# Dragon OSD Manual

Software version 4.6 (28 Nov 2007)
Board revision 1.0
Manual revision 2.0 (30 Nov 2007)

## 0.0 Changes

*Version 4.6 to 4.5*

- Satellite screen now comes up after unsetting home

*Version 4.4 to 4.5*

- Changed menu updating to be more responsive
- Fixed vertical adjustment in SAFEZONE

*Version 4.1 to 4.4*

- Added new menu item MAHMULT
- Added new menu item AMPMULT
- Added new menu item SAFEZONE.
- Added new menu item VOLT1LOWARN
- Fixed some issues in the home heading calculations
- Version 4.3 was pulled due to bugs

## 1.0 Introduction

The DragonOSD began as an open sourced OSD project that is fully featured, with innovative design, and high quality graphics. Some of the more interesting features of this OSD are:-

*Flight logging* – GPS tracks can be downloaded and viewed in Google Earth
*Autopilot* – can be set to fly the plane home automatically
*Home indicator* – will always indicate the direction back to launch site
*mAh meter* – will give an indication of mAh consumed
*Open sourced* – the user can modify the firmware
*Blank screen* – you can turn off all on screen display for video purposes
*High resolution* – through the use of the highly-acclaimed BOB-4 OSD

Of course, the OSD will also perform all the other standard functions such as the display of heading, altitude, time elapsed, battery voltage(s) and so on. The DragonOSD board itself is about 18g without the CR2032 battery, and together with the BOB-4SG it comes up to about 25g. This does not include the current sensor either.

While you can probably just plug the DragonOSD in and just fly with it, you stand to get the most out of the device if you understand the various functions and the hardware interfaces. That is the purpose of this document, to familiarize you with the various aspects of the DragonOSD.

## 2.0 Power supply

Maximum supply voltage: **16V (recommended 7.2V)**

Minimum supply voltage: **6V**

The DragonOSD was designed to run off a 2S or 3S Lithium-Polymer battery. As such the input filtration capacitors have been rated at 16V. Exceeding this voltage will likely cause the input capacitor (tantalum) to blow.

There are two regulators on the RTF modules, an onboard 3.3V one, and an outboard 5V one. What this means is that these regulators will dissipate the difference of voltage times the current drawn. So, for example if the BOB-4SG module is drawing 80mA, and the supply voltage is 11.1V, the 5V regulator will be dissipating:-

$$(11.1 - 5)V * 0.08A = 0.488W$$

On the other hand, if your supply voltage is 7.2V, this dissipation now becomes:-

$$(7.2 – 5)V * 0.08A = 0.176W$$

What we can learn from this is that it is better to power the DragonOSD from as low a voltage as possible and 2S would be ideal and will result in the minimum heat dissipation and power wastage. In fact, with this current version, the out-board 5V regulator can only dissipate 250mW safely, meaning that running the board off 11V will likely result in an overheating condition and is not recommended. Consequently*, it is recommended that this revision of the* ***DragonOSD be run off a 2S supply***. If you wish to run this off 3S, it will require a beefier regulator (and likely bigger) but is a relatively simple modification. If you would like to do this, please contact me for more information. Bear in mind that the DragonOSD was originally conceived as running from a dedicated battery, separate from the battery that runs the ESC and motor.

The DragonOSD board itself draws about 40mA, while the EB-85A GPS module draws between 40mA to 80mA depending on which phase it is in. Once the satellites are locked in, it tends to drop to around 40mA. This will yield a quiescent current of around 80mA for both the DragonOSD and the GPS module. The BOB-4 draws around 100mA according to their datasheet. This will yield a combined current draw of about **180mA** under normal operating conditions.

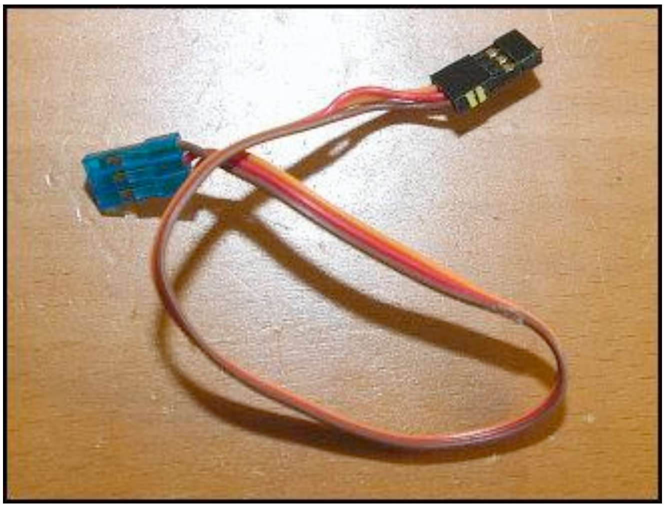
Note that one of the voltage sensors is internally connected to the supply for the DragonOSD board. In other words, it will indicate the battery voltage being supplied to the board. If you are using just one battery for everything, this will mean that the voltage will appear twice on the display.

### 2.1 Battery for GPS module back-up memory

There is a socket for a CR2032 onboard the DragonOSD board. This is used to maintain the memory on the GPS module after everything has been powered down. The EB-85A will gain a lock much faster (in the order of 10 to 15 seconds) from a warm start, compared to a cold start. If you would like to benefit from this fast start, then you can insert a CR2032 coin-sized 3V battery into the battery holder.

## 3.0 Other Requirements

Apart from the power supply, there are several other things you will need to get the OSD fully up and running. These are a receiver with a spare channel to control the functions of the OSD, a current sensor (more about this later), inter-connecting cables between the receiver and the DragonOSD (3-cables minimum), and cabling for the video in/out.



(*Figure 1*) Cable for connecting DragonOSD to receiver

Additionally, for proper implementation of the autopilot in failsafe mode, you will require a receiver that has programmable servo positions in failsafe mode. An example of this might be the JR SPCM receiver series.

Note also that all the servo signals run on a 3.3V logic level, which should be okay with the majority of the servos (Futaba uses 2.5V, JR uses 5V).

## 3.1 Guidelines for mounting of the DragonOSD

For proper operation of the DragonOSD, there are several guidelines that one should follow. This is in order to ensure minimal interference to the more sensitive electronics. In general:-

- Keep the video transmitter well clear of the GPS module
- Keep the video transmitter well clear of the receiver and antenna
- Keep the BOB-4 as far as possible from the GPS module
- Keep the camera away from the receiver and antenna
- Keep the DragonOSD away from the receiver if possible

Based on this:-

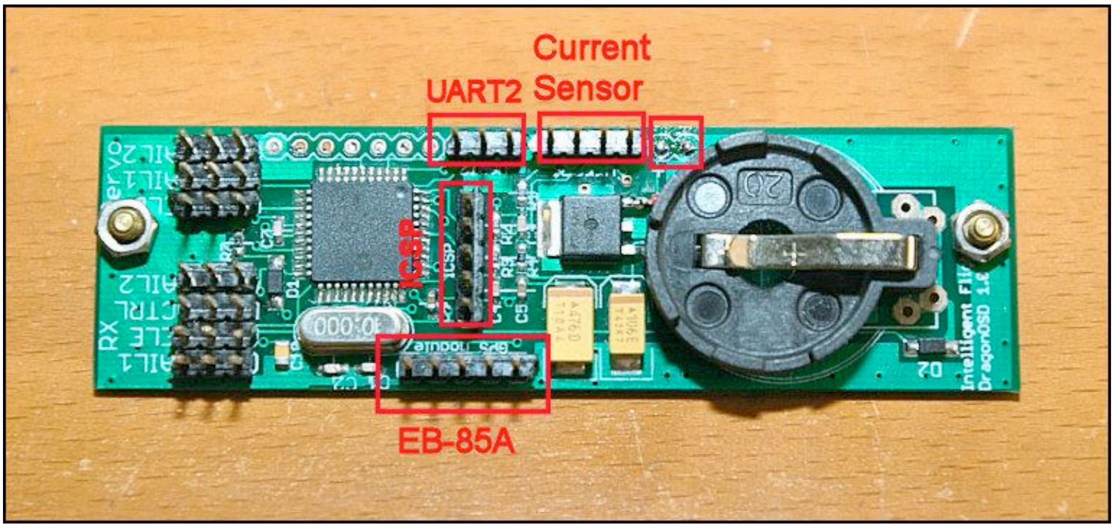
- The GPS and be close to the receiver and antenna
- The video transmitter can be close to the camera
- The camera can be close to the DragonOSD
- The ESC can be close to the camera and DragonOSD

As there are quite a number of additional cables to account for the control lines, one should also take care to:-

- Use twisted cables where possible
- Use clip-on chokes for groups of cables
- Keep the servo and control cables as a group, away from the receiver
- Keep all cables as short as possible
- Keep ESC cables away from the video cables if not shielded

### 4.0 DragonOSD board connections

The DragonOSD board has a number of headers and there are also some connections that need to be made for proper operation.




As can be seen from the photo above (*Figure 2*), there are 11-headers, and some other connections. The headers are:-

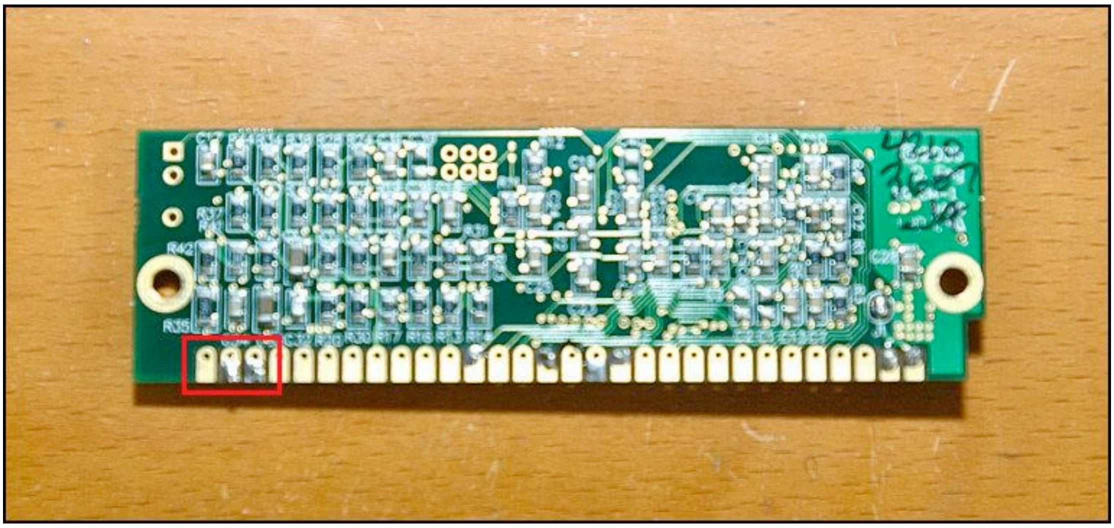
**Left top group of three (from top to bottom) - Outputs**
Aileron-2 output to servo (optional)
Aileron-1 output to servo (can also be applied to rudder servo)
Elevator output to servo

**Left bottom group of four (from top to bottom) - Inputs**
Aileron-2 input from receiver (optional)
Control channel from receiver
Elevator input from receiver

Aileron-1 input from receiver (can also be rudder input)

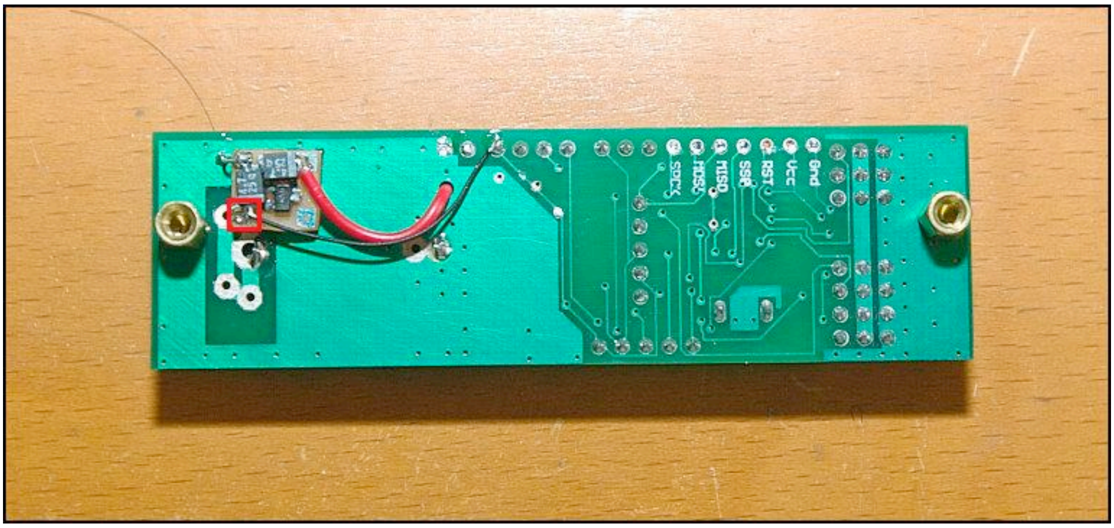
**Marked in red**
UART2 – spare serial port, currently unused
Current sensor – connects to an Allegro ACS755xCB-050
ICSP – programming header
EB-85A – GPS module

**Others**
Red box to the right of the current sensor – power supply connection
Series of pads to the left of the UART2 – BOB-4SG connections



On the underside of the BOB-4SG board picture above (*Figure 3*), the three edge connectors marked by the red box is where the video in/out are to be soldered. **From left to right**, they are:-

Video in (from camera)
Ground
Video out (to video transmitter)



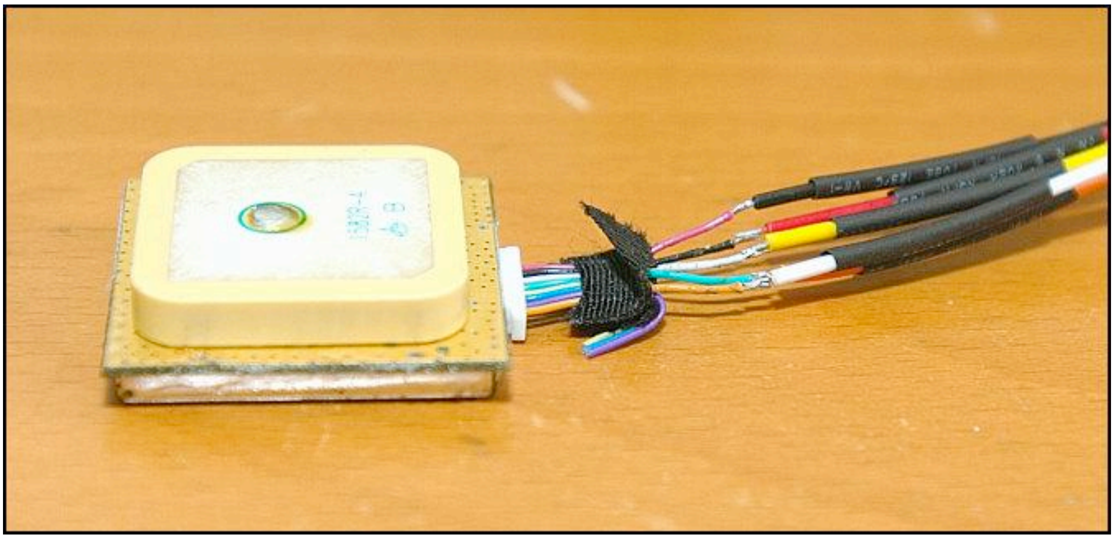
For those who are getting the boards without the BOB-4SG already connected, there are several more connections that are of interest to you. The first is the corner pad on the outboard regulator PCB, marked by the tiny red box on the left hand side of the board (pictured above, *Figure 4*). This is the 5V regulated output of the board. You can see that a wire is drawing that 5V for use with the current sensor as well. You will need to bring a wire from this board to pin-3 of the BOB-4SG to power the OSD. Additionally, you can now see that the connection pads to the BOB-4SG have some silk-screening indicating their functions. More of this will be covered in a later section.

### 4.1 Connecting the EB-85A GPS module

The EB-85A GPS module is connected to the DragonOSD through the 5-pin header marked as “GPS Module” on the board itself. The pin-outs of this header as per the above photograph, **from left to right**, is as follows:-

*RX1* on the GPS (green wire)
*TX1* on the GPS (white wire)
*BAT* on the GPS (orange wire)
*GND* on the GPS (black wire)
*Vin* on the GPS (red wire)

The easiest way to make the connections is to cut the supplied cable, strip it carefully, and solder it to normal cabling for the connectors for the DragonOSD board (see picture below). You will want to heat shrink the connections to provide insulation and strain relief. The unused wires should not be stripped.



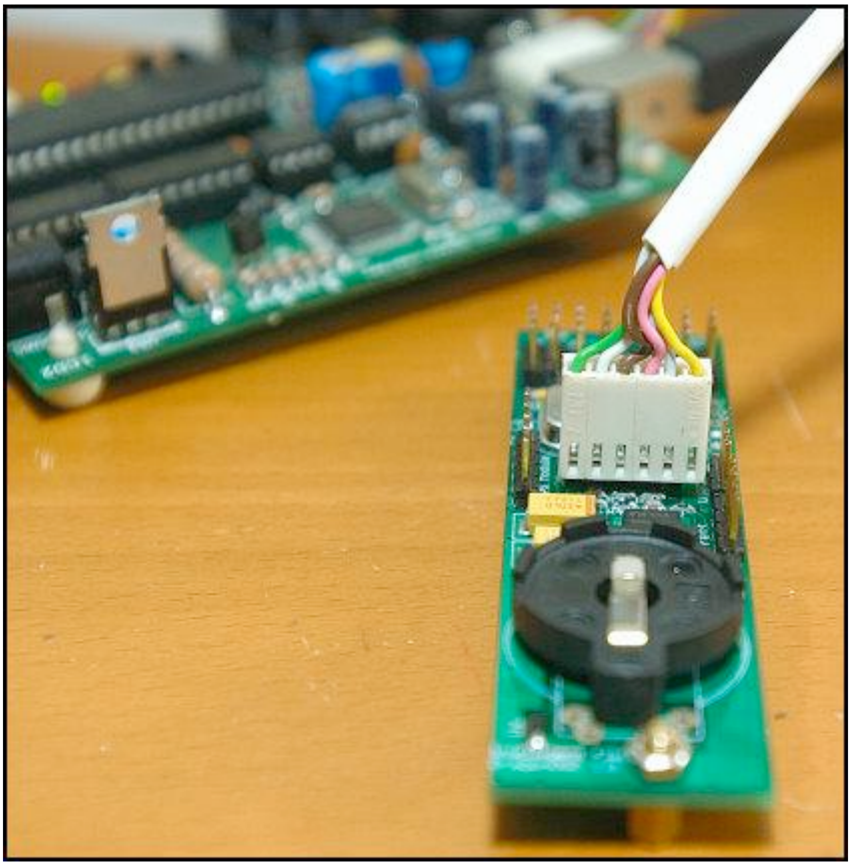
If you are buying a new EB-85A module, make sure that it has the latest firmware that fixes some troublesome bugs that found its' way into the earlier revisions. The new firmware performs much better than the old ones.

### 4.2 The ICSP programming header

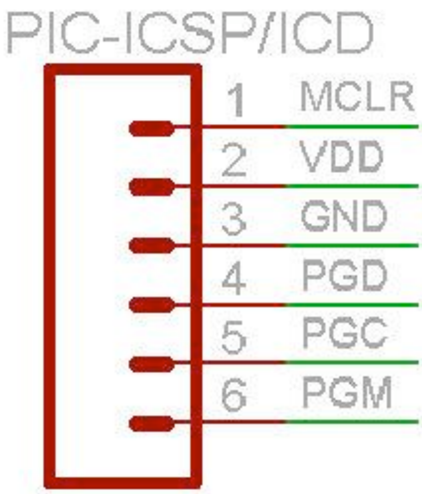
The ICSP programming header is used to program the onboard dsPIC30F4011 microcontroller. The pinout of this header conforms with the ICD2 convention which is also used by Olimex ICD2 programmers. The pinouts are (from top to bottom as per above photo):-

*PGC*
*PGD*
*GND*
*Vdd*
*MCLR*

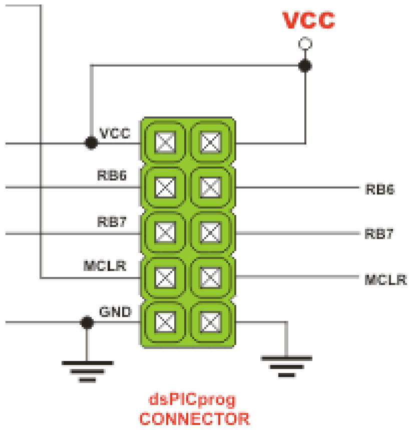
You will notice that the ICD2 header actually has 6-pins. The extra pin is for the PGM connection (yellow wire if using the Olimex cable) and is not used here.



The Olimex header is wired as follows:-

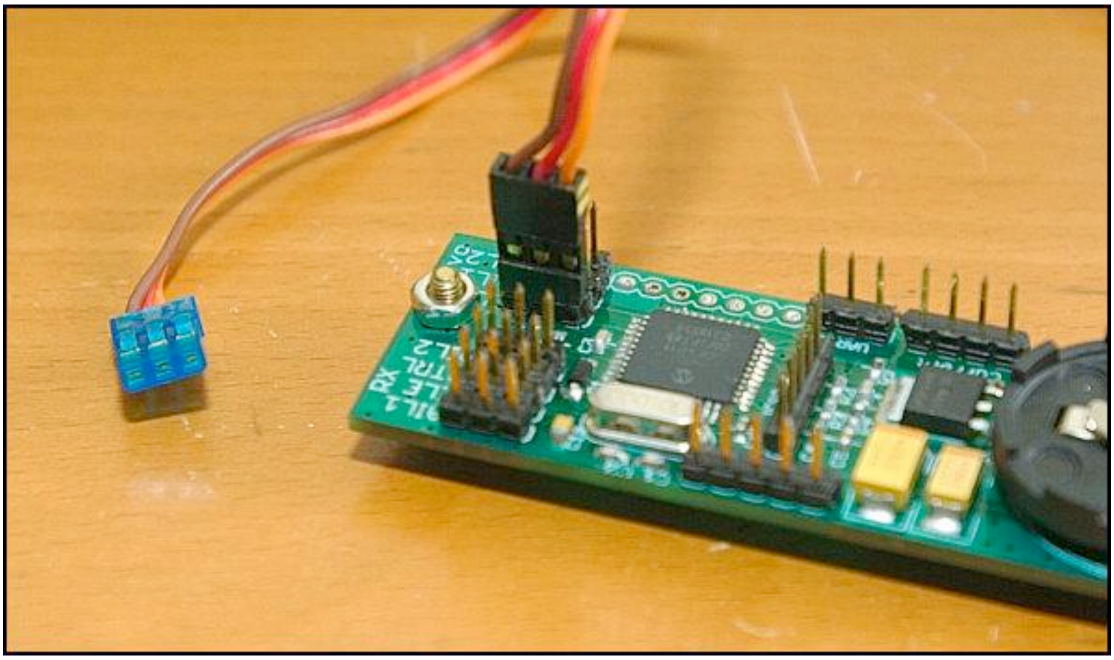



Bear in mind that if you are going to use the mikroElektronika dsPIC programmers (such as the dsPICprog2), they use a different header (pictured below) and will need to make a cable to conform to what we are using here.




### 4.3 Connecting the receiver and servos to the DragonOSD

These connections will be using standard servo cables (3-wires) except that both ends have male connectors (see picture of cable above – Figure 1). This may require some crimping to make these cables if you cannot buy them off the shelf. On the board photo, the left most pins of the headers are GND. This should correspond to either the brown or the black wire of the cable when using normal color convention.



Example of a cable connection to the servo connector.

### 4.4 Connecting the power supply

The power supply is just two pads (actually through holes) just to the right of the header for the current sensor. It is marked as a small red box in the photo (Figure 2). **From left to right**, the pads are:-

*Vcc* – positive terminal from the battery (red wire)
*GND* – negative terminal from the battery (black wire)

You can solder your wires to the pads directly or if you wish, you can clear the holes and put the wires through although this is not entirely necessary.

### 4.5 Connecting the current sensor

The current sensor is comprised of an Allegro ACS755xCB-050. This is a 50A max. current sensor and is connected directly to the current sensor header. The header pinout is (**from left to right**, Figure 2) as follows:-


*VOUT* (Current sense output, **3**)
*Voltage2* (Connects to the tabs **4 or 5**)
*GND* (**2**)
*Vcc* (+5V, **1**)

On the sensor, tab 4 goes to the positive wire from the battery (red wire), and tab 5 goes to the ESC (red wire). If you can put a 100nF capacitor across pins 1 and 2 of the sensor, that will help with noise suppression. Note that the information from the current sensor will not be displayed unless a voltage is detected at the *Voltage2* input. You may also want to connect pin 2 to the negative battery wire if you can.

### 4.6 Connecting to the BOB-4SG

For those who have got themselves an ARTF board, you will need to make the interconnections with the BOB-4SG (SIMM module). This board is made by Decade Engineering and can be purchased directly from them or from other sources. There are, in total, 6 (or optionally 7) connections that need to be made between the boards. These are (as indicated by the silk-screening under the board, see Figure 4 above) **from left to right**:-

*SPCK* (goes to tab 12 on the BOB-4 edge connector)
*MOSI* (goes to tab 22 on the BOB-4)
*MISO* (goes to tab 21 on the BOB-4)
*/SS0* (goes to tab 4 on the BOB-4)
*RST* (Optional: goes to tab 10 on the BOB-4)

For more details about the BOB-4's pinouts, you can refer to the BOB-4 manual.

Before making these interconnections, the BOB-4SG needs to be configured correctly. This can be done by connecting ***tabs 2, 6, 14, 16 and 19*** together. These are ground connections. You also need to solder over the jumper pad (marked J1) just above tabs 2 and 3 (lower right in Figure 3). You can trace the pads to tab 1 and will provide 3.3V to the tab. Having done this, you need to connect a 10k, 1/8-watt resistor from ***tab 1*** to ***tab 13***. These configuration connections are all made on the BOB-4SG itself.

Lastly, you need to bring the power-supply from the DragonOSD board to the BOB-4SG. Because the BOB-4SG runs off 5V, you ***cannot*** use the Vcc (3.3V) from the silk-screened connected under the board. You need to take the 5V from the regulator board (the red square marks the pad in Figure 4). Solder a wire from this pad to ***tab 3*** on the BOB-4SG. Finally, make a ground connection from the exposed wire connecting the regulator board to the ground plane on the main DragonOSD board, to ***tab 2*** on the BOB-4SG. The exposed wire is just above the red square in Figure 4. This will complete the interconnections necessary for the proper operation of the DragonOSD.

## 5.0 DragonOSD Operation

Upon powering up the DragonOSD, you should see briefly the word "BOB-4" on the screen. This is followed by a short countdown with the words "GPS/OSD STARTUP 2" in the middle of the screen. If you get this, then the DragonOSD with the BOB-4 module is working correctly. Once the countdown is complete, the words "Waiting for GPS data …" will appear in the middle of the screen and the DragonOSD will be sending some initialization strings to the attached EB-85A. As long as no valid data is received from the GPS, it will continue on this screen. If you find that nothing is happening, you may want to check that the GPS cable is properly connected with the correct orientation, and that the cable has been wired correctly. Then try to re-connect the power to the DragonOSD board.

If the communications with the GPS module is successful, you should then see quite a bit of data on the screen. The first screen you will see will be the satellites hunting screen (shown below).


The bars at the center of the screen indicates the signal strength of the various satellites that the GPS module is seeing. Immediately below each bar is a number, corresponding to the satellite number for that bar. Note that just because several bars are present, does not necessarily mean that the data has been locked in and so at the top of the screen, it still says “0 SATS”. Once the data becomes valid, it will get locked in and that satellite will be used for computing the current location.

Above the signal strength display are the words “HOME not set”. The way this works is that the program is waiting for the HDOP (Horizontal Dilution of Precision) figure to fall below a preset (and configurable through menu) level. Once that happens, it will start counting down from another preset number. If during the countdown, the HDOP increases, the countdown will be stopped. This way, it will force an increasingly accurate reading to be available for a given period before taking the given position as a valid home position. How long this process takes depends largely on the GPS reception in that area, and also on whether this was a cold start or a warm start.

The countdown will not happen until at least 3 satellites have been locked in with valid data, which will be indicated as “3 SATS” at the top of the screen. When this happens, the HDOP display will show up below the “SATS” display. If you’re cold starting the GPS module, you should initially see zeroes for the coordinates and 137.0m for the altitude (lower number at the top right). Otherwise you should see some positional data such as pictured above.

On the top left of the screen you should see the UTC time as soon as the satellites get locked in. This is indicated by the “UTC” suffix. However, this is only true until the HOME gets set and once it does, this will then become the display of elapsed time rather than actual UTC time. If you're cold starting, this will start with some time around 24:59 and rolling over to 00:00 until it starts getting valid data from the satellites.

During the phase where 3 or more satellites have been locked, and the program is waiting for a valid HDOP (as determined by the menu setting with the same name), it is not unusual to see the GPS coordinates and the altitude fluctuate. The altitude displayed at this time is exactly what the GPS reports, the altitude above sea level. Once HOME is set, this becomes the altitude relative to the HOME position/altitude.

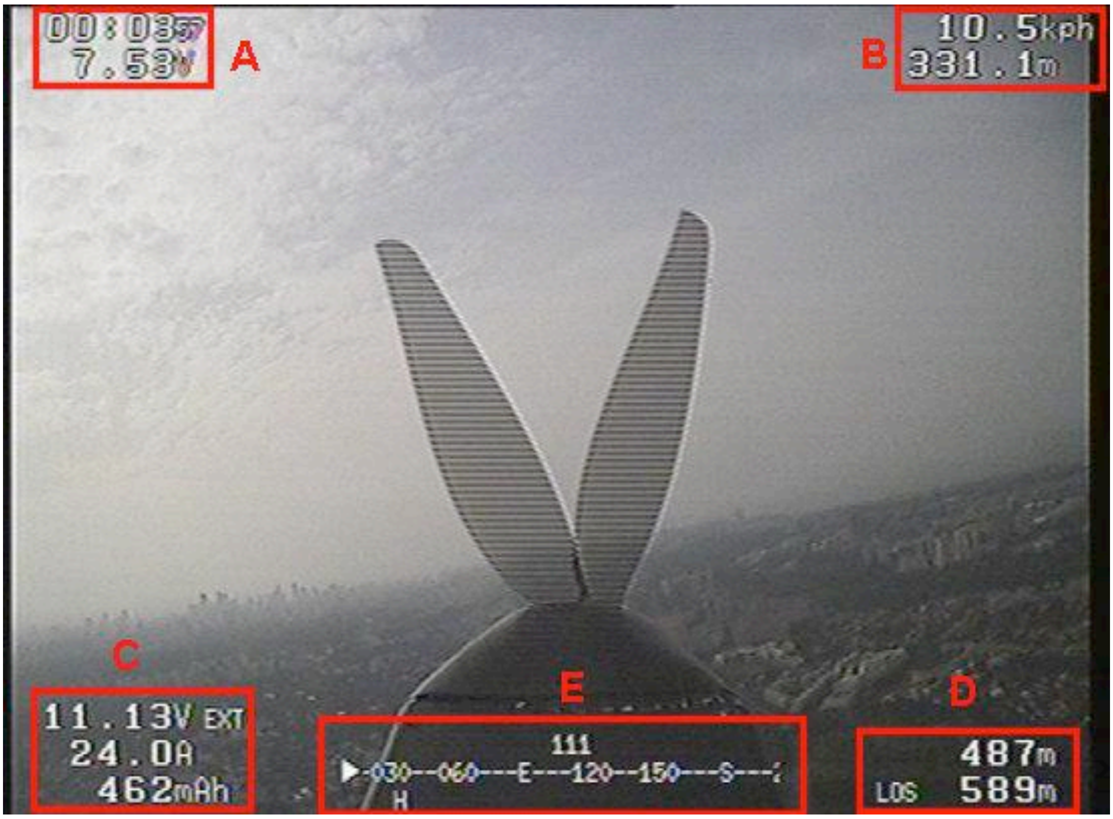
If the conditions for the HDOP and the countdown are met, the satellite signal strength display at the center of the screen will disappear, altitude will be close to zero, elapsed time will start counting up from 00:00 (and the words "UTC" will no longer be shown).


Depending on the altitude fluctuation (can be up to a few meters) the GPS coordinates, number of satellites locked in, and the HDOP display may be automatically turned off. This is determined by the LOWALTRATIO menu setting. If the relative altitude divided by the ground distance (upper number in the lower right group, "3m" in the photograph above) from the HOME position is above LOWALTRATIO, the display will be turned off automatically. Consequently, the further the plane is from HOME, the higher it must be for the display to be turned off automatically, and if the plane should lose too much altitude, the display will automatically come back up. This is so far if the plane is about to crash, the GPS coordinates will be displayed and captured by any recording device that may be recording the flight, and which can then be used to help find the downed plane.

Another thing that you will notice is that on the right side of the screen, a group of values will be displayed under the heading “PEAK VALUES”. This group will show up when HOME is set and the ground speed (topmost right number) falls below about 0.5kph. It will display the highest speed, altitude (absolute above sea level), distance (line of sight) and current drawn (if the current sensor is connected), that has been recorded so far. Once the plane starts moving, this display will usually disappear. Occasionally, it may pop up during flight if the ground speed falls below the 0.5kph level for some reason (such as executing a tight turn, or flying into strong head-wind).

### 5.1 Display elements

We will discuss the primary display elements in 5 major groups.




**Group A**

This group consists of two elements, the upper being the elapsed time (see the startup paragraph for how this display element changes when in the satellite hunting phase). The count-up begins as soon as HOME is set and will continue until the device is powered down.

The lower element is the battery voltage that is supplying the DragonOSD board. This is an internal connection and cannot be changed without extensive modification. There is an automatic voltage detection associated with this (and all voltage measurements). From the starting voltage, the program will make a guess as to the number of Li-Po cells that are in use, and from there compute an appropriate low voltage warning threshold. If the voltage should drop below that threshold for whatever reason, this element will begin to flash as a way of warning the user of impending loss of power.

**Group B**

This group consists of the upper element, the ground speed as supplied by the GPS module, and the lower element, relative altitude (see paragraph on startup conditions for how this element functions during the satellite hunting phase). By relative altitude, we mean the altitude relative to the HOME position/altitude.

When the altitude to ground distance ratio falls below LOWALTRATIO as set in the configuration menu, this lower element will begin to flash as a way of warning the user. On the right of this display will be seen an up or down arrow indicating the vertical direction of the plane. This will usually become more frequently visible for large changes of altitude rather than for smaller changes.

**Group C**

The first thing that needs to be said about this group is that it will only be displayed if the current sensor is connected, or more precisely, if a voltage is sensed on the *Voltage2* pin of the current sensor header.

The first element is the voltage as measured from the tabs of the ACS755xCB-050 sensor. This should correspond to the main battery voltage (maximum of 25V). There is an “EXT” suffix to indicate that it is the external battery voltage that is being measured.

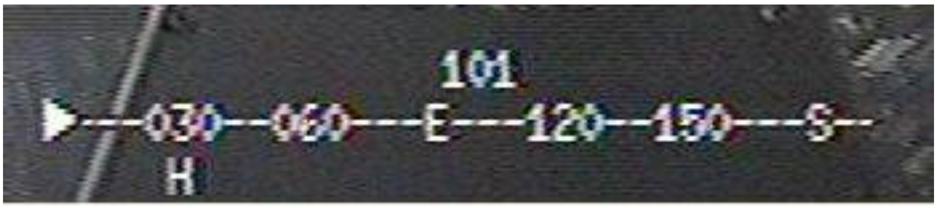
The second element is indicative of the instantaneous peak current drawn, and below that is the mAh consumed thus far. There is also a warning threshold associated with the mAh reading. When the reading exceeds the MAHWARNING value (which can be changed in the configuration menu), it will start flashing to warn the pilot of impending loss of power.

**Group D**

This group consists of two elements, the upper one being the ground distance while the lower one is the absolute, or line-of-sight (LOS) distance. The ground distance can be set to turn off automatically when beyond a certain range. The rationale for this is that at greater distances, the ground distance will approximate the LOS distance so having both may be deemed unnecessary clutter by some. You can configure this to be always on by simply setting a high value for the SHOWHOMELIM parameter in the configuration menu. The ground distance is computed using the Haversine formulae and the LOS distance uses a simple Pythagorean derivate of the ground distance and the relative altitude.

**Group E**

This is a complex group of 4 elements.




The first element is the number in the middle (“101” in the picture) which represents the GPS track. Note that the track is not necessarily the same as the heading in that it represents which way your plane is moving, and not which way it is pointing. As such, you may be pointed towards what you think is your launch point, but the track could be off due to side slip as a result of a crosswind, for example. Below that is the compass reticule which also indicates the GPS track and will scroll to show which way you are turning. This reticule represents the forward 180 degrees of the GPS track heading.

Below the reticule, you can sometimes see a “H”. This represents the heading for the launch point. It will only be visible if the launch point is somewhere in the front 180 degrees of the track heading. If the launch points goes outside of that 180 degrees, then the arrow at the left (or right as it may be) end of the reticule will become outward pointing rather than inward pointing. That will indicate the direction of the launch point but that it is behind us rather than in front of us.

When the track heading is directly towards the launch point, the heading indicator will be printed in inverse to make it more obvious.

**Others**

If the autopilot is engaged, either deliberately or automatically, there will be the words “AUTOPILOT” displayed in the middle of the screen and all GPS coordinates and satellite information will be displayed as well. Note that the autopilot will not engage unless home position is already set.


Note that, all the displayed elements can be turned off (while retaining full function) through the control channel.

### 5.2 Configuration Menu

The configuration menu allows various parameters and options to be set for the DragonOSD. Most of the settings can be saved to EEPROM so that they are retained even after being powered down. In order to use the configuration menu, the Aileron-1 and Elevator inputs must be connected to the appropriate channel on the receiver. Additionally the control channel must also be connected to a spare channel and the pulse width on that channel be set to more than 1.93ms and less than 2.01ms. Above 2.01ms, the display will be turned off.

Once the configuration menu has been called up, you can move up and down through the items using the elevator stick (up or down). To change a setting, move the aileron positions (left or right). If a stick is held in a position other than neutral, it will repeat the incrementing/decrementing action until it reaches the maximum/minimum limit. If the direction of the movement does not correspond to the stick direction, you may have to change the ELESERVODIR and/or RDRSERVODIR setting until it behaves correctly. Doing this will usually also ensure that the servo directions are correctly configured. All changes take place immediately but are not saved to EEPROM unless explicitly commanded by the user. Note also that when you re-enter the configuration menu, you will still be at the same setting as when you exited the configuration menu last. This will assist in making quick adjustments without having to move the selection to the desired item all over again.

*Note also that any changes you make are immediately active the moment you change it, but they are not saved to EEPROM unless you explicitly select the SAVE CONFIG option. Doing so will make the changes survive power cycling.*


**MAXRANGE** *(default 1200m)* determines the LOS distance beyond which the LOS distance indicator will begin to flash.

**LOWALTRATIO** *(default 0.07)* is the ratio of the altitude to the ground distance below which the altitude indicator will begin to flash and the GPS coordinates be displayed automatically.

**VALIDHDOP** *(default 1.3)* is the maximum HDOP acceptable before the HOME SET countdown begins. The lower this value is, the tighter the GPS reception criteria is (demands greater accuracy). Setting this too low will lead to difficulty in setting the home location. Setting it too high leads to inaccurate home locations.

**MINVALID** *(default 75)* is the number of good HDOP readings required before the GPS reading is considered valid for home location. The greater the value, the more stable the GPS reading must be.

**SHOWHOMELIM** *(default 5000m)* is the distance beyond which the ground distance display will automatically be turned off. By setting a high value here, you can ensure that it never gets turned off.

**ROCLIMIT** *(default 10.0 m/s)* is the maximum permissible rate of climb that the autopilot will use in trying to achieve the desired cruise altitude. This value should be adjusted with care.

**ROCSTEPGAIN** *(default 1)* is the gain used by the altitude setting loop for driving the elevator servo. By setting this to a very low value such as 1, you effectively disable the altitude control component of the autopilot. Too great a value here can lead to excessive elevator movement and severe porpoising. Change this value gradually while observing behavior.

**APROCGAIN** *(default 40.0)* is the autopilot gain that determines how sensitive the loop is to altitude errors. Change this value gradually and with care. It is preferably to adjust ROCSTEPGAIN first before adjusting this. Changes may be difficult to perceive.

**ROTLIMIT** *(default 15.0 degrees/s)* is the maximum permissible rate of turn that the autopilot will use while correcting for heading.

**ROTSTEPGAIN** *(default 8)* is the gain used by the heading correction loop for driving the aileron/rudder servo. By setting this to a very low value such as 1, you effectively disable the heading control component of the autopilot. Too great a value here can lead to excessive aileron/rudder movement and severe oscillation. Change this value gradually while observing behavior.

**APROTGAIN** *(default 48)* is the autopilot gain that determines how sensitive the loop is to heading errors. Change this value gradually and with care. It is preferably to adjust ROTSTEPGAIN first before adjusting this. Changes may be difficult to perceive.

**SPEEDLOWLIM** *(default 5.0kph)* is the speed below which the autopilot will not attempt to climb to the cruise altitude.

**CRUISEALT** *(default 100.0m)* determines the desired cruising altitude for the autopilot. You may or may not always achieve this depending on how much energy the plane has. All being well, the autopilot will reach this altitude and try to stay as close to it as possible which flying back to the launch point.

**MAXALTCHG** *(default 10m)* determines how aggressively the autopilot will make its' altitude corrections. Change gradually and with care.

**MAXHDGCHG** *(default 15 degrees)* determines how aggressively the autopilot will make its' heading corrections. Change gradually and with care. Too great a value can lead to the plane rolling over!

**MAHWARNING** *(default 1700mAh)* is the mAh threshold above which the mAh display will begin to flash.

**ELESERVODIR** *(default NOR)* is the elevator servo direction as seen by the DragonOSD board. This does not affect the user control of the servo, just the autopilot's control. If your menu item selection is moving the wrong way in relation to your up/down movement, you will want to change this setting. Note that because it takes effect immediately, it can sometimes lead to some confusing effects. If the menu movement is correct, usually it also indicates that the servo direction has correctly been set.

**RDRSERVODIR** *(default NOR)* is the aileron/rudder servo direction as seen by the DragonOSD board. This does not affect the user control of the servo, just the autopilot's control. If your menu item selection is moving the wrong way in relation to your left/right movement, you will want to change this setting. Note that because it takes effect immediately, it can sometimes lead to some confusing effects. If the menu movement is correct, usually it also indicates that the servo direction has correctly been set.

**UNITS** *(default METRIC)* determines the units used in the OSD display. If IMPERIAL is selected, then miles and feet will be used instead of meters. Note that within the configuration menu, however, the settings will remain in metric units.

**AMPSZERO** *(default 66)* is used to zero the ammeter reading. This should be adjusted so that when the ESC is not connected, the reading of the ammeter is just fluctuating between 0.0A and 0.1A.

**VIDEOMODE** *(default PAL)* is the video mode in use. Select NTSC if your video equipment does not support PAL mode (which also offers better resolution).

**LOGPERIOD** *(default 15s)* is the interval between each logged point in the flight log. At 15 seconds, you get approximately 22 minutes worth of log-time. By increasing this value, you get longer log times but at the cost of lower resolution. Conversely, by using a shorter log period, you increase log resolution at the cost of shorter total log time.

**HOMELOC** is a special function. If the HOME has already been set, this will indicate “SET”. In this case, if you wish to reset the HOME, just change it to “UNSET” and the HOME setting routine will be run all over again. This can be useful if the GPS readings have stabilized further (particularly altitude).

**MAHMULT** *(default 0.00039)* is the coefficient used to compute the mAh readout. You can increase or decrease this to adjust the readout to match the amount of energy you are putting back into the battery with the charger after each flight. As long as you are within 100mAh or so, you should be alright. Ensure that the AMPSZERO reading is correctly set before adjusting this parameter.

**AMPMULT** *(default 0.10955)* is the coefficient used to compute the current readout. You can increase or decrease this as needed to get the amperage reading as close as possible under nominal current draw conditions. Ensure that the AMPSZERO reading is correctly set before adjusting this parameter.

**SAFEZONE** *(default 0)* allows you to move the display components away from the edge thus forming a “safe zone” where the displayed text will not be obscured or cut off. This is useful for older monitors which do not display 100% of the video image. Large values can lead to text overlap.

**VOLT1LOWARN** *(default ON)* enables or disables the Voltage1 warning. This is for those who have made the modification to run the board of 5V so that this will prevent the incessant flashing. Note that this parameter does not get saved to EEPROM and needs to be explicitly disabled for each session.

**SAVE CONFIG** is used to save the changes to EEPROM if so desired. If any changes have been made that has not been saved, an asterisk “*” will be displayed. Once you change this to “DONE” the data will be saved.

## 5.3 Control Channel operation

The control channel is used to switch the DragonOSD between the various operating modes:-

| Operating mode | Control channel pulse width |
|---|---|
| Autopilot | < 1.2ms |
| Normal | 1.2ms – 1.92ms |
| Configuration menu | 1.93ms – 2.01ms |
| Normal with display turned off | > 2.01ms |

This is done by changing the pulse width on the output of the receiver channel being connected to the Control input on the DragonOSD. One way to do this easily is to assign a slider to that channel and adjust the endpoints so that you can access all the operating modes. Another way is to assign it to a three-way switch and select three of the above modes to be assigned to the three switch positions.

In the event of a failsafe, the control channel should be programmed to automatically select the autopilot mode.

## 5.4 Autopilot

The autopilot is designed primarily as a contingency for when the radio control link has failed. To properly set this mode, you need to trim out the plane for level cruising. Once this is done and the plane is stable, program that aileron (or rudder), elevator, and throttle position as your failsafe position. Also make sure that in failsafe, the control channel goes to autopilot mode.

The autopilot works by monitor the rate of turn (ROT) and climb (ROC) of the plane. It tries to correct the heading of the plane while maintaning the ROT and ROC within specified limits. Ideally, it will turn and climb very gradually, rather than turn or climb sharply. The more aggressive the settings, the higher the chance of oscillations or an unwanted roll-over. The autopilot works best with gradual turns. This also means that it won't fare as well in very strong wind conditions, or with inherently unstable planes. Planes with high-wing, and/or polyhedral wing design tends to work better with the autopilot.

Even when the autopilot is engaged, the user can still provide control inputs through the stick if the radio link is still good. The control resolution, however, won't be as good in this mode as in normal flying mode. This feature is especially useful when trying out new settings as it allows you to quickly correct from a bad situation without first having to turn the autopilot off.

When using the autopilot, you have the option of using rudder steering or aileron steering. Both should work although aileron steering is preferred. If your plane uses two separate channels for ailerons, you can connect the second receiver input and servo channel to the Aileron-2 input and output channels respectively. This should allow you to continue to use dual channel features such as flaperons and spoilerons. This is, however, and untested feature so use with caution.

Note that since the autopilot relies heavily on the GPS input, one should verify that the GPS reading is reliable before engaging the autopilot. Older versions of the EB-85A had a firmware bug that could cause serious malfunction of the GPS and hence the autopilot.

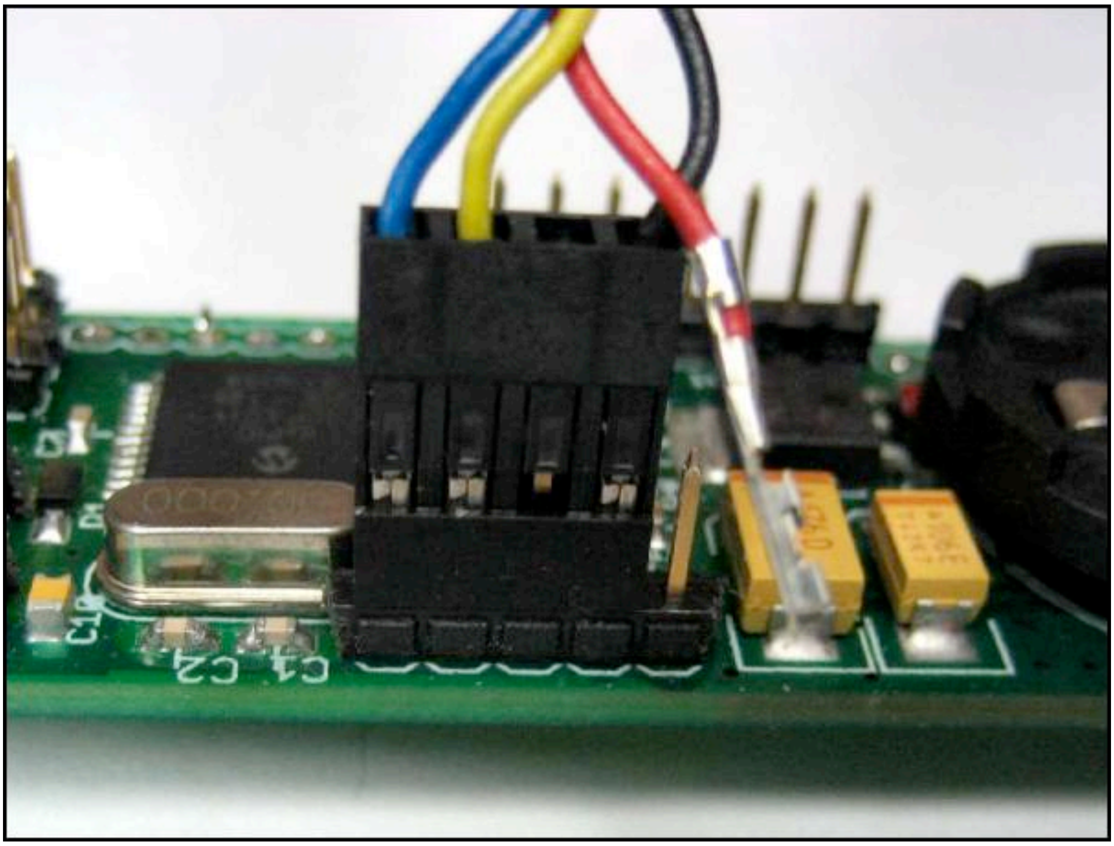
## 6.0 Flight Log

The DragonOSD will automatically log each session. A session is defined as a power cycle, which begins as soon as HOME is set and proceeds recording at the intervals defined by the LOGPERIOD in the configuration menu, for as long as the speed is more than 0.5kph. It will stop logging if it runs out of EEPROM space. Currently a 15-second logging interval will yield about 22-minutes worth of total log time.

If you make a second flight after cycling the power (such as through changing or disconnecting the power to the DragonOSD), a new session comes into effect and the new log will overwrite the previous log. If you wish to retain a log, you can set the LOGPERIOD to zero which will disable logging. This way your subsequent sessions will not overwrite the flight log. If you wish to do this, you need to set the LOGPERIOD to zero *before* you disconnect the power and thus ending the session. You can always re-enable it later by setting the LOGPERIOD to a non-zero setting.

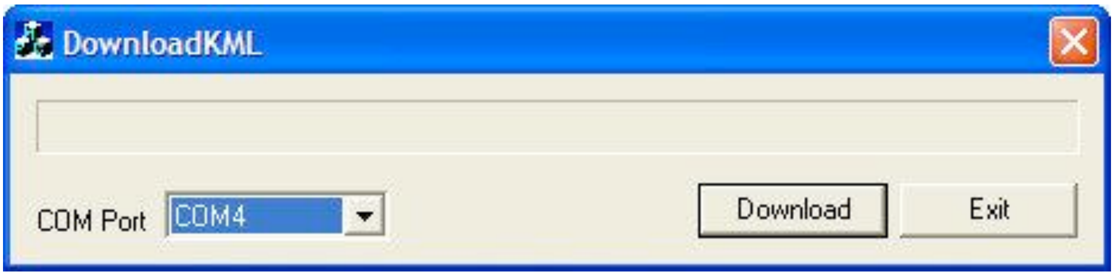
To download the flight log, you will need to connect the GPS serial port, through a level converter, to a PC. You do this by first disconnecting the GPS module from the header, and then connecting a suitably configured cable from the same header to a level converter and through that to a serial port. There are also some USB units which can do the job such as this one (from Intelligent Flight) :-

To do this, you only need to connect the RX1, TX1 and the GND pins to the respective pins on the USB module. The following photo shows how this can be connected. Note that the colour coding is for the current version of the USB adaptor from Intelligent Flight, and that the red wire (5V) is not being used. You should ensure that this wire does not accidentally touch anything else by putting a heatshrink over it. Ie. don't leave it hanging like in the photo below.



*Note: Blue wire is the USB RX which goes to the board TX. Red wire is the USB TX, black wire is GND and red wire is +5V.*

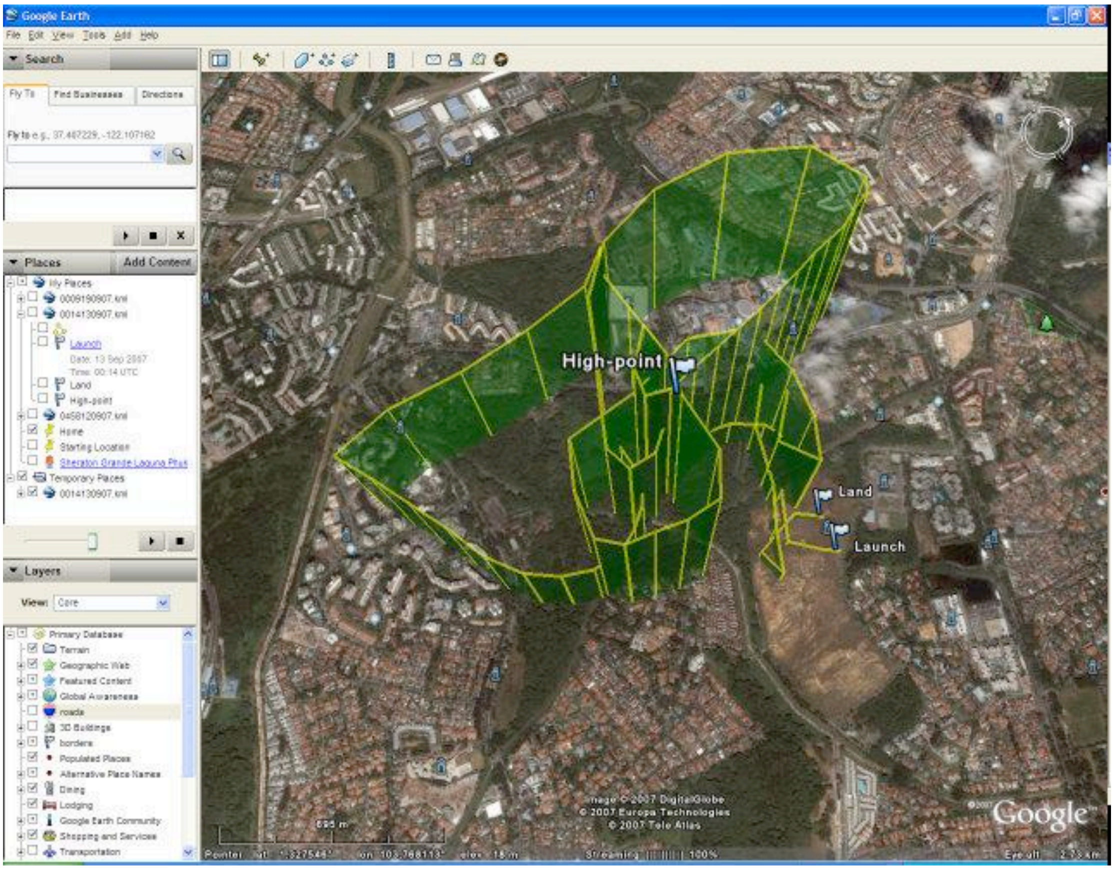
Once this is done, you can use the DownloadKML utility to download the KML file from the DragonOSD.




You must first select the correct COM port for the USB serial port from the drop down list. Make sure that the DragonOSD is connect to the level converter and powered up. Click on the “Download” button and wait for a few seconds. If all goes well, you should see the progress bar fill up while the “Download” button is grayed out. When the download is completed, the “Download” button is rendered normally again. Due to the way the software is written, you may sometimes need two tries to get the file downloaded correctly.

If nothing happens after a few seconds of waiting, you can click on the “Exit” button to close the downloader. Then run it again and repeat the above. Usually this will result in a successful download of the KML file if all goes well.

The file will usually be named as a unique 10-digit number with a KML extension, and will be found in the same directory as the DownloadKML utility. To view this file, open it up with GoogleEarth and you will get a nice 3-D flight log.




There will be 3-markers for each flight log – the Launch, Land, and High-point markers, corresponding to the launch point, the landing point, and the highest-point in the flight, respectively. If you click on the Launch marker, you will be able to see the starting date and time for the flight. Clicking on the High-point will show you the highest absolute altitude (ie. above sea level) achieved in that flight.

### 7.0 Rolling your own firmware

One of the unique aspects of this OSD is that he source code is freely available. The source code was written on and compiled using the mikroC for dsPIC30/33 and PIC24 compiler version 4.0.0.0. Once you have compiled it to a INTEL hex format file, you can burn it into the dsPIC using any of a number of programmers, such as mikroElektronika's own dsPICprog2 or Olimex's ICD2. Microchip appears to now have a very low cost programmer, the PICkit2 that may do the job (using the PICkit2 v2.40 software). You can use the ICD2 or Microchip programmers (with ICSP programming headers) with the freely available MPLAB software to update the firmware. Alternatively, you can put a tiny bootloader on the dsPIC and upload the firmware through the serial port.

Note that while the source code has been made public, the terms of use prohibit anyone from using it for commercial gain or profit. It is provided solely for individual use where no pecuniary interest is involved. If you want to use it commercially, such as with boards that you have made on your own, and intend to offer those boards commercially, please approach Intelligent Flight to discuss possible arrangements.

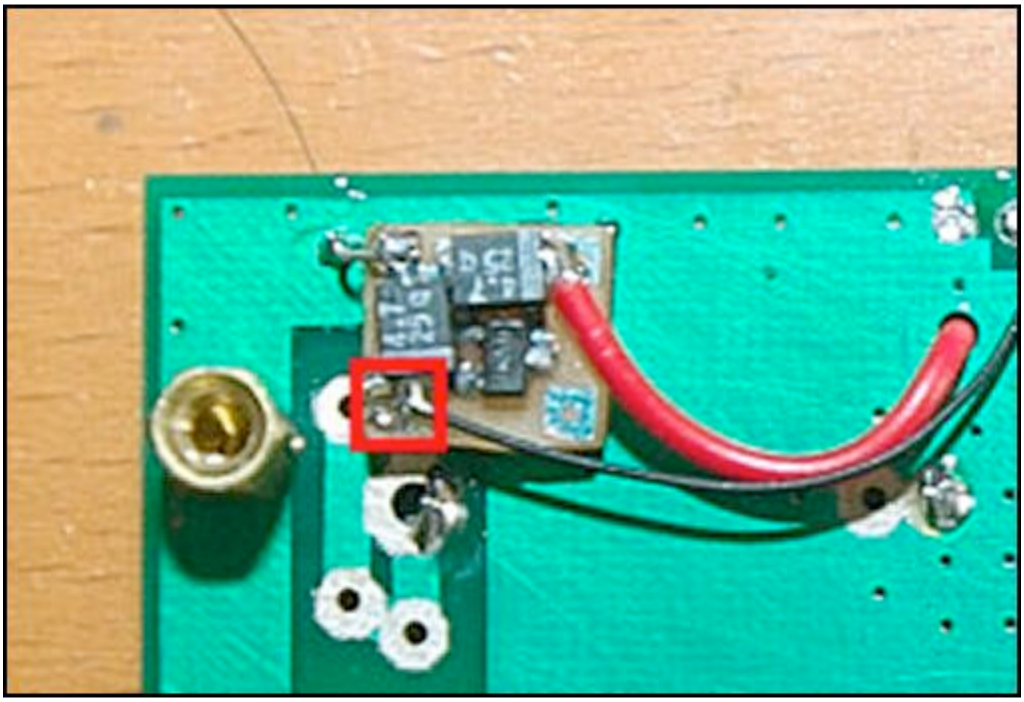
Lastly, everything here is offered in an “as is” condition. Intelligent Flight cannot and will not be held to any liabilities arising from the use of the DragonOSD or derivatives thereof. While everything has been done to ensure reliability and performance, you still use this at your own risk.

### 7.1 Modifying the board to run off 5V

In some cases you may want your board to run off 5V. For example, you may want to use an external 5V regulator that can handle 3S or higher voltages. If that is the case, you will need to bypass the outboard 5V regulator. Doing so will cause the board to take a 5V regulated supply that you supply externally. It will also cause the upper left voltage reading to read the regulated voltage.

To do this, follow the steps as outlined below:-

- Separate the boards by removing the two screws that hold the boards together
- Identify where the red wire goes on the small regulator PCB
- Using a soldering iron, remove the red wire from the top right pad
- Solder the red wire to the pad marked by the red box in the photo below
- Screw the two boards back together securely



That’s it. The red wire represents the power supply line that is going into the board, while the pad marked with the red box is the 5V rail. With this information, you can technically make your own bigger 5V regulator that can handle more than 2S if you so desire.

### 7.2 Modifying the IFOSD Current Sensor to work with DragonOSD

There are several things to note before making this modification. Firstly, the IFOSD sensor uses the ACS755-100 (100A limit) instead of the ACS755-050 (50A limit). While this allows a greater current measurement range, it also reduces current resolution by half. Usually this is not critical since the measurements are not highly precise, and are intended to give us estimations anyway. Secondly, the IFOSD sensor includes a resistor on the sense line.

To use this current sensor, you will need to remove that small surface mounted resistor (marked R300 on the board) and put a short across it. In this way you bring the sensor pins directly out to the cables.

Next, you will need to add an extra wire from tab 5 on the sensor (see diagram of the Allegro sensor above). So, together, there are now 4-wires from the sensor board. Connect these to the board as per the current sensor section (***use the wiring stated in section 4.5*** as the default wiring won't work) and you'll be fine. If necessary, use a continuity meter to ascertain which cable corresponds with which pins on the Allegro sensor.

Lastly, you will need to make some adjustments to the MAHMULT and AMPMULT coefficients (probably increasing them), and possibly the AMPSZERO parameter, in order to re-calibrate the new sensor.

## 8.0 Troubleshooting

This section is still being developed.

*Problem:* The GPS data is taking too long to stablize and the home takes a long time to set, or never at all.
*Resolution:* Try lowering the MINVALID or increasing the VALIDHDOP. Start with increasing the VALIDHDOP to where the GPS has stabilized at (for example to 1.5 instead of the default 1.3).

*Problem:* GPS reading seems to get stuck every now and then.
*Resolution:* Make sure that the GPS module is not too close to the video transmitter or the BOB-4SG board. If possible keep a distance of 10cm or more between the GPS module and the video transmitter and OSD board.

*Problem:* The OSD stops at the "Waiting for GPS data …" screen and does nothing.
*Resolution:* This usually means that the OSD is not seeing data on the GPS incoming line (TX1) and could be inidcative of an intermittent connection. Try to check the connections as well as powering down and then back up again. If you're having this problem when testing with a TV, the leakage current from

the TV can cause corruption of the voltage levels. Grounding the OSD may solve the problem, or simply just test it with an LCD display.

*Problem:* Occasionally the configuration menu does not respond to stick inputs.
*Resolution:* Try adjusting your setting for the control output a bit more. It seems that there is a zone where the control will activate the menu but is marginal enough that the other inputs are not accepted for the selections.

*Problem:* Text at the edges of the screen is getting cut off.
*Resolution:* Try increasing the SAFEZONE parameter in the menu configuration.

## 9.0 Useful Links

The original RC-CAM project development thread. Contains a lot of discussions about the design.

Intelligent Flight's website. You can get the GPS modules with the latest firmware here as well as various other useful knick-knacks.

DPCAV site which also carries GPS modules and replacement GPS cables in case you need one.

dsPIC30F4011 datasheet from MicroChip.

Datasheet for the Etek EB-85A GPS module

Datasheet for the BOB-4 OSD module