SH-2 API User's Guide

Generated by Doxygen 1.8.11

# **Contents**

1	Hillo	rest SH-2 sensor hub driver for MCU Applications	1
2	Data	Structure Index	7
	2.1	Data Structures	7
3	File	Index	9
	3.1	File List	9
4	Data	Structure Documentation	11
	4.1	sh2_Accelerometer Struct Reference	11
		4.1.1 Detailed Description	11
	4.2	sh2_AmbientLight Struct Reference	11
		4.2.1 Detailed Description	12
	4.3	sh2_AsyncEvent Struct Reference	12
		4.3.1 Detailed Description	12
	4.4	sh2_CircleDetector Struct Reference	12
		4.4.1 Detailed Description	13
	4.5	sh2_Counts Struct Reference	13
		4.5.1 Detailed Description	13
	4.6	sh2_ErrorRecord Struct Reference	13
		4.6.1 Detailed Description	14
	4.7	sh2_FlipDetector Struct Reference	14
		4.7.1 Detailed Description	14
	4.8	sh2_GyroIntegratedRV Struct Reference	15
		4.8.1 Detailed Description	15

iv CONTENTS

4.9	sh2_Gyroscope Struct Reference	15
	4.9.1 Detailed Description	16
4.10	sh2_GyroscopeUncalibrated Struct Reference	16
	4.10.1 Detailed Description	16
4.11	sh2_HeartRateMonitor Struct Reference	16
	4.11.1 Detailed Description	17
	4.11.2 Field Documentation	17
	4.11.2.1 heartRate	17
4.12	sh2_Humidity Struct Reference	17
	4.12.1 Detailed Description	17
4.13	sh2_MagneticField Struct Reference	17
	4.13.1 Detailed Description	18
4.14	sh2_MagneticFieldUncalibrated Struct Reference	18
	4.14.1 Detailed Description	18
4.15	sh2_PersonalActivityClassifier Struct Reference	19
4.16	sh2_PickupDetector Struct Reference	19
	4.16.1 Field Documentation	19
	4.16.1.1 pickup	19
4.17	sh2_PocketDetector Struct Reference	19
	4.17.1 Detailed Description	19
4.18	sh2_Pressure Struct Reference	20
	4.18.1 Detailed Description	20
4.19	sh2_ProductId_s Struct Reference	20
	4.19.1 Detailed Description	20
4.20	sh2_ProductIds_s Struct Reference	21
4.21	sh2_Proximity Struct Reference	21
	4.21.1 Detailed Description	21
4.22	sh2_Quaternion Struct Reference	21
	4.22.1 Detailed Description	22
4.23	sh2_RawAccelerometer Struct Reference	22

CONTENTS

	4.23.1 Detailed Description	22
4.24	sh2_RawGyroscope Struct Reference	22
	4.24.1 Detailed Description	23
4.25	sh2_RawMagnetometer Struct Reference	23
	4.25.1 Detailed Description	23
4.26	sh2_Reserved Struct Reference	24
	4.26.1 Detailed Description	24
4.27	sh2_RotationVector Struct Reference	24
	4.27.1 Detailed Description	24
4.28	sh2_RotationVectorWAcc Struct Reference	25
	4.28.1 Detailed Description	25
4.29	sh2_SensorConfig Struct Reference	25
	4.29.1 Detailed Description	26
4.30	sh2_SensorEvent Struct Reference	26
	4.30.1 Detailed Description	26
4.31	sh2_SensorMetadata Struct Reference	26
	4.31.1 Detailed Description	27
4.32	sh2_SensorValue Struct Reference	28
	4.32.1 Field Documentation	29
	4.32.1.1 sensorld	29
	4.32.1.2 sequence	29
	4.32.1.3 timestamp	29
	4.32.1.4 un	29
4.33	sh2_ShakeDetector Struct Reference	29
4.34	sh2_SigMotion Struct Reference	29
	4.34.1 Detailed Description	30
4.35	sh2_SleepDetector Struct Reference	30
	4.35.1 Detailed Description	30
4.36	sh2_StabilityClassifier Struct Reference	30
4.37	sh2_StabilityDetector Struct Reference	31
	4.37.1 Field Documentation	31
	4.37.1.1 stability	31
4.38	sh2_StepCounter Struct Reference	31
	4.38.1 Detailed Description	31
4.39	sh2_StepDetector Struct Reference	31
	4.39.1 Detailed Description	32
4.40	sh2_TapDetector Struct Reference	32
4.41	sh2_Temperature Struct Reference	32
	4.41.1 Detailed Description	32
4.42	sh2_TiltDetector Struct Reference	33
	4.42.1 Detailed Description	33

vi

5	File	Docume	entation		35
	5.1	sh2.h F	File Refere	nce	35
		5.1.1	Detailed	Description	39
		5.1.2	Typedef [	Documentation	40
			5.1.2.1	sh2_AsyncEvent_t	40
			5.1.2.2	sh2_Counts_t	40
			5.1.2.3	sh2_ErrorRecord_t	40
			5.1.2.4	sh2_ProductId_t	40
			5.1.2.5	sh2_Quaternion_t	40
			5.1.2.6	sh2_SensorConfig_t	40
			5.1.2.7	sh2_SensorEvent_t	40
			5.1.2.8	sh2_SensorMetadata_t	41
			5.1.2.9	sh2_TareAxis_t	41
			5.1.2.10	sh2_TareBasis_t	41
		5.1.3	Enumera	tion Type Documentation	41
			5.1.3.1	sh2_CalStatus_t	41
			5.1.3.2	sh2_OscType_t	41
			5.1.3.3	sh2_SensorId_e	41
			5.1.3.4	sh2_TareAxis	41
			5.1.3.5	sh2_TareBasis	42
		5.1.4	Function	Documentation	42
			5.1.4.1	sh2_clearCounts(sh2_SensorId_t sensorId)	42
			5.1.4.2	sh2_clearDcdAndReset(void)	42
			5.1.4.3	sh2_clearTare(void)	42
			5.1.4.4	sh2_finishCal(sh2_CalStatus_t *status)	43
			5.1.4.5	sh2_flush(sh2_SensorId_t sensorId)	43
			5.1.4.6	sh2_getCalConfig(uint8_t *pSensors)	43
			5.1.4.7	sh2_getCounts(sh2_SensorId_t sensorId, sh2_Counts_t *pCounts)	43
			5.1.4.8	sh2_getErrors(uint8_t severity, sh2_ErrorRecord_t *pErrors, uint16_t *numErrors)	44
			5.1.4.9	sh2_getFrs(uint16_t recordId, uint32_t *pData, uint16_t *words)	44

CONTENTS vii

		5.1.4.10	$sh2\_getMetadata(sh2\_SensorId\_t\ sensorId,\ sh2\_SensorMetadata\_t\ *pData)  .  .$	44
		5.1.4.11	sh2_getOscType(sh2_OscType_t *pOscType)	45
		5.1.4.12	sh2_getProdIds(sh2_ProductIds_t *prodIds)	45
		5.1.4.13	$sh2\_getSensorConfig(sh2\_SensorId\_t\ sensorId,\ sh2\_SensorConfig\_t\ *config)  .$	45
		5.1.4.14	sh2_initialize(sh2_EventCallback_t *eventCallback, void *resetCookie)	45
		5.1.4.15	sh2_persistTare(void)	46
		5.1.4.16	sh2_reinitialize(void)	46
		5.1.4.17	sh2_saveDcdNow(void)	46
		5.1.4.18	sh2_setCalConfig(uint8_t sensors)	46
		5.1.4.19	sh2_setDcdAutoSave(bool enabled)	47
		5.1.4.20	sh2_setFrs(uint16_t recordId, uint32_t *pData, uint16_t words)	47
		5.1.4.21	sh2_setReorientation(sh2_Quaternion_t *orientation)	47
		5.1.4.22	sh2_setSensorCallback(sh2_SensorCallback_t *callback, void *cookie)	47
		5.1.4.23	$sh2\_setSensorConfig(sh2\_SensorId\_t\ sensorId,\ const\ sh2\_SensorConfig\_t\ *p \leftarrow Config)\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .\$	48
		5.1.4.24	sh2_setTareNow(uint8_t axes, sh2_TareBasis_t basis)	48
		5.1.4.25	sh2_startCal(uint32_t interval_us)	48
5.2	sh2_ei	r.h File Re	eference	49
	5.2.1	Detailed	Description	49
	5.2.2	Macro De	efinition Documentation	49
		5.2.2.1	SH2_ERR	49
		5.2.2.2	SH2_ERR_BAD_PARAM	49
		5.2.2.3	SH2_ERR_HUB	49
		5.2.2.4	SH2_ERR_IO	49
		5.2.2.5	SH2_ERR_OP_IN_PROGRESS	50
		5.2.2.6	SH2_ERR_TIMEOUT	50
		5.2.2.7	SH2_OK	50
5.3	sh2_ha	al.h File Re	eference	50
	5.3.1	Detailed	Description	50
5.4	sh2_S	ensorValue	e.h File Reference	51
	5.4.1	Detailed	Description	54

viii CONTENTS

5.4.2	Macro De	efinition Documentation	54
	5.4.2.1	PAC_UNKNOWN	54
	5.4.2.2	PICKUP_LEVEL_TO_NOT_LEVEL	54
	5.4.2.3	SHAKE_X	54
	5.4.2.4	STABILITY_CLASSIFIER_UNKNOWN	55
	5.4.2.5	STABILITY_ENTERED	55
	5.4.2.6	TAPDET_X	55
5.4.3	Typedef I	Documentation	55
	5.4.3.1	sh2_Accelerometer_t	55
	5.4.3.2	sh2_AmbientLight_t	55
	5.4.3.3	sh2_CircleDetector_t	55
	5.4.3.4	sh2_FlipDetector_t	55
	5.4.3.5	sh2_GyroIntegratedRV_t	56
	5.4.3.6	sh2_Gyroscope_t	56
	5.4.3.7	sh2_GyroscopeUncalibrated_t	56
	5.4.3.8	sh2_HeartRateMonitor_t	56
	5.4.3.9	sh2_Humidity_t	56
	5.4.3.10	sh2_MagneticField_t	56
	5.4.3.11	sh2_MagneticFieldUncalibrated_t	56
	5.4.3.12	sh2_PocketDetector_t	56
	5.4.3.13	sh2_Pressure_t	57
	5.4.3.14	sh2_Proximity_t	57
	5.4.3.15	sh2_RawAccelerometer_t	57
	5.4.3.16	sh2_RawGyroscope_t	57
	5.4.3.17	sh2_RawMagnetometer_t	57
	5.4.3.18	sh2_Reserved_t	57
	5.4.3.19	sh2_RotationVector_t	57
	5.4.3.20	sh2_RotationVectorWAcc_t	57
	5.4.3.21	sh2_SigMotion_t	58
	5.4.3.22	sh2_SleepDetector_t	58
	5.4.3.23	sh2_StepCounter_t	58
	5.4.3.24	sh2_StepDetector_t	58
	5.4.3.25	sh2_Temperature_t	58
	5.4.3.26	sh2_TiltDetector_t	58
			59

Index

# **Chapter 1**

# Hillcrest SH-2 sensor hub driver for MCU Applications

#### Introduction

Hillcrest Labs, Inc. produces a line of sensor hubs that interoperate with a host processor using an interface called SH-2. (Sensor Hub 2.) This interface is comprised of a proprietary protocol, SHTP (Sensor Hub Transport Protocol), and a set of features that are based on that protocol. Some products that implement the SH-2 interface include the BNO080, BNO085 and FSP200.

In order to facilitate integration of SH-2 devices into other products, Hillcrest provides an API and driver that manage the SHTP (Sensor Hub Transport Protocol) interface and delivers application-level functionality. This document describes how to use the SH-2 API and integrate it into new systems.

#### SH-2 API

The SH-2 API makes the sensor hub's features available to an application. This section describes how the API works, beginning with a list of the API functions and brief descriptions of each. Following that, we describe a set of conventions that the API uses.

#### **API Functions**

The following functions comprise the SH-2 API.

#### Initialization

• sh2\_initialize()

This function initializes the sensor hub. It should be called before any other API functions to ensure the device starts from a known state. When called, the sensor hub is reset. Also, the underlying SHTP layer is configured to support SH-2 operations for the device.

An event handler callback can be registered at initialization time. This callback will be used to notify the application when certain events occur. For example, the reset complete event will be passed to the callback when the device is in a state where sensor configuration can start.

#### **Configuring Sensors**

- sh2 setSensorConfig()
- sh2\_getSensorConfig()
- sh2 getMetadata()

The sh2\_setSensorConfig() function is used to enable and disable sensors. It sets the desired event rate and other attributes that control data production.

The sh2\_getSensorConfig() function reads back the actual configuration of a sensor. The actual configuration can differ from the requested configuration. For example, if a particular sensor only supports a limited set of data rates, the value read will reflect the actual rate the sensor uses.

The sh2\_getMetadata() function reads out metadata record associated with a particular sensor. The metadata includes information such as the resolution and scale of the sensor data.

#### **Reading Sensors**

• sh2\_setSensorCallback()

If a sensor is enabled, it will produce periodic events to report its measurements. These are delivered to the application code using a callback mechanism. The sh2\_setSensorCallback() function registers the application's callback function. Along with the function, an opaque data value called the cookie, is registered. Afterward, each sensor event will result in one call to the callback with the cookie as one parameter and an sh2\_SensorEvent\_t pointer as the other.

#### Managing the sensor hub

- sh2\_getProdIds()
- sh2\_getFrs()
- sh2 setFrs()
- sh2\_getErrors()
- sh2\_getCounts()
- sh2 clearCounts()
- sh2\_setTareNow()
- sh2\_clearTare()
- sh2\_persistTare()
- sh2\_setReorientation()
- sh2\_reinitialize()
- sh2 saveDcdNow()
- sh2\_getOscType()
- sh2\_setCalConfig()
- sh2 setDcdAutoSave()
- sh2\_flush()

A variety of utility functions provide control over many facets of the SensorHub's operation. Some of these functions read and write FRS records (Non-volatile data, usually stored in Flash memory on the device.) Others provide access to version information, internal counters, etc. The tare operations modify the reference frame used for reporting rotation vectors.

See the reference section for details on each of these API calls.

#### **API Conventions**

The SH-2 API uses a set of conventions for function names, returns values and other aspects of its operation.

#### **Naming Conventions**

All public functions in the SH-2 API have the prefix "sh2\_". This helps distiguish them from other application functions or other APIs.

After the sh2\_ prefix, the function name starts with a verb in lower case. (This is often "get" or "set".) Additional words to describe the function each begin with upper case. So, for example, sh2\_setSensorConfig() is the function to set the configuration of a particular sensor.

Enumerations and macros (#defines) are named with the prefix SH2\_.

Data types that are exposed through the API are named with the prefix sh2\_ and end with the suffix \_t. The word or words between prefix and suffix are capitalized. So, for example, the sensor metadata record type is sh2\_Sensor  $\leftarrow$  Metadata\_t.

#### **Blocking calls**

Most of the SH-2 SPI functions are blocking. That is, they only return after they have performed their function.

#### Return values

All SH-2 API functions return a status code. The values are listed in sh2\_err.h. In general a successful API operation will return SH2\_OK, which is zero. If the operation failed for any reason, some other code will be returned. The error return values are all less than zero.

#### Memory allocation

There is no dynamic memory allocation performed in the SH-2 library.

Generally, API functions that must return blocks of data require the caller to pass an address to a structure that will receive the results.

#### SH-2 Hardware Adaptation Layer

The SH-2 HAL is an interface that adapts the SH-2 API to a particular hardware platform. Different platforms will require different HAL implementations. So this software component must be developed by the system designer.

The HAL layer provides low-level communications and control functions needed by the driver and DFU (Download Firmware Update) modules. Further details are described below for each HAL API function.

Since these functions must be implemented by the system developer, the descriptions that follow are requirements that must be met in order for the SH-2 driver to work properly.

An example SH-2 HAL is provided for the BNO080 Developer's Kit for reference. The example is based on the STM32F411 Nucleo eval board running FreeRTOS.

#### Initialization

The SH-2 HAL API doesn't specify a system initialization function, but most systems will require one. Any low level interfaces, e.g. GPIO, I2C, SPI, etc, used for control of the SH-2 device should be initialized before the sh2\_← initialize() function is used.

#### **Device Reset**

· sh2 hal reset()

This function should perform a chip level reset on the sensor hub. It takes a flag, dfuMode, that indicates whether the chip should be brought up in application mode or DFU mode. The reset process involves asserting the RSTN signal on the sensor hub, setting the BOOTN signal according to the dfuMode flag, then deasserting RSTN. Timing requirements for this process can be found in the SH-2 Reference Manual.

The HAL should store the dfuMode flag for future reference. The operation of some other HAL functions will depend on the state of dfuMode.

The reset function also takes a callback function and cookie. These should be stored for use later. When messages are received from the SH-2 device, they must be delivered to the driver by invoking the callback.

#### Communications

- sh2\_hal\_tx()
- sh2 hal rx()

sh2\_hal\_tx() will be called by the driver (or DFU code) when it needs send a message to the SH-2 device. This function should initiate the transmission but can return to the caller before the operation is complete.

For I2C and serial communications, the sh2\_hal\_tx() implementation is fairly straightforward: simply transmit the given data. For SPI communications its a bit more complex, especially considering the timing requirements for DFU mode.

In application mode with SPI, this function should initiate a write transaction by asserting WAKEN. The write transaction should continue, then, when the system responds to INTN being asserted by the sensor hub. (See Interrupt Service for further detail). If the sh2\_hal\_tx function does not block during this time, it should copy the data being transmitted.

For DFU mode, transmission can begin immediately but a different set of configuration and timing parameters need to be used with the SPI bus. CPOL and CPHA should be 0. The SPI clock can be at most 1MHz. Furthermore the timing of the operation needs to be carefully controlled. After asserting select, wait at least 20uS before transmitting the first byte. Then, after each byte, delay 28uS before sending the next byte. Finally, after writing the last byte, deassert select and wait 5ms before starting the next SPI operation. If this timing is not met, the DFU process can fail.

sh2\_hal\_rx() is called only in DFU mode. For an I2C bus, this function should implement a simple i2c read of the device. For SPI devices, it should perform a SPI operation sending NULL and placing the read data in the given buffer.

#### Interrupt Service

In application mode (as opposed to dfu mode) the HAL needs to respond to interrupts from the SH-2 device. The interrupt service routine needs to capture timestamps, initiate read operations and, for SPI devices, perform write operations. Any data read from the SH-2 device as a result of an interrupt must be delivered to the driver via the callback described above.

With the HAL autonomously performing read operations, it needs to know how many bytes of data to transfer. This can be determined by peaking into the read data since the first two bytes of each SHTP transfer contain a maximum read length.

For I2C, then, the read length is determined as follows:

- Initially, the host should read 2 bytes from the device. These will contain the first two bytes of the SHTP header, containing the size of the SHTP payload to be transferred. (Let's call this value rxRemaining.)
- If 0 < rxRemaining <= max transfer length, read rxRemaining bytes. Afterward, set rxRemaining to 0.
- If rxRemaining > max transfer length, read max-transfer-length bytes. Afterward, set rxRemaining to rx
   Remaining max-transfer-length + 4. (The additional four bytes represent a new SHTP header that will be generated.)

For SPI, the read length is determined in a similar manner but any SPI operation performed should transfer enough bytes to accommodate the transmit buffer, if non-empty.

#### **Thread Control**

- · sh2 hal block()
- sh2 hal unblock()

Some HAL implementations will use an operating system such as FreeRTOS while others will not.

If an OS is used, there are points in the SH-2 driver where the caller of an operation needs to block until the operation completes. The SH-2 library calls sh2\_hal\_block and sh2\_hal\_unblock to implement the blocking in a thread-friendly manner. (i.e., without busy waiting.)

The HAL implementation, in this case, should implement these using a binary semaphore.

If no OS is used and the HAL is implemented with blocking calls, the sh2\_hal\_block and sh2\_hal\_unblock calls can be empty functions that return immediately.

See the HAL implementations in the BNO080 Nucleo Demo code for working examples of this interface.

6	Hillcrest SH-2 sensor hub driver for MCU Applications

# Chapter 2

# **Data Structure Index**

# 2.1 Data Structures

Here are the data structures with brief descriptions:

sh2_Accelerometer	
Accelerometer	-11
sh2_AmbientLight	
Ambient Light	11
sh2_AsyncEvent	
Asynchronous Event	12
sh2_CircleDetector	
CircleDetector	12
sh2_Counts	
SensorHub Counter Record	13
sh2_ErrorRecord	
SensorHub Error Record	13
sh2_FlipDetector	
FlipDetector	14
sh2_GyroIntegratedRV	
HeartRateMonitor	15
sh2_Gyroscope	4.5
Gyroscope	15
sh2_GyroscopeUncalibrated	4.0
Uncalibrated gyroscope	16
sh2_HeartRateMonitor  HeartRateMonitor	16
sh2 Humidity	10
Humidity	17
sh2_MagneticField	17
Magnetic field	17
sh2 MagneticFieldUncalibrated	
Uncalibrated magnetic field	18
sh2 PersonalActivityClassifier	19
sh2 PickupDetector	19
sh2 PocketDetector	
PocketDetector	19
sh2 Pressure	
Atmospheric Pressure	20
sh2_ProductId_s	

8 Data Structure Index

sh2_Productlds_s	21
sh2_Proximity	
Proximity	21
sh2_Quaternion	
Quaternion (double precision floating point representation.)	21
sh2_RawAccelerometer	
Raw Accelerometer	22
sh2_RawGyroscope	
Raw gyroscope	22
sh2_RawMagnetometer	
Raw Magnetometer	23
sh2_Reserved	
Reserved	24
sh2_RotationVector	
Rotation Vector	24
sh2_RotationVectorWAcc	
Rotation Vector with Accuracy	25
sh2_SensorConfig	
Sensor Configuration settings	25
sh2_SensorEvent	
Sensor Event	26
sh2_SensorMetadata	
Sensor Metadata Record	26
sh2_SensorValue	28
sh2_ShakeDetector	29
sh2_SigMotion	
SigMotion	29
sh2_SleepDetector	
SleepDetector	30
sh2_StabilityClassifier	30
sh2_StabilityDetector	31
sh2_StepCounter	
StepCounter	31
sh2_StepDetector	
StepDetector	31
sh2_TapDetector	32
sh2_Temperature	
Temperature	32
sh2_TiltDetector	
TiltDetector	33

# **Chapter 3**

# File Index

# 3.1 File List

Here is a list of all documented files with brief descriptions:

sn2.n	
API Definition for Hillcrest SH-2 Sensor Hub	35
sh2_err.h	
Type definitions for Hillcrest SH-2 API	49
sh2_hal.h	
Hardware Adaptation Layer API for SensorHub-2 (and BNO080)	50
sh2_SensorValue.h	
Support for converting sensor events (messages) into natural data structures	51

10 File Index

# **Chapter 4**

# **Data Structure Documentation**

# 4.1 sh2\_Accelerometer Struct Reference

#### Accelerometer.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

- float x
- float y
- float z

# 4.1.1 Detailed Description

Accelerometer.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.2 sh2\_AmbientLight Struct Reference

#### Ambient Light.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

· float value

Ambient Light. [lux].

#### 4.2.1 Detailed Description

Ambient Light.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.3 sh2\_AsyncEvent Struct Reference

Asynchronous Event.

```
#include <sh2.h>
```

#### **Data Fields**

- uint32\_t eventId
- uint16\_t frsType

#### 4.3.1 Detailed Description

Asynchronous Event.

Represents reset events and other non-sensor events received from SH-2 sensor hub.

The documentation for this struct was generated from the following file:

• sh2.h

# 4.4 sh2\_CircleDetector Struct Reference

#### circleDetector

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

• uint16\_t circle

#### 4.4.1 Detailed Description

circleDetector

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.5 sh2\_Counts Struct Reference

SensorHub Counter Record.

```
#include <sh2.h>
```

#### **Data Fields**

uint32\_t offered

[events]

· uint32\_t accepted

[events]

• uint32\_t on

[events]

• uint32\_t attempted

[events]

#### 4.5.1 Detailed Description

SensorHub Counter Record.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2.h

# 4.6 sh2\_ErrorRecord Struct Reference

SensorHub Error Record.

#include <sh2.h>

#### **Data Fields**

• uint8\_t severity

Error severity, 0: most severe.

• uint8\_t sequence

Sequence number (by severity)

• uint8\_t source

1-MotionEngine, 2-MotionHub, 3-SensorHub, 4-Chip

• uint8\_t error

See SH-2 Reference Manual.

• uint8\_t module

See SH-2 Reference Manual.

• uint8\_t code

See SH-2 Reference Manual.

#### 4.6.1 Detailed Description

SensorHub Error Record.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2.h

# 4.7 sh2\_FlipDetector Struct Reference

flipDetector

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

• uint16\_t flip

#### 4.7.1 Detailed Description

flipDetector

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.8 sh2\_GyroIntegratedRV Struct Reference

#### heartRateMonitor

#include <sh2\_SensorValue.h>

#### **Data Fields**

float i

Quaternion component i.

float j

Quaternion component j.

float k

Quaternion component k.

float real

Quaternion component real.

float angVelX

Angular velocity about x [rad/s].

float angVelY

Angular velocity about y [rad/s].

float angVelZ

Angular velocity about z [rad/s].

#### 4.8.1 Detailed Description

heartRateMonitor

See SH-2 Reference Manual for details.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.9 sh2\_Gyroscope Struct Reference

### Gyroscope.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

- float x
- float y
- float z

#### 4.9.1 Detailed Description

Gyroscope.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.10 sh2\_GyroscopeUncalibrated Struct Reference

Uncalibrated gyroscope.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

float x

[rad/s]

float y

[rad/s]

float z

[rad/s]

float biasX

[rad/s]

float biasY

[rad/s]

float biasZ

[rad/s]

#### 4.10.1 Detailed Description

Uncalibrated gyroscope.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.11 sh2\_HeartRateMonitor Struct Reference

#### heartRateMonitor

#include <sh2\_SensorValue.h>

#### **Data Fields**

uint16\_t heartRate

#### 4.11.1 Detailed Description

heartRateMonitor

See SH-2 Reference Manual for details.

#### 4.11.2 Field Documentation

4.11.2.1 uint16\_t sh2\_HeartRateMonitor::heartRate

heart rate in beats per minute.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

### 4.12 sh2\_Humidity Struct Reference

#### Humidity.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

· float value

Relative Humidity. [percent].

#### 4.12.1 Detailed Description

Humidity.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.13 sh2\_MagneticField Struct Reference

Magnetic field.

#include <sh2\_SensorValue.h>

#### **Data Fields**

```
float x
    [uTesla]
float y
    [uTesla]
float z
    [uTesla]
```

#### 4.13.1 Detailed Description

Magnetic field.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.14 sh2\_MagneticFieldUncalibrated Struct Reference

Uncalibrated magnetic field.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

```
 float x
```

[uTesla]

float y

[uTesla]

float z

[uTesla]

float biasX

[uTesla]

float biasY

[uTesla]

float biasZ

[uTesla]

#### 4.14.1 Detailed Description

Uncalibrated magnetic field.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.15 sh2\_PersonalActivityClassifier Struct Reference

#### **Data Fields**

- uint8\_t page
- bool lastPage
- uint8\_t mostLikelyState
- uint8\_t confidence [10]

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.16 sh2\_PickupDetector Struct Reference

#### **Data Fields**

uint16\_t pickup

#### 4.16.1 Field Documentation

4.16.1.1 uint16\_t sh2\_PickupDetector::pickup

flag field with bits defined above.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.17 sh2\_PocketDetector Struct Reference

#### pocketDetector

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

uint16\_t pocket

#### 4.17.1 Detailed Description

pocketDetector

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

sh2\_SensorValue.h

# 4.18 sh2\_Pressure Struct Reference

#### Atmospheric Pressure.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

· float value

Atmospheric Pressure. [hectopascals].

#### 4.18.1 Detailed Description

Atmospheric Pressure.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.19 sh2\_ProductId\_s Struct Reference

#### Product Id value.

```
#include <sh2.h>
```

#### **Data Fields**

- uint8\_t resetCause
- uint8\_t swVersionMajor
- uint8\_t swVersionMinor
- uint32\_t swPartNumber
- uint32\_t swBuildNumber
- uint16\_t swVersionPatch
- uint8\_t reserved0
- uint8\_t reserved1

#### 4.19.1 Detailed Description

Product Id value.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

sh2.h

### 4.20 sh2\_ProductIds\_s Struct Reference

#### **Data Fields**

- sh2\_ProductId\_t entry [SH2\_MAX\_PROD\_ID\_ENTRIES]
- uint8\_t numEntries

The documentation for this struct was generated from the following file:

• sh2.h

# 4.21 sh2\_Proximity Struct Reference

#### Proximity.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

· float value

Proximity. [cm].

#### 4.21.1 Detailed Description

Proximity.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.22 sh2\_Quaternion Struct Reference

Quaternion (double precision floating point representation.)

```
#include <sh2.h>
```

#### **Data Fields**

- double **x**
- double y
- double z
- double w

#### 4.22.1 Detailed Description

Quaternion (double precision floating point representation.)

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2.h

# 4.23 sh2\_RawAccelerometer Struct Reference

#### Raw Accelerometer.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

```
int16_t x
    [ADC counts]
int16_t y
    [ADC counts]
int16_t z
    [ADC counts]
uint32_t timestamp
    [uS]
```

#### 4.23.1 Detailed Description

Raw Accelerometer.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.24 sh2\_RawGyroscope Struct Reference

#### Raw gyroscope.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

```
    int16_t x
        [ADC Counts]
    int16_t y
        [ADC Counts]
    int16_t z
        [ADC Counts]
    int16_t temperature
        [ADC Counts]
    uint32_t timestamp
        [uS]
```

#### 4.24.1 Detailed Description

Raw gyroscope.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2 SensorValue.h

# 4.25 sh2\_RawMagnetometer Struct Reference

Raw Magnetometer.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

```
    int16_t x
        [ADC Counts]
    int16_t y
        [ADC Counts]
    int16_t z
        [ADC Counts]
    uint32_t timestamp
        [uS]
```

#### 4.25.1 Detailed Description

Raw Magnetometer.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.26 sh2\_Reserved Struct Reference

#### Reserved.

#include <sh2\_SensorValue.h>

#### **Data Fields**

float tbd

Reserved.

#### 4.26.1 Detailed Description

Reserved.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.27 sh2\_RotationVector Struct Reference

#### Rotation Vector.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

float i

Quaternion component i.

float j

Quaternion component j.

float k

Quaternion component k.

float real

Quaternion component real.

#### 4.27.1 Detailed Description

Rotation Vector.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

### 4.28 sh2\_RotationVectorWAcc Struct Reference

Rotation Vector with Accuracy.

#include <sh2\_SensorValue.h>

#### **Data Fields**

float i

Quaternion component i.

float

Quaternion component j.

float k

Quaternion component k.

float real

Quaternion component, real.

· float accuracy

Accuracy estimate [radians].

#### 4.28.1 Detailed Description

Rotation Vector with Accuracy.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

#### 4.29 sh2\_SensorConfig Struct Reference

Sensor Configuration settings.

```
#include <sh2.h>
```

#### **Data Fields**

• bool changeSensitivityEnabled

Enable reports on change.

• bool changeSensitivityRelative

Change reports relative (vs absolute)

bool wakeupEnabled

Wake host on event.

bool alwaysOnEnabled

Sensor remains on in sleep state.

· uint16\_t changeSensitivity

Report-on-change threshold.

uint32\_t reportInterval\_us

[uS] Report interval

• uint32\_t batchInterval\_us

[uS] Batch interval

uint32\_t sensorSpecific

See SH-2 Reference Manual for details.

#### 4.29.1 Detailed Description

Sensor Configuration settings.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2.h

# 4.30 sh2\_SensorEvent Struct Reference

Sensor Event.

```
#include <sh2.h>
```

#### **Data Fields**

- uint64\_t timestamp\_uS
- uint8\_t reportId
- uint8\_t \* pReport
- uint8\_t len

#### 4.30.1 Detailed Description

Sensor Event.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2.h

# 4.31 sh2\_SensorMetadata Struct Reference

Sensor Metadata Record.

#include <sh2.h>

#### **Data Fields**

• uint8 t meVersion

Motion Engine Version.

uint8\_t mhVersion

Motion Hub Version.

uint8\_t shVersion

SensorHub Version.

• uint32\_t range

Same units as sensor reports.

uint32\_t resolution

Same units as sensor reports.

• uint16\_t revision

Metadata record format revision.

uint16\_t power\_mA

[mA] Fixed point 16Q10 format

• uint32\_t minPeriod\_uS

[uS]

uint32\_t fifoReserved

(Unused)

uint32\_t fifoMax

(Unused)

• uint32\_t batchBufferBytes

(Unused)

• uint16\_t qPoint1

q point for sensor values

uint16\_t qPoint2

q point for accuracy or bias fields

• uint32\_t vendorldLen

[bytes]

• char vendorld [48]

Vendor name and part number.

• uint32\_t sensorSpecificLen

[bytes]

• uint8\_t sensorSpecific [48]

See SH-2 Reference Manual.

#### 4.31.1 Detailed Description

Sensor Metadata Record.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2.h

### 4.32 sh2\_SensorValue Struct Reference

#### **Data Fields**

```
· uint8_t sensorld
· uint8 t sequence
     8-bit unsigned integer used to track reports.
· uint8 t status
     bits 7-5: reserved, 4-2: exponent delay, 1-0: Accuracy

    uint64 t timestamp

    uint32 t delay

     [uS] value is delay *2^{\land} exponent (see status)
    sh2_RawAccelerometer_t rawAccelerometer
    sh2 Accelerometer t accelerometer
    sh2_Accelerometer_t linearAcceleration
    sh2_Accelerometer_t gravity
    sh2_RawGyroscope_t rawGyroscope
    sh2 Gyroscope t gyroscope
   sh2_GyroscopeUncalibrated_t gyroscopeUncal
   sh2_RawMagnetometer_t rawMagnetometer
    sh2 MagneticField t magneticField
   sh2 MagneticFieldUncalibrated t magneticFieldUncal
   sh2_RotationVectorWAcc_t rotationVector
   sh2_RotationVector_t gameRotationVector
   sh2_RotationVectorWAcc_t geoMagRotationVector
    sh2 Pressure_t pressure
   sh2_AmbientLight_t ambientLight
    sh2_Humidity_t humidity
   sh2 Proximity t proximity
   sh2 Temperature t temperature
   sh2_Reserved_t reserved
   sh2_TapDetector_t tapDetector
    sh2 StepDetector t stepDetector
   sh2 StepCounter t stepCounter
   sh2_SigMotion_t sigMotion
    sh2_StabilityClassifier_t stabilityClassifier
    sh2 ShakeDetector t shakeDetector
   sh2_FlipDetector_t flipDetector
   sh2 PickupDetector t pickupDetector
   sh2 StabilityDetector t stabilityDetector
    sh2 PersonalActivityClassifier t personalActivityClassifier
   sh2 SleepDetector t sleepDetector
   sh2_TiltDetector_t tiltDetector
    sh2_PocketDetector_t pocketDetector
    sh2 CircleDetector t circleDetector
   sh2_HeartRateMonitor_t heartRateMonitor
   sh2_RotationVectorWAcc_t arvrStabilizedRV
   sh2_RotationVector_t arvrStabilizedGRV
   sh2_GyroIntegratedRV_t gyroIntegratedRV
 } un
```

Sensor Data.

#### 4.32.1 Field Documentation

4.32.1.1 uint8\_t sh2\_SensorValue::sensorId

Which sensor produced this event.

4.32.1.2 uint8\_t sh2\_SensorValue::sequence

8-bit unsigned integer used to track reports.

The sequence number increments once for each report sent. Gaps in the sequence numbers indicate missing or dropped reports.

4.32.1.3 uint64\_t sh2\_SensorValue::timestamp

[uS]

4.32.1.4 union { ... } sh2\_SensorValue::un

Sensor Data.

Use the structure based on the value of the sensor field.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.33 sh2\_ShakeDetector Struct Reference

**Data Fields** 

• uint16\_t shake

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.34 sh2\_SigMotion Struct Reference

SigMotion.

#include <sh2\_SensorValue.h>

# **Data Fields**

• uint16\_t motion

# 4.34.1 Detailed Description

SigMotion.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.35 sh2\_SleepDetector Struct Reference

#### sleepDetector

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

• uint8\_t sleepState

# 4.35.1 Detailed Description

sleepDetector

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.36 sh2\_StabilityClassifier Struct Reference

## **Data Fields**

• uint8\_t classification

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.37 sh2\_StabilityDetector Struct Reference

#### **Data Fields**

uint16\_t stability

#### 4.37.1 Field Documentation

4.37.1.1 uint16\_t sh2\_StabilityDetector::stability

flag field with bits defined above.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.38 sh2\_StepCounter Struct Reference

#### StepCounter.

```
#include <sh2_SensorValue.h>
```

# **Data Fields**

· uint32 t latency

Step counter latency [uS].

uint16\_t steps

Steps counted.

# 4.38.1 Detailed Description

StepCounter.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.39 sh2\_StepDetector Struct Reference

#### StepDetector.

```
#include <sh2_SensorValue.h>
```

# **Data Fields**

```
    uint32_t latency
    Step detect latency [uS].
```

# 4.39.1 Detailed Description

StepDetector.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.40 sh2\_TapDetector Struct Reference

# **Data Fields**

```
    uint8_t flags
    TapDetector.
```

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.41 sh2\_Temperature Struct Reference

# Temperature.

```
#include <sh2_SensorValue.h>
```

#### **Data Fields**

• float value

Temperature. [C].

# 4.41.1 Detailed Description

Temperature.

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# 4.42 sh2\_TiltDetector Struct Reference

# tiltDetector

#include <sh2\_SensorValue.h>

## **Data Fields**

• uint16\_t tilt

# 4.42.1 Detailed Description

tiltDetector

See the SH-2 Reference Manual for more detail.

The documentation for this struct was generated from the following file:

• sh2\_SensorValue.h

# **Chapter 5**

# **File Documentation**

# 5.1 sh2.h File Reference

API Definition for Hillcrest SH-2 Sensor Hub.

```
#include <stdint.h>
#include <stdbool.h>
```

# **Data Structures**

• struct sh2\_AsyncEvent

Asynchronous Event.

• struct sh2\_SensorEvent

Sensor Event.

• struct sh2\_ProductId\_s

Product Id value.

- struct sh2\_ProductIds\_s
- struct sh2\_SensorConfig

Sensor Configuration settings.

struct sh2\_SensorMetadata

Sensor Metadata Record.

· struct sh2 ErrorRecord

SensorHub Error Record.

struct sh2\_Counts

SensorHub Counter Record.

• struct sh2\_Quaternion

Quaternion (double precision floating point representation.)

#### **Macros**

- #define SH2 MAX PROD ID ENTRIES (5)
- #define **STATIC CALIBRATION AGM** (0x7979)
- #define NOMINAL\_CALIBRATION (0x4D4D)
- #define STATIC\_CALIBRATION\_SRA (0x8A8A)
- #define NOMINAL\_CALIBRATION\_SRA (0x4E4E)
- #define DYNAMIC\_CALIBRATION (0x1F1F)
- #define ME\_POWER\_MGMT (0xD3E2)
- #define SYSTEM ORIENTATION (0x2D3E)
- #define ACCEL\_ORIENTATION (0x2D41)
- #define SCREEN ACCEL ORIENTATION (0x2D43)
- #define GYROSCOPE\_ORIENTATION (0x2D46)
- #define MAGNETOMETER ORIENTATION (0x2D4C)
- #define ARVR STABILIZATION RV (0x3E2D)
- #define ARVR\_STABILIZATION\_GRV (0x3E2E)
- #define TAP DETECT CONFIG (0xC269)
- #define SIG\_MOTION\_DETECT\_CONFIG (0xC274)
- #define SHAKE DETECT CONFIG (0x7D7D)
- #define MAX\_FUSION\_PERIOD (0xD7D7)
- #define SERIAL\_NUMBER (0x4B4B)
- #define ES\_PRESSURE\_CAL (0x39AF)
- #define ES\_TEMPERATURE\_CAL (0x4D20)
- #define ES HUMIDITY CAL (0x1AC9)
- #define ES AMBIENT LIGHT CAL (0x39B1)
- #define ES\_PROXIMITY\_CAL (0x4DA2)
- #define ALS CAL (0xD401)
- #define PROXIMITY SENSOR CAL (0xD402)
- #define PICKUP DETECTOR CONFIG (0x1B2A)
- #define FLIP\_DETECTOR\_CONFIG (0xFC94)
- #define STABILITY\_DETECTOR\_CONFIG (0xED85)
- #define ACTIVITY\_TRACKER\_CONFIG (0xED88)
- #define SLEEP DETECTOR CONFIG (0xED87)
- #define TILT DETECTOR CONFIG (0xED89)
- #define POCKET DETECTOR CONFIG (0xEF27)
- #define CIRCLE DETECTOR CONFIG (0xEE51)
- #define USER\_RECORD (0x74B4)
- #define ME TIME SOURCE SELECT (0xD403)
- #define **UART\_FORMAT** (0xA1A1)
- #define GYRO INTEGRATED RV CONFIG (0xA1A2)
- #define FRS\_ID\_META\_RAW\_ACCELEROMETER (0xE301)
- #define FRS\_ID\_META\_ACCELEROMETER (0xE302)
- #define FRS\_ID\_META\_LINEAR\_ACCELERATION (0xE303)
- #define FRS\_ID\_META\_GRAVITY (0xE304)
- #define FRS ID META RAW GYROSCOPE (0xE305)
- #define FRS ID META GYROSCOPE CALIBRATED (0xE306)
- #define FRS ID META GYROSCOPE UNCALIBRATED (0xE307)
- #define FRS ID META RAW MAGNETOMETER (0xE308)
- #define FRS ID META MAGNETIC FIELD CALIBRATED (0xE309)
- #define FRS\_ID\_META\_MAGNETIC\_FIELD\_UNCALIBRATED (0xE30A)
- #define FRS\_ID\_META\_ROTATION\_VECTOR (0xE30B)
- #define FRS ID META GAME ROTATION VECTOR (0xE30C)
- #define FRS ID META GEOMAGNETIC ROTATION VECTOR (0xE30D)
- #define FRS\_ID\_META\_PRESSURE (0xE30E)
- #define FRS\_ID\_META\_AMBIENT\_LIGHT (0xE30F)

5.1 sh2.h File Reference 37

- #define FRS ID META HUMIDITY (0xE310)
- #define FRS\_ID\_META\_PROXIMITY (0xE311)
- #define FRS\_ID\_META\_TEMPERATURE (0xE312)
- #define FRS\_ID\_META\_TAP\_DETECTOR (0xE313)
- #define FRS ID META STEP DETECTOR (0xE314)
- #define FRS ID META STEP COUNTER (0xE315)
- #define FRS\_ID\_META\_SIGNIFICANT\_MOTION (0xE316)
- #define FRS\_ID\_META\_STABILITY\_CLASSIFIER (0xE317)
- #define FRS\_ID\_META\_SHAKE\_DETECTOR (0xE318)
- #define FRS\_ID\_META\_FLIP\_DETECTOR (0xE319)
- #define FRS ID META PICKUP DETECTOR (0xE31A)
- #define FRS\_ID\_META\_STABILITY\_DETECTOR (0xE31B)
- #define FRS ID META PERSONAL ACTIVITY CLASSIFIER (0xE31C)
- #define FRS\_ID\_META\_SLEEP\_DETECTOR (0xE31D)
- #define FRS\_ID\_META\_TILT\_DETECTOR (0xE31E)
- #define FRS\_ID\_META\_POCKET\_DETECTOR (0xE31F)
- #define FRS ID META CIRCLE DETECTOR (0xE320)
- #define FRS\_ID\_META\_HEART\_RATE\_MONITOR (0xE321)
- #define FRS ID META ARVR STABILIZED RV (0xE322)
- #define FRS\_ID\_META\_ARVR\_STABILIZED\_GRV (0xE323)
- #define FRS ID META GYRO INTEGRATED RV (0xE324)
- #define SH2\_CAL\_ACCEL (0x01)
- #define SH2 CAL GYRO (0x02)
- #define SH2\_CAL\_MAG (0x04)
- #define SH2 CAL PLANAR (0x08)

#### **Typedefs**

- typedef enum sh2\_AsyncEventId\_e sh2\_AsyncEventId\_t
- typedef struct sh2\_AsyncEvent sh2\_AsyncEvent\_t

Asynchronous Event.

- typedef void( sh2\_EventCallback\_t) (void \*cookie, sh2\_AsyncEvent\_t \*pEvent)
- typedef struct sh2 SensorEvent sh2 SensorEvent t

Sensor Event.

- typedef void( sh2\_SensorCallback\_t) (void \*cookie, sh2\_SensorEvent\_t \*pEvent)
- typedef struct sh2\_ProductId\_s sh2\_ProductId\_t

Product Id value.

- typedef struct sh2 Productlds s sh2 Productlds t
- typedef uint8\_t sh2\_SensorId\_t
- typedef struct sh2 SensorConfig sh2 SensorConfig t

Sensor Configuration settings.

typedef struct sh2\_SensorMetadata sh2\_SensorMetadata\_t

Sensor Metadata Record.

typedef struct sh2\_ErrorRecord sh2\_ErrorRecord\_t

SensorHub Error Record.

• typedef struct sh2\_Counts sh2\_Counts\_t

SensorHub Counter Record.

• typedef enum sh2 TareBasis sh2 TareBasis t

Values for specifying tare basis.

typedef enum sh2\_TareAxis sh2\_TareAxis\_t

Bit Fields for specifying tare axes.

typedef struct sh2\_Quaternion sh2\_Quaternion\_t

Quaternion (double precision floating point representation.)

#### **Enumerations**

```
    enum sh2 AsyncEventId e { SH2 RESET, SH2 FRS CHANGE }
```

• enum sh2 SensorId e {

SH2\_RAW\_ACCELEROMETER = 0x14, SH2\_ACCELEROMETER = 0x01, SH2\_LINEAR\_ACCELERAT ← ION = 0x04, SH2\_GRAVITY = 0x06,

SH2\_RAW\_GYROSCOPE = 0x15, SH2\_GYROSCOPE\_CALIBRATED = 0x02, SH2\_GYROSCOPE\_UN $\leftarrow$  CALIBRATED = 0x07, SH2\_RAW\_MAGNETOMETER = 0x16.

SH2\_MAGNETIC\_FIELD\_CALIBRATED = 0x03, SH2\_MAGNETIC\_FIELD\_UNCALIBRATED = 0x0f, S $\leftarrow$  H2 ROTATION VECTOR = 0x05, SH2 GAME ROTATION VECTOR = 0x08,

SH2\_GEOMAGNETIC\_ROTATION\_VECTOR = 0x09, SH2\_PRESSURE = 0x0a, SH2\_AMBIENT\_LIGHT = 0x0b, SH2\_HUMIDITY = 0x0c,

SH2\_PROXIMITY = 0x0d, SH2\_TEMPERATURE = 0x0e, SH2\_RESERVED = 0x17, SH2\_TAP\_DETECT $\leftarrow$  OR = 0x10,

 $\textbf{SH2\_STEP\_DETECTOR} = 0x18, \ \textbf{SH2\_STEP\_COUNTER} = 0x11, \ \textbf{SH2\_SIGNIFICANT\_MOTION} = 0x12, \\ \textbf{SH2\_STABILITY\_CLASSIFIER} = 0x13, \\$ 

SH2\_SHAKE\_DETECTOR = 0x19, SH2\_FLIP\_DETECTOR = 0x1a, SH2\_PICKUP\_DETECTOR = 0x1b, S $\leftarrow$  H2\_STABILITY\_DETECTOR = 0x1c,

SH2\_PERSONAL\_ACTIVITY\_CLASSIFIER = 0x1e, SH2\_SLEEP\_DETECTOR = 0x1f, SH2\_TILT\_DETE $\leftarrow$  CTOR = 0x20, SH2\_POCKET\_DETECTOR = 0x21,

SH2\_CIRCLE\_DETECTOR = 0x22, SH2\_HEART\_RATE\_MONITOR = 0x23, SH2\_ARVR\_STABILIZED\_ $\leftarrow$  RV = 0x28, SH2\_ARVR\_STABILIZED\_GRV = 0x29,

SH2 GYRO INTEGRATED RV = 0x2A, SH2 MAX SENSOR ID = 0x2A }

List of sensor types supported by the hub.

• enum sh2\_TareBasis { SH2\_TARE\_BASIS\_ROTATION\_VECTOR = 0, SH2\_TARE\_BASIS\_GAMING\_RO

TATION VECTOR = 1, SH2 TARE BASIS GEOMAGNETIC ROTATION VECTOR = 2 }

Values for specifying tare basis.

enum sh2\_TareAxis { SH2\_TARE\_X = 1, SH2\_TARE\_Y = 2, SH2\_TARE\_Z = 4 }

Bit Fields for specifying tare axes.

enum sh2\_OscType\_t { SH2\_OSC\_INTERNAL = 0, SH2\_OSC\_EXT\_CRYSTAL = 1, SH2\_OSC\_EXT\_CL
OCK = 2 }

Oscillator type: Internal or External.

enum sh2\_CalStatus\_t {

SH2\_CAL\_SUCCESS = 0, SH2\_CAL\_NO\_ZRO, SH2\_CAL\_NO\_STATIONARY\_DETECTION, SH2\_CA ← L ROTATION OUTSIDE SPEC,

 ${\tt SH2\_CAL\_ZRO\_OUTSIDE\_SPEC, SH2\_CAL\_GYRO\_GAIN\_OUTS} {\tt IDE\_SPEC, SH2\_CAL\_GYRO\_PERIOD\_OUTSIDE\_SPEC,} \\$ 

SH2\_CAL\_GYRO\_DROPS\_OUTSIDE\_SPEC }

Calibration result.

#### **Functions**

int sh2\_initialize (sh2\_EventCallback\_t \*eventCallback, void \*resetCookie)

Initialize a session with the SensorHub.

• int sh2 setSensorCallback (sh2 SensorCallback t \*callback, void \*cookie)

Register a function to receive sensor events.

int sh2\_getProdIds (sh2\_ProductIds\_t \*prodIds)

Get Product ID information from Sensorhub.

• int sh2 getSensorConfig (sh2 SensorId t sensorId, sh2 SensorConfig t \*config)

Get sensor configuration.

int sh2\_setSensorConfig (sh2\_SensorId\_t sensorId, const sh2\_SensorConfig\_t \*pConfig)

Set sensor configuration. (e.g enable a sensor at a particular rate.)

int sh2\_getMetadata (sh2\_SensorId\_t sensorId, sh2\_SensorMetadata\_t \*pData)

Get metadata related to a sensor.

5.1 sh2.h File Reference 39

• int sh2\_getFrs (uint16\_t recordId, uint32\_t \*pData, uint16\_t \*words)

Get an FRS record.

• int sh2\_setFrs (uint16\_t recordId, uint32\_t \*pData, uint16\_t words)

Set an FRS record.

• int sh2\_getErrors (uint8\_t severity, sh2\_ErrorRecord\_t \*pErrors, uint16\_t \*numErrors)

Get error counts.

int sh2 getCounts (sh2 Sensorld t sensorld, sh2 Counts t \*pCounts)

Read counters related to a sensor.

int sh2\_clearCounts (sh2\_SensorId\_t sensorId)

Clear counters related to a sensor.

int sh2\_setTareNow (uint8\_t axes, sh2\_TareBasis\_t basis)

Perform a tare operation on one or more axes.

int sh2\_clearTare (void)

Clears the previously applied tare operation.

int sh2\_persistTare (void)

Persist the results of last tare operation to flash.

int sh2\_setReorientation (sh2\_Quaternion\_t \*orientation)

Set the current run-time sensor reorientation. (Set to zero to clear tare.)

int sh2\_reinitialize (void)

Command the sensorhub to reset.

int sh2 saveDcdNow (void)

Save Dynamic Calibration Data to flash.

int sh2\_getOscType (sh2\_OscType\_t \*pOscType)

Get Oscillator type.

int sh2\_setCalConfig (uint8\_t sensors)

Enable/Disable dynamic calibration for certain sensors.

int sh2\_getCalConfig (uint8\_t \*pSensors)

Get dynamic calibration configuration settings.

• int sh2\_setDcdAutoSave (bool enabled)

Configure automatic saving of dynamic calibration data.

• int sh2\_flush (sh2\_SensorId\_t sensorId)

Immediately issue all buffered sensor reports from a given sensor.

int sh2\_clearDcdAndReset (void)

Command clear DCD in RAM, then reset sensor hub.

int sh2\_startCal (uint32\_t interval\_us)

Start simple self-calibration procedure.

int sh2\_finishCal (sh2\_CalStatus\_t \*status)

Finish simple self-calibration procedure.

#### 5.1.1 Detailed Description

API Definition for Hillcrest SH-2 Sensor Hub.

**Author** 

David Wheeler

Date

22 Sept 2015 The sh2 API provides functions for opening a session with the sensor hub and performing all supported operations with it. This includes enabling sensors and reading events as well as other housekeeping functions.

5.1.2 Typedef Documentation 5.1.2.1 typedef struct sh2\_AsyncEvent sh2\_AsyncEvent\_t Asynchronous Event. Represents reset events and other non-sensor events received from SH-2 sensor hub. 5.1.2.2 typedef struct sh2\_Counts sh2\_Counts\_t SensorHub Counter Record. See the SH-2 Reference Manual for more detail. 5.1.2.3 typedef struct sh2\_ErrorRecord sh2\_ErrorRecord\_t SensorHub Error Record. See the SH-2 Reference Manual for more detail. 5.1.2.4 typedef struct sh2\_ProductId\_s sh2\_ProductId\_t Product Id value. See the SH-2 Reference Manual for more detail. 5.1.2.5 typedef struct sh2\_Quaternion sh2\_Quaternion\_t Quaternion (double precision floating point representation.) See the SH-2 Reference Manual for more detail. 5.1.2.6 typedef struct sh2\_SensorConfig\_sh2\_SensorConfig\_t Sensor Configuration settings. See the SH-2 Reference Manual for more detail.

5.1.2.7 typedef struct sh2 SensorEvent sh2 SensorEvent t

See the SH-2 Reference Manual for more detail.

Sensor Event.

Comore

Generated by Doxygen

5.1 sh2.h File Reference 41

5.1.2.8 typedef struct sh2\_SensorMetadata sh2\_SensorMetadata\_t

Sensor Metadata Record.

See the SH-2 Reference Manual for more detail.

5.1.2.9 typedef enum sh2\_TareAxis sh2\_TareAxis\_t

Bit Fields for specifying tare axes.

See the SH-2 Reference Manual for more detail.

5.1.2.10 typedef enum sh2\_TareBasis sh2\_TareBasis\_t

Values for specifying tare basis.

See the SH-2 Reference Manual for more detail.

5.1.3 Enumeration Type Documentation

5.1.3.1 enum sh2\_CalStatus\_t

Calibration result.

See the SH-2 Reference Manual, Finish Calibration Response.

5.1.3.2 enum sh2\_OscType\_t

Oscillator type: Internal or External.

See the SH-2 Reference Manual for more detail.

5.1.3.3 enum sh2\_SensorId\_e

List of sensor types supported by the hub.

See the SH-2 Reference Manual for more information on each type.

5.1.3.4 enum sh2\_TareAxis

Bit Fields for specifying tare axes.

See the SH-2 Reference Manual for more detail.

**Enumerator** 

SH2\_TARE\_X sh2\_tareNow() axes bit field

SH2\_TARE\_Y sh2\_tareNow() axes bit field

SH2\_TARE\_Z sh2\_tareNow() axes bit field

#### 5.1.3.5 enum sh2\_TareBasis

Values for specifying tare basis.

See the SH-2 Reference Manual for more detail.

#### Enumerator

```
SH2_TARE_BASIS_ROTATION_VECTOR Use Rotation Vector.
SH2_TARE_BASIS_GAMING_ROTATION_VECTOR Use Game Rotation Vector.
SH2_TARE_BASIS_GEOMAGNETIC_ROTATION_VECTOR Use Geomagnetic R.V.
```

#### 5.1.4 Function Documentation

5.1.4.1 int sh2\_clearCounts ( sh2\_SensorId\_t sensorId )

Clear counters related to a sensor.

#### **Parameters**

sensor⊷	which sensor to operate on.
ld	

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

#### 5.1.4.2 int sh2\_clearDcdAndReset (void)

Command clear DCD in RAM, then reset sensor hub.

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

# 5.1.4.3 int sh2\_clearTare (void)

Clears the previously applied tare operation.

## Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1 sh2.h File Reference 43

## 5.1.4.4 int sh2\_finishCal ( sh2\_CalStatus\_t \* status )

Finish simple self-calibration procedure.

status contains calibration status code on return.

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

#### 5.1.4.5 int sh2\_flush ( sh2\_SensorId\_t sensorId )

Immediately issue all buffered sensor reports from a given sensor.

#### **Parameters**

sensor⊷	Which sensor reports to flush.
ld	

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

#### 5.1.4.6 int sh2\_getCalConfig ( uint8\_t \* pSensors )

Get dynamic calibration configuration settings.

#### **Parameters**

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.7 int sh2\_getCounts ( sh2\_SensorId\_t sensorId, sh2\_Counts\_t \* pCounts\_)

Read counters related to a sensor.

#### **Parameters**

sensor← Id	Which sensor to operate on.
pCounts	Pointer to Counts structure that will receive data.

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.8 int sh2\_getErrors ( uint8\_t severity, sh2\_ErrorRecord\_t \* pErrors, uint16\_t \* numErrors )

Get error counts.

#### **Parameters**

severity Only errors of this severity or greater are returned	
pErrors	Buffer to receive error codes.
numErrors	size of pErrors array

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.9 int sh2\_getFrs ( uint16\_t recordId, uint32\_t \* pData, uint16\_t \* words )

Get an FRS record.

#### **Parameters**

	record⊷	Which FRS Record to retrieve.
	ld	
	pData	pointer to buffer to receive the results
in	words	Size of pData buffer, in 32-bit words.
out	words	Number of 32-bit words retrieved.

## Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.10 int sh2\_getMetadata ( sh2\_SensorId\_t sensorId, sh2\_SensorMetadata\_t \* pData )

Get metadata related to a sensor.

#### **Parameters**

sensor⇔ Id	Which sensor to query.
pData	Pointer to structure to receive the results.

5.1 sh2.h File Reference 45

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.11 int sh2\_getOscType ( sh2\_OscType\_t \* pOscType )

Get Oscillator type.

#### **Parameters**

# Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.12 int sh2\_getProdlds ( sh2\_ProductIds\_t \* prodlds )

Get Product ID information from Sensorhub.

#### **Parameters**

prodlds	Pointer to structure that will receive results.
---------	---

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

 $5.1.4.13 \quad \text{int sh2\_getSensorConfig} \ ( \ \text{sh2\_SensorId\_t} \ sensorId, \ \text{sh2\_SensorConfig\_t} \ * \ config \ )$ 

Get sensor configuration.

#### **Parameters**

sensor⊷ Id	Which sensor to query.
config	SensorConfig structure to store results.

## Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.14 int sh2\_initialize ( sh2\_EventCallback\_t \* eventCallback, void \* resetCookie )

Initialize a session with the SensorHub.

This function should be called before any others in this API. The HAL and SHTP layers should be initialized BEFORE calling sh2\_init().

As part of the initialization process, a callback function is registered that will be invoked when the device completes the reset process.

#### **Parameters**

resetCallback	Will be called when the sensorhub completes the reset process.
resetCookie	Will be passed to resetCallback.

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

```
5.1.4.15 int sh2_persistTare (void)
```

Persist the results of last tare operation to flash.

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

```
5.1.4.16 int sh2_reinitialize (void)
```

Command the sensorhub to reset.

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

```
5.1.4.17 int sh2_saveDcdNow (void)
```

Save Dynamic Calibration Data to flash.

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

#### 5.1.4.18 int sh2\_setCalConfig ( uint8\_t sensors )

Enable/Disable dynamic calibration for certain sensors.

## **Parameters**

concore	Bit mask to configure which sensors are affected.
36113013	bit mask to configure which sensors are affected.

5.1 sh2.h File Reference 47

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.19 int sh2\_setDcdAutoSave (bool enabled)

Configure automatic saving of dynamic calibration data.

#### **Parameters**

enabled	Enable or Disable DCD auto-save.
---------	----------------------------------

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.20 int sh2\_setFrs ( uint16\_t recordId, uint32\_t \* pData, uint16\_t words )

Set an FRS record.

#### **Parameters**

record⇔ Id	Which FRS Record to set.	
pData	pointer to buffer containing the new data.	
words	number of 32-bit words to write. (0 to delete record.)	

## Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

 $5.1.4.21 \quad int \ sh2\_set Reorientation \ ( \ sh2\_Quaternion\_t * \textit{orientation} \ )$ 

Set the current run-time sensor reorientation. (Set to zero to clear tare.)

#### **Parameters**

orientation	Quaternion rotation vector to apply as new tare.

#### Returns

SH2 OK (0), on success. Negative value from sh2 err.h on error.

 $5.1.4.22 \quad \text{int sh2\_setSensorCallback ( } \text{sh2\_SensorCallback\_t} * \textit{callback, } \text{void} * \textit{cookie} \text{ )}$ 

Register a function to receive sensor events.

#### **Parameters**

callback	A function that will be called each time a sensor event is received.
cookie	A value that will be passed to the sensor callback function.

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.23 int sh2\_setSensorConfig ( sh2\_SensorId\_t sensorId, const sh2\_SensorConfig\_t \* pConfig\_)

Set sensor configuration. (e.g enable a sensor at a particular rate.)

#### **Parameters**

sensor⊷ Id	Which sensor to configure.	
pConfig	Pointer to structure holding sensor configuration.	

#### **Returns**

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.24 int sh2\_setTareNow ( uint8\_t axes, sh2\_TareBasis\_t basis )

Perform a tare operation on one or more axes.

#### **Parameters**

axes	Bit mask specifying which axes should be tared.
basis	Which rotation vector to use as the basis for Tare adjustment.

#### Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

5.1.4.25 int sh2\_startCal ( uint32\_t interval\_us )

Start simple self-calibration procedure.

interval\_us sensor report interval, uS.

## Returns

SH2\_OK (0), on success. Negative value from sh2\_err.h on error.

# 5.2 sh2\_err.h File Reference

Type definitions for Hillcrest SH-2 API.

#### **Macros**

- #define SH2\_OK (0)
- #define SH2\_ERR (-1)
- #define SH2\_ERR\_BAD\_PARAM (-2)
- #define SH2\_ERR\_OP\_IN\_PROGRESS (-3)
- #define SH2\_ERR\_IO (-4)
- #define SH2\_ERR\_HUB (-5)
- #define SH2\_ERR\_TIMEOUT (-6)

# 5.2.1 Detailed Description

Type definitions for Hillcrest SH-2 API.

Author

David Wheeler

Date

22 May 2015 Struct and type definitions supporting the Hillcrest SH-2 SensorHub API.

## 5.2.2 Macro Definition Documentation

5.2.2.1 #define SH2\_ERR (-1)

General Error

5.2.2.2 #define SH2\_ERR\_BAD\_PARAM (-2)

Bad parameter to an API call

5.2.2.3 #define SH2\_ERR\_HUB (-5)

Error reported by hub

5.2.2.4 #define SH2\_ERR\_IO (-4)

Error communicating with hub

```
5.2.2.5 #define SH2_ERR_OP_IN_PROGRESS (-3)
```

Operation in progress

```
5.2.2.6 #define SH2_ERR_TIMEOUT (-6)
```

Operation timed out

```
5.2.2.7 #define SH2_OK (0)
```

Success

# 5.3 sh2\_hal.h File Reference

Hardware Adaptation Layer API for SensorHub-2 (and BNO080)

```
#include <stdint.h>
#include <stdbool.h>
#include "sh2_hal_impl.h"
```

# **Typedefs**

• typedef void sh2\_rxCallback\_t(void \*cookie, uint8\_t \*pData, uint32\_t len, uint32\_t t\_us)

#### **Functions**

- int sh2\_hal\_reset (bool dfuMode, sh2\_rxCallback\_t \*onRx, void \*cookie)
- int sh2\_hal\_tx (uint8\_t \*pData, uint32\_t len)
- int sh2\_hal\_rx (uint8\_t \*pData, uint32\_t len)
- int sh2\_hal\_block (void)
- int sh2\_hal\_unblock (void)

# 5.3.1 Detailed Description

Hardware Adaptation Layer API for SensorHub-2 (and BNO080)

**Author** 

David Wheeler

Date

18 Nov 2016

# 5.4 sh2\_SensorValue.h File Reference

Support for converting sensor events (messages) into natural data structures.

```
#include <stdint.h>
#include "sh2.h"
```

#### **Data Structures**

· struct sh2 RawAccelerometer

Raw Accelerometer.

• struct sh2\_Accelerometer

Accelerometer.

• struct sh2\_RawGyroscope

Raw gyroscope.

• struct sh2\_Gyroscope

Gyroscope.

· struct sh2\_GyroscopeUncalibrated

Uncalibrated gyroscope.

• struct sh2\_RawMagnetometer

Raw Magnetometer.

· struct sh2\_MagneticField

Magnetic field.

• struct sh2\_MagneticFieldUncalibrated

Uncalibrated magnetic field.

• struct sh2\_RotationVectorWAcc

Rotation Vector with Accuracy.

· struct sh2 RotationVector

Rotation Vector.

• struct sh2 Pressure

Atmospheric Pressure.

• struct sh2\_AmbientLight

Ambient Light.

struct sh2\_Humidity

Humidity.

· struct sh2\_Proximity

Proximity.

• struct sh2\_Temperature

Temperature.

struct sh2\_Reserved

Reserved.

- struct sh2\_TapDetector
- struct sh2\_StepDetector

StepDetector.

struct sh2\_StepCounter

StepCounter.

• struct sh2\_SigMotion

SigMotion.

• struct sh2\_StabilityClassifier

- · struct sh2 ShakeDetector
- struct sh2\_FlipDetector

flipDetector

- struct sh2 PickupDetector
- · struct sh2 StabilityDetector
- · struct sh2 PersonalActivityClassifier
- struct sh2\_SleepDetector

sleepDetector

• struct sh2\_TiltDetector

tiltDetector

• struct sh2\_PocketDetector

pocketDetector

• struct sh2\_CircleDetector

circleDetector

· struct sh2 HeartRateMonitor

heartRateMonitor

struct sh2\_GyroIntegratedRV

heartRateMonitor

• struct sh2 SensorValue

#### **Macros**

#define TAPDET\_X (1)

TapDetector.

- #define TAPDET\_X\_POS (2)
- #define TAPDET\_Y (4)
- #define TAPDET\_Y\_POS (8)
- #define TAPDET\_Z (16)
- #define TAPDET\_Z\_POS (32)
- #define TAPDET\_DOUBLE (64)
- #define STABILITY\_CLASSIFIER\_UNKNOWN (0)

StabilityClassifier.

- #define STABILITY CLASSIFIER ON TABLE (1)
- #define STABILITY\_CLASSIFIER\_STATIONARY (2)
- #define STABILITY\_CLASSIFIER\_STABLE (3)
- #define STABILITY\_CLASSIFIER\_MOTION (4)
- #define SHAKE\_X (1)

ShakeDetector.

- #define SHAKE Y (2)
- #define SHAKE\_Z (4)
- #define PICKUP\_LEVEL\_TO\_NOT\_LEVEL (1)

pickupDetector

- #define PICKUP\_STOP\_WITHIN\_REGION (2)
- #define STABILITY\_ENTERED (1)

stabilityDetector

- #define STABILITY\_EXITED (2)
- #define PAC UNKNOWN (0)

Personal Activity Classifier.

- #define PAC\_IN\_VEHICLE (1)
- #define PAC\_ON\_BICYCLE (2)
- #define PAC ON FOOT (3)
- #define PAC\_STILL (4)
- #define PAC TILTING (5)
- #define PAC\_WALKING (6)
- #define PAC\_RUNNING (7)

#### **Typedefs**

```
    typedef struct sh2_RawAccelerometer sh2_RawAccelerometer_t

     Raw Accelerometer.

    typedef struct sh2_Accelerometer sh2_Accelerometer_t

     Accelerometer.

    typedef struct sh2_RawGyroscope sh2_RawGyroscope_t

     Raw gyroscope.

    typedef struct sh2_Gyroscope sh2_Gyroscope_t

     Gyroscope.

    typedef struct sh2_GyroscopeUncalibrated sh2_GyroscopeUncalibrated_t

     Uncalibrated gyroscope.

    typedef struct sh2_RawMagnetometer sh2_RawMagnetometer_t

     Raw Magnetometer.

    typedef struct sh2_MagneticField sh2_MagneticField_t

     Magnetic field.

    typedef struct sh2_MagneticFieldUncalibrated sh2_MagneticFieldUncalibrated_t

     Uncalibrated magnetic field.

    typedef struct sh2_RotationVectorWAcc sh2_RotationVectorWAcc_t

     Rotation Vector with Accuracy.

    typedef struct sh2_RotationVector sh2_RotationVector_t

     Rotation Vector.

    typedef struct sh2_Pressure sh2_Pressure_t

     Atmospheric Pressure.
• typedef struct sh2_AmbientLight sh2_AmbientLight_t
     Ambient Light.

    typedef struct sh2_Humidity sh2_Humidity_t

     Humidity.

    typedef struct sh2 Proximity sh2 Proximity t

     Proximity.

    typedef struct sh2_Temperature sh2_Temperature_t

     Temperature.
• typedef struct sh2 Reserved sh2 Reserved t
     Reserved.

    typedef struct sh2_TapDetector_t

    typedef struct sh2_StepDetector_sh2_StepDetector_t

     StepDetector.

    typedef struct sh2_StepCounter sh2_StepCounter_t

     StepCounter.

    typedef struct sh2_SigMotion sh2_SigMotion_t

     SigMotion.

    typedef struct sh2_StabilityClassifier sh2_StabilityClassifier_t

    typedef struct sh2_ShakeDetector sh2_ShakeDetector_t

    typedef struct sh2_FlipDetector_sh2_FlipDetector_t

     flipDetector

    typedef struct sh2_PickupDetector_t

    typedef struct sh2_StabilityDetector sh2_StabilityDetector_t

• typedef struct sh2_PersonalActivityClassifier sh2_PersonalActivityClassifier_t

    typedef struct sh2_SleepDetector sh2_SleepDetector_t

     sleepDetector

    typedef struct sh2_TiltDetector_sh2_TiltDetector_t
```

tiltDetector

 typedef struct sh2\_PocketDetector sh2\_PocketDetector\_t pocketDetector

 typedef struct sh2\_CircleDetector sh2\_CircleDetector\_t circleDetector

 typedef struct sh2\_HeartRateMonitor sh2\_HeartRateMonitor\_t heartRateMonitor

 typedef struct sh2\_GyroIntegratedRV sh2\_GyroIntegratedRV\_t heartRateMonitor

• typedef struct sh2\_SensorValue sh2\_SensorValue\_t

#### **Functions**

• int sh2\_decodeSensorEvent (sh2\_SensorValue\_t \*value, const sh2\_SensorEvent\_t \*event)

#### 5.4.1 Detailed Description

Support for converting sensor events (messages) into natural data structures.

**Author** 

David Wheeler

Date

10 Nov 2015

# 5.4.2 Macro Definition Documentation

5.4.2.1 #define PAC\_UNKNOWN (0)

Personal Activity Classifier.

See the SH-2 Reference Manual for more detail.

5.4.2.2 #define PICKUP\_LEVEL\_TO\_NOT\_LEVEL (1)

pickupDetector

See the SH-2 Reference Manual for more detail.

5.4.2.3 #define SHAKE\_X (1)

ShakeDetector.

5.4.2.4 #define STABILITY\_CLASSIFIER\_UNKNOWN (0) StabilityClassifier. See the SH-2 Reference Manual for more detail. 5.4.2.5 #define STABILITY\_ENTERED (1) stabilityDetector See the SH-2 Reference Manual for more detail. 5.4.2.6 #define TAPDET\_X (1) TapDetector. See the SH-2 Reference Manual for more detail. 5.4.3 Typedef Documentation 5.4.3.1 typedef struct sh2\_Accelerometer sh2\_Accelerometer\_t Accelerometer. See the SH-2 Reference Manual for more detail. 5.4.3.2 typedef struct sh2\_AmbientLight sh2\_AmbientLight\_t Ambient Light. See the SH-2 Reference Manual for more detail. 5.4.3.3 typedef struct sh2\_CircleDetector sh2\_CircleDetector\_t circleDetector See the SH-2 Reference Manual for more detail. 5.4.3.4 typedef struct sh2\_FlipDetector\_sh2\_FlipDetector\_t flipDetector

Generated by Doxygen

5.4.3.5 typedef struct sh2\_GyroIntegratedRV sh2\_GyroIntegratedRV\_t heartRateMonitor See SH-2 Reference Manual for details. 5.4.3.6 typedef struct sh2\_Gyroscope sh2\_Gyroscope\_t Gyroscope. See the SH-2 Reference Manual for more detail. 5.4.3.7 typedef struct sh2\_GyroscopeUncalibrated sh2\_GyroscopeUncalibrated\_t Uncalibrated gyroscope. See the SH-2 Reference Manual for more detail. 5.4.3.8 typedef struct sh2\_HeartRateMonitor sh2\_HeartRateMonitor\_t heartRateMonitor See SH-2 Reference Manual for details. 5.4.3.9 typedef struct sh2\_Humidity sh2\_Humidity\_t Humidity. See the SH-2 Reference Manual for more detail. 5.4.3.10 typedef struct sh2\_MagneticField sh2\_MagneticField\_t Magnetic field. See the SH-2 Reference Manual for more detail. 5.4.3.11 typedef struct sh2\_MagneticFieldUncalibrated\_t Uncalibrated magnetic field. See the SH-2 Reference Manual for more detail. 5.4.3.12 typedef struct sh2\_PocketDetector sh2\_PocketDetector\_t pocketDetector

5.4.3.13 typedef struct sh2\_Pressure sh2\_Pressure\_t Atmospheric Pressure. See the SH-2 Reference Manual for more detail. 5.4.3.14 typedef struct sh2\_Proximity sh2\_Proximity\_t Proximity. See the SH-2 Reference Manual for more detail. 5.4.3.15 typedef struct sh2\_RawAccelerometer\_t Raw Accelerometer. See the SH-2 Reference Manual for more detail. 5.4.3.16 typedef struct sh2\_RawGyroscope sh2\_RawGyroscope\_t Raw gyroscope. See the SH-2 Reference Manual for more detail. 5.4.3.17 typedef struct sh2\_RawMagnetometer sh2\_RawMagnetometer\_t Raw Magnetometer. See the SH-2 Reference Manual for more detail. 5.4.3.18 typedef struct sh2 Reserved sh2 Reserved t Reserved. See the SH-2 Reference Manual for more detail. 5.4.3.19 typedef struct sh2\_RotationVector sh2\_RotationVector\_t Rotation Vector. See the SH-2 Reference Manual for more detail. 5.4.3.20 typedef struct sh2\_RotationVectorWAcc sh2\_RotationVectorWAcc\_t

Generated by Doxygen

Rotation Vector with Accuracy.

```
5.4.3.21 typedef struct sh2_SigMotion sh2_SigMotion_t
SigMotion.
See the SH-2 Reference Manual for more detail.
5.4.3.22 typedef struct sh2_SleepDetector_t
sleepDetector
See the SH-2 Reference Manual for more detail.
5.4.3.23 typedef struct sh2_StepCounter_t
StepCounter.
See the SH-2 Reference Manual for more detail.
5.4.3.24 typedef struct sh2_StepDetector_sh2_StepDetector_t
StepDetector.
See the SH-2 Reference Manual for more detail.
5.4.3.25 typedef struct sh2_Temperature sh2_Temperature_t
Temperature.
See the SH-2 Reference Manual for more detail.
5.4.3.26 typedef struct sh2_TiltDetector_sh2_TiltDetector_t
tiltDetector
See the SH-2 Reference Manual for more detail.
```

# Index

heartRate	SH2_TARE_BASIS_GEOMAGNETIC_ROTATI←
sh2_HeartRateMonitor, 17	ON_VECTOR, 42
	SH2_TARE_BASIS_ROTATION_VECTOR, 42
PAC_UNKNOWN	SH2_TARE_X, 41
sh2_SensorValue.h, 54	SH2_TARE_Y, 41
PICKUP_LEVEL_TO_NOT_LEVEL	SH2_TARE_Z, 41
sh2_SensorValue.h, 54	sh2_AsyncEvent_t, 40
pickup	sh2_CalStatus_t, 41
sh2_PickupDetector, 19	sh2 Counts t, 40
	sh2_ErrorRecord_t, 40
SH2_ERR_BAD_PARAM	sh2_OscType_t, 41
sh2_err.h, 49	sh2 Productld t, 40
SH2_ERR_HUB	sh2_Quaternion_t, 40
sh2_err.h, 49	sh2_SensorConfig_t, 40
SH2_ERR_IO	sh2_SensorEvent_t, 40
sh2_err.h, 49	sh2_SensorId_e, 41
SH2_ERR_OP_IN_PROGRESS	sh2 SensorMetadata t, 40
sh2_err.h, 49	sh2 TareAxis, 41
SH2_ERR_TIMEOUT	sh2_TareAxis_t, 41
sh2_err.h, 50	
SH2_ERR	sh2_TareBasis, 41
sh2_err.h, 49	sh2_TareBasis_t, 41
SH2_OK	sh2_clearCounts, 42
sh2_err.h, 50	sh2_clearDcdAndReset, 42
SH2_TARE_BASIS_GAMING_ROTATION_VECTOR	sh2_clearTare, 42
sh2.h, 42	sh2_finishCal, 42
SH2_TARE_BASIS_GEOMAGNETIC_ROTATION_V	sh2_flush, 43
ECTOR	sh2_getCalConfig, 43
sh2.h, 42	sh2_getCounts, 43
SH2_TARE_BASIS_ROTATION_VECTOR	sh2_getErrors, 44
sh2.h, 42	sh2_getFrs, 44
SH2_TARE_X	sh2_getMetadata, 44
sh2.h, 41	sh2_getOscType, 45
SH2_TARE_Y	sh2_getProdIds, 45
sh2.h, 41	sh2_getSensorConfig, 45
SH2_TARE_Z	sh2_initialize, 45
sh2.h, 41	sh2_persistTare, 46
SHAKE_X	sh2_reinitialize, 46
sh2_SensorValue.h, 54	sh2_saveDcdNow, 46
STABILITY CLASSIFIER UNKNOWN	sh2_setCalConfig, 46
sh2 SensorValue.h, 54	sh2_setDcdAutoSave, 47
STABILITY ENTERED	sh2_setFrs, 47
sh2_SensorValue.h, 55	sh2_setReorientation, 47
sensorId	sh2_setSensorCallback, 47
sh2_SensorValue, 29	sh2_setSensorConfig, 48
sequence	sh2_setTareNow, 48
sh2_SensorValue, 29	sh2_startCal, 48
sh2.h, 35	sh2 Accelerometer, 11
SH2_TARE_BASIS_GAMING_ROTATION_VE↔	sh2_Accelerometer_t
CTOR, 42	sh2_SensorValue.h, 55

60 INDEX

sh2_	_AmbientLight, 11	sh2	_Proximity_t
sh2_	_AmbientLight_t		sh2_SensorValue.h, 57
	sh2_SensorValue.h, 55	sh2	_Quaternion, 21
sh2_	_AsyncEvent, 12	sh2	_Quaternion_t
sh2_	_AsyncEvent_t		sh2.h, 40
	sh2.h, 40	sh2	_RawAccelerometer, 22
sh2_	_CalStatus_t	sh2	_RawAccelerometer_t
	sh2.h, 41		sh2_SensorValue.h, 57
sh2_	_CircleDetector, 12	sh2	_RawGyroscope, 22
sh2_	_CircleDetector_t	sh2	_RawGyroscope_t
	sh2_SensorValue.h, 55		sh2_SensorValue.h, 57
sh2_	_Counts, 13	sh2	_RawMagnetometer, 23
sh2_	_Counts_t	sh2	_RawMagnetometer_t
	sh2.h, 40		sh2_SensorValue.h, 57
_	ErrorRecord, 13		_Reserved, 24
sh2_	_ErrorRecord_t	sh2	_Reserved_t
	sh2.h, 40		sh2_SensorValue.h, 57
sh2_	_FlipDetector, 14	sh2	_RotationVector, 24
sh2_	_FlipDetector_t	sh2	_RotationVector_t
	sh2_SensorValue.h, 55		sh2_SensorValue.h, 57
sh2_	_GyroIntegratedRV_t	sh2	_RotationVectorWAcc, 25
	sh2_SensorValue.h, 55	sh2	_RotationVectorWAcc_t
sh2_	_GyroIntegratedRV, 15		sh2_SensorValue.h, 57
sh2_	_Gyroscope, 15	sh2	_SensorConfig, 25
sh2_	_Gyroscope_t	sh2	_SensorConfig_t
	sh2_SensorValue.h, 56		sh2.h, 40
sh2_	_GyroscopeUncalibrated, 16	sh2	_SensorEvent, 26
sh2_	_GyroscopeUncalibrated_t	sh2	_SensorEvent_t
	sh2_SensorValue.h, 56		sh2.h, 40
sh2_	_HeartRateMonitor, 16	sh2	_Sensorld_e
	heartRate, 17		sh2.h, 41
sh2_	_HeartRateMonitor_t	sh2	_SensorMetadata, 26
	sh2_SensorValue.h, 56	sh2	_SensorMetadata_t
	_Humidity, 17		sh2.h, 40
sh2_	_Humidityt	sh2	_SensorValue, 28
	sh2_SensorValue.h, 56		sensorld, 29
	_MagneticField, 17		sequence, 29
sh2_	_MagneticField_t		timestamp, 29
	sh2_SensorValue.h, 56		un, 29
	_MagneticFieldUncalibrated, 18	sh2	_SensorValue.h, 51
sh2_	_MagneticFieldUncalibrated_t		PAC_UNKNOWN, 54
	sh2_SensorValue.h, 56		PICKUP_LEVEL_TO_NOT_LEVEL, 54
sh2_	_OscType_t		SHAKE_X, 54
	sh2.h, 41		STABILITY_CLASSIFIER_UNKNOWN, 54
	PersonalActivityClassifier, 19		STABILITY_ENTERED, 55
sh2_	_PickupDetector, 19		sh2_Accelerometer_t, 55
	pickup, 19		sh2_AmbientLight_t, 55
	PocketDetector, 19		sh2_CircleDetector_t, 55
sh2_	_PocketDetector_t		sh2_FlipDetector_t, 55
	sh2_SensorValue.h, 56		sh2_GyroIntegratedRV_t, 55
_	Pressure, 20		sh2_Gyroscope_t, 56
sh2_	Pressure_t		sh2_GyroscopeUncalibrated_t, 56
	sh2_SensorValue.h, 56		sh2_HeartRateMonitor_t, 56
	_ProductId_s, 20		sh2_Humidity_t, 56
sh2_	_ProductId_t		sh2_MagneticField_t, 56
	sh2.h, 40		sh2_MagneticFieldUncalibrated_t, 56
	_ProductIds_s, 21		sh2_PocketDetector_t, 56
sn2_	Proximity, 21		sh2_Pressure_t, 56

INDEX 61

	10.0		0110 017 50
	sh2_Proximity_t, 57		SH2_OK, 50
	sh2_RawAccelerometer_t, 57	sh2	_finishCal
	sh2_RawGyroscope_t, 57		sh2.h, 42
	sh2_RawMagnetometer_t, 57	sh2	_flush
	sh2_Reserved_t, 57		sh2.h, 43
	sh2_RotationVector_t, 57	sh2	_getCalConfig
	sh2_RotationVectorWAcc_t, 57		sh2.h, 43
	sh2_SigMotion_t, 57	sh2	_getCounts
	sh2_SleepDetector_t, 58		sh2.h, 43
	sh2_StepCounter_t, 58	sh2	_getErrors
	sh2_StepDetector_t, 58	_	sh2.h, 44
	sh2_Temperature_t, 58	sh2	_getFrs
	sh2_TiltDetector_t, 58		sh2.h, 44
		sh2	_getMetadata
-  - 0	TAPDET_X, 55	0112	sh2.h, 44
	ShakeDetector, 29	ch2	_getOscType
	SigMotion, 29	3112	_getOscrype _sh2.h, 45
sh2_	_SigMotion_t	٥b٥	
	sh2_SensorValue.h, 57	SIIZ	_getProdIds
	_SleepDetector, 30		sh2.h, 45
sh2_	_SleepDetector_t	sn2	_getSensorConfig
	sh2_SensorValue.h, 58		sh2.h, 45
sh2_	_StabilityClassifier, 30		_hal.h, 50
sh2_	_StabilityDetector, 31	sh2	_initialize
	stability, 31		sh2.h, 45
sh2	StepCounter, 31	sh2	_persistTare
	StepCounter_t		sh2.h, 46
_	sh2_SensorValue.h, 58	sh2	_reinitialize
sh2	StepDetector, 31		sh2.h, 46
	StepDetector_t	sh2	_saveDcdNow
0112_	sh2_SensorValue.h, 58		sh2.h, 46
ch2	TapDetector, 32	sh2	_setCalConfig
			sh2.h, 46
5112_	_TareAxis	sh2	_setDcdAutoSave
	sh2.h, 41		sh2.h, 47
sn2_	_TareAxis_t	sh2	setFrs
	sh2.h, 41	_	sh2.h, 47
sh2_	_TareBasis	sh2	setReorientation
	sh2.h, 41	o <u>-</u> _	sh2.h, 47
sh2_	_TareBasis_t	ch2	setSensorCallback
	sh2.h, 41	3112	sh2.h, 47
sh2_	_Temperature, 32	ch2	setSensorConfig
sh2_	_Temperature_t	3112	_sh2.h, 48
	sh2_SensorValue.h, 58	cho	_setTareNow
sh2_	_TiltDetector, 33	5112	
sh2	TiltDetector t	-60	sh2.h, 48
_	sh2_SensorValue.h, 58	sn2	_startCal
sh2	clearCounts		sh2.h, 48
	sh2.h, 42	stab	-
sh2	clearDcdAndReset		sh2_StabilityDetector, 31
3112_	sh2.h, 42	<b>T</b> 4 F	DET V
oh2		IAF	PDET_X
511Z_	_clearTare		sh2_SensorValue.h, 55
a-0	sh2.h, 42	time	estamp
sn2_	err.h, 49		sh2_SensorValue, 29
	SH2_ERR_BAD_PARAM, 49		
	SH2_ERR_HUB, 49	un	
	SH2_ERR_IO, 49		sh2_SensorValue, 29
	SH2_ERR_OP_IN_PROGRESS, 49		
	SH2_ERR_TIMEOUT, 50		
	SH2 ERR. 49		