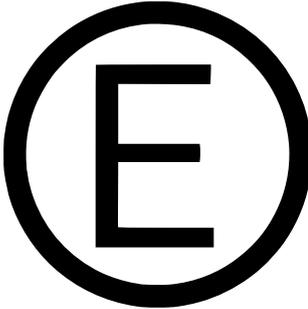

**Barsotion EA & TA:
User manual**



PRELIMINARY EDITION

Table of contents

- Overview..... 4
- EA hardware description..... 5
 - EA Concepts..... 5
 - EA Realization..... 7
 - EA Dimentions..... 8
 - EA Structural schematic..... 9
 - EA Topology..... 10
 - EA Errata..... 12
 - EA_ERR1_Anode..... 12
 - EA_ERR2_Poweroff..... 13
 - EA_ERR3_Servo_supply..... 14
 - EA GitHub repository..... 15
- TA hardware description..... 16
 - TA Concepts..... 16
 - TA Realization..... 17
 - TA v1.6..... 18
 - TA Dimentions..... 19
 - TA Structural schematic..... 20
 - TA Topology..... 21
 - TA v1.6 Topology..... 23
 - TA GitHub repository..... 25
- EA & TA placing..... 26
- EA & TA software packet..... 27
 - Abstract..... 27
 - Device connection for flashing & communication..... 27
 - Commands summary..... 27
 - Servo control commands..... 29
 - SxP & SxM commands..... 29
 - SZERO command..... 29
 - SPRINT command..... 29
 - I2C control commands..... 29
 - TWI-VERIFY command..... 29
 - TWI-RESET command..... 30
 - BMP388-ID command..... 30
 - AH3-ID command..... 30
 - Disk control commands..... 31
 - DISK-INFO command..... 31
 - DISK-ERASE command..... 31

PACKET-WROTE command..... 31

Get/set parameters commands..... 31

 BOOT-ENTIRE-READ command..... 31

 PACKET-LEN command..... 32

 IMU-CYCLE-FREQ command..... 32

 DEVICE command..... 32

 SET-BOOT-ENTIRE command..... 33

 SET-IMU-CYCLE-FREQ command..... 33

IMU commands..... 34

 IMU-CALIB command..... 34

 IMU-CALIB-COEFS command..... 34

System commands..... 34

 POWER-OFF command..... 34

 RESTART command..... 34

 ACTION command..... 34

Helper commands..... 34

 HELP command..... 34

Debug commands..... 35

 PAGE-CONTENTS command..... 35

 TEST-ACTION command..... 35

Machine mode commands..... 35

 MM-PACKET-LEN command..... 35

 MM-PACKET-WROTE command..... 35

 MM-DISK-READ command..... 35

EA & TA PC interface..... 37

 GBK Oracle utility..... 37

 GBK Oracle usage..... 37

 GBK Oracle commands..... 37

 Log decoder utility..... 39

 Log decoder usage..... 39

Revision history..... 40

Contacts..... 41

Overview

The EA & TA board computers as a part of Barsotio xAnomalain series are the devices created to control small model rockets flight and write down flight parameters. It was developed for the OKB “Iva” Pike rocket project.

These computers are the logical continuation of Barsotio GBK DeltaAnomalain board computer, used in OKB “Iva” Kilka project.

For the EA & TA developing, some radical principles were adopted:

- As possible new components.
- Manufacturability and simply usage.
- Aesthetics and sophistication.

“By the Will of God and according to the plan”



Figure 1: Pike rocket explained view

EA hardware description

The Barsotion GBK EpsilonAnomalain boards (Barsotion-EA) was created as a part of model rocket project Pike.

EA Concepts

The Pike rocket has the following view:



Figure 2: Pike rocket view

The EA board computer (GBK) should be located at the nose of rocket, consist of one board and placed longitudinally. The power source is the 3.7 V 18650 battery with typical capacity around 3000 mA*h. The board computer should contain following sensors:

- Gyroscope & accelerometer (ICM-42688-P was chosen), connected via SPI, that is important because of necessity of ensuring high data read speed;
- Barometer (BMP388 was chosen);
- AH3 air speed sensor connector (4-pin, 3,3V, GND, SDA, SCL);
- Hygrometer (AHT20 was chosen).

The microcontroller should be enough rapid for processing gyroscope's data a few thousand times per second. Reading data from the gyroscope should be synchronized with gyroscope's interrupt channel.

Barometer and hygrometer data reading interrupt synchronization is not needed.

The board computer should contain a flash memory chip, the capacity should not be less than 1 Gbit.

The board computer should have connectors for four MG90S servos. Also it should have a DC-DC boost voltage converter to provide servo feeding. The board

computer should can power down servos if they are not used, because servos have a significant static power consumption.

The board computer should can measure battery voltage and have a power-off option, that allows to power down system by the microcontroller command.

The board computer should switch on by special button pressing. It is supposed that the computer will be connected to the battery a long time before the start so that long working not wanted. The computer should be switched on shortly before the start for saving the battery charge.

The computer should have a small seven-segment led indicator on the board for system status indication and also 4 LEDs:

- Power 3.3 V indication LED;
- TXD0 LED;
- RXD0 LED;
- User-driven LED.

The power indication helps to control the computer is working or not, TXD and RXD indication helps to control the data transmission progress. User-driven LED is needed for simplifying the debug process and improving the computer using user's experience.

Also the board computer should have a buzzer that needed to signal when the rocked landed and waits to be found.

EA Realization

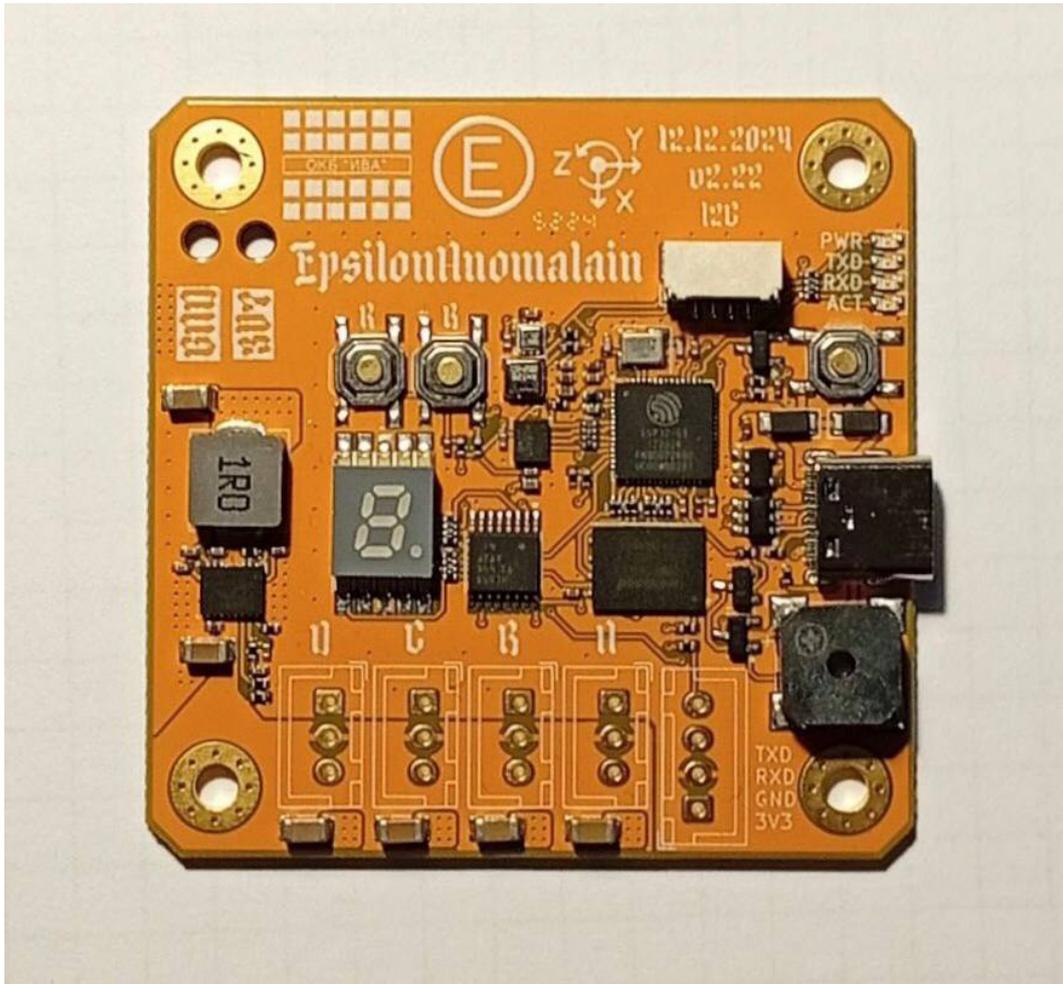


Figure 3: EA explained view

The Barsotion-EA has following characteristics:

- Microcontroller: ESP32-S3FN8;
- Gyroscope & accelerometer: ICM-42688-P;
- Barometer: BMP388;
- Hygrometer: AHT20;
- AH3 board I2C connector;
- Buzzer for sound indication;
- Seven-segment LED indicator driven by 74HC595 shift register;

- USB-C for flashing & debug;
- SPI NAND Flash memory chip W25N01GVZEIG, the capacity is 1 Gbit;
- DC-DC boost voltage converter based on TPS61088 for servo feeding, the schematic is calculated and topology drawn for continuous 3 A current;
- Four servo connectors.

EA Dimentions

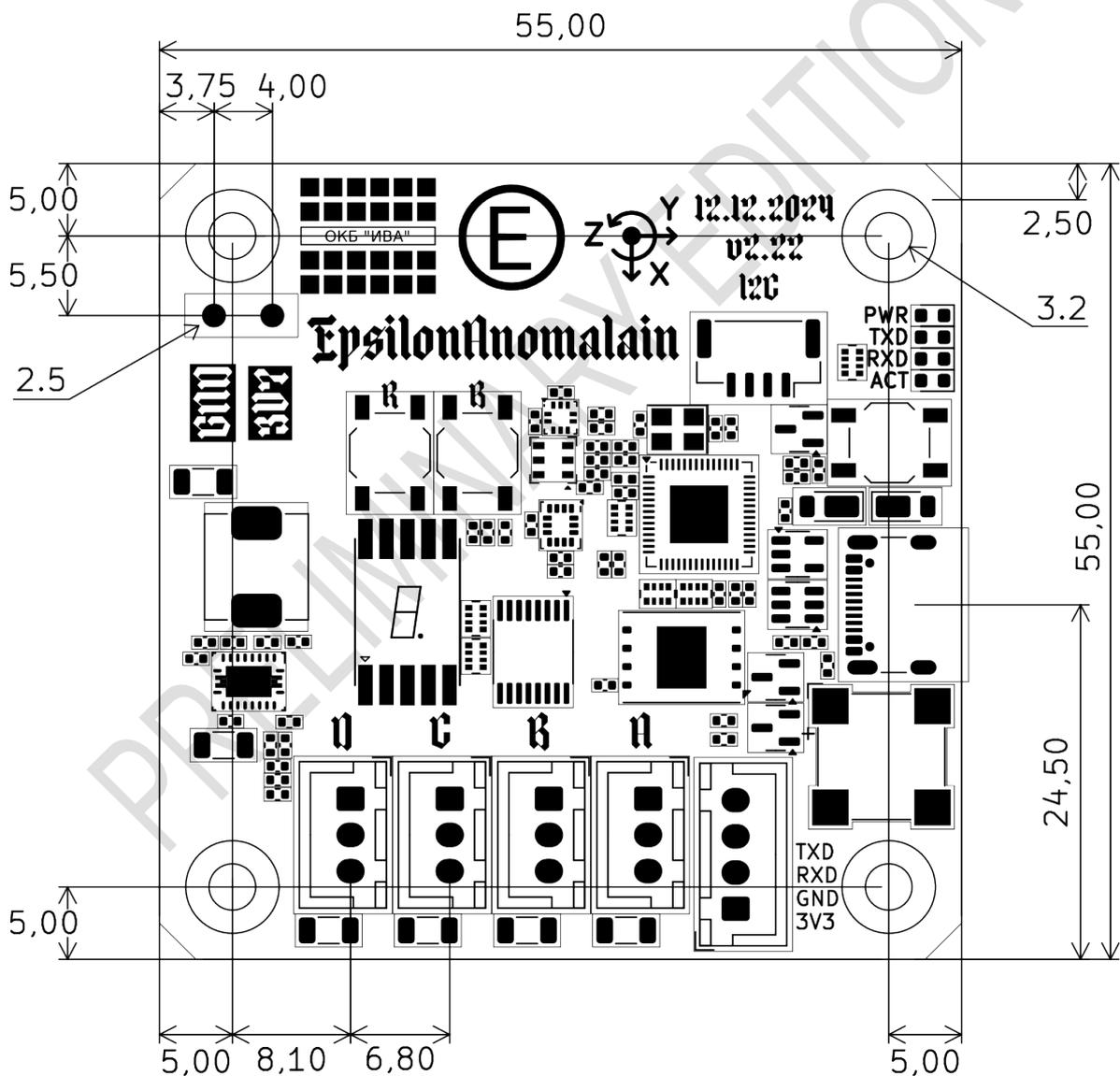


Figure 4: EA dimentions

EA Structural schematic

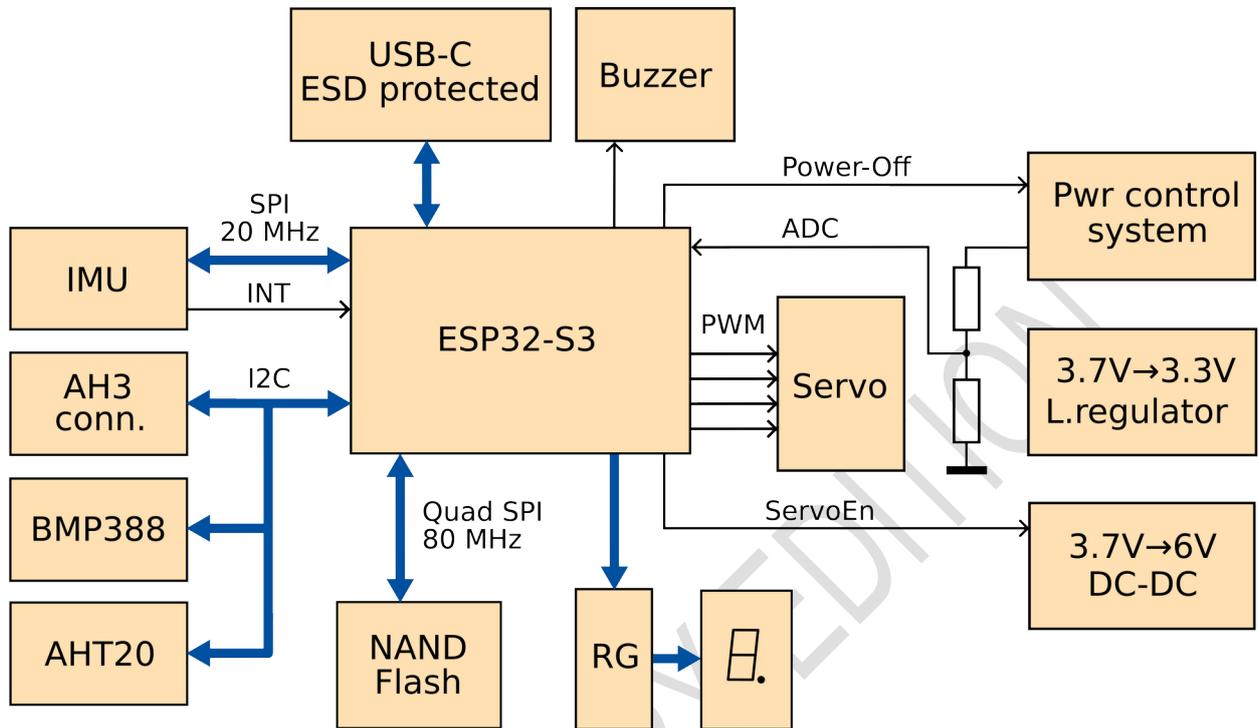


Figure 5: EA structural schematic

PRELIMINARY

EA Topology

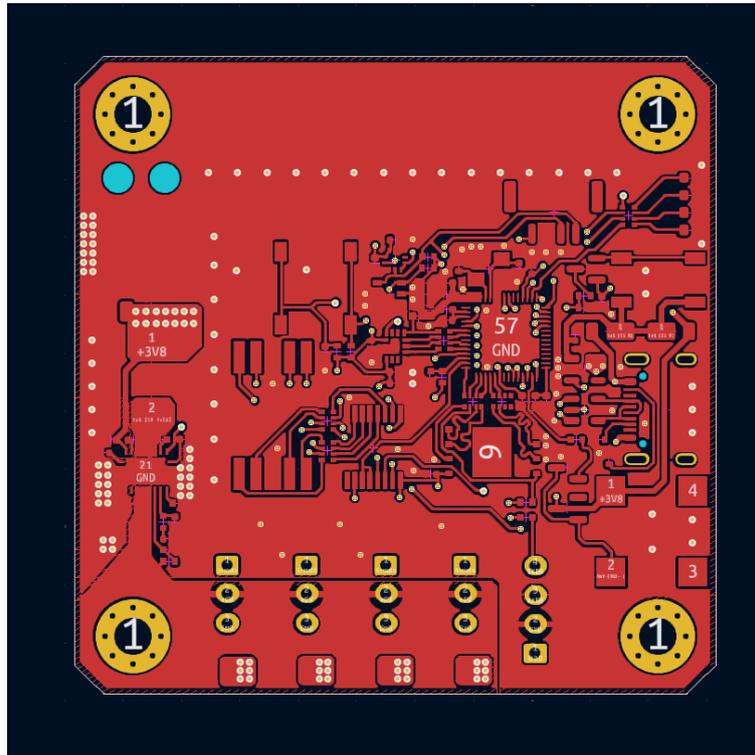


Figure 6: EA Layer 1 - Top signal

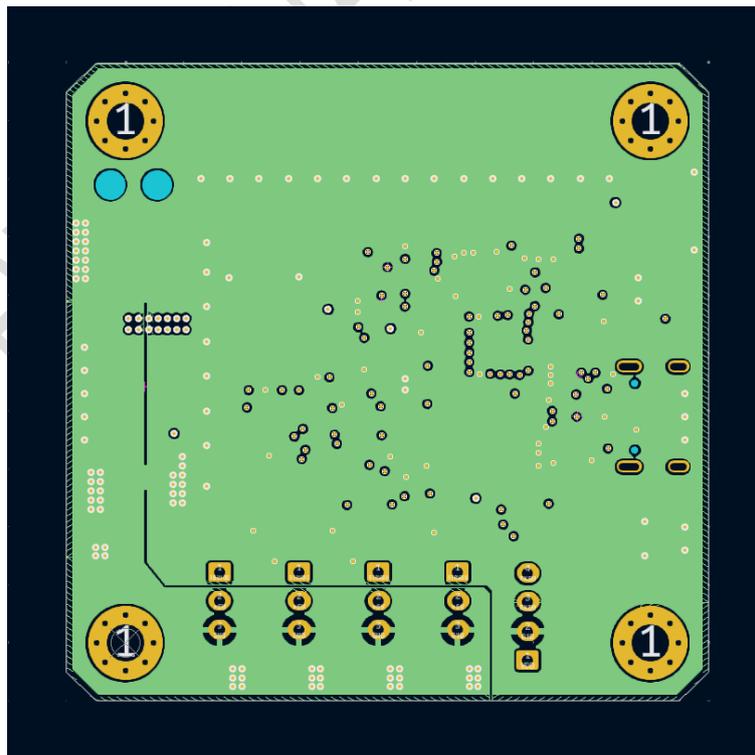


Figure 7: EA Layer 2 - GND

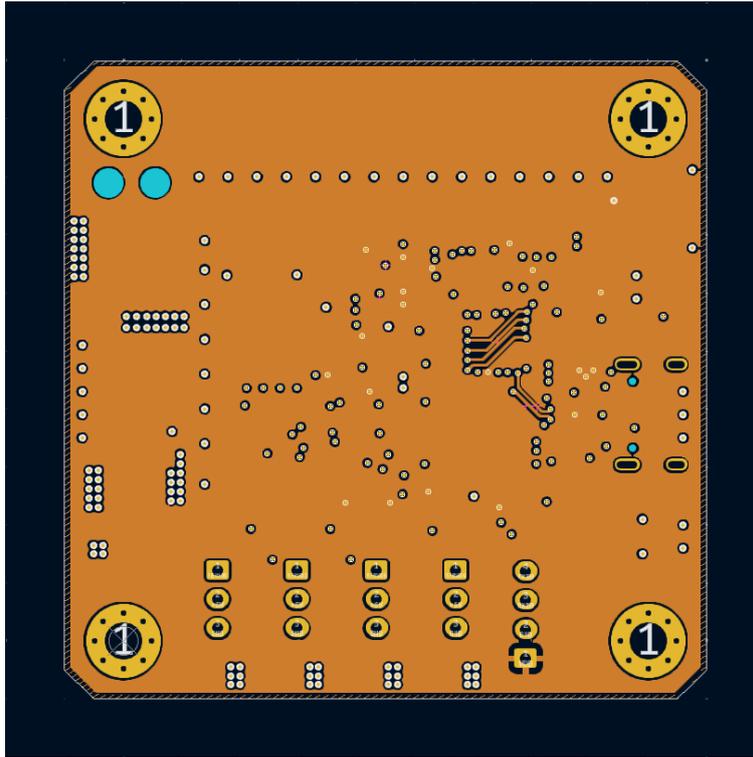


Figure 8: EA Layer 3 - Power

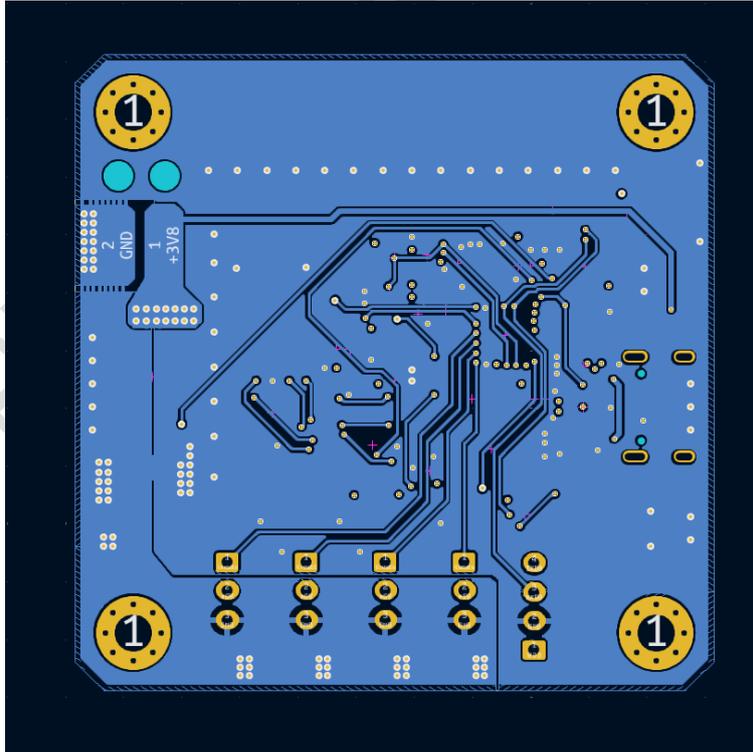


Figure 9: EA Layer 4 - Bottom signal

EA Errata

While the EA tests some errors was finded.

EA_ERR1_Anode

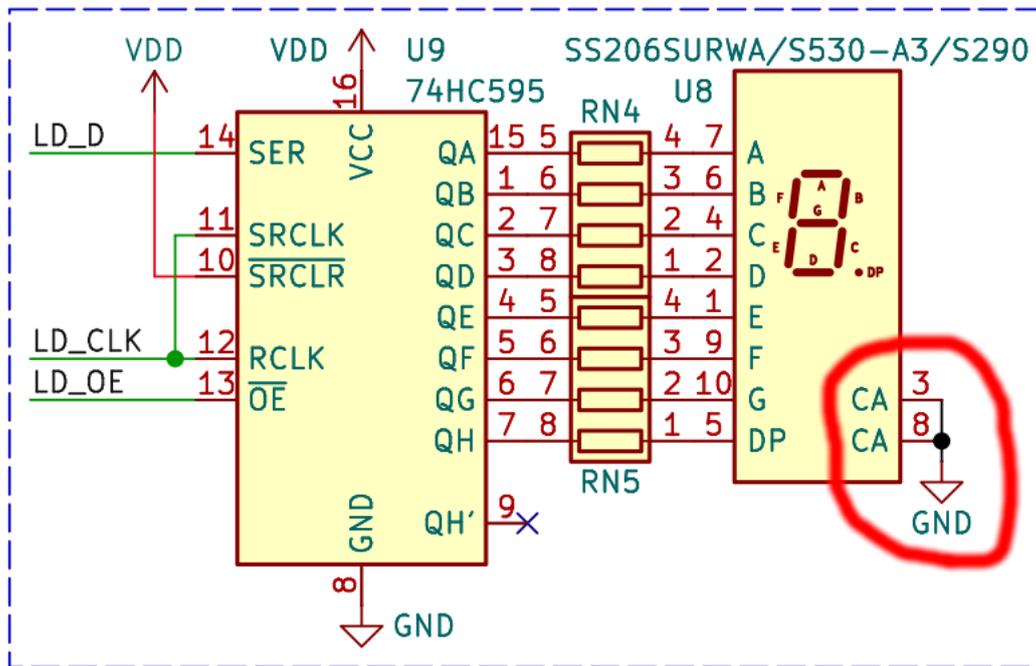


Figure 10: EA_ERR1_Anode

The LED module's common anodes should be routed to +3.3 V power supply, however they are wired to the ground. That is a reason why the led indicator on the EA does not work.

EA_ERR2_Poweroff

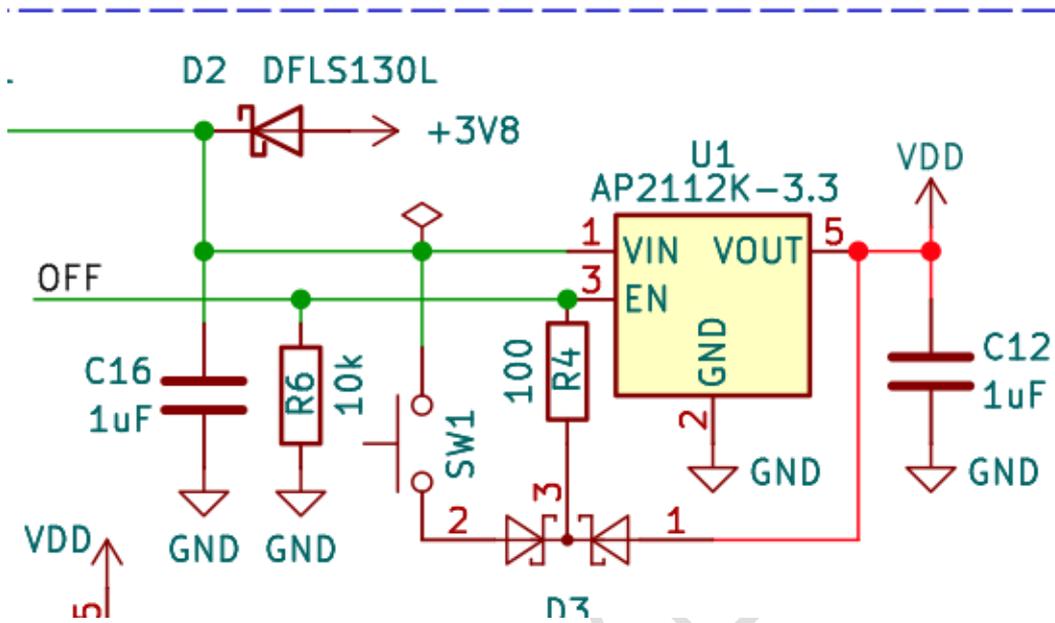


Figure 11: EA_ERR2_Poweroff

The EA boards have a hardware ability to power-off themselves by microcontroller command. It is useful if the battery voltage is low.

The power-off function does not work properly. When the OFF pin (GPIO38) is initialized as output and the logic 0 routed to this pin, the voltage on the VDD line decrease down to 1.8-1.9 V level, after that the GPIO block in the microcontroller stops the working, the OFF pin goes to tri-state condition, VDD capacitors' residual charge provides a high level on the regulator's EN pin and the device restarts.

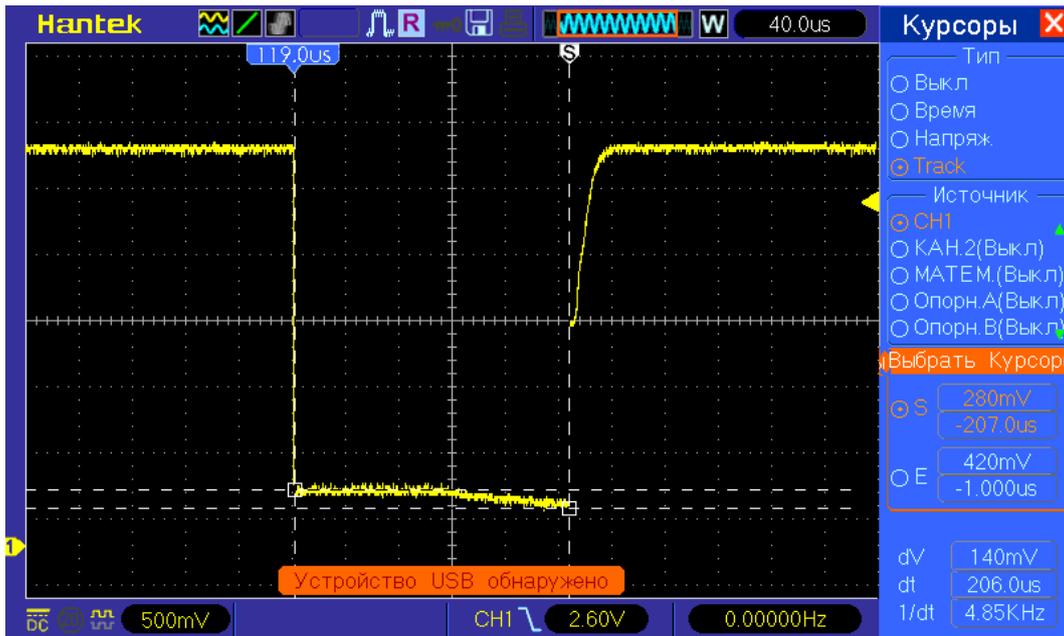


Figure 12: OFF line after POWER-OFF command receiving

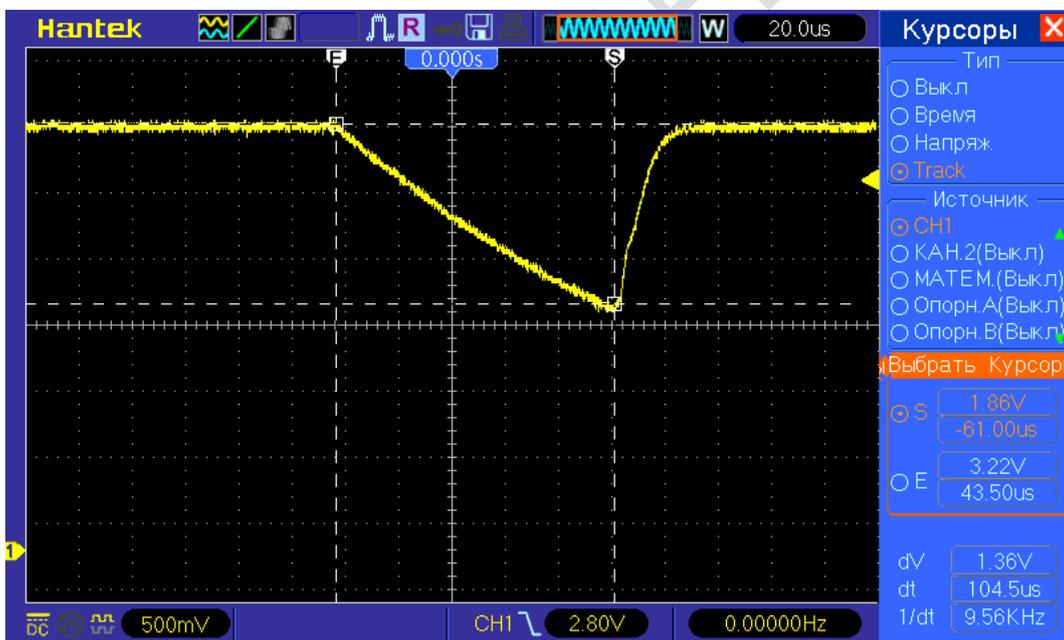


Figure 13: VDD line after POWER-OFF command receiving

EA_ERR3_Servo_supply

This error caused from mistake of understanding the TPS61088 EN (Enable) pin function.

The SERVOEN line is wired directly from microcontroller to TPS61088's EN pin. While the EA developing it means that if the EN pin is not active, there is no voltage on the 6 V output servo feeding line. However the EN pin only enables voltage conversion, not current passing by the converter IC.

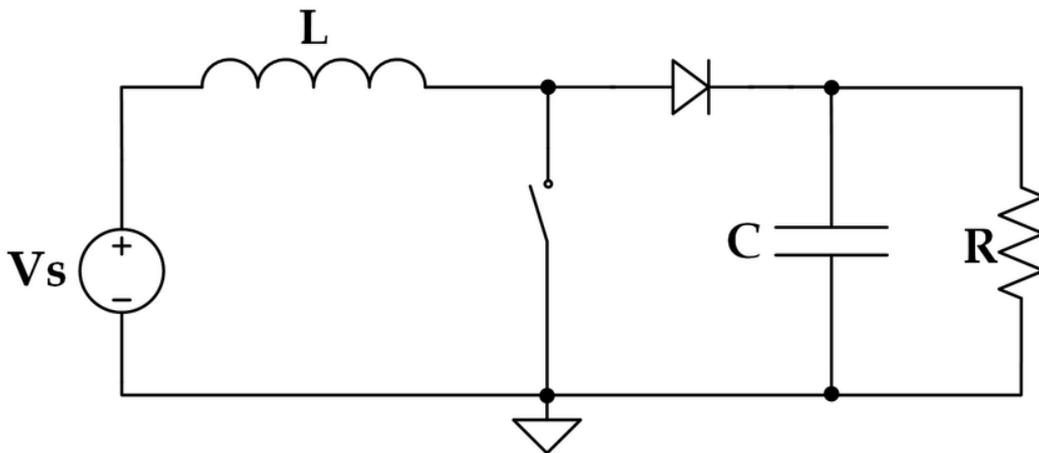


Figure 14: Simplified boost converter circuit

EN pin enables driving the switch. If it is not active, switch is always unlocked, so the converter does not work. However the voltage still can pass through the inductor and diode to the servos. The solving of the problem is to add the transistor to the servo feeding circuit that can break the circuit if it is needed.

EA GitHub repository

Barsotion-EA's KiCad sources, full schematic, pindefs-file are located at a [project GitHub repository](#).

TA hardware description

TA Concepts

The main goal of the TA creation is to fix the EA schematic error and problems. The most of device requirements are the same as for EA. However some extra features was wanted:

- Add the ICM-42688-P ODR frequency quartz stabilization
- Add 2.4 GHz WiFi / Bluetooth / BLE antenna
- Use 4 Gbit Flash chip instead of 1 Gbit
- Change the buzzer connection for allowing the buzzer to work when the board connected to the computer via USB (on the EA board the buzzer can make sound only if the battery connected)

According the EA&TA software architecture, the core 1's cycle time depends on the ICM-42688-P INT1 pin period that is the same as the maximum of accelerometer & gyroscope's ODRs. By default, the ODR frequency is generated from internal RC-source with 1% accuracy. Usual way to get the cycle time is considered in subtraction previous system timer's value from actual. The ESP32-S3's system timer has 1-microsecond accuracy, that does not allows to use ODR 16 kHz or 32 kHz because their periods have non-integer values in microseconds (62.5 & 31.25 microseconds), that causes bigger integral errors while integrating (the tests show that 8 kHz ODR gives smaller integral errors than 16 kHz, which contradicts the theory of numerical methods). The solving of the problem can be in ODR frequency quartz stabilization. If it is used, the ODR frequency has 10 ppm accuracy, so allows to use a constant as the cycle time value.

The 2.4 GHz antenna addition is not needed for rocket project tasks, however this requirement was added for getting more experience in high-frequency circuits topology design.

After the EA tests a curious fact was found: there are W25N01 memory chip pin-to-pin compatible chips with more capacity, for example W25N04. So using memory with more capacity is wanted.

TA Realization

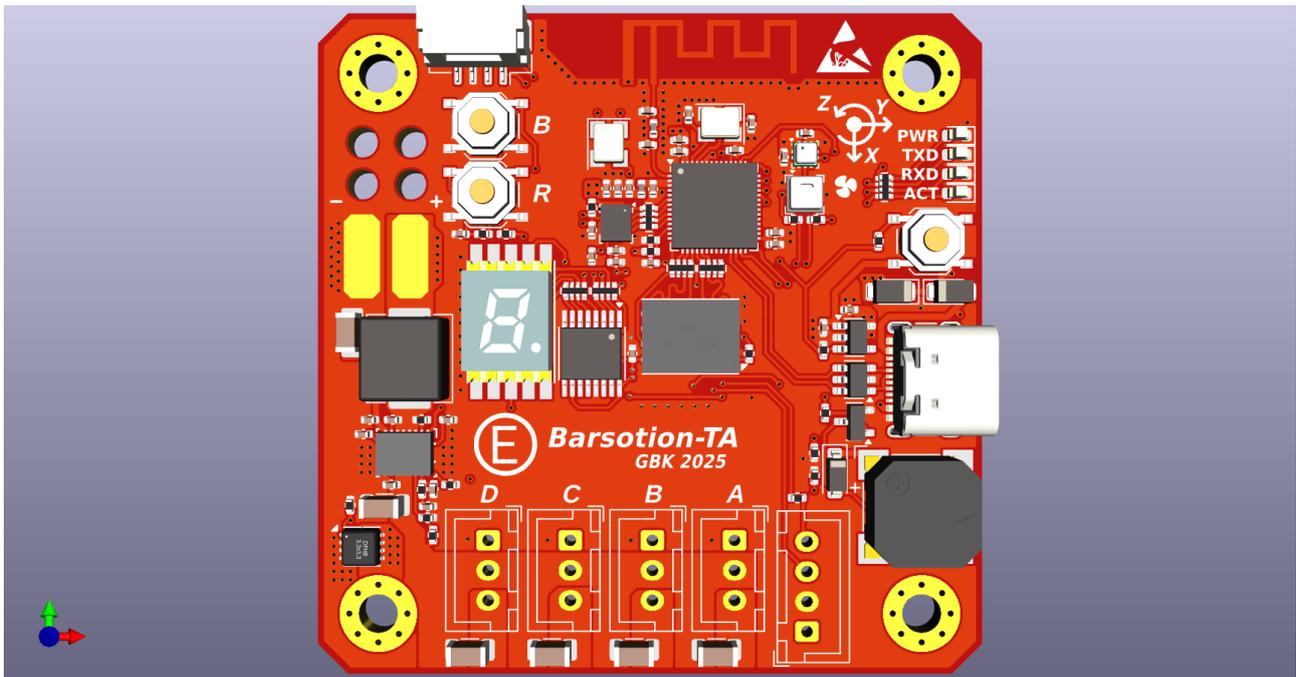


Figure 15: TA explained view

Thus, the Barsotion-EA has following characteristics:

- Microcontroller: ESP32-S3FN8;
- 2.4 GHz Wi-Fi / Bluetooth / BLE antenna driven by ESP chip;
- Gyroscope & accelerometer: ICM-42688-P (quartz-stabilized ODR frequency);
- Barometer: BMP388;
- Hygrometer: AHT20;
- AH3 board I2C connector;
- Buzzer for sound indication;
- Seven-segment LED indicator driven by 74HC595 shift register;
- USB-C for flashing & debug;
- SPI NAND Flash memory chip W25N04KVZEIG, the capacity is 4 Gbit
- DC-DC boost voltage converter based on TPS61088 for servo feeding, the schematic is calculated and topology drawn for continuous 3 A current;

- Servo powerdown ability thanks to transistor breaking the feeding line;
- Four servo connectors.

TA v1.6

Because of BOM ambiguous interpretation a wrong buzzer was ordered, PKLCS1212E2000-R1 instead of CMT-8504-100-SMT-TR. The producing company agent told that they can deliver right buzzer, however CMT-8504-100-SMT-TR delivering takes two weeks that is much. Also, a schematic error was found in TA boards: wrong transistor Q2 connection caused by incorrect package pinout.

It was decided to re-release the TA boards. A schematic, topology, Gerber, BOM, PNP was fixed and adopted for PKLCS1212E2000-R1 usage. New TA version was indexed as “TA v1.6” or “TA v1.6”.

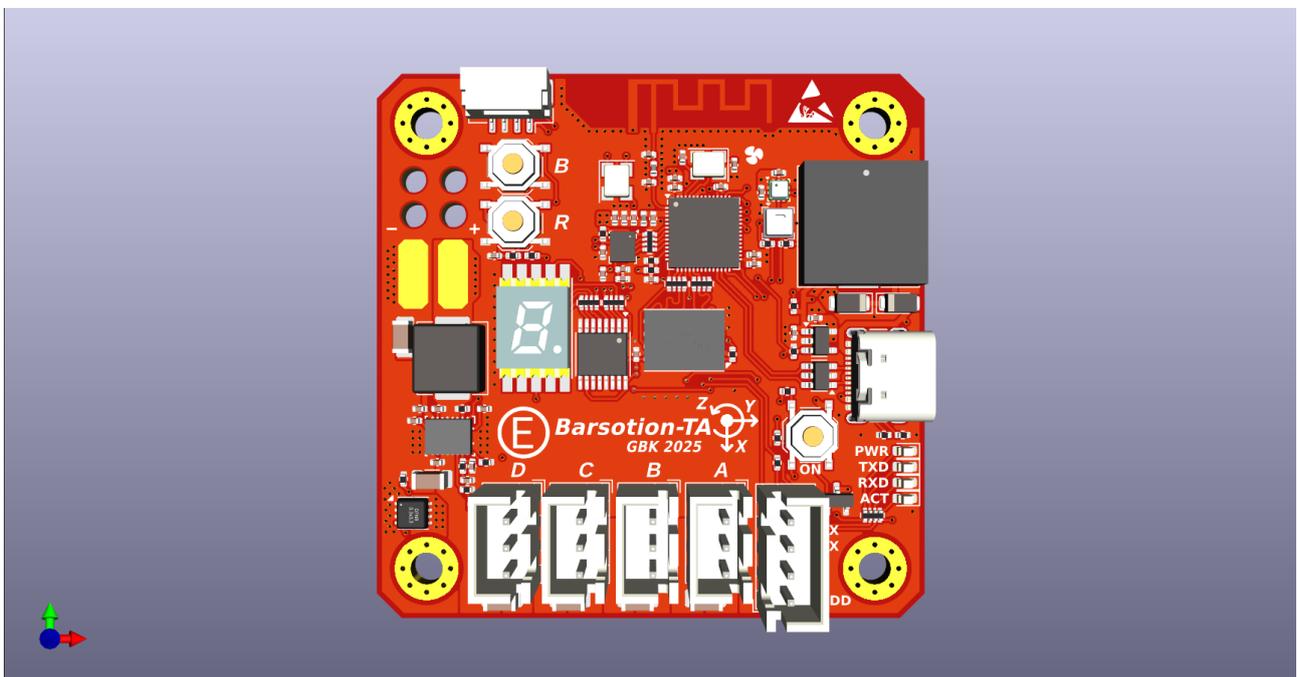


Figure 16: TA v1.6 explained view

TA Dimensions

Dimensions are common for TA & TA v1.6.

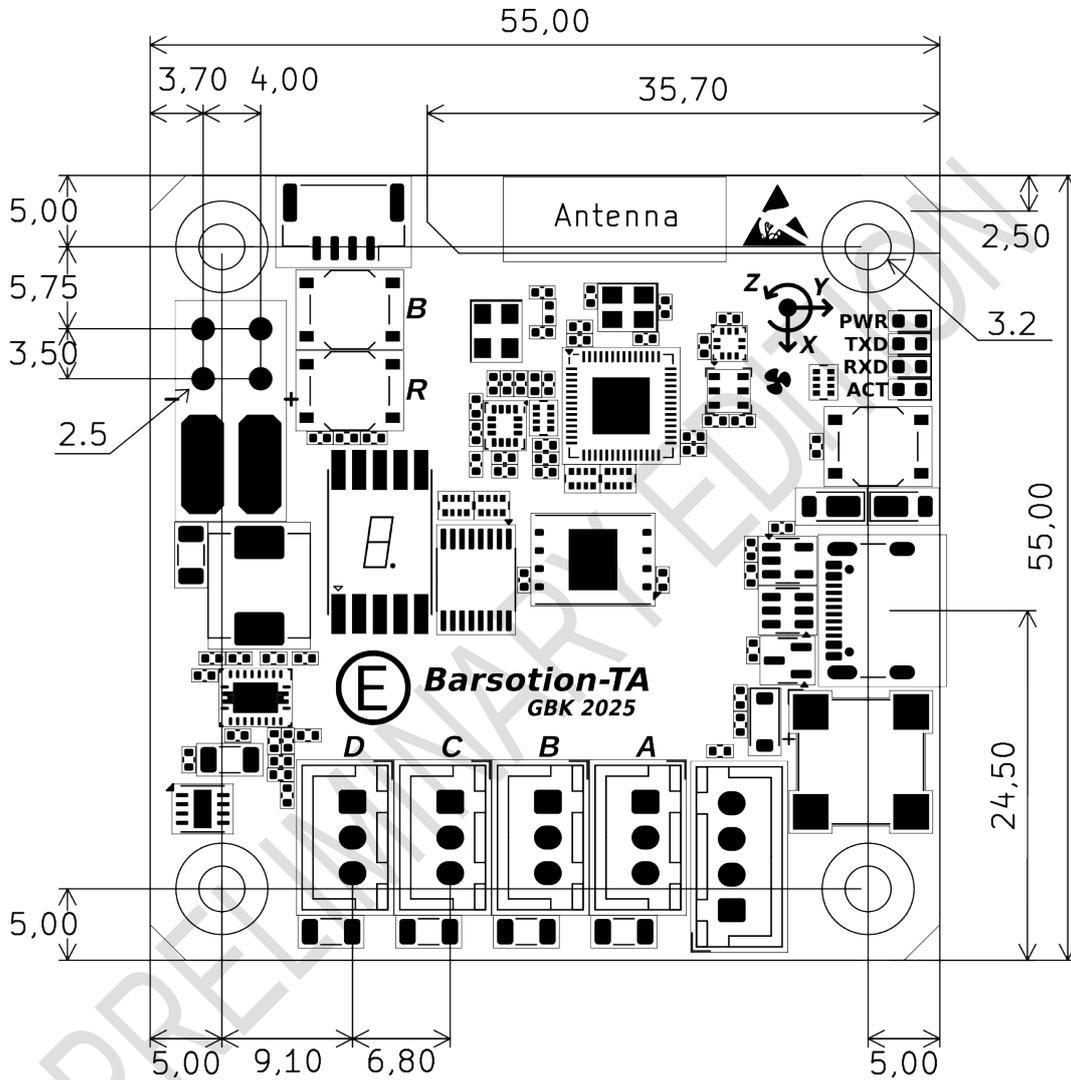


Figure 17: TA dimensions

TA Structural schematic

The TA structural schematic is the same as for EA, but the 32.768 kHz quartz oscillator was added.

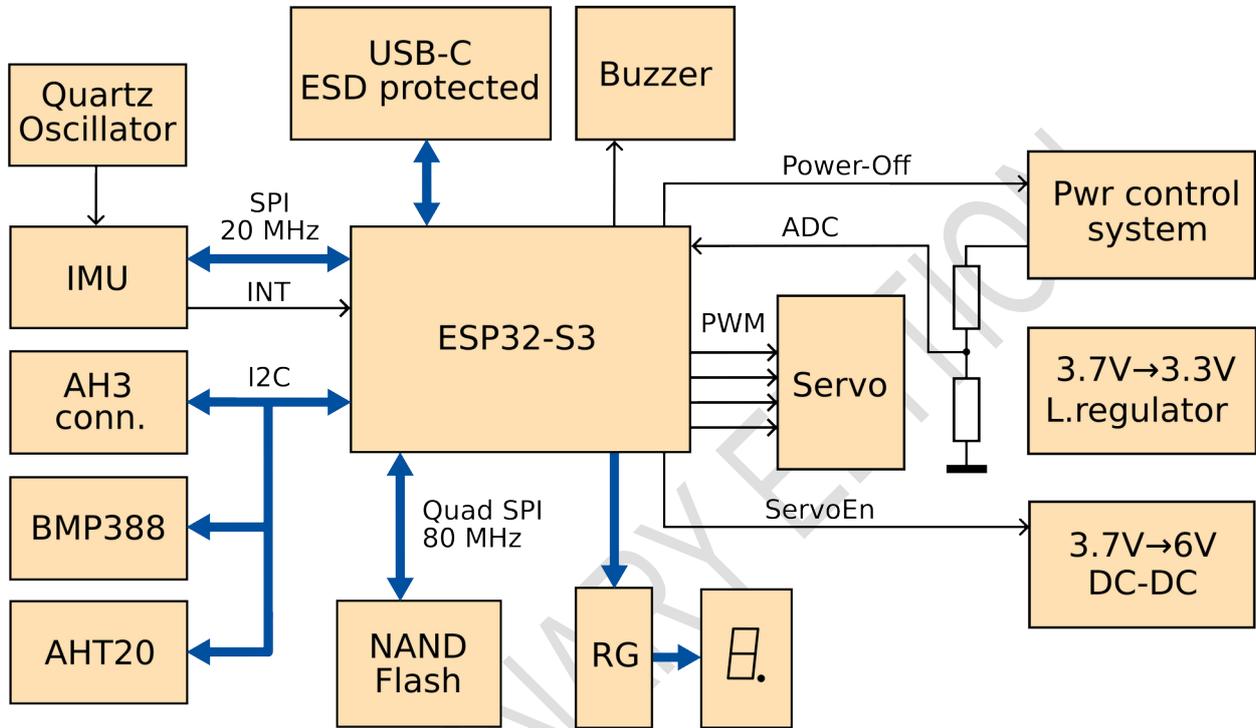


Figure 18: TA structural schematic

TA Topology

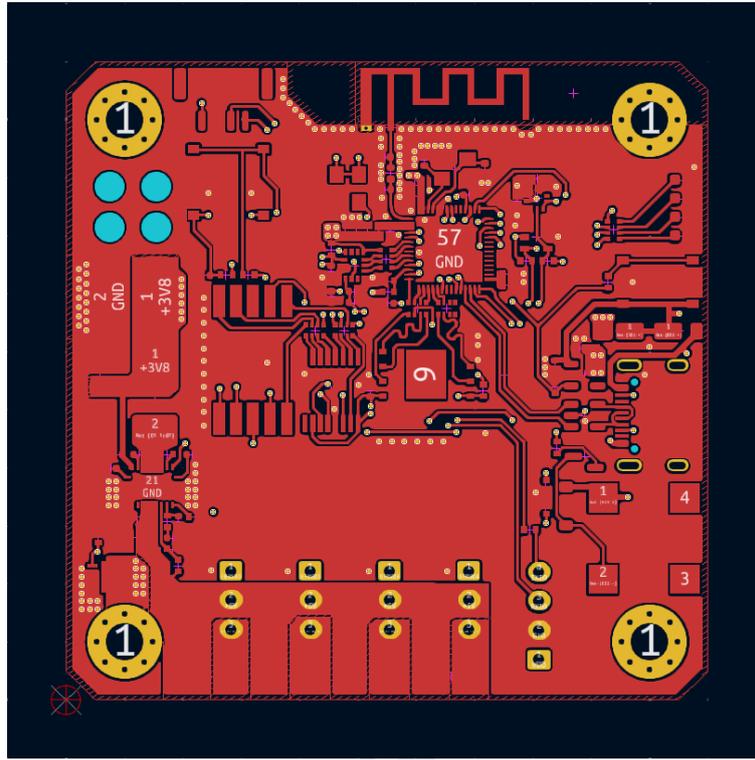


Figure 19: TA Layer 1 - Top signal

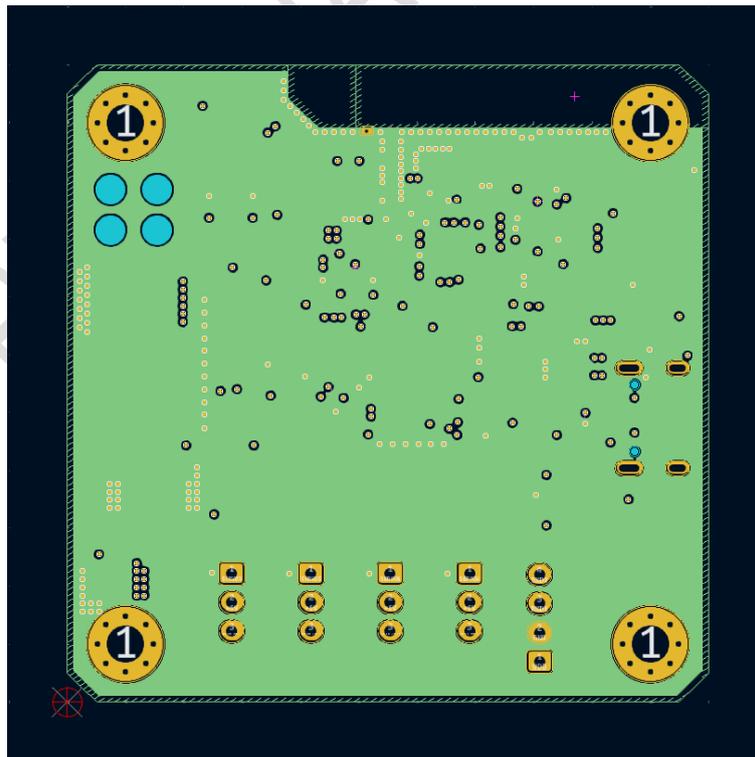


Figure 20: TA Layer 2 - GND

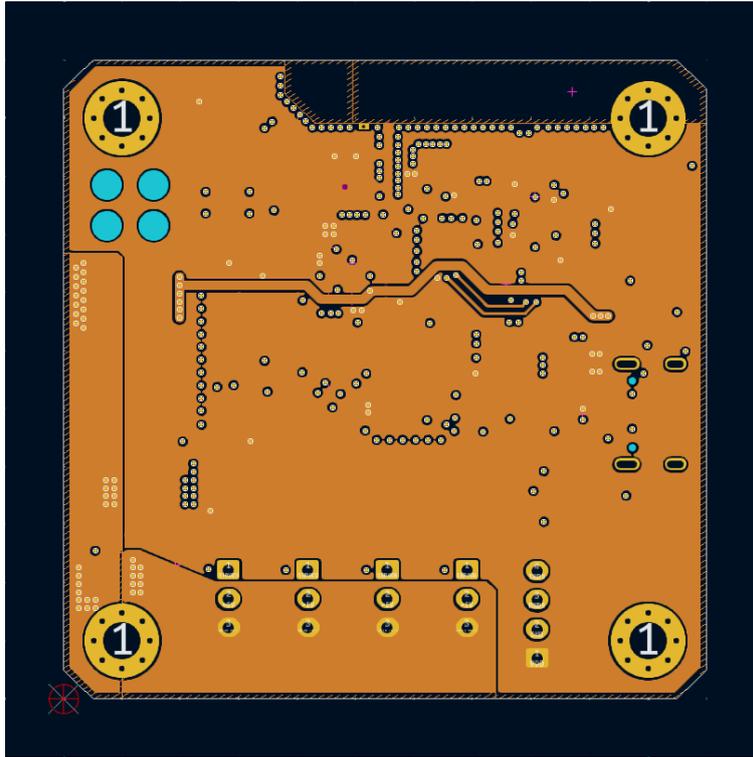


Figure 21: TA Layer 3 - Power

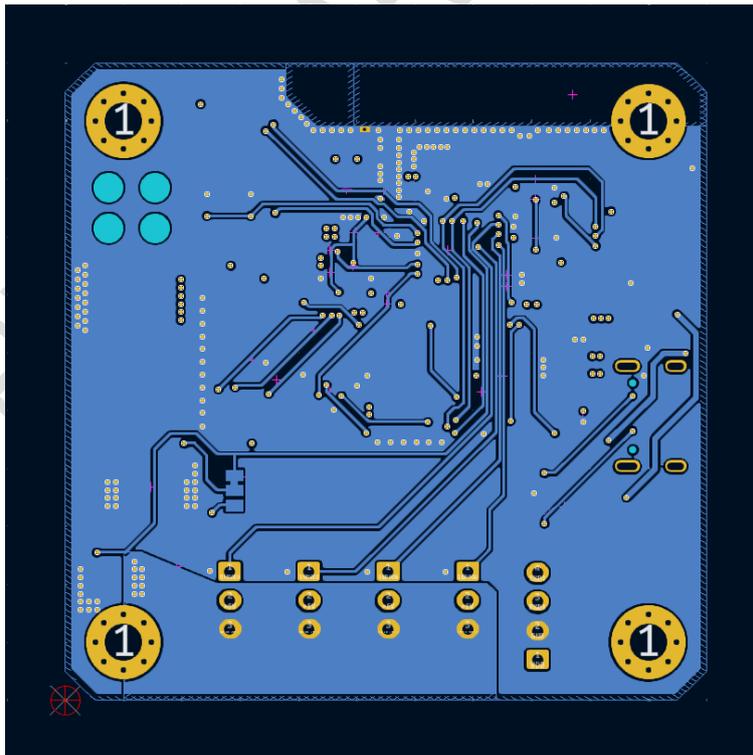


Figure 22: TA Layer 4 - Bottom signal

TAv1.6 Topology

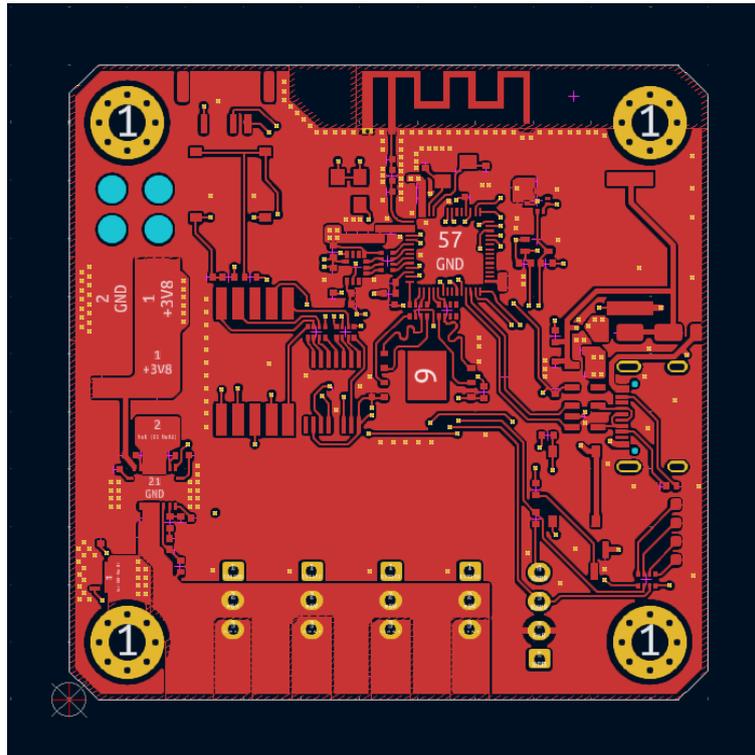


Figure 23: TAv1.6 Layer 1 - Top signal

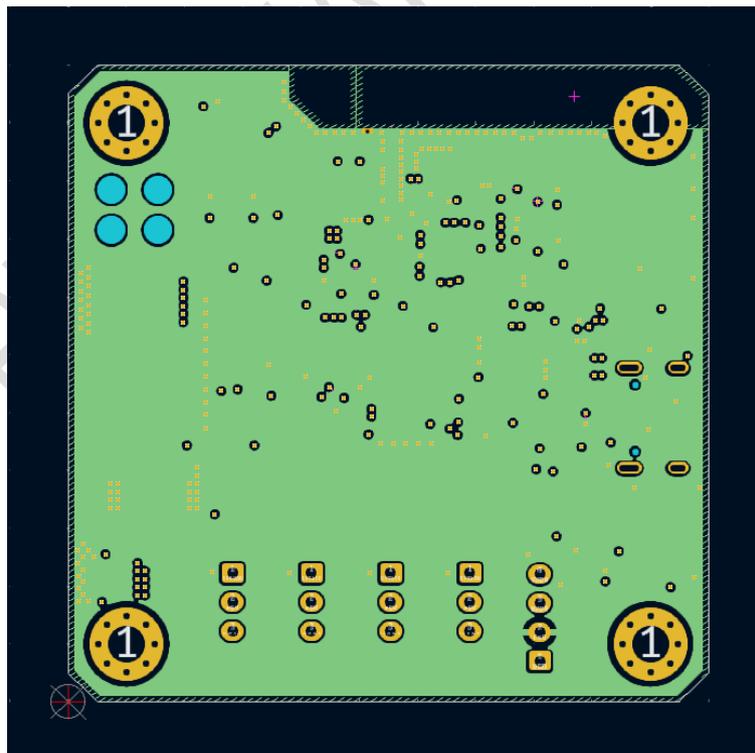


Figure 24: TAv1.6 Layer 2 - GND

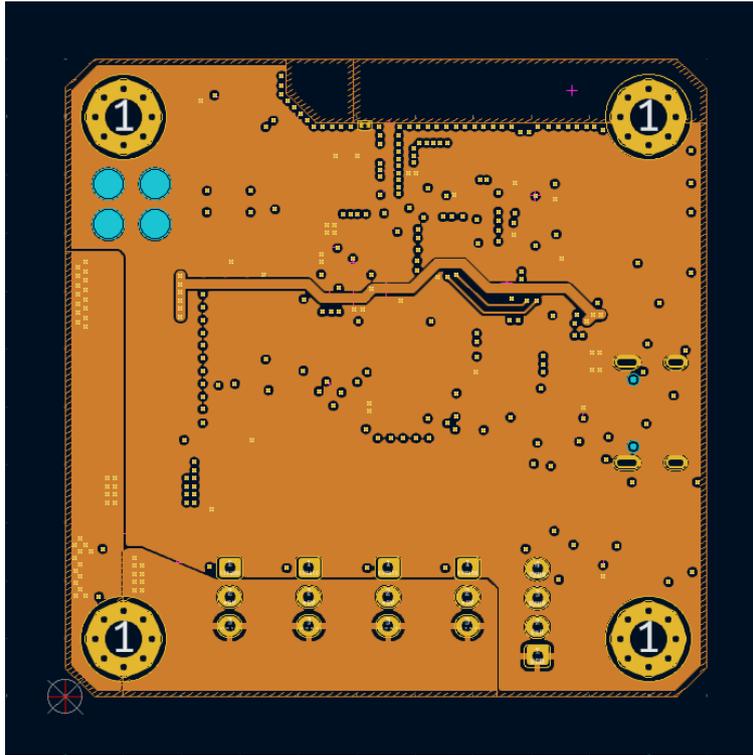


Figure 25: TA_v1.6 Layer 3 - Power

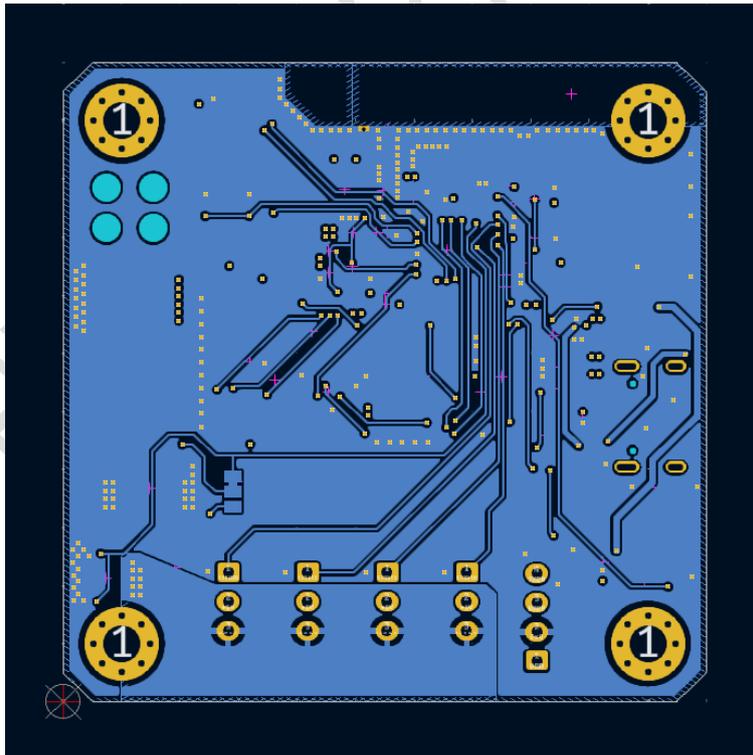


Figure 26: TA_v1.6 Layer 4 - Bottom signal

TA GitHub repository

Barsotion-TA's KiCad sources, full schematic, pindefs-file are located at a [project GitHub repository](#).

PRELIMINARY EDITION

EA & TA placing

The EA / TA board computer is placed in the nose rocket section as shown at the following renders.

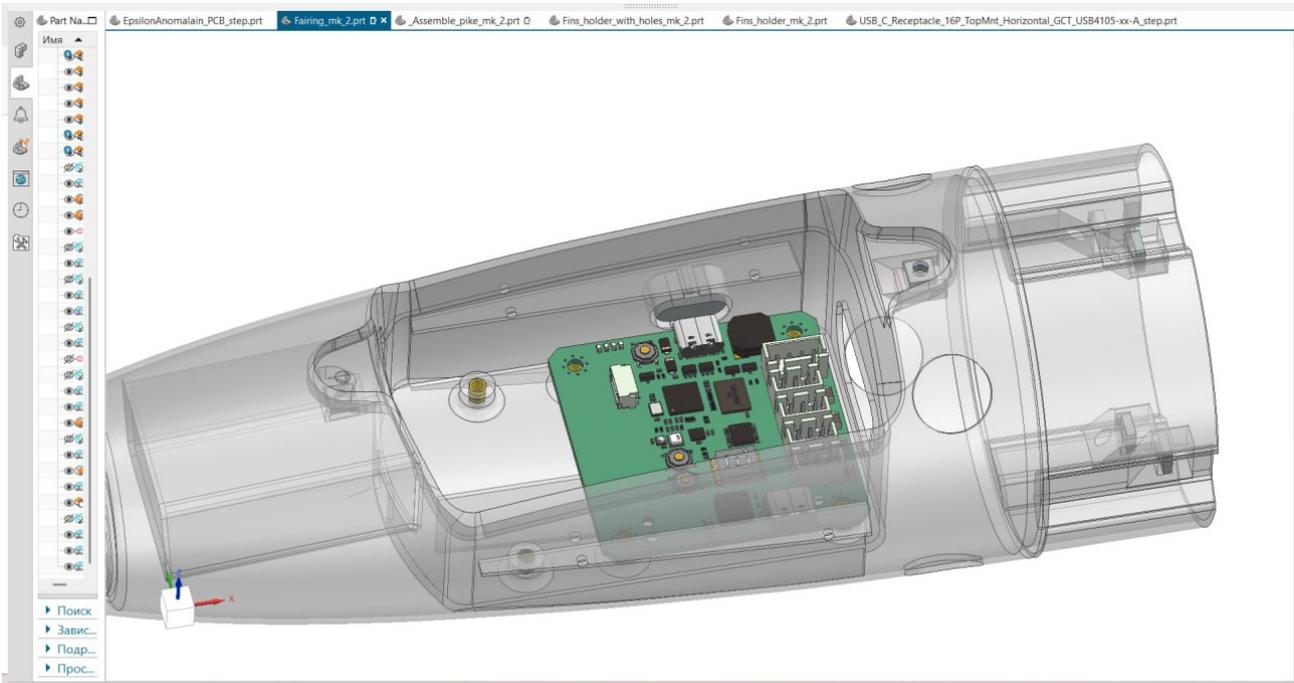


Figure 27: EA / TA placing

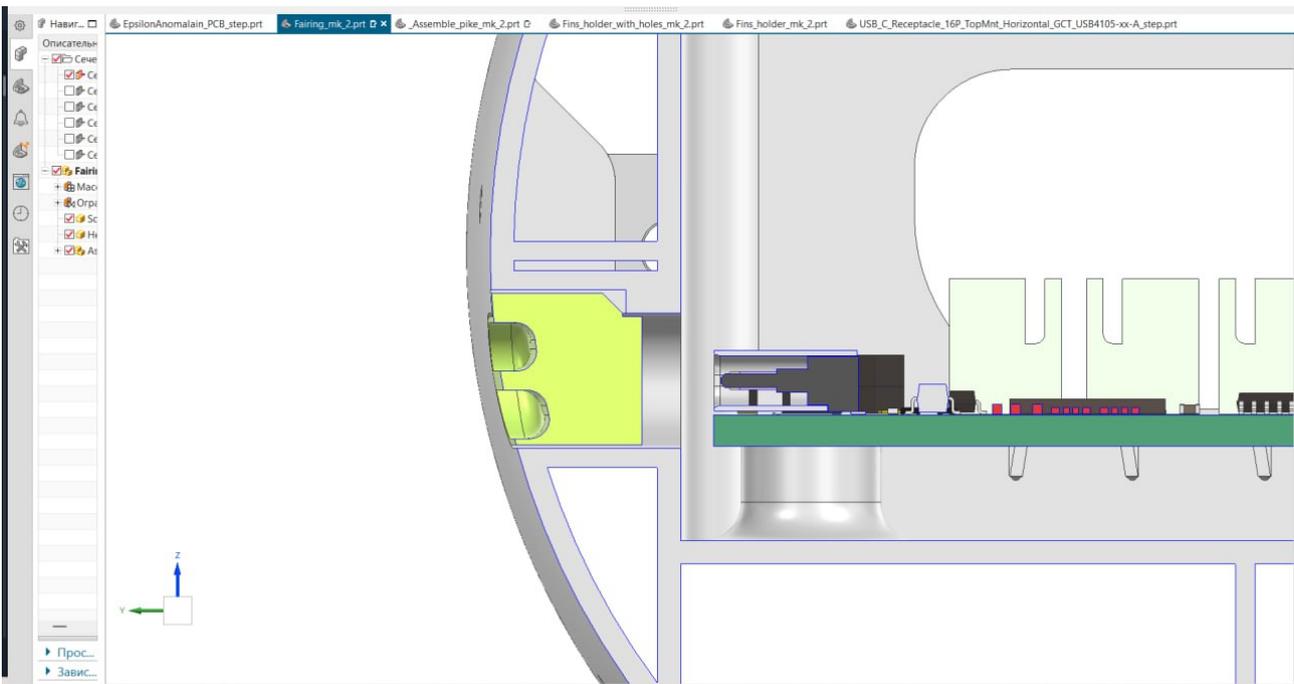


Figure 28: USB-C connector hole

EA & TA software packet

Abstract

The EA & TA software packet is a big complex software product. It is named after a modern philosopher A.Dugin.

Device connection for flashing & communication

EA / TA device PC connection shown at figure. Only one USB-C cable is necessary for connection.

Software building & device flashing are proceeded via Espressif IDE (Eclipse-based). It is recommended to restart device in BOOT-mode before flashing (press “R” button on board when “B” button is pressed).

Communication with device is possible via serial terminal on PC or special command-line utility (see [EA & TA PC interface](#)).

Communication is proceeded via a set of ASCII-commands. It can be easily entered to serial terminal. The EA / TA device is ready to receive commands when it has sent a command line marker: “> ”.

Commands summary

Command	Description
<i>Servo commands</i>	
SAP	SERVOA add a small positive offset (servo A bias plus)
SAM	SERVOA add a small negative offset (servo A bias minus)
SBP	SERVOB add a small positive offset (servo B bias plus)
SBM	SERVOB add a small negative offset (servo B bias minus)
SCP	SERVOC add a small positive offset (servo C bias plus)
SCM	SERVOC add a small negative offset (servo C bias minus)
SDP	SERVOD add a small positive offset (servo D bias plus)
SDM	SERVOD add a small negative offset (servo D bias minus)
SZERO	Reset zero offsets for all the servos and print
SPRINT	Print the servo angles (is needed if the servos were rotated)

	manually and you need to check the zero-angles)
I2C commands	
TWI-VERIFY	Check the connection of all the supported I2C peripherals
TWI-RESET	Reset the I2C bus
BMP388-ID	Get BMP388 ID and check it
AH3-ID	Get AH3 ID and check it
Disk commands	
DISK-INFO	Print information about the disk: manufacturer, type, capacity, packet length, number of written packets, disk's percent of usage
DISK-ERASE	Erase the disk. Checks the number of written packets and packet length, then erases the required amount of 128-KiB blocks
PACKET-WROTE	Prints the number of written packets
Get/set parameters commands	
BOOT-ENTIRE-READ	Reads the disk's boot entire and prints the information about entire encoding version, type of the device (EA or TA), device unique id, packet length, number of written packets, IMU calibration coefficients, IMU cycle frequency in Hz, token value (should be 0x55AA), etc.
PACKET-LEN	Prints the log packet length
IMU-CYCLE-FREQ	Get IMU cycle frequency in Hz
DEVICE	Get device's info
SET-BOOT-ENTIRE	Set boot entire to default state
SET-IMU-CYCLE-FREQ	Set IMU cycle frequency in Hz (allowed values: 500, 1000, 2000, 4000, 8000)
IMU commands	
IMU-CALIB	Calibrate IMU and save coefficients
IMU-CALIB-COEFS	Prints all the IMU calibration coefficients from boot entire
System commands	
POWER-OFF	Device power down
RESTART	Restart device
ACTION	Switches the computer to active mode
Helper commands	
HELP	Prints available commands

Debug commands	
PAGE-CONTENTS	Prints disk's page contents
TEST-ACTION	Writes 1000 packets to the disk as is in Active mode and returns control to user's terminal
Machine mode commands	
MM-PACKET-LEN	Same as PACKET-LEN, but does not beepes at the end
MM-PACKET-WROTE	Same as PACKET-WROTE, but does not beepes at the end
MM-DISK-READ	Reading disk's binary data

Servo control commands

SxP & SxM commands

Created to calibrate servos zero state before flight. This commands, when received, add a small offset to the servo zero state and print zero state to the servos.

SZERO command

Resets all the extra offsets added by SxM & SxP commands and prints zero state value to servos.

SPRINT command

Prints zero state to the servos. It is needed if servos angles have already calibrated, but were changed manually so it is necessary to set calibrated zero state value to servos.

I2C control commands

TWI-VERIFY command

Checks the BMP388 and AH3 connection and prints connection status to the serial.

```
>  
> TWI-VERIFY  
BMP connected ok.  
AH3 connected ok.  
>  
.
```

Figure 29: TWI-VERIFY command response

TWI-RESET command

Resets the I2C bus. It is needed if the connection was lost but then.

BMP388-ID command

Reads BMP388's ID and check it. The proper value is 0x50, if read value is not equal, the system raise the exception.

```
>  
> BMP388-ID  
BMP388 ID: 0x50, it is a proper value.  
>
```

Figure 30: BMP388-ID command response

AH3-ID command

Reads AH3's ID and check it. The proper value is 0x74, if read value is not equal, the system raise the exception.

```

>
> AH3-ID
AH3 ID: 0x74, it is a proper value.
>

```

Figure 31: AH3-ID command response

Disk control commands

DISK-INFO command

Prints to the serial disk connection status, disk type (from JEDEC ID), capacity information, and if the boot entire is formatted, boot entire information: number of written packets, disk usage status.

```

> DISK-INFO
Device: 25N01, 1Gbit SPI NAND Flash
Packet size: 100
Packet wrote: 3
Wrote: 0.29KiB, 0.00% of total flash
■

```

Figure 32: DISK-INFO command response

DISK-ERASE command

Erases all the log area of disk. Erasing is necessary before writing information in Flash memory. DISK-ERASE command reads number of written packets and packet length values from boot entire and erases strictly necessary amount of blocks that save user's time and device resource.

PACKET-WROTE command

Reads the number if written packets from the disk's boot entire and prints it to the serial.

Get/set parameters commands

BOOT-ENTIRE-READ command

Reads the boot entire and writes full information about it to the serial. It prints:

- Boot entire structure version (only 0x01 is today exists)

- Board type (EA or TA)
- Device unique ID (each board computer should have an unique number)
- Log packet length
- Number of written packets
- IMU calibration parameters

```

> BOOT-ENTIRE-READ
Encoding version:      1
Device:                EA
Device unique ID:     1
Log packet size:      100
Log packet wrote:     3
IMU calibration parameters:
- gyro_bias:          x:0.000000      y:2.870526      z:0.000000
- gyro_eps:           x:2.876356      y:1.572361      z:0.000000
- accel_eps:          x:0.000000      y:0.000000      z:-0.000000
- calibrating temperature: 0.000000 *C

```

Figure 33: *BOOT-ENTIRE-READ* command response

***PACKET-LEN* command**

Reads the packet length value from the disk's boot entire and prints it to the serial.

***IMU-CYCLE-FREQ* command**

Returns temporary defined IMU processing cycle frequency in Hz.

***DEVICE* command**

Prints device's info into the serial: device type (EA or TA) and device's unique ID. The data are read from boot entire.

```
>  
> DEVICE  
Device: EA; unique ID: 1  
>
```

Figure 34: DEVICE command response

SET-BOOT-ENTIRE command

Sets boot entire to the default state:

- Sets the proper device according the software;
- Sets log packet size according the software;
- Sets number of written packets to zero;
- Sets IMU cycle frequency to 8000 Hz;
- Sets IMU calibration coefficients to zero;
- Sets token field to 0x55AA.

SET-IMU-CYCLE-FREQ command

Sets IMU cycle frequency. After the command input the interface asks user to text the frequency needed. Allowed values: “500”, “1000”, “2000”, “4000”, “8000”. If the user writes wrong value, the system terminates cycle frequency update and returns an error message.

```
> SET-IMU-CYCLE-FREQ  
New IMU cycle frequency in Hz: 8000  
New IMU frequency value writed, please restart the device
```

Figure 35: SET-IMU-CYCLE-FREQ command response 1

```
> SET-IMU-CYCLE-FREQ  
New IMU cycle frequency in Hz: 9999  
Wrong value, allowed values:  
* 500  
* 1000  
* 2000  
* 4000  
* 8000
```

Figure 36: SET-IMU-CYCLE-FREQ command response 2

IMU commands

IMU-CALIB command

Calibrates the IMU and writes calibration values to the disk.

IMU-CALIB-COEFS command

Prints calculated values read from the disk.

```
> IMU-CALIB-COEFS
IMU calibration parameters:
- gyro_bias:      x:1.577705      y:1.577759      z:0.000000
- gyro_eps:       x:0.000000      y:0.000000      z:2.083296
- accel_eps:      x:2.083305      y:-0.000000     z:2.870596
- calibrating temperature: 1.574953 *C
```

Figure 37: IMU-CALIB-COEFS command response

System commands

POWER-OFF command

Power down the device. Note that the function does not work on EA.

RESTART command

Restarts the software.

ACTION command

Switch the device into the Active working mode.

Helper commands

HELP command

Prints helper message with all the supported commands descriptions.

Debug commands

PAGE-CONTENTS command

Prints page contents. After the user inputs this command, the system asks him, which page he would like to see. Then, after page number receiving, the system prints 2048 bytes of page contents in 16 columns with start addresses on the left hand.

TEST-ACTION command

Writes 1000 packets to the disk. The command is created to simulate computer working in the Action mode. After packets writing, the system returns control to user.

Machine mode commands

MM-PACKET-LEN command

Does the same as PACKET-LEN, however it does not beeps the buzzer. This command is needed for EA & TA PC interface. “MM” means “machine mode”.

MM-PACKET-WROTE command

Does the same as PACKET-WROTE, however it does not beeps the buzzer. This command is needed for EA & TA PC interface.

MM-DISK-READ command

EA & TA board computers architecture allows to write a big amount of log data. When there are more than hundred megabytes of log data, their reading becomes a difficult task.

The Barsotion DeltaAnomalain board computer used to send log data to PC via serial interface as CSV-strings, so the packets were read, then converted to CSV-string and sent via UART.

This way was not effective because CSV-strings wear more than binary data. EA & TA software packet send packet data in binary form.

MM-DISK-READ command, when received, checks the disk connection and then responds OK, if the connection is valid. Then the board computer waits to

sending any bytes via USB. When byte is sent, it sends a packet binary raw data as an answer. It sends packet 0 contents after start, then the address is incremented.

MM-DISK-READ command is used by EA & TA PC interface and allows read log data via USB at temporary maximum possible speed about 60 kilobytes per second.

PRELIMINARY EDITION

EA & TA PC interface

For the purpose of reading logs automatically and simplification EA & TA user experience, a gbk-oracle utility was created.

The log-decoder utility was created for decoding .bin file received as a result of gbk-oracle and conversion data to the JSON format.

GBK Oracle utility

GBK Oracle usage

GBK Oracle utility needs the Python 3 installed on the PC.

The standard launch query look like as shown below.

```
$ python3 gbk-oracle -p /dev/ttyACM0 --output-dir .
```

Utility's '--output-dir' parameter is optional. It sets directory where the log file will be written. By default it is the same directory when program source file is located.

The '-p' parameter defines serial port name. It is usually /dev/ttyACMx for Linux and COMx for Windows systems.

Optional '-b' parameter sets the baudrate. It is maybe useless because USB interface do not mind about baudrate in CDC mode, data samples are transmitted by packets.

GBK Oracle commands

The GBK Oracle utility commands are mostly the same as EA & TA software packet serial interface commands, but there is a new command – DISK-READ.

GBK Oracle utility allows to enter commands . For example for command DISK-READ four writing forms are allowed

- DISK-READ
- DISK_READ

- disk-read
- disk_read

Then the full list of GBK Oracle commands is presented. In the following table ‘xxx-xxx’ form of commands is used in goal to separate GBK Oracle commands from EA & TA software packet commands.

Command	Description
sap	SERVOA add a small positive offset (servo A bias plus)
sam	SERVOA add a small negative offset (servo A bias minus)
sbp	SERVOB add a small positive offset (servo B bias plus)
sbm	SERVOB add a small negative offset (servo B bias minus)
scp	SERVOC add a small positive offset (servo C bias plus)
scm	SERVOC add a small negative offset (servo C bias minus)
sdp	SERVOD add a small positive offset (servo D bias plus)
sdm	SERVOD add a small negative offset (servo D bias minus)
szero	Reset zero offsets for all the servos and print
sprint	Print the servo angles (is needed if the servos were rotated manually and you need to check the zero-angles)
twi-verify	Check the connection of all the supported I2C peripherals
twi-reset	Reset the I2C bus
disk-info	Print information about the disk: manufacturer, type, capacity, packet len, number of written packets, disk’s percent of usage
disk-erase	Erase the disk. Checks the number of written packets and packet len, then erases the required amount of 128-kiB blocks
boot-entire-read	Reads the disk’s boot entire and prints the information about entire encoding version, type of the device (EA or TA), device unique id, packet length, number of written packets, imu calibration coefficients
packet-len	Prints the log packet len
packet-wrote	Prints the number of written packets
action	Switches the computer to active mode
power-off	Device power down
help	Prints available commands

disk-read	Reading disk's data into the file
exit	Quit the program
quit	Quit the program

Log decoder utility

Log decoder usage

```
$ python3 log-decoder --decoder-version 2
```

The utility provides a simple usage. This string, send to the bash terminal, decode log.bin file converting it to log.json.

PRELIMINARY EDITION

Revision history

Date	Modification
May 8, 2025	TAv1.6, new commands, Log decoder utility addition
Apr 29, 2025	Document creation

PRELIMINARY EDITION

Contacts

If you have any questions, feel free to write via email or Telegram.

Email: barsotion@yandex.ru

Telegram: @barsybarsevich

PRELIMINARY EDITION