SH-2 API User's Guide

Generated by Doxygen 1.8.11

Contents

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↩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 accomodate 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.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

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 j

  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 vendorIdLen

  [bytes]
- char vendorId [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 sensorId
- 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^ exponent (see status)
- union {
  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)

- #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_ProductIds_s **sh2_ProductIds_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_ACCELERATION** = 0x04, **SH2_GRAVITY** = 0x06,
  SH2_RAW_GYROSCOPE = 0x15, **SH2_GYROSCOPE_CALIBRATED** = 0x02, **SH2_GYROSCOPE_UNCALIBRATED** = 0x07, **SH2_RAW_MAGNETOMETER** = 0x16,
  SH2_MAGNETIC_FIELD_CALIBRATED = 0x03, **SH2_MAGNETIC_FIELD_UNCALIBRATED** = 0x0f, **SH2_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_DETECTOR** = 0x10,
  SH2_STEP_DETECTOR = 0x18, **SH2_STEP_COUNTER** = 0x11, **SH2_SIGNIFICANT_MOTION** = 0x12, **SH2_STABILITY_CLASSIFIER** = 0x13,
  SH2_SHAKE_DETECTOR = 0x19, **SH2_FLIP_DETECTOR** = 0x1a, **SH2_PICKUP_DETECTOR** = 0x1b, **SH2_STABILITY_DETECTOR** = 0x1c,
  SH2_PERSONAL_ACTIVITY_CLASSIFIER = 0x1e, **SH2_SLEEP_DETECTOR** = 0x1f, **SH2_TILT_DETECTOR** = 0x20, **SH2_POCKET_DETECTOR** = 0x21,
  SH2_CIRCLE_DETECTOR = 0x22, **SH2_HEART_RATE_MONITOR** = 0x23, **SH2_ARVR_STABILIZED_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_ROTATION_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_CLOCK** = 2 }

  Oscillator type: Internal or External.
- enum sh2_CalStatus_t {
  SH2_CAL_SUCCESS = 0, **SH2_CAL_NO_ZRO**, **SH2_CAL_NO_STATIONARY_DETECTION**, **SH2_CAL_ROTATION_OUTSIDE_SPEC**,
  SH2_CAL_ZRO_OUTSIDE_SPEC, **SH2_CAL_ZGO_OUTSIDE_SPEC**, **SH2_CAL_GYRO_GAIN_OUTSIDE_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.

- 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_SensorId_t sensorId, 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

Sensor Event.

See the SH-2 Reference Manual for more detail.

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

sensorId	which sensor to operate on.

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.

5.1.4.4 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← Id	Which sensor reports to flush.

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

pSensors	pointer to Bit mask, set on return.

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↩ Id	Which FRS Record to retrieve.
	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.

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

pOscType	pointer to data structure to receive results.

Returns

SH2_OK (0), on success. Negative value from sh2_err.h on error.

5.1.4.12 int sh2_getProdIds (sh2_ProductIds_t ∗ *prodIds*)

Get Product ID information from Sensorhub.

Parameters

prodIds	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 int sh2_getSensorConfig (sh2_SensorId_t *sensorId*, sh2_SensorConfig_t ∗ *config*)

Get sensor configuration.

Parameters

sensorId	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

sensors	Bit mask to configure which sensors are affected.

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

recordId	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 int sh2_setReorientation (sh2_Quaternion_t ∗ *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 int sh2_setSensorCallback (sh2_SensorCallback_t ∗ *callback,* void ∗ *cookie*)

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 **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 **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.

See the SH-2 Reference Manual for more detail.

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

See the SH-2 Reference Manual for more detail.

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 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

See the SH-2 Reference Manual for more detail.

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 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

Rotation Vector with Accuracy.

See the SH-2 Reference Manual for more detail.

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 sh2_SleepDetector_t

sleepDetector

See the SH-2 Reference Manual for more detail.

5.4.3.23 typedef struct sh2_StepCounter 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