BlueFlyVario
Hardware Settings Manual

v1.2 - 23 Jul 2014

Contents

Introduction

The BlueFlyVario is a flight instrument for measuring vertical speed and altitude based on atmospheric pressure. It consists of a simple hardware device that measures the atmospheric pressure 50 times every second, with a resolution that enables the measurement of altitude differences as small as 10 cm.

A number of prototype versions of the BlueFlyVario are available. See www.blueflyvario.com for details.

This manual describes the settings which are stored on the BlueFlyVario, how to set them, and what they control. These settings are known as the hardware settings

Hardware Settings

Overview

The Microchip PIC microcontroller on the BlueFlyVario has an onboard memory (EEPROM) which is used to store hardware settings. The hardware settings are read from the onboard memory when the device is powered on. When a hardware setting is changed by the user the value is used by the onboard code immediately, and the setting is stored in the onboard memory.

Settings can be altered by powering on the BlueFlyVario Hardware and connecting to a host computer or mobile device. The host device needs to run an application which connects to the BlueFlyVario using a serial protocol. This might include any of the following methods:

BFV Hardware	Host Device	Host Device Application
BlueFlyVario (v6, v7 or v8) [Bluetooth Versions]	Android Device	BlueFlyVario app
BlueFlyVario (v6, v7 or v8)	Android Device	Third Party app which supports hardware settings
BlueFlyVario (v6, v7 or v8) or BlueFlyVario_TTL (v8) or BlueFlyVario_TTL_GPS (v9)	Desktop or Laptop computer running Windows or Linux, with an onboard Bluetooth adapter or USB to Serial converter	BFV Desktop Java Application
BlueFlyVario (v6, v7 or v8) or BlueFlyVario_TTL (v8) or BlueFlyVario_TTL_GPS (v9)	Desktop or Laptop computer running Windows or Linux, with an onboard Bluetooth adapter or USB to Serial converter	Terminal Application such as RealTerm (using the raw serial protocol described in the serial communication section)
Any BlueFlyVario hardware which can connect to a device which supports LK8000	Any device which supports LK8000 and can connect to the BlueFlyVario TTL or Bluetooth versions	LK8000 (most recent versions as of Mar 2014)

A comprehensive list of the hardware settings is included as Annex A.

General Settings

A number of general settings are self explanatory:

useAudioWhenConnected (default = false). If true the hardware audio will sound when the device is connected via Bluetooth. *Note: This setting is not used on TTL (non-Bluetooth) versions of the BlueFlyVario.*

useAudioWhenDisconnected (default = true). If true the hardware audio will sound when the device is not connected via Bluetooth. *Note: This setting is used on TTL (non-Bluetooth) versions of the BlueFlyVario as the overall setting to control if audio is on an off.*

positionNoise (default 0.1). This setting controls one of the parameters for the Kalman Filter that is built into the hardware code. A higher value gives less sensitivity by smoothing out the noisy pressure measurements. Try 1.0 to see the difference.

secondsBluetoothWait (default 180). This setting controls how many seconds the hardware will keep its bluetooth radio on while waiting for a connection. *Note: This setting is not used on TTL (non-Bluetooth) versions of the BlueFlyVario.*

greenLED (default = true) [from v9]. This setting controls if the greenLED will light for each lift beep.

Output Modes

From version 7 the BlueFlyVario can output pressure and vario data in a number of modes. This makes the vario compatible with a wider range of applications.

outputMode (default = 0) - Sets the output mode. The available output modes are:

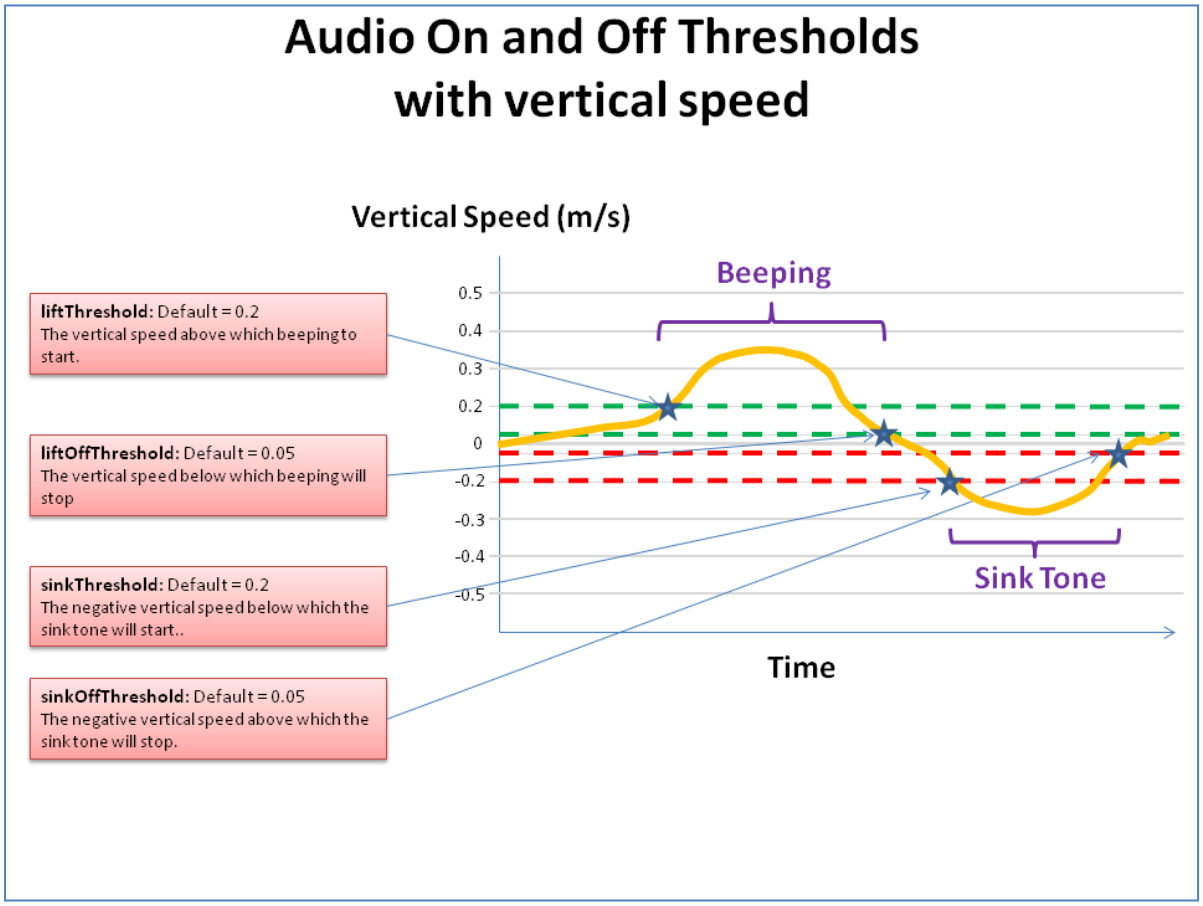
- 0 - The standard BlueFlyVario output mode. This sends raw pressure measurements in the form: "**PRS XXXXX\n**": XXXXX is the raw (unfiltered) pressure measurement in hexadecimal pascals.
- 1 - The LK8EX1 output mode for use with LK8000. This sends pressure and vario data in the form: "**$LK8EX1,pressure,altitude,vario,temperature,battery,*checksum\r\n**": pressure is sent as a decimal integer number of pascals, altitude is not sent (99999 is sent instead), vario is the decimal integer vertical climb rate in cm/s, temperature is in degrees Celsius (1 decimal place), and battery is the battery voltage of the on-board battery (2 decimal places).
- 2 - The LXWP0 output mode for use with a range of apps: **"$LXWP0,loger_stored (Y/N), IAS (kph), baroaltitude (m), vario (m/s),,,,,,heading of plane,windcourse (deg),windspeed (kph)*checksum\r\n** ": The BlueFlyVario only has a partial implementation of this sentence. It only outputs the baroaltitude and vario (all other fields are blank). Note that baroaltidude is determined from filtered pressure using the outputQNH setting.
- 3 - The FlyNet protocol: **"_PRS XXXXX\n**": In this case XXXXX is output as the filtered pressure stream. The filtering parameters used are those from the other hardware settings.
- 4 [from v9 hardware] - No output.

outputFrequency (default = 1). Sets the frequency of output sentences from the BlueFlyVario. The BlueFlyVario hardware runs on a 20ms cycle (50 cycles per second). If ***outputFrequency*** is set to 1 then the hardware will send a sentence on each cycle. If set to 2 it will send a sentence every second cycle and so on (if set to 50 is will send a sentence every 50th cycle, i.e. once per second). You might use this with the LK8EX1 output mode to send a sentence five times per second (set to 10).

outputQNH (default = 101325). See outputMode = 2 above.

Audio Thresholds

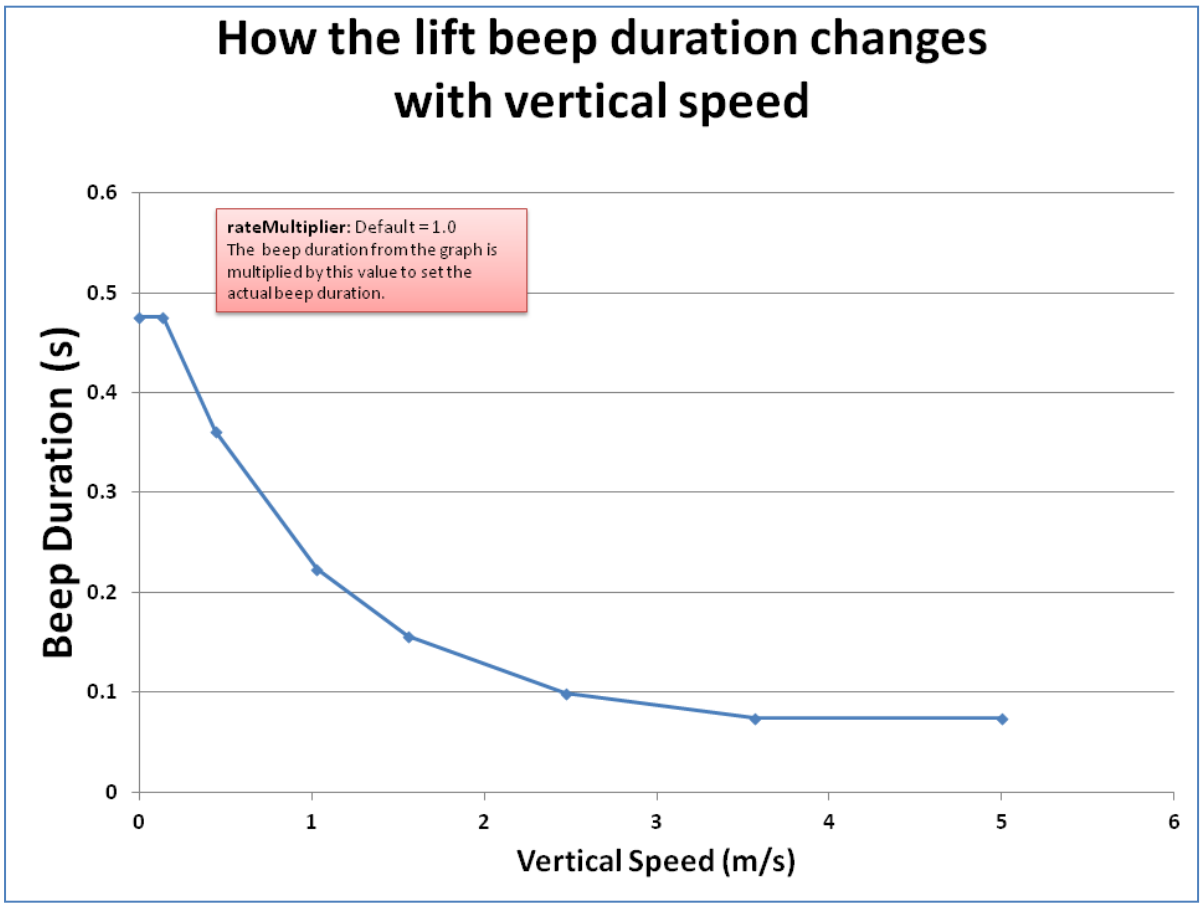
The sound on the BlueFlyVario hardware switches on and off based on the measured vertical speed. The settings ***liftThreshold***, ***liftOffThreshold***, ***sinkThreshold*** and ***sinkOffThreshold*** control when the sound comes on and off. The graph below describes how this works and what the defaults are. It is ok to set the on and off thresholds to the same value, but you will get funny results if the off threshold is higher than the on threshold.



Essentially what happens as you enter lift is that the vario senses a decreasing pressure. Note that there is a slight delay between the movement and the sensed lift due to the way the filtering works. An updated value of filtered lift gets calculated from each pressure measurement, 50 times a second (or every 20 ms). The first time the lift goes above the ***liftThreshold*** a beep commences for a duration controlled by the beep cadence formula (see below). That beep will last for as long as the beep cadence, then a period of silence is commenced, this time based on the most recently measured lift duration. Only at that time 'beeping' could end, but if the lift continues to be above ***liftOffThreshold***, then another sequence of beep followed by silence will be played.

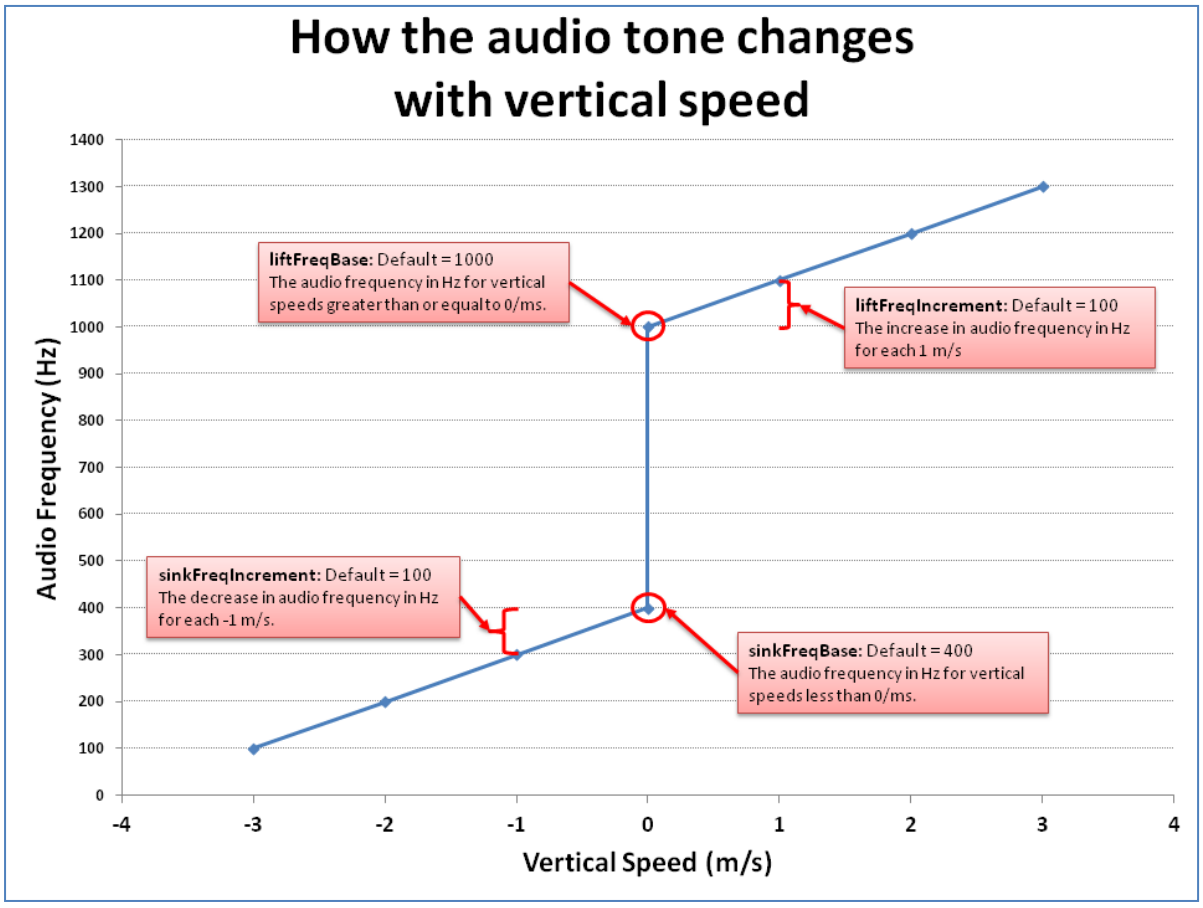
Beep Cadence

When beeping commences a beep is initiated for a duration based on the measured vertical speed. As soon as the beep stops a silent pause will then be initiated, again for a duration based on the measured vertical speed. The graph below shows the formula used to control the beep duration based on the measured lift. Essentially, it means the faster you are going up, the faster the beeps will occur. Note the ***rateMultiplier*** settings can make it go faster as shown below. A ***rateMultiplier*** setting of 0.5 will make it beep twice as fast.



Audio Tone

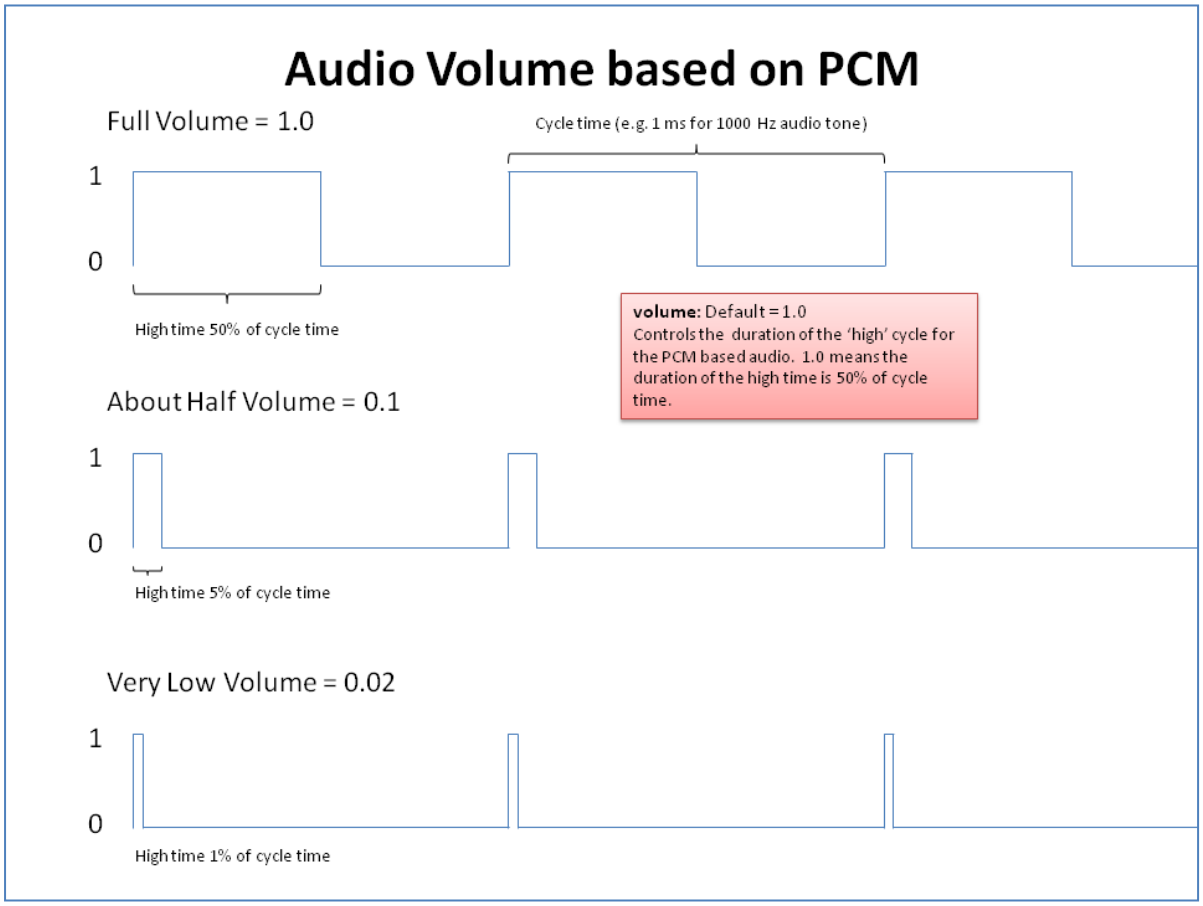
In addition to the lift and sink thresholds and cadence, the audio tone is adjusted based on the measured vertical speed by the four settings ***liftFreqBase***, ***liftFreqIncrement***, ***sinkFreqBase*** and ***sinkFreqIncrement***. The graph below shows how the audio tone (frequency) changes based on these settings, and also shows what the defaults are. The frequency of the sound is constantly being updated to control the tone as the filtered vario value changes. This occurs with every measurement (every 20 ms). The result is changes of tone while beeps are playing based on constantly updated filtered vario values. On occasion the sink tone will sound at the end of a beep if the filtered vario value changes to below the ***sinkThreshold*** while a beep being played.



Note that in the most recent versions of the BlueFlyVario (from v7 onwards), the Audio Frequency cannot be played below 125Hz. If the combination of the settings and the measured vertical speed results in a calculated audio frequency below 125Hz the resulting sound will be clamped to 125Hz.

Volume

The BlueFlyVario uses an electromagnetic transducer. These devices are driven by a square wave from a microcontroller pin (using the inbuilt PWM module of the microcontroller). Using the trick described in the graph below the volume is controlled without needing a variable resistor. Think of the transducer diaphragm being 'kicked' by a high pulse. The more gentle the 'kick' the quieter the sound. The ***volume*** setting is not linear.



Altering the volume setting also subtly alters the audio tone.

Some Notes

The settings selected as defaults are probably not the best for flying, but do allow easy testing of features during assembly. Please provide feedback on what settings work for you and why.

The trick to a well performing vario is to get a combination of ***positionNoise*** and ***liftThreshold*** that works best. If you have almost no filtering (i.e. a low ***positionNoise*** of 0.01) then the calculated vario value will be very noisy. In this circumstance the audio threshold would need to be set at 0.4m/s or higher to avoid errant beeping. These settings would be good if we wanted the vario to be ultra responsive to really jerky movements. However, for flying we bounce around a bit more gently. A ***positionNoise*** of 0.1 with an ***audioThreshold*** of 0.2 seem to work pretty well for most pilots.

Most pilots will probably want to adjust the ***sinkThreshold*** setting to less than their glider sink rate, so it is not on all the time. The ***sinkOffThreshold*** should be adjusted at the same time.

Battery life will be affected by the settings chosen. The largest consumers of power are the Bluetooth radio and the electromagnetic transducer. Having both on full time would reduce battery life to less around 8 hours. Most people want the vario to beep when they are in real lift, and the sink tone to only come on when they are in significant sink (-2.0 m/s or so). This will mean that the audio will only be sounding for less than 50% of most flights, which would give over 10 hours battery life when not connected via Bluetooth.

Serial Protocol

Primary Serial Port

A serial port on the Microcontroller (UART 2) is used to output data from the vario and send commands to it. For Bluetooth versions of the BlueFlyVario the serial port is connected to the onboard RN-42 Bluetooth Adapter. Bluetooth drivers on host devices set up a virtual serial port so the data transfer is seamless. For the BlueFlyVario_TTL the UART Tx and Rx pins are directly exposed.

The settings for the primary serial port are:

Baud Rate:	57600
Data Bits:	8
Stop Bits:	1
Parity:	No Parity
Flow Control:	None
Voltage (for TTL):	Nominally 3.0 Volts

The setting ***uart2BRG*** is used for altering the baud rate of U2 (the main output). This should really only be used with extreme care, but there are a few cases where it is the only way to communicate with a device connected on uart1 (like a GPS) via the microcontroller.
The baud rate is calculated according to the formula:

Baud Rate = 2000000/(uart2BRG+1)

For the default value of 34 this results in a baud rate of ~57143 which is about equal to 57600. The integer setting which is closest to the desired baud rate should be used.

Serial Commands for Settings

Each of the settings described in the Hardware Settings section can be set by sending raw serial commands in ASCII format. The protocol is:

$BXX INT*

The **$** symbol starts a command.

This is followed by the command code **BXX**. A list of all available command codes associated with settings is included in Annex A.

This is followed by the space character.

This is followed by a positive integer **INT** in the range described in the Integer Values section of Annex A. The integer value is converted on the BlueFlyVario in a number of different ways depending on the Type and Factor associated with the setting in the following way:

int:	Converted value = Integer value

Boolean:	Converted value = FALSE if Integer value = 0, and TRUE otherwise
double:	Converted value = Integer value * Factor
int_offset:	Converted value = Integer value + Factor

The * symbol finishes a command.

Other Serial Commands

Serial commands include:

$BST* (BlueFlyVario Settings) - This command is responded to with the following information (in version 8 – there are slightly different responses in earlier firmwares):

BFV [*VersionNumber*] \r\n
BST [*followed by a space separated list of each of the settings codes*]
SET [*followed by a space separated list of each of the settings Integer Values*]

$TMP* (Temperatue) - This command is responded to with the following information:

TMP [*Integer temperature value * 10*] \r\n

$RST* (Reset) [from v9] - This command resets the module, essentially a hot reboot.

$RSX* (Reset Settings) [from v9] - This command resets the module and reset all of the hardware settings to defaults. It is an extended hot reboot, equivalent to start up with programming pads 2 and 4 shorted.

$SLP* (Sleep)[from v9]. This command sends the module to sleep mode. Note it also sends commands over U1 to send the the PA6H GPS to sleep mode using the PMTK standby command. The microcontroller plus pressure sensor consumes about 0.05mA in sleep mode, but the GPS still consumes about 1.5mA. To wake from sleep mode without a power cycle you just send any character to the module on U2. This will then force a hot reboot, and you are back to the power on state, including the GPS.

$SLX* (Sleep)[from v9 (rev2)]. This command sends the module to sleep mode. Note it also sends commands over U1 to send the the PA6H GPS to sleep mode using the PMTK standby command. The microcontroller plus pressure sensor consumes about 0.05mA in sleep mode, but the GPS still consumes about 1.5mA. In this sleep mode the module cannot be woken with a serial command. The devices needs to be power cycled to wake it up.

Second Serial Port

From version 8 a second serial port is available as an input, and from version 9 you can also send data to it. This serial port exposes UART1 from the PIC Microcontroller. The key parameters are:

Baud Rate:	9600 (by default – see ***uart1BRG*** below)
Data Bits:	8
Stop Bits:	1
Parity:	No Parity

Flow Control: None
Voltage (for TTL): Nominally 3.0 Volts

The second serial port is set up to receive NMEA GPS sentences and send them out on UART2 (the standard output). In this way the BlueFlyVario effectively multiplexes the GPS data with the standard output controlled by the ***outputMode***. This allows the BlueFlyVario to output a single stream of data which can be read by XCSoar or other applications.

Data received on the Rx line of UART1 is processed in the following way (if the setting ***uart1Raw*** is false). Any series of ASCII characters starting with **$** and ending with **\n** are recorded as a sentence. The sentence is then scheduled for transmission at the next available 20 ms cycle. Note that it is probably possible to overload the device with too many sentences. Standard GPS output at 1Hz (with about five sentences sent each time) seems to work well.

The setting ***uart1BRG*** (default = 207) controls the baud rate on UART1. The baud rate is calculated according to the formula:

Baud Rate = 2000000/(uart1BRG+1)

For the default value of 207 this results in a baud rate of ~9615 which is about equal to 9600. The integer setting which is closest to the desired baud rate should be used.

From version 9 additional hardware settings control how data is passed between UART1 and UART2:

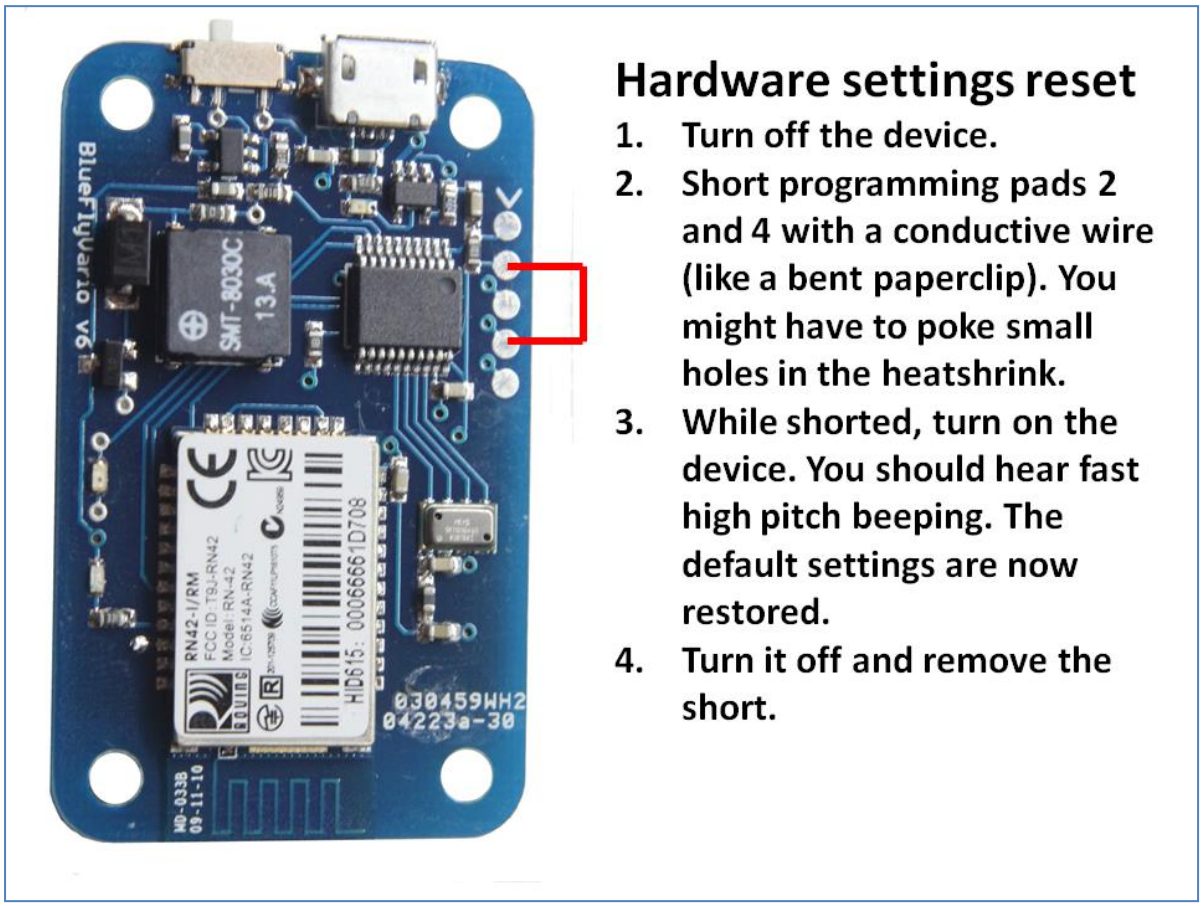
uartPassthrough (default = true) - This setting is used for passing characters received on U2 through the micro to U1. Due to different baud rates (U2 > U1) the characters are stored in a FIFO buffer of length 100 which is long enough to allows for normal commands to be sent to devices like a GPS. If you need to send bulk data into U2 and have it sent to U1, like a firmware update for a GPS for example, then you will need to adjust the baud rates of U1 and U2 so the buffer does not overflow.

uart1Raw (default = false). Normally sentences received on U1 are processed as described above. When uart1Raw is set to true the data is processed on a character by character basis. Characters received at U1 are passed straight through to U2.. This is used to allow bulk transfer of information that is sent by GPS other than standard NMEA sentences. If you set this to true be aware of the different baud rates (if you have set U1 > U2 then you will get buffer overflows), and also be aware that if the ***outputMode*** is anything other than mode 4 (off) then you will get a very unintelligent multiplexing of the U1 received characters with the pressure output.

Reset Hardware Defaults

If you really screw up the settings, and for some reason you cannot fix them through the serial connection, then it is possible to reset the hardware settings to their default values using the procedure shown below. You might have to do this if the ***bluetoothWaitTime*** gets set to less than the amount of time required to establish a connection.

If programming pad 4 (the forth pad from the arrow which indicates pad 1) is high when the device is turned on, then the reset to default sequence will be entered. Pad 2 is the positive voltage signal so shorting pads 2 and 4, then powering on the device, will achieve hardware reset.



Annex A - Technical Description of Settings

Name	BFV Version	Code	Type	Factor	Integer Values			Converted Values			Message
					Min Value	Max Value	Default Value	Min Value	Max Value	Default Value	
useAudioWhenConnected	6	BAC	boolean	1	0	1	0	FALSE	TRUE	FALSE	Check to enable hardware audio when connected.
useAudioWhenDisconnected	6	BAD	boolean	1	0	1	1	FALSE	TRUE	TRUE	Check to enable hardware audio when disconnected.
positionNoise	6	BFK	double	1000	10	10000	100	0.01	10	0.1	Kalman filter position noise.
liftThreshold	6	BFL	double	100	0	1000	20	0	10	0.2	The value in m/s of lift when the audio beeping will start.
liftOffThreshold	6	BOL	double	100	0	1000	5	0	10	0.05	The value in m/s of lift when the audio beeping will stop.
liftFreqBase	6	BFQ	int	1	500	2000	1000	500	2000	1000	The audio frequency for lift beeps in Hz of 0 m/s.
liftFreqIncrement	6	BFI	int	1	0	1000	100	0	1000	100	The increase in audio frequency for lift beeps in Hz for each 1 m/s.

Name	BFV Version	Code	Type	Factor	Integer Values			Converted Values			Message
					Min Value	Max Value	Default Value	Min Value	Max Value	Default Value	
sinkThreshold	6	BFS	double	100	0	1000	20	0	10	0.2	The value in -m/s of sink when the sink tone will start.
sinkOffThreshold	6	BOS	double	100	0	1000	5	0	10	0.05	The value in -m/s of sink when the sink tone will stop.
sinkFreqBase	6	BSQ	int	1	250	1000	400	250	1000	400	The audio frequency for the sink tone in Hz of 0 m/s.
sinkFreqIncrement	6	BSI	int	1	0	1000	100	0	1000	100	The decrease in audio frequency for sink tone in Hz for each -1 m/s.
secondsBluetoothWait	6	BTH	int	1	0	10000	180	0	10000	180	The time that the hardware will be allow establishment of a bluetooth connection for when turned on.
rateMultiplier	6	BRM	double	100	10	100	100	0.1	1	1	The lift beep cadence -> 0.5 = beeping twice as fast as normal.
volume	6	BVL	double	1000	1	1000	1000	0.001	1	1	The volume of beeps -> 0.1 is only about 1/2 as loud as 1.0.
outputMode	7	BOM	int	1	0	3	0	0	3	0	The output mode -> 0-BlueFlyVario(default), 1-LK8EX1, 2-LX, 3-FlyNet.

Name	BFV Version	Code	Type	Factor	Integer Values			Converted Values			Message
					Min Value	Max Value	Default Value	Min Value	Max Value	Default Value	
outputFrequency	7	BOF	int	1	1	50	1	1	50	1	The output frequency divisor -> 1-every 20ms ... 50-every 20msx50=1000ms
outputQNH	7	BQH	int_offset	80000	0	65535	21325	80000	145535	101325	QNH (in Pascals), used for hardware output alt for some output modes - (default 101325)
uart1BRG	8	BRB	int	1	0	655535	207	0	655535	207	BRG setting for UART1, baud = 2000000/(BRG-1) (default of 207 = approx 9600 baud)
uart2BRG	9	BR2	int	1	0	655535	207	0	655535	207	BRG setting for UART2, baud = 2000000/(BRG-1) (default of 207 = approx 9600 baud)
uartPassthrough	9	BPT	boolean	1	0	1	1	FALSE	TRUE	TRUE	Check to enable characters received on U2 to be passed through to U1.
uart1Raw	9	BUR	boolean	1	0	1	0	FALSE	TRUE	FALSE	Check to make characters received at U1 are passed straight through to U2 rather than passed through line by line (i.e. $...<LF>) and then multiplexed with pressure output.
greenLED	9	BLD	boolean	1	0	1	1	FALSE	TRUE	TRUE	Check turn on the green LED with each lift beep.