The JSON File - Firmware Configuration

SteamVR™ Tracking

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 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,
  "sensor_enable" : "0xFFFFFFFF",
  "sensor_squelch" : "0x00000000"
}
```

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 n81. 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 (default value)
  - ❏ "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**"

  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 (default)
  - ❏ false

  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
```

- "**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 (default)
- ❏ false

Example:

```
"radio" : true
```

- "**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
- ❏ false (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 (default)
- ❏ false

Note: The VRC task handles transmitting button values even when the device is configured for HMD mode; HMDs often will to define at least one button.

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 also require that the device is set to Controller mode.

Example:

```
"trackpad" : 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 also requires that the device is set to Controller mode.

Example:

```
"trigger" : false
```

- "**trigger_adc_min**" & "**trigger_adc_max**"

Two integer values defining the effective range of the trigger's Hall effect sensor measurement values.

Raw Hall effect sensor measurements are a 10-bit unsigned integer, where 0x00 indicates a strong negative field, 0x3FF 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. The trigger_adc_min & _max 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	
trigger_adc_max:	3900	
Hall reading:	500	
…. Reading, Zero-centered:	-1500	= 500 - (3900 + 100) / 2
…. Calculated maxfield strength:	1900	= (3900 - 100) / 2
…. Field strength reported:	201	= int(abs(-1500 / 1900) * 255)

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

Example:

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

- "**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:

  - ❏ 2 +/- 2 G Acceleration Scale
  - ❏ 4 +/- 4 G Acceleration Scale (HMD mode default)
  - ❏ 8 +/- 8 G Acceleration Scale (Controller mode default)
  - ❏ 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:

  - ❏ 250 250 Degrees per second
  - ❏ 500 500 Degrees per second (HMD mode default)
  - ❏ 1000 1000 Degrees per second
  - ❏ 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:

  - ❏ 250 250 Hz Report Rate (Controller mode default)
  - ❏ 500 500 Hz Report Rate
  - ❏ 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_[#]**”

  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):

  - ❏ "button_0" “pin_user_pa17”
  - ❏ "button_1" “pin_user_pa19”
  - ❏ "button_2" “pin_user_pb10”
  - ❏ "button_3" “pin_user_pb11”
  - ❏ "button_4" “pin_user_pa2”

  Valid values:

  - ❏ “pin_user_pa17”
  - ❏ “pin_user_pa19”
  - ❏ “pin_user_pb10”
  - ❏ “pin_user_pb11”
  - ❏ “pin_user_pa2”

  Note:

  Example that swaps two buttons:

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

- “**steamvr_button_[name]**”

  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):

  - ❏ "steamvr_button_menu" “button_0”
  - ❏ "steamvr_button_grip" “button_1”
  - ❏ "steamvr_button_trigger" “button_2”
  - ❏ "steamvr_button_trackpad" “button_3”
  - ❏ "steamvr_button_system" “button_4”

  Valid values:

  - ❏ "button_0"
  - ❏ "button_1"
  - ❏ "button_2"
  - ❏ "button_3"
  - ❏ "button_4"

  Note:

  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**”

  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):

  - "button_pair" “pin_user_pa2”
  - "button_pair_secondary" “pin_user_pa17”

  Valid values:

  - “pin_user_pa17”
  - “pin_user_pa19”
  - “pin_user_pb10”
  - “pin_user_pb11”
  - “pin_user_pa2”

  Note:

  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" : ""
  ```