Gen 1 Sphero Macro Executive

By Dan Danknick

document revision 0.98

Introduction

Beginning in August 2011, firmware builds of the Sphero Main Application contained a facility to execute macros, which are sequences of commands that perform actions locally on Sphero without additional client interaction. This system was intended as a way to automate and accurately reproduce actions and behaviors, with both high and low client interaction.

The Sphero macro system consists of the Executive which interprets the commands and performs the actions and the macros themselves which are linear sequences – more or less a "to-do" list. The Sphero macro system was never intended to be able to evaluate equations or make decisions, as those features are better supported in orbBasic. However they were expected to be called from orbBasic and to run in parallel as rote sequencing is a poor use of a full-blown programming language. As it is, I think you'll find macros to be a very powerful feature for games, apps and testing.

This document covers the format and behavior of the First Generation Executive (versioned as 2-4). I learned a lot from building this and seeing how our smartphone programmers used it, so major deficiencies and upgrades will be reflected in the next generation product.

Macro Format

Ultimately a macro is a linear string of bytes that is processed from beginning to end. There is no concept of jumping around in a macro (though the commands goto and gosub are implemented between macros). Symbols in all caps like MAC_END replace the underlying numerical codes that the Executive uses; you can decode these in Appendix A.

Here is the general form of a macro, with the elements explained over the next sections.

ID	Flags	ExtFlags	CMD	CMD	CMD	etc.	END

Macro IDs

This single byte is the identifier that is passed to external commands like Execute and Kill, internal commands like goto and gosub, and included in asynchronous marker messages sent to the client. The 256 ID possibilities are broken down as follows:

0	Signals that no macro is currently executing so its use is illegal.
1-31	System macros that are compiled in to the Main App for normal use. You can also call these externally under certain circumstances – some may have unexpected side effects!
32-253	User macros that are stored persistently in Flash for reuse during a single power-up session. The index table is held in RAM so they are lost once Sphero goes to sleep. These are useful for games and apps that may want to call 1 of 15 different macros on the fly without paying the latency of a temporary download.
254	Stream macro – explained below
255	Temporary macro – also explained below

User Macros

The V2 Executive maintains a block in Flash called the macro heap, indexed by an external RAM table. The heap is 2K in size and the indexing table supports 16 entries. Adding a user macro will fail if either the heap or the index is full. Since there is no provision to delete any single macro from the heap the only recourse is to reset both by reinitializing the Executive.

The Temporary Macro

This is a quick and dirty way for a client to execute a macro as it's stored in a RAM buffer and sending it also implies that you want to immediately run it. You don't need to kill the temporary macro if you send another one – the system handles that seamlessly for you. This permits lengthy behaviors to be sent as a temporary macro and then to be repurposed on the fly by sending a new temporary macro to replace it on the fly.

The Stream Macro

Since macros need to fit within the data payload space of our standard API commands, their length is limited to 254 bytes (255 is the maximum for the data length field and one byte is reserved for the packet checksum). That might seem like plenty of space but if you're sending a lot of drive and RGB LED commands, it goes quickly. Thus the creation of stream macro support.

The stream macro buffer is a full 1K bytes and has the ability to execute those of unlimited length by repeatedly accepting additional chunks. If there is space in the buffer, the new chunk is appended. If not, the remaining live command stream is shifted down to make as much room as possible to append the new chunk. If there still isn't enough room, an error code is returned which essentially means "try again in a little while."

Stream macros use the same RAM buffer as the temporary macro so executing one terminates the other. Every chunk of a stream macro that is sent is automatically terminated with a MAC_STREAM_PAUSE command in case a new chunk doesn't arrive before the macro ends. When a stream macro is active, the MAC_END command is ignored; send it if you wish. You can terminate a stream macro with the special MAC_STREAM_END command. Or you can kill it.

The byte format of a stream macro is exactly the same as all of the rest: ID byte, flag(s), commands, and optionally an end. The flags in chunks 2..N are ignored however.

An unobvious behavior of starting a stream macro is that until the MAC_STREAM_END command is encountered, the macro is still considered "running" even if it is out of commands to process. If the MF_STEALTH flag isn't set, this will have the effect of preventing Sphero from automatically going to sleep after the client inactivity time is met.

Flags

The flags byte is a bitfield that turns on or off certain useful behaviors. These flags are modal only to the macro being executed. This means gosub overrides the flags of the caller (temporarily) and goto overrides them permanently.

Bit	Hex Value	Symbolic Name	Description
0	01h	MF_MOTOR_CONTROL	Kills the drive motors automatically on exit or abort. If the stabilization system is enabled, this executes a "stop roll" command, otherwise zero PWMs are sent to the motor drivers.
1	02h	MF_EXCLUSIVE_DRV	Gives the macro exclusive control of driving, excluding commands from the Bluetooth client and orbBasic.
2	04h	MF_USE_VER3	Execute this macro with the second generation Executive (V3) – currently unimplemented
3	08h	MF_INH_IF_CONN	Inhibit execution of this macro if a smartphone client is connected.
4	10h	MF_ENDSIG	Emit a macro marker with parameter 00h when the end of the macro is reached.
5	20h	MF_STEALTH	Macro execution does NOT reset the client inactivity (sleep) timer.
6	40h	MF_UNKILLABLE	This macro cannot be aborted (killed). This is only valid for system macros and ignored for all other macro ID types.
7	80h	MF_EXT_FLAGS	The extended flags byte is present and follows this one. (I ran out of bits in this flag, so there is an extended one.)

Extended Flags

More bits for cool behaviors. None of the bits are currently assigned.

Bit	Hex Value	Symbolic Name	Description
all	n/a	n/a	Unassigned

Commands

These are the meat of the macro system and although each is explained in detail below, a few basic concepts are important to know.

- The V2 macro executive works on a 1 millisecond granularity. All times are measured in ms. Depending on how things play out, I may move this to a 10 ms granularity in the next version of the executive to save CPU time.
- Macros run in parallel with everything else in the system and as such, are non-blocking. Some macro commands even run in parallel with each other!
- Most macro commands have parameters and are followed by a post-command delay (PCD) byte. But not all.
- There is a concept of two modal system delays, SD1 and SD2. These are 16-bit times in milliseconds that special versions of certain commands can inherit. This saves space by omitting the explicit PCD byte, increases delays to over 255 ms, and also offers a flexibility where the timing of an entire macro can be changed in one place.
- There are two modal system speeds, SPD1 and SPD2 that apply to special versions of the roll command.
- There is a SystemLoops variable that is accessed by Loop Start System; it behaves exactly like usual LoopStart but the number of loops must be set via API call (see below).

In addition to setting SD1, SD2, SPD2 and SPD2 in the macro itself, these four system modals plus SystemLoops can be set via API call (DID 02h, CID 57h). Refer to the Sphero API document for complete documentation. Note that this API call can be made while the macro is executing; the altered parameters will be reflected in the next command that inherits one of the values.

Set Stabilization

Cmd	Flag	PCD
03h	<bool>	<any>

This turns on and off the control system which actively stabilizes Sphero. If you intend to drive around, you should make sure the system that allows you to do it is enabled. Note that sending raw motor commands implicitly disables the stabilization system. Flag is 00h for OFF, 01h for ON with control system reset and 02h for ON without a reset.

Set Heading

Cmd	Heading	Heading	PCD
04h	<msb>	<lsb>	<any>

This reassigns Sphero's current heading to the supplied value. The units are degrees so the valid range is 0 to 359. This forms the basis for future roll commands. For example if you assign the current heading to zero and issue a roll command along heading 90, Sphero will make a right turn.

Set Rotation Rate

Cmd	Rate
13h	<any>

Sphero's control system implements an intermediate rate limiter for the yaw axis, feeding smoothed transitions to that servo loop. This sets the maximum increment. As of firmware version 0.92 the formula for converting the rate parameter R to degrees/second is:

$$40 + \frac{R}{2}$$

which yields a smoothed range from 40 to 167 deg/s. This only applies to Roll commands; if you use the macro command Rotate Over Time this setting is bypassed.

Delay

Cmd	Time	Time
0Bh	<msb>	<lsb>

This causes an immediate delay in the execution of additional macro commands, while allowing the background ones to keep running.

Set SD1, SD2

Cmd	System Delay 1	System Delay 1
01h	<msb>	<lsb>

Cmd	System Delay 2	System Delay 2
02h	<msb>	<lsb>

Two system delay settings are provided. Certain commands inherit these values in place of the PCD byte.

Set SPD1, SPD2

Cmd	System Speed 1	System Speed 1
0fh	<msb>	<lsb>

Cmd	System Speed 2	System Speed 2
10h	<msb>	<lsb>

Two system speed settings are provided. Certain roll commands use these values in place of explicit speed values.

Roll

Cmd	Speed	Heading	Heading	PCD
05h	<any>	<msb>	<lsb>	<any>

This command gets Sphero to start rolling along the commanded speed and heading. If the stabilization system is off, this command will do nothing. A speed of 00h also engages ramped down braking of roll speed.

Roll2

Cmd	Speed	Heading	Heading	Delay	Delay
1Dh	<any>	<msb>	<lsb>	<msb>	<lsb>

This is just like the Roll command above but it accepts a 2-byte delay value.

Set Speed

Cmd	Speed	PCD
25h	<any>	<any>

This is like the Roll command but it does not effect the heading. It is especially useful when you don't know the current heading but want to stop without experiencing a turning glitch (Speed = 0).

Roll with SD1

Cmd	Heading	Heading	Speed
06h	<msb>	<lsb>	<any>

This is just like the roll command 05h but the PCD is omitted and instead derived from the SD1 value.

Roll at SPD1 (or SPD2) with SD1

Cmd	Heading	Heading
11h	<msb>	<lsb>

Cmd	Heading	Heading
12h	<msb>	<lsb>

This is the ultimate in roll commands: the speed comes from one of the system speed values and the post command delay from SD1. All you need to provide is a heading. Use command code 11h to select SPD1 and 12h for SPD2.

Send Raw Motor Commands

Cmd	Left Mode	Left Power	Right Mode	Right Power	PCD
0Ah	<see table>	<any>	<see table>	<any>	<any>

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-FFh. This command will disable stabilization if both modes aren't "ignore" so you'll need to re-enable it 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)

Rotate Over Time

Cmd	Angle	Angle	Time	Time
1Ah	<msb>	<lsb>	<msb>	<lsb>

This command drives the yaw control system directly to effect an angular change over time. The angle parameter is a signed number of degrees and time is of course in milliseconds. For example, Sphero will spin around clockwise twice in four seconds if your parameters are 720 and 4000 (the byte sequence would be 02h, D0h, 0Fh, A0h). Counterclockwise in five seconds would be -720, 5000 (bytes FDh, 30h, 13h, 88h).

NOTE: This command runs in the background. Any roll commands executed before it is finished will be ignored. In the above examples you need to be doing something for 4 and 5 seconds to give it time to finish (either other commands or simply the delay command).

Rotate Over SD1 (or SD2)

Cmd	Angle	Angle
21h	<msb>	<lsb>

Cmd	Angle	Angle
22h	<msb>	<lsb>

This is the same as Rotate Over Time but instead of requiring an immediate value, command code 21h inherits this value from System Delay 1. Likewise use code 22h to inherit from SD2.

Wait Until Stopped

Cmd	Time	Time
19h	<msb>	<lsb>

This clever command will pause execution of macros until Sphero is determined "stopped" by the stabilization system or until the provided timeout expires. You can use this, for example, at corners where you want roll commands to make sharp turns.

Loop Start

Cmd	Count
1Eh	<any>

Begins a looping block, repeating the commands between this one and Loop End the specified number of times. A count of 0 is treated as 1, neither of which do anything additional. A second Loop Start before a Loop End replaces the previous Loop Start. You can use Goto and Gosub from within loop blocks.

Loop Start System

Cmd
24h

This is a different way to begin a loop block, where the loop count is specified by API command instead of by an immediate value.

Loop End

Cmd
1Fh

Terminates a looping block. If no actual loop is in process, or if the ID of the current macro doesn't match that of the Loop Start, this command is ignored.

Comment

Cmd	Length	Length	Data
20h	<msb>	<lsb>	<…>

This is out of band data and no processing is performed upon it. The macro is aborted if the Length points to a place outside of the current macro or outside of the valid data area on the Temp or Stream macro buffer.

Set RGB LED

Cmd	Red	Green	Blue	PCD
07h	<any>	<any>	<any>	<any>

This command drives the RGB LED to the desired values. When macros are running, RGB LED commands take precedence over all others in the system (except for battery warnings).

Set RGB LED with SD2

Cmd	Red	Green	Blue
08h	<any>	<any>	<any>

Just like the command above but the delay is inherited from SD2.

Fade to LED Over Time

Cmd	Red	Green	Blue	Time	Time
14h	<any>	<any>	<any>	<msb>	<lsb>

This powerful command fades the RGB LED from its current value to the provided one over the time provided. The current LED value is from the last LED macro command. Intermediate colors are derived from the individual fractional movements of each of the red, green and blue components, not some clever movement through the color space. ☺

NOTE: This command runs in the background so you will need to provide a suitable delay to allow it to complete.

Set Back LED

Cmd	Value	PCD
09h	<any>	<any>

This controls the intensity of the blue "aiming" LED. That's it.

Goto

Cmd	Target ID
0Ch	<any *>

You can chain macros with this and the Gosub command. The sole parameter is the Macro ID of where you want to go to. If the target ID doesn't exist, the macro aborts. If it does then control is transferred with the current system state intact. The macro flags of the target macro replace those currently in use. *You cannot specify the Stream Macro ID as a destination.

Gosub

Cmd	Target ID
0Dh	<any *>

You can factor out common command sets and then call them using Gosub. An illegal target aborts the macro and it is ignored in stream macro mode. The call stack is currently one level deep and once it's full this command is just ignored. There is no explicit return needed, just a macro end command. *Like Goto, you cannot specify the Stream Macro ID as a destination.

Branch On Collision

Cmd	Target ID
23h	<any *>

This command ties the macro system to collision detection system as of FW ver 1.10 and Macro Ver 4. When enabled, the collision detection system sets a flag which the macro executive acknowledges and then executes a Goto command to the specified Macro ID. *Like Goto, you cannot specify the Stream Macro ID as a destination. But you can set the target as ID 00h which makes sure this feature is turned off – required if you're chaining between macros that alternately enable and disable this feature.

You must program and arm the collision detector separately (through an API, macro or orbBasic command) before this macro command will have any effect.

Configure Collision Detection

Cmd	Method	Xthreshold	Xspeed	Ythreshold	Yspeed	DeadTime
27h	0..1	<any>	<any>	<any>	<any>	<any>

This configures the collision detection subsystem that ties in with the Branch On Collision command. Rather than reproduce the details here, please refer to the collision detection document.

Note that there is no PCD for this command.

Go to Sleep Now

Cmd	Time	Time
0Eh	<msb>	<lsb>

This puts Sphero to sleep, able to be awaken from a double-shake. The time parameter is optional and is the number of milliseconds for him to automatically reawaken. If set to zero, he goes to sleep forever. If set to FFFFh the actual time is inherited from the API command (DID 00h, CID 22h). Which just proves that the API command calls a system macro which implements this.

End

Cmd
00h

This signals the end of a normal macro. If there is an address in the gosub stack then execution will resume after the Gosub that called it. If a stream macro is running this command is ignored.

The macro flags contain some options that can be executed at the end of a macro.

Stream End

Cmd
1Bh

This signals the end of a stream macro. Most of the same rules as above apply, but if you use this command out of a stream macro then processing will abort.

Emit Marker

Cmd	Marker
15h	<val>

This emits an asynchronous message to the client with the supplied marker value. Marker value zero is reserved for the end of macro option, so don't emit it unless you want to confuse yourself. The format of the async message payload is below:

Marker	**Macro ID**	**Command #**	**Command #**
<val>	<lsb>	<msb>	<lsb>

Async ID code 06h is reserved for macro notifications, the Marker field comes from the command and the last two bytes are the command number of this marker within the current macro. You can read more about async messages in the Sphero API document.

Tips and Tricks

- Put a Gosub command at the end of your macro and Gosub yourself. It will have the ultimate effect of running your macro to the depth of the call stack + 1 before exiting, which would currently be twice.
- Consider implementing an often-used motion in a game using the system variables and place it in a User macro ID. Then form a Temp macro that assigns these settings and uses Goto to chain to the User macro. You can easily scale the motion in both space and time.

Appendix A: Enumerated Codes Quick Reference

Macro Commands (defined in macro.h)	
00h	Macro End
01h	Set SD1
02h	Set SD2
03h	Set Stabilization
04h	Set Heading
05h	Roll
06h	Roll with SD1
07h	Set RGB LED
08h	Set RGB LED with SD2
09h	Set Front LED
0Ah	Set Raw Motor Values
0Bh	Delay
0Ch	Goto
0Dh	Gosub
0Eh	Go To Sleep
0Fh	Set SPD1
10h	Set SPD2
11h	Roll at SPD1 with SD1
12h	Roll at SPD2 with SD1
13h	Set Rotation Rate
14h	Fade to RGB
15h	Emit Marker
19h	Wait Until Stopped
1Ah	Rotate Over Time
1Bh	Stream End
1Ch	Reserved
1Dh	Roll2
1Eh	Loop Start
1Fh	Loop End
20h	Comment
21h	Rotate Over SD1
22h	Rotate Over SD2
23h	Branch On Collision
24h	Loop Start System
25h	Set Speed
27h	Configure Collision Detection

Revision History

Revision	Date	Who	Description
0.98	6 Sept 2012	Dan Danknick	Added Set Speed, Configure Collision Detection.
0.97	31 May 2012	Dan Danknick	Added LoopStartSystem and references to the associated API that assigns system modal variables.
0.96	24 Feb 2012	Dan Danknick	Executive V4. Added Branch on Collision.
0.95	2 Feb 2012	Dan Danknick	Added Rotate Over SD1 and SD2 commands.
0.94	27 Jan 2012	Dan Danknick	Executive V3. Added new commands to support looping, Roll with a two-byte delay and comments. Cannot transfer control to the Stream ID but gotos out of it are now permitted.
0.93	29 Nov 2011	Dan Danknick	Noted that the stream macro will keep the ball alive forever if the MF_STEALTH flag isn't set.
0.92	15 Nov 2011	Dan Danknick	The order of heading and speed were wrong in the Roll command description.
0.91	30 Oct 2011	Dan Danknick	Additional parameter value for Set Stabilization command, added Tips and Tricks section, clarified the use of streaming macros.
0.90	17 Oct 2011	Dan Danknick	Initial stab at putting this all together. Hugs!