

# Microcontroller Unit (MCU)

## Block Data Sheet

MCU 080115

### SPECIFICATIONS

- > **Sampling Rate:** 1, 10, 100, or 1000Hz
- > **Analog Ports:** 6 in (A1-A6)
- > **Digital Ports:** 4 in (I1-I4) + 4 out (O1-O4)
- > **Resolution:** 10-bit (A1-A4) + 6-bit (A5&A6)
- > **Communication:** UART
- > **Operating Voltage:** 3.3V
- > **Consumption:** ~5mA

### FEATURES

- > Real-time data sampling and streaming
- > High performance open-source firmware
- > Crystal oscillator for maximum precision
- > Status and low-battery indicators
- > Cross-talk cancelation
- > Raw data sampling and output
- > Plug & play operation
- > Easy-to-use OpenSignals software & APIs

### APPLICATIONS

- > Biosignal research
- > Rapid prototyping of custom hardware
- > Real-time data visualization
- > Real-time data recording

### GENERAL DESCRIPTION

Our MCU block is designed for accurate and reliable real-time data streaming over any UART-compatible interface (e.g. Bluetooth, FTDI). Its high performance firmware, made available in open source, can acquire and control up to 6 analog inputs, 4 digital inputs, and 4 digital outputs at up to 1kHz. Together with the OpenSignals software and a wide array of programming APIs, our MCU enables hassle-free access to the raw data, making it ideal for rapid prototyping of cross-platform real-time applications. Here are a few examples:

<https://www.youtube.com/watch?v=pVAaFeym8TQ>  
<https://www.youtube.com/watch?v=hQ3BUBV-BBM>

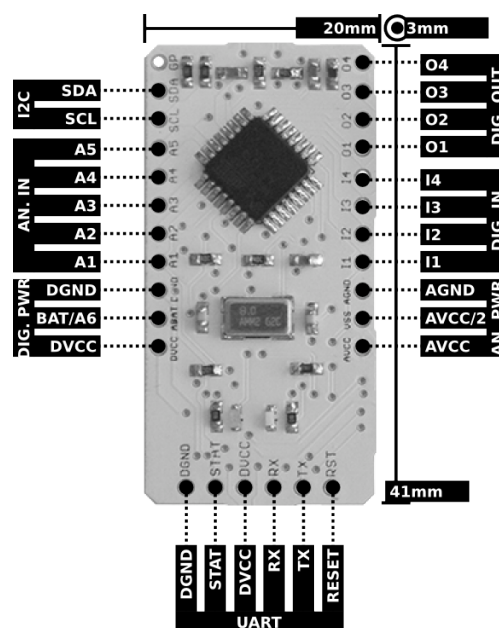


Fig. 1. Pin-out and physical dimensions.

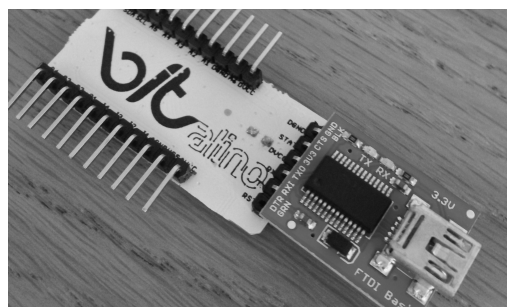


Fig. 2. Our MCU can be used with any UART-compatible accessory (RST pin may need to be left unconnected).

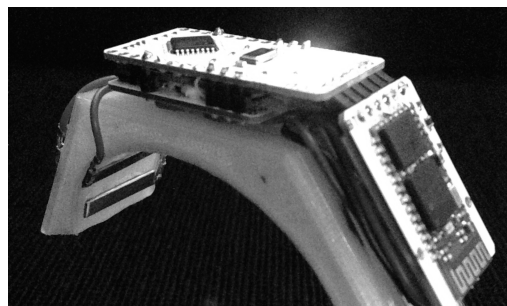


Fig. 3. Prototype wearable device using the MCU and Bluetooth blocks for real-time wireless data streaming.

# bitalino

PLUX – Wireless Biosignals, S.A.  
Av. 5 de Outubro, n. 70 – 8.  
1050-059 Lisbon, Portugal  
bitalino@plux.info  
<http://bitalino.com/>

REV A

© 2015 PLUX

This information is provided "as is," and we make no express or implied warranties whatsoever with respect to functionality, operability, use, fitness for a particular purpose, or infringement of rights. We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost revenues, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted, even if advised of the possibility of such damages.



BEWARE: DIRECT OR INDIRECT COUPLING TO THE MAINS MAY RESULT IN SHOCKING HAZARD



# Microcontroller Unit (MCU)

## Block Data Sheet

---

### STATE COMMANDS

The BITalino firmware is prepared to receive a set of commands that can be used to control the state of the device (Fig. 4.):

- > **Idle:** Corresponds to the state in which the device is in standby (status LED fades at 0.5Hz); it's the default state when the device is turned on.
- > **Live:** Corresponds to the state in which the device is acquiring and streaming data from all the input channels in real time (LED blinks at 1Hz); setting the channel mask bits (labeled A0-A5) to 0 or 1 deactivates / activates the real-time streaming of the corresponding channel A1-A6 from the device.
- > **Simulated:** This is a facility provided for developers, and in this state the device streams synthetic data generated by the firmware (sine, saw tooth, and square waves, together with a pre-recorded ECG time series); this enables the development and testing of the software layer at the receiving end using the actual physical and transport layers, but without requiring actual sensors to be connected to a user to have values changing on the streamed data.

0	1	2	3	4	5	6	7	
0	0	0	0	0	0	0	0	idle mode
1	0	A0	A1	A2	A3	A4	A5	live mode with analog channel selection
0	1	A0	A1	A2	A3	A4	A5	simulated mode with analog channel selection

Fig. 4. State commands

### ACTION COMMANDS

The BITalino firmware is also prepared to receive a set of action commands (Fig. 5):

- > **Sampling Rate:** While in idle, the sampling rate can be selected by setting the bits labeled Fs to 00 (1Hz), 01 (10Hz), 10 (100Hz), or 11 (1000Hz).
- > **Trigger Digital Outputs:** While in live mode, the digital output pins D1-D4 can be activated / deactivated by setting the pins labeled Do0-Do4 to 1 or 0.
- > **Set Battery Threshold:** By default, the firmware is prepared to control the low battery LED based on the A6 analog input readings (if the ABAT pin from the PWR block is connected to A6 monitors the battery level); the battery threshold can be set to any 6-bit value (ranging from 0-63), in which values closer to 0 trigger the low battery indicator for battery voltages bellow 3.4V (5-10% charge), and values closer to 63 trigger the low battery indicator for battery voltages bellow 3.8V (90-95% charge).

0	1	2	3	4	5	6	7	
0	0	threshold						battery threshold definition
1	1	Do0	Do1	Do2	Do3	-	-	set digital output (when in live mode)
1	1	-	-	-	-	Fs		sampling rate definition (when in idle mode)

Fig. 5. Action commands

# Microcontroller Unit (MCU)

## Block Data Sheet

---

### DATA PACKETS

While in live or simulated mode, the BITalino firmware streams the acquired data in real time, formatted as structured sequences of bits corresponding to:

- > **S**: 4-bit sequential number generated by the firmware to identify the packet, which can be used on the receiver to detect loss of packets.
- > **CRC**: 4-bit Cyclic Redundancy Check code, useful for the evaluation of the data packet consistency on the receiver.
- > **D0-D3**: State of the digital input ports D1-D4 on the device.
- > **A0-A5**: Digital code produced by the ADC for the voltage at the corresponding analog input ports A1-A6; the first four channels have 10-bit resolution (ranging from 0-1024), while the last two have 6-bit (ranging from 0-63).

		bits							
		0	1	2	3	4	5	6	7
bytes	0	A5	A5	A5	A5	A5	A5	A4	A4
	1	A4	A4	A4	A4	A3	A3	A3	A3
	2	A3	A3	A3	A3	A3	A3	A2	A2
	3	A2	A2	A2	A2	A2	A2	A2	A2
	4	A1	A1	A1	A1	A1	A1	A1	A1
	5	A1	A1	A0	A0	A0	A0	A0	A0
	6	A0	A0	A0	A0	D3	D2	D1	D0
	7	CRC	CRC	CRC	CRC	S	S	S	S

Fig. 6. Data packets structure