Orbotix Communication API

revision 1.50

Introduction 5
Bluetooth 5
Client/Server 5
Virtual Devices 5
Expectations 5
Sphero Overview 6
Raw Inputs 6
Raw Outputs 6
Packet Structures 7
Client Command Packets 7
Sphero Response Packets 8
Sphero Asynchronous Packets 9
Data Packing 10
Device ID 00h – The Core 11
Ping – 01h 11
NOTE 11
Get Versioning – 02h 12
Control UART Tx Line – 03h 12
Set Device Name – 10h 13
Get Bluetooth Info – 11h 13
Set Auto Reconnect – 12h 14
Get Auto Reconnect – 13h 14
Get Power State – 20h 15
Set Power Notification – 21h 15
Sleep – 22h 16
Get Voltage Trip Points – 23h 16
Set Voltage Trip Points – 24h 17
Set Inactivity Timeout – 25h 17
Jump To Bootloader – 30h 18
Perform Level 1 Diagnostics – 40h 19
Perform Level 2 Diagnostics – 41h 20
Clear Counters – 42h 20
Assign Time Value – 50h 21

Poll Packet Times – 51h 21
Device ID 01h – Bootloader 22
Device ID 02h – Sphero 23
Set Heading – 01h 23
Set Stabilization – 02h 23
Set Rotation Rate – 03h 24
Set Creation Date – 04h 24
~~Get Application Configuration Block – 05h~~ 24
~~Re-enable Demo Mode – 06h~~ 24
Get Chassis ID – 07h 25
Set Chassis ID – 08h 25
Self Level – 09h 26
~~Set Vector Drive Limit – 0Ah~~ 27
Set Data Streaming – 11h 28
Configure Collision Detection – 12h 31
Configure Locator – 13h 32
Set Accelerometer Range – 14h 33
Read Locator – 15h 33
Set RGB LED Output – 20h 33
Set Back LED Output – 21h 34
Get RGB LED – 22h 34
Roll – 30h 35
Boost – 31h 35
Set Raw Motor Values – 33h 36
Set Motion Timeout – 34h 36
Set Permanent Option Flags – 35h 37
Get Permanent Option Flags – 36h 37
Set Temporary Option Flags – 37h 38
Get Temporary Option Flags – 38h 38
Get Configuration Block – 40h 39
Set SSB Modifier Block – 41h 39
Set Device Mode – 42h 40
Set Configuration Block – 43h 41

Get Device Mode – 44h 41
Get SSB – 46h 41
Set SSB – 47h 42
Refill Bank – 48h 42
Buy Consumable – 49h 42
Use Consumable – 4Ah 43
Grant Cores – 4Bh 43
Add XP – 4Ch 43
Level Up Attribute – 4Dh 44
Get Password Seed – 4Eh 44
Enable SSB Async Messages – 4Fh 44
Run Macro – 50h 45
Save Temporary Macro – 51h 45
Save Macro – 52h 45
Reinit Macro Executive – 54h 46
Abort Macro – 55h 46
Get Macro Status – 56h 46
Set Macro Parameter – 57h 47
Append Macro Chunk – 58h 47
Erase orbBasic Storage – 60h 48
Append orbBasic Fragment – 61h 48
Execute orbBasic Program – 62h 48
Abort orbBasic Program – 63h 49
Submit Value to Input Statement – 64h 49
Commit RAM Program to Flash – 65h 49
Appendix A: Enumerated Codes Quick Reference 50
Appendix C: Understanding the Roll Command Parameters 53
Revision History 54

Introduction

Welcome to the wide world of robot control over Bluetooth. What follows is a description of our lightweight command and control protocol which you can use to build up applications offering a higher level of functionality. But before we expose you to all the gory details there are some concepts and limitations you'll need to become familiar with.

Bluetooth

You've heard about this for years (mostly with hands-free headsets) but what is it? In short, it's a low-cost, easily configurable data radio link that smartphones natively support (along with some desktops). Bluetooth implements what is called a *stream* interface – that is, data is sent and received in a constant stream of bytes. This is in contrast to a packetized data format which Ethernet, Wi-Fi and other communications protocols implement. One advantage to a stream interface is its simplicity: just open the port and start sending data. The disadvantage is in synchronizing the two ends of the link. Without an inherent packet structure, you may suddenly be listening in on the middle of a transaction and the data won't make any sense. So, some extra effort must be placed in constructing a packet framework that has a resilient boundary sequence and detection method. More on that later.

Client/Server

This relationship describes the order of information movement between your app and Sphero. In 95% of all cases your app is the initiator (the client) and Sphero acts on the command (as a server). This is also known as synchronous communication and other than for a special mode Sphero can be placed in, he never asynchronously sends data back to the client (that is, without being specifically asked).

Virtual Devices

Sphero is an actual device (obviously) but in his core software, many *virtual* devices are implemented. This makes the separation of tasks more clear: the control system accepts direction and speed commands, the Bootloader device handles firmware downloads, the orbBasic device manages downloaded user programs, etc.

Expectations

This document doesn't expect you to be a nuclear genius but some familiarity with programming and data communications will help. It also expects you to be able to move between decimal and hexadecimal numbering bases seamlessly, though numbers in the latter have an 'h' suffix for clarity.

Sphero Overview

Before you can start talking to Sphero, you should probably know the extents of what he can offer. At the most basic level he's electronically a collection of raw inputs and outputs.

Raw Inputs

- Three axis rotation rate gyro
- Three axis accelerometer
- Approximate ground speed
- Data from radio link
- Battery voltage

Raw Outputs

- Power to left and right drive wheels
- RGB LED color value
- Back LED intensity
- Data to radio link

Internal software builds up more useful data constructs from these raw hardware I/Os: heading control systems, distance measurement, data integrators/differentiators and more. You'll be surprised at what you can do if you tie these basic elements together with some cleverness.

Packet Structures

Client Command Packets

Packets are sent from Client → Sphero in the following byte format:

SOP1	SOP2	DID	CID	SEQ	DLEN	<data>	CHK

A brief description of the fields:

SOP1	Start of Packet #1	Always FFh
SOP2	Start of Packet #2	F8 to FFh encoding 4 bits of per-message options (see below)
DID	Device ID	The virtual device this packet is intended for
CID	Command ID	The command code
SEQ	Sequence Number	This client field is echoed in the response for all synchronous commands (and ignored by Sphero when SOP2 has bit 0 clear)
DLEN	Data Length	The number of bytes following through the end of the packet
<data>	Data	Optional data to accompany the Command
CHK	Checksum	The modulo 256 sum of all the bytes from the DID through the end of the data payload, bit inverted (1's complement)

SOP2 bitfield encoding

bits 7-4	bit 3	bit 2	bit 1	bit 0
1111	1	1	Reset timeout	Answer

- Answer – When set to 1, act upon this command and send a reply. When 0, act but do not reply.
- Reset timeout – When set to 1, reset the client inactivity timeout after executing this command. When 0 do not reset the timer.

The Answer bit has essentially existed since FW 0.99; the remainder of the bit definitions came into existence with FW 1.26 in late 2012.

Sphero Response Packets

Commands are acknowledged from Sphero → Client in a similar format:

SOP1	SOP2	MRSP	SEQ	DLEN	<data>	CHK

A brief description of the fields:

SOP1	Start of Packet #1	Always FFh
SOP2	Start of Packet #2	Set to FFh when this is an acknowledgement, FEh when this is an asynchronous message
MRSP	Message Response	This is generated by the message decoder of the virtual device (refer to the appropriate appendix for a list of values)
SEQ	Sequence Number	Echoed to the client when this is a direct message response (set to 00h when SOP2 = FEh)
DLEN	Data Length	The number of bytes following through the end of the packet
<data>	Data	Optional data in response to the Command or based on "streaming" data settings
CHK	Checksum	Packet checksum (as computed above)

There are a few things to note:

- Asynchronous (aka "streaming") packets are implemented by changing the value of SOP2 and clearing the Answer bit. This can improve responsiveness (and decrease command latency) but through non-guaranteed delivery. The packet format is slightly different in the Sphero → Client direction.
- DLEN is always at least 01h since the CHK byte follows. In some special cases it is set to FFh to signify a fixed <data> length greater than 254 bytes. This is specific to certain DID/CID combinations.
- The SOP1/SOP2 and CHK fields are used to identify correctly formed packets before they're submitted to a DID for processing.
- Here is an example of computing a checksum to transmit a Ping packet. The bytes for the packet (with a sequence number of 52h) are: FFh FFh 00h 01h 52h 01h <chk>. The checksum equals the sum of the underlined bytes (54h) modulo 256 (still 54h) and then bit inverted (ABh).

Commands are grouped into two categories: set and get. Set commands assign a defined variable in Sphero and include a non-zero data payload that contains the assignment. Responses are in the most simple form, without a data payload. Rather than duplicate them all through the document, here is the Simple Response to a successful set command:

	SOP1	SOP2	MRSP	SEQ	DLEN	CHK
Simple Response:	FFh	FFh	00h	<echoed>	01h	<computed>

Get commands request settings, status or the current value of dynamic values. The formats of these responses are detailed in each CID.

Sphero Asynchronous Packets

As mentioned previously, the format of asynchronous packets originating from Sphero is slightly different:

SOP1	SOP2	ID CODE	DLEN-MSB	DLEN-LSB	data	CHK
FFh	FEh	<code>	<msb>	<lsb>	<data>	<cmp>

There are no MRSP or SEQ bytes, since they don't make sense in this context. The ID CODE field identifies what type of data is arriving in this packet and as you can see, the DLEN field has been expanded to (clearly) permit payloads exceeding 254 bytes. The following is a list of the currently defined ID codes and the DID/CID commands that control generation of those packets where applicable.

ID CODE	Description	Generating DID	CID
01h	Power notifications	00h	21h
02h	Level 1 Diagnostic response	00h	40h
03h	Sensor data streaming	02h	11h
04h	Config block contents	02h	40h
05h	Pre-sleep warning (10 sec)	n/a	n/a
06h	Macro markers	n/a	n/a
07h	Collision detected	02h	12h
08h	orbBasic PRINT message	n/a	n/a
09h	orbBasic error message, ASCII	n/a	n/a
0Ah	orbBasic error message, binary	n/a	n/a
0Bh	Self Level Result	02h	09h
0Ch	Gyro axis limit exceeded (FW ver 3.10 and later)	n/a	n/a
0Dh	Sphero's soul data	02h	43h
0Eh	Level up notification	n/a	n/a
0Fh	Shield damage notification	n/a	n/a
10h	XP update notification	n/a	n/a
11h	Boost update notification	n/a	n/a

Power notification (01h) details are included with “Set Power Notification”.

Level 1 diagnostic response details are included with “Perform Level 1 Diagnostics”.

Sensor data streaming details are included with “Set Data Streaming”.

Config block contents details are included with “Get Configuration Block”.

The Pre-Sleep warning is sent once, 10 seconds prior to Sphero entering sleep due to client inactivity.

Macro markers come from special macro commands and optionally at the end of a macro.

Collision detection messages are based on the accelerometer, measured speed, etc.

The orbBasic PRINT ID 08h is akin to STDOUT, 09h to STDERR and 0Ah a machine readable version of STDERR.

Self Level Result is sent after the self level routine completes, but only if the routine was initiated by an API call.

The Gyro Axis Limit Exceeded message contains one byte of data where the bits signify the axes that exceeded the limit: bit 0 = X positive, bit 1 = X negative, bit 2 = Y+, bit 3 = Y-, bit 4 = Z+ and bit 5 = Z-. The message is emitted when one threshold is exceeded and all of the max measurements are cleared upon receipt of a Set Heading API command (DID 02h, CID 01h).

The level up notification contains two 16-bit unsigned integers. The first is the new robot level. The second is the total number of attribute points the user has to spend.

The Shield damage notification contains one unsigned byte representing the portion of shield left (out of 255). The shields are damaged when Sphero collides with other objects. The shields are regenerate automatically over time. Both collisions and regeneration generate asynchronous updates.

The XP update notification contains one byte representing how much experience Sphero has gained toward the next robot level. The scale is from 0=0% to 255=100%.

The boost update notification contains one byte representing how much boost capability Sphero has. The value goes down when boost is used and automatically regenerates over time. Regeneration and use both generate asynchronous updates. The scale is from 0=0% to 255=100%.

Data Packing

Multi-byte numbers are sent MSB first in both directions. Here are two examples of how the data looks "on the wire."

22h	78h	00h	41h
byte 0			byte 3

= 22780041h
(unsigned 32-bit integer)

40h	49h	0Fh	DBh
byte 0			byte 3

= 3.1415927
(single precision IEEE-754)

Device ID 00h – The Core

The Core Device encapsulates actions that are fundamental to all Orbotix devices.

Ping – 01h

Command:

DID	CID	SEQ	DLEN	CHK
00h	01h	<any>	01h	<computed>

Response:

MRSP	SEQ	DLEN	CHK
00h	<echoed>	01h	<computed>

The Ping command is used to verify both a solid data link with the Client and that Sphero is awake and dispatching commands. Even though Ping is neither a set or get format command, it still enjoys a Simple Response.

NOTE

From here forward the redundant fields in both transmit and receive packets will be omitted for clarity; we assume the MRSP is 00h (for success), SEQ is echoed and CHK is computed correctly both ways.

Get Versioning – 02h

Command:

DID	CID
00h	02h

Response:

DLEN	<data>
0Bh	see below

The Get Versioning command returns a whole slew of software and hardware information. It's useful if your Client Application requires a minimum version number of some resource within Sphero in order to operate. The data record structure is comprised of fields for each resource that encodes the version number according to the specified format.

Name	Byte index	Description
RECV	0	This record version number, currently set to 02h. This will increase when more resources are added.
MDL	1	Model number; currently 02h for Sphero
HW	2	Hardware version code (ranges 1 through 9)
MSA-ver	3	Main Sphero Application version byte
MSA-rev	4	Main Sphero Application revision byte
BL	5	Bootloader version in packed nibble format (i.e. 32h is version 3.2)
BAS	6	orbBasic version in packed nibble format (i.e. 4.4)
MACRO	7	Macro executive version in packed nibble format (4.4)
API-maj	8	API major revision code this firmware implements
API-min	9	API minor revision code this firmware implements

Control UART Tx Line – 03h

Command:

DID	CID	SEQ	DLEN	FLAG
00h	03h	<any>	02h	0 or 1

Response:

Simple Response

This is a factory command that either enables or disables the CPU's UART transmit line so that another physically connected client can configure the Bluetooth module. The receive line is always listening, which is how you can re-enable the Tx line later. Or just reboot as this setting is not persistent.

Set Device Name – 10h

Command:

DID	CID	SEQ	DLEN	<data>
00h	10h	<any>	<data> + 01h	text name

Response:

Simple Response

This formerly reprogrammed the Bluetooth module to advertise with a different name, but this is no longer the case. This assigned name is held internally and produced as part of the Get Bluetooth Info service below. Names are clipped at 48 characters in length to support UTF-8 sequences; you can send something longer but the extra will be discarded. This field defaults to the Bluetooth advertising name.

To alter the Bluetooth advertising name from the standard Sphero-RGB pattern you will need to $$$ into the RN-42 within 60 seconds after power up, issue the command SN,mynewname and finish with r,1 to reboot the module.

Get Bluetooth Info – 11h

Command:

DID	CID
00h	11h

Response:

DLEN	<16 bytes>	<12 bytes>	<byte>	<3 bytes>
21h	ASCII name	ASCII BTA	00h	ID colors

This returns a structure containing the textual name in ASCII of the ball (defaults to the Bluetooth advertising name but can be changed), the Bluetooth address in ASCII and the ID colors the ball blinks when not connected to a smartphone.

The ASCII name field is padded with zeros to its maximum size.

This is provided as a courtesy for Clients that have don't have a method to interrogate their underlying Bluetooth stack for this information.

Set Auto Reconnect – 12h

Command:

DID	CID	SEQ	DLEN	data 0	data 1
00h	12h	<any>	03h	flag	time

Response:

Simple Response

This configures the control of the Bluetooth module in its attempt to automatically reconnect with the last mobile Apple device. This is a courtesy behavior since the Apple Bluetooth stack doesn't initiate automatic reconnection on its own.

The two parameters are simple: flag is 00h to disable or 01h to enable, and time is the number of seconds after power-up in which to enable auto reconnect mode. For example, if time = 30 then the module will be attempt reconnecting 30 seconds after waking up. (refer to RN-APL-EVAL pg. 7 for more info)

Get Auto Reconnect – 13h

Command:

DID	CID	SEQ	DLEN
00h	13h	<any>	01h

Response:

DLEN	data 0	data 1
03h	flag	time

This returns the Bluetooth auto reconnect values as defined in the “Set Auto Reconnect” command.

Get Power State – 20h

Command:

DID	CID
00h	20h

Response:

DLEN	<data>
09h	see below

This returns the current power state and some additional parameters to the Client. They are detailed below.

offset	name	description
00h	RecVer	Record version code – the following definition is for 01h
01h	Power State	High-level state of the power system as concluded by the power manager: 01h = Battery Charging, 02h = Battery OK, 03h = Battery Low, 04h = Battery Critical
02h	BattVoltage	Current battery voltage scaled in 100ths of a volt; 02EFh would be 7.51 volts (unsigned 16-bit value)
04h	NumCharges	Number of battery recharges in the life of this Sphero (unsigned 16-bit value)
06h	TimeSinceChg	Seconds awake since last recharge (unsigned 16-bit value)

Set Power Notification – 21h

Command:

DID	CID	SEQ	DLEN	data
00h	21h	<any>	02h	flag

Response:

Simple Response

This enables Sphero to asynchronously notify the Client periodically with the power state or immediately when the power manager detects a state change. Timed notifications arrive every 10 seconds until they're explicitly disabled or Sphero is unpaired. The flag is as you would expect, 00h to disable and 01h to enable. This setting is volatile and therefore not retained across sleep cycles.

The complete power notification message is of the form:

SOP1	SOP2	CODE	DLEN-MSB	DLEN-LSB	data	CHK
FFh	FEh	01h	00h	02h	state	<cmp>

The power state byte mimics that of CID 20h above: 01h = Battery Charging, 02h = Battery OK, 03h = Battery Low, 04h = Battery Critical

Sleep – 22h

Command:

DID	CID	SEQ	DLEN	Wakeup	Macro	orbBasic
00h	22h	<any>	06h	<16-bit val>	<val>	<16-bit val>

Response:

Simple Response

This command puts Sphero to sleep immediately. There are three optional parameters that program the robot for future actions:

name	description
Wakeup	The number of seconds for Sphero to sleep for and then automatically reawaken. Zero does not program a wakeup interval, so he sleeps forever. FFFFh attempts to put him into deep sleep (if supported in hardware) and returns an error if the hardware does not support it.
Macro	If non-zero, Sphero will attempt to run this macro ID upon wakeup.
orbBasic	If non-zero, Sphero will attempt to run an orbBasic program in Flash from this line number.

Get Voltage Trip Points – 23h

Command:

DID	CID	SEQ	DLEN
00h	23h	<any>	01h

Response:

DLEN	<Vlow>	<Vcrit>
05h	<16-bit val>	<16-bit val>

This returns the voltage trip points for what Sphero considers Low battery and Critical battery. The values are expressed in 100ths of a volt, so the defaults of 7.00V and 6.50V respectively are returned as 700 and 650.

Set Voltage Trip Points – 24h

Command:

DID	CID	SEQ	DLEN	<Vlow>	<Vcrit>
00h	24h	<any>	05h	<16-bit val>	<16-bit val>

Response:

Simple Response

This assigns the voltage trip points for Low and Critical battery voltages. The values are specified in 100ths of a volt and the limitations on adjusting these away from their defaults are:

- Vlow must be in the range 675 to 725 (±25)
- Vcrit must be in the range 625 to 675 (±25)
- There must be 0.25V of separation between the two values

Shifting these values too low could result in very little warning before Sphero forces himself to sleep, depending on the age and history of the battery pack. So be careful.

Set Inactivity Timeout – 25h

Command:

DID	CID	SEQ	DLEN	TIME
00h	25h	<any>	03h	<16-bit val>

Response:

Simple Response

To save battery power, Sphero normally goes to sleep after a period of inactivity. From the factory this value is set to 600 seconds (10 minutes) but this API command can alter it to any value of 60 seconds or greater.

The inactivity timer is reset every time an API command is received over Bluetooth or a shell command is executed in User Hack mode. In addition, the timer is continually reset when a macro is running *unless* the MF_STEALTH flag is set, and the same for orbBasic unless the BF_STEALTH flag is set.

Jump To Bootloader – 30h

Command:

DID	CID
00h	30h

Response:

Simple Response

This command requests a jump into the Bootloader to prepare for a firmware download. It always succeeds, because you can always stop where you are, shut everything down and transfer execution. All commands after this one must comply with the Bootloader Protocol Specification, which is a separate document.

Note that just because you can always vector into the Bootloader, it doesn't mean you can get anything done. Further details are explained in the associated document but in short: the Bootloader doesn't implement the entire Core Device message set and if the battery is deemed too low to execute reflashing operations, all you can do is return to the Main Application.

Perform Level 1 Diagnostics – 40h

Command:

DID	CID
00h	40h

Response 1:

Simple Response

Response 2:

SOP1	SOP2	CODE	DLEN-MSB	DLEN-LSB	data	CHK
FFh	FEh	02h	<msb>	<lsb>	<data>	<cmp>

This is a developer-level command to help diagnose aberrant behavior. Most system counters, process flags, and system states are decoded into human readable ASCII. There are two responses to this command: a Simple Response followed by a large async message containing the results of the diagnostic tests. As of FW version 0.99, the answer was well over 1K in length and similar to:

```
[System]
Mode F, Boot code 12
0 rechrg, 32 min
since last, 0:51
alive
Cold:13, Warm:0,
Wakeup:0, NMI:0,
Hard:0
Dist rolled: 0, Vbatt
7.85, state: OK
SensorsHthy:1
BTError:0
AuthOK:1
Stabilize:1
TestPin:0
AutoRN:0
Mac:0
Bootldr=1.7
MA=0.98
Board=2
OrbBasic=0.8
MacExec=2
CB=111

AutoRecon En=1
AutoReconDel=0
ClientTimeOut=300
WakeUpSec=0
```

```
[Network]
Rx good:7, bad:1,
Tx:780
Rx overruns:0, Tx:0
Dev name:Sphero-OWG,
BTA:0006664440B8
BTver:Ver 5.36 IAP
11/04/11

[Sensors]
Fail: 0 Loc: 0 Code:
0
[Accel]
Xsc=0.0039 Ysc=0.0040
Zsc=0.0039
Xb=-0.0078 Yb=0.0010
Zb=0.0552
[Gyro]
Xsc=0.0680 Ysc=0.0683
Zsc=0.0680
Xb=-12.3322 Yb=-
10.2964 Zb=-28.3654
Temp=35
Therm: Xb1=-11.9700
Xb2=-37.6833 Yb1=-
10.0140 Yb2=-30.0675
Zb1=-29.7397
Zb2=67.8367
Tmp1=34 Tmp2=105
```

```
Xsl=-0.3622
Xint=0.3434 Ysl=-
0.2824 Yint=-0.4109
Zsl=1.3743 Zint=-
76.4665
GyroAdjCnt=0

[Control]
Pitch P=60.000
I=0.200 D=100.000
Roll P=21.000 I=0.300
D=50.000
Yaw P=90.000 I=0.230
D=1200.000
RotRate=0.228

[Test Res]
PCBAtr=0x3ff Stn=7
AGtr=0x1 Stn=1
GTtr=0x1

[Idle loop]
MinClks:777
MaxClks:73987
MinFreq:51875
MaxFreq:104952
CPU 56% idle
```

Perform Level 2 Diagnostics – 41h

	DID	CID
Command:	00h	41h

Response:	**See below**

This is a developers-only command to help diagnose aberrant behavior. It is much less informative than the Level 1 command but it is in binary format and easier to parse. Here is the layout of the data record which is currently 58h bytes long:

offset	name	description
00h	RecVer	Record version code – the following definition is for 01h
02h	<empty>	Reserved
03h	Rx_Good	Good packets received (unsigned 32-bit value)
07h	Rx_Bad_DID	Packets with a bad Device ID (unsigned 32-bit value)
0Bh	Rx_Bad_DLEN	Packets with a bad data length (unsigned 32-bit value)
0Fh	Rx_Bad_CID	Packets with a bad Command ID (unsigned 32-bit value)
13h	Rx_Bad_CHK	Packets with a bad checksum (unsigned 32-bit value)
17h	Rx_Buff_Ovr	Receive buffer overruns (unsigned 32-bit value)
1Bh	Tx_Msgs	Messages transmitted (unsigned 32-bit value)
1Fh	Tx_Buff_Ovr	Transmit buffer overruns (unsigned 32-bit value)
23h	LastBootReason	Reason for last boot (8-bit value)
24h	BootCounters	16 different counts of boot reasons
44h	<empty>	Reserved
46h	ChargeCount	Charge cycles (unsigned 16-bit value)
48h	SecondsSinceCharge	Awake time in seconds since last charge (unsigned 16-bit value)
4Ah	SecondsOn	Life awake time in seconds (unsigned 32-bit value)
4Eh	DistanceRolled	Distance rolled (unsigned 32-bit value)
52h	Sensor Failures	Count of I^2C bus failures (unsigned 16-bit value)
54h	Gyro Adjust Count	Lifetime count of automatic GACs (unsigned 32-bit value)

Clear Counters – 42h

	DID	CID
Command:	00h	42h

Response:	**Simple Response**

This is a developers-only command to clear the various system counters described in command 41h. It is denied when Sphero is in Normal mode.

Assign Time Value – 50h

Command:

DID	CID	SEQ	DLEN	data
00h	50h	<any>	05h	32-bit value

Response:

Simple Response

Sphero contains a 32-bit counter that increments every millisecond. It has no absolute temporal meaning, just a relative one. This command assigns the counter a specific value for subsequent sampling. Though it starts at zero when Sphero wakes up, assigning it too high of a value with this command could cause it to roll over.

Poll Packet Times – 51h

Command:

DID	CID	SEQ	DLEN	Client Tx time
00h	51h	<any>	05h	32-bit value

Response:

DLEN	Client Tx time, T1	Sphero Rx time, T2	Sphero Tx time, T3
0Dh	32-bit value (echoed)	32-bit value	32-bit value

This command helps the Client application profile the transmission and processing latencies in Sphero so that a relative synchronization of timebases can be performed. This technique is based upon the scheme in the Network Time Protocol (RFC 5905) and allows the Client to reconcile time stamped messages from Sphero to its own time stamped events. In the following discussion, each 32-bit value is a count of milliseconds from some reference within the device.

The scheme is as follows: the Client sends the command with the Client Tx time (T1) filled in. Upon receipt of the packet, the command processor in Sphero copies that time into the response packet and places the current value of the millisecond counter into the Sphero Rx time field (T2). Just before the transmit engine streams it into the Bluetooth module, the Sphero Tx time value (T3) is filled in. If the Client then records the time at which the response is received (T4) the relevant time segments can be computed from the four time stamps T1-T4:

- The value *offset* represents the maximum-likelihood time offset of the Client clock to Sphero's system clock.
  offset = 1/2 * [(T2 - T1) + (T3 - T4)]
- The value *delay* represents the round-trip delay between the Client and Sphero:
  delay = (T4 - T1) - (T3 - T2)

Device ID 01h – Bootloader

Communication with the Bootloader is thoroughly explained in its own document, so please refer to it for all the details. Note that the "Jump To Bootloader" command is specified in DID 00h, the Core.

Device ID 02h – Sphero

These commands are specific to the features that Sphero offers.

Set Heading – 01h

Command:

DID	CID	SEQ	DLEN	HEADING
02h	01h	<any>	03h	16-bit value

Response:

Simple Response

This allows the smartphone client to adjust the orientation of Sphero by commanding a new reference heading in degrees, which ranges from 0 to 359. You will see the ball respond immediately to this command if stabilization is enabled.

In FW version 3.10 and later this also clears the maximum value counters for the rate gyro, effectively re-enabling the generation of an async message alerting the client to this event.

Set Stabilization – 02h

Command:

DID	CID	SEQ	DLEN	FLAG
02h	02h	<any>	02h	<bool>

Response:

Simple Response

This turns on or off the internal stabilization of Sphero, in which the IMU is used to match the ball's orientation to its various set points. The flag value is as you would expect, 00h for off and 01h for on. Stabilization is enabled by default when Sphero powers up. You will want to disable stabilization when using Sphero as an external input controller or even to save battery power during testing that doesn't involve movement (orbBasic, etc.)

An error is returned if the sensor network is dead; without sensors the IMU won't operate and thus there is no feedback to control stabilization.

Set Rotation Rate – 03h

	DID	CID	SEQ	DLEN	RATE
Command:	02h	03h	<any>	02h	<value>

Response:	**Simple Response**

This allows you to control the rotation rate that Sphero will use to meet new heading commands (DID 02h, CID 01h). A lower value offers better control but with a larger turning radius. A higher value will yield quick turns but Sphero may roll over on itself and lose control.

The commanded value is in units of 0.784 degrees/sec. So, setting a value of C8h will set the rotation rate to 157 degrees/sec. A value of 255 jumps to the maximum (currently 400 degrees/sec). A value of zero doesn't make much sense so it's interpreted as 1, the minimum.

Set Creation Date – 04h

	DID	CID	SEQ	DLEN	DATA
Command:	02h	04h	<any>	21h	<20h bytes>

Response:	**Simple Response**

This command will only execute at the factory. So don't try it.

~~Get Application Configuration Block – 05h~~

	~~DID~~	~~CID~~	~~SEQ~~	~~DLEN~~
~~Command:~~	~~02h~~	~~05h~~	~~<any>~~	~~01h~~

~~Response:~~	**~~Simple Response~~**

~~This allows you to retrieve the application configuration block that is set aside for exclusive use by applications.~~

Removed in FW 3.33+.

~~Re-enable Demo Mode – 06h~~

	~~DID~~	~~CID~~	~~SEQ~~	~~DLEN~~
~~Command:~~	~~02h~~	~~06h~~	~~<any>~~	~~01h~~

~~Response:~~

~~**Simple Response**~~

~~Demo mode is disabled once an application sends an API packet to Sphero. This special packet re-enables that mode without requiring a power cycle. As of FW 0.99 there are no actions associated with demo mode, making this command essentially ineffective for now.~~

Get Chassis ID – 07h

Command:

DID	CID	SEQ	DLEN
02h	07h	<any>	01h

Response:

DLEN	CHASSIS ID
03h	<16-bit val>

Returns the Chassis ID, a 16-bit value, which was set at the factory.

Set Chassis ID – 08h

Command:

DID	CID	SEQ	DLEN	CHASSIS ID
02h	08h	<any>	03h	<16-bit val>

Response:

Simple Response

Assigns the Chassis ID, a 16-bit value. This command only works if you're at the factory.

Self Level – 09h

	DID	CID	SEQ	DLEN	Options	Angle Limit	Timeout	True Time
Command:	02h	09h	<any>	05h	<byte>	<byte>	<byte>	<byte>

This command controls the self level routine. The self level routine attempts to achieve a horizontal orientation where pitch and roll angles are less than the provided Angle Limit. After both angle limits are satisfied, option bits control sleep, final angle (heading), and control system on/off. An asynchronous message is returned when the self level routine completes (only when started by API call). The required parameters are:

Name	Value	Description
Start/Stop	Bit 0	0 aborts the routine if in progress. 1 starts the routine.
Final Angle	Bit 1	0 just stops. 1 rotates to heading equal to beginning heading.
Sleep	Bit 2	0 stays awake after leveling. 1 goes to sleep after leveling.
Control System	Bit 3	0 leaves control system off. 1 leaves control system on (after leveling).
Angle Limit	0	Use the default value
	1 to 90	Set the max angle for completion (in degrees)
Timeout	0	Use the default value
	1 to 255	Set maximum seconds to run the routine
True Time	0	Use the default value
	1 to 255	Set the required "test for levelness" time to 10*True Time (in milliseconds)

Default values are: Angle = 3, Timeout = 15, True Time = 30 (300 milliseconds)
True Time*10 specifies the number of milliseconds that the pitch and roll angles must remain below the Angle Limit after the routine completes. If one of the values exceeds the Angle Limit, the ball will self level again and the accuracy timer will start again from 0.

Response:	**Simple Response**

The complete self level asynchronous response message is of the form:

SOP1	SOP2	CODE	DLEN-MSB	DLEN-LSB	data	CHK
FFh	FEh	0Bh	00h	02h	result	<cmp>

The result byte can be: 00h = Unknown, 01h = Timed Out (level was not achieved), 02h = Sensors Error, 03h = Self Level Disabled (see Option Flags), 04h = Aborted (by API call), 05h = Charger not found, 06h = Success

Self Level Angle Accuracy:
We have found that the real angle lags a bit behind the measured angle. Also, the angles may shift some after "level" is achieved as the motors stop and the system comes to a rest. A True Time value of 30 (300 milliseconds) is generally good enough to keep the angles within a degree or two of the specified Angle Limit. If greater accuracy is required the True Time value can be increased up to 255 (2.55 seconds).

Control System On/Off:
When the control system is off, obviously self leveling can not happen. There are several paths to this state:

- If the sensors are determined to be in an error state, self leveling will be skipped. Sleep requests will still trigger the go to sleep routine.
- The control system can be turned off using the "Set Stabilization" API call. This is used for certain games where Sphero is held in the hand as a controller.
- The control system can be turned off by a macro.
- The control system can be turned off using the shell command "l0".
- The control system can be turned off using the shell command "x11".
- The control system can be turned off through an orbBasic program.

When self level is called, leveling is skipped if the sensors are dead, as there is no recourse to this. For all the other cases, the self level routine runs. Since we have the System Options Flag to disable the self level routine, it is easy to override this behavior. Use the control system on/off bit to specify whether to leave the control system on or off after the self level routine is complete.

The current behavior is if a macro or orbBasic program is running and the ball starts charging, the self level routine runs (but it doesn't go to sleep). This could be desired behavior for some programs.

System Options Flag:
Refer to DID 02h, CID 35h for details. Sleep requests made using *this* self level API call while the disable flag is asserted will still cause the ball to go to sleep.

~~Set Vector Drive Limit – 0Ah~~

	~~DID~~	~~CID~~	~~SEQ~~	~~DLEN~~	~~Speed~~
~~Command:~~	~~02h~~	~~0Ah~~	~~<any>~~	~~02h~~	~~<value>~~

~~Response:~~	~~**Simple Response**~~

~~This allows you to set the speed at which Vector Drive is engaged (if the System Option Flag enables it). The default speed is 50 when Sphero first wakes up and if you change it with this command, it does *not* persist across power cycles.~~

Set Data Streaming – 11h

Command:

DID	CID	SEQ	DLEN	N	M	MASK	PCNT	MASK2
02h	11h	<any>	0ah or 0eh	16-bit val	16-bit val	32-bit val	8-bit val	32-bit val

Response:

Simple Response

Sphero supports asynchronous data streaming of certain control system and sensor parameters. This command selects the internal sampling frequency, packet size, parameter mask and optionally, the total number of packets.

param	description
N	Divisor of the maximum sensor sampling rate
M	Number of sample frames emitted per packet
MASK	Bitwise selector of data sources to stream
PCNT	Packet count 1-255 (or 0 for unlimited streaming)
MASK2	Bitwise selector of more data sources to stream (optional)

MASK and PCNT are pretty obvious but the N, M terms bear a little more explanation. Currently the control system runs at 400Hz and because it's pretty unlikely you will want to see data at that rate, N allows you to divide that down. N = 2 yields data samples at 200Hz, N = 10, 40Hz, etc. Every data sample consists of a "frame" made up of the individual sensor values as defined by the MASK. The M value defines how many frames to collect in memory before the packet is emitted. In this sense, it controls the latency of the data you receive. Increasing N and the number of bits set in MASK drive the required throughput. You should experiment with different values of N, M and MASK to see what works best for you.

The MASK2 bitfield was added to extend MASK when we developed more than 32 data sources. The API processor is implemented so that this value is optional; if it isn't included then all of its bits are set to zero. (Added in FW 1.15)

Each parameter is returned as a 16-bit signed integer. The table below defines the bits in MASK to those parameters with the indicated ranges and units. If the command is issued with a MASK of zero, then data streaming is disabled.

MASK			
bit	**sensor**	**range**	**units/LSB**
8000 0000h	accelerometer axis X, raw	-2048 to 2047	4mG
4000 0000h	accelerometer axis Y, raw	-2048 to 2047	4mG
2000 0000h	accelerometer axis Z, raw	-2048 to 2047	4mG
1000 0000h	gyro axis X, raw	-32768 to 32767	0.068 degrees
0800 0000h	gyro axis Y, raw	-32768 to 32767	0.068 degrees
0400 0000h	gyro axis Z, raw	-32768 to 32767	0.068 degrees
0200 0000h	Reserved		
0100 0000h	Reserved		
0080 0000h	Reserved		
0040 0000h	right motor back EMF, raw	-32768 to 32767	22.5 cm
0020 0000h	left motor back EMF, raw	-32768 to 32767	22.5 cm
0010 0000h	left motor, PWM, raw	-2048 to 2047	duty cycle
0008 0000h	right motor, PWM raw	-2048 to 2047	duty cycle
0004 0000h	IMU pitch angle, filtered	-179 to 180	degrees
0002 0000h	IMU roll angle, filtered	-179 to 180	degrees
0001 0000h	IMU yaw angle, filtered	-179 to 180	degrees
0000 8000h	accelerometer axis X, filtered	-32768 to 32767	1/4096 G
0000 4000h	accelerometer axis Y, filtered	-32768 to 32767	1/4096 G
0000 2000h	accelerometer axis Z, filtered	-32768 to 32767	1/4096 G
0000 1000h	gyro axis X, filtered	-20000 to 20000	0.1 dps
0000 0800h	gyro axis Y, filtered	-20000 to 20000	0.1 dps
0000 0400h	gyro axis Z, filtered	-20000 to 20000	0.1 dps
0000 0200h	Reserved		
0000 0100h	Reserved		
0000 0080h	Reserved		
0000 0040h	right motor back EMF, filtered	-32768 to 32767	22.5 cm
0000 0020h	left motor back EMF, filtered	-32768 to 32767	22.5 cm
0000 0010h	Reserved 1		
0000 0008h	Reserved 2		
0000 0004h	Reserved 3		
0000 0002h	Reserved 4		
0000 0001h	Reserved 5		

MASK2			
bit	**sensor**	**range**	**units**
8000 0000h	Quaternion Q0	-10000 to 10000	1/10000 Q
4000 0000h	Quaternion Q1	-10000 to 10000	1/10000 Q
2000 0000h	Quaternion Q2	-10000 to 10000	1/10000 Q
1000 0000h	Quaternion Q3	-10000 to 10000	1/10000 Q
0800 0000h	Odometer X	-32768 to 32767	cm
0400 0000h	Odometer Y	-32768 to 32767	cm
0200 0000h	AccelOne	0 to 8000	1 mG
0100 0000h	Velocity X	-32768 to 32767	mm/s
0080 0000h	Velocity Y	-32768 to 32767	mm/s

Configure Collision Detection – 12h

Command:

DID	CID	SEQ	DLEN	Meth	Xt	Xspd	Yt	Yspd	Dead
02h	12h	<any>	07h	<val>	<val>	<val>	<val>	<val>	<val>

Response:

Simple Response

Sphero contains a powerful analysis function to filter accelerometer data in order to detect collisions. Because this is a great example of a high-level concept that humans excel and – but robots do not – a number of parameters control the behavior. When a collision is detected an asynchronous message is generated to the client . The configuration fields are defined as follows:

param	description
Meth	Detection method type to use. Supported methods are 01h, 02h, and 03h (see the collision detection document for details). Use 00h to completely disable this service.
Xt, Yt	An 8-bit settable threshold for the X (left/right) and Y (front/back) axes of Sphero. A value of 00h disables the contribution of that axis.
Xspd, Yspd	An 8-bit settable speed value for the X and Y axes. This setting is ranged by the speed, then added to Xt, Yt to generate the final threshold value.
Dead	An 8-bit post-collision dead time to prevent retriggering; specified in 10ms increments.

The data payload of the async message is 10h bytes long and formatted as follows:

X	Y	Z	Axis	xMagnitude	yMagnitude	Speed	Timestamp
<16-bit val>	<16-bit val>	<16-bit val>	<8-bit field>	<16-bit val>	<16-bit val>	<8-bit val>	<32-bit val>

The fields are defined as:

param	description
X, Y, Z	Impact components normalized as a signed 16-bit value. Use these to determine the direction of collision event. If you don't require this level of fidelity, the two Magnitude fields encapsulate the same data in pre-processed format.
Axis	This bitfield specifies which axes had their trigger thresholds exceeded to generate the event. Bit 0 (01h) signifies the X axis and bit 1 (02h) the Y axis.
xMagnitude	This is the power that crossed the programming threshold Xt + Xs.
yMagnitude	This is the power that crossed the programming threshold Yt + Ys.
Speed	The speed of Sphero when the impact was detected.
Timestamp	The millisecond timer value at the time of impact; refer to the documentation of CID 50h and 51h to make sense of this value.

For additional information, refer to *SPAN01*, "Sphero Collision Detection Feature." Note also that this feature relies on the accelerometer range being set to ±8Gs; if altered with the Set Accelerometer Range command then don't count on it working in a useful way.

Configure Locator – 13h

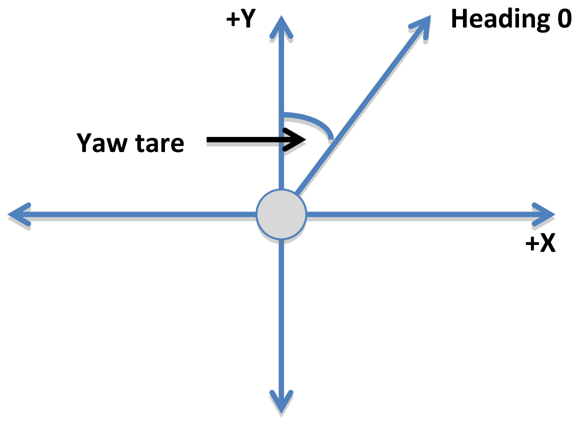
Command:

DID	CID	SEQ	DLEN	Flags	X	Y	Yaw Tare
02h	13h	<any>	02h	<8 bit val>	<16 bit signed val>	<16 bit signed val>	<16 bit signed val>

Response:

Simple Response

Through the streaming interface, Sphero provides real-time location data in the form of (X,Y) coordinates on the ground plane. When Sphero wakes up it has coordinates (0,0) and heading 0, which corresponds to facing down the positive Y-axis with the positive X-axis to your right. This command allows you to move Sphero to a new location and change the alignment of locator coordinates with IMU headings.



When Sphero receives a Set Heading command it changes which direction corresponds to heading 0. By default, the locator compensates for this by modifying its value for yaw tare so that the Y-axis is still pointing in the same real-world direction. For instance, if you wake up Sphero and drive straight, you will be driving down the Y-axis. If you use the Set Heading feature in the drive app to turn 90 degrees, you will still have heading 0, but the locator knows you have turned 90 degrees and are now facing down the X-axis. This feature can be turned off, in which case the locator knows nothing about the Set Heading command. This can lead to some strange results. For instance, if you drive using only roll commands with heading 0 and set heading commands to change direction the locator will perceive your entire path as lying on the Y-axis.

Parameters	Description
Flags	Bit 0 – Determines whether calibrate commands automatically correct the yaw tare value. When false, the positive Y axis coincides with heading 0 (assuming you do not change the yaw tare manually using this API command). Other Bits - Reserved
X, Y	The current (X,Y) coordinates of Sphero on the ground plane in centimeters.
Yaw Tare	Controls how the X,Y-plane is aligned with Sphero's heading coordinate system. When this parameter is set to zero, it means that having yaw = 0 corresponds to facing down the Y-axis in the positive direction. The value will be interpreted in the range 0-359 inclusive.

Set Accelerometer Range – 14h

Command:

DID	CID	SEQ	DLEN	Range Idx
02h	14h	<any>	02h	<8-bit val>

Response:

Simple Response

Normally, Sphero's solid state accelerometer is set for a range of ±8Gs. There may be times when you would like to alter this, say to resolve finer accelerations. This command takes an index for the supported range as explained below.

Idx	Range
0	±2Gs
1	±4Gs
2	±8Gs (default)
3	±16Gs

Note that setting this to other than the default value will have indeterminate consequences for driving and collision detection; you shouldn't expect either to work.

Read Locator – 15h

Command:

DID	CID	SEQ	DLEN
02h	15h	<any>	01h

Response:

DLEN	XPOS	YPOS	XVEL	YVEL	SOG
0Bh	<16-bit val>	<16-bit val>	<16-bit val>	<16-bit val>	<16-bit val>

This reads Sphero's current position (X,Y), component velocities and SOG (speed over ground). The position is a signed value in centimeters, the component velocities are signed cm/sec while the SOG is unsigned cm/sec.

Set RGB LED Output – 20h

Command:

DID	CID	SEQ	DLEN	RED	GREEN	BLUE	FLAG
02h	20h	<any>	05h	<value>	<value>	<value>	<bool>

Response:

Simple Response

This allows you to set the RGB LED color. The composite value is stored as the "application LED color" and immediately driven to the LED (if not overridden by a macro or orbBasic operation). If FLAG is true, the value is *also* saved as the "user LED color" which persists across power cycles and is rendered in the gap between an application connecting and sending this command.

Set Back LED Output – 21h

Command:

DID	CID	SEQ	DLEN	BRIGHT
02h	21h	<any>	02h	<value>

Response:

Simple Response

This allows you to control the brightness of the back LED. The value does not persist across power cycles.

Get RGB LED – 22h

Command:

DID	CID	SEQ	DLEN
02h	22h	<any>	01h

Response:

DLEN	RED	GREEN	BLUE
04h	<value>	<value>	<value>

This retrieves the "user LED color" which is stored in the config block (which may or may not be actively driven to the RGB LED).

Roll – 30h

Command:

DID	CID	SEQ	DLEN	Speed	Heading	Heading	STATE
02h	30h	<any>	05h	<val>	<msb>	<lsb>	<val>

Response:

Simple Response

This commands Sphero to roll along the provided vector. Both a speed and a heading are required; the latter is considered relative to the last calibrated direction. A state value is also provided. In the CES firmware, this was used to gate the control system to either obey the roll vector or ignore it and apply optimal braking to zero speed. Please refer to Appendix C for detailed information.

The client convention for heading follows the 360 degrees on a circle, relative to the ball: 0 is straight ahead, 90 is to the right, 180 is back and 270 is to the left. The valid range is 0..359.

Boost – 31h

Command:

DID	CID	SEQ	DLEN	STATE
02h	31h	<any>	02h	<bool>

Response:

Simple Response

Beginning with FW 1.46 (S2) and 3.25 (S3), this executes the boost macro from within the SSB. It takes a 1 byte parameter which is either 01h to begin boosting or 00h to stop.

Set Raw Motor Values – 33h

Command:

DID	CID	SEQ	DLEN	L-MODE	L-POWER	R-MODE	R-POWER
02h	33h	<any>	05h	<value>	<value>	<value>	<value>

Response:

Simple Response

This allows you to take over one or both of the motor output values, instead of having the stabilization system control them. Each motor (left and right) requires a mode (see below) and a power value from 0-255. This command will disable stabilization if both modes aren't "ignore" so you'll need to re-enable it via CID 02h once you're done.

MODE	description
00h	Off (motor is open circuit)
01h	Forward
02h	Reverse
03h	Brake (motor is shorted)
04h	Ignore (motor mode and power is left unchanged)

Set Motion Timeout – 34h

Command:

DID	CID	SEQ	DLEN	TIME
02h	34h	<any>	03h	16-bit val

Response:

Simple Response

This sets the ultimate timeout for the last motion command to keep Sphero from rolling away in the case of a crashed (or paused) client app. The TIME parameter is expressed in milliseconds and defaults to 2000 upon wake-up.

If the control system is enabled, the timeout triggers a stop otherwise it commands zero PWM to both motors. This "termination behavior" is inhibited if a macro is running with the flag MF_EXCLUSIVE_DRV set, or an orbBasic program is executing with a similar flag, BF_EXCLUSIVE_DRV.

Note that you must enable this action by setting System Option Flag #4.

Set Permanent Option Flags – 35h

	DID	CID	SEQ	DLEN	FLAGS
Command:	02h	35h	<any>	05h	<32-bit val>

Response:	**Simple Response**

Assigns the permanent option flags to the provided value and writes them immediately to the config block for persistence across power cycles. See below for the bit definitions.

Get Permanent Option Flags – 36h

	DID	CID	SEQ	DLEN
Command:	02h	36h	<any>	01h

	DLEN	FLAGS
Response:	05h	<32-bit val>

Returns the permanent option flags as a bitfield as defined below:

bit #	description
0	Set to prevent Sphero from immediately going to sleep when placed in the charger **and** connected over Bluetooth.
1	Set to enable Vector Drive, that is, when Sphero is stopped and a new roll command is issued it achieves the heading before moving along it.
2	Set to disable self-leveling when Sphero is inserted into the charger.
3	Set to force the tail LED always on.
4	Set to enable motion timeouts (see DID 02h, CID 34h)
5	Set to enable retail Demo Mode (when placed in the charger, ball runs a slow rainbow macro for 60 minutes and then goes to sleep).
6	Set double tap awake sensitivity to Light
7	Set double tap awake sensitivity to Heavy
8	Enable gyro max async message (NOT SUPPORTED IN VERSION 1.47)
6-31	Unassigned

Set Temporary Option Flags – 37h

	DID	CID	SEQ	DLEN	FLAGS
Command:	02h	37h	<any>	05h	<32-bit val>

Response:	**Simple Response**

Assigns the temporary option flags to the provided value. These do not persist across a power cycle. See below for the bit definitions.

Get Temporary Option Flags – 38h

	DID	CID	SEQ	DLEN
Command:	02h	38h	<any>	01h

	DLEN	FLAGS
Response:	05h	<32-bit val>

Returns the temporary option flags as a bitfield as defined below:

bit #	description
0	Enable Stop On Disconnect behavior: when the Bluetooth link transitions from connected to disconnected, Sphero is commanded to stop rolling. This is ignored if a macro or orbBasic program is running though both have option flags to allow this during their execution. This flag is cleared after it is obeyed, thus it is a one-shot.
1-31	Unassigned, return zero

Get Configuration Block – 40h

Command:

DID	CID	SEQ	DLEN	ID
02h	40h	<any>	02h	value

Response:

Simple Response

This command retrieves one of the configuration blocks. The response is a simple one; an error code of 08h is returned when the resources are currently unavailable to send the requested block back. The actual configuration block data returns in an asynchronous message of type 04h due to its length (if there is no error).

ID = 00h requests the factory configuration block

ID = 01h requests the user configuration block, which is updated with current values first

Set SSB Modifier Block – 41h

Command:

DID	CID	SEQ	DLEN	PWD	DATA
02h	41h	<any>	FFh	<32 bits>	272 bytes

Response:

Simple Response

This development-only command allows the SSB to be patched with a new modifier block, including the Boost macro. The changes take effect immediately.

Set Device Mode – 42h

Command:

DID	CID	SEQ	DLEN	MODE
02h	42h	<any>	02h	value

Response:

Simple Response

Assigns the operation mode of Sphero based on the supplied mode value:

MODE	description
00h	Normal mode
01h	User Hack mode (see below)

User Hack mode enables ASCII shell commands; refer to the associated document for a detailed list of operations.

Set Configuration Block – 43h

Command:

DID	CID	SEQ	DLEN	data
02h	43h	<any>	FFh	<block>

Response:

Simple Response

This command accepts an exact copy of the configuration block and loads it into the RAM copy of the configuration block. Then the RAM copy is saved to flash. The configuration block can be obtained by using the Get Configuration Block command.

Get Device Mode – 44h

Command:

DID	CID	SEQ	DLEN
02h	44h	<any>	01h

Response:

DLEN	Mode
02h	<val>

This returns the current device mode, 00h for Normal mode or 01h for User Hack mode.

Get SSB – 46h

Command:

DID	CID	SEQ	DLEN
02h	46h	<any>	01h

Response:

Simple Response

This command retrieves Sphero's Soul Block. The response is simple and then the actual block of soulular data returns in an asynchronous message of type 0Dh due to its 0x400 byte length (if there is no error).

Set SSB – 47h

Command:

DID	CID	SEQ	DLEN	PW	SSB
02h	46h	<any>	ffh	<32-bit val>	<400h bytes>

Response:

Simple Response

This command sets Sphero's Soul Block. The actual payload length is 404h bytes but if you use the special DLEN encoding of ffh, Sphero will know what to expect. You need to supply the password in order for it to work.

Refill Bank – 48h

Command:

DID	CID	SEQ	DLEN	TYPE
02h	48h	<any>	02h	<byte>

Response:

DLEN	CORES REMAINING
05h	<32-bit val>

This command attempts to refill either the Boost bank (TYPE = 00h) or the Shield bank (TYPE = 01h) by attempting to deduct the respective refill cost from the current number of cores. If it succeeds the bank is set to the maximum attainable for that level, the cores are spent and an API success response is returned with the lower core balance. This also commits the SSB to flash to register this transaction.

If there are not enough cores available to spend the API responds with an EEXEC error (code 08h).

Buy Consumable – 49h

Command:

DID	CID	SEQ	DLEN	ID	QTY
02h	49h	<any>	03h	<byte>	<byte>

Response:

DLEN	QTY REMAINING	CORES REMAINING
06h	<byte>	<32-bit val>

This command attempts to spend cores on consumables. The consumable ID (0..7) is given as well as the quantity requested to purchase. If the purchase succeeds the consumable count is increased, the cores are spent and an API success response is returned with both the increased consumable quantity and lower core balance. This also commits the SSB to flash to register this transaction.

If there are not enough cores available to spend or the purchase would exceed the max consumable quantity of 255 the API responds with an EEXEC error (code 08h).

Use Consumable – 4Ah

Command:

DID	CID	SEQ	DLEN	ID
02h	4Ah	<any>	02h	<byte>

Response:

DLEN	ID	QTY REMAINING
03h	<byte>	<byte>

Attempt to use a consumable (run a macro) if the quantity remaining is non-zero. On success the return message echoes the ID of this consumable and how many of them remain. Note that this will NOT immediately commit the SSB to flash.

If the associated macro is already running or the quantity remaining is zero, this returns an EEXEC error (code 08h).

Grant Cores – 4Bh

Command:

DID	CID	SEQ	DLEN	PW	QTY	FLAGS
02h	4Bh	<any>	09h	<32-bit val>	<32-bit val>	<8 bit val>

Response:

DLEN	NEW CORES COUNT
05h	<32-bit val>

This command adds the supplied number of cores. If the first bit in the flags byte is set the command immediately commits the SSB to flash. Otherwise it does not. All other bits are reserved. If the password is not accepted, this command fails without consequence.

Add XP – 4Ch

Command:

DID	CID	SEQ	DLEN	PW	QTY
02h	4Ch	<any>	06h	<32-bit val>	<8-bit val>

Response:

DLEN	XP PERCENT TO NEXT LEVEL
05h	<8-bit val>

This command increases XP by adding the supplied number of minutes of drive time and immediately commits the SSB to flash. If the password is not accepted, this command fails without consequence.

Level Up Attribute – 4Dh

Command:

DID	CID	SEQ	DLEN	PW	ATTR ID
02h	4Dh	<any>	06h	<32-bit val>	<byte>

Response:

DLEN	ATTR ID	ATTR LEVEL	ATTR PTS REMAINING
05h	<byte>	<byte>	<16-bit value>

This command attempts to increase the level of the specified attribute by spending attribute points. The IDs are 00h = Speed, 01h = Boost, 02h = Brightness and 03h = Shield. If successful the SSB is committed to flash. If there are not enough attribute points, this command returns an EEXEC error (code 08h).

If the password is not accepted, this command fails without consequence.

On success the response packet contains the attribute ID, the new level and the remaining attribute points.

Get Password Seed – 4Eh

Command:

DID	CID	SEQ	DLEN
02h	4Eh	<any>	01h

Response:

DLEN	SEED
05h	<32-bit value>

Protected Sphero commands require a password and this returns the seed to you. Refer to Appendix D for what to do next.

Enable SSB Async Messages – 4Fh

Command:

DID	CID	SEQ	DLEN	FLAG
02h	4Fh	<any>	02h	<bool>

Response:

Simple Response

Turn on/off soul block related asynchronous messages. This includes shield collision and regrowth messages, boost use and regrowth messages, XP growth and level-up messages. This feature defaults to off.

Run Macro – 50h

	DID	CID	SEQ	DLEN	ID
Command:	02h	50h	<any>	02h	<8-bit val>

Response: **Simple Response**

This attempts to execute the specified macro. Macro IDs are organized into groups: 01 – 31 are System Macros, that is, they are compiled into the Main Application. As such they are always available to be run and cannot be deleted. Macro IDs 32 – 253 are User Macros that are downloaded and persistently stored. They can be deleted in total. Macro ID 255 is a special user macro called the Temporary Macro as it is held in RAM for execution. Macro ID 254 is also a special user macro called the Stream Macro that doesn't require this call to begin execution.

This command will fail if there is currently an executing macro or the specified ID Code isn't found. In the case of the former, send an abort command first.

Save Temporary Macro – 51h

	DID	CID	SEQ	DLEN	MACRO
Command:	02h	51h	<any>	<len + 1>	<data>

Response: **Simple Response**

This stores the attached macro definition into the temporary RAM buffer for later execution. Any existing macro ID can be sent through this command and it is then renamed to ID FFh. If this command is sent while a Temporary or Stream Macro is executing it will be terminated so that its storage space can be overwritten. As with all macros, the longest definition that can be sent is 254 bytes (thus requiring DLEN to be FFh).

You must follow this with a Run Macro command to begin execution.

Save Macro – 52h

	DID	CID	SEQ	DLEN	MACRO
Command:	02h	52h	<any>	<len + 1>	<data>

Response: **Simple Response**

This stores the attached macro definition into the persistent store for later execution. This command can be sent even if other macros are executing. You will receive a failure response if you attempt to send an ID number in the System Macro range, 255 for the Temp Macro and ID of an existing user macro in the storage block. As with all macros, the longest definition that can be sent is 254 bytes (thus requiring DLEN to be FFh).

A special case of this command is to start and continue execution of the Stream Macro, ID 254. If a Temporary Macro is running it will be terminated and the Stream Macro will begin. If a Stream Macro is already running, this chunk will be appended (if there is room). Stream Macros terminate via Abort or with a special END code. Refer to the Sphero Macro documentation for more detail.

Reinit Macro Executive – 54h

Command:

DID	CID	SEQ	DLEN
02h	54h	<any>	01h

Response:

Simple Response

This terminates any running macro and reinitializes the macro system. The table of any persistent user macros is cleared.

Abort Macro – 55h

Command:

DID	CID	SEQ	DLEN
02h	55h	<any>	01h

Response:

DLEN	ID	Cmd Num	Cmd Num
04h	<any>	<msb>	<lsb>

This command aborts any executing macro and returns both its ID code and the command number currently in process. An exception is a System Macro that is executing with the UNKILLABLE flag set. A normal return code indicates the ID Code of the aborted macro as well as the command number at which execution was stopped. A return ID code of 00h indicates that no macro was running and an ID code with FFFFh as the CmdNum that the macro was unkillable.

Get Macro Status – 56h

Command:

DID	CID	SEQ	DLEN
02h	56h	<any>	01h

DLEN	ID code	Cmd Num	Cmd Num

Response:

04h	<any>	<msb>	<lsb>

This command returns the ID code and command number of the currently executing macro. If no macro is currently running, 00h is returned for the ID code while the command number is left over from the last macro.

Set Macro Parameter – 57h

Command:

DID	CID	SEQ	DLEN	Param	Val1	Val2
02h	57h	<any>	04h	<idx>	<any>	<any>

Response:

Simple Response

This command allows system globals that influence certain macro commands to be selectively altered from outside of the macro system itself. The values of Val1 and Val2 depend on the parameter index.

Index	Description
00h	Assign System Delay 1: Val1 = MSB, Val2 = LSB
01h	Assign System Delay 2: Val1 = MSB, Val2 = LSB
02h	Assign System Speed 1: Val1 = speed, Val2 = 0 (ignored)
03h	Assign System Speed 2: Val1 = speed, Val2 = 0 (ignored)
04h	Assign System Loops: Val1 = loop count, Val2 = 0 (ignored)

Details of what these system variables change are presented in the Sphero Macro document.

Append Macro Chunk – 58h

Command:

DID	CID	SEQ	DLEN	MACRO Chunk
02h	58h	<any>	<len + 1>	<data>

Response:

Simple Response

This stores the attached macro definition into the temporary RAM buffer for later execution. It is similar to the Save Temporary Macro call but allows you to build up longer temporary macros.

Any existing macro ID can be sent through this command and executed through the Run Macro call using ID FFh. If this command is sent while a Temporary or Stream Macro is executing it will be terminated so that its storage space can be overwritten. As with all macros, the longest chunk that can be sent is 254 bytes (thus requiring DLEN to be FFh).

You must follow this with a Run Macro command (ID FFh) to actually get it to go and it is best to prefix this command with an Abort call to make certain the larger buffer is completely initialized.

Erase orbBasic Storage – 60h

Command:

DID	CID	SEQ	DLEN	Area
02h	60h	<any>	02h	<val>

Response:

Simple Response

This erases any existing program in the specified storage area. Specify 00h for the temporary RAM buffer or 01h for the persistent storage area.

Append orbBasic Fragment – 61h

Command:

DID	CID	SEQ	DLEN	Area	Program Code
02h	61h	<any>	<val>	<val>	<any>

Response:

Simple Response

Sending an orbBasic program to Sphero involves appending blocks of text to existing ones in the specified storage area (00h for RAM, 01h for persistent). Complete lines are not required. A line begins with a decimal line number followed by a space and is terminated with a <LF>. See the orbBasic Interpreter document for complete information.

Possible error responses would be ORBOTIX_RSP_CODE_EPARAM if an illegal storage area is specified or ORBOTIX_RSP_CODE_EEXEC if the specified storage area is full.

Execute orbBasic Program – 62h

Command:

DID	CID	SEQ	DLEN	Area	Start Line	Start Line
02h	62h	<any>	04h	<val>	<msb>	<lsb>

Response:

Simple Response

This attempts to run a program in the specified storage area beginning at the specified line number. This command will fail if there is already an orbBasic program executing.

Abort orbBasic Program – 63h

	DID	CID	SEQ	DLEN
Command:	02h	63h	<any>	01h

Response: **Simple Response**

Aborts execution of any currently running orbBasic program.

Submit Value to Input Statement – 64h

	DID	CID	SEQ	DLEN	VAL
Command:	02h	64h	<any>	05h	(32-bit signed val)

Response: **Simple Response**

This takes the place of the typical user console in orbBasic and allows a user to answer an input request. If there is no pending input request when this API command is sent, the supplied value is ignored without error. Refer to the orbBasic language document for further information.

Commit RAM Program to Flash – 65h

	DID	CID	SEQ	DLEN
Command:	02h	65h	<any>	01h

Response: **Simple Response**

This copies the current orbBasic RAM program to persistent storage in Flash. It will fail if a program is currently executing out of Flash.

Appendix A: Enumerated Codes Quick Reference

Device IDs (defined in OrbotixMsgSet.h)	
00h	DID_CORE
01h	DID_BOOTLOADER
02h	DID_SPHERO

Core Commands, DID = 00h (defined in OrbotixMsgSet.h)	
01h	CMD_PING
02h	CMD_VERSION
03h	CMD_CONTROL_UART_TX
10h	CMD_SET_BT_NAME
11h	CMD_GET_BT_NAME
12h	CMD_SET_AUTO_RECONNECT
13h	CMD_GET_AUTO_RECONNECT
20h	CMD_GET_PWR_STATE
21h	CMD_SET_PWR_NOTIFY
22h	CMD_SLEEP
23h	GET_POWER_TRIPS
24h	SET_POWER_TRIPS
25h	SET_INACTIVE_TIMER
30h	CMD_GOTO_BL
40h	CMD_RUN_L1_DIAGS
41h	CMD_RUN_L2_DIAGS
42h	CMD_CLEAR_COUNTERS
50h	CMD_ASSIGN_TIME
51h	CMD_POLL_TIMES

Bootloader Commands, DID = 01h (defined in OrbotixMsgSet.h)	
02h	BEGIN_REFLASH
03h	HERE_IS_PAGE
04h	LEAVE_BOOTLOADER
05h	IS_PAGE_BLANK
06h	CMD_ERASE_USER_CONFIG

Sphero Commands, DID = 02h (defined in OrbotixMsgSet.h)	
01h	CMD_SET_CAL
02h	CMD_SET_STABILIZ
03h	CMD_SET_ROTATION_RATE
04h	CMD_SET_CREATION_DATE
05h	~~CMD_GET_BALL_REG_WEBSITE~~
06h	CMD_REENABLE_DEMO
07h	CMD_GET_CHASSIS_ID
08h	CMD_SET_CHASSIS_ID
09h	CMD_SELF_LEVEL
0Ah	CMD_SET_VDL
11h	CMD_SET_DATA_STREAMING
12h	CMD_SET_COLLISION_DET
13h	CMD_LOCATOR
14h	CMD_SET_ACCELERO
15h	CMD_READ_LOCATOR
20h	CMD_SET_RGB_LED
21h	CMD_SET_BACK_LED
22h	CMD_GET_RGB_LED
30h	CMD_ROLL
31h	CMD_BOOST
32h	CMD_MOVE
33h	CMD_SET_RAW_MOTORS
34h	CMD_SET_MOTION_TO
35h	CMD_SET_OPTIONS_FLAG
36h	CMD_GET_OPTIONS_FLAG
37h	CMD_SET_TEMP_OPTIONS_FLAG
38h	CMD_GET_TEMP_OPTIONS_FLAG
40h	CMD_GET_CONFIG_BLK
41h	CMD_SET_SSB_PARAMS
42h	CMD_SET_DEVICE_MODE
43h	CMD_SET_CFG_BLOCK
44h	CMD_GET_DEVICE_MODE
46h	CMD_GET_SSB
47h	CMD_SET_SSB
48h	CMD_SSB_REFILL
49h	CMD_SSB_BUY
4Ah	CMD_SSB_USE_CONSUMEABLE
4Bh	CMD_SSB_GRANT_CORES
4Ch	CMD_SSB_ADD_XP
4Dh	CMD_SSB_LEVEL_UP_ATTR
4Eh	CMD_GET_PW_SEED
4Fh	CMD_SSB_ENABLE_ASYNC
50h	CMD_RUN_MACRO
51h	CMD_SAVE_TEMP_MACRO

52h	CMD_SAVE_MACRO
54h	CMD_INIT_MACRO_EXECUTIVE
55h	CMD_ABORT_MACRO
56h	CMD_MACRO_STATUS
57h	CMD_SET_MACRO_PARAM
58h	CMD_APPEND_TEMP_MACRO_CHUNK
60h	CMD_ERASE_ORBBAS
61h	CMD_APPEND_FRAG
62h	CMD_EXEC_ORBBAS
63h	CMD_ABORT_ORBBAS
64h	CMD_ANSWER_INPUT
65h	CMD_COMMIT_TO_FLASH
70h	CMD_COMMIT_TO_FLASH

Message Response Codes (defined in OrbotixMsgSet.h)		**Description**
00h	ORBOTIX_RSP_CODE_OK	Command succeeded
01h	ORBOTIX_RSP_CODE_EGEN	General, non-specific error
02h	ORBOTIX_RSP_CODE_ECHKSUM	Received checksum failure
03h	ORBOTIX_RSP_CODE_EFRAG	Received command fragment
04h	ORBOTIX_RSP_CODE_EBAD_CMD	Unknown command ID
05h	ORBOTIX_RSP_CODE_EUNSUPP	Command currently unsupported
06h	ORBOTIX_RSP_CODE_EBAD_MSG	Bad message format
07h	ORBOTIX_RSP_CODE_EPARAM	Parameter value(s) invalid
08h	ORBOTIX_RSP_CODE_EEXEC	Failed to execute command
09h	ORBOTIX_RSP_CODE_EBAD_DID	Unknown Device ID
0Ah	ORBOTIX_RSP_CODE_MEM_BUSY	Generic RAM access needed but it is busy
0Bh	ORBOTIX_RSP_CODE_BAD_PASSWORD	Supplied password incorrect
31h	ORBOTIX_RSP_CODE_POWER_NOGOOD	Voltage too low for reflash operation
32h	ORBOTIX_RSP_CODE_PAGE_ILLEGAL	Illegal page number provided
33h	ORBOTIX_RSP_CODE_FLASH_FAIL	Page did not reprogram correctly
34h	ORBOTIX_RSP_CODE_MA_CORRUPT	Main Application corrupt
35h	ORBOTIX_RSP_CODE_MSG_TIMEOUT	Msg state machine timed out

Appendix C: Understanding the Roll Command Parameters

The roll command takes three parameters: heading, speed and a state variable (internally referred to as the "go" value). The heading parameter is self explanatory and always acted upon by the control system but the other two bear additional explanation.

As of the 1.13 Sphero firmware their relationship is as follows:

Go	Speed	Result
1	> 0	Normal driving
1	0	Rotate in place for setting heading if speed is very small. (If sent when Sphero is driving then it plugs the pitch controller for a far too aggressive stop. *This should be avoided.*)
2	X	Force fast rotation to this heading independent of speed.
0	X	Commence optimal braking to zero speed

Note that beginning in the 1.16 firmware, there are two different rotation speeds employed when acting upon the heading parameter. The first is the value set with the Set Rotation Rate command in the Sphero DID and is used for normal driving. The second is a much faster rate used to improve performance while rotating in place and setting the heading. It defaults to 1,000 degrees/sec but can be accessed through the shell commands *hss* and *hgs*.

Beginning in the 1.21 firmware the "go" parameter will also act on a value of 2 to override the speed-dependent nature of fast turning.

Revision History

Be sure to reflect this revision code in CmdGetVersioning() in cmd.c

Revision	Date	Who	Description
1.50	8 August 2013	DH, DD, FP	Updated Add XP to only send 8 bits of XP quantity, and return % toward next level. Added SSB asynch enable API command (DID 02h, CID 4Fh), added new error message ORBOTIX_RSP_CODE_MEM_BUSY. Also added TEST ONLY API command 02h, 41h to set the modifier table of the SSB which should be REMOVED before a public FW release is made.
1.49	2 August 2013	FP	Added level up and shield damage async messages. Added new collision detection methods. Added Get Password Seed command (DID 02h CID 4Eh) and added Set SSB command back in. SSB command clean-up. Added XP update async message. Added a boolean flag to the boost command.
1.48	25 July 2013	DD	Change the Application Data block function to store the creation date from the factory, renaming DID 02h CID 04h and removing DID 02h CID 04h in FW 3.33+. Added permanent option bit #8 to enable gyro async messages (off by default because they blow up Android apps). Added SSB support beginning in FW 3.37. Added three byte ID color to the Get Bluetooth Info command (DID 00h CID 11h).
1.47	29 May 2013	DD	OrbBasic RAM to 4K, added Commit RAM Program to Flash (DID 02h, CID 65h), SSB support (DID 02h, CID 46h). Reworked Boost (DID 02h, CID 31h) to work with a macro in the SSB. I've also discovered that our packet format is just a copy of the MFi Accessory Spec.
1.46	4 Apr 2013	DD	Added Control UART Tx Line (DID 00h, CID 03h), async message for exceeding a gyro axis maximum, increased orbBasic temp size to 3K, added API commands for temporary option flags (TOF), and implemented Stop On Disconnect in the TOFs.
1.40	5 Dec 2012	DD	Added Submit Value to Input Statement (DID 02h, CID 64h) supporting orbBasic.
1.39	28 Nov 2012	DD	Added a Demo Mode to the system options flag, bumped this document version to match FW release.
1.32	25 Sept 2012	DD	Added Read Locator (DID 02h, CID 15h), Input Request (DID 02h, CID 64h), fixed the description of Append orbBasic fragment, removed Re-enable Dem Mode (DID 02h, CID 06h). Now decodes 2nd byte in API prefix code as a bitfield.
1.31	18 Sept 2012	DD	Added Set Vector Drive Limit (DID 02h, CID 0Ah), changed Abort Macro return value for unkillable macros for FW 1.19+, added additional value to the go parameter in Roll and updated Appendix C.
1.30	23 July 2012	DD	Clarified the size of the PCNT parameter for data streaming. Added FFFFh special sleep duration as attempt to enter deep sleep (hardware permitting).

1.29	23 July 2012	FP	Expanded locator discussion and added diagram.
1.28	10 July 2012	DD	Added Appendix C
1.27	5 July 2012	FP	Added configure locator command (DID 02h, CID 13h).
1.25	29 June 2012	JA	Changed CMD_SET_ACCELERO from 13h to 14h Marked Boost command as not supported Changed DLEN from 05 to 06 for sleep command Changed DLEN from 02 to 03 for inactivity timeout
1.24	28 June 2012	JA	Modified the Self Level API call. Rotation rate byte is now Accuracy byte. New control system on/off bit.
1.23	25 June 2012	DH	Added range and units to streaming data. Updated collision detection response table. Added Set Accelerometer Range (DID 02h, CID 13h).
1.22	21 June 2012	DD	Added more System Option Flags.
1.21	15 June 2012	JA	Added Self Level asynchronous message details
1.20	4 June 2012	DD	Added a new field to the Get Version command (this API doc version), added Self Level command (available in FW 1.16+), top to bottom document cleanup.
1.19	9 May 2012	DD	Added second set of optional flags to stream command, API call to set the client timeout (DID 00h, CID 25h). Documented Sleep (DID 00h, CID 22h), Get/Set Power Trip Points (DID 00h, CID 23h/24h).
1.18	30 Apr 2012	DD	Added GAC field to the end of the L2 diagnostic response.
1.17	9 Apr 2012	DD	Removed termination line number in orbBasic Execute call.
1.16	06 Mar 2012	DH	Added Get/Set calls for Chassis ID Flags (DID 02h, CID 07h, 08h).
1.15	29 Feb 2012	DD	Added Get/Set calls for Option Flags (DID 02h, CID 35h, 36h).
1.14	22 Feb 2012	DD	General cleanup.
1.13	17 Feb 2012	DD	Revised collision detection programming and the async message contents to reflect implementation in FW 1.10.
1.12	3 Feb 2012	DD	Added orbBasic async channel ID codes, API commands for downloading orbBasic programs and assigning system parameters in the macro environment. Did some top-to-bottom description cleanup as well; the code had diverged from these fine sounding descriptions. Many great corrections from Fabrizio, too.
1.11	3 Nov 2011	DD	Clarified time referencing API commands, added collision detection async ID code and API command.
1.10	31 Oct 2011	DD	Clarified behavior of Get Configuration Block.
1.09	7 Oct 2011	DD	Added command to re-enable demo mode (DID 02h CID 06h)
1.08	5 Oct 2011	DD	Added a persistence flag to the RGB LED command, Get RGB LED command (DID 02h CID 22h), added a bunch of macro interface calls
1.07	28 Sept 2011	DD	Added another counter to the L2 Diag response
1.06	22 Sept 2011	DD	Added Set Motion Timeout (DID 02h, CID 34h).
1.05	21 Sept 2011	JA	Added Set (CID 04h) and Get (CID 05h) for a 32 byte application configuration block. (DID 02h)
1.04	9 Sept 2011	DD	Added async packet code 05h for pre-sleep warning
1.03	01 Sept 2011	JA	Changed “Set Auto Pair” to “Set Auto Reconnect” Added “Get Auto AutoReconnect”
1.02	29 Aug 2011	JA	Added Get Device Mode (DID 02h, CID 44h)
1.01	22 Aug 2011	DD	Clarified Set Device Mode (DID 02h, CID 42h), added Go To Sleep (DID 00h, CID 22h), Appendix B.

1.00	21 Aug 2011	JA	Added Load Config Block (DID 02h, CID 43h)
0.99	19 Aug 2011	JA	Added DID 00h, CID 22h (go to sleep)
0.98	12 Aug 2011	JA	Expanded Level 2 Diagnostics content Added DID 00h, CID 42h (clear counters)
0.97	8 Aug 2011	DD	Added definitions for macro support. Also changed the logo from Sphero to Orbotix.
0.96	2 Aug 2011	DD	Changed DID 00h, CID 11h from GetBluetoothName to GetBluetoothInfo as it now also includes the BT MAC address.
0.95	28 July 2011	DD	Added Appendix A
0.94	15 July 2011	DD	Cleaned up power state messages Added DID 02h, CIDs 40h and 42h
0.93	24 June 2011	DD	Revised DID 01h, CID 33h (raw motor control) Revised DID 00h, CID 02h (versioning) Bug in checksum calculation example
0.92	17 June 2011	DD	Added CID for streaming, DID 01h for all Bootloader services.
0.91	16 June 2011	DD	Added the commands for DID 02h.
0.90	15 May 2011	Dan Danknick	Initial stab at putting this all together.