

## iRobot® Create® 2 Open Interface (OI) Specification based on the iRobot® Roomba® 600



# Table of Contents

|   |    |
|---|----|
| Table of Contents .....                                       | 2  |
| iRobot Roomba® Open Interface Overview .....                  | 3  |
| Physical Connections .....                                    | 3  |
| Roomba's External Serial Port Mini-DIN Connector Pinout ..... | 3  |
| Serial Port Settings .....                                    | 4  |
| Power Saving .....  | 4  |
| Roomba Physical Dimensions .....                              | 5  |
| Roomba Internal Screw Boss Locations .....                    | 6  |
| Roomba Open Interface Modes .....                             | 7  |
| Roomba Open Interface Command Reference .....                 | 8  |
| Getting Started Commands .....                                | 8  |
| Mode Commands .....   | 10 |
| Cleaning Commands .....                                       | 11 |
| Actuator Commands .....                                       | 13 |
| Input Commands .....  | 21 |
| Roomba Open Interface Sensor Packets .....                    | 23 |
| Characters sent by iRobot devices .....                       | 26 |
| Roomba Open Interface Sensors Quick Reference .....           | 37 |
| Appendix: What's new in Create 2 .....                        | 40 |

## iRobot Roomba® Open Interface Overview

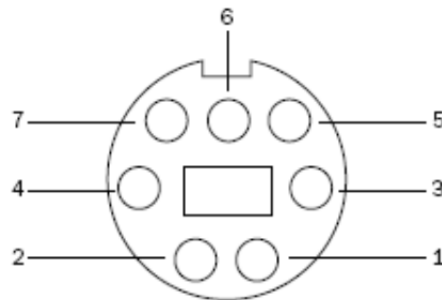
The Roomba Open Interface (OI) is a software interface for controlling and manipulating Roomba's behavior. The software interface lets you manipulate Roomba's behavior and read its sensors through a series of commands, including mode commands, actuator commands, song commands, cleaning commands, and sensor commands that you send to the Roomba's serial port by way of a PC or microcontroller that is connected to the Mini-DIN connector.

### Physical Connections

To use the OI, a processor capable of generating serial commands such as a PC or a microcontroller must be connected to the external Mini-DIN connector on Roomba. This connector provides two-way, serial communication at TTL (0 – 5V) levels. The connector also provides an unregulated direct connection to Roomba's battery, which you can use to power the OI applications. The Mini-DIN connector is located in the rear right side of Roomba, beneath a snap-away plastic guard.

### Roomba's External Serial Port Mini-DIN Connector Pinout

This diagram shows the pinout of the top view of the female connector in Roomba. Note that pins 6 and 7 are towards the outside circumference of Roomba.



| Pin | Name | Description                      |
|-----|------|----------------------------------|
| 1   | Vpwr | Roomba battery + (unregulated)   |
| 2   | Vpwr | Roomba battery + (unregulated)   |
| 3   | RXD  | 0 – 5V Serial input to Roomba    |
| 4   | TXD  | 0 – 5V Serial output from Roomba |
| 5   | BRC  | Baud Rate Change                 |
| 6   | GND  | Roomba battery ground            |
| 7   | GND  | Roomba battery ground            |

Since the RXD and TXD pins use 0 – 5V logic voltage and the PC serial ports use different voltages (RS-232 levels), it is necessary to shift voltage levels. You will need to use a level shifting cable such as the iRobot Create USB cable.

Pins 1 and 2 (Vpwr) are connected to the Roomba battery through a 200 mA PTC resettable fuse. The continuous draw from these two pins together should not exceed 200 mA. Do not draw more than 500 mA peak from these pins, or the fuse will reset.

## Serial Port Settings

**Baud:** 115200 or 19200 (see below)

**Data bits:** 8

**Parity:** None

**Stop bits:** 1

**Flow control:** None

By default, Roomba communicates at 115200 baud. If you are using a microcontroller that does not support 115200 baud, there are two ways to force Roomba to switch to 19200:

### Method 1:

While powering off Roomba, continue to hold down the Clean/Power button after the light has turned off. After about 10 seconds, Roomba plays a tune of descending pitches. Roomba will communicate at 19200 baud until the processor loses battery power or the baud rate is explicitly changed by way of the OI.

### Method 2:

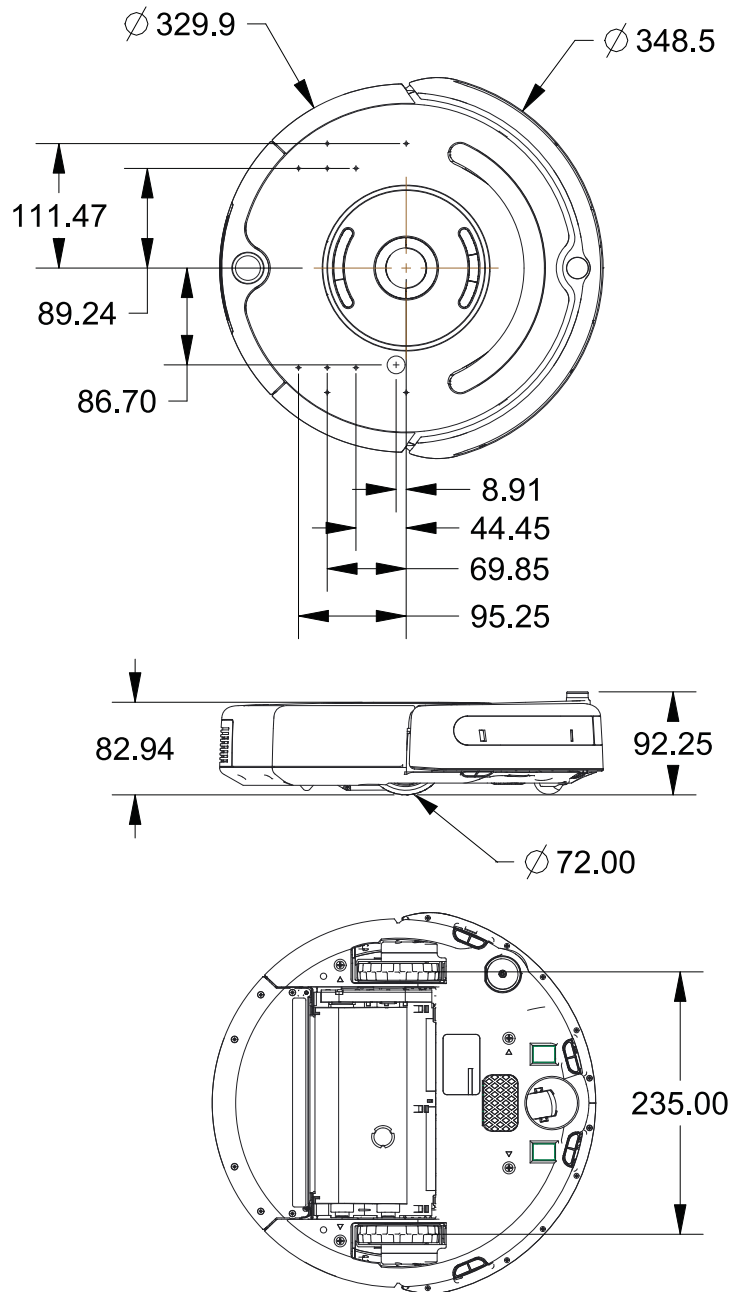
Use the Baud Rate Change pin (pin 5 on the Mini-DIN connector) to change Roomba's baud rate. After turning on Roomba, wait 2 seconds and then pulse the Baud Rate Change low three times. Each pulse should last between 50 and 500 milliseconds. Roomba will communicate at 19200 baud until the processor loses battery power or the baud rate is explicitly changed by way of the OI.

## Power Saving

In Passive mode, Roomba will sleep after 5 minutes of inactivity to preserve battery power. In Safe and Full modes, Roomba will never sleep, and if left in this state for an extended period of time, will deeply discharge its battery, even if plugged into the charger. The charger will power Roomba in all modes, but it will not charge the battery in Safe or Full mode. It is important that when you are finished working with Roomba that you issue either the Passive or Stop command to protect the battery.

## Roomba Physical Dimensions

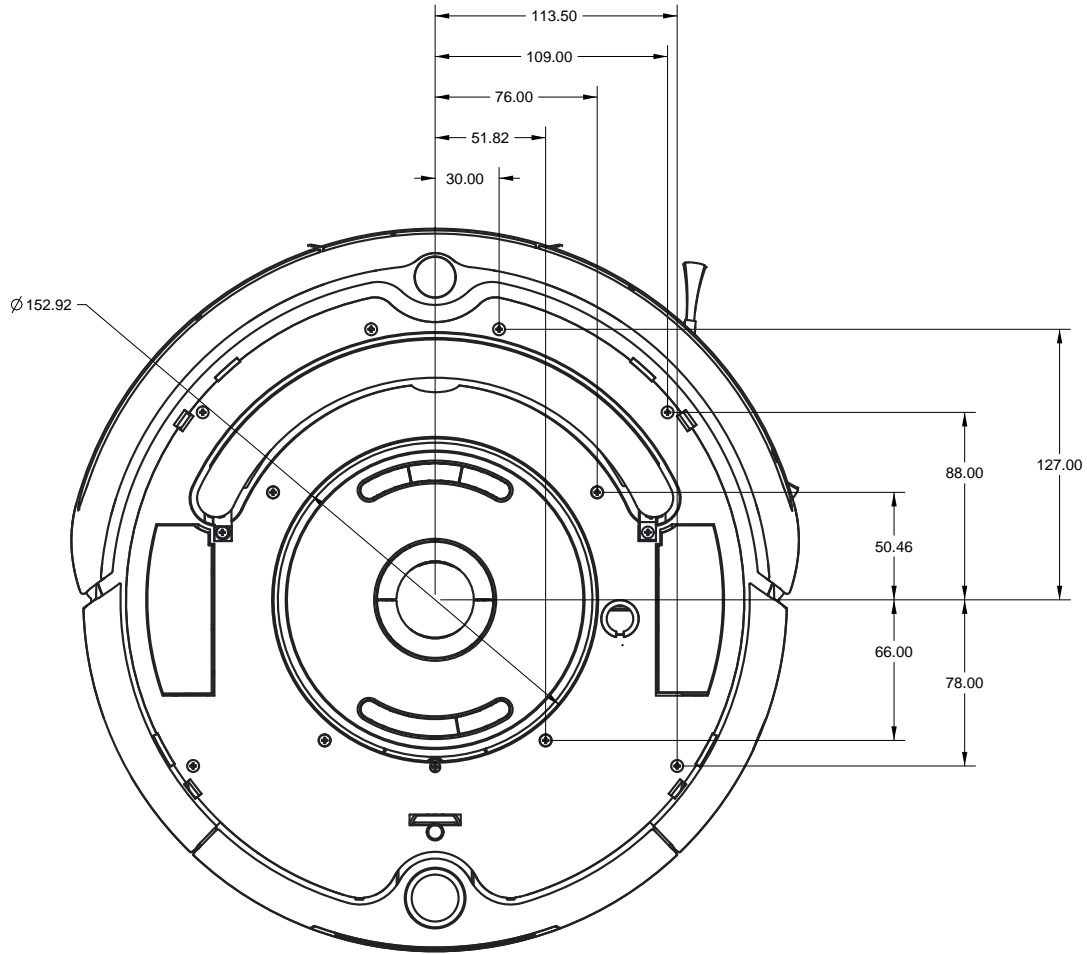
### iRobot Create 2 Anatomy



Robot Weight: ~3.5 kilograms  
Encoder: 508.8 counts/revolution

## Roomba Internal Screw Boss Locations

These hole locations are for reference only. Screws may be replaced with threaded standoffs. Drilling into the robot is never a good idea!



## Roomba Open Interface Modes

The Roomba OI has four operating modes: Off, Passive, Safe, and Full. After a battery change or when power is first turned on, the OI is in “off” mode. When it is off, the OI listens at the default baud rate (115200 or 19200 - see Serial Port Settings above) for an OI Start command. Once it receives the Start command, you can enter into any one of the four operating modes by sending a mode command to the OI. You can also switch between operating modes at any time by sending a command to the OI for the operating mode that you want to use.

### Passive Mode

Upon sending the Start command or any one of the cleaning mode commands (e.g., Spot, Clean, Seek Dock), the OI enters into Passive mode. When the OI is in Passive mode, you can request and receive sensor data using any of the sensor commands, but you cannot change the current command parameters for the actuators (motors, speaker, lights, low side drivers, digital outputs) to something else. To change how one of the actuators operates, you must switch from Passive mode to Full mode or Safe mode.

While in Passive mode, you can read Roomba’s sensors, watch Roomba perform a cleaning cycle, and charge the battery.

In Passive mode, Roomba will go into power saving mode to conserve battery power after five minutes of inactivity. To disable sleep, pulse the BRC pin low periodically before these five minutes expire. Each pulse resets this five minute counter. (One example that would not cause the baud rate to inadvertently change is to pulse the pin low for one second, every minute, but there are other periods and duty cycles that would work, as well.)

### Safe Mode

When you send a Safe command to the OI, Roomba enters into Safe mode. Safe mode gives you full control of Roomba, with the exception of the following safety-related conditions:

- Detection of a cliff while moving forward (or moving backward with a small turning radius, less than one robot radius).
- Detection of a wheel drop (on any wheel).
- Charger plugged in and powered.

Should one of the above safety-related conditions occur while the OI is in Safe mode, Roomba stops all motors and reverts to the Passive mode.

If no commands are sent to the OI when in Safe mode, Roomba waits with all motors and LEDs off and does not respond to button presses or other sensor input.

Note that charging terminates when you enter Safe Mode, and Roomba will not power save.

### Full Mode

When you send a Full command to the OI, Roomba enters into Full mode. Full mode gives you complete control over Roomba, all of its actuators, and all of the safety-related conditions that are restricted when the OI is in Safe mode, as Full mode shuts off the cliff, wheel-drop and internal charger safety features. To put the OI back into Safe mode, you must send the Safe command.

If no commands are sent to the OI when in Full mode, Roomba waits with all motors and LEDs off and does not respond to button presses or other sensor input.

Note that charging terminates when you enter Full Mode, and Roomba will not power save.

## Roomba Open Interface Command Reference

The following is a list of all of Roomba's Open Interface commands. Each command starts with a one-byte opcode. Some of the commands must be followed by data bytes. All of Roomba's OI commands including their required data bytes are described below.

---

### NOTE:

Always send the required number of data bytes for the command, otherwise, the processor will enter and remain in a "waiting" state until all of the required data bytes are received.

---

### Getting Started Commands

The following commands start the Open Interface and get it ready for use.

---

|              |                    |                      |
|--------------|--------------------|----------------------|
| <b>Start</b> | <b>Opcode: 128</b> | <b>Data Bytes: 0</b> |
|--------------|--------------------|----------------------|

---

This command starts the OI. You must always send the Start command before sending any other commands to the OI.

- Serial sequence: [128].
- Available in modes: Passive, Safe, or Full
- Changes mode to: Passive. Roomba beeps once to acknowledge it is starting from "off" mode.

---

|              |                  |                      |
|--------------|------------------|----------------------|
| <b>Reset</b> | <b>Opcode: 7</b> | <b>Data Bytes: 0</b> |
|--------------|------------------|----------------------|

---

This command resets the robot, as if you had removed and reinserted the battery.

- Serial sequence: [7].
- Available in modes: Always available.
- Changes mode to: Off. You will have to send [128] again to re-enter Open Interface mode.

---

|             |                    |                      |
|-------------|--------------------|----------------------|
| <b>Stop</b> | <b>Opcode: 173</b> | <b>Data Bytes: 0</b> |
|-------------|--------------------|----------------------|

---

This command stops the OI. All streams will stop and the robot will no longer respond to commands. Use this command when you are finished working with the robot.

- Serial sequence: [173].
- Available in modes: Passive, Safe, or Full
- Changes mode to: Off. Roomba plays a song to acknowledge it is exiting the OI.



**Baud****Opcode: 129****Data Bytes: 1**

This command sets the baud rate in bits per second (bps) at which OI commands and data are sent according to the baud code sent in the data byte. The default baud rate at power up is 115200 bps, but the starting baud rate can be changed to 19200 by following the method outlined on page 4. Once the baud rate is changed, it persists until Roomba is power cycled by pressing the power button or removing the battery, or when the battery voltage falls below the minimum required for processor operation. You must wait 100ms after sending this command before sending additional commands at the new baud rate.

- Serial sequence: [129][Baud Code]
- Available in modes: Passive, Safe, or Full
- Changes mode to: No Change
- Baud data byte 1: Baud Code (0 - 11)

| Baud Code | Baud Rate in BPS |
|-----------|------------------|
| 0         | 300              |
| 1         | 600              |
| 2         | 1200             |
| 3         | 2400             |
| 4         | 4800             |
| 5         | 9600             |
| 6         | 14400            |
| 7         | 19200            |
| 8         | 28800            |
| 9         | 38400            |
| 10        | 57600            |
| 11        | 115200           |

## Mode Commands

Roomba has four operating modes: Off, Passive, Safe, and Full. Roomba powers on in the Off mode. The following commands change Roomba's OI mode.

---

|             |                    |                      |
|-------------|--------------------|----------------------|
| <b>Safe</b> | <b>Opcode: 131</b> | <b>Data Bytes: 0</b> |
|-------------|--------------------|----------------------|

---

This command puts the OI into Safe mode, enabling user control of Roomba. It turns off all LEDs. The OI can be in Passive, Safe, or Full mode to accept this command. If a safety condition occurs (see above) Roomba reverts automatically to Passive mode.

- Serial sequence: [131]
- Available in modes: Passive, Safe, or Full
- Changes mode to: Safe

---

**Note:** The effect and usage of the Control command (130) are identical to the Safe command (131).

---

---

|             |                    |                      |
|-------------|--------------------|----------------------|
| <b>Full</b> | <b>Opcode: 132</b> | <b>Data Bytes: 0</b> |
|-------------|--------------------|----------------------|

---

This command gives you complete control over Roomba by putting the OI into Full mode, and turning off the cliff, wheel-drop and internal charger safety features. That is, in Full mode, Roomba executes any command that you send it, even if the internal charger is plugged in, or command triggers a cliff or wheel drop condition.

- Serial sequence: [132]
- Available in modes: Passive, Safe, or Full
- Changes mode to: Full

---

**Note:** Use the Start command (128) to change the mode to Passive.

---

## Cleaning Commands

The following are commands to start Roomba's built-in cleaning modes and set the clock and schedule.

---

|              |                    |                      |
|--------------|--------------------|----------------------|
| <b>Clean</b> | <b>Opcode: 135</b> | <b>Data Bytes: 0</b> |
|--------------|--------------------|----------------------|

---

This command starts the default cleaning mode. This is the same as pressing Roomba's Clean button, and will pause a cleaning cycle if one is already in progress.

- Serial sequence: [135]
- Available in modes: Passive, Safe, or Full
- Changes mode to: Passive

---

|            |                    |                      |
|------------|--------------------|----------------------|
| <b>Max</b> | <b>Opcode: 136</b> | <b>Data Bytes: 0</b> |
|------------|--------------------|----------------------|

---

This command starts the Max cleaning mode, which will clean until the battery is dead. This command will pause a cleaning cycle if one is already in progress.

- Serial sequence: [136]
- Available in modes: Passive, Safe, or Full
- Changes mode to: Passive

---

|             |                    |                      |
|-------------|--------------------|----------------------|
| <b>Spot</b> | <b>Opcode: 134</b> | <b>Data Bytes: 0</b> |
|-------------|--------------------|----------------------|

---

This command starts the Spot cleaning mode. This is the same as pressing Roomba's Spot button, and will pause a cleaning cycle if one is already in progress.

- Serial sequence: [134]
- Available in modes: Passive, Safe, or Full
- Changes mode to: Passive

---

|                  |                    |                      |
|------------------|--------------------|----------------------|
| <b>Seek Dock</b> | <b>Opcode: 143</b> | <b>Data Bytes: 0</b> |
|------------------|--------------------|----------------------|

---

This command directs Roomba to drive onto the dock the next time it encounters the docking beams. This is the same as pressing Roomba's Dock button, and will pause a cleaning cycle if one is already in progress.

- Serial sequence: [143]
- Available in modes: Passive, Safe, or Full
- Changes mode to: Passive

---

|              |                    |                      |
|--------------|--------------------|----------------------|
| <b>Power</b> | <b>Opcode: 133</b> | <b>Data Bytes: 0</b> |
|--------------|--------------------|----------------------|

---

This command powers down Roomba. The OI can be in Passive, Safe, or Full mode to accept this command.

- Serial sequence: [133]
- Available in modes: Passive, Safe, or Full
- Changes mode to: Passive

**Schedule****Opcode: 167****Data Bytes: 15**

This command sends Roomba a new schedule. To disable scheduled cleaning, send all 0s.

- Serial sequence: [167] [Days] [Sun Hour] [Sun Minute] [Mon Hour] [Mon Minute] [Tue Hour] [Tue Minute] [Wed Hour] [Wed Minute] [Thu Hour] [Thu Minute] [Fri Hour] [Fri Minute] [Sat Hour] [Sat Minute]
- Available in modes: Passive, Safe, or Full.
- If Roomba's schedule or clock button is pressed, this command will be ignored.
- Changes mode to: No change
- Times are sent in 24 hour format. Hour (0-23) Minute (0-59)

**Days**

| Bit   | 7        | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|
| Value | Reserved | Sat | Fri | Thu | Wed | Tue | Mon | Sun |

Example:

To schedule the robot to clean at 3:00 PM on Wednesdays and 10:36 AM on Fridays, send: [167] [40] [0] [0] [0] [0] [0] [0] [15] [0] [0] [0] [10] [36] [0] [0]

To disable scheduled cleaning, send: [167] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0] [0]

**Set Day/Time****Opcode: 168****Data Bytes: 3**

This command sets Roomba's clock.

- Serial sequence: [168] [Day] [Hour] [Minute]
- Available in modes: Passive, Safe, or Full.
- If Roomba's schedule or clock button is pressed, this command will be ignored.
- Changes mode to: No change
- Time is sent in 24 hour format. Hour (0-23) Minute (0-59)

| Code | Day       |
|------|-----------|
| 0    | Sunday    |
| 1    | Monday    |
| 2    | Tuesday   |
| 3    | Wednesday |
| 4    | Thursday  |
| 5    | Friday    |
| 6    | Saturday  |

## Actuator Commands

The following commands control Roomba's actuators: wheels, brushes, vacuum, speaker, LEDs, and buttons.

---

|              |                    |                      |
|--------------|--------------------|----------------------|
| <b>Drive</b> | <b>Opcode: 137</b> | <b>Data Bytes: 4</b> |
|--------------|--------------------|----------------------|

---

This command controls Roomba's drive wheels. It takes four data bytes, interpreted as two 16-bit signed values using two's complement. ([http://en.wikipedia.org/wiki/Two%27s\\_complement](http://en.wikipedia.org/wiki/Two%27s_complement)) The first two bytes specify the average velocity of the drive wheels in millimeters per second (mm/s), with the high byte being sent first. The next two bytes specify the radius in millimeters at which Roomba will turn. The longer radii make Roomba drive straighter, while the shorter radii make Roomba turn more. The radius is measured from the center of the turning circle to the center of Roomba. A Drive command with a positive velocity and a positive radius makes Roomba drive forward while turning toward the left. A negative radius makes Roomba turn toward the right. Special cases for the radius make Roomba turn in place or drive straight, as specified below. A negative velocity makes Roomba drive backward.

### NOTE:

Internal and environmental restrictions may prevent Roomba from accurately carrying out some drive commands. For example, it may not be possible for Roomba to drive at full speed in an arc with a large radius of curvature.

- Serial sequence: [137] [Velocity high byte] [Velocity low byte] [Radius high byte] [Radius low byte]
- Available in modes: Safe or Full
- Changes mode to: No Change
- Velocity (-500 – 500 mm/s)
- Radius (-2000 – 2000 mm)

### Special cases:

Straight = 32768 or 32767 = 0x8000 or 0x7FFF

Turn in place clockwise = -1 = 0xFFFF

Turn in place counter-clockwise = 1 = 0x0001

### Example:

To drive in reverse at a velocity of -200 mm/s while turning at a radius of 500mm, send the following serial byte sequence:

[137] [255] [56] [1] [244]

#### *Explanation:*

Desired value → two's complement and convert to hex → split into 2 bytes → convert to decimal

Velocity = -200 = 0xFF38 = [0xFF] [0x38] = [255] [56]

Radius = 500 = 0x01F4 = [0x01] [0xF4] = [1] [244]

**Drive Direct**

**Opcode: 145**

**Data Bytes: 4**

---

This command lets you control the forward and backward motion of Roomba's drive wheels independently. It takes four data bytes, which are interpreted as two 16-bit signed values using two's complement. The first two bytes specify the velocity of the right wheel in millimeters per second (mm/s), with the high byte sent first. The next two bytes specify the velocity of the left wheel, in the same format. A positive velocity makes that wheel drive forward, while a negative velocity makes it drive backward.

- Serial sequence: [145] [Right velocity high byte] [Right velocity low byte] [Left velocity high byte] [Left velocity low byte]
- Available in modes: Safe or Full
- Changes mode to: No Change
- Right wheel velocity (-500 – 500 mm/s)
- Left wheel velocity (-500 – 500 mm/s)

**Drive PWM**

**Opcode: 146**

**Data Bytes: 4**

---

This command lets you control the raw forward and backward motion of Roomba's drive wheels independently. It takes four data bytes, which are interpreted as two 16-bit signed values using two's complement. The first two bytes specify the PWM of the right wheel, with the high byte sent first. The next two bytes specify the PWM of the left wheel, in the same format. A positive PWM makes that wheel drive forward, while a negative PWM makes it drive backward.

- Serial sequence: [146] [Right PWM high byte] [Right PWM low byte] [Left PWM high byte] [Left PWM low byte]
- Available in modes: Safe or Full
- Changes mode to: No Change
- Right wheel PWM (-255 – 255)
- Left wheel PWM (-255 – 255)

**Motors****Opcode: 138****Data Bytes: 1**

This command lets you control the forward and backward motion of Roomba's main brush, side brush, and vacuum independently. Motor velocity cannot be controlled with this command, all motors will run at maximum speed when enabled. The main brush and side brush can be run in either direction. The vacuum only runs forward.

Serial sequence: [138] [Motors]

- Available in modes: Safe or Full
- Changes mode to: No Change
- Bits 0-2: 0 = off, 1 = on at 100% pwm duty cycle
- Bits 3 & 4: 0 = motor's default direction, 1 = motor's opposite direction. Default direction for the side brush is counterclockwise. Default direction for the main brush/flapper is inward.

| Bit   | 7        | 6 | 5 | 4                    | 3                      | 2          | 1      | 0          |
|-------|----------|---|---|----------------------|------------------------|------------|--------|------------|
| Value | Reserved |   |   | Main Brush Direction | Side Brush Clock-wise? | Main Brush | Vacuum | Side Brush |

**Example:**

To turn on the main brush inward and the side brush clockwise, send: [138] [13]

**PWM Motors****Opcode: 144****Data Bytes: 3**

This command lets you control the speed of Roomba's main brush, side brush, and vacuum independently. With each data byte, you specify the duty cycle for the low side driver (max 128). For example, if you want to control a motor with 25% of battery voltage, choose a duty cycle of  $128 * 25\% = 32$ . The main brush and side brush can be run in either direction. The vacuum only runs forward. Positive speeds turn the motor in its default (cleaning) direction. Default direction for the side brush is counterclockwise. Default direction for the main brush/flapper is inward.

Serial sequence: [144] [Main Brush PWM] [Side Brush PWM] [Vacuum PWM]

- Available in modes: Safe or Full
- Changes mode to: No Change
- Main Brush and Side Brush duty cycle (-127 – 127)
- Vacuum duty cycle (0 – 127)

**LEDs****Opcode: 139****Data Bytes: 3**

This command controls the LEDs common to all models of Roomba 600. The power LED is specified by two data bytes: one for the color and the other for the intensity.

- Serial sequence: [139] [LED Bits] [Power Color] [Power Intensity]
- Available in modes: Safe or Full
- Changes mode to: No Change
- LED Bits (0 – 255)

**Home and Spot** use green LEDs: 0 = off, 1 = on

**Check Robot** uses an orange LED.

**Debris** uses a blue LED.

**Power** uses a bicolor (red/green) LED. The intensity and color of this LED can be controlled with 8-bit resolution.

**LED Bits (0-255)**

| Bit   | 7        | 6 | 5 | 4 | 3           | 2    | 1    | 0      |
|-------|----------|---|---|---|-------------|------|------|--------|
| Value | Reserved |   |   |   | Check Robot | Dock | Spot | Debris |

**Power LED Color (0 – 255)**

0 = green, 255 = red. Intermediate values are intermediate colors (orange, yellow, etc).

**Power LED Intensity (0 – 255)**

0 = off, 255 = full intensity. Intermediate values are intermediate intensities.

**Example:**

To turn on the Home LED and light the Power LED green at half intensity, send the serial byte sequence [139] [4] [0] [128].

**Scheduling LEDs****Opcode: 162****Data Bytes: 2**

This command controls the state of the scheduling LEDs present on the Roomba 560 and 570.

- Serial sequence: [162] [Weekday LED Bits][Scheduling LED Bits]
- Available in modes: Safe or Full
- Changes mode to: No Change
- Weekday LED Bits (0 – 255)
- Scheduling LED Bits (0 – 255)
- All use red LEDs: 0 = off, 1 = on

**Weekday LED Bits**

| Bit   | 7        | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|
| Value | Reserved | Sat | Fri | Thu | Wed | Tue | Mon | Sun |

**Scheduling LED Bits**

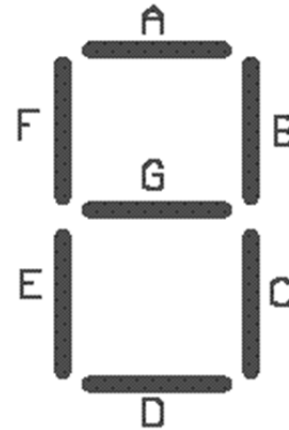
| Bit   | 7        | 6 | 5 | 4        | 3     | 2  | 1  | 0         |
|-------|----------|---|---|----------|-------|----|----|-----------|
| Value | Reserved |   |   | Schedule | Clock | AM | PM | Colon (:) |



**Digit LEDs Raw****Opcode: 163****Data Bytes: 4**

This command controls the four 7 segment displays on the Roomba 560 and 570.

- Serial sequence: [163] [Digit 3 Bits] [Digit 2 Bits] [Digit 1 Bits] [Digit 0 Bits]
- Available in modes: Safe or Full
- Changes mode to: No Change
- Digit N Bits (0 – 255)
- All use red LEDs: 0 = off, 1 = on. Digits are ordered from left to right on the robot 3,2,1,0.

**Digit N Bits**

| Bit   | 7        | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|----------|---|---|---|---|---|---|---|
| Value | Reserved | G | F | E | D | C | B | A |

**Buttons****Opcode: 165****Data Bytes: 1**

This command lets you push Roomba's buttons. The buttons will automatically release after 1/6<sup>th</sup> of a second.

- Serial sequence: [165] [Buttons]
- Available in modes: Passive, Safe, or Full
- Changes mode to: No Change
- Buttons (0-255) 1 = Push Button, 0 = Release Button

**Buttons**

| Bit   | 7     | 6        | 5   | 4    | 3      | 2    | 1    | 0     |
|-------|-------|----------|-----|------|--------|------|------|-------|
| Value | Clock | Schedule | Day | Hour | Minute | Dock | Spot | Clean |

**Digit LEDs ASCII****Opcode: 164****Data Bytes: 4**

This command controls the four 7 segment displays on the Roomba 560 and 570 using ASCII character codes. Because a 7 segment display is not sufficient to display alphabetic characters properly, all characters are an approximation, and not all ASCII codes are implemented.

- Serial sequence: [164] [Digit 3 ASCII] [Digit 2 ASCII] [Digit 1 ASCII] [Digit 0 ASCII]
- Available in modes: Safe or Full
- Changes mode to: No Change
- Digit N ASCII (32 – 126)
- All use red LEDs. Digits are ordered from left to right on the robot 3,2,1,0.

**Example:**

To write ABCD to the display, send the serial byte sequence: [164] [65] [66] [67] [68]

**Table of ASCII codes:**

| Code | Display | Code    | Display | Code        | Display | Code    | Display |
|------|---------|---------|---------|-------------|---------|---------|---------|
| 32   |         | 53      | 5       | 70, 102     | F       | 86, 118 | V       |
| 33   | !       | 54      | 6       | 71, 103     | G       | 87, 119 | W       |
| 34   | "       | 55      | 7       | 72, 104     | H       | 88, 120 | X       |
| 35   | #       | 56      | 8       | 73, 105     | I       | 89, 121 | Y       |
| 37   | %       | 57      | 9       | 74, 106     | J       | 90, 122 | Z       |
| 38   | &       | 58      | :       | 75, 107     | K       | 91, 40  | [       |
| 39   | '       | 59      | ;       | 76, 108     | L       | 92      | \       |
| 44   | ,       | 60      | i       | 77, 109     | M       | 93, 41  | ]       |
| 45   | -       | 61      | =       | 78, 110     | N       | 94      | ^       |
| 46   | .       | 62      | z       | 79, 111     | O       | 95      | _       |
| 47   | /       | 63      | ?       | 80, 112     | P       | 96      | '       |
| 48   | 0       | 65, 97  | A       | 81, 113     | Q       | 123     | {       |
| 49   | 1       | 66, 98  | B       | 82, 114     | R       | 124     | —       |
| 50   | 2       | 67, 99  | C       | 83, 36, 115 | S       | 125     | }       |
| 51   | 3       | 68, 100 | D       | 84, 116     | T       | 126     | ~       |
| 52   | 4       | 69, 101 | E       | 85, 117     | U       |         |         |

**Song****Opcode: 140****Data Bytes: 2N+2,**where **N** is the number of notes in the song

---

This command lets you specify up to four songs to the OI that you can play at a later time. Each song is associated with a song number. The Play command uses the song number to identify your song selection. Each song can contain up to sixteen notes. Each note is associated with a note number that uses MIDI note definitions and a duration that is specified in fractions of a second. The number of data bytes varies, depending on the length of the song specified. A one note song is specified by four data bytes. For each additional note within a song, add two data bytes.

- Serial sequence: [140] [Song Number] [Song Length] [Note Number 1] [Note Duration 1] [Note Number 2] [Note Duration 2], etc.
- Available in modes: Passive, Safe, or Full
- Changes mode to: No Change
- Song Number (0 – 4)

The song number associated with the specific song. If you send a second Song command, using the same song number, the old song is overwritten.

- Song Length (1 – 16)
- Song data bytes 3, 5, 7, etc.: Note Number (31 – 127)

The length of the song, according to the number of musical notes within the song.

The pitch of the musical note Roomba will play, according to the MIDI note numbering scheme. The lowest musical note that Roomba will play is Note #31. Roomba considers all musical notes outside the range of 31 – 127 as rest notes, and will make no sound during the duration of those notes.

- Song data bytes 4, 6, 8, etc.: Note Duration (0 – 255)

The duration of a musical note, in increments of 1/64<sup>th</sup> of a second. Example: a half-second long musical note has a duration value of 32.

| Number | Note | Frequency | Number | Note | Frequency | Number | Note | Frequency |
|--------|------|-----------|--------|------|-----------|--------|------|-----------|
| 31     | G    | 49.0      | 58     | A#   | 233.1     | 85     | C#   | 1108.8    |
| 32     | G#   | 51.9      | 59     | B    | 246.9     | 86     | D    | 1174.7    |
| 33     | A    | 55.0      | 60     | C    | 261.6     | 87     | D#   | 1244.5    |
| 34     | A#   | 58.3      | 61     | C#   | 277.2     | 88     | E    | 1318.5    |
| 35     | B    | 61.7      | 62     | D    | 293.7     | 89     | F    | 1396.9    |
| 36     | C    | 65.4      | 63     | D#   | 311.1     | 90     | F#   | 1480.0    |
| 37     | C#   | 69.3      | 64     | E    | 329.6     | 91     | G    | 1568.0    |
| 38     | D    | 73.4      | 65     | F    | 349.2     | 92     | G#   | 1661.3    |
| 39     | D#   | 77.8      | 66     | F#   | 370.0     | 93     | A    | 1760.0    |
| 40     | E    | 82.4      | 67     | G    | 392.0     | 94     | A#   | 1864.7    |
| 41     | F    | 87.3      | 68     | G#   | 415.3     | 95     | B    | 1975.6    |
| 42     | F#   | 92.5      | 69     | A    | 440.0     | 96     | C    | 2093.1    |
| 43     | G    | 98.0      | 70     | A#   | 466.2     | 97     | C#   | 2217.5    |
| 44     | G#   | 103.8     | 71     | B    | 493.9     | 98     | D    | 2349.4    |
| 45     | A    | 110.0     | 72     | C    | 523.3     | 99     | D#   | 2489.1    |
| 46     | A#   | 116.5     | 73     | C#   | 554.4     | 100    | E    | 2637.1    |
| 47     | B    | 123.5     | 74     | D    | 587.3     | 101    | F    | 2793.9    |
| 48     | C    | 130.8     | 75     | D#   | 622.3     | 102    | F#   | 2960.0    |
| 49     | C#   | 138.6     | 76     | E    | 659.3     | 103    | G    | 3136.0    |
| 50     | D    | 146.8     | 77     | F    | 698.5     | 104    | G#   | 3322.5    |
| 51     | D#   | 155.6     | 78     | F#   | 740.0     | 105    | A    | 3520.1    |
| 52     | E    | 164.8     | 79     | G    | 784.0     | 106    | A#   | 3729.4    |
| 53     | F    | 174.6     | 80     | G#   | 830.6     | 107    | B    | 3951.2    |
| 54     | F#   | 185.0     | 81     | A    | 880.0     |        |      |           |
| 55     | G    | 196.0     | 82     | A#   | 932.4     |        |      |           |
| 56     | G#   | 207.7     | 83     | B    | 987.8     |        |      |           |
| 57     | A    | 220.0     | 84     | C    | 1046.5    |        |      |           |

**Play****Opcode: 141****Data Bytes: 1**

This command lets you select a song to play from the songs added to Roomba using the Song command. You must add one or more songs to Roomba using the Song command in order for the Play command to work.

- Serial sequence: [141] [Song Number]
- Available in modes: Safe or Full
- Changes mode to: No Change
- Song Number (0 – 4)

The number of the song Roomba is to play.

## Input Commands

The following commands let you read the state of Roomba's built-in sensors, digital and analog inputs, and some internal state variables. Roomba updates these values internally every 15 ms. Do not send these commands more frequently than that.

---

|                |                    |                      |
|----------------|--------------------|----------------------|
| <b>Sensors</b> | <b>Opcode: 142</b> | <b>Data Bytes: 1</b> |
|----------------|--------------------|----------------------|

---

This command requests the OI to send a packet of sensor data bytes. There are 58 different sensor data packets. Each provides a value of a specific sensor or group of sensors.

For more information on sensor packets, refer to the next section, "*Roomba Open Interface Sensors Packets*".

- Serial sequence: [142] [Packet ID]
- Available in modes: Passive, Safe, or Full
- Changes mode to: No Change

Identifies which of the 58 sensor data packets should be sent back by the OI. A value of 6 indicates a packet with all of the sensor data. Values of 0 through 5 indicate specific subgroups of the sensor data.

---

|                   |                    |   |
|-------------------|--------------------|---|
| <b>Query List</b> | <b>Opcode: 149</b> | <b>Data Bytes: N + 1,</b><br>where <b>N</b> is the number of packets requested. |
|-------------------|--------------------|---|

---

This command lets you ask for a list of sensor packets. The result is returned once, as in the Sensors command. The robot returns the packets in the order you specify.

- Serial sequence: [149][Number of Packets][Packet ID 1][Packet ID 2]...[Packet ID N]
- Available in modes: Passive, Safe, or Full
- Changes modes to: No Change

### Example:

To get the state of the bumpers and the virtual wall sensor, send the following sequence:

[149] [2] [7] [13]

---

|               |                    |   |
|---------------|--------------------|---|
| <b>Stream</b> | <b>Opcode: 148</b> | <b>Data Bytes: N + 1,</b><br>where <b>N</b> is the number of packets requested. |
|---------------|--------------------|---|

---

This command starts a stream of data packets. The list of packets requested is sent every 15 ms, which is the rate Roomba uses to update data.

This method of requesting sensor data is best if you are controlling Roomba over a wireless network (which has poor real-time characteristics) with software running on a desktop computer.

- Serial sequence: [148] [Number of packets] [Packet ID 1] [Packet ID 2] [Packet ID 3] etc.
- Available in modes: Passive, Safe, or Full
- Changes mode to: No Change

The format of the data returned is:

[19][N-bytes][Packet ID 1][Packet 1 data...][Packet ID 2][Packet 2 data...][Checksum]

N-bytes is the number of bytes between the n-bytes byte and the checksum.

The checksum is a 1-byte value. It is the 8-bit complement of all of the bytes in the packet, excluding the checksum itself. That is, if you add all of the bytes in the packet, including the checksum, the low byte of the result will be 0.

**Example:**

To get data from Roomba's left cliff signal (packet 29) and virtual wall sensor (packet 13), send the following command string to Roomba:

[148] [2] [29] [13]

---

**NOTE:**

The left cliff signal is a 2-byte packet and the virtual wall is a 1-byte packet.

---

Roomba starts streaming data that looks like this:

|        |         |             |                         |             |                        |          |
|--------|---------|-------------|-------------------------|-------------|------------------------|----------|
| 19     | 5       | 29          | 2 25                    | 13          | 0                      | 163      |
| header | n-bytes | packet ID 1 | Packet data 1 (2 bytes) | packet ID 2 | packet data 2 (1 byte) | Checksum |

---

**NOTE:**

Checksum computation:  $(19 + 5 + 29 + 2 + 25 + 13 + 0 + 163) = 256$  and  $(256 \& 0xFF) = 0$ .

---

In the above stream segment, Roomba's left cliff signal value was 549 (0x0225) and there was no virtual wall signal.

It is up to you not to request more data than can be sent at the current baud rate in the 15 ms time slot. For example, at 115200 baud, a maximum of 172 bytes can be sent in 15 ms:

$$15 \text{ ms} / 10 \text{ bits } (8 \text{ data} + \text{start} + \text{stop}) * 115200 = 172.8$$

If more data is requested, the data stream will eventually become corrupted. This can be confirmed by checking the checksum.

The header byte and checksum can be used to align your receiving program with the data. All data chunks start with 19 and end with the 1-byte checksum.

---

**Pause/Resume Stream**

**Opcode: 150**

**Data Bytes: 1**

---

This command lets you stop and restart the steam without clearing the list of requested packets.

- Serial sequence: [150][Stream State]
- Available in modes: Passive, Safe, or Full
- Changes modes to: No Change
- Range: 0-1

An argument of 0 stops the stream without clearing the list of requested packets. An argument of 1 starts the stream using the list of packets last requested.

## Roomba Open Interface Sensor Packets

Roomba sends back one of 58 different sensor data packets, depending on the value of the packet data byte, when responding to a Sensors command, Query List command, or Stream command's request for a packet of sensor data bytes. Some packets contain groups of other packets. Some of the sensor data values are 16 bit values.

Most of the packets (numbers 7 – 58) contain the value of a single sensor or variable, which can be either 1 byte or 2 bytes. Two byte packets correspond to 16-bit values, sent high byte first.

Some of the packets (0-6, 100-107) contain groups of the single-value packets.

| Group Packet ID | Packet Size | Contains Packets |
|-----------------|-------------|------------------|
| 0               | 26          | 7 - 26           |
| 1               | 10          | 7 - 16           |
| 2               | 6           | 17 - 20          |
| 3               | 10          | 21 - 26          |
| 4               | 14          | 27 - 34          |
| 5               | 12          | 35 - 42          |
| 6               | 52          | 7 - 42           |
| 100             | 80          | 7 - 58           |
| 101             | 28          | 43 - 58          |
| 106             | 12          | 46 - 51          |
| 107             | 9           | 54 - 58          |

### Bumps and Wheel Drops

**Packet ID: 7**

**Data Bytes: 1, unsigned**

The state of the bumper (0 = no bump, 1 = bump) and wheel drop sensors (0 = wheel raised, 1 = wheel dropped) are sent as individual bits.

Range: 0 – 15

| Bit   | 7        | 6 | 5 | 4 | 3                | 2                 | 1          | 0           |
|-------|----------|---|---|---|------------------|-------------------|------------|-------------|
| Value | Reserved |   |   |   | Wheel Drop Left? | Wheel Drop Right? | Bump Left? | Bump Right? |

### Wall

**Packet ID: 8**

**Data Bytes: 1, unsigned**

The state of the wall sensor is sent as a 1 bit value (0 = no wall, 1 = wall seen).

Range: 0 – 1

**NOTE:** This packet is deprecated and only kept for backwards compatibility. It is recommended you use the "Light Bumper" (ID: 45) packet instead, which will show you all of the bumper wall signals.

**NOTE:** This packet is a binary version of the "Wall Signal" (ID: 27) packet.

**NOTE:** The wall sensor is equivalent to Light Bumper Right.

### Cliff Left

**Packet ID: 9**

**Data Bytes: 1, unsigned**

The state of the cliff sensor on the left side of Roomba is sent as a 1 bit value (0 = no cliff, 1 = cliff).

Range: 0 – 1

**NOTE:** This packet is a binary version of the “Cliff Left Signal” (ID: 28) packet.

**Cliff Front Left** **Packet ID: 10** **Data Bytes: 1, unsigned**

The state of the cliff sensor on the front left of Roomba is sent as a 1 bit value (0 = no cliff, 1 = cliff).

Range: 0 – 1

**NOTE:** This packet is a binary version of the “Cliff Front Left Signal” (ID: 29) packet.

**Cliff Front Right** **Packet ID: 11** **Data Bytes: 1, unsigned**

The state of the cliff sensor on the front right of Roomba is sent as a 1 bit value (0 = no cliff, 1 = cliff)

Range: 0 – 1

**NOTE:** This packet is a binary version of the “Cliff Front Right Signal” (ID: 30) packet.

**Cliff Right** **Packet ID: 12** **Data Bytes: 1, unsigned**

The state of the cliff sensor on the right side of Roomba is sent as a 1 bit value (0 = no cliff, 1 = cliff)

Range: 0 – 1

**NOTE:** This packet is a binary version of the “Cliff Right Signal” (ID: 31) packet.

**Virtual Wall** **Packet ID: 13** **Data Bytes: 1, unsigned**

The state of the virtual wall detector is sent as a 1 bit value (0 = no virtual wall detected, 1 = virtual wall detected).

Range: 0 – 1

**Wheel Overcurrents** **Packet ID: 14** **Data Bytes: 1, unsigned**

The state of the four wheel overcurrent sensors are sent as individual bits (0 = no overcurrent, 1 = overcurrent). There is no overcurrent sensor for the vacuum on Roomba 600.

Range: 0 – 31

| Bit   | 7        | 6 | 5 | 4          | 3           | 2          | 1        | 0          |
|-------|----------|---|---|------------|-------------|------------|----------|------------|
| Value | Reserved |   |   | Left Wheel | Right Wheel | Main Brush | Reserved | Side Brush |

**Dirt Detect** **Packet IDs: 15** **Data Bytes: 1**

The level of the dirt detect sensor.

Range: 0-255

**Unused Byte** **Packet IDs: 16** **Data Bytes: 1**

Unused bytes: One unused byte is sent after the dirt detect byte when the requested packet is 0, 1, or 6. The value of the unused byte is always 0.

Range: 0



**Infrared Character Omni****Packet ID: 17****Data Bytes: 1, unsigned**

This value identifies the 8-bit IR character currently being received by Roomba's omnidirectional receiver. A value of 0 indicates that no character is being received. These characters include those sent by the Roomba Remote, Dock, Virtual Walls, Create robots using the Send-IR command, and user-created devices.

Range: 0 – 255

**Infrared Character Left****Packet ID: 52****Data Bytes: 1, unsigned**

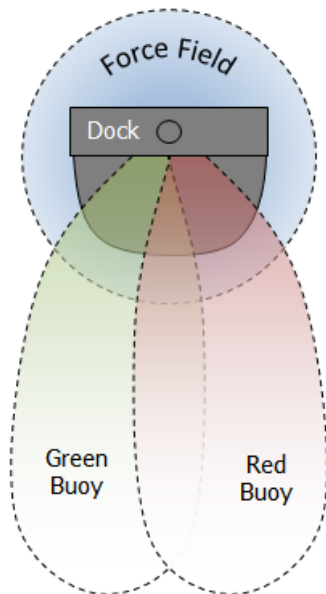
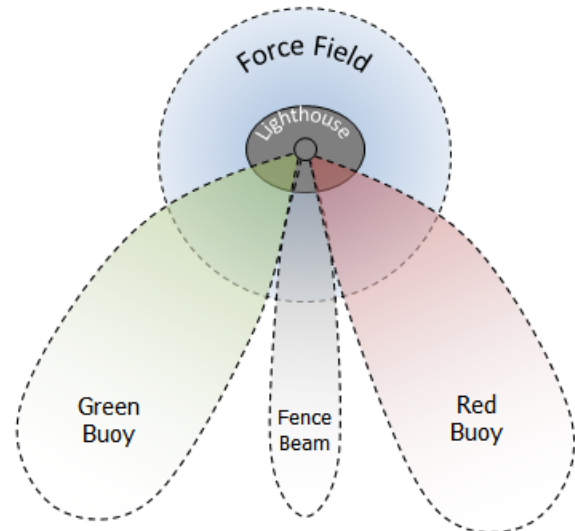
This value identifies the 8-bit IR character currently being received by Roomba's left receiver. A value of 0 indicates that no character is being received. These characters include those sent by the Roomba Remote, Dock, Virtual Walls, Create robots using the Send-IR command, and user-created devices.

Range: 0 – 255

**Infrared Character Right****Packet ID: 53****Data Bytes: 1, unsigned**

This value identifies the 8-bit IR character currently being received by Roomba's right receiver. A value of 0 indicates that no character is being received. These characters include those sent by the Roomba Remote, Dock, Virtual Walls, Create robots using the Send-IR command, and user-created devices.

Range: 0 – 255

**Dock beam configuration****Lighthouse beam configuration**

## Characters sent by iRobot devices

| Sent by iRobot Device             | Character Value                      | Character Name  |
|-----------------------------------|--------------------------------------|---|
| IR Remote Control                 | 129                                  | Left  |
|                                   | 130                                  | Forward   |
|                                   | 131                                  | Right   |
|                                   | 132                                  | Spot  |
|                                   | 133                                  | Max   |
|                                   | 134                                  | Small   |
|                                   | 135                                  | Medium  |
|                                   | 136                                  | Large / Clean   |
|                                   | 137                                  | Stop  |
|                                   | 138                                  | Power   |
|                                   | 139                                  | Arc Left  |
| 140                               | Arc Right                            |   |
| 141                               | Stop                                 |   |
| Scheduling Remote                 | 142                                  | Download  |
|                                   | 143                                  | Seek Dock   |
| Roomba Discovery Drive-on Charger | 240                                  | Reserved  |
|                                   | 248                                  | Red Buoy  |
|                                   | 244                                  | Green Buoy  |
|                                   | 242                                  | Force Field   |
|                                   | 252                                  | Red Buoy and Green Buoy   |
|                                   | 250                                  | Red Buoy and Force Field  |
|                                   | 246                                  | Green Buoy and Force Field  |
| 254                               | Red Buoy, Green Buoy and Force Field |   |
| Roomba 600 Drive-on Charger       | 160                                  | Reserved  |
|                                   | 161                                  | Force Field   |
|                                   | 164                                  | Green Buoy  |
|                                   | 165                                  | Green Buoy and Force Field  |
|                                   | 168                                  | Red Buoy  |
|                                   | 169                                  | Red Buoy and Force Field  |
|                                   | 172                                  | Red Buoy and Green Buoy   |
|                                   | 173                                  | Red Buoy, Green Buoy and Force Field  |
| Roomba 600 Virtual Wall           | 162                                  | Virtual Wall  |
| Roomba 600 Auto-on Virtual Wall   | 0LLLLOBB                             | <u>LLLL = Auto-on virtual wall ID</u> (assigned automatically by Roomba 660 robots)<br>1-10: Valid ID<br>11: Unbound<br>12-15: Reserved<br><u>BB = Beam set</u> |

| Sent by iRobot Device            | Character Value | Character Name   |
|----------------------------------|-----------------|------------------|
| Roomba 600                       |                 | 00 = Fence       |
| Auto-on Virtual Wall (continued) |                 | 01 = Force Field |
|                                  |                 | 10 = Green Buoy  |
|                                  |                 | 11 = Red Buoy    |

**Buttons****Packet ID: 18****Data Bytes: 1, unsigned**

The state of the Roomba buttons are sent as individual bits (0 = button not pressed, 1 = button pressed). The day, hour, minute, clock, and scheduling buttons that exist only on Roomba 560 and 570 will always return 0 on a Roomba 510 or 530 robot.

Range: 0 – 255

| Bit   | 7     | 6        | 5   | 4    | 3      | 2    | 1    | 0     |
|-------|-------|----------|-----|------|--------|------|------|-------|
| Value | Clock | Schedule | Day | Hour | Minute | Dock | Spot | Clean |

**Distance****Packet ID: 19****Data Bytes: 2, signed**

The distance that Roomba has traveled in millimeters since the distance it was last requested is sent as a signed 16-bit value, high byte first. This is the same as the sum of the distance traveled by both wheels divided by two. Positive values indicate travel in the forward direction; negative values indicate travel in the reverse direction. If the value is not polled frequently enough, it is capped at its minimum or maximum.

Range: -32768 – 32767

**NOTE:** Create 2 and Roomba 500/600 firmware versions prior to 3.3.0 return an incorrect value for sensors measured in millimeters. It is recommended that you read the left and right encoder counts directly (packets IDs 43 and 44) and do the unit conversion yourself.

To determine the firmware version on your robot, send a 7 via the serial port to reset it. The robot will print a long welcome message which will include the firmware version, for example: r3\_robot/tags/release-3.3.0.

**Angle****Packet ID: 20****Data Bytes: 2, signed**

The angle in degrees that Roomba has turned since the angle was last requested is sent as a signed 16-bit value, high byte first. Counter-clockwise angles are positive and clockwise angles are negative. If the value is not polled frequently enough, it is capped at its minimum or maximum.

Range: -32768 – 32767

**NOTE:** Create 2 and Roomba firmware versions 3.4.0 and earlier return an incorrect value for angle measured in degrees. The value returned must be divided by 0.324056 to get degrees. Or for more accurate results, you can read the left and right encoder counts directly (packet IDs 43 and 44) and calculate the angle yourself with this equation: angle in radians = (right wheel distance – left wheel distance) / wheel base distance.

**Charging State****Packet ID: 21****Data Bytes: 1, unsigned**

---

This code indicates Roomba's current charging state.

Range: 0 – 5

| Code | Charging State           |
|------|--------------------------|
| 0    | Not charging             |
| 1    | Reconditioning Charging  |
| 2    | Full Charging            |
| 3    | Trickle Charging         |
| 4    | Waiting                  |
| 5    | Charging Fault Condition |

**Voltage****Packet ID: 22****Data Bytes: 2, unsigned**

---

This code indicates the voltage of Roomba's battery in millivolts (mV).

Range: 0 – 65535 mV

**Current****Packet ID: 23****Data Bytes: 2, signed**

---

The current in milliamps (mA) flowing into or out of Roomba's battery. Negative currents indicate that the current is flowing out of the battery, as during normal running. Positive currents indicate that the current is flowing into the battery, as during charging.

Range: -32768 – 32767 mA

**Temperature****Packet ID: 24****Data Bytes: 1, signed**

---

The temperature of Roomba's battery in degrees Celsius.

Range: -128 – 127

**Battery Charge****Packet ID: 25****Data Bytes: 2, unsigned**

---

The current charge of Roomba's battery in milliamp-hours (mAh). The charge value decreases as the battery is depleted during running and increases when the battery is charged.

Range: 0 – 65535 mAh

**Battery Capacity****Packet ID: 26****Data Bytes: 2, unsigned**

---

The estimated charge capacity of Roomba's battery in milliamp-hours (mAh).

Range: 0 – 65535 mAh

**Wall Signal****Packet ID: 27****Data Bytes: 2, unsigned**

---

The strength of the wall signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-1023

**NOTE:** This packet is deprecated and only kept for backwards compatibility. It is recommended you use the "Light Bump Right Signal" (ID: 51) packet instead, which has a higher resolution.

**Cliff Left Signal** **Packet ID: 28** **Data Bytes: 2, unsigned**

The strength of the cliff left signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-4095

**Cliff Front Left Signal** **Packet ID: 29** **Data Bytes 2, unsigned**

The strength of the cliff front left signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-4095

**Cliff Front Right Signal** **Packet ID: 30** **Data Bytes 2, unsigned**

The strength of the cliff front right signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-4095

**Cliff Right Signal** **Packet ID: 31** **Data Bytes 2, unsigned**

The strength of the cliff right signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-4095

**Unused** **Packet ID: 32-33** **Data Bytes, 3****Charging Sources Available** **Packet ID: 34** **Data Bytes 1, unsigned**

Roomba's connection to the Home Base and Internal Charger are returned as individual bits, as below.

Range: 0-3

1 = charging source present and powered; 0 = charging source not present or not powered.

| Bit   | 7        | 6 | 5 | 4 | 3 | 2 | 1         | 0                |
|-------|----------|---|---|---|---|---|-----------|------------------|
| Value | Reserved |   |   |   |   |   | Home Base | Internal Charger |

**OI Mode** **Packet ID: 35** **Data Bytes 1, unsigned**

The current OI mode is returned. See table below.

Range: 0-3

| Number | Mode    |
|--------|---------|
| 0      | Off     |
| 1      | Passive |
| 2      | Safe    |
| 3      | Full    |

**Song Number** **Packet ID: 36** **Data Bytes 1, unsigned**

The currently selected OI song is returned.

Range: 0-15

---

**Song Playing** **Packet ID: 37** **Data Bytes 1, unsigned**

---

The state of the OI song player is returned. 1 = OI song currently playing; 0 = OI song not playing.

Range: 0-1

---

**Number of Stream Packets** **Packet ID: 38** **Data Bytes 1, unsigned**

---

The number of data stream packets is returned.

Range: 0-108

---

**Requested Velocity** **Packet ID: 39** **Data Bytes 2, signed**

---

The velocity most recently requested with a Drive command is returned as a signed 16-bit number, high byte first.

Range: -500 - 500 mm/s

---

**Requested Radius** **Packet ID: 40** **Data Bytes 2, signed**

---

The radius most recently requested with a Drive command is returned as a signed 16-bit number, high byte first.

Range: -32768 - 32767 mm

**NOTE:** Create 2 and Roomba 500/600 firmware versions prior to 3.3.0 return an incorrect value for sensors measured in millimeters. To determine the firmware version on your robot, send a 7 via the serial port to reset it. The robot will print a long welcome message which will include the firmware version, for example: r3\_robot/tags/release-3.3.0.

---

**Requested Right Velocity** **Packet ID: 41** **Data Bytes 2, signed**

---

The right wheel velocity most recently requested with a Drive Direct command is returned as a signed 16-bit number, high byte first.

Range: -500 - 500 mm/s

---

**Requested Left Velocity** **Packet ID: 42** **Data Bytes 2, signed**

---

The left wheel velocity most recently requested with a Drive Direct command is returned as a signed 16-bit number, high byte first.

Range: -500 - 500 mm/s

---

**Left Encoder Counts** **Packet ID: 43** **Data Bytes 2, signed**

---

The cumulative number of raw left encoder counts is returned as a signed 16-bit number, high byte first. This number will roll over if it passes the max value (at approx. 14.5 meters).

Range: -32768 - 32767 counts

**NOTE:** These encoders are square wave, not quadrature, so they rely on the robot's commanded velocity direction to know when to count up/down. So if the robot is trying to drive forward, and you force the

wheels to spin in reverse, the encoders will count up, (and vice-versa). Additionally, the encoders will count up when the commanded velocity is zero and the wheels spin.

To convert counts to distance, simply do a unit conversion using the equation for circle circumference.

$$N \text{ counts} * (\text{mm in 1 wheel revolution} / \text{counts in 1 wheel revolution}) = \text{mm}$$

$$N \text{ counts} * (\pi * 72.0 / 508.8) = \text{mm}$$

---

**Right Encoder Counts** **Packet ID: 44** **Data Bytes 2, signed**

---

The cumulative number of raw right encoder counts is returned as a signed 16-bit number, high byte first. This number will roll over if it passes the max value (at approx. 14.5 meters).

Range: -32768 - 32767 counts

**NOTE:** See discussion in “Left Encoder Counts” (ID: 43).

---

**Light Bumper** **Packet ID: 45** **Data Bytes 1, unsigned**

---

The light bumper detections are returned as individual bits.

| Bit   | 7        | 6 | 5                      | 4                               | 3                                | 2                               | 1                              | 0                     |
|-------|----------|---|------------------------|---------------------------------|----------------------------------|---------------------------------|--------------------------------|-----------------------|
| Value | Reserved |   | Lt<br>Bumper<br>Right? | Lt<br>Bumper<br>Front<br>Right? | Lt<br>Bumper<br>Center<br>Right? | Lt<br>Bumper<br>Center<br>Left? | Lt<br>Bumper<br>Front<br>Left? | Lt<br>Bumper<br>Left? |

Range: 0-127

**NOTE:** This is a binary version of “Light Bump Left Signal”, “Light Bump Front Left Signal”, “Light Bump Center Left Signal”, “Light Bump Center Right Signal”, “Light Bump Front Right Signal”, and “Light Bump Right Signal” packets (IDs 46-51).

---

**Light Bump Left Signal** **Packet ID: 46** **Data Bytes 2, unsigned**

---

The strength of the light bump left signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-4095

---

**Light Bump Front Left Signal** **Packet ID: 47** **Data Bytes 2, unsigned**

---

The strength of the light bump front left signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-4095

---

**Light Bump Center Left Signal** **Packet ID: 48** **Data Bytes 2, unsigned**

---

The strength of the light bump center left signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-4095

---

**Light Bump Center Right Signal** **Packet ID: 49** **Data Bytes 2, unsigned**

---

The strength of the light bump center right signal is returned as an unsigned 16-bit value, high byte first.

Range: 0-4095

**Light Bump Front Right Signal**                      **Packet ID: 50**                      **Data Bytes 2, unsigned**

---

The strength of the light bump front right signal is returned as an unsigned 16-bit value, high byte first.  
Range: 0-4095

**Light Bump Right Signal**                      **Packet ID: 51**                      **Data Bytes 2, unsigned**

---

The strength of the light bump right signal is returned as an unsigned 16-bit value, high byte first.  
Range: 0-4095

**Left Motor Current**                      **Packet ID: 54**                      **Data Bytes: 2, signed**

---

This returns the current being drawn by the left wheel motor as an unsigned 16 bit value, high byte first.  
Range: -32768 – 32767 mA

**Right Motor Current**                      **Packet ID: 55**                      **Data Bytes: 2, signed**

---

This returns the current being drawn by the right wheel motor as an unsigned 16 bit value, high byte first.  
Range: -32768 – 32767 mA

**Main Brush Motor Current**                      **Packet ID: 56**                      **Data Bytes: 2, signed**

---

This returns the current being drawn by the main brush motor as an unsigned 16 bit value, high byte first.  
Range: -32768 – 32767 mA

**Side Brush Motor Current**                      **Packet ID: 57**                      **Data Bytes: 2, signed**

---

This returns the current being drawn by the side brush motor as an unsigned 16 bit value, high byte first.  
Range: -32768 – 32767 mA

**Stasis**                      **Packet ID: 58**                      **Data Bytes: 1**

---

The stasis caster sensor returns 1 when the robot is making forward progress and 0 when it is not. It always returns 0 when the robot is turning, driving backward, or not driving. If the stasis wheel is too dirty to be read, a value of 2 is returned. If this happens, remove the stasis wheel and clean it with a damp cloth, then dry it thoroughly before reinstalling the wheel.

Range: 0 – 3

| Bit   | 7        | 6 | 5 | 4 | 3 | 2 | 1                 | 0                |
|-------|----------|---|---|---|---|---|-------------------|------------------|
| Value | Reserved |   |   |   |   |   | Stasis Dis-abled? | Stasis Toggling? |



## Roomba Open Interface Commands Quick Reference

| Command            | Opcode | Data Bytes:1        | Data Bytes:2          | Data Bytes:3       | Data Bytes:4      | Etc. |
|--------------------|--------|---------------------|-----------------------|--------------------|-------------------|------|
| Start              | 128    |                     |                       |                    |                   |      |
| Baud               | 129    | baud-code           |                       |                    |                   |      |
| Control            | 130    |                     |                       |                    |                   |      |
| Safe               | 131    |                     |                       |                    |                   |      |
| Full               | 132    |                     |                       |                    |                   |      |
| Power              | 133    |                     |                       |                    |                   |      |
| Spot               | 134    |                     |                       |                    |                   |      |
| Clean              | 135    |                     |                       |                    |                   |      |
| Max Clean          | 136    |                     |                       |                    |                   |      |
| Drive              | 137    | velocity-high       | velocity-low          | radius-high        | radius-low        |      |
| Drive Wheels       | 145    | right-velocity-high | right-velocity-low    | left-velocity-high | left-velocity-low |      |
| Motors             | 138    | motors-state        |                       |                    |                   |      |
| Pwm Motors         | 144    | main-brush-pwm      | side-brush-pwm        | vacuum-pwm         |                   |      |
| Drive Pwm          | 146    | right-pwm-high      | right-pwm-low         | left-pwm-high      | left-pwm-low      |      |
| Leds               | 139    | leds-state          | power-color           | power-intensity    |                   |      |
| Song               | 140    | song-num            | song-length           |                    |                   |      |
| Play               | 141    | song-num            |                       |                    |                   |      |
| Stream             | 148    | num-packets         |                       |                    |                   |      |
| Query List         | 149    | num-packets         |                       |                    |                   |      |
| Do Stream          | 150    | stream-state        |                       |                    |                   |      |
| Query              | 142    | packet              |                       |                    |                   |      |
| Force Seeking Dock | 143    |                     |                       |                    |                   |      |
| Scheduling Leds    | 162    | weekdays            | scheduling-leds-state |                    |                   |      |
| Digit Leds Raw     | 163    | digit-3             | digit-2               | digit-1            | digit-0           |      |
| Digit Leds Ascii   | 164    | digit-3             | digit-2               | digit-1            | digit-0           |      |
| Buttons            | 165    | buttons             |                       |                    |                   |      |
| Schedule           | 167    | days                | sun-hour              | sun-min            | mon-hour          | Etc. |
| Set Day/Time       | 168    | day                 | hour                  | minute             |                   |      |
| Stop               | 173    |                     |                       |                    |                   |      |

### LED Data Bytes 1: LED Bits (0 – 255)

**Home and Spot** use green LEDs: 0 = off, 1 = on

**Check Robot** uses an orange LED.

**Debris** uses a blue LED.

| Bit   | 7        | 6 | 5 | 4 | 3           | 2    | 1    | 0      |
|-------|----------|---|---|---|-------------|------|------|--------|
| Value | Reserved |   |   |   | Check Robot | Dock | Spot | Debris |

**Clean/Power LED**

The Clean/Power LED uses a bicolor (red/green) LED. The intensity and color of this LED can be controlled with 8-bit resolution.

**Clean/Power LED Color (0 – 255)**

0 = green, 255 = red. Intermediate values are intermediate colors (orange, yellow, etc).

**Clean/Power LED Intensity (0 – 255)**

0 = off, 255 = full intensity. Intermediate values are intermediate intensities.

**Weekday LED Bits**

| Bit   | 7        | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|
| Value | Reserved | Sat | Fri | Thu | Wed | Tue | Mon | Sun |

**Scheduling LED Bits**

| Bit   | 7        | 6 | 5 | 4        | 3     | 2  | 1  | 0         |
|-------|----------|---|---|----------|-------|----|----|-----------|
| Value | Reserved |   |   | Schedule | Clock | AM | PM | Colon (:) |

**Digit N Bits**

| Bit   | 7        | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------|----------|---|---|---|---|---|---|---|
| Value | Reserved | G | F | E | D | C | B | A |

**Buttons**

| Bit   | 7     | 6        | 5   | 4    | 3      | 2    | 1    | 0     |
|-------|-------|----------|-----|------|--------|------|------|-------|
| Value | Clock | Schedule | Day | Hour | Minute | Dock | Spot | Clean |

**Table of ASCII codes**

| Code | Display | Code    | Display | Code        | Display | Code    | Display |
|------|---------|---------|---------|-------------|---------|---------|---------|
| 32   |         | 53      | 5       | 70, 102     | F       | 86, 118 | V       |
| 33   | !       | 54      | 6       | 71, 103     | G       | 87, 119 | W       |
| 34   | "       | 55      | 7       | 72, 104     | H       | 88, 120 | X       |
| 35   | #       | 56      | 8       | 73, 105     | I       | 89, 121 | Y       |
| 37   | %       | 57      | 9       | 74, 106     | J       | 90, 122 | Z       |
| 38   | &       | 58      | :       | 75, 107     | K       | 91, 40  | [       |
| 39   | '       | 59      | ;       | 76, 108     | L       | 92      | \       |
| 44   | ,       | 60      | i       | 77, 109     | M       | 93, 41  | ]       |
| 45   | -       | 61      | =       | 78, 110     | N       | 94      | ^       |
| 46   | .       | 62      | ¿       | 79, 111     | O       | 95      | -       |
| 47   | /       | 63      | ?       | 80, 112     | P       | 96      | '       |
| 48   | 0       | 65, 97  | A       | 81, 113     | Q       | 123     | {       |
| 49   | 1       | 66, 98  | B       | 82, 114     | R       | 124     | —       |
| 50   | 2       | 67, 99  | C       | 83, 36, 115 | S       | 125     | }       |
| 51   | 3       | 68, 100 | D       | 84, 116     | T       | 126     | ~       |
| 52   | 4       | 69, 101 | E       | 85, 117     | U       |         |         |

## Baud Codes

---

| Baud Code | Baud Rate in BPS |
|-----------|------------------|
| 0         | 300              |
| 1         | 600              |
| 2         | 1200             |
| 3         | 2400             |
| 4         | 4800             |
| 5         | 9600             |
| 6         | 14400            |
| 7         | 19200            |
| 8         | 28800            |
| 9         | 38400            |
| 10        | 57600            |
| 11        | 115200           |

## Note Frequencies

---

| Number | Note | Frequency | Number | Note | Frequency | Number | Note | Frequency |
|--------|------|-----------|--------|------|-----------|--------|------|-----------|
| 31     | G    | 49.0      | 58     | A#   | 233.1     | 85     | C#   | 1108.8    |
| 32     | G#   | 51.9      | 59     | B    | 246.9     | 86     | D    | 1174.7    |
| 33     | A    | 55.0      | 60     | C    | 261.6     | 87     | D#   | 1244.5    |
| 34     | A#   | 58.3      | 61     | C#   | 277.2     | 88     | E    | 1318.5    |
| 35     | B    | 61.7      | 62     | D    | 293.7     | 89     | F    | 1396.9    |
| 36     | C    | 65.4      | 63     | D#   | 311.1     | 90     | F#   | 1480.0    |
| 37     | C#   | 69.3      | 64     | E    | 329.6     | 91     | G    | 1568.0    |
| 38     | D    | 73.4      | 65     | F    | 349.2     | 92     | G#   | 1661.3    |
| 39     | D#   | 77.8      | 66     | F#   | 370.0     | 93     | A    | 1760.0    |
| 40     | E    | 82.4      | 67     | G    | 392.0     | 94     | A#   | 1864.7    |
| 41     | F    | 87.3      | 68     | G#   | 415.3     | 95     | B    | 1975.6    |
| 42     | F#   | 92.5      | 69     | A    | 440.0     | 96     | C    | 2093.1    |
| 43     | G    | 98.0      | 70     | A#   | 466.2     | 97     | C#   | 2217.5    |
| 44     | G#   | 103.8     | 71     | B    | 493.9     | 98     | D    | 2349.4    |
| 45     | A    | 110.0     | 72     | C    | 523.3     | 99     | D#   | 2489.1    |
| 46     | A#   | 116.5     | 73     | C#   | 554.4     | 100    | E    | 2637.1    |
| 47     | B    | 123.5     | 74     | D    | 587.3     | 101    | F    | 2793.9    |
| 48     | C    | 130.8     | 75     | D#   | 622.3     | 102    | F#   | 2960.0    |
| 49     | C#   | 138.6     | 76     | E    | 659.3     | 103    | G    | 3136.0    |
| 50     | D    | 146.8     | 77     | F    | 698.5     | 104    | G#   | 3322.5    |
| 51     | D#   | 155.6     | 78     | F#   | 740.0     | 105    | A    | 3520.1    |
| 52     | E    | 164.8     | 79     | G    | 784.0     | 106    | A#   | 3729.4    |
| 53     | F    | 174.6     | 80     | G#   | 830.6     | 107    | B    | 3951.2    |
| 54     | F#   | 185.0     | 81     | A    | 880.0     |        |      |           |
| 55     | G    | 196.0     | 82     | A#   | 932.4     |        |      |           |
| 56     | G#   | 207.7     | 83     | B    | 987.8     |        |      |           |
| 57     | A    | 220.0     | 84     | C    | 1046.5    |        |      |           |

## Set Schedule Days

---

| Bit   | 7        | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|-------|----------|-----|-----|-----|-----|-----|-----|-----|
| Value | Reserved | Sat | Fri | Thu | Wed | Tue | Mon | Sun |

### Motors State

---

| Bit   | 7        | 6 | 5 | 4                    | 3                      | 2          | 1      | 0          |
|-------|----------|---|---|----------------------|------------------------|------------|--------|------------|
| Value | Reserved |   |   | Main Brush Direction | Side Brush Clock-wise? | Main Brush | Vacuum | Side Brush |

## Roomba Open Interface Sensors Quick Reference

Roomba sends back one of 58 different sensor data packets in response to a Sensors command, depending on the value of the packet ID data byte. Some packets contain groups of other packets. The sensor values are specified below in the order in which they will be sent. Some of the sensor data values are 16 bit values. These values are sent as two bytes, high byte first.

### Group Packet Sizes and Contents

| Group Packet ID | Packet Size | Contains Packets |
|-----------------|-------------|------------------|
| 0               | 26          | 7 - 26           |
| 1               | 10          | 7 - 16           |
| 2               | 6           | 17 - 20          |
| 3               | 10          | 21 - 26          |
| 4               | 14          | 27 - 34          |
| 5               | 12          | 35 - 42          |
| 6               | 52          | 7 - 42           |
| 100             | 80          | 7 - 58           |
| 101             | 28          | 43 - 58          |
| 106             | 12          | 46 - 51          |
| 107             | 9           | 54 - 58          |

### Bumps and Wheel Drops

| Bit   | 7        | 6 | 5 | 4 | 3                | 2                 | 1          | 0           |
|-------|----------|---|---|---|------------------|-------------------|------------|-------------|
| Value | Reserved |   |   |   | Wheel Drop Left? | Wheel Drop Right? | Bump Left? | Bump Right? |

### Buttons

| Bit   | 7     | 6        | 5   | 4    | 3      | 2    | 1    | 0     |
|-------|-------|----------|-----|------|--------|------|------|-------|
| Value | Clock | Schedule | Day | Hour | Minute | Dock | Spot | Clean |

### Charger Available

| Bit   | 7        | 6 | 5 | 4 | 3 | 2 | 1         | 0                |
|-------|----------|---|---|---|---|---|-----------|------------------|
| Value | Reserved |   |   |   |   |   | Home Base | Internal Charger |

### Overcurrents

| Bit   | 7        | 6 | 5 | 4          | 3           | 2          | 1        | 0          |
|-------|----------|---|---|------------|-------------|------------|----------|------------|
| Value | Reserved |   |   | Left Wheel | Right Wheel | Main Brush | Reserved | Side Brush |

### Charging State Codes

| Code | Charging state           |
|------|--------------------------|
| 0    | Not charging             |
| 1    | Reconditioning Charging  |
| 2    | Full Charging            |
| 3    | Trickle Charging         |
| 4    | Waiting                  |
| 5    | Charging Fault Condition |

### Open Interface Modes

| Number | Mode    |
|--------|---------|
| 0      | Off     |
| 1      | Passive |
| 2      | Safe    |
| 3      | Full    |

## Light Bumper

| Bit   | 7        | 6 | 5                      | 4                               | 3                                | 2                               | 1                              | 0                     |
|-------|----------|---|------------------------|---------------------------------|----------------------------------|---------------------------------|--------------------------------|-----------------------|
| Value | Reserved |   | Lt<br>Bumper<br>Right? | Lt<br>Bumper<br>Front<br>Right? | Lt<br>Bumper<br>Center<br>Right? | Lt<br>Bumper<br>Center<br>Left? | Lt<br>Bumper<br>Front<br>Left? | Lt<br>Bumper<br>Left? |

## Stasis

| Bit   | 7        | 6 | 5 | 4 | 3 | 2 | 1                     | 0                   |
|-------|----------|---|---|---|---|---|-----------------------|---------------------|
| Value | Reserved |   |   |   |   |   | Stasis Dis-<br>abled? | Stasis<br>Toggling? |

## Sensor Packet Membership Table

| Packet Group Membership |   |     |         | Packet | Name                     | Bytes | Value Range    | Units   |
|-------------------------|---|-----|---------|--------|--------------------------|-------|----------------|---------|
| 0                       | 1 | 6   | 100     | 7      | Bumps Wheeldrops         | 1     | 0 - 15         |         |
| 0                       | 1 | 6   | 100     | 8      | Wall                     | 1     | 0 - 1          |         |
| 0                       | 1 | 6   | 100     | 9      | Cliff Left               | 1     | 0 - 1          |         |
| 0                       | 1 | 6   | 100     | 10     | Cliff Front Left         | 1     | 0 - 1          |         |
| 0                       | 1 | 6   | 100     | 11     | Cliff Front Right        | 1     | 0 - 1          |         |
| 0                       | 1 | 6   | 100     | 12     | Cliff Right              | 1     | 0 - 1          |         |
| 0                       | 1 | 6   | 100     | 13     | Virtual Wall             | 1     | 0 - 1          |         |
| 0                       | 1 | 6   | 100     | 14     | Overcurrents             | 1     | 0 - 29         |         |
| 0                       | 1 | 6   | 100     | 15     | Dirt Detect              | 1     | 0 - 255        |         |
| 0                       | 1 | 6   | 100     | 16     | Unused 1                 | 1     | 0 - 255        |         |
| 0                       | 2 | 6   | 100     | 17     | Ir Opcode                | 1     | 0 - 255        |         |
| 0                       | 2 | 6   | 100     | 18     | Buttons                  | 1     | 0 - 255        |         |
| 0                       | 2 | 6   | 100     | 19     | Distance                 | 2     | -32768 - 32767 | mm      |
| 0                       | 2 | 6   | 100     | 20     | Angle                    | 2     | -32768 - 32767 | degrees |
| 0                       | 3 | 6   | 100     | 21     | Charging State           | 1     | 0 - 6          |         |
| 0                       | 3 | 6   | 100     | 22     | Voltage                  | 2     | 0 - 65535      | mV      |
| 0                       | 3 | 6   | 100     | 23     | Current                  | 2     | -32768 - 32767 | mA      |
| 0                       | 3 | 6   | 100     | 24     | Temperature              | 1     | -128 - 127     | deg C   |
| 0                       | 3 | 6   | 100     | 25     | Battery Charge           | 2     | 0 - 65535      | mAh     |
| 0                       | 3 | 6   | 100     | 26     | Battery Capacity         | 2     | 0 - 65535      | mAh     |
|                         | 4 | 6   | 100     | 27     | Wall Signal              | 2     | 0 - 1023       |         |
|                         | 4 | 6   | 100     | 28     | Cliff Left Signal        | 2     | 0 - 4095       |         |
|                         | 4 | 6   | 100     | 29     | Cliff Front Left Signal  | 2     | 0 - 4095       |         |
|                         | 4 | 6   | 100     | 30     | Cliff Front Right Signal | 2     | 0 - 4095       |         |
|                         | 4 | 6   | 100     | 31     | Cliff Right Signal       | 2     | 0 - 4095       |         |
|                         | 4 | 6   | 100     | 32     | Unused 2                 | 1     | 0 - 255        |         |
|                         | 4 | 6   | 100     | 33     | Unused 3                 | 2     | 0 - 65535      |         |
|                         | 4 | 6   | 100     | 34     | Charger Available        | 1     | 0 - 3          |         |
|                         | 5 | 6   | 100     | 35     | Open Interface Mode      | 1     | 0 - 3          |         |
|                         | 5 | 6   | 100     | 36     | Song Number              | 1     | 0 - 4          |         |
|                         | 5 | 6   | 100     | 37     | Song Playing?            | 1     | 0 - 1          |         |
|                         | 5 | 6   | 100     | 38     | Oi Stream Num Packets    | 1     | 0 - 108        |         |
|                         | 5 | 6   | 100     | 39     | Velocity                 | 2     | -500 - 500     | mm/s    |
|                         | 5 | 6   | 100     | 40     | Radius                   | 2     | -32768 - 32767 | mm      |
|                         | 5 | 6   | 100     | 41     | Velocity Right           | 2     | -500 - 500     | mm/s    |
|                         | 5 | 6   | 100     | 42     | Velocity Left            | 2     | -500 - 500     | mm/s    |
|                         |   | 100 | 101     | 43     | Encoder Counts Left      | 2     | -32768 - 32767 |         |
|                         |   | 100 | 101     | 44     | Encoder Counts Right     | 2     | -32768 - 32767 |         |
|                         |   | 100 | 101     | 45     | Light Bumper             | 1     | 0 - 127        |         |
|                         |   | 100 | 101 106 | 46     | Light Bump Left          | 2     | 0 - 4095       |         |
|                         |   | 100 | 101 106 | 47     | Light Bump Front Left    | 2     | 0 - 4095       |         |
|                         |   | 100 | 101 106 | 48     | Light Bump Center Left   | 2     | 0 - 4095       |         |
|                         |   | 100 | 101 106 | 49     | Light Bump Center Right  | 2     | 0 - 4095       |         |
|                         |   | 100 | 101 106 | 50     | Light Bump Front Right   | 2     | 0 - 4095       |         |
|                         |   | 100 | 101 106 | 51     | Light Bump Right         | 2     | 0 - 4095       |         |
|                         |   | 100 | 101     | 52     | Ir Opcode Left           | 1     | 0 - 255        |         |
|                         |   | 100 | 101     | 53     | Ir Opcode Right          | 1     | 0 - 255        |         |
|                         |   | 100 | 101 107 | 54     | Left Motor Current       | 2     | -32768 - 32767 | mA      |
|                         |   | 100 | 101 107 | 55     | Right Motor Current      | 2     | -32768 - 32767 | mA      |
|                         |   | 100 | 101 107 | 56     | Main Brush Current       | 2     | -32768 - 32767 | mA      |
|                         |   | 100 | 101 107 | 57     | Side Brush Current       | 2     | -32768 - 32767 | mA      |
|                         |   | 100 | 101 107 | 58     | Stasis                   | 1     | 0 - 3          |         |

## Appendix: What's new in Create 2

There have been a number of modifications since the Create Open Interface specification.

The default baud rate has changed from 57600 to 115200 kbps.

New commands allow full control of Roomba 600's rich user interface and scheduling functionality.

- 162: Scheduling Leds
- 163: Digit Leds Raw
- 164: Digit Leds ASCII
- 165: Buttons
- 167: Schedule
- 168: Set Day/Time
- 173: Stop

New sensor groups: 100-107

New sensor packets:

- 43: Encoder Counts Left
- 44: Encoder Counts Right
- 45: Light Bumper
- 46: Light Bump Left
- 47: Light Bump Front Left
- 48: Light Bump Center Left
- 49: Light Bump Center Right
- 50: Light Bump Front Right
- 51: Light Bump Right
- 52: IR Opcode Left
- 53: IR Opcode Right
- 54: Left Motor Current
- 55: Right Motor Current
- 56: Main Brush Current
- 57: Side Brush Current
- 58: Stasis

Scripting, demos, low side drivers, IR signal generation, and digital inputs are no longer available in Roomba OI.

Removed commands:

- 147: Digital Outputs
- 151: Send IR
- 152: Script
- 153: Play Script
- 154: Show Script
- 155: Wait Time
- 156: Wait Distance
- 157: Wait Angle
- 158: Wait Event

Removed sensors:

- Cargo Bay Digital Inputs
- Wheel-drop Castor