

The JSON File - Firmware Configuration

SteamVR™ Tracking HDK

Introduction

Each tracked object in the SteamVR™ Tracking system stores a JSON file in flash memory that describes its sensor geometry and other device properties. Each time SteamVR™ starts the JSON file is downloaded and cached on the PC.

With all firmware releases after January 1, 2017, the JSON file may configure some firmware behavior. This document describes which firmware parameters may be set through the JSON file, valid values, and how to adjust them.

Note: For information on how the JSON file is used by the SteamVR™ host PC, please refer to the HDK document **The JSON File**.

Firmware Operations

The device's firmware dedicates a region in flash memory to store the JSON file. As one of the first steps during booting, the firmware will scan this region for the presence of a JSON file. If found, the firmware will parse the file and scan for an array titled "firmware_config".

Firmware will initialize based upon the "firmware_config" section.

Note: the firmware_config section, and all its values, are optional. The firmware will load default values for any parameters not set within the JSON file.

Example "firmware_config" JSON:

```
"firmware_config" : {  
  "mode" : "controller",  
  "vrc" : true,  
  "trackpad" : true,  
  "trigger" : true,  
  "spi_flash" : false,  
  "radio" : true  
}
```

Key-Value Configuration

To verify that the JSON file is correctly loaded by the firmware, you may check the device's debug UART. The UART settings are 460800, 8n1. The debug command "config print" will list all firmware settings, whether set by the JSON file or the baked-in defaults.

If the JSON file is missing, or fails to parse, the MCU status LED will blink red once per second.

Note: the JSON file is parsed at device power-on. If you modify the JSON and re-upload it, changes will take effect after you power-cycle the device.

Adjustable Firmware Values

The following parameters are exposed and adjustable through the JSON file. The values must be set within the JSON file's "firmware_config" section. Please take care to use only the valid ranges/values listed below. While parameters are sanitized before being consumed, out-of-bound values may still result in undefined device behavior.

■ "mode"

A string value to set the device in either HMD or Controller mode. The two modes represent a collection of default system behaviors described in the following table:

Mode	IMU			Controller Interfaces (Trackpad, Haptics Motor, Trigger Sensor)
	Rate	Gyro Scale	Accelerometer Scale	
HMD	1000 Hz	500 DPS	4 G	Disabled
Controller	250 Hz	2000 DPS	8 G	Enabled through additional parameters (see below)

Valid values:

- ☐ "controller" Selects the Controller profile
- ☐ "hmd" Selects the HMD profile

Note: The physical HMD/Controller toggle switch is now ignored in favor of this parameter.

Example:

```
"mode" : "controller"
```

■ "battery_manager"

*** REMOVED IN 2017-07-10 RELEASE ***

A boolean value to control whether the Battery Manager task should run. The Battery Manager task is responsible for interacting with the fuel gauge to monitor battery status, throttle charging currents, and report values to the SteamVR™ host.

Valid values:

- ☐ true (Controller default)
- ☐ false (HMD default)

Note: When enabled, the firmware will wait until the battery charger detects a USB host prior to starting the USB stack. This is necessary as starting USB early can interfere with battery charger host detection.

Example:

```
"battery_manager" : true
```

■ BQ24297 Specific Parameters *(new with 2017-07-10 firmware release)*

Several parameters are available for configuring how the battery charger circuitry controls the battery. For detailed information about these settings, please refer to the BQ24297 Datasheet.

Note: These settings are only available for Controller mode.

Parameter	JSON Reference	Min Value	Max Value	Step Size	Default Value
Fast Charge Current Limit (mA)*	charge_current_ma	102.4	512	12.8	512
		512	3008	64	
Charge Voltage Limit (mV)	charge_voltage_limit_mv	3504	4400	16	4208
Battery Recharge Threshold (mV)	recharge_voltage_mv	100	300	200	100
Minimum System Voltage Limit (mV)	sys_min_voltage_mv	3000	3700	100	3500
Termination Current Limit (mA)	charge_term_current_ma	128	2048	128	128

Valid values are the minimum value + n * step size. For example, the Fast Charge Current can be set to 512, 576, 640,...

***Note: New with firmware 2018-04-30:** The charge current can be limited to below 512 mA, starting at 102.4. The true step size is 12.8, however, all entries into the JSON file should be whole integers rounded down. To obtain a fast charge limit of 102.4, a limit of 102 should be set in the JSON file.

Example:

```
"charge_current_ma" : 576,
"charge_voltage_limit_mv" : 3520,
"recharge_voltage_mv" : 300,
"sys_min_voltage_mv" : 3300,
"charge_term_current_ma" : 512
```

■ "batt_percentage_auto_off" *(new with 2017-07-10 firmware release)*

The minimum battery-level percentage that is needed to power the device. Should the power level drop below this threshold, the battery manager task will automatically power-down the system.

Valid range is from 0% to 50% (5% default).

Example:

```
"batt_percentage_auto_off" : 10
```

■ "radio"

A boolean value to control whether the Wireless task should run. The Wireless task manages the Nordic nRF52 radio to provide wireless support.

Valid values:

- ☐ true (Controller default)
- ☐ false (HMD default)

Example:

```
"radio" : true
```

■ “tx_power”

An integer value defining the transmit power for the controller radio, in dBm. Possible values are: -40, -20, -16, -8, -4, 0, 4 (default).

Note: This parameter is only available in Controller mode when radio is enabled.

Example:

```
"tx_power" : -4
```

■ “spi_flash”

A boolean value to control whether the device supports SPI flash data access (e.g. User Data) by the SteamVR™ host. SPI flash is only used by HMDs; it stores the display panel calibration data. SPI flash and the trackpad are mutually exclusive, since they share a Chip Select line.

Valid values:

- ☐ true (HMD default)
- ☐ false (Controller default)

Example:

```
"spi_flash" : true
```

■ “vrc”

A boolean value to control whether the Virtual Reality Controller (VRC) task should run. The VRC task handles reporting all Controller data to the SteamVR™ host, including trackpad, trigger, and button events.

Valid values:

- ☐ true (Controller default)
- ☐ false (HMD default)

Note: The VRC task handles transmitting button values even when the device is configured for HMD mode.

Example:

```
"vrc" : true
```

■ “trackpad”

A boolean value to control whether the trackpad drivers (sensor pad & haptics motor) are enabled.

Valid values:

- ☐ true
- ☐ false (default)

Note: The trackpad drivers require that the device is set to Controller mode.

Example:

```
"trackpad" : false
```

■ “haptics”

A boolean value to control whether the haptics feedback motor is enabled.

Valid values:

- ☐ true
- ☐ false (default)

Note: The haptics motor requires that the device is set to Controller mode.

Example:

```
"haptics" : false
```

■ “trigger”

A boolean value to control whether the analog trigger (Hall effect sensor) driver is enabled.

Valid values:

- ☐ true
- ☐ false (default)

Note: The trigger driver requires that the device is set to Controller mode.

Example:

```
"trigger" : false
```

■ “trigger_adc_min”, “trigger_adc_max”, “trigger_adc_zero”

Three integer values used to adjust the trigger’s Hall effect sensor’s raw measurement values.

Raw Hall effect sensor measurements are a 12-bit unsigned integer, where 0x00 indicates a strong negative field, 0xFFF indicates a strong positive field, and the midpoint indicates no magnetic field measured. The trigger measurement reported to SteamVR is an 8-bit absolute field strength measurement. These parameters are used to clamp and scale Hall effect sensor measurements into this 8-bit range.

Example of how these parameters scale a Hall effect sensor reading:

trigger_adc_min:	100	(200 default)
trigger_adc_max:	3900	(3895 default)
trigger_adc_zero:	200	(0 default)
Hall reading:	500	
.... Reading, Zero-centered:	-1300	$= 500 - ((3900 + 100) / 2) + 200$
.... Calculated maxfield strength:	1700	$= ((3900 - 100) / 2) - 200$
.... Field strength reported:	201	$= \text{int}(\text{abs}(-1300 / 1700) * 255)$

Valid range is from 0 to 4095. “trigger_adc_min” must be lower than “trigger_adc_max”.

Note: These settings are only available for Controller mode.

Example:

```
"trigger_adc_min" : 200,  
"trigger_adc_max" : 3895,  
"trigger_adc_zero" : 0
```

■ “sensor_enable”

A hex-string mask of which optical sensors are enabled. The default value is “0xFFFFFFFF”.

Example:

```
"sensor_enable" : "0xFFFFFFFF"
```

■ “imu_accel_scale”

An integer value to set the IMU’s acceleration scale. Setting this field will override the values configured as per the device’s mode.

Valid values:

- | | |
|-----------------------------|--|
| <input type="checkbox"/> 2 | +/- 2 G Acceleration Scale |
| <input type="checkbox"/> 4 | +/- 4 G Acceleration Scale (HMD mode default) |
| <input type="checkbox"/> 8 | +/- 8 G Acceleration Scale (Controller mode default) |
| <input type="checkbox"/> 16 | +/- 16 G Acceleration Scale |

Notes: This value is polled by the SteamVR host. Reported IMU measurements in lighthouse_console will be automatically scaled based on this setting. Please refer to the MPU-6500 Datasheet & Register Map (Register 28 – Accelerometer Configuration, ACCEL_FS_SEL) for further information regarding this field.

Example:

```
"imu_accel_scale" : 8
```

■ “imu_gyro_scale”

An integer value to set the IMU’s gyroscope scale. Setting this field will override the values configured as per the device’s mode.

Valid values:

- | | |
|-------------------------------|---|
| <input type="checkbox"/> 250 | 250 Degrees per second |
| <input type="checkbox"/> 500 | 500 Degrees per second (HMD mode default) |
| <input type="checkbox"/> 1000 | 1000 Degrees per second |
| <input type="checkbox"/> 2000 | 2000 Degrees per second (Controller mode default) |

Notes: This value is polled by the SteamVR host. Reported IMU measurements in lighthouse_console will be automatically scaled based on this setting. Please refer to the MPU-6500 Datasheet & Register Map (Register 27 – Gyroscope Configuration, GYRO_FS_SEL) for further information regarding this field.

Example:

```
"imu_gyro_scale" : 2000
```

■ “imu_rate”

An integer value to set the IMU’s reporting rate. Setting this field will override the values configured as per the device’s mode.

Valid values:

- | | |
|-------------------------------|--|
| <input type="checkbox"/> 250 | 250 Hz Report Rate (Controller mode default) |
| <input type="checkbox"/> 500 | 500 Hz Report Rate |
| <input type="checkbox"/> 1000 | 1000 Hz Report Rate (HMD mode default) |

Note: Please refer to the MPU-6500 Datasheet & Register Map (Register 25 – Sample Rate Divider, SMPLRT_DIV) for further information regarding this field.

Example:

```
"imu_gyro_scale" : 250
```

■ “button_[#]”

*** REPLACED IN 2017-07-10 RELEASE - SEE ‘BUTTON REMAPPING’ ***

A string value to assign a physical GPIO pin to a logical button name. Any of the valid values can be uniquely assigned to the five button names. Assigning a single physical pin to multiple logical buttons is possible, but may lead to unexpected button pressed reports.

Valid names (and their default values):

- | | |
|-------------------------------------|-----------------|
| <input type="checkbox"/> "button_0" | "pin_user_pa17" |
| <input type="checkbox"/> "button_1" | "pin_user_pa19" |
| <input type="checkbox"/> "button_2" | "pin_user_pb10" |
| <input type="checkbox"/> "button_3" | "pin_user_pb11" |
| <input type="checkbox"/> "button_4" | "pin_user_pa2" |

Valid values:

- ☐ "pin_user_pa17"
- ☐ "pin_user_pa19"
- ☐ "pin_user_pb10"
- ☐ "pin_user_pb11"
- ☐ "pin_user_pa2"

Note: Only “pin_user_pa17” and “pin_user_pb11” are available in HMD mode.

Example that swaps two buttons:

```
"button_0" : "pin_user_pa19",  
"button_1" : "pin_user_pa17"
```

■ “steamvr_button_[name]”

*** REPLACED IN 2017-07-10 RELEASE - SEE ‘BUTTON REMAPPING’ ***

A string value to assign a logical button name to a SteamVR event name. Any of the event names can be assigned to any of the valid button names. Assigning a single button name to multiple events is possible, causing multiple events to be reported for a single button press.

Valid names (and their default values):

<input type="checkbox"/> "steamvr_button_menu"	"button_0"
<input type="checkbox"/> "steamvr_button_grip"	"button_1"
<input type="checkbox"/> "steamvr_button_trigger"	"button_2"
<input type="checkbox"/> "steamvr_button_trackpad"	"button_3"
<input type="checkbox"/> "steamvr_button_system"	"button_4"

Valid values:

- ☐ "button_0"
- ☐ "button_1"
- ☐ "button_2"
- ☐ "button_3"
- ☐ "button_4"

Note: Only "steamvr_button_system" and "steamvr_button_menu" are available in HMD mode.

Example that swaps the button assignments for two events:

```
"steamvr_button_menu" : "button_2",  
"steamvr_button_trigger" : "button_0"
```

■ "button_pair" and "button_pair_secondary"

***** REPLACED IN 2017-07-10 RELEASE - SEE 'BUTTON REMAPPING' *****

The name of the buttons to initiate pairing of a tracked device radio and a host-connected radio dongle. If the trackable device does not include a secondary button, an empty string can be used to start pairing with a single press and hold.

Valid names (and their default values):

<input type="checkbox"/> "button_pair"	"pin_user_pa2"
<input type="checkbox"/> "button_pair_secondary"	"pin_user_pa17"

Valid values:

- ☐ "pin_user_pa17"
- ☐ "pin_user_pa19"
- ☐ "pin_user_pb10"
- ☐ "pin_user_pb11"
- ☐ "pin_user_pa2"

Note: These settings are only available for Controller mode.

Example that changes the two buttons used to initiate wireless pairing:

```
"button_pair" : "pin_user_pb10",  
"button_pair_secondary" : "pin_user_pb11"
```

Example configuration that uses a single button to initiate wireless pairing:


```
"button_pair" : "pin_user_pb10",  
"button_pair_secondary" : ""
```

■ Button Remapping *(new with 2017-07-10 firmware release)*

Several hardware GPIO lines are dedicated to receiving active-low button input sources. These button inputs are assigned to firmware functions for controlling higher-level tasks, with the assignments being adjustable through the JSON file. Multiple functions can be installed onto a single buttons.

Valid button GPIO lines for the HDK hardware: PA2, PA17, PA19, PB10, PB11

Default HDK function assignments:

Firmware Function	Function Reference	Default Button Source
SteamVR Grip	button_function_0	PA19
SteamVR Trigger	button_function_1	PB10
SteamVR Trackpad	button_function_2	PB11
SteamVR System	button_function_3	PA2
SteamVR Menu	button_function_4	PA17
Wireless Pair	button_function_5	PA2
Wireless Pair - Secondary	button_function_6	PA17

To reassign a button onto a different function, add the corresponding Function Reference to the JSON as the key and the Button Source as the value (both as strings).

Example switching the Trigger button with the Trackpad button:

```
"button_function_1" : "PB11",  
"button_function_2" : "PB10"
```

Example configuration that uses a single button to initiate wireless pairing:

```
"button_function_5" : "PA2",  
"button_function_6" : ""
```