API and hardware interfaces of the Marvelmind Boxie 2

Version 2025.07.31
Valid for SW pack v8.157 and newer

www.marvelmind.com

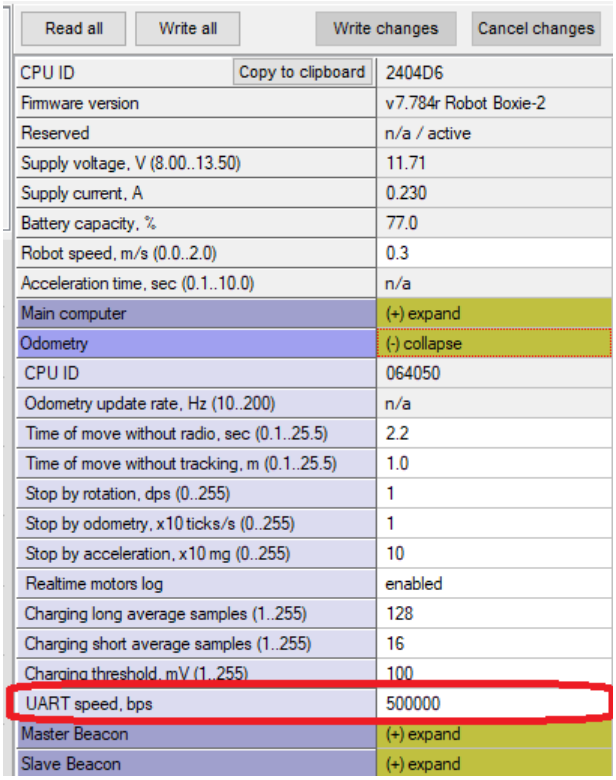
Table of contents

1. Connection to Boxie 2

For communication with Boxie 2 robot, it shall be connected to an external device via any of the following interfaces:

1. Connect to UART – 3 wires for bidirectional communication required. See the picture of hardware interface below. Logic level of UART transmitter is CMOS 3.3V. Default baudrate is 500 kbps, configurable from the Dashboard from following list: 4.8, 9.6, 19.2, 38.4, 57.6, 115.2, 500 kbps. Format of data: 8 bit, no parity, 1 stop bit.

UART baudrate can be changed in the “odometry” section of robot settings:



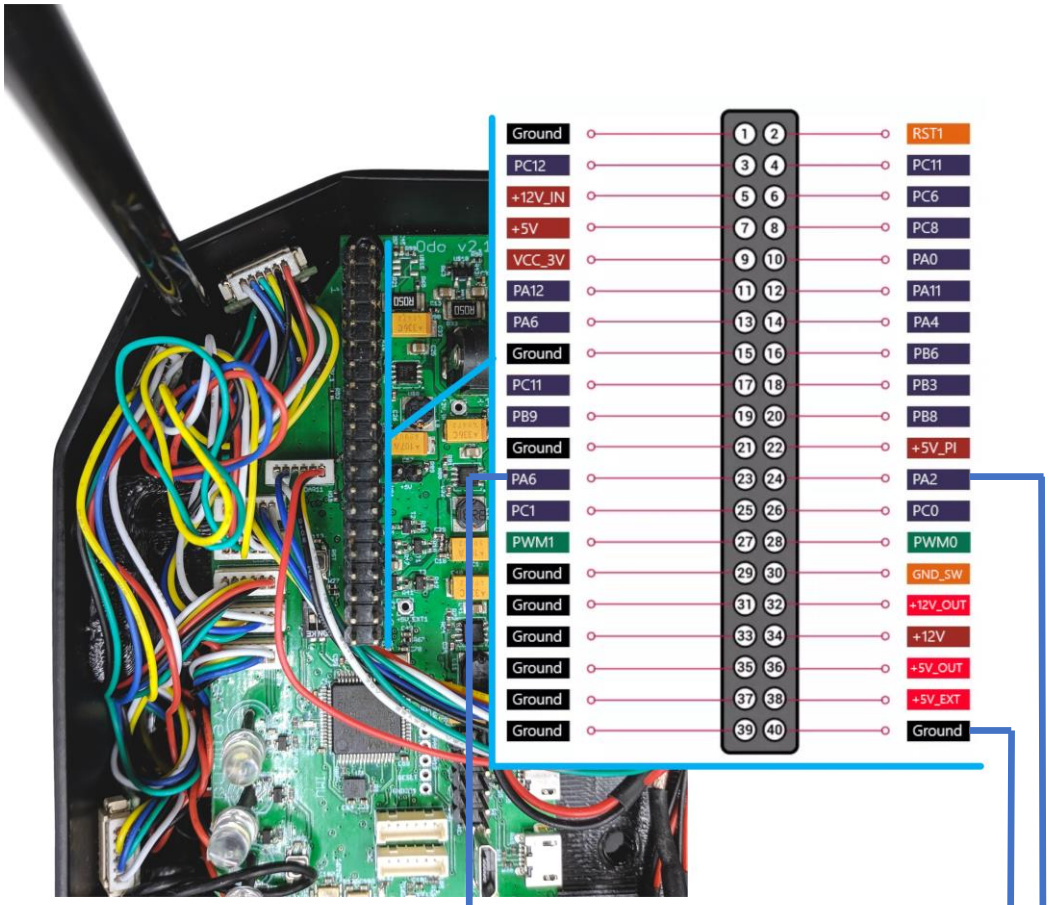
Connection settings summary:

Interface	Bitrate	Other settings
UART	4.8, 9.6, 19.2, 38.4, 57.6, 115.2, 500 Kbit/s Can be selected in dashboard	8 bits of data, 1 stop bit, no parity

Other interfaces (USB, UDP, I²C, SPI etc) to be supported in future updates.

1.1 UART interface for Odometry board of Boxie 2

Odometry board pinout:



User device:

UART TX
UART RX
GND

2. Protocols of communication via UART

2.1 Controlling the Boxie 2 (user device acts as Master)

Supported hardware:

Odometry board:	supported
Raspberry Pi:	not supported in the current SW version

If the user device needs to send control data to Boxie 2, it should send following frame:

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0x49
2	2	uint16_t	Code of data in packet	
4	1	uint8_t	Number of bytes of data transmitting	N
5	N	N bytes	Payload data	
5+N	2	uint16_t	CRC-16 (see appendix 1)	

Multibyte numbers are transmitted starting from low byte (little endian format)

Next subsections contain detailed information for certain codes of data.

If Boxie 2 processed the request, it sends a response in following format:

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0x49
2	2	uint16_t	Code of data (copied from the request)	
4	2	uint16_t	CRC-16 (see appendix 1)	

If Boxie 2 failed to process the request, it sends response in following format:

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0xc9
2	2	uint16_t	Code of data (copied from the request)	
4	1	uint8_t	Code of error (see appendix 2)	1
5	2	uint16_t	CRC-16 (see appendix 1)	

2.1.1 Setting state of the GPIO pin

Disclaimer: note that wrong voltages or currents via the pins can cause physical damage of the device. Be aware. Use with caution.

This packet is transmitted to setup state of the GPIO pin of the odometry board of Boxie 2.

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0x49
2	2	uint16_t	Code of data in packet	0x220
4	1	uint8_t	Number of bytes of data transmitting	8
5	8	8 bytes	Payload data – see details below	
13	2	uint16_t	CRC-16 (see appendix 1)	

Payload data for GPIO pin setup

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Pin index: 0 = PA4 1 = PA5 2 = PC8 (connected to Buzzer) 3 = PA0 (connected to backside LED) 4 = PA11 (connected to USB) 5 = PA12 (connected to USB) 6 = PA14 7 = PA13 8 = PC11 9 = PC1 (connected to Button 1) 10 = PB11 (Audio enable pin, not present on output pins grid. Output 0 disables audio)	
1	1	uint8_t	Pin mode: 0 – default mode (internal function) 1 – GPIO output push-pull 2 – GPIO output open-drain 3 – GPIO input 4 – DAC	
2	1	uint8_t	Pull-up: 0 – no pull-up or pull-down 1 – internal pull-up (~ 50 KOhm) 2 – internal pull-down (~ 50 KOhm)	
3	2	uint16_t	Output pin state: For pin mode 1 (GPIO output push-pull): 0 – logic 0 output (0 V) 1…65535 – logic 1 output (3.3 V) For pin mode 2 (GPIO output open-drain): 0 – logic 0 output (0 V) 1…65535 – logic 1 output (Z state) For pin mode 4 (DAC): 0…4095 - output voltage 3.3*(A/4095) V 4096…65535 – output voltage 3.3V	A

5	2	uint16_t	Timeout of returning pin to system. Pin will be returned to default mode if new command is not coming until timeout. 0 – default system timeout 1…65535 – user control timeout, seconds	
7	1	1 byte	Reserved	

Note: Some pins are internally connected to some robot hardware. If you configure it as GPIO, this function will stop working. For example, if you configure pin 4 or pin 5 as GPIO, odometry board will be not available via USB. If you configure pin 2, the buzzer will not work. Pins are configured to their default states after rebooting the robot or after configuring the pin mode to 0 (default).

Example 1: Setting PC11 to GPIO push-pull mode with output logic 1 with default timeout. Payload is highlighted green.

0xff 0x49 0x20 0x02 0x08 0x08 0x01 0x00 0x01 0x00 0x00 0x00 0x00 0x5a 0x0b

Example 2: Setting PA5 to DACl mode with output 2.34V with timeout 120 sec. Payload is highlighted green.

0xff 0x49 0x20 0x02 0x08 0x01 0x04 0x00 0x57 0x0b 0x78 0x00 0x00 0x01 0x54

3. Contacts

For additional support, please send your questions to info@marvelmind.com

Appendix 1. Calculating CRC-16

For checksum the CRC-16 is used. Last two bytes of N-bytes frame are filled with CRC-16, applied to first (N-2) bytes of frame. To check data, you can apply CRC-16 to all frame of N bytes, the result value should be zero.

Below is the implementation of the algorithm in the 'C':

```
typedef uint16_t ModbusCrc;

typedef union {
	uint16_t w;
	struct{
		uint8_t lo;
		uint8_t hi;
	} b;
	uint8_t bs[2];
} Bytes;

static ModbusCrc modbusCalcCrc(const void *buf, uint16_t length)
{
	uint8_t *arr = (uint8_t *)buf;
	Bytes crc;

	crc.w = 0xffff;

	while(length--){
		char i;
		bool odd;

		crc.b.lo ^= *arr++;
		for(i = 0; i< 8; i++){
			odd = crc.w& 0x01;
			crc.w>>= 1;
			if (odd)
				crc.w ^= 0xa001;
		}
	}
	return (ModbusCrc) crc.w;
}
```

Appendix 2. Codes of errors

Code of error in reply may be one of following:

1 – unknown type of packet in request
2 – unknown code of data in request
3 – error in data field of request
6 – device is busy