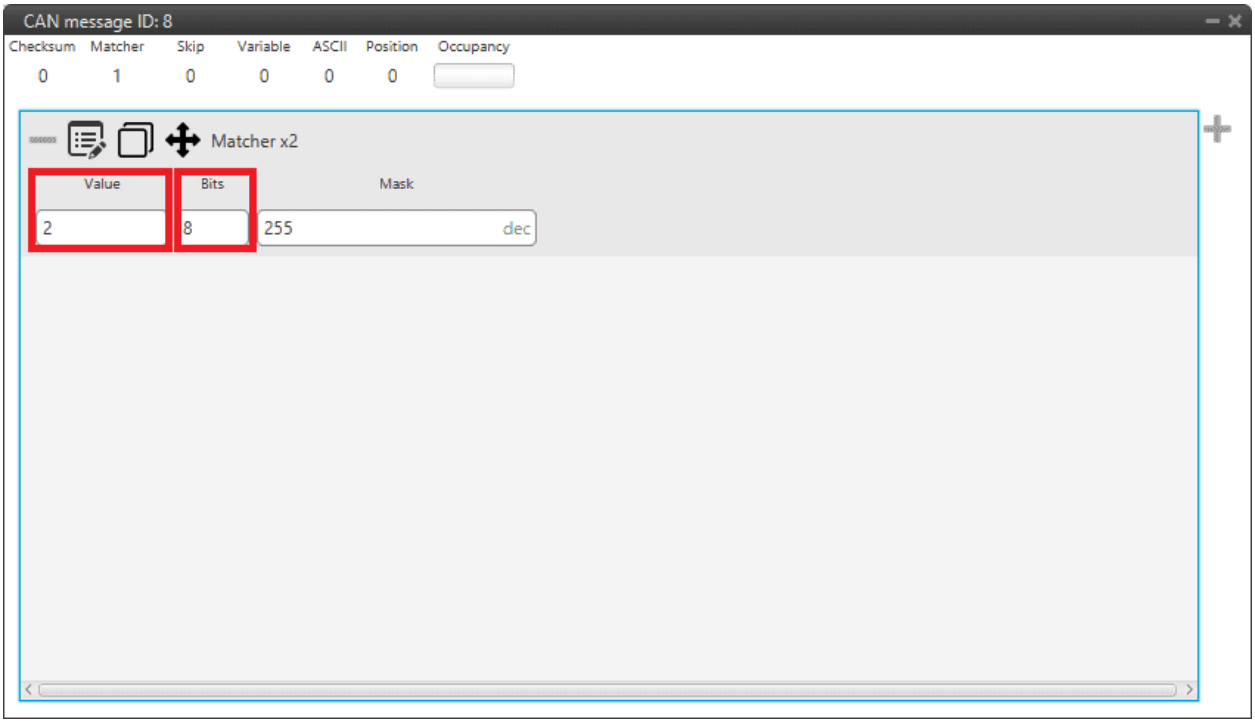


Fig. 1: **Message Type example**

- **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**.

The following examples include complete messages, so each beginning corresponds to *Message Type*.

3.1 Arbitration messages

3.1.1 Status Message

Status message summarizes the status of all autopilots. This message is a **producer** in the **4x PDI Builder** configuration, since it is created from the CAN messages of all three or four autopilots.

Byte	Position	Value	Description
0	0 - 7	0x00	Header
1	0 - 7	0xFF	Status message Header
2	0 - 6	0 - 3	Selected AP (0 = AP1, 1 = AP2, 2 = AP3, 3 = External AP)
	7	0 - 1	Arbitration OFF/ON
3	0	0 - 1	AP1 Alive
	1	0 - 1	AP2 Alive
	2	0 - 1	AP3 Alive
	3	0 - 1	External AP Alive
	4	0 - 1	AP1 Ready
	5	0 - 1	AP2 Ready
	6	0 - 1	AP3 Ready
	7	0 - 1	External AP Ready
4	0	0 - 1	CBIT. System error. Will fail if any of the below items fail.
	1	0 - 1	PBIT. System Boot Ok.
	2	0 - 1	PDI Ok. Will fail if there is an error in configuration files.
	3	0 - 1	Memory Allocation OK
	4	0 - 1	CAN A Ok
	5	0 - 1	CAN B Ok
	6	0 - 1	CIO Low Task Ok
	7	0 - 1	CIO High Task Ok
5	0	0 - 1	Power OK. All power indicators are OK.
	1	0 - 1	A Bus Voltage Ok
	2	0 - 1	B Bus Voltage Ok
	3	0 - 1	Arbiter Voltage Ok
	4	0 - 1	AP1 Voltage Ok
	5	0 - 1	AP2 Voltage Ok
	6	0 - 1	AP3 Voltage Ok
	7	0 - 1	Arbiter Mode. 1 = Normal, 0 = Maintenance

3.1.2 Score Message

Score message contains the final score of an specific autopilot. This message is a **producer** in the **4x PDI Builder** configuration, so it is created from the CAN messages of all three or four autopilots.

Byte	Position	Value	Description
0	0 - 7	0x00	Header
1	0 - 7	0 - 3	Autopilot ID (0 = AP1, 1 = AP2, 2 = AP3, 3 = External AP)
2 - 5	0 - 31	0 - 0xFFFF FFFF	Autopilot Arbitration Score **Note:** The variable in which this score message will be stored must be a **real variable** (single-precision float)

3.1.3 Ready Message

Ready message is sent from **Autopilot 1x** to **Arbiter**. It tells whether an **Autopilot 1x** is ready to fly or not. It is as **consumer** in the **4x PDI Builder** configuration, so it is stored in memory.

Byte	Position	Value	Description
0	0 - 7	0x00	Header
1	0 - 7	0xFF	Ready Message Header
2	0 (1 bit)	0 - 1	Ready/Not Ready

3.1.4 Arbitration Message

Arbitration message tells the arbitration value of a specific variable. It is as **consumer** in the **4x PDI Builder** configuration, since it is stored in memory.

Byte	Position	Value	Description
0	0 - 7	0x00	Header
1	0 - 7	0 - 31	Arbitration Variable Number
2 - 5	0 - 31	0 - 0xFFFF FFFF	Arbitration Variable Value **Note:** Variables sent as arbitration variables must be **real variables** (single-precision float)

CHAPTER

FOUR

STATUS MANAGEMENT

4.1 Arbiter boot

When the **Arbiter** is booted, it enters an **Iddle** state, trying to verify the status of the system.

While **Iddle**, the **Status Message** is not sent.

If all checks success, the **Arbiter** will enter **Normal mode**.

If, after 30 seconds, system errors are still present, **Arbiter** will enter **Maintenance mode**.

4.2 Status message

Once in **Normal mode**, the **Arbiter** will start sending the **Status** and **Scores** telemetry messages.

The frequency of these messages is configurable. They can also be disabled.

The status message contains information such as:

- Arbitration ON/OFF
- Selected autopilot
- System BITs
- etc.

While in **Maintenance Mode**, the **Arbiter** will send the **Status** message, even if it is disabled, but will not send the **Scores** messages.

4.3 Ready status

After the **Arbiter** enters **Normal mode**, it will wait until a Ready Message from each autopilot is received. Only then the Arbitration will start.

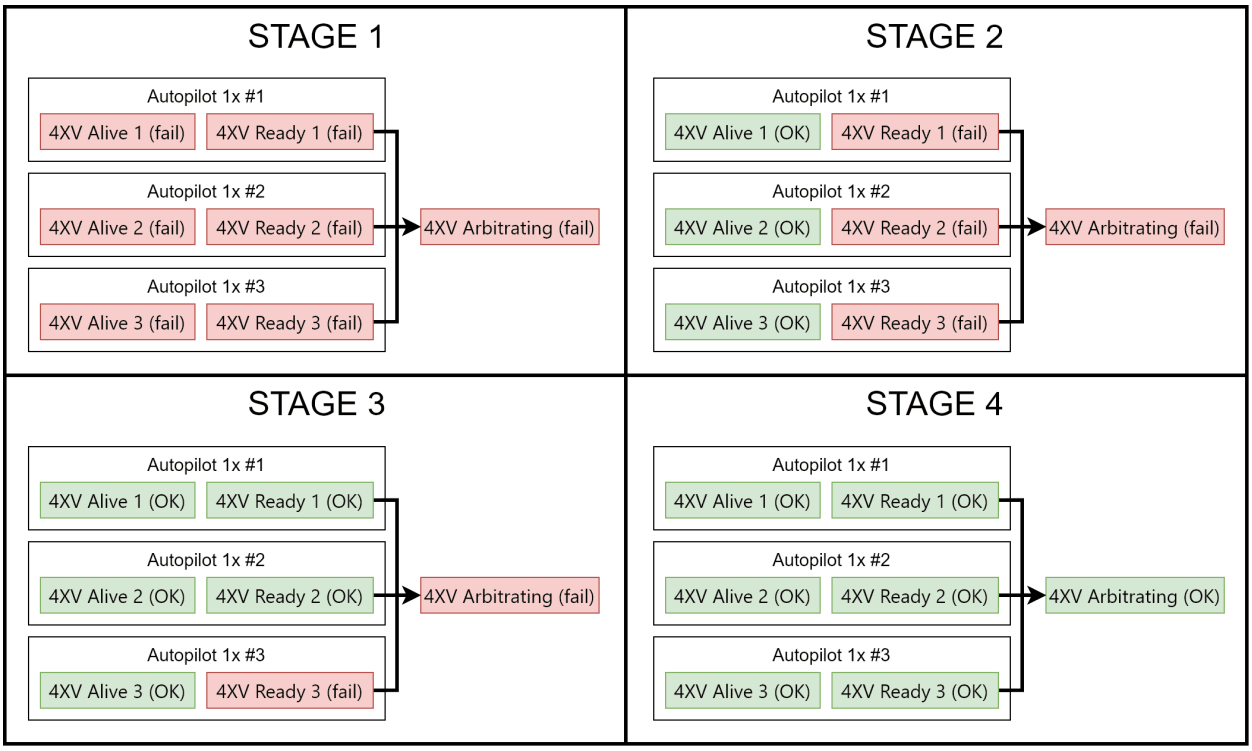


Fig. 1: **Example of normal arbitration start**

If the Arbiter is in maintenance mode, the arbitration will not start. Even if the **Ready messages** are received.

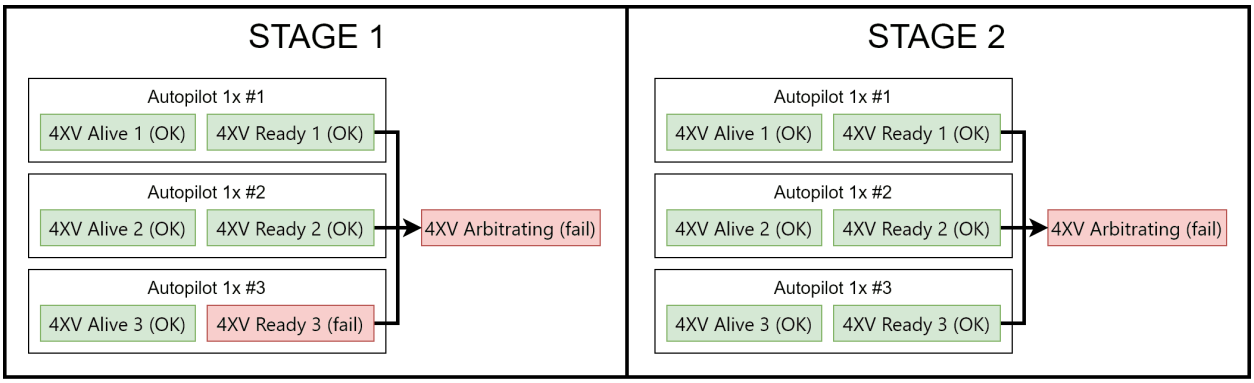


Fig. 2: **Example of failed arbitration start**

4.4 Alive status

Once arbitration starts, all autopilots are declared as alive by default, but it is possible that they are declared as **Dead** if a critical error is found.

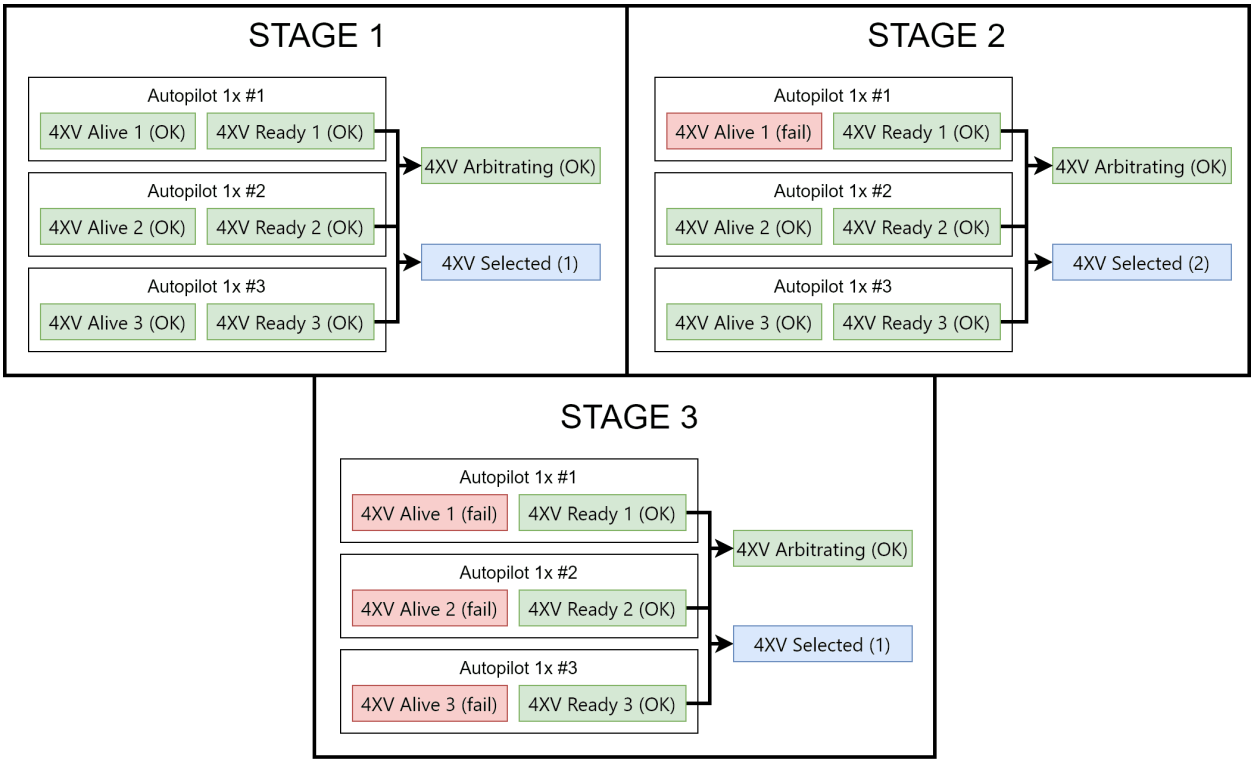


Fig. 3: **Example of APs being declared Dead**

The **Arbiter** will declare an autopilot dead if one of the following incidences is found:

- One of the arbitration messages (including the Ready message) is not received for 0.1 seconds.
- A **Not Ready** message is received.
- A **System Not OK** error is raised on any of the autopilots, activating the **FTS signal**.
- The **watchdog** signal for any autopilot is not ok.

Warning: Make sure to configure the sending of arbitration messages as **High priority**. Otherwise, the sending of messages could be shortly interrupted by a higher priority task and the autopilot will be declared **Dead**.

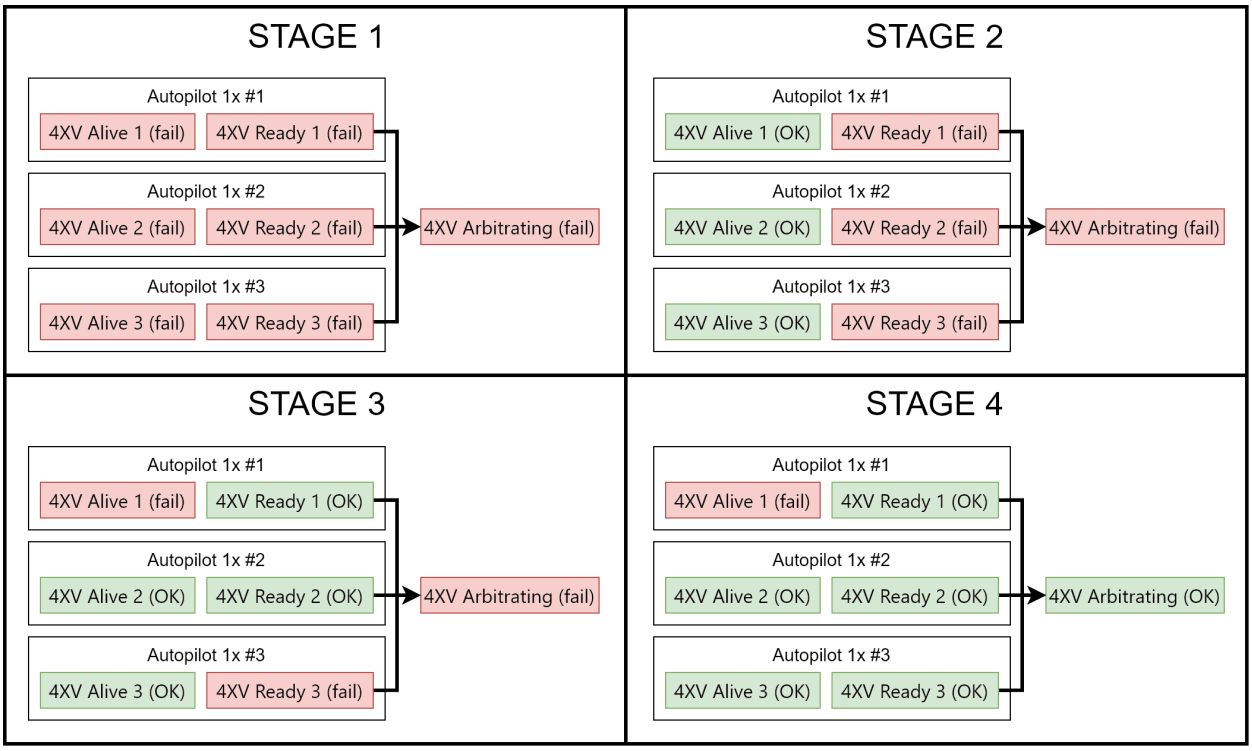


Fig. 4: **APs being declared Dead when arbitration starts due to missed messages**

A **Dead** autopilot can never be selected again as long as arbitration persists.

The **Dead** status is not reversible, it is necessary to reboot the whole system in order to recover a **Dead** autopilot.

If the number of **Alive** autopilots is 2 or less, relative arbitration variables are disabled (since at least 3 autopilots are needed).

If **only one** autopilot is **Alive**, it will be selected no matter the score.

If all autopilots are **Dead**, the **Preferred** autopilot will be selected.

4.5 Maintenance Mode

Maintenance mode is used for changing the **Arbiter** configuration.

Arbitration is disabled while in maintenance mode.

Arbiter will also enter maintenance Mode after a failed boot.

The reason of the failed boot can be checked in the **Status message**.