WAX9 Application Developer's Guide

A guide for application development using the WAX9 platform

Contents

Reference Tables

Introduction

This document describes the interface protocol between the host application and a WAX9 wireless sensor. It is written to help the software developer quickly gain access to the device features and to answer the common questions that are asked when interfacing to a WAX9.

Control and Interface

After the device has been charged and disconnected from the USB cable for the first time it will quickly become discoverable as a Bluetooth device called WAX9-XXXX where XXXX is the device identifier (device id). If the device id is left as its default setting then the XXXX becomes the last 4 hexadecimal characters of the Bluetooth media access control (MAC) address. The default pairing code for standard Bluetooth pairing will be "0000". For low energy (LE) connections, the device is in 'Just works' configuration and will accept one connection at a time. Simultaneous standard Bluetooth and LE connections are not permitted at the time of writing; Future firmware releases may support additional features. Future releases are planned to add secure pairing for LE connections as well.

RFCOMM Control Interface

Control of the device can be performed over an RFCOMM channel which can be established by pairing to the device and opening a port. Each operating system (OS) offers many different options for using communication (COM) ports, however, it is recommended to initially connect to the device using a terminal program such as hyperterminal to familiarise the user with the interface (other options for terminals). Some of the OS's application programmers interfaces (APIs) used to control COM ports may not function as expected using RFCOMM. The terminal application may ask for baud settings, these are not applicable to RFCOMM ports and the default settings should be used (usually 8 bit, no parity, 1 stop bit or abbreviated "8N1"). Open connected to the device, type a command such as "`settings<CR>`"; where `<CR>` is the carriage return character, usually the return key. The device will respond in plain text to display its current settings. "`sample<CR>`" will display the sampled sensor data in the format described in the data output section.

Bluetooth Low Energy Attribute Control

The Bluetooth low energy (LE) mode of the device shares some of the characteristics of the standard RFCOMM interface by exposing a serial communication channel for commands and responses. The LE control mode also allows direct manipulation of the device settings and lacks the ability to stream raw text data (due to the reduced bandwidth). The profile of the device can be freely explored using an LE attribute explorer on a LE enabled device. Skip to the LE communication section to get started quickly with the LE modes of operation.

Common Initial Issues

Device not visible: The devices are shipped in hibernate mode and are not discoverable. Connect to the USB cable and charge the device, on disconnecting the device will be discoverable.

Cannot pair with device: There may be an active connection already formed that was not closed during device discovery. Briefly connect to USB and disconnect again to reset the device; then retry. For LE pairing issues, see the LE communication section.

Cannot open a COM port: Firstly, view the device using the OS (e.g. device manager) to confirm that the OS has successfully installed the device and designated it a COM port number. Select the assigned COM port number using the terminal program and then connect. If this fails, try another terminal program; some terminal programs are not compatible with RFCOMM; try teraterm, a free open source terminal project.

No response from device: For some OSs, the device may have formed more than one COM port for the device to expose the RFCOMM control channel; the data channel is usually port with the lower number. Also, check the behaviour of the return key for the terminal used; it may send a line feed (<LF>) instead of <CR>.

LED Indicator

The WAX9 has a tri-colour LED indicator capable of several colour outputs. If a transparent packaging option is chosen, the indicator colours and flashes may be used to identify the modes of the device an aid in device identification. There are two main modes of operation which each have different indicator outputs.

USB or Charging mode

The following table describes how the device indicates its status when connected to a charger or host PC.

Indicator LED	Mode	Note
Flashing red	Charging , pre-charge	Battery too low to bootload
Solid colour red	Bootloading active	Do not disconnect
Flickering	Bootloading active	Do not disconnect
Solid colour red	Charging, battery <10%	Gradual red to green change
Solid colour yellow	Charging, battery >10%	Gradual red to green change
Solid colour green	Charging, battery >80%	Gradual red to green change
Flashing green	Charging, battery full	Fully charged

Table 1. LED codes, USB connected

Not Charging

The following table describes how the device indicates its status when not charging.

Indicator LED	Mode	Note
Flashing red	Battery empty	Not discoverable, charge
Solid yellow	Starting	Startup takes ~1 second
Flashing yellow	Device discoverable	Low duty cycle flash at 2 Hz
Flashing blue	Device connected	Low duty cycle flash at 2 Hz
Flashing blue fast	Device streaming	Flashes at data output rate
Flashing green	Device hibernating	Flashes once per 4 seconds
Other colours	LED override set	LED command issued

Table 2. LED codes, USB not connected

Error Codes

LED error codes are as follows.

Indicator LED	Mode	Note
Flashing magenta	Radio fail	~4s, wrong firmware or damaged
Flashing cyan	Sensor fail	~4s, sensors damaged
Magenta 3s + reset	Device exception	Please send bug report

Table 3. Error LED codes

Serial Data Output

The device has three serial data output modes:

Single Sample

This is the response to the sample command and is always in text mode including a comma separated value (CSV) list of the variable positions. This mode is intended for sparsely point sampled data or debugging; This mode is also available using the LE serial port connection.

e.g. DATA: N,Ax,Ay,Az,Gx,Gy,Gz,Mx,My,-Mz,Batmv,Temp0.1C,PresPa,Ia<CR><LF>
0,101,-25,4050,12,-61,37,-2078,187,3698,3890,205,100257,0<CR><LF>

Text Stream

The text stream data output is in the same format as the single sample (without variable position list) and is in plain text (UTF-8). The data rate is set by the RATEX variable in samples per second (default 50 Hz). The sample number (first variable in the list) should be used to identify if samples are missing (e.g. due to the device going out of range briefly). The default behaviour (DATAMODE variable is zero) is to only transmit the extended data (battery, temperature, pressure and inactivity count) when it is sampled; approximately twice per second. Text streams are not available for LE connections.

e.g.

Normal: **0,101,-25,4050,12,-61,37,-2078,187,3698<CR><LF>**

Long: **0,101,-25,4050,12,-61,37,-2078,187,3698,3890,205,100257,0<CR><LF>**

Binary Stream

This is the recommended operating mode for high performance systems. The sensor data is transmitted in raw binary using slip encoding (RFC 1055), this

allows the framing of the data to be extracted. Binary stream mode offers greatly reduced channel usage and hence higher sample rates. LE connections use a different binary format as described in the LE section of this document. The binary format is:

Byte *	Value hex**	Note
0	0xC0	SLIP END
1	0x39	Ascii '9'
2	0x01 or 0x02	Packet format
3+4	0xXXXX	Sample number
5+6+7+8	0xXXXXXXXX	Sample timestamp***
9+10, 11+12, 13+14	0xXXXX, 0xXXXX, 0xXXXX	Accelerometer values, 16 bit signed integers, order x,y,z
15+16, 17+18, 19+20	0xXXXX, 0xXXXX, 0xXXXX	Gyroscope values, 16 bit signed integers, order x,y,z
21+22, 23+24, 25+26	0xXXXX, 0xXXXX, 0xXXXX	Magnetometer values, 16 bit signed integers, order x,y,z
27+28****	0xXXXX	Battery voltage in millivolts
29+30****	0xXXXX	Signed temperature in 0.1˚C steps
31+32+33+34 ****	0xXXXXXXXX	Unsigned barometric pressure in pascals (Pa), see altitude conversion
27 or 35	0xC0	SLIP END

Table 4. Binary slip encoded data format (RFCOMM)

** Multi byte fields are ordered little-endian (LE) or least significant byte first.*
*** Bytes equal to slip codes are replaced with the RFC 1055 escape sequence.*
**** Timestamps are in 1/65536 of a second.*
***** For packet formats of 0x02 only, otherwise absent.*

Serial Commands Summary

This section covers the serial commands that can be used to configure and customise the device. All serial commands are text based, case insensitive and terminated with a <CR> symbol. The user may also send a <LF> character which will be ignored. Some commands using an RFCOMM connection can be concatenated using the '|' symbol (0x7C) to replace the <CR> in this case the maximum command length is 64 characters. For LE connections, concatenation is not possible, the command length must be 20 characters maximum and the <CR> does not have to be added. However, LE connection may directly read and write the settings controlled by these serial commands as described in the LE section of this document. Some commands may cause the device to lose communication with the host computer and stall the COM port, if the device becomes unresponsive or un-connectable there are four options:

- The "clear<CR>" command restores default settings, the "reset<CR>" command will reset the device and load these settings.
- Connecting the device to a charger or host using the USB will reset the device and allow reconnection.

- Boot loading the firmware to the device again will factory reset the device. This is covered in a preceding section.
- If the device will not reset to bootloader, leave it disconnected. An internal timeout will reset it after approximately one minute.

Terminating Commands

These commands can not be followed by additional commands. It is assumed that the text is followed by a <CR> after the arguments, the device will immediately execute the command(s).

Command	Action
"help"	Accepts no arguments
	Prints all commands in capitals separated by "<CR><LF>"
"echo"	Accepts no arguments or "=0" or "=1" e.g. "echo=1<CR>"
	Prints "ECHO: %u<CR><LF>" where %u is 0 or 1 depending on setting
"reset"	Accepts no arguments, preceding setting commands are lost
	Prints "RESET:<CR><LF>", closes connection and resets
"sample"	Accepts no arguments
	Prints two text strings containing the current sensor data, this is detailed in the data output section of this document
"stream"	Accepts no arguments, can be preceded by settings
	Will stream data if sample rate > 0 and sensors are enabled
"name"	Accepts either no terms or string to be used as the device name e.g. "name=newname<CR>"
	Prints "NAME: new name<CR><LF>"
	Note that the device id will be appended to the Bluetooth name
"pin"	Accepts no terms or 4 digit number string to be used as the pin code e.g. "pin=1234<CR>"
	Prints "PIN: 1234<CR><LF>"

Table 5. Terminating commands

Group-able Commands

These commands may be concatenated into a single command up to 64 characters long total.

Command	Action
"led"	Accepts "%d<CR>", -1 LED under normal device control, 0 - 7 is binary LED value to be set e.g. "led=3<CR>", 3 is binary 0b011, RGB LED is set to red=0, green=1, blue=1 yielding cyan
	Prints "LED: %d<CR><LF>"
"device"	Accepts numbers 1-65534 to become new device id. 0 and 65536 are invalid and the device will adopt the last two bytes of the Bluetooth MAC address (little endian) as its id on reset.
	Prints all settings (see below format)
"clear"	Accepts no arguments, sets all settings to defaults
	Prints all settings (see below format)
"settings"	Accepts no arguments
	Prints all settings (see below format)
"datamode"	Accepts number to become the new datamode setting, see variable summary (unsupported modes will result in no streamed data)
	Prints all settings (see below format)

Command	Action
"sleepmode"	Accepts number to become new sleep mode setting, see variable summary (sleep mode 4 requires reset to latch)
	Prints all settings (see below format)
"inactive"	Accepts numbers 3-65534 to become the inactivity count threshold in seconds, see variable summary for sleepmode
	Prints all settings (see below format)

Table 6. Group-able commands (RFCOMM)

Command	Action				
"rate"	Accepts 4 arguments "%c %u1 %u2 %u3" (space or comma separated), see variable summary for valid ranges:				
	Variable	%c	%u1	%u2 (or omit)	%u3 (or omit)
	Output rate	"x"	"0" (or omit)	"0"	New rate
	Accel setting	"a"	On = 1 Off = 0	Internal rate	Sensor range
	Gyro setting	"g"	On = 1 Off = 0	Internal rate	Sensor range
	Mag setting	"m"	On = 1 Off = 0	Internal rate	"0"
	e.g. "rate x 0 0 10<CR>" sets the output data rate to 10 Hz e.g. "rate a 1 200 8<CR>" sets the accelerometer on with 200 Hz internal rate and +/-8g range. e.g. "rate g 0|rate x 10|stream<CR>" (26 characters) turns off the gyro and starts streaming at 10 Hz				
	Prints all settings (see below format)				

Table 7. Sensor settings (rate) command

All Settings Output Print Format

Command	Action
"WAX9, HW: %s, FW: %s, CS: %s <CR><LF> ID: %u<CR><LF>"	%s Hardware version
	%s Firmware version
	%s Chipset (CC2564 / CC2560)
	%u Device id
"NAME: %s, PIN: %s<CR><LF>"	%s Bluetooth name
	%s Bluetooth pin
"MAC: %02X:%02X:%02X:%02X:%02X:%02X<CR><LF>"	XX:XX:XX:XX:XX:XX Bluetooth MAC address
"ACCEL: %u, %u, %u<CR><LF>"	%u 1 = on, 0 = off
	%u Accelerometer rate (Hz)
	%u Accelerometer range (g)
"GYRO: %u, %u, %u<CR><LF>"	%u 1 = on, 0 = off
	%u Gyroscope rate (Hz)
	%u Gyroscope range (dps)
"MAG: %u, %u<CR><LF>"	%u 1 = on, 0 = off
	%u Magnetometer rate (Hz)
"RATEX: %u<CR><LF>"	%u Output data rate (Hz)
"DATA MODE: %u%c<CR><LF>"	%u Output data mode
	%c '!' only if setting requires high power operation

Command	Action
"SLEEP MODE:%u<CR><LF>"	%u Sleep mode setting
"INACTIVE:%usec<CR><LF>"	%u Inactivity timeout value

Table 8. Settings output response format

Sensor Settings

The sensors on the WAX9 are listed in the device datasheet, they are all digital sensors with their own independent sample clocks. This makes synchronising the sample point across all sensors impossible and can only be solved by two solutions:

1) Allow each sensor to have its own stream and send packeted data over the port separately. This makes synchronisation difficult since the sensors will have slightly different sample rates (e.g. 101 Hz and 98 Hz) resulting in changes of packet ordering on the channel. Secondly, if the samples are sent sample by sample, each packet must also have a timestamp and identifier resulting in a large additional data flow and potential channel contention. Thirdly, interpolation must be used on the receiver side to re-synchronise the streams before processing, thus adding latency and processing time.

2) The sensors can be configured to oversample at a higher rate than the sample clock and are then re-sampled synchronously using a precision clock. This offers very accurate sample rates, minimal signal latency and synchronous sensor data; This is the method employed by the WAX9. The disadvantage is that a timing error up to the sensor internal sample interval may be present. E.g. If the accelerometer and gyroscope have internal sample rates set to 200 Hz and the output data rate is 100 Hz, a sample may contain accelerometer and gyroscope sample up to 5 ms (1/200 Hz) old or up to 5 ms apart; For some applications, it is necessary to use high oversample ratios to remove this error at the expense of slightly shorter battery life. For some applications, an under sample scheme can be employed to reduce the data flow from that sensor (e.g. magnetometer defaults to 10Hz rate).

The sensors each have their own independent internal sample rates because of the sampling scheme described above. The sampling rates may be independent of the output data rate. This table summarises the settings available for each sensor. Using variables outside this table will result in the default value being chosen.

Variable	Values	Effect
Accelerometer on	1 or 0	On or Off resp.
Accelerometer rate	12, 50, 100, 200, 400, 800	Internal rate Hz
Accelerometer range	2, 4, 8	Range in +/-g
Gyroscope on	1 or 0	On or Off resp.
Gyroscope rate	100, 200, 400, 800	Internal rate Hz
Gyroscope range	250, 500, 2000	Range in dps
Magnetometer on	1 or 0	On or Off resp.
Magnetometer rate	5, 10, 20, 40, 80	Internal rate Hz

Table 9. Valid sensor setting values

Other Variables

Data Mode Setting

The data mode setting determines the format of the serial data sent over the RFCOMM channel. Sensor data is sent as text mode or slip-encoded binary. LE connections use a different transmission format irrespective of the data mode. The following summarises the currently supported sub-modes:

Value of datamode	Data output format*
0	Ascii mode with batt, temp, pressure, inactivity transmitted as they are sampled (~1Hz).
128	Ascii mode with batt, temp, pressure, inactivity transmitted every packet.
1	Binary mode with ~1Hz batt, temp, pressure, inactivity update. Packet types 1 and 2
129	Binary mode with continuous batt, temp, pressure, inactivity update. Packet type 2 only.

Table 10. RFCOMM data mode values

** Not applicable to LE connections*

Sleep Mode Setting

The sleep modes of the device allow significant power saving when the device is not being used yielding longer battery life.

Value of sleepmode	Behaviour
0	Highest power, allows slightly faster re-connects and can be powered over USB instead of re-setting (once).
1	Normal mode, never hibernates, low power discoverable.
2	Hibernates after inactivity timer reaches setting (inactive value) without movement being detected.
3	Hibernates after inactivity timer reaches setting (inactive value) without orientation changing.
4	Hibernate upon disconnect until next USB connect, then change to sleep mode 1. This is shipping mode also.
5	This is a test mode used during manufacture. If set, connect and disconnect from the USB twice to exit.

Table 11. Sleep mode settings

Low Energy Connections

The WAX9 can be accessed using the Bluetooth low energy attribute protocol. This section summarises the attribute profile and its various features. The attribute protocol is briefly described here.

The Attribute Protocol

In its simplest form, the attribute protocol describes a set of locations (handles) and associates descriptions (**<Characteristic User Description>**), characteristics (**<Characteristic>**) and client settings (**<Client Characteristic Configuration>**). These are clustered into services (**<Primary Service>, <Secondary Service>**) of which some are mandatory and some are optional. Each of the above fields in **<angled brackets>** is known as an attribute and every attribute has a universally

unique identifier (UUID). The common UUIDs are assigned by the Bluetooth special interest group (SIG) and are 16 bits. Custom UUIDs must all be 128 bits. The following tables describes the WAX9 profile UUIDs; **<angled brackets>** are Bluetooth SIG UUIDs and **{curly brackets}** are custom 128 bit UUIDs, [sqare brackets are notes] and (normal brackets) show the flags field for characteristics; Colour coding helps identify the main attribute types.

ATTRIBUTE UUID	TYPE	VALUE
0x2800	<Primary Service>	Service type UUID
0x2801	<Secondary Service>	Service type UUID
0x2803	<Characteristic>	Value flags (8), Value handle (16), Value UUID (16/128)
0x2901	<Characteristic User Description>	String (UTF8)
0x2902	<Client Characteristic Configuration>	Alert setting (16)
0x2A00	<Device Name>	String (UTF8)
0x2A01	<Appearance>	Appearance (16)
0x2A05	<Service Changed>	Range Start (16), End handle (16)
0x2A27	<Hardware Version>	String (UTF8)
0x2A26	<Firmware Version>	String (UTF8)
0x2A19	<Battery State>	Percentage (8)

Table 12. Bluetooth SIG assigned attribute UUIDs in the WAX9 profile

SERVICE TYPE UUIDs	TYPE (follow <Primary Service>)
0x1800	<Generic Access Profile>
0x1801	<Generic Attribute Profile>
6E400001-B5A3-F393-E0A9-E50E24DCCA9E	Custom service {Serial port}, compatible with this Android and iOS application from Nordic Semi.
00000000-0008-A8BA-E311-F48C90364D99	Custom service {Sensor data and command}
00000005-0008-A8BA-E311-F48C90364D99	Custom service {Service settings and status}

Table 13. Attribute protocol services on the WAX9

ATTRIBUTE VALUE UUIDs	TYPE	PERMISSIONS
6E400002-B5A3-F393-E0A9-E50E24DCCA9E	{Serial data input}	WRITE NR*
6E400003-B5A3-F393-E0A9-E50E24DCCA9E	{Serial data output}	INDICATE, NOTIFY
00000001-0008-A8BA-E311-F48C90364D99	{Enumerated command input}	WRITE, WRITE NR*
00000002-0008-A8BA-E311-F48C90364D99	{Sensor data} accelerometer, gyroscope, magnetometer	INDICATE, NOTIFY, READ
00000004-0008-A8BA-E311-F48C90364D99	{Meta data} altimeter, temperature, battery	INDICATE, NOTIFY, READ
00000006-**	{Streaming}	READ
00000007-**	{LED}	READ, WRITE, WRITE NR*
00000008-**	{Sleep mode}	READ, WRITE, WRITE NR*
00000009-**	{Inact. threshold}	READ, WRITE, WRITE NR*
0000000A-**	{Sample rate}	READ, WRITE, WRITE NR*
0000000B-**	{Accel. on}	READ, WRITE, WRITE NR*
0000000C-**	{Accel. rate}	READ, WRITE, WRITE NR*
0000000D-**	{Accel. range}	READ, WRITE, WRITE NR*
0000000E-**	{Gyro. on}	READ, WRITE, WRITE NR*
0000000F-**	{Gyro. rate}	READ, WRITE, WRITE NR*
00000010-**	{Gyro. range}	READ, WRITE, WRITE NR*
00000011-**	{Mag. on}	READ, WRITE, WRITE NR*
00000012-**	{Mag. rate}	READ, WRITE, WRITE NR*

Table 14. UUIDs of the values used in the WAX9 profile

** WRITE NR is write with no response, also known as write command*
*** The remaining UUID is truncated for readability - 0008-A8BA-E311-F48C90364D99*

The WAX9 Attribute Profile

This section lists the attribute profile ordering with the handle numbers used. However, most systems have an API to explore devices and discover services. Since the handle numbers are liable to change in future revisions of the the firmware, the user should aim to use a robust service discovery method using only the UUID values.

Handle	UUID	VALUE
0x0001	<Primary Service>	<Generic Access Profile>
0x0002	<Characteristic>	(READ) <Device Name>
0x0003	<Device Name>	"WAX9"
0x0004	<Characteristic>	(READ) <Appearance>
0x0005	<Appearance>	0x0000
0x0010	<Primary Service>	<Generic Attribute Profile>
0x0011	<Characteristic>	(INDICATE|READ|WRITENR) <Service Changed>
0x0012	<Service Changed>	0x0020,0xFFFF
0x0013	<Characteristic>	(READ) <Hardware Version>
0x0014	<Hardware Version>	"1.0"
0x0015	<Characteristic>	(READ) <Firmware Version>
0x0016	<Firmware Version>	"3.0"
0x0020	<Primary Service>	{Serial port}
0x0021	<Characteristic>	(WRITENR) {Serial data input}
0x0022	{Serial data input}	[up to 20 character input]
0x0023	<Characteristic User Description>	"Serial input"
0x0024	<Characteristic>	(INDICATE|NOTIFY) {Serial data output}
0x0025	{Serial data output}	[response, as indications or notifications]
0x0026	<Client Characteristic Cfg.>	0xXXXX [Write: 0=Off, 1=Notify, 2=Indicate]
0x0027	<Characteristic User Description>	"Serial output"
0x0030	<Primary Service>	{Sensor data and command}
0x0031	<Characteristic>	(READ) <Battery State>
0x0032	<Battery State>	0xXX [in percent]
0x0040	<Characteristic>	(WRITE|WRITENR) {Enumerated command input}
0x0041	{Enumerated command input}	0xXXXX
0x0042	<Characteristic User Description>	"Command: SAVE=0|STREAM=1|RESET=2|LED=3|CLEAR=4|STOP=5"
0x0050	<Characteristic>	(READ|INDICATE|NOTIFY) {Sensor data}
0x0051	{Sensor data}	[see output data format section]
0x0052	<Client Characteristic Cfg.>	0xXXXX [Write: 0=Off, 1=Notify, 2=Indicate]
0x0053	<Characteristic User Description>	"Data output"
0x0070	<Characteristic>	(READ|INDICATE|NOTIFY) {Meta data}
0x0072	{Meta data}	[see output data format section]
0x0071	<Client Characteristic Cfg.>	0xXXXX [Write: 0=Off, 1=Notify, 2=Indicate]
0x0073	<Characteristic User Description>	"Pressure Pa (uint32), Temp x10C (sint16), Battery mV (uint16)"
0x0080	<Primary Service>	{Service settings and status}
0x0081	<Characteristic>	(READ) {Streaming}
0x0082	{Streaming}	0xXX [Streaming status on/off]
0x0083	<Characteristic User Description>	"Streaming on/off"
0x0090	<Characteristic>	(READ|WRITE|WRITENR) {LED}
0x0091	{LED}	0xXX [LED setting, see command section]
0x0092	<Characteristic User Description>	"LED setting"
0x00A0	<Characteristic>	(READ|WRITE|WRITENR) {Sleep mode}
0x00A1	{Sleep mode}	0xXX [Sleep setting, see sensor settings]
0x00A2	<Characteristic User Description>	"Sleep mode setting"
0x00B0	<Characteristic>	(READ|WRITE|WRITENR) {Inact. threshold}
0x00B1	{Inact. threshold}	0xXXXX [Inactivity val, see sensor settings]
0x00B2	<Characteristic User Description>	"Inactivity timeout"
0x00C0	<Characteristic>	(READ|WRITE|WRITENR){Sample rate}
0x00C1	{Sample rate}	0xXXXX [Sample rate, see sensor settings]
0x00C2	<Characteristic User Description>	"Sample rate Hz"
0x00D0	<Characteristic>	(READ|WRITE|WRITENR) {Accel. on}
0x00D1	{Accel. on}	0xXXXX [Accel. on/off, see sensor settings]

0x00D2	<Characteristic User Description>	"Accel. on/off"
0x00E0	<Characteristic>	(READ|WRITE|WRITENR) {Accel. rate}
0x00E1	{Accel. rate}	0xXXXX [Accel. rate, see sensor settings]
0x00E2	<Characteristic User Description>	"Accel. rate Hz"
0x00F0	<Characteristic>	(READ|WRITE|WRITENR) {Accel. range}
0x00F1	{Accel. range}	0xXXXX [Accel. range, see sensor settings]
0x00F2	<Characteristic User Description>	"Accel. range g"
0x0100	<Characteristic>	(READ|WRITE|WRITENR) {Gyro. on}
0x0101	{Gyro. on}	0xXXXX [Gyro. on/off, see sensor settings]
0x0102	<Characteristic User Description>	"Gyro. on/off"
0x0110	<Characteristic>	(READ|WRITE|WRITENR) {Gyro. rate}
0x0111	{Gyro. rate}	0xXXXX [Gyro. rate, see sensor settings]
0x0112	<Characteristic User Description>	"Gyro. rate Hz"
0x0120	<Characteristic>	(READ|WRITE|WRITENR) {Gyro. range}
0x0121	{Gyro. range}	0xXXXX [Gyro. range, see sensor settings]
0x0122	<Characteristic User Description>	"Gyro. range dps"
0x0130	<Characteristic>	(READ|WRITE|WRITENR) {Mag. on}
0x0131	{Mag. on}	0xXXXX [Mag. on/off, see sensor settings]
0x0132	<Characteristic User Description>	"Mag. on/off"
0x0140	<Characteristic>	(READ|WRITE|WRITENR) {Mag. rate}
0x0141	{Mag. rate}	0xXXXX [Mag. rate, see sensor settings]
0x0142	<Characteristic User Description>	"Mag. rate Hz"

Table 15. Summary of the WAX9 attribute profile

Attribute Protocol Communication

There are several ways of obtaining data from and controlling the WAX9 using the attribute protocol. This can be simply reading or writing the values directly or using notifications and indications. Reading is always a request-response transaction whereas writing can be preformed without a write confirmation (WRITENR also know as COMMAND transactions). Reading and writing of attributes allows variables much longer than the maximum data unit size to be transferred reliably at the expense of longer transaction times.

The preferable method of streaming the sensor data is to use notifications or indications initiated by the WAX9. Notifications are used for guaranteed data flow and require a response, furthermore, there can only be one pending notification at a time. The result is that notifications are not the preferred streaming method unless low sample rates and guaranteed throughput is required. If a sample time elapses while waiting for the indication response then the pending sample is lost. Notifications are better for higher data rates as they require no response or acknowledgement and can be sent at any time. The problem with notifications and indications is that the maximum data packet size is restricted; the Bluetooth LE specification only guarantees a 20 byte payload to be supported.

The serial port service uses WRITENR packets to the serial input (WAX9) and NOTICIFATIONS for the serial output (from the WAX9). To receive these notifications the user should write the value 0x0001 to the `<Client Characteristic Configuration>` associated with the `{Serial data output}`, handle 0x0026. Notifications can not be polled for and this value must be set to receive notifications of serial messages. For indications the value 0x0002 should be written and to disable the serial output a value of 0x0000 is used. A value of 0x0003 enables notifications and indications, however, in the WAX9 implementation the indication flag has precedence. It should be noted that the

serial port service is not controlled, specified or recommended by the Bluetooth SIG and is purely to facilitate transiting from RFCOMM to LE modes. The UUID values used by Nordic Semiconductor in their LE UART program were used to allow compatibility with their mobile phone applications.

The same `<Client Characteristic Configuration>` values are associated with the `{Sensor data}` and `{Meta data}` attributes. Writing values of 0x0001 and 0x0002 will enable notifications and indications respectively in the same fashion as the serial port service. It is suggested that notifications be used for sensor data and indications or notifications be used for meta data depending on its importance. In either mode, the same data format is always transmitted for each handle and framing is unimportant due to the fixed packet boundaries.

Device Control

The user will likely control the device in LE mode using the dedicated variable mappings provided in the profile. These settings are identical to those set by the serial commands as summarised in the sensor settings section. The majority of the settings (excluding inactivity timeout and LED) will not take effect until the device receives a command to the {Enumerated command input} attribute. This attribute should be written with a uint16 value as enumerated in the following table; It is write only.

Command number	Behaviour
0x0000	Save: Latches current settings, useful for new sleep mode settings. Will stop streaming and sensors are turned off.
0x0001	Stream: Latches settings, powers on sensors and begins streaming.
0x0002	Reset: Device will close Bluetooth connection and execute a reset, any new settings are lost.
0x0003	LED: The current LED override setting is restored to device control (-1). Will stop streaming and sensors are turned off.
0x0004	Clear: The current settings are replaced with the defaults and saved. Will stop streaming and sensors are turned off.
0x0005	Stop: Latches current settings, useful for new sleep mode settings. Will stop streaming and sensors are turned off.
0x0006	Pause: The device stops streaming, sensors remain on and sampling continues*.
0x0007	Play: Valid only after pause, resumes streaming*.

** Sample number continues to increment, new settings are not latched.*

Table 16. Enumerated command input for LE connections

Data Output Format

The following table shows the 20 byte data format for sensor data indications and notifications.

Byte *	Value hex	Note
0+1	0xXXXX	Sample number uint16
2+3, 4+5, 6+7	0xXXXX, 0xXXXX, 0xXXXX	Accelerometer values, 16 bit signed integers, order x,y,z
8+9, 10+11, 12+13	0xXXXX, 0xXXXX, 0xXXXX	Gyroscope values, 16 bit signed integers, order x,y,z
14+15, 16+17, 18+19	0xXXXX, 0xXXXX, 0xXXXX	Magnetometer values, 16 bit signed integers, order x,y,-z

Table 17. Data output format for LE connections

The following table shows the 8 byte data format for indications and notifications of the meta data (pressure, temperature and battery).

Byte *	Value hex	Note
0+1+2+3	0xXXXXXXXX	Pressure Pascal's (pa) **uint**32
4+5	0xXXXX	Temperature °C x10 signed int16
6+7	0xXXXX	Battery mill volts uint16

Table 18. Meta data output format for LE connections

** All values are little endian i.e. Battery …,0x40,0x10 is 0x1040 or 4.160 V*

Channel Contention Issues

It is possible to set very high packet rates to the client by setting high sample rates and this may create channel contention. Typical LE baseband controllers have a **maximum** throughput of ~4 kB/s which is a 200 Hz notification rate. If the channel becomes blocked it may not be possible to send a stop command or clear the notification flag and in this instance it becomes necessary to cycle the device connection to restore control.

Data Conversion

All sensors have a digital output which can be converted into SI units using simple transforms. This section summarises these transforms.

Battery

The battery output is in millivolts and the battery used is a lithium polymer 100 mAh type. The fully charged voltage of lithium polymer cells at 25 ˚C is 4.2 V or 4200 mV. The discharge curve of lithium batteries is non-linear and it is not possible to linearly approximate the percentage of capacity remaining. After 3300 mV the battery is nearly depleted (see Figure 1) and the voltage will drop rapidly, this could result in a device reset and an unexpected loss of communication; by this point the user should be recharging the device.

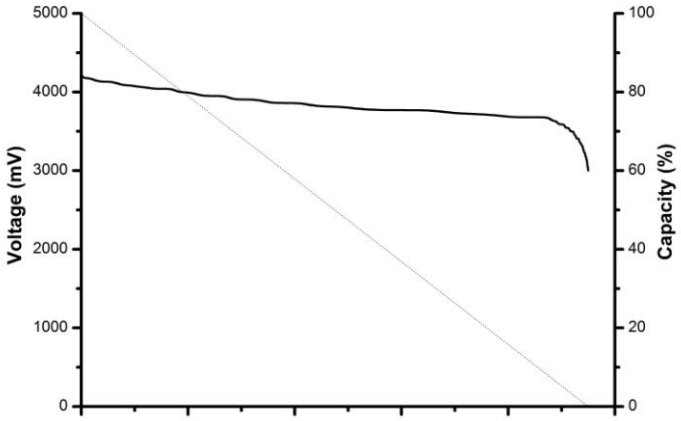

Figure 1 Typical lithium polymer battery discharge curve (Y-axis 10 hours, current = 0.1C)

Temperature

The temperature sensor is sampled every second and is expresses as a signed 16bit integer in 0.1˚C steps; Divide by 10 to convert to Celsius (SI unit).

Barometric Pressure

The pressure is expressed as a signed 32bit integer in Pascal's (SI unit). Conversion from Pascal's to altitude is a complicated process involving thermodynamic and physical models to account for the changes in density and pressure through the atmosphere. However, a simple conversion can be accomplished assuming typical conditions to reasonably accurately determine a change in altitude from a given reference point at which the pressure is known (P_0). The conversion is explained here and can be summarised as:

$$\partial h \approx K \, ln \frac{P}{P_0}$$

where ∂h is the change in height, P is the pressure in Pascal's and P_0 is the pressure at the origin (e.g. Sea level).

$$K = -\frac{RT_0}{gM}$$

where R is the universal gas constant, T_0 is origins temperature, g is gravity and M is the molar mass of air (all SI units). Typically $K \simeq$ -8434.6678

Since there is an approximate exponential relationship at moderate altitudes, a simple exponential function can be used as described on page 16 of the barometric pressure sensors datasheet. This approximation uses the faster pow(x,y) function from the standard C library math.h and can be implemented to yield the height in cm using the following example.

```
#include <math.h>
long CalculateAltitudeCm(long pressure, long pressure_sea_level)
{
	long altitude = 0;
	float temp1,temp2;
	temp1 = ((float)pressure)/((float)pressure_sea_level);
	temp2 = powf(temp1,0.19029495718363462); /*double*/
	temp1 = (float)4433000*(1-temp2); /*note extra zeros are for cm*/
	altitude = ((temp1 >= 0)?(long)(temp1+0.5):(long)(temp1-0.5));
	return altitude;
}
```

Example Code 1 Converting barometric pressure to altitude (cm)

The noise from the sensor yields a noise level of ~25cm between readings (when stationary) making the sensor less useful for small deviations.

Accelerometer

The accelerometer output is expressed as three signed 16bit integers in the order x, y, z where the axis orientations are as follows:

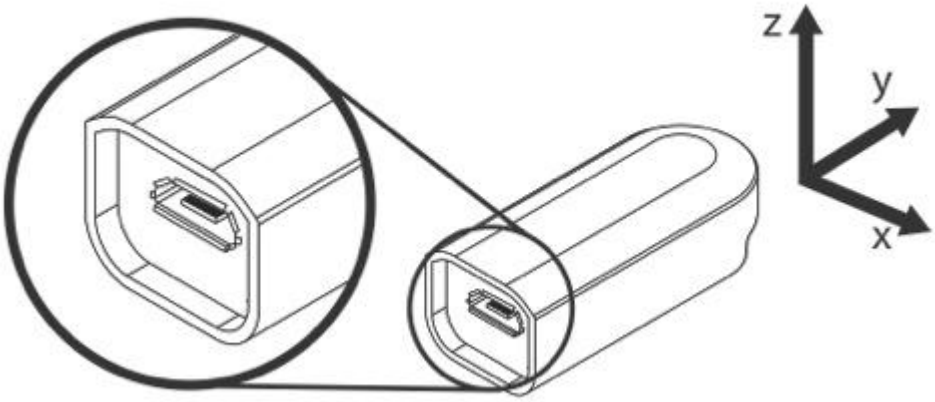

Figure 2: Sensor axis orientation

The sensor measures linear acceleration in each of three axis. The accelerometer resolution is 14 bits and is left justified in all resolution modes, this makes the last two bits of the sensor output always zero. The range setting of the accelerometer determines the scaling factor required to convert the integer to standard units as described in the following table.

Range setting	Convert to g	Dynamic range
2	divide by 16384	+/- 2g
4	divide by 8192	+/- 4g
8	divide by 4096	+/- 8g

Table 19. Conversion of accelerometer values

The accelerometer resolution, accuracy and noise are described in the device datasheet; some degree of calibration may be required for precision measurements.

Gyroscope

The gyroscope sensor output is expressed as three signed 16bit integers in the order x, y, z. The orientation of the sensor axis is the same as the accelerometer (see previous section). The output of the gyroscope is proportional to the angular velocity around each or the axis and is usually described in degrees per second (dps). The conversion to dps depends upon the range setting of the sensor as summarised below:

Range setting	Convert to dps	Dynamic range
250	multiply by 0.00875	+/- 250 dps
500	multiply by 0.01750	+/- 500 dps
2000	multiply by 0.07000	+/- 2000 dps

Table 20. Conversion of gyroscope values

The linearity, offset and noise level of the sensor is described in the device datasheet; some degree of calibration may be required for precision measurements.

Magnetometer

The magnetometer sensor output is expressed as three signed 16 bit integers in the order x, y, z. The orientation of the x and y axis are aligned to the accelerometer and gyroscope. However, **the z axis is in the opposite direction** to the other two sensors. It is simple to correct this is software by changing the sign of the z axis output i.e. z = -z.

The output of the magnetometer sensor is the measured magnetic field in the direction of the axis in approximately 0.1 µT steps (1 mGs, milli-gauss) and the range of the sensor is +/- 20,000 (2 mT or 0.2 Gs). The scaling factor will vary between devices (scaling error), the magnetometer will be influenced by magnetised nearby materials (offset error) and the earth's magnetic field varies widely around the earths surface. For this reason, most magnetometer calculations that determine heading (compass calculations) do not rely on the absolute field strength other than to determine if the sensor is being interfered with. For details on implementing a compass functionality the user should investigate the many options available such as this application note which uses the same sensor pair (both magnetometer and accelerometer).

Upgrading Firmware

The firmware is under continual development and new updates may become available to fix bugs or add features. This section details how to upgrade to a new firmware using the windows PC software, a new .hex file and the micro USB cable.

1) If you received an email of the software, the bootloader "WAX9 Bootloader.exe" may have been renamed to "WAX9 Bootloader.bin" to email it past virus checkers; Rename it to "WAX9 Bootloader.**exe**".
2) Run the software on a windows PC.
3) Plug in **one** device to the machine. If the batter is flat it will flash red until the battery reaches ~10% for a few minutes. When the device stops flashing, the bootloader will be able to discover it. It appears as a HID device to the operating system and requires no driver.
4) Open the new .hex file using the "Open Hex File" dialogue. Select the firmware to bootload e.g. "WAX9 bleStack FW3.2.hex"*.
5) Select "Program/Verify" and wait. The LED on the device will go red and flicker during bootloading - do not disconnect.**
6) Reset the device using the bootloader. The device will reset and check that the new firmware is compatible showing error codes if not. When the device returns to the bootloader application it will be in (shipping mode). Disconnect, wait (5 seconds) and reconnect the USB to exit shipping mode.
7) Now disconnect the device. Its default name immediately after bootloading will be "WAX9-FFFF" where FFFF will be replaced by the hexadecimal representation of the last 2 bytes of its Bluetooth MAC address on next reset.

** Firmware before FW3.2 did not detect the chipset used and the user had to select the correct version. FW3.2 auto detects at runtime, however, the older cc2560 chipset does not support LE connections and this functionality will be unavailable.*
*** After the process completes the verify step may report verify fails above address 0x100000, this is ok and indicates that some of the unused data memory contained information from a previous firmware release. A new bootloader will be available shortly to fix this issue.*

Support

Please visit www.openmovement.co.uk for support contacts.

Open Source

The WAX9 sensor project is open source hardware and firmware, it could be customised to perform other functions and be a useful addition to your project. Please see www.openmovement.co.uk to access the source code, design files and new firmware releases.

Acknowledgments

The WAX9 uses hardware developed by the author of this document, K. Ladha, for the Open Movement Project at Newcastle University. Packaging design and manufacture are attributed to the work of C. Ladha and T. Nappy. Firmware uses parts of the Open Movement framework written by D. Jackson and K. Ladha.

The WAX9 uses bleStack, an open source light weight dual mode embedded Bluetooth stack developed by K. Ladha. The RFCOMM and SDP layers are adapted from the code written and distributed by G. Gangolells (2012).

Disclaimers

Revisions

2014/2/21: First release of documentation for FW3.0
2014/2/28: Firmware 3.1 fixed gyro sleep bug, added pause and play commands to documentation
2014/3/11: Firmware 3.2 released, fixed LE UART bug, added chipset auto detection and printout