User Manual

DA14681 Wearable Development Kit API

UM-B-085

Abstract

This document describes the API specifications of software modules of the application running on the Dialog DA14681 Wearable Development Kit. This manual intends to assist software developers who implement applications using the DA14681 Wearable Development Kit.

Contents

 © 2017 Dialog Semiconductor

 © 2017 Dialog Semiconductor

 © 2017 Dialog Semiconductor

 © 2017 Dialog Semiconductor

Figures

 © 2017 Dialog Semiconductor

Tables

1 Terms and Definitions

AHRS	Attitude and Heading Reference System
API	Application Programming Interface
BAS	Battery Service
BD	Bluetooth Device (address)
BLE	Bluetooth low energy
BMR	Basal Metabolic Rate
CAS	Custom Alarm Service
CC	Constant Current (charging)
CIB	Connection Interface Board
CV	Constant Voltage (charging)

 © 2017 Dialog Semiconductor

DK	Development Kit
DWS	Dialog Wearable Service
epoch	time slot of a constant duration (wearables: 1 min)
FW	Firmware
GAP	Generic Access Profile
GATT	Generic ATTribute profile
GDI	Graphical Device Interface
GUI	Graphical User Interface
HCS	Health Care Service
HT	Health Toolbox
KBSCN	Keyboard Scanner
MEMS	Micro Electro-Mechanical Systems
MTU	Maximum Transmission Unit
NED	North East Down (coordinate system)
NVDS	Non-Volatile Data Storage
NVMS	Non Volatile Memory Storage
RAM	Random Access Memory
REM	Rapid Eye Movement
Report	Notification of sensor data and control
RTC	Real Time Clock
SF	Sensor Fusion
SFL	Sensor Fusion Library
SOC	State Of Charge
SUOTA	Software Update Over The Air
UI	User Interface

© 2017 Dialog Semiconductor

2 References

[1] UM-B-076_DA14681 Wearable Development Kit Software Manual.

[2] UM-B-044, DA1468x Software Platform reference, User Manual, Dialog Semiconductor.

[3] UM-B-056, DA1468x Software Developer's guide, User Manual, Dialog Semiconductor.

[4] AN-B-003, DA14580 Software Patching over the Air (SPotA), Application Note, Dialog Semiconductor.

[5] HRS_SPEC, HEART RATE SERVICE SPECIFICATION v1.0, Bluetooth SIG.

[6] CTS_SPEC, CURRENT TIME SERVICE SPECIFICATION V1.1, Bluetooth SIG.

[7] BAS_SPEC, BATTERY SERVICE SPECIFICATION V1.1, Bluetooth SIG.

[8] LS013B7DH03, Datasheet, Sharp Microelectronics.

[9] UM-B-077, DA14681 Wearable Development Kit Hardware Manual, User Manual, Dialog Semiconductor.

[10] BMI160, Datasheet, Bosch Sensortec.

[11] BMM150, Datasheet, Bosch Sensortec.

[12] BME280, Datasheet, Bosch Sensortec.

[13] FXL6408, Datasheet, Fairchild Semiconductor.

 © 2017 Dialog Semiconductor

3 Introduction

The Wearable Development Kit is a platform for development of wearables applications using Dialog's DA14681 System on Chip solution combining an application processor, memories, cryptography engine, power management unit, digital and analog peripherals and a Bluetooth® Smart MAC engine and radio transceiver.

Figure 1: The Wearable Development Kit

The purpose of this user manual is to provide a brief overview of the architecture of the reference software for Wearable Development Kit, provide the specification of the software modules and the corresponding APIs.A more detailed description of the architecture, the set of necessary hardware and software components for development on Wearable Development Kit and guidelines to install and use the reference Smartphone/Tablet apps can be found in the user manual of the DA14681 wearable development kit [1].

4 Software Architecture Overview

The software architecture of the wearable application is outlined in the following image.

© 2017 Dialog Semiconductor

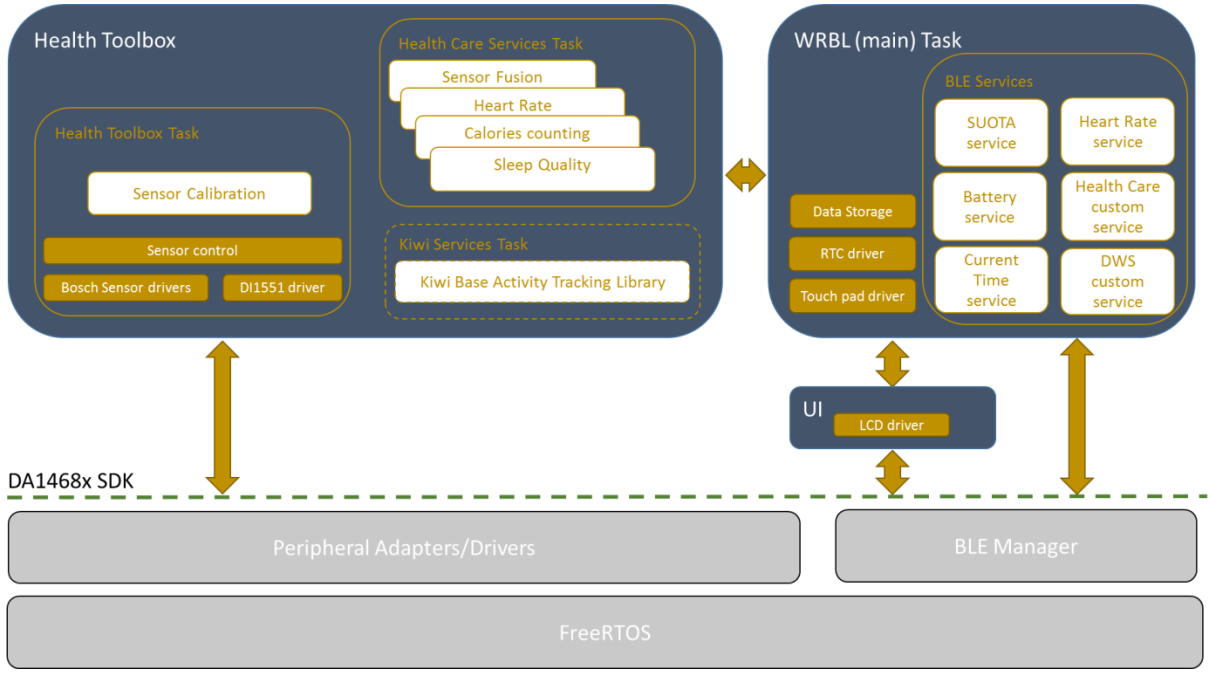

Figure 2: Software Architecture

The application software is organized in FreeRTOS tasks running on the top of the DA1468x SDK. All the details of DA1468x SDK can be found in [2] and [3].

The main software modules and tasks of the wearable application are the following:

- **WRBL Task.** This is the main task of the application and is responsible for the control and the coordination of the other application tasks and transferring data from/to the BLE Services. It is also responsible for controlling and handling events coming from the RTC, capacitive touchpads and mechanical button. Finally, it is responsible for writing/reading application and configuration data to/from the Flash memory.
- **Health Toolbox.** The health toolbox software module consists of two tasks.
  - **Health Toolbox task.** This task is responsible for controlling and accessing the peripheral sensors for movement and HR monitoring. Health toolbox gets commands from WRBL task regarding the configuration of sensors and health care services, it accesses sensors via the Sensor Control software component, and provides the sensor data to WRBL task and Health Care Services task. Finally it processes the output of Health Care Services task to send the required health care data to Wearable task for further manipulation (storage in Flash memory, displayed on TFT Display or transmitted over BLE via the corresponding service)
  - **Health Care Services task.** This task consists of the algorithms for health care (Sleep Quality Monitoring, Calories Counting and Heart Rate Monitoring), sensor fusion and magneto calibration. It gets input sensor and configuration data from the Health toolbox task and sends back the output of the algorithms.
- **UI Task.** This task is responsible for the graphical user interface of the wearable device display. It gets commands from the WRBL task regarding the screen and the data to be displayed, the status bar information, and other graphic effects such as blinking.

© 2017 Dialog Semiconductor

5 Wearable Task

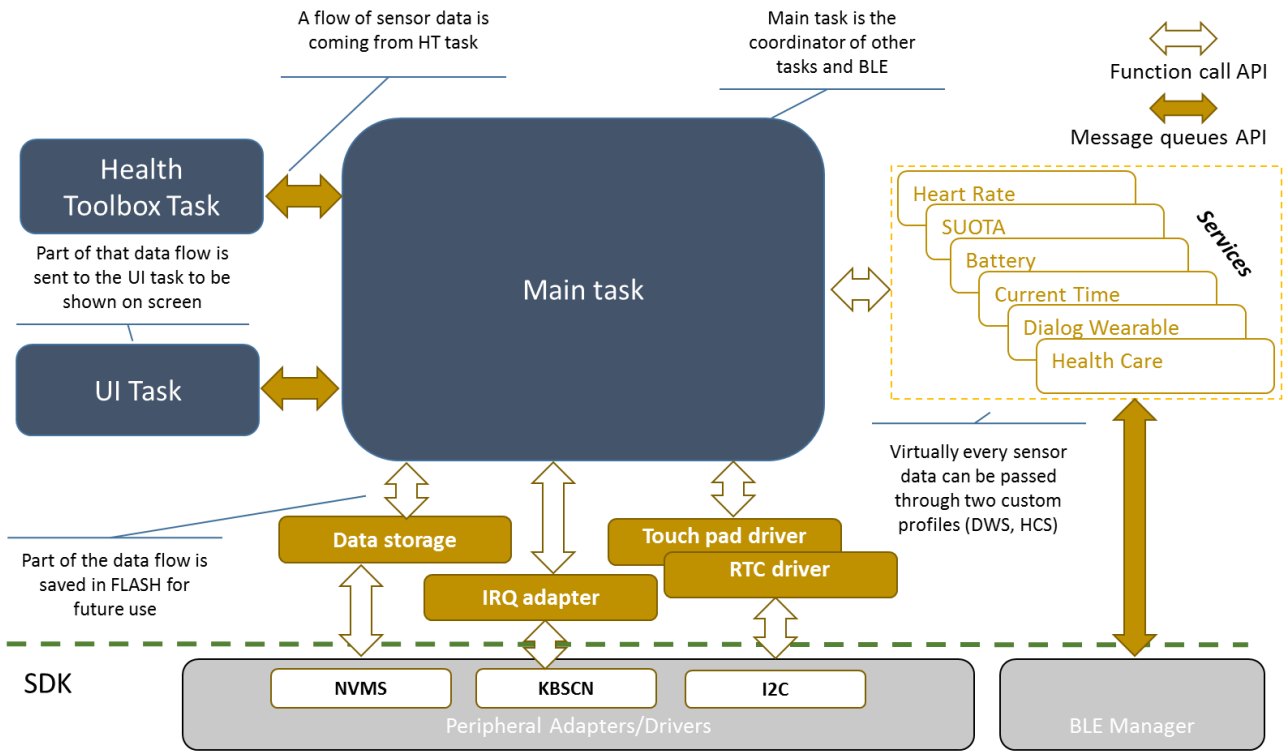

Figure 3: Main Application Components

The wearable main task (WRBL) handles the data transfers between other tasks (e.g. step counter data generated in the Health Toolbox task transferred to the UI task to be displayed and/or to the BLE Manager task to be sent to a peer device). At a high level it controls the data flow to/from the Health Toolbox, User Interface and BLE Manager tasks by sending the appropriate messages using the corresponding API function calls.

It controls the data stored in the SPI Flash memory with the use of the data storage functions, which provide a structured storage of the required data. Data types that are stored are the heart rate, steps, calories and sleep quality, using minimum space in Flash memory. The lower level Flash operations are performed by the SDK's NVMS adapter, which handles all necessary signaling.

5.1 BLE

Wearable task is using BLE profiles to communicate the produced data and receive commands from a connected peer device. The task - profile interaction of the following profiles, due to their complexity, are implemented in separate files andtheir API is described below.

5.1.1 CTS

Table 1: Wearable CTS API

Functions	
wrbl_cts_init()	wrbl_cts_notify_time_all()

© 2017 Dialog Semiconductor

5.1.1.1 Functions

wrbl_cts_init()

Function Name	`void wrbl_cts_init(void);`
Function Description	Initialize the CTS component of the wearable task
Parameters	None
Return Values	None
Notes	

wrbl_cts_notify_time_all()

Function Name	`void wrbl_cts_notify_time_all(bool update_time);`
Function Description	Notifies all connected peer devices of the (current) time/date.
Parameters	`update_time` Set true to force the update of the current time
Return Values	None
Notes	Current time is received directly by the RTC module.

5.1.2 DWS

Table 2: Wearable DWS API

Functions	
wrbl_dws_init() wrbl_dws_send_cmd_report()	wrbl_dws_send_data_report()

© 2017 Dialog Semiconductor

5.1.2.1 Functions

wrbl_dws_init()

Function Name	`void wrbl_dws_init(void);`
Function Description	Initialize the DWS component of the wearable task.
Parameters	None
Return Values	None
Notes	

wrbl_dws_send_data_report()

Function Name	`void wrbl_dws_send_data_report(iot_report_id_t type, void * const data);`
Function Description	Send data reports to the connected peer device.
Parameters	`type` Report type `data` Data to be included in the report
Return Values	None
Notes	Depending on the type the following data MUST be provided:

Type	Data type
IOT_ACCELEROMETER_REPORT_ID	ht_data_acc_t
IOT_GYROSCOPE_REPORT_ID	ht_data_gyr_t
IOT_MAGNETOMETER_REPORT_ID	ht_data_mag_t
IOT_PRESSURE_REPORT_ID	ht_data_env_t
IOT_HUMIDITY_REPORT_ID	ht_data_env_t
IOT_TEMPERATURE_REPORT_ID	ht_data_env_t
IOT_SENSOR_FUSION_REPORT_ID	ht_data_sfl_t

wrbl_dws_send_cmd_report()

Function Name	`void wrbl_dws_send_cmd_report(iot_control_byte_t type, void * const data);`
Function Description	Send data reports to the connected peer device.
Parameters	`type` Report type `data` Data to be included in the report
Return Values	None
Notes	Depending on the type the following data MUST be provided:

Type	Data type
IOT_STOP_CMD	uint8_t
IOT_START_CMD	uint8_t
IOT_GET_RUNNING_STATUS_CMD	uint8_t
IOT_BASIC_CONFIG_RD_REQ	iot_basic_setup_t
IOT_SFL_COEFF_RD_REQ	iot_sfl_coeffs_t
IOT_CAL_COEFF_RD_REQ	iot_calibr_coeffs_t
IOT_CAL_CTRL_RD_REQ	iot_calibr_ctrl_t
IOT_START_ACC_ALLIGN	uint8_t

© 2017 Dialog Semiconductor

5.1.3 HCS

Table 3: Wearable HCS API

Functions	
wrbl_hcs_init()	handle_connected_hcs()
hcs_update_remote_db()	wrbl_hcs_send_acc_counter()
wrbl_hcs_send_epoch()	wrbl_hcs_send_status()
handle_hcs_msg()	

© 2017 Dialog Semiconductor

5.1.3.1 Functions

wrbl_hcs_init()

Function Name	`void wrbl_hcs_init(void);`
Function Description	Initialize the HCS component of the wearable task.
Parameters	None
Return Values	None
Notes	

handle_connected_hcs()

Function Name	`void handle_connected_hcs(void);`
Function Description	Function to inform the component of the new connection.
Parameters	None
Return Values	None
Notes	

hcs_update_remote_db()

Function Name	`bool hcs_update_remote_db(uint16_t conn_idx, uint32_t time, hcs_op_code_t type);`
Function Description	Function to initiate a database update to the peer device when there is an update in the stored values.
Parameters	`conn_idx` Connection index `time` Time of the oldest updated sample `type` Type of the samples the update occurred
Return Values	True if the update was performed successfully, false if the update is scheduled to be performed when the peer device connects and synchronizes database.
Notes	

wrbl_hcs_send_acc_counter()

Function Name	`void wrbl_hcs_send_acc_counter(hcs_op_code_t op_code, uint32_t value);`
Function Description	Send accumulated counter to connected peer device
Parameters	`op_code` Type of the accumulated counter `value` Value of the accumulated counter
Return Values	None

Notes	Valid types (op_codes) are:
	`HCS_OP_CODE_STEP`
	`HCS_OP_CODE_LIGHT_SLEEP`
	`HCS_OP_CODE_DEEP_SLEEP`
	`HCS_OP_CODE_CALORIES`

wrbl_hcs_send_epoch()

Function Name	`void wrbl_hcs_send_epoch(hcs_op_code_t op_code, uint32_t value, uint32_t timestamp);`
Function Description	Send accumulated counter to connected peer device

© 2017 Dialog Semiconductor

Parameters	`op_code`	Type of the accumulated counter
	`value`	Value of the accumulated counter
	`timestamp`	Timestamp of the epoch
Return Values	None	

Notes	Valid types (op_codes) are:
	`HCS_OP_CODE_STEP`
	`HCS_OP_CODE_HEART_RATE`
	`HCS_OP_CODE_SLEEP_QUALITY`
	`HCS_OP_CODE_CALORIES`
	`HCS_OP_CODE_BODY_STATE`	

wrbl_hcs_send_status()

Function Name	`void wrbl_hcs_send_status(hcs_op_code_t op_code, bool status);`	
Function Description	Send service status to connected peer device	
Parameters	`op_code`	Type of the accumulated counter
	`status`	Status of the service
Return Values	None	

Notes	Valid types (op_codes) are:
	`HCS_OP_CODE_STEP`
	`HCS_OP_CODE_HEART_RATE`
	`HCS_OP_CODE_SLEEP_QUALITY`
	`HCS_OP_CODE_CALORIES`
	`HCS_OP_CODE_BODY_STATE`	

handle_hcs_msg()

Function Name	`void handle_hcs_msg(msg *wrbl_msg);`	
Function Description	Function handles all the HCS messages	
Parameters	`type`	Report type
	`wrbl_msg`	Message to be handled
Return Values	None	
Notes		

5.1.4 SUOTA

Table 4: Wearable SUOTA API

Functions	
wrbl_suota_init()	

© 2017 Dialog Semiconductor

5.1.4.1 Functions

wrbl_suota_init()

Function Name	`void wrbl_suota_init(void);`
Function Description	Initialize the SUOTA module.
Parameters	None
Return Values	None
Notes	

5.2 Interface

The wearable device's user inputs are the push button and two touch buttons. The two touch buttons are used together to create a slide detection mechanism. Depending on what is displayed on the screen the user can set time, date and alarm, and also control various states of the sensors and services.

5.2.1 Touch Pad

Table 5: Wearable touch pad API

Structures	
wrbl_touch_event_t	
Functions	
wrbl_touch_init() handle_touch_event()	wrbl_touch_set_enable()
Callback functions	
wrbl_touch_event_callback()	

5.2.1.1 Data Structures and Types

wrbl_touch_event_t

Type Definition Name	Description

`wrbl_touch_event_t`	Touch event enumeration type defined as:
	`WRBL_TOUCH_SWIPE_RIGHT,`
	`WRBL_TOUCH_SWIPE_UP,`
	`WRBL_TOUCH_SWIPE_LEFT,`
	`WRBL_TOUCH_SWIPE_DOWN,`

© 2017 Dialog Semiconductor

5.2.1.2 Functions

wrbl_touch_init()

Function Name	`void wrbl_touch_init(wrbl_touch_event_callback event_cb);`
Function Description	Initialize the touch module.
Parameters	`event_cb` Callback function to be called when a touch event is detected
Return Values	None
Notes	

wrbl_touch_set_enable()

Function Name	`void wrbl_touch_set_enable(bool enable);`
Function Description	Change the operation state of the touch sensor module.
Parameters	`enable` Enable/disable the touch module
Return Values	None
Notes	

handle_touch_event()

Function Name	`void handle_touch_event(void);`
Function Description	Function handles the touch events of the wearable task.
Parameters	None
Return Values	None
Notes	

5.2.1.3 Callback Functions

wrbl_touch_event_callback()

Function Name	`void wrbl_touch_event_callback(wrbl_touch_event_t evt);`
Function Description	Callback function to be called when a new event is detected.
Parameters	`evt` Event type
Return Values	None
Notes	

5.2.2 Push Button

Table 6: Wearable push button API

Structures	
btn_press_t	
Functions	
wrbl_button_init()	handle_button_event()
Callback functions	
handle_button_callback()	

© 2017 Dialog Semiconductor

5.2.2.1 Data Structures and Types

btn_press_t

Type Definition Name	Description
btn_press_t	Button press enumeration type defined as: BTN_SHORT_PRESS, BTN_LONG_PRESS,

5.2.2.2 Functions

wrbl_button_init()

Function Name	void wrbl_button_init(handle_button_callback cb);
Function Description	Initialize button component of the wearable task.
Parameters	cb Callback function to be called when an event is produced
Return Values	None
Notes	

handle_button_event()

Function Name	void handle_button_event();
Function Description	Function that handles the button notifications.
Parameters	None
Return Values	None
Notes	

5.2.2.3 Callback Functions

handle_button_callback()

Function Name	void handle_button_callback(btn_press_t press);
Function Description	Callback function to be called when a new event is detected.
Parameters	press Event type
Return Values	None
Notes	

5.3 System

5.3.1 Application Data Storage in NVM

Data Storage is an abstraction layer that provides the functionality for storing epoch data in Flash memory using the Non-Volatile Memory Storage (NVMS) adapter of the SDK. A more detailed description of the library can be found in the user manual of the DA14681 wearable development kit [1].

Table 7: Data Storage API

Structures	
ds_sample_size_t ds_err_t	ds_data_type_t

© 2017 Dialog Semiconductor

Functions	
data_storage_init()	data_storage_start_record()
data_storage_stop_record()	data_storage_add_new_epoch()
data_storage_find_epoch_addr()	data_storage_get_samples()
data_storage_update_sample()	data_storage_erase_hist()
data_storage_handle_exception()	

5.3.1.1 Data Structures and Types

ds_sample_size_t

Type Definition Name	Description
ds_sample_size_t	Sample size enumeration type defined as: DS_SAMPLE_8_BIT = 2, DS_SAMPLE_4_BIT = 3, DS_SAMPLE_MAX = 4

ds_data_type_t

Type Definition Name	Description
ds_data_type_t	Data type enumeration type defined as: DS_HEART_RATE, DS_STEP_COUNT, DS_CALORIES_BURNED, DS_SLEEP_QUALITY, DS_DATA_TYPE_MAX

ds_err_t

Type Definition Name	Description
ds_err_t	Error type enumeration type defined as: DS_ERR_OK, DS_ERR_WRITE_NOT_STARTED, DS_ERR_TIMESTAMP_NOT_FOUND, DS_ERR_DATA_FOUND, DS_ERR_FLASH_ERROR

© 2017 Dialog Semiconductor

5.3.1.2 Functions

data_storage_init()

Function Name	`data_storage_t* data_storage_init(uint16_t heart_rate_size, uint16_t step_size, uint16_t calories_size, uint16_t sleep_size);`
Function Description	Initializes the data storage module allocating required resources and initializes the NVMS adapter.
Parameters	`heart_rate_size` Heart rate region size `step_size` Step region size `calories_size` Calories region size `sleep_size` Sleep quality region size
Return Values	Pointer to the data storage object.
Notes	This function should be called only once (at system initialization) and if flash is not erased, parameters must not change, otherwise a data corruption is to be presumed.

data_storage_start_record()

Function Name	`ds_err_t data_storage_start_record(data_storage_t *ds, uint32_t timestamp, ds_data_type_t data_type, ds_sample_size_t sample_size, uint32_t epoch_duration);`
Function Description	Creates a new chunk of data (epochs) in the data storage: finds the chronologically last chunk and updates its length field (if needed), writes a new header and frees space (if needed) by erasing or shrinking the chunks that follow it in Flash memory.
Parameters	`ds` Pointer of the data storage object `timestamp` Initial timestap (timestamp of the first epoch) `data_type` Data type of the epochs in the specific chunk `sample_size` Size in bits of each epoch (valid: 2, 4, 8, 16 and 32 bits) `epoch_duration` Duration of each epoch (constant for every sample in the chunk)
Return Values	Status of the operation
Notes	Function must be called before any data can be written into flash using data_storage_add_new_epoch().

data_storage_stop_record()

Function Name	`ds_err_t data_storage_stop_record(data_storage_t *ds, ds_data_type_t data_type);`
Function Description	Closes the selected record by writing out the number of epochs in the chunk in the chunk's header or erasing the chunk header completely, if no epoch has been added.
Parameters	`ds` Pointer of the data storage object `data_type` Data type of the epochs in the specific chunk
Return Values	Status of the operation
Notes	Can be called by the application at any time it is decided that no more epochs are going to be written in the foreseeable future, e.g. when a sensor is turned off.

© 2017 Dialog Semiconductor

data_storage_add_new_epoch()

Function Name	`ds_err_t data_storage_add_new_epoch(data_storage_t *ds, ds_data_type_t data_type, uint32_t data);`
Function Description	Adds a new epoch in the chunk. Checks if there is space available and if not, creates space by erasing or shrinking the next chunk.
Parameters	`ds` Pointer of the data storage object `data_type` Data type of the epochs in the specific chunk `data` Value of the epoch
Return Values	Status of the operation. Function fails when there is no open record.
Notes	

data_storage_find_epoch_addr()

Function Name	`ds_err_t data_storage_find_epoch_addr(data_storage_t *ds, ds_data_type_t data_type, uint32_t req_timestamp, hcs_data_chunk_header_t *header, uint32_t *address);`
Function Description	Takes a time reference (timestamp) and finds an epoch from that point onwards. Returns a calculated header with the number of epochs from the specific epoch to the chunk end.
Parameters	`ds` Pointer of the data storage object `data_type` Data type of the epoch `req_timestamp` Timestamp to search for `header` Pointer to be filled with calculated header `address` Pointer to be filled with address of the first epoch
Return Values	Status if an epoch with the provided parameters was found or not.
Notes	

data_storage_get_samples()

Function Name	`uint16_t data_storage_get_samples(data_storage_t *ds, uint8_t *data, hcs_data_chunk_header_t *header, uint32_t address, uint16 cnt);`
Function Description	Copies the referenced epochs from the specific address to the provided buffer. Uses the header for integrity checks and size calculations.
Parameters	`ds` Pointer of the data storage object `data` Pointer to put the epochs read from flash `header` Header of the chunk (as calculated by data_storage_add_new_epoch()) `address` Address of the epoch (as calculated by data_storage_add_new_epoch()) `cnt` Number of epochs to copy to the buffer
Return Values	Number of epochs copied by the function.
Notes	

© 2017 Dialog Semiconductor

data_storage_update_sample()

Function Name	`ds_err_t data_storage_update_sample(data_storage_t *ds, uint32_t timestamp, ds_data_type_t data_type, uint8_t *new_data, uint16_t cnt);`	
Function Description	Updates the values of the referenced epochs with the ones provided. Epochs must be already written once and also must be contained in a single chunk.	
Parameters	`ds`	Pointer of the data storage object
	`timestamp`	Timestamp of the first epoch to update
	`data_type`	Data type of the epochs
	`new_data`	Pointer to the buffer containing the new values
	`cnt`	Number of epochs to update
Return Values	Status of the operation.	
Notes		

data_storage_erase_hist()

Function Name	`ds_err_t data_storage_erase_hist(data_storage_t *ds, uint32_t timestamp, ds_data_type_t data_type);`	
Function Description	Finds and erases epochs of the specified data type from the provided timestamp onwards.	
Parameters	`ds`	Pointer of the data storage object
	`timestamp`	Timestamp of the first epoch to erase
	`data_type`	Data type of the epochs
Return Values	Status of the operation.	
Notes		

data_storage_handle_exception()

Function Name	`void data_storage_handle_exception(data_storage_t *ds, ds_data_type_t data_type);`	
Function Description	Handles any data storage exception (unexpected states or data) and erases the corresponding section of the data storage Flash memory when in production mode.	
Parameters	`ds`	Pointer of the data storage object
	`data_type`	Data type of the epochs
Return Values	Status of the operation.	
Notes		

© 2017 Dialog Semiconductor

5.3.2 Battery

State of Charge is a functionality of the wearable device that produces an estimation of the battery charge state as a percentage. To produce that estimation, the fuel gauge module is combined with the ADC module using a Vbat + Ibat method to calculate the remaining battery charge.

Table 8: Wearable battery API

Functions	
wrbl_battery_init() handle_charging_msg()	wrbl_battery_status_update()

5.3.2.1 Functions

wrbl_battery_init()

Function Name	`void wrbl_battery_init(void);`
Function Description	Initialize battery component of the wearable task
Parameters	None
Return Values	None
Notes	

wrbl_battery_status_update()

Function Name	`void wrbl_battery_status_update (void);`
Function Description	Updates the battery and status of the component. Function reads the current battery voltage to determine the power saving state and also reads the state of charge of the system to update the value in the display and BAS profile.
Parameters	None
Return Values	None
Notes	

handle_charging_msg()

Function Name	`void handle_charging_msg(msg *wrbl_msg);`
Function Description	Function handles messages of the wearable task about the charger state
Parameters	`wrbl_msg` Message to be handled
Return Values	None
Notes	

5.3.3 Clock manager

Table 9: Wearable clock manager API

Functions	
wrbl_clk_mngr_init() wrbl_clk_mngr_lower_clk()	wrbl_clk_mngr_raise_clk()

© 2017 Dialog Semiconductor

5.3.3.1 Functions

wrbl_clk_mngr_init()

Function Name	`void wrbl_clk_mngr_init(void);`
Function Description	Initialize wearable clock manager. Function creates required structures that need to run.
Parameters	None
Return Values	None
Notes	

wrbl_clk_mngr_raise_clk()

Function Name	`bool wrbl_clk_mngr_raise_clk(void);`
Function Description	Request wearable clock to increase frequency. Function checks if already called and set frequency to high value (96MHz).
Parameters	None
Return Values	True if the request had any effect or false if the clock was already high.
Notes	

wrbl_clk_mngr_lower_clk()

Function Name	`bool wrbl_clk_mngr_lower_clk(void);`
Function Description	Request wearable clock to decrease frequency. Function checks if no other request is active and restores frequency to initial value.
Parameters	None
Return Values	True if the request had any effect or false if the clock was already low.
Notes	

5.3.4 Real Time Clock

Table 10: Wearable RTC API

Structures	
rtc_callbacks_t	
Functions	
wrbl_rtc_init()	handle_rtc()
Callback Functions	
minute_evt	alarm_evt

5.3.4.1 Data Structures and Types

rtc_callbacks_t

Field	Type	Description
`minute_evt`	`rtc_event_callback`	Minute change event callback function
`alarm_evt`	`rtc_event_callback`	Alarm event callback function

 © 2017 Dialog Semiconductor

5.3.4.2 Functions

wrbl_rtc_init()

Function Name	**`void wrbl_rtc_init(const rtc_callbacks_t *cb);`**
Function Description	Initialize the RTC module.
Parameters	`cb` Callback function to be called when an RTC event is detected
Return Values	None
Notes	

handle_rtc()

Function Name	**`void handle_rtc(void);`**
Function Description	Function handles the RTC events of the wearable task.
Parameters	None
Return Values	None
Notes	

5.3.4.3 Callback Functions

minute_evt

Function Name	`void minute_evt(void);`
Function Description	Function is called at each minute change to indicate the event.
Parameters	None
Return Values	None
Notes	

alarm_evt

Function Name	`void alarm_evt(void);`
Function Description	Function is called when the alarm event is triggered.
Parameters	None
Return Values	None
Notes	

5.4 Wearable Health Toolbox Control

The Wearable Health Toolbox Control is an abstraction layer that provides easy control to health toolbox functionality used by the wearable task and also combines the requirements of the various interfaces that need access to the health toolbox resources. It operates as a bridge between the DWS, HCS and UI control entry points and the health toolbox task.

 © 2017 Dialog Semiconductor

Table 11: Wearable Health Toolbox Control API

Data Structures and Types	
srvc_iface_t	srvc_sens_t
ht_conf_type_t	ht_ctrl_state_t
sens_config_t	srvc_state_t
acc_values_t	epoch_timestamps
Functions	
wrbl_ht_ctrl_init()	wrbl_ht_ctrl_stop()
wrbl_ht_ctrl_get_sens_state()	wrbl_ht_ctrl_get_iface_state()
wrbl_ht_ctrl_config_sens()	wrbl_ht_ctrl_switch_sens()
wrbl_ht_ctrl_switch_iface()	wrbl_ht_ctrl_set_init()
wrbl_ht_ctrl_reset_sens()	wrbl_ht_ctrl_config_basic_attrs()
wrbl_ht_ctrl_config_sfl_coeffs()	wrbl_ht_ctrl_config_calibr_coeffs()
wrbl_ht_ctrl_config_calibr_control()	wrbl_ht_ctrl_get_config()
wrbl_ht_ctrl_reset_conf()	wrbl_ht_ctrl_reset_cal_sfl_conf()
wrbl_ht_ctrl_set_user_profile()	wrbl_ht_ctrl_get_user_profile()
handle_ht_config()	handle_ht_data()
wrbl_ht_ctrl_store_conf()	wrbl_ht_ctrl_store_clbr()
wrbl_ht_ctrl_read_nv()	

5.4.1 Data Structures and Types

srvc_iface_t

Type Definition Name	Description

`srvc_iface_t`	Service interface enumeration type defined as:
	`SRVC_IFACE_DWS,`
	`SRVC_IFACE_HCS,`
	`SRVC_IFACE_MAX`

srvc_sens_t

Type Definition Name	Description

`srvc_sens_t`	Service/sensor enumeration type defined as:
	`SRVC_SENS_ACC,`
	`SRVC_SENS_GYR,`
	`SRVC_SENS_MAG,`
	`SRVC_SENS_SF,`
	`SRVC_SENS_ENV,`
	`SRVC_SENS_SRVC_START,`
	`SRVC_SENS_SENS_END = SRVC_SENS_SRVC_START - 1,`
	`SRVC_SENS_STEP,`
	`SRVC_SENS_HRM,`
	`SRVC_SENS_SQ,`
	`SRVC_SENS_CAL,`
	`SRVC_SENS_BSC,`
	`SRVC_SENS_MAX`

© 2017 Dialog Semiconductor

ht_conf_type_t

Type Definition Name	Description
ht_conf_type_t	Configuration type enumeration type defined as: CONF_START_HT, CONF_STOP_HT, CONF_BASIC_SETUP, CONF_CONFIG_DONE, CONF_RESET_DONE, CONF_SFL_CLBR_COEFFS, CONF_MAG_CLBR_CTRL, CONF_MAG_CLBR_CTRL_STAT, CONF_MAG_CLBR_CTRL_BASIC, CONF_MAG_CLBR_CTRL_SMART, CONF_MAG_CLBR_CTRL_CMPLT, CONF_MAG_CLBR_COEFFS, CONF_MAG_CLBR_COEFFS_STAT, CONF_MAG_CLBR_COEFFS_BASIC, CONF_MAG_CLBR_COEFFS_SMART, CONF_MAG_CLBR_COEFFS_CMPLT, CONF_ACC_CLBR, CONF_USER_PROFILE, CONF_STEP_SETTING, CONF_HRM_SETTING, CONF_SQ_SETTING , CONF_CAL_SETTING, CONF_BSC_SETTING, CONF_MAX

ht_ctrl_state_t

Type Definition Name	Description
ht_ctrl_state_t	Service/sensor control state enumeration type defined as: SRVC_SUSPEND = 0x00, SRVC_RESUME = 0x01, SRVC_DISABLE = 0x02, SRVC_ENABLE = 0x03

sens_config_t

Field	Type	Description
acc	ht_acc_config_t	Accelerometer configuration
gyr	ht_gyr_config_t	Gyroscope configuration
mag	ht_mag_config_t	Magnetometer configuration
env	ht_env_config_t	Environmental configuration
step	ht_step_config_t	Step counting configuration
sf		Sensor fusion library configuration
hr	ht_hr_config_t	Heart rate configuration
sq	ht_sq_config_t	Sleep quality configuration
cal	ht_cc_config_t	Calories counting configuration
bsc	ht_bs_config_t	Body state classification configuration

© 2017 Dialog Semiconductor

srvc_state_t

Field	Type	Description
config	sens_config_t	Corresponding sensor/service configuration
ticks	uint8_t	Samples to drop in order to achieve required rate

acc_values_t

Field	Type	Description
step	uint32_t	Steps accumulated value
cal	uint32_t	Calories accumulated value
sq_l	uint32_t	Sleep quality light accumulated value
sq_d	uint32_t	Sleep quality deep accumulated value
sq_r	uint32_t	Sleep quality REM accumulated value

epoch_timestamps

Field	Type	Description
step	uint32_t	Epoch timestamp of the steps
hr	uint32_t	Epoch timestamp of the heart rate
cal	uint32_t	Epoch timestamp of the calories counting
sq	uint32_t	Epoch timestamp of the sleep quality
bsc	uint32_t	Epoch timestamp of the body state classification

© 2017 Dialog Semiconductor

5.4.2 Functions

wrbl_ht_ctrl_init()

Function Name	`void wrbl_ht_ctrl_init(void);`
Function Description	Stores DWS service handler and starts the Health Toolbox.
Parameters	None
Return Values	None
Notes	

wrbl_ht_ctrl_stop()

Function Name	`void wrbl_ht_ctrl_stop(void);`
Function Description	Stop Health Toolbox
Parameters	None
Return Values	None
Notes	

wrbl_ht_ctrl_get_sens_state()

Function Name	`bool wrbl_ht_ctrl_get_sens_state(srvc_iface_t iface, srvc_sens_t sens);`
Function Description	Returns the state of the corresponding sensor/service of the specific interface.
Parameters	`iface` Interface `sens` Sensor/service
Return Values	True if sensor/service is configured as on, false otherwise.
Notes	

wrbl_ht_ctrl_get_iface_state()

Function Name	`bool wrbl_ht_ctrl_get_iface_state(srvc_iface_t iface);`
Function Description	Returns the state of the corresponding sensor/service of the specific interface.
Parameters	`iface` Interface
Return Values	True if sensor/service is configured as on, false otherwise.
Notes	

wrbl_ht_ctrl_config_sens()

Function Name	`void wrbl_ht_ctrl_config_sens(srvc_iface_t iface, srvc_sens_t sens, sens_config_t *config);`
Function Description	Stores the configuration and if interface is enabled it configures the referenced sensor.
Parameters	`iface` Interface `sens` Sensor/service `config` Configuration to be applied
Return Values	None
Notes	

wrbl_ht_ctrl_switch_sens()

© 2017 Dialog Semiconductor

Function Name	`void wrbl_ht_ctrl_switch_sens(srvc_iface_t iface, srvc_sens_t sens, ht_ctrl_state_t sens_state);`
Function Description	Enables or disables the referenced sensor/service without changing its settings. It stores the new state to be restored at startup when interface is enabled/disabled.
Parameters	`iface` Interface `sens` Sensor/service `sens_state` State of the sensor/service to be applied
Return Values	None
Notes	

wrbl_ht_ctrl_switch_iface()

Function Name	`void wrbl_ht_ctrl_switch_iface (srvc_iface_t iface, bool iface_state);`
Function Description	Enables or disables the referenced interface and stores previous state to reapply it when turned on.
Parameters	`iface` Interface `iface_state` State of the interface to be applied
Return Values	None
Notes	

wrbl_ht_ctrl_set_init()

Function Name	`void wrbl_ht_ctrl_set_init(srvc_sens_t sens);`
Function Description	Set data of the provided sensor/service to the initial value
Parameters	`sens` Sensor/service
Return Values	None

Notes	Valid sensors/services:
	`SRVC_SENS_STEP`
	`SRVC_SENS_SQ`
	`SRVC_SENS_CAL`

wrbl_ht_ctrl_reset_sens()

Function Name	`void wrbl_ht_ctrl_reset_sens(srvc_sens_t sens);`
Function Description	Reset data of the provided sensor/service
Parameters	`sens` Sensor/service
Return Values	None

Notes	Valid sensors/services:
	`SRVC_SENS_STEP`
	`SRVC_SENS_SQ`
	`SRVC_SENS_CAL`

wrbl_ht_ctrl_config_basic_attrs()

Function Name	`void wrbl_ht_ctrl_config_basic_attrs(iot_basic_setup_t *iot_basic_setup);`
Function Description	Configures the BLE interface configuration applied.
Parameters	`iot_basic_setup` Basic attributes configuration
Return Values	None

© 2017 Dialog Semiconductor

Notes	

wrbl_ht_ctrl_config_sfl_coeffs()

Function Name	**void wrbl_ht_ctrl_config_sfl_coeffs(iot_sfl_coeffs_t *iot_sfl_coeffs);**
Function Description	Configures the BLE interface configuration applied.
Parameters	iot_sfl_coeffs SFL coefficients
Return Values	None
Notes	

wrbl_ht_ctrl_config_calibr_coeffs()

Function Name	**void wrbl_ht_ctrl_config_calibr_coeffs(iot_calibr_coeffs_t *iot_calibr_coeffs);**
Function Description	Configures the BLE interface configuration applied.
Parameters	iot_calibr_coeffs Calibration coefficients
Return Values	None
Notes	

wrbl_ht_ctrl_config_calibr_control()

Function Name	**void wrbl_ht_ctrl_config_calibr_control(iot_calibr_ctrl_t *iot_calibr_ctrl);**
Function Description	Configures the BLE interface configuration applied.
Parameters	iot_calibr_ctrl Calibration control
Return Values	None
Notes	

wrbl_ht_ctrl_get_config()

Function Name	void wrbl_ht_ctrl_get_config(ht_conf_type_t conf);
Function Description	Calls the appropriate get configuration from the Health Toolbox in order to be sent to peer device with the DWS profile.
Parameters	conf Reference of configuration type
Return Values	None
Notes	

wrbl_ht_ctrl_reset_conf()

Function Name	**void wrbl_ht_ctrl_reset_conf(void);**
Function Description	Resets the BLE interface sensors/services settings to the default ones.
Parameters	None
Return Values	None
Notes	

wrbl_ht_ctrl_reset_cal_sfl_conf()

Function Name	**void wrbl_ht_ctrl_reset_cal_sfl_conf(void);**

© 2017 Dialog Semiconductor

Function Description	Resets the BLE iface sensors/services calibration settings to the default ones
Parameters	None
Return Values	None
Notes	

wrbl_ht_ctrl_set_user_profile()

Function Name	**void wrbl_ht_ctrl_set_user_profile(hcs_user_profile_t *user_profile);**
Function Description	Configures the HCS user profile and configures the Health Toolbox services.
Parameters	user_profile Reference to user profile
Return Values	None
Notes	

wrbl_ht_ctrl_get_user_profile()

Function Name	**void wrbl_ht_ctrl_get_user_profile(hcs_user_profile_t *user_profile);**
Function Description	Returns the configuration of the user profile stored in Flash memory if available or default values otherwise.
Parameters	user_profile Reference to user profile
Return Values	None
Notes	

handle_ht_config()

Function Name	**bool handle_ht_config(msg *wrbl_msg);**
Function Description	Handles the reception of configurations by the Health Toolbox and performs the appropriate actions.
Parameters	wrbl_msg The configuration message
Return Values	True if message type was handled, false otherwise.
Notes	

handle_ht_data()

Function Name	**bool handle_ht_data(msg *wrbl_msg);**
Function Description	Function handles the reception of data by the Health Toolbox and performs the appropriate actions.
Parameters	wrbl_msg The data message
Return Values	True if message type was handled, false otherwise.
Notes	

wrbl_ht_ctrl_store_conf()

Function Name	**void wrbl_ht_ctrl_store_conf(void);**
Function Description	Stores the current configuration of the Health Toolbox to Flash memory.
Parameters	None
Return Values	None
Notes	

© 2017 Dialog Semiconductor

wrbl_ht_ctrl_store_clbr()

Function Name	`void wrbl_ht_ctrl_store_clbr(void);`
Function Description	Sores the current calibrations of the Health Toolbox to Flash memory.
Parameters	None
Return Values	None
Notes	

wrbl_ht_ctrl_read_nv()

Function Name	`void wrbl_ht_ctrl_read_nv(void);`
Function Description	Retrieves the configurations/calibrations of the Health Toolbox from Flash memory if available. If not available, it sets the default ones.
Parameters	None
Return Values	None
Notes	

6 BLE Services

This section presents the standard and custom BLE services used by the Wearable application.

6.1 Dialog Wearable Service

The Dialog Wearable Service (DWS) is a Dialog proprietary GATT service providing a means to:

- Transfer raw and calibrated sensor data.
- Transfer sensor fusion data.
- Configure the device, such as setting of operating parameters.
- Control the device, such as start/stop and load/store to non-volatile memory.

The DWS API specification structures and types are divided into two parts: the strictly DWS profile (file `dws.h`, described in Section 6.1.1) and the IoT extension definitions (file `IoT.h`, described in Section 6.1.2).

6.1.1 DWS Data Structures and Types

dws_char_ref_t

Type Definition Name	Description

`dws_char_ref_t`	DWS reference characteristic enumeration type defined as:
	`DWS_ACCEL_CHAR,`
	`DWS_GYRO_CHAR,`
	`DWS_MAG_CHAR,`
	`DWS_BARO_CHAR,`
	`DWS_HUM_CHAR,`
	`DWS_TEMP_CHAR,`
	`DWS_SENSF_CHAR,`
	`DWS_FEAT_CHAR,`
	`DWS_CONTROL_CHAR,`
	`DWS_CONTROL_REPLY_CHAR`

© 2017 Dialog Semiconductor

dws_features_t

Field	Type	Description
accelerometer_en	uint8_t	Determines active characteristics. 0: disabled >0: enabled
gyro_en	uint8_t	
magn_en	uint8_t	
pressure_en	uint8_t	
humidity_en	uint8_t	
temp_en	uint8_t	
s_fusion_en	uint8_t	
version[16]	uint8_t	SW version, in BCD format.
device_type	uint8_t	Device type. 0: Wearable 58x 1: Wearable 68x

6.1.2 IoT Data Structures and Types

iot_control_byte_t

Type Definition Name	Description
iot_control_byte_t	IoT control command byte enumeration type defined as: IOT_STOP_CMD = 0, IOT_START_CMD = 1, IOT_READ_NV_CMD = 2, IOT_RESET_CMD = 3, IOT_STORE_CONF_CMD = 4, IOT_STORE_CLBR_CMD = 5, IOT_GET_RUNNING_STATUS_CMD = 6, IOT_RESET_CAL_SF_DATA_CMD = 7, IOT_BASIC_CONFIG_WR_CMD = 10, IOT_BASIC_CONFIG_RD_REQ = 11, IOT_SFL_COEFF_WR_CMD = 12, IOT_SFL_COEFF_RD_REQ = 13, IOT_CAL_COEFF_WR_CMD = 14, IOT_CAL_COEFF_RD_REQ = 15, IOT_CAL_CTRL_WR_CMD = 16, IOT_CAL_CTRL_RD_REQ = 17, IOT_START_ACC_ALLIGN = 18

iot_report_id_t

Type Definition Name	Description
iot_report_id_t	IoT report ID enumeration type defined as: IOT_ACCELEROMETER_REPORT_ID = 1, IOT_GYROSCOPE_REPORT_ID = 2, IOT_MAGNETOMETER_REPORT_ID = 3, IOT_PRESSURE_REPORT_ID = 4, IOT_HUMIDITY_REPORT_ID = 5, IOT_TEMPERATURE_REPORT_ID = 6, IOT_SENSOR_FUSION_REPORT_ID = 7, IOT_COMMAND_REPLY_REPORT_ID = 8

© 2017 Dialog Semiconductor

iot_sensor_type_t

Type Definition Name	Description
iot_sensor_type_t	IoT sensor type enumeration type defined as: IOT_SENSOR_ACC, IOT_SENSOR_GYR, IOT_SENSOR_MAG, IOT_SENSOR_MAX

iot_flags_t

Type Definition Name	Description
iot_flags_t	IoT sensor enable/combination enumeration type defined as: IOT_ENABLE_ACC_FLAG = (1 << 0), IOT_ENABLE_GYR_FLAG = (1 << 1), IOT_ENABLE_MAG_FLAG = (1 << 2), IOT_ENABLE_ENV_FLAG = (1 << 3)

iot_sfl_comb_t

Type Definition Name	Description
iot_sfl_comb_t	IoT SFL sensor combination enumeration type defined as: IOT_SFL_COMB_G = IOT_ENABLE_GYR_FLAG, IOT_SFL_COMB_GA = IOT_ENABLE_ACC_FLAG | IOT_ENABLE_GYR_FLAG, IOT_SFL_COMB_AM = IOT_ENABLE_ACC_FLAG | IOT_ENABLE_MAG_FLAG, IOT_SFL_COMB_GAM = IOT_ENABLE_ACC_FLAG | IOT_ENABLE_GYR_FLAG | IOT_ENABLE_MAG_FLAG

iot_bmi160_accel_range_t

Type Definition Name	Description
iot_bmi160_accel_range_t	IoT BMI160 accelerometer range enumeration type defined as: IOT_BMI160_ACCEL_RANGE_2G = 0X03, IOT_BMI160_ACCEL_RANGE_4G = 0X05, IOT_BMI160_ACCEL_RANGE_8G = 0X08, IOT_BMI160_ACCEL_RANGE_16G = 0X0C

iot_bmi160_accel_rate_t

Type Definition Name	Description
iot_bmi160_accel_rate_t	IoT BMI160 accelerometer rate enumeration type defined as: IOT_BMI160_ACCEL_RATE_0_78HZ = 0x01, IOT_BMI160_ACCEL_RATE_1_56HZ = 0x02, IOT_BMI160_ACCEL_RATE_3_12HZ = 0x03, IOT_BMI160_ACCEL_RATE_6_25HZ = 0x04, IOT_BMI160_ACCEL_RATE_12_5HZ = 0x05, IOT_BMI160_ACCEL_RATE_25HZ = 0x06, IOT_BMI160_ACCEL_RATE_50HZ = 0x07, IOT_BMI160_ACCEL_RATE_100HZ = 0x08

© 2017 Dialog Semiconductor

iot_bmi160_gyro_range_t

Type Definition Name	Description
iot_bmi160_gyro_range_t	IoT BMI160 gyroscope range enumeration type defined as: IOT_BMI160_GYRO_RANGE_2000_DEG_SEC = 0x00, IOT_BMI160_GYRO_RANGE_1000_DEG_SEC = 0x01, IOT_BMI160_GYRO_RANGE_500_DEG_SEC = 0x02, IOT_BMI160_GYRO_RANGE_250_DEG_SEC = 0x03

iot_bmi160_gyro_rate_t

Type Definition Name	Description
iot_bmi160_gyro_rate_t	IoT BMI160 gyroscope rate enumeration type defined as: IOT_BMI160_GYRO_RATE_25HZ = 0x06, IOT_BMI160_GYRO_RATE_50HZ = 0x07, IOT_BMI160_GYRO_RATE_100HZ = 0x08

iot_bmm150_mag_rate_t

Type Definition Name	Description
iot_bmm150_mag_rate_t	IoT BMM150 magneto rate enumeration type defined as: IOT_BMM150_MAG_RATE_0_78HZ = 0x01, IOT_BMM150_MAG_RATE_1_56HZ = 0x02, IOT_BMM150_MAG_RATE_3_12HZ = 0x03, IOT_BMM150_MAG_RATE_6_25HZ = 0x04, IOT_BMM150_MAG_RATE_12_5HZ = 0x05, IOT_BMM150_MAG_RATE_25HZ = 0x06, IOT_BMM150_MAG_RATE_50HZ = 0x07, IOT_BMM150_MAG_RATE_100HZ = 0x08

iot_env_rate_t

Type Definition Name	Description
iot_env_rate_t	IoT environmental rate enumeration type defined as: IOT_ENV_RATE_0_5_HZ = 1, IOT_ENV_RATE_1HZ = 2, IOT_ENV_RATE_2HZ = 4

iot_sfl_rate_t

Type Definition Name	Description
iot_sfl_rate_t	IoT SFL report rate enumeration type defined as: IOT_SFL_REPORT_RATE_OFF = 0x00, IOT_SFL_REPORT_RATE_0_78 = 0x01, IOT_SFL_REPORT_RATE_1_56 = 0x02, IOT_SFL_REPORT_RATE_3_12 = 0x03, IOT_SFL_REPORT_RATE_6_25 = 0x04, IOT_SFL_REPORT_RATE_12_5 = 0x05, IOT_SFL_REPORT_RATE_25 = 0x06, IOT_SFL_REPORT_RATE_50 = 0x07

© 2017 Dialog Semiconductor

iot_cal_mode_t

Type Definition Name	Description
iot_cal_mode_t	IoT calibration mode enumeration type defined as: IOT_CAL_MODE_NONE = 0, IOT_CAL_MODE_STATIC = 1, IOT_CAL_MODE_AUTO_CONT = 2, IOT_CAL_MODE_AUTO_ONE_SHOT = 3

iot_auto_cal_mode_t

Type Definition Name	Description
iot_auto_cal_mode_t	IoT auto calibration mode enumeration type defined as: IOT_AUTO_CAL_BASIC = 0, IOT_AUTO_CAL_SMART = 1

iot_cal_state_t

Type Definition Name	Description
iot_cal_state_t	IoT calibration state/status enumeration type defined as: IOT_CAL_STATE_DISABLED = 0, IOT_CAL_STATE_INIT = 1, IOT_CAL_STATE_BAD = 2, IOT_CAL_STATE_OK = 3, IOT_CAL_STATE_GOOD = 4, IOT_CAL_STATE_ERROR = 5

iot_sfl_raw_t

Type Definition Name	Description
iot_sfl_raw_t	IoT SFL raw data state enumeration type defined as: IOT_SFL_RAW_EN = 0, IOT_SFL_RAW_DIS = 1

iot_basic_setup_t

Field	Type	Description
sensor_combination	uint8_t	Active sensor combination (iot_flags_t or iot_sfl_comb_t)
acc_rng	uint8_t	Accelerometer range (as in iot_bmi160_accel_range_t)
acc_rate	uint8_t	Accelerometer rate (as in iot_bmi160_accel_rate_t)
gyr_rng	uint8_t	Gyroscope range (as in iot_bmi160_gyro_range_t)
gyr_rate	uint8_t	Gyroscope rate (as in iot_bmi160_gyro_rate_t)
mag_rate	uint8_t	Magneto rate (as in iot_bmm150_mag_rate_t)
env_rate	uint8_t	Environmental sensors rate (as in iot_env_rate_t)
sfl_rate	uint8_t	Sensor fusion library report rate (as in iot_sfl_rate_t)
sfl_raw_en	uint8_t	Sensor fusion library raw enable (as in iot_sfl_raw_t)
cal_mode	uint8_t	Calibration mode (as in iot_cal_mode_t)
auto_cal_mode	uint8_t	Auto calibration mode (as in iot_auto_cal_mode_t)

© 2017 Dialog Semiconductor

iot_sfl_coeffs_t

Field	Type	Description
beta_a	uint16_t	Beta A SFL coefficient
beta_m	uint16_t	Beta M SFL coefficient
reserved[4]	uint8_t	Reserved

iot_calibr_coeffs_t

Field	Type	Description
sensor_type	uint8_t	Sensor type (as in iot_sensor_type_t)
q_format	uint8_t	Q format of matrix
offset_vector[3]	int16_t	Offset vectors for raw sensor data
matrix[3][3]	int16_t	Matrix of coefficients to apply to raw sensor data

iot_calibr_ctrl_t

Field	Type	Description
sensor_type	uint8_t	Sensor type (as in iot_sensor_type_t)
flags	uint16_t	Calibration control flags (as defined in Table 12 and Table 13)
ref_mag	uint16_t	Reference magnitude
mag_range	uint16_t	Magnitude range
mag_alpha	uint16_t	Magnitude filter coefficient
mag_delta_thresh	uint16_t	Magnitude gradient threshold
mu_offset	uint16_t	Offset update rate
mu_matrix	uint16_t	Matrix update rate
err_alpha	uint16_t	Overall error filter coefficient
err_thresh	uint16_t	Overall error threshold

Table 12: Calibration Control Flags (byte 1)

Calibration Mode	Bit 0 Reserved	Bit 1 Reserved	Bit 2 Offset Apply	Bit 3 Matrix Apply	Bit 4 Offset Update	Bit 5 Matrix Update	Bit 7 Init. from Static	Bit 7 Offset Post Apply
Static	X	X	1: Yes	1: Yes	0: No	0: No	0: No	1: Yes
Basic Auto	X	X	1: Yes	1: Yes	1: Yes	1: Yes	0: No	1: Yes
Smart Fusion Auto	X	X	1: Yes	1: Yes	1: Yes	1: Yes	1: Yes	1: Yes

Table 13: Calibration control flags (byte 2)

Calibration mode	Bit 0 Reserved	Bit 1 Reserved	Bit 2 Reserved	Bit 3 Reserved	Bit 4 Reserved	Bit 5 Reserved	Bit 6 Converged (read only)	Bit 7 Settled (read only)
Static	X	X	X	X	X	X	X	0: No
Basic Auto	X	X	X	X	X	X	0: No	1: Yes
Smart Fusion Auto	X	X	X	X	X	X	1: Yes	1: Yes

© 2017 Dialog Semiconductor

iot_ctrl_params_t

Field	Type	Description
sensor_status	uint8_t	Sensor state status
basic_setup	iot_basic_setup_t	Basic setup of the sensors
sfl_coefficients	iot_sfl_coeffs_t	Smart fusion library coefficients
calibr_coeffs	iot_calibr_coeffs_t	Calibration coefficients
calibr_ctrl	iot_calibr_ctrl_t	Calibration control configuration

iot_control_t

Field	Type	Description
ctrl_cmd	uint8_t	Control command
ctrl_params	iot_ctrl_params_t	Control command parameters of the corresponding command type

iot_control_reply_t

Field	Type	Description
ucReportId	uint8_t	Report ID type (as in iot_report_id_t)
ctrl_cmd	uint8_t	Control command
ctrl_params	iot_ctrl_params_t	Control command parameters of the corresponding command type

iot_snsr_state_t

Field	Type	Description
in_data_valid	bool:1	Input (pre calibration) data valid flag
out_data_valid	bool:1	Output (post calibration) data valid flag
cal_enabled	bool:1	Calibration enabled flag
cal_settled	bool:1	Calibration settled flag
cal_converged	bool:1	Calibration converged flag
cal_mode	bool:3	Calibration mode

iot_data_report_agm_t

Field	Type	Description
ucReportId	uint8_t	Report ID type (as in iot_report_id_t)
ucSensorState	iot_snsr_state_t	Sensor state
ucCalState	uint8_t	Calibration state (as in iot_cal_state_t)
val_x	int16_t	Sensor axis values of accelerometer/gyroscope/magneto
val_y	int16_t	
val_z	int16_t	

iot_data_report_pht_t

Field	Type	Description
ucReportId	uint8_t	Report ID type (as in iot_report_id_t)

© 2017 Dialog Semiconductor

Field	Type	Description
ucSensorState	uint8_t	Sensor state
ucSensorEvent	uint8_t	Sensor event (always 3, reserved FFU)
val32	int32_t	Sensor value of pressure/humidity/temperature

iot_data_report_sf_t

Field	Type	Description
ucReportId	uint8_t	Report ID type (as in iot_report_id_t)
ucSensorState	uint8_t	Sensor state
ucCalState	uint8_t	Calibration state
val_w	int16_t	Sensor fusion library values
val_x	int16_t	
val_y	int16_t	
val_z	int16_t	

6.1.3 Functions

dws_init()

Function Name	ble_service_t *dws_init(const dws_callbacks_t * const cb, const dws_features_t * const dws_features);	
Function Description	Registers Dialog wearable service instance.	
Parameters	cb	Application callbacks
	dws_features	Application supported features
Return Values	Service instance	
Notes		

dws_notify_value()

Function Name	void dws_notify_value(ble_service_t *svc, uint16_t conn_idx, dws_char_ref_t dws_char, uint8_t *data, uint16_t length);	
Function Description	Sends notification to peer device.	
Parameters	svc	Instance of service
	conn_idx	Connection index
	dws_char	Reference to characteristic
	data	Data to be notified
	length	Length of data in bytes
Return Values	None	
Notes	Upon transmission of data, service will call dws_ntf_cfm callback.	

© 2017 Dialog Semiconductor

6.1.4 Callback Functions

dws_control_val_wr_valid()

Function Name	`att_error_t dws_control_val_wr_valid(ble_service_t *svc, uint16_t conn_idx, const uint8_t *msg, uint16_t length);`	
Function Description	Callback function to be called when a write control value is received in order to check validity of the command.	
Parameters	`svc`	Service instance
	`conn_idx`	Connection index
	`msg`	Command message
	`length`	Length of command message in bytes
Return Values	Status of validity check.	
Notes		

dws_control_ind()

Function Name	`void dws_control_ind(ble_service_t *svc, uint16_t conn_idx, const uint8_t *msg, uint16_t length);`	
Function Description	Callback function to be called when a write control value is received	
Parameters	`svc`	Service instance
	`conn_idx`	Connection index
	`msg`	Command message
	`length`	Length of command message in bytes
Return Values	None	
Notes	It is called only after validation.	

dws_notify_cfm()

Function Name	`void dws_notify_cfm(ble_service_t *svc, uint16_t conn_idx);`	
Function Description	Callback function to be called when a notification confirmation is received by BLE stack.	
Parameters	`svc`	Service instance
	`conn_idx`	Connection index
Return Values	None	
Notes		

© 2017 Dialog Semiconductor

6.2 Health Care Service

The Health Care Service (HCS) is a Dialog proprietary GATT service used to control, configure and reading output data from Health Care services running on the Wearable DK. By supporting the GATT client role a remote BLE central connected to the Wearable DK can perform the following operations:

- Enable/disable a Health Care service.
- Get/reset the current day's live counters of the Health Care services.
- Get updates of the latest epoch values for all services, while it is connected to the Wearable DK.
- Retrieve Health Care data stored in the Wearable DK Flash memory.
- Set the user's biometric information in the Wearable DK.

6.2.1 Data Structures and Types

hcs_features_t

Type Definition Name	Description
hcs_features_t	HCS supported features enumeration type defined as: HCS_FEAT_STEP = 1 << 0, HCS_FEAT_HEART_RATE = 1 << 1, HCS_FEAT_SLEEP_QUAL = 1 << 2, HCS_FEAT_CALORIES = 1 << 3, HCS_FEAT_BSC = 1 << 4

hcs_sq_state_t

Type Definition Name	Description
hcs_sq_state_t	HCS sleep quality state enumeration type defined as: HCS_SLEEP_AWAKE, HCS_SLEEP_LIGHT, HCS_SLEEP_DEEP, HCS_SLEEP_REM, HCS_SLEEP_NOT_ATTACHED, HCS_SLEEP_ERROR

hcs_bsc_state_t

Type Definition Name	Description
hcs_bsc_state_t	HCS body state classification enumeration type defined as: HCS_BSC_STATE_OTHER, HCS_BSC_STATE_WALK, HCS_BSC_STATE_RUN, HCS_BSC_STATE_SIT, HCS_BSC_STATE_STAND, HCS_BSC_STATE_LAY_DOWN

© 2017 Dialog Semiconductor

hcs_ctrl_cmd_t

Type Definition Name	Description
hcs_ctrl_cmd_t	HCS control command enumeration type defined as: HCS_CTRL_STOP, HCS_CTRL_START, HCS_CTRL_RESET, HCS_HEART_RATE_MODE_ONE_SHOT, HCS_HEART_RATE_MODE_CONTINOUS, HCS_CALORIES_ACT_WALK_RUN, HCS_CALORIES_ACT_CYCLING, HCS_READ_HISTORICAL_DATA, HCS_READ_OPERATION_STATE, HCS_READ_COUNTER

hcs_op_code_t

Type Definition Name	Description
hcs_op_code_t	HCS operation code enumeration type defined as: HCS_OP_CODE_STEP = 0x01, HCS_OP_CODE_HEART_RATE, HCS_OP_CODE_SLEEP_QUALITY, HCS_OP_CODE_CALORIES, HCS_OP_CODE_BODY_STATE, HCS_OP_CODE_LIGHT_SLEEP = 0xE0, HCS_OP_CODE_DEEP_SLEEP, HCS_OP_CODE_REM_SLEEP, HCS_OP_CODE_CALORIES_WALKING = 0xF0, HCS_OP_CODE_CALORIES_CYCLING

hcs_sample_size_t

Type Definition Name	Description
hcs_sample_size_t	HCS sample size enumeration type defined as: HCS_SAMPLE_2_BITS = 0x01, HCS_SAMPLE_4_BITS, HCS_SAMPLE_8_BITS, HCS_SAMPLE_16_BITS

hcs_status_t

Type Definition Name	Description
hcs_status_t	HCS operation status enumeration type defined as: HCS_STAT_SUCCESS, HCS_STAT_UNSUPPORTED_CMD, HCS_STAT_MISSING_DATA, HCS_STAT_INVALID_OP_CODE, HCS_STAT_TIMESTAMP_NOT_FOUND, HCS_STAT_CHUNK_HEADER, HCS_STAT_OPERATION_STATE

hcs_operation_state_t

Type Definition Name	Description
hcs_operation_state_t	HCS sensor operation state enumeration type defined as: HCS_OPERATION_INACTIVE = 0x00, HCS_OPERATION_ACTIVE

© 2017 Dialog Semiconductor

hcs_calories_mode_t

Type Definition Name	Description
hcs_calories_mode_t	HCS burned calories mode enumeration type defined as: HCS_CALORIES_WALKING = 0xF0, HCS_CALORIES_CYCLING

hcs_heart_rate_mode_t

Type Definition Name	Description
hcs_heart_rate_mode_t	HCS heart rate mode enumeration type defined as: HCS_HR_ONE_SHOT = 0x00, HCS_HR_CONTIN

hcs_operation_status_t

Field	Type	Description
op_state	hcs_operation_state_t	Sensor operation state
cal_mode	hcs_calories_mode_t	Calories burned mode
hr_mode	hcs_heart_rate_mode_t	Heart rate mode

hcs_user_profile_t

Field	Type	Description
age	uint8_t	Profile age
height	uint8_t	Profile height in cm
weight	uint8_t	Profile weight in kg
sex	uint8_t	Profile gender. Use 'M' for male (ASCII 0x4D) or 'F' for female (ASCII 0x46).

hcs_step_t

Field	Type	Description
timestamp	uint32_t	Step counting epoch timestamp in UNIX format
steps	uint32_t	Step counting epoch value

hcs_heart_rate_t

Field	Type	Description
timestamp	uint32_t	Heart rate epoch timestamp
rate	uint16_t	Heart rate value

hcs_calories_t

Field	Type	Description
timestamp	uint32_t	Calories burned epoch timestamp
value	uint8_t	Calories burned epoch value
mode	uint8_t	Calories burned epoch mode

© 2017 Dialog Semiconductor

hcs_sleep_quality_t

Field	Type	Description
timestamp	uint32_t	Sleep quality epoch timestamp
state	hcs_sq_state_t:8	Sleep quality epoch state

hcs_data_chunk_header_t

Field	Type	Description
timestamp	uint32_t	Historical data chunk header timestamp
epochs_cnt	uint16_t	Historical data chunk header epochs found
sample_size	uint8_t	Historical data chunk header sample size
data_type	uint8_t	Historical data chunk header data type
epoch_duration	uint32_t	Historical data chunk header epoch duration in ms

hcs_initial_values_t

Field	Type	Description
user_profile	hcs_user_profile_t	Initial user profile
steps	hcs_step_t	Initial steps epoch
heart_rate	hcs_heart_rate_t	Initial heart rate epoch
calories	hcs_calories_t	Initial calories burned epoch
sleep_quality	hcs_sleep_quality_t	Initial sleep quality epoch
body_state	hcs_bsc_state_t	Initial body state classification value

© 2017 Dialog Semiconductor

6.2.2 Functions

hcs_init()

Function Name	`ble_service_t *hcs_init(const hcs_callbacks_t * const cb, const hcs_features_t features, const hcs_user_profile_t * const user_profile);`	
Function Description	Registers health care service instance.	
Parameters	`cb`	Application callbacks
	`hcs_features`	Application supported features
	`user_profile`	User profile settings
Return Values	Service instance	
Notes		

hcs_init_values()

Function Name	`void hcs_init_values(ble_service_t *svc, hcs_initial_values_t *initial_values);`	
Function Description	Initializes service values. Sets specific service values that can be read without prior notification sent by the server.	
Parameters	`svc`	Service instance
	`initial_values`	Values to set
Return Values	None	
Notes		

hcs_notify_status()

Function Name	`ble_error_t hcs_notify_status(ble_service_t *svc, uint16_t conn_idx, hcs_status_t status, const uint8_t *value, uint16_t length);`	
Function Description	Notifies client of the operation status.	
Parameters	`svc`	Service instance
	`conn_idx`	Connection index
	`status`	Operation status value
	`value`	Status payload
	`length`	Status payload length
Return Values	Status of the operation	
Notes		

© 2017 Dialog Semiconductor

hcs_notify_[service name]

Function Name	`ble_error_t hcs_notify_step(ble_service_t *svc, uint16_t conn_idx, hcs_step_t steps);`
	`ble_error_t hcs_notify_heart_rate(ble_service_t *svc, uint16_t conn_idx, hcs_heart_rate_t heart_rate);`
	`ble_error_t hcs_notify_calories(ble_service_t *svc, uint16_t conn_idx, hcs_calories_t calories);`
	`ble_error_t hcs_notify_sleep_quality(ble_service_t *svc, uint16_t conn_idx, hcs_sleep_quality_t sleep_quality);`
	`ble_error_t hcs_notify_body_state_class(ble_service_t *svc, uint16_t conn_idx, hcs_bsc_state_t bsc_state);`

Function Description	Notifies client of the step/heart rate/calories burned/sleep quality/body state classification epoch.
Parameters	`svc` Service instance `conn_idx` Connection index Corresponding value of `steps` Steps epoch `heart_rate` Heart rate epoch `calories` Calories burned epoch `sleep_quality` Sleep quality epoch `bsc_state` Body state classification value
Return Values	Status of the operation
Notes	

hcs_notify_acc_counter()

Function Name	`ble_error_t hcs_notify_acc_counter(ble_service_t *svc, uint16_t conn_idx, hcs_op_code_t op_code, uint32_t value);`
Function Description	Notifies client of the accumulated counter (daily total).
Parameters	`svc` Service instance `conn_idx` Connection index `op_code` Counter identification `value` Status payload
Return Values	Status of the operation
Notes	

hcs_notify_historical_data()

Function Name	`ble_error_t hcs_notify_historical_data(ble_service_t *svc, uint16_t conn_idx, uint8_t num_of_epochs, const uint8_t *data, uint16_t length);`
Function Description	Notifies client of the historical data.
Parameters	`svc` Service instance `conn_idx` Connection index `num_of_epochs` Number of epochs to notify `data` Historical data payload `length` Historical data payload length
Return Values	Status of the operation
Notes	

© 2017 Dialog Semiconductor

6.2.3 Callback Functions

user_profile_cfg()

Function Name	`void user_profile_cfg(ble_service_t *svc, uint16_t conn_idx, hcs_user_profile_t *user_profile);`
Function Description	Callback function to be called when a user profile is received.
Parameters	`svc` Service instance `conn_idx` Connection index `user_profile` User profile settings
Return Values	None
Notes	

[sensor name]_cfg()

Function Name	`void step_cfg(ble_service_t *svc, uint16_t conn_idx, hcs_ctrl_cmd_t cmd);`
	`void heart_rate_cfg(ble_service_t *svc, uint16_t conn_idx, hcs_ctrl_cmd_t cmd);`
	`void sleep_cfg(ble_service_t *svc, uint16_t conn_idx, hcs_ctrl_cmd_t cmd);`
	`void calories_cfg(ble_service_t *svc, uint16_t conn_idx, hcs_ctrl_cmd_t cmd);`
	`void body_state_cfg(ble_service_t *svc, uint16_t conn_idx, hcs_ctrl_cmd_t cmd);`

Function Description	Callback function to be called when a step counting/heart rate/sleep quality/calories burned/body state classification configuration command is received.
Parameters	`svc` Service instance `conn_idx` Connection index `cmd` Service command
Return Values	None
Notes	

operation_status()

Function Name	`hcs_operation_status_t operation_status(ble_service_t *svc, uint16_t conn_idx, hcs_op_code_t op_code);`
Function Description	Callback function to be called when operation status of a service is requested.
Parameters	`svc` Service instance `conn_idx` Connection index `op_code` Service reference
Return Values	Requested status
Notes	

© 2017 Dialog Semiconductor

get_acc_counter_val()

Function Name	`uint32_t get_acc_counter_val(ble_service_t *svc, uint16_t conn_idx, hcs_op_code_t op_code);`
Function Description	Callback function to be called when a service's accumulated counter is requested.
Parameters	`svc` Service instance `conn_idx` Connection index `op_code` Value reference
Return Values	Requested value
Notes	

read_historical_data()

Function Name	`hcs_status_t read_historical_data(ble_service_t *svc, uint16_t conn_idx, hcs_op_code_t op_code, uint32_t timestamp, hcs_data_chunk_header_t *header);`
Function Description	Callback function to be called when a service's historical data is requested.
Parameters	`svc` Service instance `conn_idx` Connection index `op_code` Service reference `timestamp` Timestamp to start searching from `header` Recovered header if data matching the criteria was found
Return Values	Status of the operation
Notes	Upon return of the callback function, application has to send the corresponding data using the `hcs_notify_historical_data()` function.

historical_notify_cfm()

Function Name	`void historical_notify_cfm(ble_service_t *svc, uint16_t conn_idx);`
Function Description	Callback function to be called when a confirmation of a historical data notification is received by BLE stack.
Parameters	`svc` Service instance `conn_idx` Connection index
Return Values	None
Notes	

notif_en()

Function Name	`void notif_en(ble_service_t *svc, uint16_t conn_idx, hcs_op_code_t op_code);`
Function Description	Callback function to be called when a service's notifications configuration is enabled.
Parameters	`svc` Service instance `conn_idx` Connection index `op_code` Service reference
Return Values	None
Notes	

© 2017 Dialog Semiconductor

6.3 SUOTA Service

The SUOTA Service is a Dialog proprietary BLE service used for updating the firmware of BLE devices. Details and service specification can be found in [4].

A complete API specification of the SUOTA implementation can be found in the Doxygen generated files of the SDK and specifically in the file `sdk_680/doc/html/dlg__suota_8h.html`.

6.4 Current Time Service

The Current Time Service (CTS) is a standard BLE SIG service. It is used by the Wearable DK software to update the system time from the peer central device (smartphone). The specification of the service can be found in [6].

A complete API specification of the CTS implementation can be found in the Doxygen generated files of the SDK and specifically in the file `sdk_680/doc/html/cts_8h.html`.

6.5 Heart Rate Service

The Heart Rate Service (HRS) is a standard BLE SIG service. It is used by the Wearable DK software, as an alternative to the HCS service, for sending heart rate samples to the peer central device (smartphone). The specification of the service can be found in [5].

A complete API specification of the HRS implementation can be found in the Doxygen generated files of the SDK and specifically in the file `sdk_680/doc/html/hrs_8h.html`.

6.6 Battery Service

The Battery Service (BAS) is a standard BLE SIG service. It is used by the Wearable DK software to send the current battery status to the peer central device (smartphone). The specification of the service can be found in [7].

A complete API specification of the BAS implementation can be found in the Doxygen generated files of the SDK and specifically in the file `sdk_680/doc/html/bas_8h.html`.

© 2017 Dialog Semiconductor

7 User Interface

7.1 UI Task

The UI task is the upper layer in communication with the LS013B7DH03 display. The UI task functions can be called from other tasks (e.g. the WRBL task) for specific operations on the LS013B7DH03 via the GUI and the display driver API.

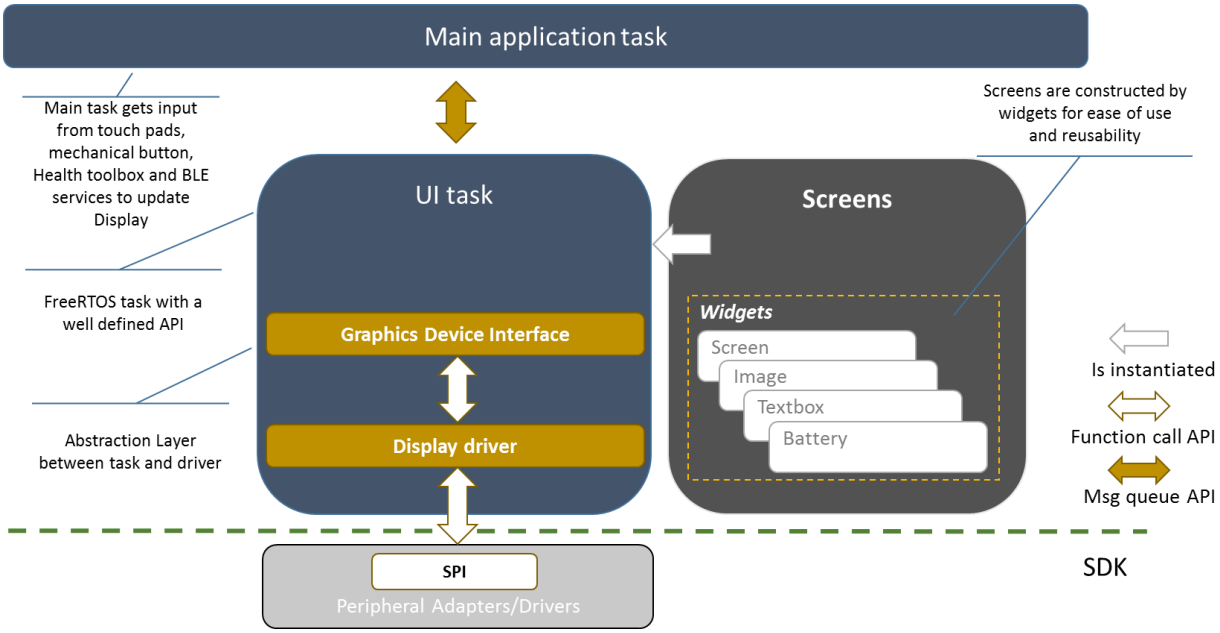

Figure 4: UI Task Block Diagram

© 2017 Dialog Semiconductor

Table 14: User Interface Task API

Structures	
ui_screen_num_t ui_operation_mode_t ui_t	ui_change_mode_t ui_screen_element_t
Functions	
ui_task_init() ui_task_register_screen_[change/updated]_cb() ui_task_swipe_screen_[left/right/up/down]() ui_task_blink_screen_elem() ui_task_change_opmode() ui_task_set_battery_level() ui_task_set_connection_state() ui_task_set_time_[hours/minutes]_val() ui_task_set_alarm_[hours/minutes]_val()	ui_task_[enable/disable]_display() ui_task_set_screen() ui_task_show_splash_screen() ui_task_set_change_mode() ui_task_set_vibration_state() ui_task_set_battery_state() ui_task_set_alarm_state() ui_task_set_date_[day/month/year]_val() ui_task_set_[sensor or service]_val()
Callbacks	
screen_changed_cb()	screen_updated_cb()

7.1.1 Data Structures and Types

ui_screen_num_t

Type Definition Name	Description
ui_screen_num_t	Screen number enumeration type defined as: UI_TIME_SCREEN = 0, UI_DATE_SCREEN, UI_ALARM_SCREEN, UI_NFC_SCREEN, UI_HEART_RATE_SCREEN, UI_STEPS_SCREEN, UI_DISTANCE_SCREEN, UI_CALORIES_SCREEN, UI_LIGHT_SLEEP_SCREEN, UI_DEEP_SLEEP_SCREEN, UI_TEMPERATURE_SCREEN, UI_HUMIDITY_SCREEN, UI_PRESSURE_SCREEN, UI_MAX_SCREENS

ui_change_mode_t

Type Definition Name	Description
ui_change_mode_t	Time change mode enumeration type defined as: UI_DISPLAY_MODE = 0, UI_TIME_HOURS_SETUP, UI_TIME_MINUTES_SETUP, UI_DATE_DAY_SETUP, UI_DATE_MONTH_SETUP, UI_DATE_YEAR_SETUP, UI_ALARM_SETUP, UI_ALARM_HOURS_SETUP, UI_ALARM_MINUTES_SETUP, UI_ALARM_EVENT,

© 2017 Dialog Semiconductor

ui_operation_mode_t

Type Definition Name	Description
ui_operation_mode_t	Operation mode enumeration type defined as: UI_OPERATION_MODE_ON, UI_OPERATION_MODE_OFF, UI_OPERATION_MODE_PWR_SAVE

ui_screen_element_t

Type Definition Name	Description
ui_screen_element_t	Screen element enumeration type defined as: UI_ELEM_ICON, UI_ELEM_LABEL

ui_t

Field	Type	Description
handle	OS_TASK	Task handle of UI task
queue	OS_QUEUE	Queue handle of UI task
lock	OS_MUTEX	Mutex lock of UI task
active_screen	ui_screen_num_t	ID number of active screen
screen_changed_cb	screen_changed_cb_t	Callback for screen changed
screen_updated_cb	screen_changed_cb_t	Callback for screen updated

© 2017 Dialog Semiconductor

7.1.2 Functions

ui_task_init()

Function Name	`OS_BASE_TYPE ui_task_init(void);`
Function Description	Initializes UI task.
Parameters	
Return Values	OS_OK if initialization completed successfully, OS_FAIL otherwise.
Notes	

ui_task_[enable/disable]_display()

Function Name	`void ui_task_enable_display();` `void ui_task_disable_display();`
Function Description	Enables\disables display.
Parameters	None
Return Values	
Notes	

ui_task_register_screen_[change/updated]_cb()

Function Name	`void ui_task_register_screen_change_cb(screen_changed_cb_t screen_changed_cb);` `void ui_task_register_screen_updated_cb(screen_changed_cb_t screen_updated_cb);`
Function Description	Registers callback function which is called after screen changes/updates.
Parameters	`screen_[changed/updated]_cb` Pointer to the callback function
Return Values	
Notes	

ui_task_set_screen()

Function Name	`void ui_task_set_screen(ui_screen_num_t screen);`
Function Description	Sets active screen.
Parameters	`screen` Screen number to set
Return Values	

Notes	Valid types (screens) are:
	`UI_TIME_SCREEN`
	`UI_DATE_SCREEN`
	`UI_ALARM_SCREEN`
	`UI_NFC_SCREEN`
	`UI_HEART_RATE_SCREEN`
	`UI_STEPS_SCREEN`
	`UI_DISTANCE_SCREEN`
	`UI_CALORIES_SCREEN`
	`UI_LIGHT_SLEEP_SCREEN`
	`UI_DEEP_SLEEP_SCREEN`
	`UI_TEMPERATURE_SCREEN`
	`UI_HUMIDITY_SCREEN`
	`UI_PRESSURE_SCREEN`

© 2017 Dialog Semiconductor

ui_task_swipe_screen_[left/right/up/down]()

Function Name	`void ui_task_swipe_screen_left();`
	`void ui_task_swipe_screen_right();`
	`void ui_task_swipe_screen_up();`
	`void ui_task_swipe_screen_down();`

Function Description	Swipes to left/right/up/down screen.
Parameters	None
Return Values	
Notes	

ui_task_show_splash_screen()

Function Name	`void ui_task_show_splash_screen();`
Function Description	Shows splash screen with Dialog Semi logo.
Parameters	None
Return Values	
Notes	

ui_task_blink_screen_elem()

Function Name	void ui_task_blink_screen_elem(ui_screen_num_t screen, ui_screen_element_t elem, uint8_t enable);
Function Description	Enables or disables selected screen element blinking.
Parameters	`screen` Screen number `elem` Element type `enable` true to blink element, false to stop blinking
Return Values	

Notes	Valid types (screens) are:
	`UI_NFC_SCREEN`
	`UI_HEART_RATE_SCREEN`
	`UI_STEPS_SCREEN`
	`UI_DISTANCE_SCREEN`
	`UI_CALORIES_SCREEN`
	`UI_LIGHT_SLEEP_SCREEN`
	`UI_DEEP_SLEEP_SCREEN`
	`UI_TEMPERATURE_SCREEN`
	`UI_HUMIDITY_SCREEN`
	`UI_PRESSURE_SCREEN`

ui_task_set_change_mode()

Function Name	`void ui_task_set_change_mode(uint8_t mode);`
Function Description	Set change mode for time/date/alarm screen
Parameters	`mode` The mode of change (highlighted value)
Return Values	
Notes	

ui_task_change_opmode()

© 2017 Dialog Semiconductor

Function Name	`void ui_task_change_opmode(ui_screen_num_t screen, ui_op_mode_t opmode);`
Function Description	Changes operation mode for the selected screen.
Parameters	`screen` Screen number `opmode` Operation mode to set
Return Values	

Notes	Valid types (screens) are:
	`UI_NFC_SCREEN`
	`UI_HEART_RATE_SCREEN`
	`UI_STEPS_SCREEN`
	`UI_DISTANCE_SCREEN`
	`UI_CALORIES_SCREEN`
	`UI_LIGHT_SLEEP_SCREEN`
	`UI_DEEP_SLEEP_SCREEN`

ui_task_set_vibration_state()

Function Name	`void ui_task_set_vibration_state (ui_op_mode_t opmode);`
Function Description	Set the vibration state
Parameters	`opmode` Operation mode to set
Return Values	

Notes	Valid types (opmode) are:
	`UI_OP_MODE_ON`
	`UI_OP_MODE_OFF`

ui_task_set_battery_level()

Function Name	`void ui_task_set_battery_level(uint8_t level);`
Function Description	Sets new value for battery level.
Parameters	`level` New value for battery level
Return Values	
Notes	

ui_task_set_battery_state()

Function Name	`void ui_task_set_battery_state(ui_battery_state_t state);`
Function Description	Sets state of battery status.
Parameters	`state` Value of battery status
Return Values	
Notes	

ui_task_set_connection_state()

Function Name	`void ui_task_set_connection_state(uint8_t connected);`
Function Description	Sets connection state.
Parameters	`connected` Connection state
Return Values	
Notes	

© 2017 Dialog Semiconductor

ui_task_set_alarm_state()

Function Name	`void ui_task_set_alarm_state(bool value);`
Function Description	Sets state of alarm.
Parameters	`value` Alarm state (true = ON, false = OFF)
Return Values	
Notes	

ui_task_set_time_[hours/minutes]_val()

Function Name	`void ui_task_set_time_hours_val(uint8_t value);` `void ui_task_set_time_minutes_val(uint8_t value);`
Function Description	Sets new value for hours/minutes.
Parameters	`value` New value for hours/minutes
Return Values	
Notes	

ui_task_set_date_[day/month/year]_val()

Function Name	`void ui_task_set_date_day_val(uint8_t value);`
	`void ui_task_set_date_month_val(uint8_t value);`
	`void ui_task_set_date_year_val(uint8_t value);`

Function Description	Sets new value for date day/month/year.
Parameters	`value` New value for day/month/year
Return Values	
Notes	

ui_task_set_alarm_[hours/minutes]_val()

Function Name	`void ui_task_set_alarm_hours_val(uint8_t value);` `void ui_task_set_alarm_minutes_val(uint8_t value);`
Function Description	Sets new value for alarm hours/minutes.
Parameters	`value` New value for hours/minutes
Return Values	
Notes	

ui_task_set_[sensor or service]_val()

Function Name	`void ui_task_set_heart_rate_val(uint8_t value);`
	`void ui_task_set_steps_val(uint16_t value);`
	`void ui_task_set_distance_val(uint16_t value);`
	`void ui_task_set_calories_val(uint16_t value);`
	`void ui_task_set_light_sleep_val(uint16_t value);`
	`void ui_task_set_deep_sleep_val(uint16_t value);`
	`void ui_task_set_temperature_val(uint16_t value);`
	`void ui_task_set_humidity_val(uint8_t value);`
	`void ui_task_set_pressure_val(uint16_t value);`

Function Description	Sets new value for heart rate / steps / distance / calories / light sleep / deep sleep / temperature / humidity / pressure screen.
Parameters	`value` New value for corresponding label
Return Values	

© 2017 Dialog Semiconductor

Notes	

7.1.2.2 Callbacks

screen_changed_cb()

Function Name	`void screen_changed_cb(ui_screen_num_t active_screen);`
Function Description	Informs of change screen event.
Parameters	`active_screen` Number of active screen
Return Values	
Notes	

screen_updated_cb()

Function Name	`void screen_updated_cb(ui_screen_num_t active_screen);`
Function Description	Informs of refresh screen event.
Parameters	`active_screen` Number of active screen
Return Values	
Notes	

7.2 GUI

The GUI module is the middle layer in communication with the LS013B7DH03 display. There is a set of Graphical Device Interface (GDI) functions, images and widgets that can be called from the upper layers (UI task) for specific operations on the LS013B7DH03 via the display driver API.

7.2.1 GDI

Table 15: GDI API

Structures	
gdi_color_t	gdi_t
Functions	
gdi_init() gdi_display_isenabled() gdi_display_clear() gdi_set_bg_color() gdi_set_next_frame_buffer() gdi_set_frame_buffer() gdi_draw_[fill_]circle() gdi_draw_image() gdi_draw_arc()	gdi_display_[enable/display]() gdi_update_display() gdi_swipe_[left/right/up/down]() gdi_flush_frame_buffers() gdi_get_current_frame_buffer() gdi_[set/clear]_dot() gdi_draw_line() gdi_draw_[fill_]rect()

7.2.1.1 Data Structures and Types

gdi_color_t

Type Definition Name	Description

`gdi_color_t`	GDI color enumeration type defined as:
	`GDI_COLOR_BLACK = 0x00,`
	`GDI_COLOR_WHITE = 0xFF`

© 2017 Dialog Semiconductor

gdi_t

Field	Type	Description
buffers[GDI_NUM_OF_BUFFERS]	uint8_t *	Frame buffers table
active_buffer	uint8_t *	Pointer to active frame buffer
buffer_num	uint8_t	Number of active frame buffer
bg_color	gdi_color_t	Display background color
display_enabled	bool	Display enabled state

© 2017 Dialog Semiconductor

7.2.1.2 Functions

gdi_init()

Function Name	`void gdi_init(void);`
Function Description	Initializes the GDI instance – allocate memory and set default background color
Parameters	None
Return Values	None
Notes	

gdi_display_[enable/display]()

Function Name	`void gdi_display_enable(void);` `void gdi_display_disable(void);`
Function Description	Enable / disable the display
Parameters	None
Return Values	None
Notes	

gdi_display_isenabled()

Function Name	`bool gdi_display_isenabled(void);`
Function Description	Checks if display is enabled
Parameters	None
Return Values	True if display is enabled, false otherwise.
Notes	

gdi_update_display()

Function Name	`void gdi_update_display(void);`
Function Description	Moves data from active frame buffer to the display memory.
Parameters	None
Return Values	None
Notes	

gdi_display_clear()

Function Name	`void gdi_display_clear(void);`
Function Description	Clears the display.
Parameters	None
Return Values	None
Notes	

gdi_swipe_[left/right/up/down]()

Function Name	`void gdi_swipe_left(void);`
	`void gdi_swipe_right(void);`
	`void gdi_swipe_up(void);`
	`void gdi_swipe_down(void);`

© 2017 Dialog Semiconductor

Function Description	Updates display in left/right/up/down direction using data from two frame buffers with swipe animation.
Parameters	None
Return Values	None
Notes	

gdi_set_bg_color()

Function Name	`void gdi_set_bg_color(gdi_color_t color);`
Function Description	Sets the background color for all frame buffers. The existing frame buffer content will be erased.
Parameters	`Color` Background color: black or white
Return Values	None
Notes	

gdi_flush_frame_buffers()

Function Name	`void gdi_flush_frame_buffers(void);`
Function Description	Clears all frame buffers.
Parameters	None
Return Values	None
Notes	

gdi_set_next_frame_buffer()

Function Name	`void gdi_set_next_frame_buffer(void);`
Function Description	Selects next available frame buffer.
Parameters	None
Return Values	None
Notes	

gdi_get_current_frame_buffer()

Function Name	`uint8_t gdi_get_current_frame_buffer(void);`
Function Description	Gets number of current frame buffer.
Parameters	None
Return Values	Number of currently selected frame buffer
Notes	

gdi_set_frame_buffer()

Function Name	`void gdi_set_frame_buffer(uint8_t frame);`
Function Description	Sets active frame buffer.
Parameters	`Frame` Number of frame buffer
Return Values	None
Notes	

© 2017 Dialog Semiconductor

gdi_[set/clear]_dot()

Function Name	`void gdi_set_dot(uint8_t x, uint8_t y);` `void gdi_clear_dot(uint8_t x, uint8_t y);`	
Function Description	Draws/clears pixel.	
Parameters	`x` `y`	Pixel x coordinate Pixel y coordinate
Return Values	None	
Notes		

gdi_draw_[fill_]circle()

Function Name	`void gdi_draw_circle(uint8_t x0, uint8_t y0, uint8_t radius);` `void gdi_draw_fill_circle(uint8_t x0, uint8_t y0, uint8_t radius);`	
Function Description	Draws circle/filled circle.	

Parameters	`x0`
	`y0`
	`radius`
		Circle center point x coordinate
		Circle center point y coordinate
		Circle radius

Return Values	None	
Notes		

gdi_draw_line()

Function Name	`void gdi_draw_line(uint8_t x0, uint8_t y0, uint8_t x1, uint8_t y1);`	
Function Description	Draws line.	

Parameters	`x0`
	`y0`
	`x1`
	`y1`
		Start point x coordinate
		Start point y coordinate
		End point x coordinate
		End point y coordinate

Return Values	None	
Notes		

gdi_draw_image()

Function Name	`void gdi_draw_image(uint8_t x, uint8_t y, const uint8_t *image,` `uint8_t width, uint8_t height);`	
Function Description	Draw image	

Parameters	`x`
	`y`
	`image`
	`width`
	`height`
		Upper left corner x coordinate
		Upper left corner y coordinate
		Pointer to image bitmap
		Image width
		Image height

Return Values	None	
Notes		

gdi_draw_[fill_]rect()

Function Name	`void gdi_draw_rect(uint8_t x, uint8_t y, uint8_t w, uint8_t h);` `void gdi_draw_fill_rect(uint8_t x, uint8_t y, uint8_t w, uint8_t h);`

© 2017 Dialog Semiconductor

Function Description	Draw rectangle/filled rectangle.	
Parameters	`x`	Upper left corner x coordinate
	`y`	Upper left corner y coordinate
	`w`	Rectangle width
	`h`	Rectangle height
Return Values	None	
Notes		

gdi_draw_arc()

Function Name	`void gdi_draw_arc(uint8_t x, uint8_t y, uint8_t r, uint8_t t, int start_angle, int end_angle);`	
Function Description	Informs of refresh screen event.	
Parameters	`x`	Arc center point x coordinate
	`y`	Arc center point y coordinate
	`r`	Arc radius
	`t`	Line thickness
	`start_angle`	Arc start angle
	`end_angle`	Arc end angle
Return Values	None	
Notes		

7.2.2 Widgets

Table 16: Widgets API

Structures	
ui_battery_state_t ui_align_t ui_screen_item_t ui_static_label_t ui_textbox_t	ui_battery_widget_t ui_item_type_t ui_screen_t ui_status_bar_t
Battery Widget Functions	
ui_draw_battery() ui_battery_widget_set_state()	ui_battery_widget_set_level()
Circle Progress Bar Functions	
ui_draw_circle_progress()	ui_circle_progress_set_value()
Common Functions	
ui_measure_string()	ui_draw_string()
Image Functions	
ui_draw_image()	
Screen Functions	
ui_set_active_screen() ui_prepare_next_screen() ui_set_item_[visibility/validity/blinking]() ui_toggle_item_visibility()	ui_draw_active_screen() ui_swipe_[left/right/up/down]() ui_get_item_[visibility/validity/blinking]()

© 2017 Dialog Semiconductor

Static Label Functions	
ui_draw_static_label()	
Status Bar Functions	
ui_draw_status_bar() ui_register_status_bar()	ui_status_bar_redraw_needed() ui_unregister_status_bar()
Textbox Functions	
ui_draw_textbox()	ui_textbox_set_text()

7.2.2.1 Data Structures and Types

ui_battery_state_t

Type Definition Name	Description
ui_battery_state_t	Battery state enumeration type defined as: UI_BATTERY_NOT_CHARGING, UI_BATTERY_CHARGING, UI_BATTERY_WARNING

ui_battery_widget_t

Field	Type	Description
level	uint8_t	Battery level
state	ui_battery_state_t	Battery state

ui_align_t

Type Definition Name	Description
ui_align_t	Text alignment enumeration type defined as: UI_ALIGN_LEFT, UI_ALIGN_CENTER, UI_ALIGN_RIGHT

ui_item_type_t

Type Definition Name	Description
ui_item_type_t	Screen item type enumeration type defined as: UI_STATIC_LABEL, UI_CIRCLE_PROGRESS_BAR, UI_TEXTBOX, UI_IMAGE, UI_BATTERY_WIDGET

ui_screen_item_t

Field	Type	Description
x	const uint8_t	X coordinate of screen item
y	const uint8_t	Y coordinate of screen item
width	const uint8_t	Width of screen item
height	const uint8_t	Height of screen item
type	const ui_item_type_t	Item type

© 2017 Dialog Semiconductor

Field	Type	Description
properties	const void *	Can be used to store additional item parameters. Each widget has its own parameters.
status	void *	Can be used to handle item status. Currently used only by textbox widget to store initial text.
flags	uint8_t *	Flags used to define item state. These flags are used to determine whether the item should be redrawn or whether it should be visible or not.

ui_screen_t

Field	Type	Description
flags	uint8_t *	Flags used to define the screen state. These flags are used to determine whether the screen should be redrawn or whether it supports the status bar.
items	const ui_screen_item_t* const *	Array of ui_screen_item_t which contains all screen items.
param	void *	Can be used to store additional screen parameters. Currently not used
on_main	on_main_cb_t	Callback that will be called every time the screen is refreshed.

ui_static_label_t

Field	Type	Description
text	const char *	Label text
align	const ui_align_t	Label text alignment.
font	const font_info_t *	Pointer to the font descriptor.

ui_status_bar_t

Field	Type	Description
height	uint8_t	Height of status bar widget
spacer	uint8_t	Defines the space in pixels between two items in the status bar.
items	const ui_screen_item_t* const *	Pointer to the screen item

ui_textbox_t

Field	Type	Description
border	const bool	Display textbox border or not.
font	const font_info_t *	Pointer to the font descriptor.
align	const ui_align_t	Text alignment.
text	char *	Pointer to the textbox text.

© 2017 Dialog Semiconductor

7.2.2.2 Battery Widget Functions

ui_draw_battery()

Function Name	`void ui_draw_battery(const ui_screen_item_t *item, uint8_t x, uint8_t y);`	
Function Description	Draws battery widget.	
Parameters	`item`	Pointer to suitable screen widget
	`x`	X coordinate of the widget
	`y`	Y coordinate of the widget
Return Values	None	
Notes		

ui_battery_widget_set_level()

Function Name	`void ui_battery_widget_set_level(const ui_screen_item_t *item, uint8_t level);`	
Function Description	Sets battery level.	
Parameters	`item`	Pointer to suitable screen widget
	`level`	Battery level from 0 to 100
Return Values	None	
Notes		

ui_battery_widget_set_state()

Function Name	`void ui_battery_widget_set_state(const ui_screen_item_t *item, ui_battery_state_t state);`	
Function Description	Sets battery state.	
Parameters	`item`	Pointer to suitable screen widget
	`level`	Battery state
Return Values	None	
Notes		

7.2.2.3 Circle Progress Bar Functions

ui_draw_circle_progress()

Function Name	`void ui_draw_circle_progress(const ui_screen_item_t *item);`	
Function Description	Draws circle progress bar widget.	
Parameters	`item`	Pointer to suitable screen widget
Return Values	None	
Notes		

ui_circle_progress_set_value()

Function Name	`void ui_circle_progress_set_value(const ui_screen_item_t *item, uint8_t value);`	
Function Description	Sets progress value.	
Parameters	`item`	Pointer to suitable screen widget
	value	Progress bar value

© 2017 Dialog Semiconductor

Return Values	None
Notes	

7.2.2.4 Common Functions

ui_measure_string()

Function Name	`uint8_t ui_measure_string(const font_info_t * const font, const char * const str);`	
Function Description	Measures the string length.	
Parameters	`font`	Pointer to the font definition
	`str`	Pointer to the string
Return Values	String length in pixels.	
Notes		

ui_draw_string()

Function Name	`void ui_draw_string(uint8_t x, uint8_t y, const font_info_t * const font, const char * const str);`	
Function Description	Draws string.	
Parameters	`x`	Upper left corner x coordinate
	`y`	Upper left corner y coordinate
	`font`	Pointer to the font definition
	`str`	Pointer to the string
Return Values	None	
Notes		

7.2.2.5 Image Functions

ui_draw_image()

Function Name	`void ui_draw_image(const ui_screen_item_t *item);`	
Function Description	Draws image widget.	
Parameters	`item`	Pointer to suitable screen widget
Return Values	None	
Notes		

© 2017 Dialog Semiconductor

7.2.2.6 Screen Functions

ui_set_active_screen()

Function Name	`void ui_set_active_screen(const ui_screen_t * const screen);`
Function Description	Sets active screen.
Parameters	`screen` Pointer to suitable screen
Return Values	None
Notes	

ui_draw_active_screen()

Function Name	`uint8_t ui_draw_active_screen(bool force_redraw);`
Function Description	Draws active screen.
Parameters	`force_redraw` True if screen should be redraw
Return Values	1 if screen was redrawn, 0 otherwise
Notes	

ui_prepare_next_screen()

Function Name	`void ui_prepare_next_screen(const ui_screen_t * const screen);`
Function Description	Draws next screen in next available frame buffer.
Parameters	`screen` Pointer to suitable screen
Return Values	None
Notes	

ui_swipe_[left/right/up/down]()

Function Name	`void ui_swipe_left(const ui_screen_t * const screen);`
	`void ui_swipe_right(const ui_screen_t * const screen);`
	`void ui_swipe_up(const ui_screen_t * const screen);`
	`void ui_swipe_down(const ui_screen_t * const screen);`

Function Description	Switches to left/right/up/down screen using swipe animation.
Parameters	`screen` Pointer to suitable screen
Return Values	None
Notes	

ui_set_item_[visibility/validity/blinking]()

Function Name	`void ui_set_item_visibility(const ui_screen_item_t * const item,`
	`  bool visibility);`
	`void ui_set_item_validity(const ui_screen_item_t * const item,`
	`  bool validity);`
	`void ui_set_item_blinking(const ui_screen_item_t * const item,`
	`  bool blinking);`

Function Description	Sets visibility / validity / blinking state of screen widget.

© 2017 Dialog Semiconductor

Parameters	`item`	Pointer to suitable screen widget
	Corresponding parameter	
	`visibility`	Visibility state of screen widget
	`validity`	Validity state of screen widget
	`blinking`	Blinking state of screen widget
Return Values	None	
Notes		

ui_get_item_[visibility/validity/blinking]()

Function Name	`bool ui_get_item_visibility(const ui_screen_item_t * const item);`
	`bool ui_get_item_validity(const ui_screen_item_t * const item);`
	`bool ui_get_item_blinking(const ui_screen_item_t * const item);`

Function Description	Gets current visibility /validity / blinking state of screen widget.
Parameters	`item` Pointer to suitable screen widget
Return Values	Current visibility / validity / blinking state of screen widget
Notes	

ui_toggle_item_visibility()

Function Name	`void ui_toggle_item_visibility(const ui_screen_item_t * const item);`
Function Description	Toggles visibility state of screen widget.
Parameters	`item` Pointer to suitable screen widget
Return Values	None
Notes	

7.2.2.7 Static Label Functions

ui_draw_static_label()

Function Name	`void ui_draw_static_label(const ui_screen_item_t *item);`
Function Description	Draws static label widget.
Parameters	`item` Pointer to suitable screen widget
Return Values	None
Notes	

7.2.2.8 Status Bar Functions

ui_draw_status_bar()

Function Name	`void ui_draw_status_bar(const ui_status_bar_t *status_bar);`
Function Description	Draws status bar widget.
Parameters	`status_bar` Pointer to status bar widget
Return Values	None
Notes	

ui_status_bar_redraw_needed()

Function Name	`uint8_t ui_status_bar_redraw_needed(const ui_status_bar_t *status_bar);`

© 2017 Dialog Semiconductor

Function Description	Checks if status bar widget should be redrawn.
Parameters	`status_bar` Pointer to status bar widget
Return Values	None
Notes	

ui_register_status_bar()

Function Name	`void ui_register_status_bar(const ui_status_bar_t *bar);`
Function Description	Registers status bar.
Parameters	`bar` Pointer to status bar
Return Values	None
Notes	

ui_unregister_status_bar()

Function Name	`void ui_unregister_status_bar(void);`
Function Description	Unregisters status bar.
Parameters	None
Return Values	None
Notes	

7.2.2.9 Text Box Functions

ui_draw_textbox()

Function Name	`void ui_draw_textbox(const ui_screen_item_t *item);`
Function Description	Draws textbox widget.
Parameters	`item` Pointer to suitable screen widget
Return Values	None
Notes	

ui_textbox_set_text()

Function Name	`void ui_textbox_set_text(const ui_screen_item_t *item, const char *format, ...);`
Function Description	Sets textbox widget text.
Parameters	`item` Pointer to suitable screen widget `format` Format string
Return Values	None
Notes	

8 Health Toolbox

The main objective of the Health Toolbox software component is to implement all appropriate functionality for the control and data manipulation related to:

- Sensor modules integrated in the Wearable DK device, targeting mainly environment, human motion, and healthcare oriented applications.

 © 2017 Dialog Semiconductor

- Services/algorithms processing sensor data and producing human motion and healthcare related metrics, which determine an estimate of human physical activity and health status.

Based on this basic classification of control and data flows determining the general execution flow inside the Health Toolbox component, three main types of functionality can be identified and corresponding software implementation parts, as shown in Figure 5:

- The relatively low level control of sensor modules for the appropriate data acquisition, involving register accesses and interrupt-handling processes and being implemented by:
  - Sensor Control component
  - BMI160, BMM150 and BME280 sensor module drivers
  - DI5115 sensor module driver
- The processing of sensor data and their transformation to more sophisticated ones, involving the execution of specialized algorithms and being implemented by:
  - Health Care Services Task, supporting the following services:
    - Heart Rate Estimation
    - Sensor Fusion
    - Calories Counting
    - Sleep Quality Estimation
  - KIWI Services Task, supporting the following services:
    - Step Counting
    - Calories Counting
    - Sleep Quality Estimation
    - Body State/Activity Classification
- The core functionality of coordinating the control and data flows inside the Health Toolbox component, so that a unified easy-to-use sensor services configuration and data acquisition framework can be offered to the upper application layers (i.e. Wearable Main Application). That functionality is implemented by the Health Toolbox Task.

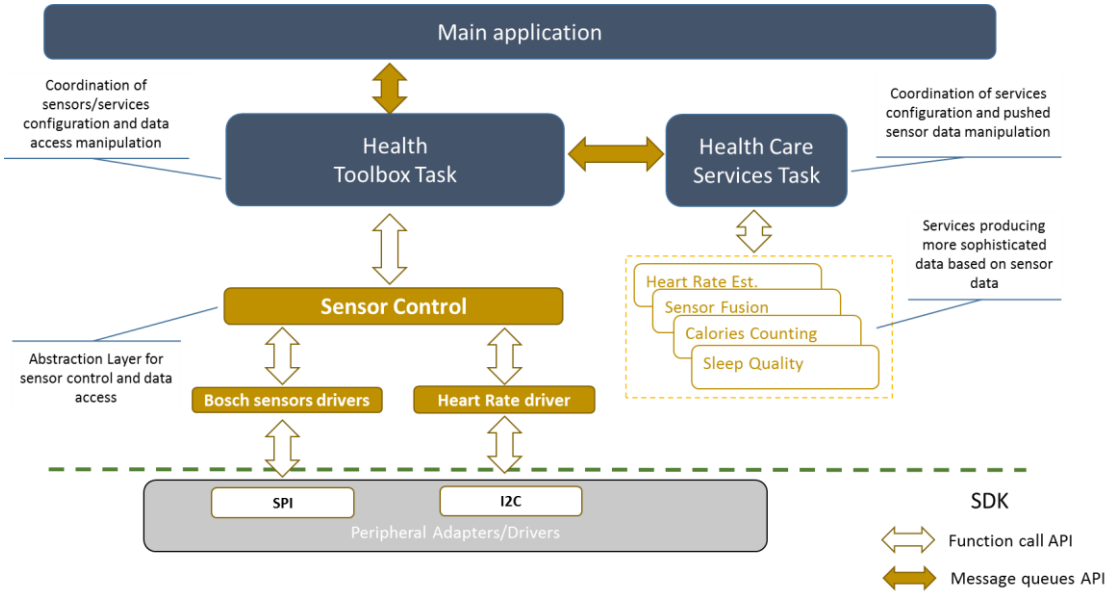

Figure 5: Health Toolbox Component Software Architecture

© 2017 Dialog Semiconductor

Table 17: Basic Health Toolbox API Sensors/Services Configuration Functions

Sensor or Service Name	API	Configuration Parameters			
		Enable/ Disable	Mode	Rate	Other
Sensors					
Accelerometer	`ht_config_acc()`	√	not available	0.78, 1.56, 3.12, 6.25, 12.5, 25, 50, 100 (Hz)	Range: ±2, ±4, ±8, ±16 (G)
Gyroscope	`ht_config_gyr()`	√	not available	25, 50, 100 (Hz)	Range: 125, 250, 500, 1000, 2000 (deg/s)
Magnetometer	`ht_config_mag()`	√	(calibration) mode: none, static, basic/smart, auto-continuous or one-shot	0.78, 1.56, 3.12, 6.25, 12.5, 25, 50, 100 (Hz)	not available
Environmental Sensors: humidity, temperature, pressure	`ht_config_env()`	√	not available	no rate, 0.125, 0.25, 0.5, 1, 2, 4, 8 (Hz)	not available
Step Counting	`ht_config_step()`	√	not available	no rate, 0.125, 0.25, 0.5, 1, 2, 4, 8 (Hz)	not available
Services					
Sensor Fusion	`ht_config_sf()`	√	sensor combination: Gyr, Gyr+Mag, Acc+Mag, Acc+Gyr+Mag	12.5, 25, 50 (Hz)	not available
Heart Rate	`ht_config_hr()`	√	(operation) mode: regular (always on) intermittent	Interval duration: no rate, 1, 3, 5, 7, 15, 30, 60 (s)	not available
Sleep Quality	`ht_config_sq()`	√	not available	not available	not available
Calories Counting	`ht_config_cc()`	√	(operation) mode: regular robust	Interval duration: no rate, 10, 20, 30, 40, 50, 60, 120, 180, 240, 300 (s)	not available
Body State Classification (Note 1)	`ht_config_bs()`	√	not available	not available	not available

Note 1 Only when KIWI Services component is enabled.

8.1 Health Toolbox Task

The API provided by the Health Toolbox Task includes functions for initializing and controlling its operation, configuring the supported/integrated sensors and services, and finally acquiring the sensors/services' configuration state and last data. All these functions and respective data structures/types being defined in the API are summarized below:

© 2017 Dialog Semiconductor

Table 18: Health Toolbox Task API

Data Structures and Types		
ht_error_t	ht_upf_config_t	ht_sf_coeffs_t
ht_calibr_offsets_t	ht_calibr_coeffs_t	ht_calibr_ctrl_t
ht_acc_config_t	ht_gyr_config_t	ht_mag_config_t
ht_env_config_t	ht_sc_config_t	ht_sf_config_t
ht_hr_config_t	ht_sq_config_t	ht_cc_config_t
ht_bs_config_t	ht_s1_config_t	ht_upf_t
ht_acc_t	ht_gyr_t	ht_mag_t
ht_env_t	ht_sc_t	ht_sf_t
ht_hr_t	ht_sq_t	ht_cc_t
ht_bs_t	ht_s1_t	ht_handle_t
ht_data_acc_t	ht_data_gyr_t	ht_data_mag_t
ht_data_env_t	ht_data_sc_t	ht_data_sf_t
ht_data_hr_t	ht_data_sq_t	ht_data_cc_t
ht_data_bs_t	ht_data_s1_t	

Dependency Functions	
ht_handle_t -> ht_clear_done()	ht_handle_t -> ht_start_done()
ht_handle_t -> ht_stop_done()	ht_handle_t -> ht_config_done()
ht_handle_t -> ht_state_get_done()	ht_handle_t -> ht_calibr_done()
ht_handle_t -> ht_data_read_done()	ht_handle_t -> ht_data_reset_done()
ht_handle_t -> ht_data_indication()	ht_handle_t -> ht_error()
ht_handle_t -> ht_get_rtc_notifs_config()	

Initialization and Control Functions	
ht_init()	ht_get_handle()
ht_clear()	ht_start()
ht_stop()	ht_rtc_send_notif()

Configuration Functions	
ht_config_upf()	ht_config_sf_coeffs()
ht_config_calibr_coeffs_mag()	ht_config_calibr_control_mag()
ht_config_calibr_offsets_acc()	ht_config_calibr_offsets_acc_update()
ht_config_acc()	ht_config_gyr()
ht_config_mag()	ht_config_env()
ht_config_sc()	ht_config_sf()
ht_config_hr()	ht_config_sq()
ht_config_cc()	ht_config_bs()
ht_config_s1()	

State Functions	
ht_isrunning()	ht_time_get_now_ms()
ht_state_get_upf()	ht_state_get_sf_coeffs()
ht_state_get_calibr_coeffs_mag()	ht_state_get_calibr_control_mag()
ht_state_get_acc()	ht_state_get_gyr()
ht_state_get_mag()	ht_state_get_env()
ht_state_get_sc()	ht_state_get_sf()
ht_state_get_hr()	ht_state_get_sq()
ht_state_get_cc()	ht_state_get_bs()
ht_state_get_s1()	

© 2017 Dialog Semiconductor

Data Acquisition Functions	
ht_data_read_acc() ht_data_read_mag() ht_data_read_sc() ht_data_read_hr() ht_data_read_cc() ht_data_read_s1()	ht_data_read_gyr() ht_data_read_env() ht_data_read_sf() ht_data_read_sq() ht_data_read_bs()
Data-Reset Functions	
ht_data_reset_sc() ht_data_reset_cc()	ht_data_reset_sq()

8.1.1 Data Structures and Types

ht_error_t

Type Definition Name	Description
ht_error_t	Execution status of a function: HT_SUCCESS, HT_FAIL

ht_upf_config_t

Data Structure Fields	Type	Description
gender	uint8_t	User's gender type (m/f): HT_UPF_GENDER_TYPE_FEMALE, HT_UPF_GENDER_TYPE_MALE
age	uint8_t	User's age (in year).
height	uint8_t	User's height (in cm).
weight	uint8_t	User's weight (in kg).
mv_source	uint8_t	User's wrist as source of movement (right/left): HT_UPF_MV_SOURCE_LEFT_WRIST, HT_UPF_MV_SOURCE_RIGHT_WRIST

ht_sf_coeffs_t

Data Structure Fields	Type	Description
beta_a	uint16_t	Parameter controlling relative weight of accelerometer data.
beta_m	uint16_t	Parameter controlling relative weight of magnetometer data.

ht_calibr_offsets_t

Data Structure Fields	Type	Description
offset_x	int16_t	Applied x-axis offset to sensor data.
offset_y	int16_t	Applied y-axis offset to sensor data.
offset_z	int16_t	Applied z-axis offset to sensor data.

© 2017 Dialog Semiconductor

ht_calibr_coeffs_t

Data Structure Fields	Type	Description
cal_mode	uint8_t	Magnetometer calibration mode: HT_MAG_CAL_MODE_NONE, HT_MAG_CAL_MODE_STATIC, HT_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC, HT_MAG_CAL_MODE_CONT_AUTO_BASIC, HT_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART, HT_MAG_CAL_MODE_CONT_AUTO_SMART
q_format	uint8_t	Calibration coefficients Q format.
offset_vector	int16_t[3]	Offset coefficients.
matrix	int16_t[3][3]	Matrix coefficients.

ht_calibr_ctrl_t

Data Structure Fields	Type	Description
cal_mode	uint8_t	Magnetometer calibration mode: HT_MAG_CAL_MODE_NONE, HT_MAG_CAL_MODE_STATIC, HT_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC, HT_MAG_CAL_MODE_CONT_AUTO_BASIC, HT_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART, HT_MAG_CAL_MODE_CONT_AUTO_SMART
flags	uint16_t	Calibration control flags.
ref_mag	uint16_t	Calibration control param 1.
mag_range	uint16_t	Calibration control param 2.
mag_alpha	uint16_t	Calibration control param 3.
mag_delta_thresh	uint16_t	Calibration control param 4.
mu_offset	uint16_t	Calibration control param 5.
mu_matrix	uint16_t	Calibration control param 6.
err_alpha	uint16_t	Calibration control param 7.
err_thresh	uint16_t	Calibration control param 8.

ht_acc_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable accelerometer sensor: HT_DISABLE, HT_ENABLE
range	uint8_t	Accelerometer sensor data range: HT_ACCEL_RANGE_2G, HT_ACCEL_RANGE_4G, HT_ACCEL_RANGE_8G, HT_ACCEL_RANGE_16G

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
Rate	uint8_t	Accelerometer sensor data rate: HT_ACCEL_DATA_RATE_INVALID, HT_ACCEL_DATA_RATE_0_78HZ, HT_ACCEL_DATA_RATE_1_56HZ, HT_ACCEL_DATA_RATE_3_12HZ, HT_ACCEL_DATA_RATE_6_25HZ, HT_ACCEL_DATA_RATE_12_5HZ, HT_ACCEL_DATA_RATE_25HZ, HT_ACCEL_DATA_RATE_50HZ, HT_ACCEL_DATA_RATE_100HZ

ht_gyr_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable gyroscope sensor: HT_DISABLE, HT_ENABLE
range	uint8_t	Gyroscope sensor data range: HT_GYRO_RANGE_2000_DEG_SEC, HT_GYRO_RANGE_1000_DEG_SEC, HT_GYRO_RANGE_500_DEG_SEC, HT_GYRO_RANGE_250_DEG_SEC, HT_GYRO_RANGE_125_DEG_SEC
rate	uint8_t	Gyroscope sensor data rate: HT_GYRO_DATA_RATE_INVALID, HT_GYRO_DATA_RATE_25HZ, HT_GYRO_DATA_RATE_50HZ, HT_GYRO_DATA_RATE_100HZ

ht_mag_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable magnetometer sensor: HT_DISABLE, HT_ENABLE
rate	uint8_t	Magnetometer sensor data rate: HT_MAG_DATA_RATE_INVALID, HT_MAG_DATA_RATE_0_78HZ, HT_MAG_DATA_RATE_1_56HZ, HT_MAG_DATA_RATE_3_12HZ, HT_MAG_DATA_RATE_6_25HZ, HT_MAG_DATA_RATE_12_5HZ, HT_MAG_DATA_RATE_25HZ, HT_MAG_DATA_RATE_50HZ, HT_MAG_DATA_RATE_100HZ
cal_mode	uint8_t	Magnetometer sensor calibration mode: HT_MAG_CAL_MODE_NONE, HT_MAG_CAL_MODE_STATIC, HT_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC, HT_MAG_CAL_MODE_CONT_AUTO_BASIC, HT_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART, HT_MAG_CAL_MODE_CONT_AUTO_SMART

© 2017 Dialog Semiconductor

ht_env_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable environmental sensors: HT_DISABLE, HT_ENABLE
rate	uint32_t	Environmental sensors data rate: HT_ENV_DATA_RATE_0_125HZ, HT_ENV_DATA_RATE_0_25HZ, HT_ENV_DATA_RATE_0_5HZ, HT_ENV_DATA_RATE_1HZ, HT_ENV_DATA_RATE_2HZ, HT_ENV_DATA_RATE_4HZ, HT_ENV_DATA_RATE_8HZ, HT_ENV_DATA_RATE_DISABLE

ht_sc_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable step counting service: HT_DISABLE, HT_ENABLE
rate	uint32_t	Step counting service data rate: HT_SC_DATA_RATE_0_125HZ, HT_SC_DATA_RATE_0_25HZ, HT_SC_DATA_RATE_0_5HZ, HT_SC_DATA_RATE_1HZ, HT_SC_DATA_RATE_2HZ, HT_SC_DATA_RATE_4HZ, HT_SC_DATA_RATE_8HZ, HT_SC_DATA_RATE_DISABLE

ht_sf_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable sensor fusion service: HT_DISABLE, HT_ENABLE
mode	uint8_t	Sensor fusion mode: HT_SF_MODE_GYR, HT_SF_MODE_ACC_GYR, HT_SF_MODE_ACC_MAG, HT_SF_MODE_ACC_GYR_MAG
rate	uint8_t	Sensor fusion service data rate: HT_SF_DATA_RATE_INVALID, HT_SF_DATA_RATE_0_78HZ, HT_SF_DATA_RATE_1_56HZ, HT_SF_DATA_RATE_3_12HZ, HT_SF_DATA_RATE_6_25HZ, HT_SF_DATA_RATE_12_5HZ, HT_SF_DATA_RATE_25HZ, HT_SF_DATA_RATE_50HZ
coeffs	ht_sf_coeffs_t	Sensor fusion coefficients.

© 2017 Dialog Semiconductor

ht_hr_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable heart rate estimation service: HT_DISABLE, HT_ENABLE
mode	uint8_t	Heart rate estimation mode: HT_HR_MODE_REGULAR, HT_HR_MODE_INTERMITTENT
rate	uint32_t	Heart rate estimation service data rate: HT_HR_DATA_RATE_1_PER_60SEC_HZ, HT_HR_DATA_RATE_1_PER_30SEC_HZ, HT_HR_DATA_RATE_1_PER_15SEC_HZ, HT_HR_DATA_RATE_1_PER_7SEC_HZ, HT_HR_DATA_RATE_1_PER_3SEC_HZ, HT_HR_DATA_RATE_1_PER_1SEC_HZ, HT_HR_DATA_RATE_DISABLE

ht_sq_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable sleep quality monitoring service: HT_DISABLE, HT_ENABLE

ht_cc_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable calories counting service: HT_DISABLE, HT_ENABLE
mode	uint8_t	Calories counting mode: HT_CC_MODE_REGULAR, HT_CC_MODE_ROBUST
rate	uint8_t	Calories counting service data rate: HT_CC_DATA_RATE_1_PER_5MIN_HZ, HT_CC_DATA_RATE_1_PER_4MIN_HZ, HT_CC_DATA_RATE_1_PER_3MIN_HZ, HT_CC_DATA_RATE_1_PER_2MIN_HZ, HT_CC_DATA_RATE_1_PER_1MIN_HZ, HT_CC_DATA_RATE_1_PER_50SEC_HZ, HT_CC_DATA_RATE_1_PER_40SEC_HZ, HT_CC_DATA_RATE_1_PER_30SEC_HZ, HT_CC_DATA_RATE_1_PER_20SEC_HZ, HT_CC_DATA_RATE_1_PER_10SEC_HZ, HT_CC_DATA_RATE_DISABLE

ht_bs_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable body state classification service: HT_DISABLE, HT_ENABLE

ht_s1_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable (template) s1 service: HT_DISABLE, HT_ENABLE

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
mode	uint8_t	(Template) s1 mode: HT_S1_MODE_TYPE_MODE_1, HT_S1_MODE_TYPE_MODE_2
rate	uint8_t	(Template) s1 service data rate: HT_S1_DATA_RATE_INVALID, HT_S1_DATA_RATE_0_78HZ, HT_S1_DATA_RATE_1_56HZ, HT_S1_DATA_RATE_3_12HZ, HT_S1_DATA_RATE_6_25HZ, HT_S1_DATA_RATE_12_5HZ, HT_S1_DATA_RATE_25HZ, HT_S1_DATA_RATE_50HZ, HT_S1_DATA_RATE_100HZ
acc_range	uint8_t	(Template) s1 accelerometer (as input) sensor data range: HT_ACCEL_RANGE_2G, HT_ACCEL_RANGE_4G, HT_ACCEL_RANGE_8G, HT_ACCEL_RANGE_16G
acc_rate	uint8_t	(Template) s1 accelerometer (as input) sensor data rate: HT_ACCEL_DATA_RATE_INVALID, HT_ACCEL_DATA_RATE_0_78HZ, HT_ACCEL_DATA_RATE_1_56HZ, HT_ACCEL_DATA_RATE_3_12HZ, HT_ACCEL_DATA_RATE_6_25HZ, HT_ACCEL_DATA_RATE_12_5HZ, HT_ACCEL_DATA_RATE_25HZ, HT_ACCEL_DATA_RATE_50HZ, HT_ACCEL_DATA_RATE_100HZ
gyr_range	uint8_t	(Template) s1 gyroscope (as input) sensor data range: HT_GYRO_RANGE_2000_DEG_SEC, HT_GYRO_RANGE_1000_DEG_SEC, HT_GYRO_RANGE_500_DEG_SEC, HT_GYRO_RANGE_250_DEG_SEC, HT_GYRO_RANGE_125_DEG_SEC
gyr_rate	uint8_t	(Template) s1 gyroscope (as input) sensor data rate: HT_GYRO_DATA_RATE_INVALID, HT_GYRO_DATA_RATE_25HZ, HT_GYRO_DATA_RATE_50HZ, HT_GYRO_DATA_RATE_100HZ
mag_rate	uint8_t	(Template) s1 magnetometer (as input) sensor data rate: HT_MAG_DATA_RATE_INVALID, HT_MAG_DATA_RATE_0_78HZ, HT_MAG_DATA_RATE_1_56HZ, HT_MAG_DATA_RATE_3_12HZ, HT_MAG_DATA_RATE_6_25HZ, HT_MAG_DATA_RATE_12_5HZ, HT_MAG_DATA_RATE_25HZ, HT_MAG_DATA_RATE_50HZ, HT_MAG_DATA_RATE_100HZ

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
gender	uint8_t	Calories counting user's gender type (m/f): HT_UPF_GENDER_TYPE_FEMALE, HT_UPF_GENDER_TYPE_MALE
height	uint8_t	Calories counting user's height (in cm).
weight	uint8_t	Calories counting user's weight (in kg).

ht_upf_t

Type Definition Name	Description
ht_upf_t	Configuration state structure for user profile attributes: ht_upf_config_t

ht_acc_t

Type Definition Name	Description
ht_acc_t	Configuration state structure for accelerometer sensor: ht_acc_config_t

ht_gyr_t

Type Definition Name	Description
ht_gyr_t	Configuration state structure for gyroscope sensor: ht_gyr_config_t

ht_mag_t

Type Definition Name	Description
ht_mag_t	Configuration state structure for magnetometer sensor: ht_mag_config_t

ht_env_t

Type Definition Name	Description
ht_env_t	Configuration state structure for environmental sensors: ht_env_config_t

ht_sc_t

Type Definition Name	Description
ht_sc_t	Configuration state structure for step counting service: ht_sc_config_t

ht_sf_t

Type Definition Name	Description
ht_sf_t	Configuration state structure for sensor fusion setvice: ht_sf_config_t

© 2017 Dialog Semiconductor

ht_hr_t

Type Definition Name	Description
ht_hr_t	Configuration state structure for heart rate estimation service: ht_hr_config_t

ht_sq_t

Type Definition Name	Description
ht_sq_t	Configuration state structure for sleep quality monitoring service: ht_sq_config_t

ht_cc_t

Type Definition Name	Description
ht_cc_t	Configuration state structure for calories counting service: ht_cc_config_t

ht_bs_t

Type Definition Name	Description
ht_bs_t	Configuration state structure for body state classification service: ht_bs_config_t

ht_s1_t

Type Definition Name	Description
ht_s1_t	Configuration state structure for (template) s1 service: ht_s1_config_t

ht_handle_t

Data Structure Fields	Type	Description
ht_clear_done	Function pointer	Function for clear-done indication.
ht_start_done	Function pointer	Function for start-done indication.
ht_stop_done	Function pointer	Function for stop-done indication.
ht_config_done	Function pointer	Function for config-done indication.
ht_state_get_done	Function pointer	Function for state-get-done indication.
ht_calibr_done	Function pointer	Function for calibr-done-done indication.
ht_data_read_done	Function pointer	Function for data-read-done indication.
ht_data_reset_done	Function pointer	Function for data-reset-done indication.
ht_data_indication	Function pointer	Function for data indication.
ht_error	Function pointer	Function for error indication.
ht_get_rtc_notifs_config	Function pointer	Function for acquiring RTC availability and interval.

ht_data_acc_t

Data Structure Fields	Type	Description
xyz.x	int16_t	Accelerometer x-axis sensor data value.

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
xyz.y	int16_t	Accelerometer y-axis sensor data value.
xyz.z	int16_t	Accelerometer z-axis sensor data value.
calibr_state. uncalibr_data_valid	uint8_t:1	Pre-calibrated data valid flag.
calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
calibr_state. cal_enabled	uint8_t:1	Calibration enabled flag.
calibr_state. cal_settled	uint8_t:1	Calibration settled flag.
calibr_state. reserved4	uint8_t:1	Reserved flag.
calibr_state. cal_mode	uint8_t:3	Calibration mode: 0 for NONE, 1 for STATIC

ht_data_gyr_t

Data Structure Fields	Type	Description
xyz.x	int16_t	Gyroscope x-axis sensor data value.
xyz.y	int16_t	Gyroscope y-axis sensor data value.
xyz.z	int16_t	Gyroscope z-axis sensor data value.

ht_data_mag_t

Data Structure Fields	Type	Description
xyz.x	int16_t	Magnetometer x-axis sensor data value.
xyz.y	int16_t	Magnetometer y-axis sensor data value.
xyz.z	int16_t	Magnetometer z-axis sensor data value.
calibr_state. uncalibr_data_valid	uint8_t:1	Pre-calibrated data valid flag.
calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
calibr_state. cal_enabled	uint8_t:1	Calibration enabled flag.
calibr_state. cal_settled	uint8_t:1	Calibration settled flag.
calibr_state. cal_converged	uint8_t:1	Calibration converged flag.
calibr_state. cal_mode	uint8_t:3	Calibration mode: HT_MAG_CAL_MODE_NONE. HT_MAG_CAL_MODE_STATIC, HT_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC, HT_MAG_CAL_MODE_CONT_AUTO_BASIC, HT_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART, HT_MAG_CAL_MODE_CONT_AUTO_SMART

ht_data_env_t

Data Structure Fields	Type	Description
temperature	int32_t	Temperature sensor data (in 0.01 degrees Celsius).
pressure	uint32_t	Pressure sensor data (in Pa).

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
humidity	uint32_t	Humidity sensor data (in %RH as unsigned 32-bit integer in Q22.10 format (22 integer 10 fractional bits)).

ht_data_sc_t

Data Structure Fields	Type	Description
step_count	uint32_t	Number of steps counted.

ht_data_sf_t

Data Structure Fields	Type	Description
quaternion_w	int32_t	Sensor fusion service data quartenion w.
quaternion_x	int32_t	Sensor fusion service data quartenion x.
quaternion_y	int32_t	Sensor fusion service data quartenion y.
quaternion_z	int32_t	Sensor fusion service data quartenion z.
mag_calibr_state. uncalibr_data_valid	uint8_t:1	Pre-calibrated data valid flag.
mag_calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
mag_calibr_state. cal_enabled	uint8_t:1	Calibration enabled flag.
mag_calibr_state. cal_settled	uint8_t:1	Calibration settled flag.
mag_calibr_state. cal_converged	uint8_t:1	Calibration converged flag.
calibr_state. cal_mode	uint8_t:3	Calibration mode: HT_MAG_CAL_MODE_NONE. HT_MAG_CAL_MODE_STATIC, HT_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC, HT_MAG_CAL_MODE_CONT_AUTO_BASIC, HT_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART, HT_MAG_CAL_MODE_CONT_AUTO_SMART

ht_data_hr_t

Data Structure Fields	Type	Description
heart_rate	uint8_t	Heart rate (in bpm).

ht_data_sq_t

Data Structure Fields	Type	Description
invalid	int32_t	Number of consecutive epochs classified as INVALID.
awake	int32_t	Number of consecutive epochs classified as AWAKE.
light_sleep	int32_t	Number of consecutive epochs classified as LIGHT SLEEP.
deep_sleep	int32_t	Number of consecutive epochs classified as DEEP SLEEP.
rem	int32_t	Number of consecutive epochs classified as REM.
error	int32_t	Number of consecutive epochs classified as ERROR.

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
epoch	uint32_t	First epoch number sleep quality monitoring data refer to.

ht_data_cc_t

Data Structure Fields	Type	Description
calories	int32_t	Amount of calories burned (in cal).
distance	int32_t	Distance covered (in cm).
num_of_epochs	int32_t	Number of epochs included.
epoch	uint32_t	First epoch number calories counting data refer to.

ht_data_bs_t

Data Structure Fields	Type	Description
body_state	uint8_t	Body state classification value: HT_BODY_STATE_OTHER, HT_BODY_STATE_WALK, HT_BODY_STATE_RUN, HT_BODY_STATE_SIT, HT_BODY_STATE_STAND, HT_BODY_STATE_LAYING_DOWN

ht_data_s1_t

Data Structure Fields	Type	Description
dataVal1	int16_t	(Template) s1 service dataVal1 value.
dataVal2	int16_t	(Template) s1 service dataVal2 value.
dataVal3	int32_t	(Template) s1 service dataVal3 value.

© 2017 Dialog Semiconductor

8.1.2 Dependency Functions

ht_handle_t -> ht_clear_done()

Function Name	`void (*ht_clear_done)(ht_error_t error)`
Function Description	(Optional) Handles indication that Health Toolbox Task has been cleared.
Parameters	`error` the execution status of 'clear' operation `HT_SUCCESS, ≥ HT_FAIL`
Return Values	None
Notes	

ht_handle_t -> ht_start_done()

Function Name	`void (*ht_start_done)(ht_error_t error)`
Function Description	(Optional) Handles indication that Health Toolbox Task has been started.
Parameters	`error` the execution status of 'start' operation `HT_SUCCESS, ≥ HT_FAIL`
Return Values	None
Notes	

ht_handle_t -> ht_stop_done()

Function Name	`void (*ht_stop_done)(ht_error_t error)`
Function Description	(Optional) Handles indication that Health Toolbox Task has been stopped.
Parameters	`error` the execution status of 'stop' operation `HT_SUCCESS, ≥ HT_FAIL`
Return Values	None
Notes	

ht_handle_t -> ht_config_done()

Function Name	`void (*ht_config_done)(uint8_t type, ht_error_t error)`
Function Description	(Optional) Handles indication that Health Toolbox Task has been configured in terms of a specific sensor or service.

Parameters	`type` the sensor or service type
	`HT_CONFIG_UPF , HT_CONFIG_SF_COEFFS,`
	`HT_CONFIG_CALIBR_COEFFS_MAG , HT_CONFIG_CALIBR_CONTROL_MAG,`
	`HT_CONFIG_CALIBR_OFFSETS_ACC , HT_CONFIG_CALIBR_OFFSETS_ACC_UPDATE,`
	`HT_CONFIG_ACC , HT_CONFIG_GYR,`
	`HT_CONFIG_MAG , HT_CONFIG_ENV,`
	`HT_CONFIG_SC , HT_CONFIG_SF,`
	`HT_CONFIG_HR , HT_CONFIG_SQ,`
	`HT_CONFIG_CC , HT_CONFIG_BS,`
	`HT_CONFIG_S1`
	`error` the execution status of 'config' operation
	`HT_SUCCESS, ≥ HT_FAIL`

Return Values	None
Notes	

© 2017 Dialog Semiconductor

ht_handle_t -> ht_state_get_done()

Function Name	`void (*ht_state_get_done)(uint8_t type, void* state, ht_error_t error)`
Function Description	(Optional) Handles indication that the configuration state of a specific sensor or service in Health Toolbox component has been acquired.

Parameters	`type` the sensor or service type
	`HT_STATE_UPF , HT_STATE_BASIC_ATTRS,`
	`HT_STATE_SF_COEFFS , HT_STATE_CALIBR_COEFFS_MAG,`
	`HT_STATE_CALIBR_CONTROL_MAG, HT_STATE_CALIBR_OFFSETS_ACC,`
	`HT_STATE_ACC, HT_STATE_GYR , HT_STATE_MAG,`
	`HT_STATE_ENV, HT_STATE_SC , HT_STATE_SF,`
	`HT_STATE_HR , HT_STATE_SQ , HT_STATE_CC,`
	`HT_STATE_BS , HT_STATE_S1`
	`state` the configuration state of the sensor or service
	`error` the execution status of 'state-get' operation
	`HT_SUCCESS, ≥ HT_FAIL`

Return Values	None
Notes	

ht_handle_t -> ht_calibr_done()

Function Name	`void (*ht_calibr_done)(uint8_t type, void* data, ht_error_t error)`
Function Description	(Optional) Handles indication that the calibration process of a specific sensor in Health Toolbox component has been completed.

Parameters	`type` the sensor type
	`HT_CALIBR_OFFSETS_ACC,`
	`HT_CALIBR_CONTROL_MAG,`
	`HT_CALIBR_COEFFS_MAG`
	`data` the calibration data for the sensor
	`error` the execution status of calibration operation
	`HT_SUCCESS, ≥ HT_FAIL`

Return Values	None
Notes	

ht_handle_t -> ht_data_read_done()

Function Name	`void (*ht_data_read_done)(uint8_t type, void* data, ht_error_t error)`
Function Description	(Optional) Handles indication that reading data of a specific sensor or service in Health Toolbox component has been completed.

Parameters	`type` the sensor or service type
	`HT_SENSOR_TYPE_ACC , HT_SENSOR_TYPE_GYR , HT_SENSOR_TYPE_MAG,`
	`HT_SENSOR_TYPE_ENV , HT_SERVICE_TYPE_SC , HT_SERVICE_TYPE_SF,`
	`HT_SERVICE_TYPE_HR , HT_SERVICE_TYPE_HRI,`
	`HT_SERVICE_TYPE_SQ , HT_SERVICE_TYPE_SQU, HT_SERVICE_TYPE_SQT,`
	`HT_SERVICE_TYPE_CC , HT_SERVICE_TYPE_CCT,`
	`HT_SERVICE_TYPE_BS , HT_SERVICE_TYPE_S1`
	`data` the acquired sensor or service data
	`error` the execution status of 'data-read' operation
	`HT_SUCCESS, ≥ HT_FAIL`

Return Values	None
Notes	

© 2017 Dialog Semiconductor

ht_handle_t -> ht_data_reset_done()

Function Name	`void (*ht_data_reset_done)(uint8_t type, ht_error_t error)`
Function Description	(Optional) Handles indication that resetting data of a specific sensor or service in Health Toolbox component has been completed.

Parameters	`type` the sensor or service type
	`HT_SERVICE_TYPE_SC, HT_SERVICE_TYPE_SQT, HT_SERVICE_TYPE_CCT`
	`error` the execution status of 'data-reset' operation
	`HT_SUCCESS, ≥ HT_FAIL`

Return Values	None
Notes	

ht_handle_t -> ht_data_indication()

Function Name	`void (*ht_data_indication)(uint8_t type, void* data, ht_error_t error)`
Function Description	(Optional) Handles indication that new data of a specific sensor or service in Health Toolbox component have been produced.

Parameters	`type` the sensor or service type
	`HT_SENSOR_TYPE_ACC, HT_SENSOR_TYPE_GYR, HT_SENSOR_TYPE_MAG,`
	`HT_SENSOR_TYPE_ENV, HT_SERVICE_TYPE_SC, HT_SERVICE_TYPE_SF,`
	`HT_SERVICE_TYPE_HR, HT_SERVICE_TYPE_HRI,`
	`HT_SERVICE_TYPE_SQ, HT_SERVICE_TYPE_SQU, HT_SERVICE_TYPE_SQT,`
	`HT_SERVICE_TYPE_CC, HT_SERVICE_TYPE_CCT,`
	`HT_SERVICE_TYPE_BS, HT_SERVICE_TYPE_S1`
	`data` the produced/notified sensor or service data
	`error` the execution status of new data indication operation
	`HT_SUCCESS, ≥ HT_FAIL`

Return Values	None
Notes	

ht_handle_t -> ht_error()

Function Name	`void (*ht_error)(uint8_t type, ht_error_t error)`
Function Description	(Optional) Handles indication of an error in Health Toolbox component being produced.

© 2017 Dialog Semiconductor

Parameters	`type` the error type
	`HT_ACQ_STATE_ERROR,`
	`HT_INIT_ERROR,`
	`HT_CONFIG_TYPE_ERROR,`
	`HT_STATE_GET_TYPE_ERROR,`
	`HT_CALIBR_TYPE_ERROR,`
	`HT_FORCE_READ_DATA_TYPE_ERROR,`
	`HT_RESET_DATA_TYPE_ERROR,`
	`HT_PROCESS_SERVICE_DATA_SF_ERROR,`
	`HT_PROCESS_SERVICE_DATA_SF_TYPE_ERROR,`
	`HT_PROCESS_SERVICE_DATA_HR_ERROR,`
	`HT_PROCESS_SERVICE_DATA_HR_TYPE_ERROR,`
	`HT_PROCESS_SERVICE_DATA_SQ_ERROR,`
	`HT_PROCESS_SERVICE_DATA_SQ_TYPE_ERROR,`
	`HT_PROCESS_SERVICE_DATA_CC_ERROR,`
	`HT_PROCESS_SERVICE_DATA_CC_TYPE_ERROR,`
	`HT_PROCESS_SERVICE_DATA_BS_TYPE_ERROR,`
	`HT_PROCESS_SERVICE_DATA_S1_TYPE_ERROR,`
	`HT_CATCH_INTR_1_ERROR,`
	`HT_HEALTH_CARE_SERVICES_CONFIG_REQ_ERROR,`
	`HT_KIWI_SERVICES_CONFIG_REQ_ERROR,`
	`HT_OTHER_SERVICES_CONFIG_REQ_ERROR,`
	`HT_TASK_CONTEXT_REQ_ERROR,`
	`HT_OS_TASKNOTIFY_TASK_CONTEXT_REQ_ERROR,`
	`HT_OS_TASKNOTIFY_STATE_CHANGE_TIME_ERROR,`
	`HT_OS_TASKNOTIFY_STATE_GET_ERROR,`
	`HT_OS_TASKNOTIFY_CALIBR_ERROR,`
	`HT_OS_TASKNOTIFY_FORCE_READ_DATA_ERROR,`
	`HT_OS_TASKNOTIFY_RESET_DATA_ERROR,`
	`HT_OS_TASKNOTIFY_CONTROL_ERROR,`
	`HT_OS_TASKNOTIFY_PROCESS_SERVICE_DATA_SF_ERROR,`
	`HT_OS_TASKNOTIFY_PROCESS_SERVICE_DATA_HR_ERROR,`
	`HT_OS_TASKNOTIFY_PROCESS_SERVICE_DATA_SQ_ERROR,`
	`HT_OS_TASKNOTIFY_PROCESS_SERVICE_DATA_CC_ERROR,`
	`HT_OS_TASKNOTIFY_PROCESS_SERVICE_DATA_BS_ERROR,`
	`HT_OS_TASKNOTIFY_PROCESS_SERVICE_DATA_S1_ERROR`
	`error` the execution status in Health Toolbox component
	`HT_SUCCESS, ≥ HT_FAIL`

Return Values	`None`
Notes	

ht_handle_t -> ht_get_rtc_notifs_config()

Function Name	`void (*ht_get_rtc_notifs_config)(bool* enable, uint32_t* interval)`
Function Description	(Optional) Acquires RTC configuration, i.e. availability/status and interval.
Parameters	`enable` the status of RTC operation `HT_DISABLE, HT_ENABLE` `interval` the (periodic) interval of RTC ticks (in ms)
Return Values	`None`
Notes	

8.1.3 Initialization and Control Functions

ht_init()

Function Name	`ht_error_t ht_init(ht_handle_t* handle)`

 © 2017 Dialog Semiconductor

Function Description	Initializes Health Toolbox, setting initial values to the internal variables and structures, creating a FreeRTOS task for control coordination and data manipulation, and initializing services, sensors (putting to suspend mode) and other processes being supported. Function callbacks may be set to the `ht_handle_t` instance handle, so that control and data indications can be sent.
Parameters	`handle` the handle structure of Health Toolbox
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	All callback functions defined in `ht_handle_t` structure are optional.

ht_get_handle()

Function Name	`ht_handle_t* ht_get_handle(void)`
Function Description	Gets the handle structure of Health Toolbox.
Parameters	None
Return Values	Pointer to the handle structure of Health Toolbox.
Notes	

ht_clear()

Function Name	`ht_error_t ht_clear(void)`
Function Description	Clears Health Toolbox, freeing previously allocated resources. If the respective function pointer is defined in `ht_handle_t` instance handle, a clear-done indication is sent via `ht_clear_done()` callback function.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_start()

Function Name	`ht_error_t ht_start(void)`
Function Description	Starts Health Toolbox, configuring with default parameter values the services being supported. If the respective function pointer is defined in `ht_handle_t` instance handle, a start-done indication is sent via `ht_start_done()` callback function.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_stop()

Function Name	`ht_error_t ht_stop(void)`
Function Description	Stops Health Toolbox, disabling all services being supported. If the respective function pointer is defined in `ht_handle_t` instance handle, a stop-done indication is sent via `ht_stop_done()` callback function.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_rtc_send_notif()

© 2017 Dialog Semiconductor

Function Name	`ht_error_t ht_rtc_send_notif(void)`
Function Description	Sends RTC tick notification to Health Toolbox. If RTC interval is not appropriate for Health Toolbox operation, RTC tick notification is ignored.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

8.1.3.2 Configuration Functions

ht_config_upf()

Function Name	`ht_error_t ht_config_upf(ht_upf_config_t* config)`
Function Description	Configures user profile attributes. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_UPF`.
Parameters	`config` the configuration structure of user profile attributes
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_sf_coeffs()

Function Name	`ht_error_t ht_config_sf_coeffs(ht_sf_coeffs_t* config)`
Function Description	Configures sensor fusion coefficients. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_SF_COEFFS`.
Parameters	`config` the configuration structure of sensor fusion coefficients
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_calibr_coeffs_mag()

Function Name	`ht_error_t ht_config_calibr_coeffs_mag(ht_calibr_coeffs_t* config)`
Function Description	Configures magnetometer calibration coefficients. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_CALIBR_COEFFS_MAG`.
Parameters	`config` the configuration structure of magnetometer calibration coefficients
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_calibr_control_mag()

Function Name	`ht_error_t ht_config_calibr_control_mag(ht_calibr_ctrl_t* config)`

© 2017 Dialog Semiconductor

Function Description	Configures magnetometer calibration control. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_CALIBR_CONTROL_MAG`. A calibration-control-changed indication is sent via `ht_calibr_done()` callback function, setting parameter type to `HT_CALIBR_CONTROL_MAG`. A calibration-done indication is sent via `ht_calibr_done()` callback function, setting parameter type to `HT_CALIBR_COEFFS_MAG`.
Parameters	`config` the configuration structure of magnetometer calibration control
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_calibr_offsets_acc()

Function Name	`ht_error_t ht_config_calibr_offsets_acc(ht_calibr_offsets_t* config)`
Function Description	Configures accelerometer calibration offsets. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_CALIBR_OFFSETS_ACC`.
Parameters	`config` the configuration structure of accelerometer calibration offsets
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_calibr_offsets_acc_update()

Function Name	`ht_error_t ht_config_calibr_offsets_acc_update(void)`
Function Description	Updates accelerometer calibration offsets. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_CALIBR_OFFSETS_ACC_UPDATE`. A calibration-done indication is sent via `ht_calibr_done()` callback function, setting parameter type to `HT_CALIBR_OFFSETS_ACC`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_acc()

Function Name	`ht_error_t ht_config_acc(ht_acc_config_t* config)`
Function Description	Configures accelerometer sensor. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_ACC`. Accelerometer data indications are sent via `ht_data_indication()` callback function, setting parameter type to `HT_SENSOR_TYPE_ACC`.
Parameters	`config` the configuration structure of accelerometer sensor
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_gyr()

© 2017 Dialog Semiconductor

Function Name	`ht_error_t ht_config_gyr(ht_gyr_config_t* config)`
Function Description	Configures gyroscope sensor. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_GYR`. Gyroscope data indications are sent via `ht_data_indication()` callback function, setting parameter type to `HT_SENSOR_TYPE_GYR`.
Parameters	`config` the configuration structure of gyroscope sensor
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_mag()

Function Name	`ht_error_t ht_config_mag(ht_mag_config_t* config)`
Function Description	Configures magnetometer sensor. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_MAG`. Magnetometer data indications are sent via `ht_data_indication()` callback function, setting parameter type to `HT_SENSOR_TYPE_MAG`.
Parameters	`config` the configuration structure of magnetometer sensor
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_env()

Function Name	`ht_error_t ht_config_env(ht_env_config_t* config)`
Function Description	Configures environmental sensors. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_ENV`. Environmental data indications are sent via `ht_data_indication()` callback function, setting parameter type to `HT_SENSOR_TYPE_ENV`.
Parameters	`config` the configuration structure of environmental sensors
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_sc()

Function Name	`ht_error_t ht_config_sc(ht_sc_config_t* config)`
Function Description	Configures step counting service. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_SC`. Step counting service data indications are sent via `ht_data_indication()` callback function, setting parameter type to `HT_SERVICE_TYPE_SC`.
Parameters	`config` the configuration structure of step counting service
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_sf()

Function Name	`ht_error_t ht_config_sf(ht_sf_config_t* config)`

© 2017 Dialog Semiconductor

Function Description	Configures sensor fusion service. If the respective function pointer is defined in ht_handle_t instance handle, a config-done indication is sent via ht_config_done() callback function, setting parameter type to HT_CONFIG_SF. Sensor fusion service data indications are sent via ht_data_indication() callback function, setting parameter type to HT_SERVICE_TYPE_SF.
Parameters	config the configuration structure of sensor fusion service
Return Values	HT_SUCCESS for successful execution; ≥ HT_FAIL for error
Notes	

ht_config_hr()

Function Name	ht_error_t ht_config_hr(ht_hr_config_t* config)
Function Description	Configures heart rate estimation service. If the respective function pointer is defined in ht_handle_t instance handle, a config-done indication is sent via ht_config_done() callback function, setting parameter type to HT_CONFIG_HR. Heart rate estimation service data indications are sent via ht_data_indication() callback function, setting parameter type to HT_SERVICE_TYPE_HR.
Parameters	config the configuration structure of heart rate estimation service
Return Values	HT_SUCCESS for successful execution; ≥ HT_FAIL for error
Notes	

ht_config_sq()

Function Name	ht_error_t ht_config_sq(ht_sq_config_t* config)
Function Description	Configures sleep quality monitoring service. If the respective function pointer is defined in ht_handle_t instance handle, a config-done indication is sent via ht_config_done() callback function, setting parameter type to HT_CONFIG_SQ. Sleep quality monitoring service data indications are sent via ht_data_indication() callback function, setting parameter type to HT_SERVICE_TYPE_SQ.
Parameters	config the configuration structure of slep quality monitoring service
Return Values	HT_SUCCESS for successful execution; ≥ HT_FAIL for error
Notes	

ht_config_cc()

Function Name	ht_error_t ht_config_cc(ht_cc_config_t* config)
Function Description	Configures calories counting service. If the respective function pointer is defined in ht_handle_t instance handle, a config-done indication is sent via ht_config_done() callback function, setting parameter type to HT_CONFIG_CC. Calories counting service data indications are sent via ht_data_indication() callback function, setting parameter type to HT_SERVICE_TYPE_CC.
Parameters	config the configuration structure of calories counting service
Return Values	HT_SUCCESS for successful execution; ≥ HT_FAIL for error
Notes	

ht_config_bs()

Function Name	ht_error_t ht_config_bs(ht_bs_config_t* config)

© 2017 Dialog Semiconductor

Function Description	Configures body state classification service. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to `HT_CONFIG_BS`. Body state classification service data indications are sent via `ht_data_indication()` callback function, setting parameter type to `HT_SERVICE_TYPE_BS`.
Parameters	`config` the configuration structure of body state classification service
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_config_s1()

Function Name	`ht_error_t ht_config_s1(ht_s1_config_t* config)`
Function Description	Configures (template) s1 service. If the respective function pointer is defined in `ht_handle_t` instance handle, a config-done indication is sent via `ht_config_done()` callback function, setting parameter type to HT_CONFIG_S1. (Template) s1 service data indications are sent via `ht_data_indication()` callback function, setting parameter type to `HT_SERVICE_TYPE_S1`.
Parameters	`config` the configuration structure of (template) s1 service
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

8.1.3.3 State Functions

ht_isrunning()

Function Name	`bool ht_isrunning(void)`
Function Description	Checks if Health Toolbox task is running.
Parameters	None
Return Values	`true` if running; false if not running
Notes	

ht_time_get_now_ms()

Function Name	`ht_error_t ht_time_get_now_ms(uint32_t* now)`
Function Description	Gets local time indicated by a sensor module clock in ms.
Parameters	`now` the address to store the local time
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_upf()

Function Name	`ht_error_t ht_state_get_upf(void)`
Function Description	Gets configuration status of user profile attributes. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_UPF`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error

© 2017 Dialog Semiconductor

Notes	

ht_state_get_sf_coeffs()

Function Name	`ht_error_t ht_state_get_sf_coeffs(void)`
Function Description	Gets configuration status of sensor fusion coefficients. If the respective function pointer is defined in ht_handle_t instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_SF_COEFFS`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_calibr_coeffs_mag()

Function Name	`ht_error_t ht_state_get_calibr_coeffs_mag(uint8_t cal_mode)`
Function Description	Gets configuration status of magnetometer calibration coefficients for a specific calibration mode. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_CALIBR_COEFFS_MAG`.
Parameters	`cal_mode` the calibration mode to which calibration coefficients refer `HT_MAG_CAL_MODE_NONE`, `HT_MAG_CAL_MODE_STATIC`, `HT_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC`, `HT_MAG_CAL_MODE_CONT_AUTO_BASIC`, `HT_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART`, `HT_MAG_CAL_MODE_CONT_AUTO_SMART`
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_calibr_control_mag()

Function Name	`ht_error_t ht_state_get_calibr_control_mag(uint8_t cal_mode)`
Function Description	Gets configuration status of magnetometer calibration control for a specific calibration mode. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_CALIBR_CONTROL_MAG`.
Parameters	`cal_mode` the calibration mode to which calibration control refers `HT_MAG_CAL_MODE_NONE`, `HT_MAG_CAL_MODE_STATIC`, `HT_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC`, `HT_MAG_CAL_MODE_CONT_AUTO_BASIC`, `HT_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART`, `HT_MAG_CAL_MODE_CONT_AUTO_SMART`
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_acc()

Function Name	`ht_error_t ht_state_get_acc(void)`

© 2017 Dialog Semiconductor

Function Description	Gets configuration status of accelerometer sensor. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_ACC`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_gyr()

Function Name	`ht_error_t ht_state_get_gyr(void)`
Function Description	Gets configuration status of gyroscope sensor. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_GYR`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_mag()

Function Name	`ht_error_t ht_state_get_mag(void)`
Function Description	Gets configuration status of magnetometer sensor. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_MAG`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_env()

Function Name	`ht_error_t ht_state_get_env(void)`
Function Description	Gets configuration status of environmental sensors. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_ENV`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_sc()

Function Name	`ht_error_t ht_state_get_sc(void)`
Function Description	Gets configuration status of step counting service. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_SC`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error

© 2017 Dialog Semiconductor

Notes	

ht_state_get_sf()

Function Name	`ht_error_t ht_state_get_sf(void)`
Function Description	Gets configuration status of sensor fusion service. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_SF`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_hr()

Function Name	`ht_error_t ht_state_get_hr(void)`
Function Description	Gets configuration status of heart rate estimation service. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_HR`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_sq()

Function Name	`ht_error_t ht_state_get_sq(void)`
Function Description	Gets configuration status of sleep quality monitoring service. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_SQ`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_cc()

Function Name	`ht_error_t ht_state_get_cc(void)`
Function Description	Gets configuration status of calories counting service. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_CC`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_bs()

Function Name	`ht_error_t ht_state_get_bs(void)`

© 2017 Dialog Semiconductor

Function Description	Gets configuration status of body state classification service. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_BS`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_state_get_s1()

Function Name	`ht_error_t ht_state_get_s1(void)`
Function Description	Gets configuration status of (template) s1 service. If the respective function pointer is defined in `ht_handle_t` instance handle, a state-get-done indication is sent via `ht_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HT_STATE_S1`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

8.1.3.4 Data Acquisition Functions

ht_data_read_acc()

Function Name	`ht_error_t ht_data_read_acc(void)`
Function Description	Reads last accelerometer data. If accelerometer sensor is enabled and the respective function pointer is defined in `ht_handle_t` instance handle, a data-read-done indication is sent via `ht_data_read_done()` callback function, carrying last accelerometer sensor data values and setting parameter type to `HT_SENSOR_TYPE_ACC`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_data_read_gyr()

Function Name	`ht_error_t ht_data_read_gyr(void)`
Function Description	Reads last gyroscope data. If gyroscope sensor is enabled and the respective function pointer is defined in `ht_handle_t` instance handle, a data-read-done indication is sent via `ht_data_read_done()` callback function, carrying last gyroscope sensor data values and setting parameter type to `HT_SENSOR_TYPE_GYR`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_data_read_mag()

Function Name	`ht_error_t ht_data_read_mag(void)`
Function Description	Reads last magnetometer data. If magnetometer sensor is enabled and the respective function pointer is defined in `ht_handle_t` instance handle, a data-read-done indication is sent via `ht_data_read_done()` callback function, carrying last magnetometer sensor data values and setting parameter type to `HT_SENSOR_TYPE_MAG`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_data_read_env()

Function Name	`ht_error_t ht_data_read_env(void)`
Function Description	Reads last (if environmental indications enabled) or force read environmental data. If the respective function pointer is defined in `ht_handle_t` instance handle, a data-read-done indication is sent via `ht_data_read_done()` callback function, carrying either last environmental sensor data values or force-read onesm depending on whether environmental sensors are periodically sampled or not, respectively, and setting parameter type to `HT_SENSOR_TYPE_ENV`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_data_read_sc()

Function Name	`ht_error_t ht_data_read_sc(void)`

© 2017 Dialog Semiconductor

Function Description	Reads last step counting data. If step counting service is enabled and the respective function pointer is defined in `ht_handle_t` instance handle, a data-read-done indication is sent via `ht_data_read_done()` callback function, carrying total step counter value and setting parameter type to `HT_SERVICE_TYPE_SC`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_data_read_sf()

Function Name	`ht_error_t ht_data_read_sf(void)`
Function Description	Reads last sensor fusion data. If sensor fusion service is enabled and the respective function pointer is defined in `ht_handle_t` instance handle, a data-read-done indication is sent via `ht_data_read_done()` callback function, carrying last sensor fusion data values and setting parameter type to `HT_SERVICE_TYPE_SF`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_data_read_hr()

Function Name	`ht_error_t ht_data_read_hr(void)`
Function Description	Reads last heart rate estimation data. If heart rate estimation service is enabled and the respective function pointer is defined in `ht_handle_t` instance handle, a data-read-done indication is sent via `ht_data_read_done()` callback function, carrying last heart rate estimation data value and setting parameter type to `HT_SERVICE_TYPE_HRI`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_data_read_sq()

Function Name	`ht_error_t ht_data_read_sq(void)`
Function Description	Reads last sleep quality monitoring data. If sleep quality monitoring service is enabled and the respective function pointer is defined in `ht_handle_t` instance handle, a data-read-done indication is sent via `ht_data_read_done()` callback function, carrying total sleep quality data values and setting parameter type to `HT_SERVICE_TYPE_SQT`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_data_read_cc()

Function Name	`ht_error_t ht_data_read_cc(void)`
Function Description	Reads last calories counting data. If calories counting service is enabled and the respective function pointer is defined in `ht_handle_t` instance handle, a data-read-done indication is sent via `ht_data_read_done()` callback function, carrying total calories data values and setting parameter type to `HT_SERVICE_TYPE_CCT`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error

© 2017 Dialog Semiconductor

Notes	

ht_data_read_bs()

Function Name	`ht_error_t ht_data_read_bs(void)`
Function Description	Reads last body state classification data. If body state classification service is enabled and the respective function pointer is defined in `ht_handle_t` instance handle, a data-read-done indication is sent via `ht_data_read_done()` callback function, carrying last body state classification data value and setting parameter type to `HT_SERVICE_TYPE_BS`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_data_read_s1()

Function Name	`ht_error_t ht_data_read_s1(void)`
Function Description	Reads last (template) s1 data. If (template) s1 service is enabled and the respective function pointer is defined in `ht_handle_t` instance handle, a data-read-done indication is sent via `ht_data_read_done()` callback function, carrying last s1 service data values and setting parameter type to `HT_SERVICE_TYPE_S1`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

8.1.3.5 Data-Reset Functions

ht_data_reset_sc()

Function Name	`ht_error_t ht_data_reset_sc(void)`
Function Description	Resets total step counter value. If the respective function pointer is defined in `ht_handle_t` instance handle, a data-reset-done indication is sent via `ht_data_reset_done()` callback function, setting parameter type to `HT_SERVICE_TYPE_SC`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_data_reset_sq()

Function Name	`ht_error_t ht_data_reset_sq(void)`
Function Description	Resets total values in sleep quality. If the respective function pointer is defined in `ht_handle_t` instance handle, a data-reset-done indication is sent via `ht_data_reset_done()` callback function, setting parameter type to `HT_SERVICE_TYPE_SQT`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

ht_data_reset_cc()

Function Name	`ht_error_t ht_data_reset_cc(void)`
Function Description	Resets total values in calories counting. If the respective function pointer is defined in `ht_handle_t` instance handle, a data-reset-done indication is sent via `ht_data_reset_done()` callback function, setting parameter type to `HT_SERVICE_TYPE_CCT`.
Parameters	None
Return Values	`HT_SUCCESS` for successful execution; ≥ `HT_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

8.2 Health Toolbox Implementations

Health Toolbox provides a list of sub-components, called “Health Toolbox implementations”, which implement the creation of multiple instances of control and data manipulation (i.e. control and data flows) over the supported/integrated sensors/services by the Health Toolbox. The source code and header files for those sub-components are located at `wrbl/health_toolbox/health_toolbox_impl`, and they refer to multiple instance control and data manipulation of:

- *inertial sensors* (`ht_impl_inertial.c|h`)
- *environmental sensors* (`ht_impl_environmental.c|h`)
- *health care sensors* (`ht_impl_healthcare.c|h`)
- *sensor fusion service* (`ht_impl_sensor_fusion.c|h`)
- *heart rate estimation service* (`ht_impl_heart_rate.c|h`)
- *step counting service* (`ht_impl_step_counting.c|h`)
- *calories counting service* (`ht_impl_calories_counting.c|h`)
- *sleep quality monitoring service* (`ht_impl_sleep_quality.c|h`)

The APIs provided by the Health Toolbox implementations include functions for initializing and controlling their operation, configuring the supported/integrated sensors/services, and finally acquiring the sensors/services' configuration state and last data. All these functions and respective data structures/types being defined in the APIs are described in the following sub-sections.

The functions and respective data structures/types being referenced by all Health Toolbox implemenations (and declared in `wrbl/health_toolbox/health_toolbox_impl/include/ht_impl.h`) are:

ht_impl_error_t

Type Definition Name	Description
`ht_impl_error_t`	Execution status of Health Toolbox Implementation functions: `HT_IMPL_SUCCESS, HT_IMPL_FAIL`

ht_impl_task_cntx_req_cb_t

Type Definition Name	Description
`ht_impl_task_cntx_req_cb_t`	Function pointer type for callback functions which are called when a task-context request is sent (i.e. `ht_impl_task_context_request()`).

ht_impl_data_ref_released_cb_t

Type Definition Name	Description
`ht_impl_data_ref_released_cb_t`	Function pointer type for callback functions indicating that a data reference has been processed, and thus released by the “consumer” process.

ht_impl_data_rdy_cb_t

Type Definition Name	Description
`ht_impl_data_rdy_cb_t`	Function pointer type for callback functions which are called when data need to be sent over a defined data flow instance of type `ht_impl_xx_handle_t` (where “xx” can be “inertial”, “env” etc.).

© 2017 Dialog Semiconductor

ht_impl_task_context_request()

Function Name	`void ht_impl_task_context_request(ht_impl_task_cntx_req_cb_t* cb, bool* notif_pending)`
Function Description	Requests (new) task context execution flow, providing the callback function through which the execution flow will be returned.
Parameters	`cb` the callback function through which the execution flow will be returned `notif_pending` the address of the variable which is used for tracking pending task context requests
Return Values	None
Notes	

8.2.2 Inertial Sensors HT-Implementation – API Specification

All functions and respective data structures/types being defined in the API are summarized below:

Table 19: Inertial Sensors HT-Implementation API

Data Structures and Types	
ht_impl_acc_config_t	ht_impl_gyr_config_t
ht_impl_mag_config_t	ht_impl_inertial_config_t
ht_impl_acc_t	ht_impl_gyr_t
ht_impl_mag_t	ht_impl_data_acc_t
ht_impl_data_gyr_t	ht_impl_data_mag_t
ht_impl_data_acc_gyr_t	ht_impl_data_acc_mag_t
ht_impl_data_gyr_mag_t	ht_impl_data_acc_gyr_mag_t
ht_impl_inertial_handle_t	
Dependency Functions	
ht_impl_inertial_handle_t -> inertial_data_cb()	
Functions	
ht_impl_init_inertial()	ht_impl_clear_inertial()
ht_impl_start_acc()	ht_impl_stop_acc()
ht_impl_start_gyr()	ht_impl_stop_gyr()
ht_impl_start_mag()	ht_impl_stop_mag()
ht_impl_config_acc()	ht_impl_config_gyr()
ht_impl_config_mag()	ht_impl_config_inertial()
ht_impl_state_get_acc()	ht_impl_state_get_gyr()
ht_impl_state_get_mag()	ht_impl_data_read_acc()
ht_impl_data_read_gyr()	ht_impl_data_read_mag()
ht_impl_data_pushing_and_indication_inertial_event()	ht_impl_data_indication_inertial_event()

8.2.2.1 Data Structures and Types

ht_impl_acc_config_t

Data Structure Fields	Type	Description
`enable`	`bool`	Enable/disable accelerometer sensor data flow: `HT_IMPL_DISABLE, HT_IMPL_ENABLE`

 © 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
mode	uint8_t	Accelerometer sensor operation mode: HT_IMPL_ACC_OPMODE_NORMAL, HT_IMPL_ACC_OPMODE_LOWPOWER_LP1, HT_IMPL_ACC_OPMODE_LOWPOWER_LP2, HT_IMPL_ACC_OPMODE_LOWPOWER_LP3, HT_IMPL_ACC_OPMODE_LOWPOWER_LP4, HT_IMPL_ACC_OPMODE_LOWPOWER_LP5, HT_IMPL_ACC_OPMODE_LOWPOWER_LP6
range	uint8_t	Accelerometer sensor data range: HT_IMPL_ACC_RANGE_2G, HT_IMPL_ACC_RANGE_4G, HT_IMPL_ACC_RANGE_8G, HT_IMPL_ACC_RANGE_16G
ind_rate	uint8_t	Accelerometer sensor indication data rate: HT_IMPL_ACC_RATE_INVALID, HT_IMPL_ACC_RATE_0_78HZ, HT_IMPL_ACC_RATE_1_56HZ, HT_IMPL_ACC_RATE_3_12HZ, HT_IMPL_ACC_RATE_6_25HZ, HT_IMPL_ACC_RATE_12_5HZ, HT_IMPL_ACC_RATE_25HZ, HT_IMPL_ACC_RATE_50HZ, HT_IMPL_ACC_RATE_100HZ
op_rate	uint8_t	Accelerometer sensor operation data rate: HT_IMPL_ACC_RATE_INVALID, HT_IMPL_ACC_RATE_0_78HZ, HT_IMPL_ACC_RATE_1_56HZ, HT_IMPL_ACC_RATE_3_12HZ, HT_IMPL_ACC_RATE_6_25HZ, HT_IMPL_ACC_RATE_12_5HZ, HT_IMPL_ACC_RATE_25HZ, HT_IMPL_ACC_RATE_50HZ, HT_IMPL_ACC_RATE_100HZ

ht_impl_gyr_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable gyroscope sensor data flow: HT_IMPL_DISABLE, HT_IMPL_ENABLE
mode	uint8_t	Gyroscope sensor operation mode: HT_IMPL_GYR_OPMODE_NORMAL, HT_IMPL_GYR_OPMODE_LOWPOWER_LP1, HT_IMPL_GYR_OPMODE_LOWPOWER_LP2, HT_IMPL_GYR_OPMODE_LOWPOWER_LP3, HT_IMPL_GYR_OPMODE_LOWPOWER_LP4, HT_IMPL_GYR_OPMODE_LOWPOWER_LP5, HT_IMPL_GYR_OPMODE_LOWPOWER_LP6
range	uint8_t	Gyroscope sensor data range: HT_IMPL_GYR_RANGE_2000_DEG_SEC, HT_IMPL_GYR_RANGE_1000_DEG_SEC, HT_IMPL_GYR_RANGE_500_DEG_SEC, HT_IMPL_GYR_RANGE_250_DEG_SEC, HT_IMPL_GYR_RANGE_125_DEG_SEC

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
ind_rate	uint8_t	Gyroscope sensor indication data rate: HT_IMPL_GYR_RATE_INVALID, HT_IMPL_GYR_RATE_0_78HZ, HT_IMPL_GYR_RATE_1_56HZ, HT_IMPL_GYR_RATE_3_12HZ, HT_IMPL_GYR_RATE_6_25HZ, HT_IMPL_GYR_RATE_12_5HZ, HT_IMPL_GYR_RATE_25HZ, HT_IMPL_GYR_RATE_50HZ, HT_IMPL_GYR_RATE_100HZ
op_rate	uint8_t	Gyroscope sensor operation data rate: HT_IMPL_GYR_RATE_INVALID, HT_IMPL_GYR_RATE_0_78HZ, HT_IMPL_GYR_RATE_1_56HZ, HT_IMPL_GYR_RATE_3_12HZ, HT_IMPL_GYR_RATE_6_25HZ, HT_IMPL_GYR_RATE_12_5HZ, HT_IMPL_GYR_RATE_25HZ, HT_IMPL_GYR_RATE_50HZ, HT_IMPL_GYR_RATE_100HZ

ht_impl_mag_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable magnetometer sensor data flow: HT_IMPL_DISABLE, HT_IMPL_ENABLE
mode	uint8_t	Magnetometer sensor operation mode: HT_IMPL_MAG_OPMODE_NORMAL, HT_IMPL_MAG_OPMODE_LOWPOWER_LP1, HT_IMPL_MAG_OPMODE_LOWPOWER_LP2, HT_IMPL_MAG_OPMODE_LOWPOWER_LP3, HT_IMPL_MAG_OPMODE_LOWPOWER_LP4, HT_IMPL_MAG_OPMODE_LOWPOWER_LP5, HT_IMPL_MAG_OPMODE_LOWPOWER_LP6
ind_rate	uint8_t	Magnetometer sensor indication data rate: HT_IMPL_MAG_RATE_INVALID, HT_IMPL_MAG_RATE_0_78HZ, HT_IMPL_MAG_RATE_1_56HZ, HT_IMPL_MAG_RATE_3_12HZ, HT_IMPL_MAG_RATE_6_25HZ, HT_IMPL_MAG_RATE_12_5HZ, HT_IMPL_MAG_RATE_25HZ, HT_IMPL_MAG_RATE_50HZ, HT_IMPL_MAG_RATE_100HZ
op_rate	uint8_t	Magnetometer sensor operation data rate: HT_IMPL_MAG_RATE_INVALID, HT_IMPL_MAG_RATE_0_78HZ, HT_IMPL_MAG_RATE_1_56HZ, HT_IMPL_MAG_RATE_3_12HZ, HT_IMPL_MAG_RATE_6_25HZ, HT_IMPL_MAG_RATE_12_5HZ, HT_IMPL_MAG_RATE_25HZ, HT_IMPL_MAG_RATE_50HZ, HT_IMPL_MAG_RATE_100HZ

© 2017 Dialog Semiconductor

ht_impl_inertial_config_t

Data Structure Fields	Type	Description
acc_config	ht_impl_acc_config_t	Accelerometer sensor data flow configuration structure.
gyr_config	ht_impl_gyr_config_t	Gyroscope sensor data flow configuration structure.
mag_config	ht_impl_mag_config_t	Magnetometer sensor data flow configuration structure.

ht_impl_acc_t

Type Definition Name	Description
ht_impl_acc_t	Configuration state structure for accelerometer sensor data flow: ht_impl_acc_config_t

ht_impl_gyr_t

Type Definition Name	Description
ht_impl_gyr_t	Configuration state structure for gyroscope sensor data flow: ht_impl_gyr_config_t

ht_impl_mag_t

Type Definition Name	Description
ht_impl_mag_t	Configuration state structure for magnetometer sensor data flow: ht_impl_mag_config_t

ht_impl_data_acc_t

Data Structure Fields	Type	Description
xyz.x	int16_t	Accelerometer x-axis sensor data value.
xyz.y	int16_t	Accelerometer y-axis sensor data value.
xyz.z	int16_t	Accelerometer z-axis sensor data value.
calibr_state. uncalibr_data_valid	uint8_t:1	Pre-calibrated data valid flag.
calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
calibr_state. cal_enabled	uint8_t:1	Calibration enabled flag.
calibr_state. cal_settled	uint8_t:1	Calibration settled flag.
calibr_state. reserved4	uint8_t:1	Reserved flag.
calibr_state. cal_mode	uint8_t:3	Calibration mode: SC_ACC_CAL_MODE_NONE, SC_ACC_CAL_MODE_STATIC

ht_impl_data_gyr_t

Data Structure Fields	Type	Description
xyz.x	int16_t	Gyroscope x-axis sensor data value.
xyz.y	int16_t	Gyroscope y-axis sensor data value.
xyz.z	int16_t	Gyroscope z-axis sensor data value.

© 2017 Dialog Semiconductor

ht_impl_data_mag_t

Data Structure Fields	Type	Description
xyz.x	int16_t	Magnetometer x-axis sensor data value.
xyz.y	int16_t	Magnetometer y-axis sensor data value.
xyz.z	int16_t	Magnetometer z-axis sensor data value.
calibr_state. uncalibr_data_valid	uint8_t:1	Pre-calibrated data valid flag.
calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
calibr_state. cal_enabled	uint8_t:1	Calibration enabled flag.
calibr_state. cal_settled	uint8_t:1	Calibration settled flag.
calibr_state. cal_converged	uint8_t:1	Calibration converged flag.
calibr_state. cal_mode	uint8_t:3	Calibration mode: SC_MAG_CAL_MODE_NONE. SC_MAG_CAL_MODE_STATIC, SC_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC, SC_MAG_CAL_MODE_CONT_AUTO_BASIC, SC_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART, SC_MAG_CAL_MODE_CONT_AUTO_SMART

ht_impl_data_acc_gyr_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HT_IMPL_VALID_ACC | HT_IMPL_VALID_GYR
acc_data	ht_impl_data_acc_t	Accelerometer xyz-axis sensor data.
gyr_data	ht_impl_data_gyr_t	Gyroscope xyz-axis sensor data.

ht_impl_data_acc_mag_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HT_IMPL_VALID_ACC | HT_IMPL_VALID_MAG
acc_data	ht_impl_data_acc_t	Accelerometer xyz-axis sensor data.
mag_data	ht_impl_data_mag_t	Magnetometer xyz-axis sensor data.

ht_impl_data_gyr_mag_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HT_IMPL_VALID_GYR | HT_IMPL_VALID_MAG
gyr_data	ht_impl_data_gyr_t	Gyroscope xyz-axis sensor data.
mag_data	ht_impl_data_mag_t	Magnetometer xyz-axis sensor data.

© 2017 Dialog Semiconductor

ht_impl_data_acc_gyr_mag_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HT_IMPL_VALID_ACC | HT_IMPL_VALID_GYR | HT_IMPL_VALID_MAG
acc_data	ht_impl_data_acc_t	Accelerometer xyz-axis sensor data.
gyr_data	ht_impl_data_gyr_t	Gyroscope xyz-axis sensor data.
mag_data	ht_impl_data_mag_t	Magnetometer xyz-axis sensor data.

ht_impl_inertial_handle_t

Data Structure Fields	Type	Description
inertial_sensor_types	uint8_t	Inertial sensor types referred by the data flow instance: HT_IMPL_VALID_ACC | HT_IMPL_VALID_GYR | HT_IMPL_VALID_MAG
least_ind_rate	uint8_t	Least rate required for the indication of multiple inertial sensor data. HT_IMPL_INERTIAL_IND_RATE_INVALID, HT_IMPL_INERTIAL_IND_RATE_0_78HZ, HT_IMPL_INERTIAL_IND_RATE_1_56HZ, HT_IMPL_INERTIAL_IND_RATE_3_12HZ, HT_IMPL_INERTIAL_IND_RATE_6_25HZ, HT_IMPL_INERTIAL_IND_RATE_12_5HZ, HT_IMPL_INERTIAL_IND_RATE_25HZ, HT_IMPL_INERTIAL_IND_RATE_50HZ, HT_IMPL_INERTIAL_IND_RATE_100HZ,
inertial_data_cb	ht_impl_data_rdy_cb_t	Inertial sensor data ready function pointer.
acc_config. config	ht_impl_acc_config_t	Accelerometer sensor data flow instance configuration.
acc_config. ind_samples_passed	uint8_t	Number of accelerometer sensor data samples passed.
acc_config. ind_samples_to_wait	uint8_t	Number of accelerometer sensor data samples to wait for, before indication.
gyr_config. config	ht_impl_gyr_config_t	Gyroscope sensor data flow instance configuration.
gyr_config. ind_samples_passed	uint8_t	Number of gyroscope sensor data samples passed.
gyr_config. ind_samples_to_wait	uint8_t	Number of gyroscope sensor data samples to wait for, before indication.
mag_config. config	ht_impl_mag_config_t	Magnetometer sensor data flow instance configuration.
mag_config. ind_samples_passed	uint8_t	Number of magnetometer sensor data samples passed.
mag_config. ind_samples_to_wait	uint8_t	Number of magnetometer sensor data samples to wait for, before indication.
next	ht_impl_inertial_handle_t	Next inertial sensor data flow instance handle in the list.

© 2017 Dialog Semiconductor

8.2.2.2 Dependency Functions

ht_impl_inertial_handle_t -> inertial_data_cb()

Function Name	`ht_impl_error_t (*inertial_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when inertial sensor data need to be sent over a defined data flow instance of type `ht_impl_inertial_handle_t`, based on the defined `inertial_sensor_types` field.
Parameters	`data` the multiple combined inertial sensor data samples, organized in frames/structures of the following types: `ht_impl_data_acc_t, ht_impl_data_gyr_t, ht_impl_data_mag_t, ht_impl_data_acc_gyr_t, ht_impl_data_acc_mag_t, ht_impl_data_gyr_mag_t, ht_impl_data_acc_gyr_mag_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.2.3 Functions

ht_impl_init_inertial()

Function Name	`ht_impl_error_t ht_impl_init_inertial(ht_impl_inertial_handle_t* handle, uint8_t inertial_sensor_types, ht_impl_data_rdy_cb_t inertial_data_cb)`
Function Description	Initializes inertial sensors data flow instance, by setting initial values to the internal variables and structures of the particular Health Toolbox implementation. Callback function may be set to the `ht_impl_inertial_handle_t` instance handle, so that data indications can be sent.
Parameters	`handle` the handle structure of the data flow instance `inertial_sensor_types` the inertial sensor types referred by the data flow instance `HT_IMPL_VALID_ACC | HT_IMPL_VALID_GYR | HT_IMPL_VALID_MAG` `inertial_data_cb` the callback function indicating inertial sensor data
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_clear_inertial()

Function Name	`ht_impl_error_t ht_impl_clear_inertial(ht_impl_inertial_handle_t* handle)`
Function Description	Clears inertial sensors data flow instance. The configuration of inertial sensors is determined by the rest of the data flow instances (if any), and it is updated accordingly.
Parameters	`handle` the handle structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_start_acc()

Function Name	`ht_impl_error_t ht_impl_start_acc(void)`
Function Description	Starts accelerometer sensor based on the configured data flow instances.
Parameters	None

© 2017 Dialog Semiconductor

Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_stop_acc()

Function Name	ht_impl_error_t ht_impl_stop_acc(void)
Function Description	Stops accelerometer sensor, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_start_gyr()

Function Name	ht_impl_error_t ht_impl_start_gyr(void)
Function Description	Starts gyroscope sensor based on the configured data flow instances.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_stop_gyr()

Function Name	ht_impl_error_t ht_impl_stop_gyr(void)
Function Description	Stops gyroscope sensor, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_start_mag()

Function Name	ht_impl_error_t ht_impl_start_mag(void)
Function Description	Starts magnetometer sensor based on the configured data flow instances.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_stop_mag()

Function Name	ht_impl_error_t ht_impl_stop_mag(void)
Function Description	Stops magnetometer sensor, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_config_acc()

© 2017 Dialog Semiconductor

Function Name	`ht_impl_error_t ht_impl_config_acc(ht_impl_inertial_handle_t* handle, ht_impl_acc_config_t* config, uint8_t least_ind_rate)`
Function Description	Configures accelerometer sensor data flow instance, i.e. enable/disable data indication, and set the required operation mode, the data range, the operation data rate and the indication data rate of accelerometer sensor for the particular instance. The configuration for the rest of the inertial sensors required by the particular instance remains the same. Furthermore, sets the least indication data rate required for the particular data flow instance, which in case it is slower than the configured inertial sensor indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `inertial_data_cb()` callback function.

Parameters	`handle` the handle structure of the data flow instance
	`config` the accelerometer sensor configuration structure of the data flow instance
	`least_ind_rate` least indication data rate of inertial sensors
	`HT_IMPL_INERTIAL_IND_RATE_INVALID,`
	`HT_IMPL_INERTIAL_IND_RATE_0_78HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_1_56HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_3_12HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_6_25HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_12_5HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_25HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_50HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_100HZ`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_gyr()

Function Name	`ht_impl_error_t ht_impl_config_gyr(ht_impl_inertial_handle_t* handle, ht_impl_gyr_config_t* config, uint8_t least_ind_rate)`
Function Description	Configures gyroscope sensor data flow instance, i.e. enable/disable data indication, and set the required operation mode, the data range, the operation data rate and the indication data rate of gyroscope sensor for the particular instance. The configuration for the rest of the inertial sensors required by the particular instance remains the same. Furthermore, sets the least indication data rate required for the particular data flow instance, which in case it is slower than the configured inertial sensor indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `inertial_data_cb()` callback function.

Parameters	`handle` the handle structure of the data flow instance
	`config` the gyroscope sensor configuration structure of the data flow instance
	`least_ind_rate` least indication data rate of inertial sensors
	`HT_IMPL_INERTIAL_IND_RATE_INVALID,`
	`HT_IMPL_INERTIAL_IND_RATE_0_78HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_1_56HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_3_12HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_6_25HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_12_5HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_25HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_50HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_100HZ`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error

© 2017 Dialog Semiconductor

Notes	

ht_impl_config_mag()

Function Name	`ht_impl_error_t ht_impl_config_mag(ht_impl_inertial_handle_t* handle, ht_impl_mag_config_t* config, uint8_t least_ind_rate)`
Function Description	Configures magneometer sensor data flow instance, i.e. enable/disable data indication, and set the required operation mode, the operation data rate and the indication data rate of magnetometer sensor for the particular instance. The configuration for the rest of the inertial sensors required by the particular instance remains the same. Furthermore, sets the least indication data rate required for the particular data flow instance, which in case it is slower than the configured inertial sensor indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `inertial_data_cb()` callback function.

Parameters	`handle` the handle structure of the data flow instance
	`config` the magnetometer sensor configuration structure of the data flow instance
	`least_ind_rate` least indication data rate of inertial sensors
	`HT_IMPL_INERTIAL_IND_RATE_INVALID,`
	`HT_IMPL_INERTIAL_IND_RATE_0_78HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_1_56HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_3_12HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_6_25HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_12_5HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_25HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_50HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_100HZ`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_inertial()

Function Name	`ht_impl_error_t ht_impl_config_inertial(ht_impl_inertial_handle_t* handle, ht_impl_inertial_config_t* config, uint8_t least_ind_rate)`
Function Description	Configures inertial sensors data flow instance and set the least indication data rate required, which in case it is slower than the configured inertial sensor indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `inertial_data_cb()` callback function.

Parameters	`handle` the handle structure of the data flow instance
	`config` the inertial sensors configuration structures of the data flow instance
	`least_ind_rate` least indication data rate of inertial sensors
	`HT_IMPL_INERTIAL_IND_RATE_INVALID,`
	`HT_IMPL_INERTIAL_IND_RATE_0_78HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_1_56HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_3_12HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_6_25HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_12_5HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_25HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_50HZ,`
	`HT_IMPL_INERTIAL_IND_RATE_100HZ`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

ht_impl_state_get_acc()

Function Name	`ht_impl_error_t ht_impl_state_get_acc(ht_impl_acc_t* state)`
Function Description	Gets configuration status of accelerometer sensor, i.e. the accelerometer sensor configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the accelerometer sensor data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_state_get_gyr()

Function Name	`ht_impl_error_t ht_impl_state_get_gyr(ht_impl_gyr_t* state)`
Function Description	Gets configuration status of gyroscope sensor, i.e. the gyroscope sensor configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the gyroscope sensor data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_state_get_mag()

Function Name	`ht_impl_error_t ht_impl_state_get_mag(ht_impl_mag_t* state)`
Function Description	Gets configuration status of magnetometer sensor, i.e. the magnetometer sensor configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the magnetometer sensor data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_acc()

Function Name	`ht_impl_error_t ht_impl_data_read_acc(ht_impl_inertial_handle_t* handle, ht_impl_data_acc_t* data)`
Function Description	Reads accelerometer sensor data. If accelerometer sensor is enabled for the particular data flow instance, then in case indication data rate is set, last read/indicated accelerometer sensor data value is returned (through `data` parameter), otherwise a force-read is performed and new accelerometer sensor data is acquired and returned (through `data` parameter). Accelerometer sensor data are converted to the appropriate range indicated by the particular data flow instance configuration.
Parameters	`handle` the handle structure of the data flow instance `data` the address to store accelerometer sensor data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_gyr()

Function Name	`ht_impl_error_t ht_impl_data_read_gyr(ht_impl_inertial_handle_t* handle, ht_impl_data_gyr_t* data)`

© 2017 Dialog Semiconductor

Function Description	Reads gyroscope sensor data. If gyroscope sensor is enabled for the particular data flow instance, then in case indication data rate is set, last read/indicated gyroscope sensor data value is returned (through `data` parameter), otherwise a force-read is performed and new gyroscope sensor data is acquired and returned (through `data` parameter). Gyroscope sensor data are converted to the appropriate range indicated by the particular data flow instance configuration.
Parameters	`handle` the handle structure of the data flow instance `data` the address to store gyroscope sensor data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_mag()

Function Name	`ht_impl_error_t ht_impl_data_read_mag(ht_impl_inertial_handle_t* handle, ht_impl_data_mag_t* data)`
Function Description	Reads magnetometer sensor data. If magnetometer sensor is enabled for the particular data flow instance, then in case indication data rate is set, last read/indicated magnetomater sensor data value is returned (through `data` parameter), otherwise a force-read is performed and new magnetometer sensor data is acquired and returned (through `data` parameter).
Parameters	`handle` the handle structure of the data flow instance `data` the address to store magnetometer sensor data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_pushing_and_indication_inertial_event()

Function Name	`ht_impl_error_t ht_impl_data_pushing_and_indication_inertial_event( HT_IMPL_INERTIAL_MULT_DATA_T* data, uint16_t size)`
Function Description	Pushes inertial sensor data and indicates (i.e. triggers a data-indication event) data over the initialized data flows, based on their configuration.
Parameters	`data` the inertial sensor data being pushed in a structure of type: `ht_impl_data_acc_t`, `ht_impl_data_gyr_t`, `ht_impl_data_mag_t`, `ht_impl_data_acc_gyr_t`, `ht_impl_data_acc_mag_t`, `ht_impl_data_gyr_mag_t`, `ht_impl_data_acc_gyr_mag_t` `size` the size of data (i.e. number of data samples) being pushed
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_indication_inertial_event()

Function Name	`ht_impl_error_t ht_impl_data_indication_inertial_event(void)`
Function Description	Triggers inertial sensor data indication event, flushing internal storage after processing and sending data indications over the initialized data flow instances, based on their configuration (i.e. indication data rates).
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.3 Environmental Sensors HT-Implementation – API Specification

 © 2017 Dialog Semiconductor

All functions and respective data structures/types being defined in the API are summarized below:

Table 20: Environmental Sensors HT-Implementation API

Data Structures and Types	
ht_impl_temp_config_t ht_impl_hum_config_t ht_impl_temp_t ht_impl_hum_t ht_impl_data_pres_t ht_impl_data_temp_pres_t ht_impl_data_pres_hum_t ht_impl_env_handle_t	ht_impl_pres_config_t ht_impl_env_config_t ht_impl_pres_t ht_impl_data_temp_t ht_impl_data_hum_t ht_impl_data_temp_hum_t ht_impl_data_temp_pres_hum_t
Dependency Functions	
ht_impl_env_handle_t -> env_data_cb()	
Functions	
ht_impl_init_env() ht_impl_start_temp() ht_impl_start_pres() ht_impl_start_hum() ht_impl_config_env_acquisition_event() ht_impl_config_pres() ht_impl_config_env() ht_impl_state_get_pres() ht_impl_data_read_temp() ht_impl_data_read_hum() ht_impl_data_pushing_env_event()	ht_impl_clear_env() ht_impl_stop_temp() ht_impl_stop_pres() ht_impl_stop_hum() ht_impl_config_temp() ht_impl_config_hum() ht_impl_state_get_temp() ht_impl_state_get_hum() ht_impl_data_read_pres() ht_impl_data_acquisition_env_event() ht_impl_data_indication_env_event()

8.2.3.1 Data Structures and Types

ht_impl_temp_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable temperature sensor data flow: HT_IMPL_DISABLE, HT_IMPL_ENABLE
mode	uint8_t	Temperature sensor operation mode: HT_IMPL_ENV_OPMODE_NORMAL, HT_IMPL_ENV_OPMODE_LOWPOWER
ind_rate	uint32_t	Temperature sensor indication data rate: HT_IMPL_ENV_RATE_0_125HZ, HT_IMPL_ENV_RATE_0_25HZ, HT_IMPL_ENV_RATE_0_5HZ, HT_IMPL_ENV_RATE_1HZ, HT_IMPL_ENV_RATE_2HZ, HT_IMPL_ENV_RATE_4HZ, HT_IMPL_ENV_RATE_8HZ, HT_IMPL_ENV_RATE_INVALID

© 2017 Dialog Semiconductor

ht_impl_pres_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable pressure sensor data flow: HT_IMPL_DISABLE, HT_IMPL_ENABLE
mode	uint8_t	Pressure sensor operation mode: HT_IMPL_ENV_OPMODE_NORMAL, HT_IMPL_ENV_OPMODE_LOWPOWER
ind_rate	uint32_t	Pressure sensor indication data rate: HT_IMPL_ENV_RATE_0_125HZ, HT_IMPL_ENV_RATE_0_25HZ, HT_IMPL_ENV_RATE_0_5HZ, HT_IMPL_ENV_RATE_1HZ, HT_IMPL_ENV_RATE_2HZ, HT_IMPL_ENV_RATE_4HZ, HT_IMPL_ENV_RATE_8HZ, HT_IMPL_ENV_RATE_INVALID

ht_impl_hum_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable humidity sensor data flow: HT_IMPL_DISABLE, HT_IMPL_ENABLE
mode	uint8_t	Humidity sensor operation mode: HT_IMPL_ENV_OPMODE_NORMAL, HT_IMPL_ENV_OPMODE_LOWPOWER
ind_rate	uint8_t	Humidity sensor indication data rate: HT_IMPL_ENV_RATE_0_125HZ, HT_IMPL_ENV_RATE_0_25HZ, HT_IMPL_ENV_RATE_0_5HZ, HT_IMPL_ENV_RATE_1HZ, HT_IMPL_ENV_RATE_2HZ, HT_IMPL_ENV_RATE_4HZ, HT_IMPL_ENV_RATE_8HZ, HT_IMPL_ENV_RATE_INVALID

ht_impl_env_config_t

Data Structure Fields	Type	Description
temp_config	ht_impl_temp_config_t	Temperature sensor data flow configuration structure.
pres_config	ht_impl_pres_config_t	Pressure sensor data flow configuration structure.
hum_config	ht_impl_hum_config_t	Humidity sensor data flow configuration structure.

ht_impl_temp_t

Type Definition Name	Description
ht_impl_temp_t	Configuration state structure for temperature sensor data flow: ht_impl_temp_config_t

© 2017 Dialog Semiconductor

ht_impl_pres_t

Type Definition Name	Description
ht_impl_pres_t	Configuration state structure for pressure sensor data flow: ht_impl_pres_config_t

ht_impl_hum_t

Type Definition Name	Description
ht_impl_hum_t	Configuration state structure for humidity sensor data flow: ht_impl_hum_config_t

ht_impl_data_temp_t

Data Structure Fields	Type	Description
temperature	int32_t	Temperature sensor data (in 0.01 Celsius degrees).

ht_impl_data_pres_t

Data Structure Fields	Type	Description
pressure	uint32_t	Pressure sensor data (in Pa).

ht_impl_data_hum_t

Data Structure Fields	Type	Description
humidity	uint32_t	Humidity sensor data (in %rH as unsigned 32bit integer in Q22.10 format(22 integer 10 fractional bits)).

ht_impl_data_temp_pres_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HT_IMPL_VALID_TEMP | HT_IMPL_VALID_PRES
temp_data	ht_impl_data_temp_t	Temperature sensor data.
pres_data	ht_impl_data_pres_t	Pressure sensor data.

ht_impl_data_temp_hum_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HT_IMPL_VALID_TEMP | HT_IMPL_VALID_HUM
temp_data	ht_impl_data_temp_t	Temperature sensor data.
hum_data	ht_impl_data_hum_t	Humidity sensor data.

© 2017 Dialog Semiconductor

ht_impl_data_pres_hum_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HT_IMPL_VALID_PRES | HT_IMPL_VALID_HUM
pres_data	ht_impl_data_pres_t	Pressure sensor data.
hum_data	ht_impl_data_hum_t	Humidity sensor data.

ht_impl_data_temp_pres_hum_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HT_IMPL_VALID_TEMP | HT_IMPL_VALID_PRES | HT_IMPL_VALID_HUM
temp_data	ht_impl_data_temp_t	Temperature sensor data.
pres_data	ht_impl_data_pres_t	Pressure sensor data.
hum_data	ht_impl_data_hum_t	Humidity sensor data.

ht_impl_env_handle_t

Data Structure Fields	Type	Description
env_sensor_types	uint8_t	Environmental sensor types referred by the data flow instance: HT_IMPL_VALID_TEMP | HT_IMPL_VALID_PRES | HT_IMPL_VALID_HUM
least_ind_rate	uint32_t	Least rate required for the indication of multiple environmental sensor data. HT_IMPL_ENV_RATE_0_125HZ, HT_IMPL_ENV_RATE_0_25HZ, HT_IMPL_ENV_RATE_0_5HZ, HT_IMPL_ENV_RATE_1HZ, HT_IMPL_ENV_RATE_2HZ, HT_IMPL_ENV_RATE_4HZ, HT_IMPL_ENV_RATE_8HZ, HT_IMPL_ENV_RATE_INVALID
env_data_cb	ht_impl_data_rdy_cb_t	Environmental sensor data ready function pointer.
temp_config. config	ht_impl_temp_config_t	Temperature sensor data flow instance configuration.
temp_config. ind_samples_passed	uint8_t	Number of temperature sensor data samples passed.
temp_config. ind_samples_to_wait	uint8_t	Number of temperature sensor data samples to wait for, before indication.
pres_config. config	ht_impl_pres_config_t	Pressure sensor data flow instance configuration.
pres_config. ind_samples_passed	uint8_t	Number of pressure sensor data samples passed.

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
pres_config. ind_samples_to_wait	uint8_t	Number of pressure sensor data samples to wait for, before indication.
hum_config. config	ht_impl_hum_config_t	Humidity sensor data flow instance configuration.
hum_config. ind_samples_passed	uint8_t	Number of humidity sensor data samples passed.
hum_config. ind_samples_to_wait	uint8_t	Number of humidity sensor data samples to wait for, before indication.
next	ht_impl_env_handle_t	Next environmental sensor data flow instance handle in the list.

© 2017 Dialog Semiconductor

8.2.3.2 Dependency Functions

ht_impl_env_handle_t -> env_data_cb()

Function Name	`ht_impl_error_t (*env_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when environmental sensor data need to be sent over a defined data flow instance of type `ht_impl_env_handle_t`, based on the defined `env_sensor_types` field.
Parameters	`data` the multiple combined environmental sensor data samples, organized in frames/structures of the following types: `ht_impl_data_temp_t, ht_impl_data_pres_t, ht_impl_data_hum_t, ht_impl_data_temp_pres_t, ht_impl_data_temp_hum_t, ht_impl_data_pres_hum_t, ht_impl_data_temp_pres_hum_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.3.3 Functions

ht_impl_init_env()

Function Name	`ht_impl_error_t ht_impl_init_env(ht_impl_env_handle_t* handle, uint8_t env_sensor_types, ht_impl_data_rdy_cb_t env_data_cb)`
Function Description	Initializes environmental sensors data flow instance, by setting initial values to the internal variables and structures of the particular Health Toolbox implementation. Callback function may be set to the `ht_impl_env_handle_t` instance handle, so that data indications can be sent.
Parameters	`handle` the handle structure of the data flow instance `env_sensor_types` the environmental sensor types referred by the data flow instance `HT_IMPL_VALID_TEMP | HT_IMPL_VALID_PRES | HT_IMPL_VALID_HUM` `env_data_cb` the callback function indicating environmental sensor data
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_clear_env()

Function Name	`ht_impl_error_t ht_impl_clear_env(ht_impl_env_handle_t* handle)`
Function Description	Clears environmental sensors data flow instance. The configuration of environmental sensors is determined by the rest of the data flow instances (if any), and it is updated accordingly.
Parameters	`handle` the handle structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_start_temp()

Function Name	`ht_impl_error_t ht_impl_start_temp(void)`
Function Description	Starts temperature sensor based on the configured data flow instances.
Parameters	None

 © 2017 Dialog Semiconductor

Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_stop_temp()

Function Name	ht_impl_error_t ht_impl_stop_temp(void)
Function Description	Stops temperature sensor, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_start_pres()

Function Name	ht_impl_error_t ht_impl_start_pres(void)
Function Description	Starts pressure sensor based on the configured data flow instances.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_stop_pres()

Function Name	ht_impl_error_t ht_impl_stop_pres(void)
Function Description	Stops pressure sensor, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_start_hum()

Function Name	ht_impl_error_t ht_impl_start_hum(void)
Function Description	Starts humidity sensor based on the configured data flow instances.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_stop_hum()

Function Name	ht_impl_error_t ht_impl_stop_hum(void)
Function Description	Stops humidity sensor, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_config_env_acquisition_event()

© 2017 Dialog Semiconductor

Function Name	`ht_impl_error_t ht_impl_config_env_acquisition_event(bool enable, uint32_t rate)`
Function Description	Configures environmental sensor data (external) acquisition event source. If external source is disabled then an internal source is used. Acquisition event rate defines the base rate based on which the data indication rates of the initialized data flow instances are determined. External acquisition event source can be set only when all data flow instances are disabled.

Parameters	`enable` enable/disable (external) acquisition event source
	`HT_IMPL_DISABLE, HT_IMPL_ENABLE`
	`rate` the base rate (in msec) of the acquisition event
	`HT_IMPL_ENV_RATE_0_125HZ,`
	`HT_IMPL_ENV_RATE_0_25HZ,`
	`HT_IMPL_ENV_RATE_0_5HZ,`
	`HT_IMPL_ENV_RATE_1HZ,`
	`HT_IMPL_ENV_RATE_2HZ,`
	`HT_IMPL_ENV_RATE_4HZ,`
	`HT_IMPL_ENV_RATE_8HZ`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_temp()

Function Name	`ht_impl_error_t ht_impl_config_temp(ht_impl_env_handle_t* handle, ht_impl_temp_config_t* config, uint32_t least_ind_rate)`
Function Description	Configures temperature sensor data flow instance, i.e. enable/disable data indication, and set the required operation mode and the indication data rate of temperature sensor for the particular instance. The configuration for the rest of the environmental sensors required by the particular instance remains the same. Furthermore, sets the least indication data rate required for the particular data flow instance, which in case it is slower than the configured environmental sensor indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `env_data_cb()` callback function.

Parameters	`handle` the handle structure of the data flow instance
	`config` the temperature sensor configuration structure of the data flow instance
	`least_ind_rate` least indication data rate of environmental sensors
	`HT_IMPL_ENV_RATE_0_125HZ,`
	`HT_IMPL_ENV_RATE_0_25HZ,`
	`HT_IMPL_ENV_RATE_0_5HZ,`
	`HT_IMPL_ENV_RATE_1HZ,`
	`HT_IMPL_ENV_RATE_2HZ,`
	`HT_IMPL_ENV_RATE_4HZ,`
	`HT_IMPL_ENV_RATE_8HZ,`
	`HT_IMPL_ENV_RATE_INVALID`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_pres()

Function Name	`ht_impl_error_t ht_impl_config_pres(ht_impl_env_handle_t* handle, ht_impl_pres_config_t* config, uint32_t least_ind_rate)`

© 2017 Dialog Semiconductor

Function Description	Configures pressure sensor data flow instance, i.e. enable/disable data indication, and set the required operation mode and the indication data rate of pressure sensor for the particular instance. The configuration for the rest of the environmental sensors required by the particular instance remains the same. Furthermore, sets the least indication data rate required for the particular data flow instance, which in case it is slower than the configured environmental sensor indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `env_data_cb()` callback function.

Parameters	`handle` the handle structure of the data flow instance
	`config` the pressure sensor configuration structure of the data flow instance
	`least_ind_rate` least indication data rate of environmental sensors
	`HT_IMPL_ENV_RATE_0_125HZ,`
	`HT_IMPL_ENV_RATE_0_25HZ,`
	`HT_IMPL_ENV_RATE_0_5HZ,`
	`HT_IMPL_ENV_RATE_1HZ,`
	`HT_IMPL_ENV_RATE_2HZ,`
	`HT_IMPL_ENV_RATE_4HZ,`
	`HT_IMPL_ENV_RATE_8HZ,`
	`HT_IMPL_ENV_RATE_INVALID`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_hum()

Function Name	`ht_impl_error_t ht_impl_config_hum(ht_impl_env_handle_t* handle, ht_impl_hum_config_t* config, uint32_t least_ind_rate)`
Function Description	Configures humidity sensor data flow instance, i.e. enable/disable data indication, and set the required operation mode and the indication data rate of humidity sensor for the particular instance. The configuration for the rest of the environmental sensors required by the particular instance remains the same. Furthermore, sets the least indication data rate required for the particular data flow instance, which in case it is slower than the configured environmental sensor indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `env_data_cb()` callback function.

Parameters	`handle` the handle structure of the data flow instance
	`config` the humidity sensor configuration structure of the data flow instance
	`least_ind_rate` least indication data rate of environmental sensors
	`HT_IMPL_ENV_RATE_0_125HZ,`
	`HT_IMPL_ENV_RATE_0_25HZ,`
	`HT_IMPL_ENV_RATE_0_5HZ,`
	`HT_IMPL_ENV_RATE_1HZ,`
	`HT_IMPL_ENV_RATE_2HZ,`
	`HT_IMPL_ENV_RATE_4HZ,`
	`HT_IMPL_ENV_RATE_8HZ,`
	`HT_IMPL_ENV_RATE_INVALID`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_env()

Function Name	`ht_impl_error_t ht_impl_config_env(ht_impl_env_handle_t* handle, ht_impl_env_config_t* config, uint32_t least_ind_rate)`

© 2017 Dialog Semiconductor

Function Description	Configures environmental sensors data flow instance, i.e. enable/disable data indication, and set the required operation mode and the indication data rate of environmental sensors for the particular instance. Furthermore, sets the least indication data rate required for the particular data flow instance, which in case it is slower than the configured environmental sensor indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `env_data_cb()` callback function..

Parameters	`handle` the handle structure of the data flow instance
	`config` the environmental sensors configuration structures of the data flow instance
	`least_ind_rate` least indication data rate of environmental sensors
	`HT_IMPL_ENV_RATE_0_125HZ,`
	`HT_IMPL_ENV_RATE_0_25HZ,`
	`HT_IMPL_ENV_RATE_0_5HZ,`
	`HT_IMPL_ENV_RATE_1HZ,`
	`HT_IMPL_ENV_RATE_2HZ,`
	`HT_IMPL_ENV_RATE_4HZ,`
	`HT_IMPL_ENV_RATE_8HZ,`
	`HT_IMPL_ENV_RATE_INVALID`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_state_get_temp()

Function Name	`ht_impl_error_t ht_impl_state_get_temp(ht_impl_temp_t* state)`
Function Description	Gets configuration status of temperature sensor, i.e. the temperature sensor configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the temperature sensor data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_state_get_pres()

Function Name	`ht_impl_error_t ht_impl_state_get_pres(ht_impl_pres_t* state)`
Function Description	Gets configuration status of pressure sensor, i.e. the pressure sensor configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the pressure sensor data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_state_get_hum()

Function Name	`ht_impl_error_t ht_impl_state_get_hum(ht_impl_hum_t* state)`
Function Description	Gets configuration status of humidity sensor, i.e. the humidity sensor configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the humidity sensor data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

ht_impl_data_read_temp()

Function Name	`ht_impl_error_t ht_impl_data_read_temp(ht_impl_env_handle_t* handle, ht_impl_data_temp_t* data)`
Function Description	Reads temperature sensor data. If temperature sensor is enabled for the particular data flow instance, then in case indication data rate is set, last read/indicated temperature sensor data value is returned (through `data` parameter), otherwise a force-read is performed and new temperature sensor data is acquired and returned via `env_data_cb()`.
Parameters	`handle` the handle structure of the data flow instance `data` the address to store temperature sensor data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_pres()

Function Name	`ht_impl_error_t ht_impl_data_read_pres(ht_impl_env_handle_t* handle, ht_impl_data_pres_t* data)`
Function Description	Reads pressure sensor data. If pressure sensor is enabled for the particular data flow instance, then in case indication data rate is set, last read/indicated pressure sensor data value is returned (through `data` parameter), otherwise a force-read is performed and new pressure sensor data is acquired and returned via `env_data_cb()`.
Parameters	`handle` the handle structure of the data flow instance `data` the address to store pressure sensor data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_hum()

Function Name	`ht_impl_error_t ht_impl_data_read_hum(ht_impl_env_handle_t* handle, ht_impl_data_hum_t* data)`
Function Description	Reads humidity sensor data. If humidity sensor is enabled for the particular data flow instance, then in case indication data rate is set, last read/indicated humidity sensor data value is returned (through `data` parameter), otherwise a force-read is performed and new humidity sensor data is acquired and returned via `env_data_cb()`.
Parameters	`handle` the handle structure of the data flow instance `data` the address to store humidity sensor data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_acquisition_env_event()

Function Name	`ht_impl_error_t ht_impl_data_acquisition_env_event(void)`
Function Description	Triggers environmental sensor data (external) acquisition event. Based on the configured indication rate of environmental sensors, the data of the latter are (force-) read and stored to the internal storage (i.e. FIFO). If the internal storage (i.e. FIFO) is full or reaches the (watermark) level indicated by the data-flow-instance-wide configured least indication rate, then data indications are sent over the initialized data flow instances, based on their configuration (i.e. indication data rates), and the data storage is flushed.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error

© 2017 Dialog Semiconductor

Notes	

ht_impl_data_pushing_env_event()

Function Name	`ht_impl_error_t ht_impl_data_pushing_env_event(` `HT_IMPL_ENV_MULT_DATA_T* data, uint16_t size)`
Function Description	Pushes environmental sensor data to Health Toolbox implementation of environmental sensor data flows. If the internal storage (i.e. FIFO) is full or reaches the (watermark) level indicated by the data-flow-instance-wide configured least indication rate, then data indications are sent over the initialized data flow instances, based on their configuration (i.e. indication data rates), and the data storage is flushed.
Parameters	`data` the environmental sensor data being pushed in a structure of type: `ht_impl_data_temp_t, ht_impl_data_pres_t, ht_impl_data_hum_t, ht_impl_data_temp_pres_t, ht_impl_data_temp_hum_t, ht_impl_data_pres_hum_t, ht_impl_data_temp_pres_hum_t` `size` the size of data (i.e. number of data samples) being pushed
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_indication_env_event()

Function Name	`ht_impl_error_t ht_impl_data_indication_env_event(void)`
Function Description	Triggers environmental sensor data indication event, flushing internal storage after processing and sending data indications over the initialized data flow instances, based on their configuration (i.e. indication data rates).
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.4 Health Care Sensors HT-Implementation – API Specification

All functions and respective data structures/types being defined in the API are summarized below:

Table 21: Health Care Sensors HT-Implementation API

Data Structures and Types	
ht_impl_hc_opt_config_t ht_impl_hc_config_t ht_impl_hc_acc_t ht_impl_data_hc_acc_t ht_impl_hc_handle_t	ht_impl_hc_acc_config_t ht_impl_hc_opt_t ht_impl_data_hc_opt_t ht_impl_data_hc_opt_acc_t
Dependency Functions	
ht_impl_hc_handle_t -> hc_data_cb()	

© 2017 Dialog Semiconductor

Functions	
ht_impl_init_hc()	ht_impl_clear_hc()
ht_impl_start_hc_opt()	ht_impl_stop_hc_opt()
ht_impl_start_hc_acc()	ht_impl_stop_hc_acc()
ht_impl_config_hc_acquisition_event()	ht_impl_config_hc_opt()
ht_impl_config_hc_acc()	ht_impl_config_hc()
ht_impl_state_get_hc_opt()	ht_impl_state_get_hc_acc()
ht_impl_data_read_hc_opt()	ht_impl_data_read_hc_acc()
ht_impl_data_acquisition_hc_event()	ht_impl_data_pushing_hc_event()
ht_impl_data_indication_hc_event()	

8.2.4.1 Data Structures and Types

ht_impl_hc_opt_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable health care optical sensor data flow: HT_IMPL_DISABLE, HT_IMPL_ENABLE
mode	uint8_t	Health care optical sensor operation mode: HT_IMPL_HC_OPT_OPMODE_NORMAL, HT_IMPL_HC_OPT_OPMODE_LOWPOWER
ind_rate	uint8_t	Health care optical sensor indication data rate: HT_IMPL_HC_OPT_RATE_INVALID, HT_IMPL_HC_OPT_RATE_0_78HZ, HT_IMPL_HC_OPT_RATE_1_56HZ, HT_IMPL_HC_OPT_RATE_3_12HZ, HT_IMPL_HC_OPT_RATE_6_25HZ, HT_IMPL_HC_OPT_RATE_12_5HZ, HT_IMPL_HC_OPT_RATE_25HZ, HT_IMPL_HC_OPT_RATE_50HZ, HT_IMPL_HC_OPT_RATE_100HZ

ht_impl_hc_acc_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable health care accelerometer sensor data flow: HT_IMPL_DISABLE, HT_IMPL_ENABLE
mode	uint8_t	Health care accelerometer sensor operation mode: HT_IMPL_HC_ACC_OPMODE_NORMAL, HT_IMPL_HC_ACC_OPMODE_LOWPOWER_LP1, HT_IMPL_HC_ACC_OPMODE_LOWPOWER_LP2, HT_IMPL_HC_ACC_OPMODE_LOWPOWER_LP3, HT_IMPL_HC_ACC_OPMODE_LOWPOWER_LP4, HT_IMPL_HC_ACC_OPMODE_LOWPOWER_LP5, HT_IMPL_HC_ACC_OPMODE_LOWPOWER_LP6
range	uint8_t	Health care accelerometer sensor data range: HT_IMPL_HC_ACC_RANGE_2G, HT_IMPL_HC_ACC_RANGE_4G, HT_IMPL_HC_ACC_RANGE_8G, HT_IMPL_HC_ACC_RANGE_16G

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
ind_rate	uint8_t	Health care accelerometer sensor indication data rate: HT_IMPL_HC_ACC_RATE_INVALID, HT_IMPL_HC_ACC_RATE_0_78HZ, HT_IMPL_HC_ACC_RATE_1_56HZ, HT_IMPL_HC_ACC_RATE_3_12HZ, HT_IMPL_HC_ACC_RATE_6_25HZ, HT_IMPL_HC_ACC_RATE_12_5HZ, HT_IMPL_HC_ACC_RATE_25HZ, HT_IMPL_HC_ACC_RATE_50HZ, HT_IMPL_HC_ACC_RATE_100HZ
op_rate	uint8_t	Health care accelerometer sensor operation data rate: HT_IMPL_HC_ACC_RATE_INVALID, HT_IMPL_HC_ACC_RATE_0_78HZ, HT_IMPL_HC_ACC_RATE_1_56HZ, HT_IMPL_HC_ACC_RATE_3_12HZ, HT_IMPL_HC_ACC_RATE_6_25HZ, HT_IMPL_HC_ACC_RATE_12_5HZ, HT_IMPL_HC_ACC_RATE_25HZ, HT_IMPL_HC_ACC_RATE_50HZ, HT_IMPL_HC_ACC_RATE_100HZ

ht_impl_hc_config_t

Data Structure Fields	Type	Description
hc_opt_config	ht_impl_hc_opt_config_t	Health care optical sensor data flow configuration structure.
hc_acc_config	ht_impl_hc_acc_config_t	Health care accelerometer sensor data flow configuration structure.

ht_impl_hc_opt_t

Type Definition Name	Description
ht_impl_hc_opt_t	Configuration state structure for health care optical sensor data flow: ht_impl_hc_opt_config_t

ht_impl_hc_acc_t

Type Definition Name	Description
ht_impl_hc_acc_t	Configuration state structure for health care accelerometer sensor data flow: ht_impl_hc_acc_config_t

ht_impl_data_hc_opt_t

Data Structure Fields	Type	Description
green	uint16_t	Health care optical sensor Green LED light intensity data.
ir	uint16_t	Health care optical sensor IR LED light intensity data.

© 2017 Dialog Semiconductor

ht_impl_data_hc_acc_t

Data Structure Fields	Type	Description
xyz.x	int16_t	Health care accelerometer x-axis sensor data value.
xyz.y	int16_t	Health care accelerometer y-axis sensor data value.
xyz.z	int16_t	Health care accelerometer z-axis sensor data value.
calibr_state. uncalibr_data_valid	uint8_t:1	Pre-calibrated data valid flag.
calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
calibr_state. cal_enabled	uint8_t:1	Calibration enabled flag.
calibr_state. cal_settled	uint8_t:1	Calibration settled flag.
calibr_state. reserved4	uint8_t:1	Reserved flag.
calibr_state. cal_mode	uint8_t:3	Calibration mode: SC_ACC_CAL_MODE_NONE, SC_ACC_CAL_MODE_STATIC

ht_impl_data_hc_opt_acc_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HT_IMPL_VALID_ACC | HT_IMPL_VALID_OPT
hc_opt_data	ht_impl_data_hc_opt_t	Health care optical sensor data.
hc_acc_data	ht_impl_data_hc_acc_t	Health care accelerometer sensor data.

ht_impl_hc_handle_t

Data Structure Fields	Type	Description
hc_sensor_types	uint8_t	Health care sensor types referred by the data flow instance: HT_IMPL_VALID_ACC | HT_IMPL_VALID_OPT
least_ind_rate	uint8_t	Least rate required for the indication of multiple health care sensor data. HT_IMPL_HC_IND_RATE_INVALID, HT_IMPL_HC_IND_RATE_0_78HZ, HT_IMPL_HC_IND_RATE_1_56HZ, HT_IMPL_HC_IND_RATE_3_12HZ, HT_IMPL_HC_IND_RATE_6_25HZ, HT_IMPL_HC_IND_RATE_12_5HZ, HT_IMPL_HC_IND_RATE_25HZ, HT_IMPL_HC_IND_RATE_50HZ, HT_IMPL_HC_IND_RATE_100HZ
hc_data_cb	ht_impl_data_rdy_cb_t	Health care sensor data ready function pointer.
hc_opt_config. config	ht_impl_hc_opt_config_t	Health care optical sensor data flow instance configuration.
hc_opt_config. ind_samples_passed	uint8_t	Number of health care optical sensor data samples passed.

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
hc_opt_config. ind_samples_to_wait	uint8_t	Number of health care optical sensor data samples to wait for, before indication.
hc_acc_config. config	ht_impl_hc_acc_config_t	Health care accelerometer sensor data flow instance configuration.
hc_acc_config. ind_samples_passed	uint8_t	Number of health care accelerometer sensor data samples passed.
hc_acc_config. ind_samples_to_wait	uint8_t	Number of health care accelerometer sensor data samples to wait for, before indication.
next	ht_impl_hc_handle_t	Next health care sensor data flow instance handle in the list.

© 2017 Dialog Semiconductor

8.2.4.2 Dependency Functions

ht_impl_hc_handle_t -> hc_data_cb()

Function Name	`ht_impl_error_t (*hc_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when health care sensor data need to be sent over a defined data flow instance of type `ht_impl_hc_handle_t`, based on the defined `hc_sensor_types` field.
Parameters	`data` the multiple combined health care sensor data samples, organized in frames/structures of the following types: `ht_impl_data_hc_opt_t, ht_impl_data_hc_acc_t, ht_impl_data_hc_opt_acc_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.4.3 Functions

ht_impl_init_hc()

Function Name	`ht_impl_error_t ht_impl_init_hc(ht_impl_hc_handle_t* handle, uint8_t hc_sensor_types, ht_impl_data_rdy_cb_t hc_data_cb)`
Function Description	Initializes health care sensors data flow instance, by setting initial values to the internal variables and structures of the particular Health Toolbox implementation. Callback function may be set to the `ht_impl_hc_handle_t` instance handle, so that data indications can be sent.
Parameters	`handle` the handle structure of the data flow instance `hc_sensor_types` the health care sensor types referred by the data flow instance `HT_IMPL_VALID_ACC | HT_IMPL_VALID_OPT` `hc_data_cb` the callback function indicating health care sensor data
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_clear_hc()

Function Name	`ht_impl_error_t ht_impl_clear_hc(ht_impl_hc_handle_t* handle)`
Function Description	Clears health care sensors data flow instance. The configuration of health care sensors is determined by the rest of the data flow instances (if any), and it is updated accordingly.
Parameters	`handle` the handle structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_start_hc_opt()

Function Name	`ht_impl_error_t ht_impl_start_hc_opt(void)`
Function Description	Starts health care optical sensor based on the configured data flow instances.
Parameters	None

© 2017 Dialog Semiconductor

Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_stop_hc_opt()

Function Name	ht_impl_error_t ht_impl_stop_hc_opt(void)
Function Description	Stops health care optical sensor, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_start_hc_acc()

Function Name	ht_impl_error_t ht_impl_start_hc_acc(void)
Function Description	Starts health care accelerometer sensor based on the configured data flow instances.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_stop_hc_acc()

Function Name	ht_impl_error_t ht_impl_stop_hc_acc(void)
Function Description	Stops health care accelerometer sensor, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_config_hc_acquisition_event()

Function Name	ht_impl_error_t ht_impl_config_hc_acquisition_event(bool enable, uint8_t rate)
Function Description	Configures health care sensor data (external) acquisition event source. If external source is disabled then an internal source is used. Acquisition event rate defines the base rate based on which the data indication rates of the initialized data flow instances are determined. External acquisition event source can be set only when all data flow instances are disabled.
Parameters	enable — enable/disable (external) acquisition event source HT_IMPL_DISABLE, HT_IMPL_ENABLE rate — the base rate of the acquisition event HT_IMPL_HC_IND_RATE_INVALID, HT_IMPL_HC_IND_RATE_0_78HZ, HT_IMPL_HC_IND_RATE_1_56HZ, HT_IMPL_HC_IND_RATE_3_12HZ, HT_IMPL_HC_IND_RATE_6_25HZ, HT_IMPL_HC_IND_RATE_12_5HZ, HT_IMPL_HC_IND_RATE_25HZ, HT_IMPL_HC_IND_RATE_50HZ, HT_IMPL_HC_IND_RATE_100HZ
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error

© 2017 Dialog Semiconductor

Notes	

ht_impl_config_hc_opt()

Function Name	`ht_impl_error_t ht_impl_config_hc_opt(ht_impl_hc_handle_t* handle, ht_impl_hc_opt_config_t* config, uint8_t least_ind_rate)`
Function Description	Configures health care optical sensor data flow instance, i.e. enable/disable data indication, and set the required operation mode and the indication data rate of optical sensor for the particular instance. The configuration for the health care accelerometer sensor required by the particular instance remains the same. Furthermore, sets the least indication data rate required for the particular data flow instance, which in case it is slower than the configured health care sensor indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `hc_data_cb()` callback function.

Parameters	`handle` the handle structure of the data flow instance
	`config` the health care optical sensor configuration structure of the data flow instance
	`least_ind_rate` least indication data rate of health care sensors
	`HT_IMPL_HC_IND_RATE_INVALID,`
	`HT_IMPL_HC_IND_RATE_0_78HZ,`
	`HT_IMPL_HC_IND_RATE_1_56HZ,`
	`HT_IMPL_HC_IND_RATE_3_12HZ,`
	`HT_IMPL_HC_IND_RATE_6_25HZ,`
	`HT_IMPL_HC_IND_RATE_12_5HZ,`
	`HT_IMPL_HC_IND_RATE_25HZ,`
	`HT_IMPL_HC_IND_RATE_50HZ,`
	`HT_IMPL_HC_IND_RATE_100HZ`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_hc_acc()

Function Name	`ht_impl_error_t ht_impl_config_hc_acc(ht_impl_hc_handle_t* handle, ht_impl_hc_acc_config_t* config, uint8_t least_ind_rate)`
Function Description	Configures health care accelerometer sensor data flow instance, i.e. enable/disable data indication, and set the required operation mode, the data range, the operation data rate and the indication data rate of accelerometer sensor for the particular instance. The configuration for the health care optical sensor required by the particular instance remains the same. Furthermore, sets the least indication data rate required for the particular data flow instance, which in case it is slower than the configured health care sensor indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `hc_data_cb()` callback function.

© 2017 Dialog Semiconductor

Parameters	`handle` the handle structure of the data flow instance
	`config` the health care accelerometer sensor configuration structure of the data flow instance
	`least_ind_rate` least indication data rate of health care sensors
	`HT_IMPL_HC_IND_RATE_INVALID,`
	`HT_IMPL_HC_IND_RATE_0_78HZ,`
	`HT_IMPL_HC_IND_RATE_1_56HZ,`
	`HT_IMPL_HC_IND_RATE_3_12HZ,`
	`HT_IMPL_HC_IND_RATE_6_25HZ,`
	`HT_IMPL_HC_IND_RATE_12_5HZ,`
	`HT_IMPL_HC_IND_RATE_25HZ,`
	`HT_IMPL_HC_IND_RATE_50HZ,`
	`HT_IMPL_HC_IND_RATE_100HZ`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_hc()

Function Name	`ht_impl_error_t ht_impl_config_hc(ht_impl_hc_handle_t* handle, ht_impl_hc_config_t* config, uint8_t least_ind_rate)`
Function Description	Configures health care sensors data flow instance and set the least indication data rate required, which in case it is slower than the configured health care sensor indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `hc_data_cb()` callback function.

Parameters	`handle` the handle structure of the data flow instance
	`config` the health care sensors configuration structures of the data flow instance
	`least_ind_rate` least indication data rate of health care sensors
	`HT_IMPL_HC_IND_RATE_INVALID,`
	`HT_IMPL_HC_IND_RATE_0_78HZ,`
	`HT_IMPL_HC_IND_RATE_1_56HZ,`
	`HT_IMPL_HC_IND_RATE_3_12HZ,`
	`HT_IMPL_HC_IND_RATE_6_25HZ,`
	`HT_IMPL_HC_IND_RATE_12_5HZ,`
	`HT_IMPL_HC_IND_RATE_25HZ,`
	`HT_IMPL_HC_IND_RATE_50HZ,`
	`HT_IMPL_HC_IND_RATE_100HZ`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_state_get_hc_opt()

Function Name	`ht_impl_error_t ht_impl_state_get_hc_opt(ht_impl_hc_opt_t* state)`
Function Description	Gets configuration status of health care optical sensor, i.e. the health care optical sensor configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the health care optical sensor data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_state_get_hc_acc()

Function Name	`ht_impl_error_t ht_impl_state_get_hc_acc(ht_impl_hc_acc_t* state)`

© 2017 Dialog Semiconductor

Function Description	Gets configuration status of health care accelerometer sensor, i.e. the health care accelerometer sensor configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the health care accelerometer sensor data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_hc_opt()

Function Name	`ht_impl_error_t ht_impl_data_read_hc_opt(ht_impl_hc_handle_t* handle, ht_impl_data_hc_opt_t* data)`
Function Description	Reads health care optical sensor data. If optical sensor is enabled for the particular data flow instance, then in case indication data rate is set, last read/indicated optical sensor data value is returned (through `data` parameter), otherwise a force-read is performed and new optical sensor data is acquired and returned (through `data` parameter).
Parameters	`handle` the handle structure of the data flow instance `data` the address to store health care optical sensor data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_hc_acc()

Function Name	`ht_impl_error_t ht_impl_data_read_hc_acc(ht_impl_hc_handle_t* handle, ht_impl_data_hc_acc_t* data)`
Function Description	Reads health care accelerometer sensor data. If accelerometer sensor is enabled for the particular data flow instance, then in case indication data rate is set, last read/indicated accelerometer sensor data value is returned (through `data` parameter), otherwise a force-read is performed and new accelerometer sensor data is acquired and returned (through `data` parameter).
Parameters	`handle` the handle structure of the data flow instance `data` the address to store health care accelerometer sensor data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_acquisition_hc_event()

Function Name	`ht_impl_error_t ht_impl_data_acquisition_hc_event(void)`
Function Description	Triggers health care sensor data (external) acquisition event. Based on the configured indication rate of health care sensors, the data of the latter are (force-)read and stored to the internal storage (i.e. FIFO). If the internal storage (i.e. FIFO) is full or reaches the (watermark) level indicated by the data-flow-instance-wide configured least indication rate, then data indications are sent over the initialized data flow instances, based on their configuration (i.e. indication data rates), and the data storage is flushed.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_pushing_hc_event()

© 2017 Dialog Semiconductor

Function Name	`ht_impl_error_t ht_impl_data_pushing_hc_event( HT_IMPL_HC_MULT_DATA_T* data, uint16_t size)`
Function Description	Pushes health care sensor data to Health Toolbox implementation of health care sensor data flows. If the internal storage (i.e. FIFO) is full or reaches the (watermark) level indicated by the data-flow-instance-wide configured least indication rate, then data indications are sent over the initialized data flow instances, based on their configuration (i.e. indication data rates), and the data storage is flushed.
Parameters	`data` the health care sensor data being pushed in a structure of type: `ht_impl_data_hc_opt_t, ht_impl_data_hc_acc_t, ht_impl_data_hc_opt_acc_t` `size` the size of data (i.e. number of data samples) being pushed
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_indication_hc_event()

Function Name	`ht_impl_error_t ht_impl_data_indication_hc_event(void)`
Function Description	Triggers health care sensor data indication event, flushing internal storage after processing and sending data indications over the initialized data flow instances, based on their configuration (i.e. indication data rates).
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.5 Sensor Fusion Service HT-Implementation – API Specification

All functions and respective data structures/types being defined in the API are summarized below:

Table 22: Sensor Fusion Service HT-Implementation API

Data Structures and Types	
ht_impl_sf_config_t hc_svcs_sf_attrs_t ht_impl_sf_handle_t	ht_impl_sf_t ht_impl_data_sf_t
Dependency Functions	
ht_impl_sf_handle_t -> sf_data_cb()	
Functions	
ht_impl_init_sf() ht_impl_set_sf_attrs() ht_impl_stop_sf() ht_impl_state_get_sf() ht_impl_data_pushing_and_indication_sf_event()	ht_impl_clear_sf() ht_impl_start_sf() ht_impl_config_sf() ht_impl_data_read_sf()

© 2017 Dialog Semiconductor

8.2.5.1 Data Structures and Types

ht_impl_sf_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable sensor fusion service data flow: HT_IMPL_DISABLE, HT_IMPL_ENABLE

ht_impl_sf_t

Type Definition Name	Description
ht_impl_sf_t	Configuration state structure for sensor fusion service data flow: ht_impl_sf_config_t

hc_svcs_sf_attrs_t

Data Structure Fields	Type	Description
mode	uint8_t	Sensor combination mode: HT_IMPL_SF_MODE_GYR, HT_IMPL_SF_MODE_ACC_GYR, HT_IMPL_SF_MODE_ACC_MAG, HT_IMPL_SF_MODE_ACC_GYR_MAG
sf_rate	uint8_t	Sensor fusion service data rate: HT_IMPL_SF_RATE_INVALID, HT_IMPL_SF_RATE_0_78HZ, HT_IMPL_SF_RATE_1_56HZ, HT_IMPL_SF_RATE_3_12HZ, HT_IMPL_SF_RATE_6_25HZ, HT_IMPL_SF_RATE_12_5HZ, HT_IMPL_SF_RATE_25HZ, HT_IMPL_SF_RATE_50HZ
acc_range	uint8_t	Accelerometer sensor data range: HT_IMPL_ACC_RANGE_2G, HT_IMPL_ACC_RANGE_4G, HT_IMPL_ACC_RANGE_8G, HT_IMPL_ACC_RANGE_16G
acc_rate	uint8_t	Accelerometer sensor data rate: HT_IMPL_ACC_RATE_INVALID, HT_IMPL_ACC_RATE_0_78HZ, HT_IMPL_ACC_RATE_1_56HZ, HT_IMPL_ACC_RATE_3_12HZ, HT_IMPL_ACC_RATE_6_25HZ, HT_IMPL_ACC_RATE_12_5HZ, HT_IMPL_ACC_RATE_25HZ, HT_IMPL_ACC_RATE_50HZ, HT_IMPL_ACC_RATE_100HZ
gyr_range	uint8_t	Gyroscope sensor data range: HT_IMPL_GYR_RANGE_2000_DEG_SEC, HT_IMPL_GYR_RANGE_1000_DEG_SEC, HT_IMPL_GYR_RANGE_500_DEG_SEC, HT_IMPL_GYR_RANGE_250_DEG_SEC, HT_IMPL_GYR_RANGE_125_DEG_SEC

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
gyr_rate	uint8_t	Gyroscope sensor data rate: HT_IMPL_GYR_RATE_INVALID, HT_IMPL_GYR_RATE_25HZ, HT_IMPL_GYR_RATE_50HZ, HT_IMPL_GYR_RATE_100HZ
mag_rate	uint8_t	Magnetometer sensor data rate: HT_IMPL_MAG_RATE_INVALID, HT_IMPL_MAG_RATE_0_78HZ, HT_IMPL_MAG_RATE_1_56HZ, HT_IMPL_MAG_RATE_3_12HZ, HT_IMPL_MAG_RATE_6_25HZ, HT_IMPL_MAG_RATE_12_5HZ, HT_IMPL_MAG_RATE_25HZ, HT_IMPL_MAG_RATE_50HZ, HT_IMPL_MAG_RATE_100HZ
beta_a	uint16_t	Control value for relative weight of accelerometer sensor data.
beta_m	uint16_t	Control value for relative weight of magnetometer sensor data.

ht_impl_data_sf_t

Data Structure Fields	Type	Description
quaternion_w	int32_t	Sensor fusion service data quartenion w.
quaternion_x	int32_t	Sensor fusion service data quartenion x.
quaternion_y	int32_t	Sensor fusion service data quartenion y.
quaternion_z	int32_t	Sensor fusion service data quartenion z.

ht_impl_sf_handle_t

Data Structure Fields	Type	Description
sf_data_cb	ht_impl_data_rdy_cb_t	Sensor fusion service data ready function pointer.
config	ht_impl_sf_config_t	Sensor fusion service data flow instance configuration.
next	ht_impl_sf_handle_t	Next sensor fusion service data flow instance handle in the list.

© 2017 Dialog Semiconductor

8.2.5.2 Dependency Functions

ht_impl_sf_handle_t -> sf_data_cb()

Function Name	`ht_impl_error_t (*sf_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when sensor fusion service data need to be sent over a defined data flow instance of type `ht_impl_sf_handle_t`.
Parameters	`data` the sensor fusion service data samples of the following structure type: `ht_impl_data_sf_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.5.3 Functions

ht_impl_init_sf()

Function Name	`ht_impl_error_t ht_impl_init_sf(ht_impl_sf_handle_t* handle, ht_impl_data_rdy_cb_t sf_data_cb)`
Function Description	Initializes sensor fusion service data flow instance, by setting initial values to the internal variables and structures of the particular Health Toolbox implementation. Callback function may be set to the `ht_impl_sf_handle_t` instance handle, so that data indications can be sent.
Parameters	`handle` the handle structure of the data flow instance `sf_data_cb` the callback function indicating sensor fusion service data
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_clear_sf()

Function Name	`ht_impl_error_t ht_impl_clear_sf(ht_impl_sf_handle_t* handle)`
Function Description	Clears sensor fusion service data flow instance. The configuration of sensor fusion service is determined by the rest of the data flow instances (if any), and it is updated accordingly.
Parameters	`handle` the handle structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_set_sf_attrs()

Function Name	`ht_impl_error_t ht_impl_set_sf_attrs(ht_impl_sf_attrs_t* attrs)`
Function Description	Sets sensor fusion service attributes, which are applied to every active data flow instance.
Parameters	`attrs` the structure of the given sensor fusion service attributes
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_start_sf()

© 2017 Dialog Semiconductor

Function Name	`ht_impl_error_t ht_impl_start_sf(void)`
Function Description	Starts sensor fusion service based on the configured data flow instances.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_stop_sf()

Function Name	`ht_impl_error_t ht_impl_stop_sf(void)`
Function Description	Stops sensor fusion service, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_sf()

Function Name	`ht_impl_error_t ht_impl_config_sf(ht_impl_sf_handle_t* handle, ht_impl_sf_config_t* config)`
Function Description	Configures sensor fusion service data flow instance, i.e. enable/disable sensor fusion service data indication for the particular instance. Data indications are sent via `sf_data_cb()` callback function..
Parameters	`handle` the handle structure of the data flow instance `config` the sensor fusion service configuration structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_state_get_sf()

Function Name	`ht_impl_error_t ht_impl_state_get_sf(ht_impl_sf_t* state)`
Function Description	Gets configuration status of sensor fusion service, i.e. the sensor fusion service configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the sensor fusion service data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_sf()

Function Name	`ht_impl_error_t ht_impl_data_read_sf(ht_impl_sf_handle_t* handle, ht_impl_data_sf_t* data)`
Function Description	Reads sensor fusion service data. If sensor fusion service is enabled for the particular data flow instance, last sensor fusion service data value is returned (through `data` parameter). Sensor fusion service data is updated based on the data rate which is set using `ht_impl_set_sf_attrs()`.
Parameters	`handle` the handle structure of the data flow instance `data` the address to store sensor fusion service data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error

© 2017 Dialog Semiconductor

Notes	

ht_impl_data_pushing_and_indication_sf_event()

Function Name	`ht_impl_error_t ht_impl_data_pushing_and_indication_sf_event( ht_impl_data_sf_t* data, uint16_t size)`
Function Description	Pushes sensor fusion service data and indicate (i.e. trigger a data-indication event) data over the initialized data flows, based on their configuration.
Parameters	`data` the sensor fusion service data being pushed `size` the size of data (i.e. number of data samples) being pushed
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.6 Heart Rate Estimation Service HT-Implementation – API Specification

All functions and respective data structures/types being defined in the API are summarized below:

Table 23: Heart Rate Estimation Service HT-Implementation API

Data Structures and Types	
ht_impl_hr_config_t ht_impl_data_hr_t	ht_impl_hr_t ht_impl_hr_handle_t
Dependency Functions	
ht_impl_hr_handle_t -> hr_data_cb()	ht_impl_hr_handle_t -> hri_data_cb()
Functions	
ht_impl_init_hr() ht_impl_set_hr_opmode() ht_impl_start_hr() ht_impl_config_hr_acquisition_event() ht_impl_state_get_hr() ht_impl_data_acquisition_hr_event()	ht_impl_clear_hr() ht_impl_set_hr_rate() ht_impl_stop_hr() ht_impl_config_hr() ht_impl_data_read_hr() ht_impl_data_pushing_hr_event()

8.2.6.1 Data Structures and Types

ht_impl_hr_config_t

Data Structure Fields	Type	Description
`enable`	`bool`	Enable/disable heart rate estimation service data flow: `HT_IMPL_DISABLE, HT_IMPL_ENABLE`

ht_impl_hr_t

Type Definition Name	Description
`ht_impl_hr_t`	Configuration state structure for heart rate estimation service data flow: `ht_impl_hr_config_t`

© 2017 Dialog Semiconductor

ht_impl_data_hr_t

Data Structure Fields	Type	Description
heart_rate	uint8_t	Heart rate (in bpm).

ht_impl_hr_handle_t

Data Structure Fields	Type	Description
hr_data_cb	ht_impl_data_rdy_cb_t	Heart rate estimation service (rate/epoch) data ready function pointer.
hri_data_cb	ht_impl_data_rdy_cb_t	Heart rate estimation service instant data ready function pointer.
config	ht_impl_hr_config_t	Heart rate estimation service data flow instance configuration.
next	ht_impl_hr_handle_t	Next heart rate estimation service data flow instance handle in the list.

© 2017 Dialog Semiconductor

8.2.6.2 Dependency Functions

ht_impl_hr_handle_t -> hr_data_cb()

Function Name	`ht_impl_error_t (*hr_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when heart rate estimation service (rate/epoch) data need to be sent over a defined data flow instance of type `ht_impl_hr_handle_t`.
Parameters	`data` the heart rate estimation service data samples of the following structure type: `ht_impl_data_hr_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_hr_handle_t -> hri_data_cb()

Function Name	`ht_impl_error_t (*hri_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when heart rate estimation service instant data need to be sent over a defined data flow instance of type `ht_impl_hr_handle_t`.
Parameters	`data` the heart rate estimation service data samples of the following structure type: `ht_impl_data_hr_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.6.3 Functions

ht_impl_init_hr()

Function Name	`ht_impl_error_t ht_impl_init_hr(ht_impl_hr_handle_t* handle, ht_impl_data_rdy_cb_t hr_data_cb, ht_impl_data_rdy_cb_t hri_data_cb)`
Function Description	Initializes heart rate estimation service data flow instance, by setting initial values to the internal variables and structures of the particular Health Toolbox implementation. Callback functions may be set to the `ht_impl_hr_handle_t` instance handle, so that data indications can be sent.
Parameters	`handle` the handle structure of the data flow instance `hr_data_cb` the callback function indicating heart rate estimation service (rate/epoch) data `hri_data_cb` the callback function indicating heart rate estimation service instant data
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_clear_hr()

© 2017 Dialog Semiconductor

Function Name	`ht_impl_error_t ht_impl_clear_hr(ht_impl_hr_handle_t* handle)`
Function Description	Clears heart rate estimation service data flow instance. The configuration of heart rate estimation service is determined by the rest of the data flow instances (if any), and it is updated accordingly.
Parameters	`handle` the handle structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_set_hr_opmode()

Function Name	`ht_impl_error_t ht_impl_set_hr_opmode(uint8_t op_mode)`
Function Description	Sets heart rate estimation service operation mode, which is applied to every active data flow instance.
Parameters	`op_mode` the value of the heart rate estimation service operation mode `HT_IMPL_HR_OPMODE_REGULAR`, `HT_IMPL_HR_OPMODE_INTERMITTENT`
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_set_hr_rate()

Function Name	`ht_impl_error_t ht_impl_set_hr_rate(uint32_t rate)`
Function Description	Sets heart rate estimation service indication (epoch) data rate, which is applied to every active data flow instance.
Parameters	`rate` the value of the heart rate estimation service data rate
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_start_hr()

Function Name	`ht_impl_error_t ht_impl_start_hr(void)`
Function Description	Starts heart rate estimation service based on the configured data flow instances.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_stop_hr()

Function Name	`ht_impl_error_t ht_impl_stop_hr(void)`
Function Description	Stops heart rate estimation service, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_hr_acquisition_event()

© 2017 Dialog Semiconductor

Function Name	`ht_impl_error_t ht_impl_config_hr_acquisition_event(bool enable, uint32_t rate)`
Function Description	Configures heart rate estimation service data (external) acquisition event source. If external source is disabled, then an internal source is used. Acquisition event rate defines the base rate based on which the data indication rates of the initialized data flow instances are determined. External acquisition event source can be set only when all data flow instances are disabled.
Parameters	`enable` enable/disable (external) acquisition event source `HT_IMPL_DISABLE, HT_IMPL_ENABLE` `rate` the base rate (in msec) of the acquisition event
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_hr()

Function Name	`ht_impl_error_t ht_impl_config_hr(ht_impl_hr_handle_t* handle, ht_impl_hr_config_t* config)`
Function Description	Configures heart rate estimation service data flow instance, i.e. enable/disable heart rate estimation service data indication for the particular instance. Data indications are sent via `hr_data_cb()` and `hri_data_cb` callback functions, indicating per epoch/interval heart rate estimation service data and instant ones, respectively. A moving average is used for the produced/indicated heart rate estimation service data.
Parameters	`handle` the handle structure of the data flow instance `config` the heart rate estimation service configuration structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_state_get_hr()

Function Name	`ht_impl_error_t ht_impl_state_get_hr(ht_impl_hr_t* state)`
Function Description	Gets configuration status of heart rate estimation service, i.e. the heart rate estimation service configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the heart rate estimation service data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_hr()

Function Name	`ht_impl_error_t ht_impl_data_read_hr(ht_impl_hr_handle_t* handle, ht_impl_data_hr_t* data)`
Function Description	Reads heart rate estimation service data. If heart rate estimation service is enabled for the particular data flow instance, then in case indication data rate is set, last instant heart rate estimation service data value is returned (through `data` parameter), otherwise a force-read is performed and new heart rate estimation service data indications are sent via `hr_data_cb()` and `hri_data_cb()`.
Parameters	`handle` the handle structure of the data flow instance `data` the address to store heart rate estimation service data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error

© 2017 Dialog Semiconductor

Notes	

ht_impl_data_acquisition_hr_event()

Function Name	`ht_impl_error_t ht_impl_data_acquisition_hr_event(void)`
Function Description	Triggers heart rate estimation service data (external) acquisition event. Based on the configured indication rate of heart rate estimation service, the data of the latter are (force-)read and data indications are sent over the initialized data flow instances and based on their configuration (i.e. indication data rates).
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_pushing_hr_event()

Function Name	`ht_impl_error_t ht_impl_data_pushing_hr_event(` `ht_impl_data_hr_t* data, uint16_t size)`
Function Description	Pushes heart rate estimation service data to Health Toolbox implementation of heart rate estimation service data flows. In the particular implementation only the last sample is pushed to the internally maintained moving average buffer.
Parameters	`data` the heart rate estimation service data being pushed `data` the size of data (i.e. number of data samples) being pushed
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.7 Step Counting Service HT-Implementation – API Specification

All functions and respective data structures/types being defined in the API are summarized below:

Table 24: Step Counting Service HT-Implementation API

Data Structures and Types	
ht_impl_sc_config_t ht_impl_data_sc_t	ht_impl_sc_t ht_impl_hr_handle_t
Dependency Functions	
ht_impl_sc_handle_t -> sc_data_cb()	
Functions	
ht_impl_init_sc() ht_impl_set_sc_opmode() ht_impl_stop_sc() ht_impl_config_sc() ht_impl_data_read_sc() ht_impl_data_pushing_sc_event()	ht_impl_clear_sc() ht_impl_start_sc() ht_impl_config_sc_acquisition_event() ht_impl_state_get_sc() ht_impl_data_acquisition_sc_event() ht_impl_data_indication_sc_event()

© 2017 Dialog Semiconductor

8.2.7.1 Data Structures and Types

ht_impl_sc_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable step counting service data flow: HT_IMPL_DISABLE, HT_IMPL_ENABLE
ind_rate	uint32_t	Step counting service indication data rate: HT_IMPL_SC_RATE_0_125HZ, HT_IMPL_SC_RATE_0_25HZ, HT_IMPL_SC_RATE_0_5HZ, HT_IMPL_SC_RATE_1HZ, HT_IMPL_SC_RATE_2HZ, HT_IMPL_SC_RATE_INVALID

ht_impl_sc_t

Type Definition Name	Description
ht_impl_sc_t	Configuration state structure for step counting service data flow: ht_impl_sc_config_t

ht_impl_data_sc_t

Data Structure Fields	Type	Description
steps	uint32_t	Number of steps counted.

ht_impl_hr_handle_t

Data Structure Fields	Type	Description
least_ind_rate	uint32_t	Least rate required for the indication of multiple step counting service data.
sc_data_cb	ht_impl_data_rdy_cb_t	Step counting service data ready function pointer.
config	ht_impl_sc_config_t	Step counting service data flow instance configuration.
ind_samples_passed	uint8_t	Number of step counting service data samples passed.
ind_samples_to_wait	uint8_t	Number of step counting service data samples to wait .for, before indication.
ref_data	ht_impl_data_sc_t	Step counting service data reference values
total_data	ht_impl_data_sc_t	Step counting service data total values
next	ht_impl_sc_handle_t	Next setp counting service data flow instance handle in the list.

© 2017 Dialog Semiconductor

8.2.7.2 Dependency Functions

ht_impl_sc_handle_t -> sc_data_cb()

Function Name	`ht_impl_error_t (*sc_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when step counting service data need to be sent over a defined data flow instance of type `ht_impl_sc_handle_t`.
Parameters	`data` the step counting service data samples of the following structure type: `ht_impl_data_sc_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.7.3 Functions

ht_impl_init_sc()

Function Name	`ht_impl_error_t ht_impl_init_sc(ht_impl_sc_handle_t* handle, ht_impl_data_rdy_cb_t sc_data_cb)`
Function Description	Initializes step counting service data flow instance, by setting initial values to the internal variables and structures of the particular Health Toolbox implementation. Callback function may be set to the `ht_impl_sc_handle_t` instance handle, so that data indications can be sent.
Parameters	`handle` the handle structure of the data flow instance `sc_data_cb` the callback function indicating total step counting service data, since last activation or reset of step counting service
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_clear_sc()

Function Name	`ht_impl_error_t ht_impl_clear_sc(ht_impl_sc_handle_t* handle)`
Function Description	Clears step counting service data flow instance. The configuration of step counting service is determined by the rest of the data flow instances (if any), and it is updated accordingly.
Parameters	`handle` the handle structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_set_sc_opmode()

Function Name	`ht_impl_error_t ht_impl_set_sc_opmode(uint8_t op_mode)`
Function Description	Sets step counting service operation mode, which is applied to every active data flow instance.

© 2017 Dialog Semiconductor

Parameters	op_mode the value of the step counting service operation mode HT_IMPL_SC_OPMODE_SENSITIVE, // light-weighted, small persons HT_IMPL_SC_OPMODE_REGULAR, // normal HT_IMPL_SC_OPMODE_ROBUST // reducing great amounts of false positive step detections
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_start_sc()

Function Name	ht_impl_error_t ht_impl_start_sc(void)
Function Description	Starts step counting service based on the configured data flow instances.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_stop_sc()

Function Name	ht_impl_error_t ht_impl_stop_sc(void)
Function Description	Stops step counting service, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_config_sc_acquisition_event()

Function Name	ht_impl_error_t ht_impl_config_sc_acquisition_event(bool enable, uint32_t rate)
Function Description	Configures step counting service data (external) acquisition event source. If external source is disabled, then an internal source is used. Acquisition event rate defines the base rate based on which the data indication rates of the initialized data flow instances are determined. External acquisition event source can be set only when all data flow instances are disabled.
Parameters	enable enable/disable (external) acquisition event source HT_IMPL_DISABLE, HT_IMPL_ENABLE rate the base rate (in msec) of the acquisition event HT_IMPL_SC_RATE_0_125HZ, HT_IMPL_SC_RATE_0_25HZ, HT_IMPL_SC_RATE_0_5HZ, HT_IMPL_SC_RATE_1HZ, HT_IMPL_SC_RATE_2HZ
Return Values	HT_IMPL_SUCCESS for successful execution; ≥ HT_IMPL_FAIL for error
Notes	

ht_impl_config_sc()

Function Name	ht_impl_error_t ht_impl_config_sc(ht_impl_sc_handle_t* handle, ht_impl_sc_config_t* config, uint32_ least_ind_rate)

© 2017 Dialog Semiconductor

Function Description	Configures step counting service data flow instance, i.e. enable/disable data indication, and set the required indication data rate of step counting service for the particular instance. Furthermore, sets the least indication data rate required for the particular data flow instance, which in case it is slower than the configured step counting service indication rate, an internal intermediate storage (i.e. FIFO) is used and multiple data (samples) per indication event are sent. Data indications are sent via `sc_data_cb()` callback function..

Parameters	`handle` the handle structure of the data flow instance
	`config` the step counting service configuration structure of the data flow instance
	`least_ind_rate` the least indication data rate of step counting service
	`HT_IMPL_SC_RATE_0_125HZ,`
	`HT_IMPL_SC_RATE_0_25HZ,`
	`HT_IMPL_SC_RATE_0_5HZ,`
	`HT_IMPL_SC_RATE_1HZ,`
	`HT_IMPL_SC_RATE_2HZ,`
	`HT_IMPL_SC_RATE_INVALID`

Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_state_get_sc()

Function Name	`ht_impl_error_t ht_impl_state_get_sc(ht_impl_sc_t* state)`
Function Description	Gets configuration status of step counting service, i.e. the step counting service configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the step counting service data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_sc()

Function Name	`ht_impl_error_t ht_impl_data_read_sc(ht_impl_sc_handle_t* handle, ht_impl_data_sc_t* data)`
Function Description	Reads step counting service data. If step counting service is enabled for the particular data flow instance, then in case indication data rate is set, last read/indicated total step counting service data values (since last activation or reset of step counting service) are returned (through `data` parameter), otherwise a force-read is performed and new step counting service data is acquired via `sc_data_cb()`.
Parameters	`handle` the handle structure of the data flow instance `data` the address to store step counting service data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_acquisition_sc_event()

Function Name	`ht_impl_error_t ht_impl_data_acquisition_sc_event(void)`

© 2017 Dialog Semiconductor

Function Description	Triggers step counting service data (external) acquisition event. Based on the configured indication rate of step counting service, the data of the latter are (force-) read and stored to the internal storage (i.e. FIFO). If the internal storage (i.e. FIFO) is full or reaches the (watermark) level indicated by the data-flow-instance-wide configured least indication rate, then data indications are sent over the initialized data flow instances, based on their configuration (i.e. indication data rates), and the data storage is flushed.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_pushing_sc_event()

Function Name	`ht_impl_error_t ht_impl_data_pushing_sc_event(` `ht_impl_data_sc_t* data, uint16_t size)`
Function Description	Pushes step counting service data to Health Toolbox implementation of step counting service data flows. If the internal storage (i.e. FIFO) is full or reaches the (watermark) level indicated by the data-flow-instance-wide configured least indication rate, then data indications are sent over the initialized data flow instances, based on their configuration (i.e. indication data rates), and the data storage is flushed.
Parameters	`data` the step counting service data being pushed `data` the size of data (i.e. number of data samples) being pushed
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_indication_sc_event()

Function Name	`ht_impl_error_t ht_impl_data_indication_sc_event(void)`
Function Description	Triggers step counting service data indication event, flushing internal storage after processing and sending data indications over the initialized data flow instances, based on their configuration (i.e. indication data rates).
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.8 Calories Counting Service HT-Implementation – API Specification

All functions and respective data structures/types being defined in the API are summarized below:

Table 25: Calories Counting Service HT-Implementation API

Data Structures and Types	
ht_impl_cc_config_t ht_impl_cc_attrs_t ht_impl_cc_handle_t	ht_impl_cc_t ht_impl_data_cc_t
Dependency Functions	
ht_impl_cc_handle_t -> cc_data_cb()	ht_impl_cc_handle_t -> cct_data_cb()

© 2017 Dialog Semiconductor

Functions	
ht_impl_init_cc()	ht_impl_clear_cc()
ht_impl_set_cc_attrs()	ht_impl_start_cc()
ht_impl_stop_cc()	ht_impl_config_cc()
ht_impl_state_get_cc()	ht_impl_data_read_cc()
ht_impl_data_reset_cc()	ht_impl_data_pushing_and_indication_cc_event()

8.2.8.1 Data Structures and Types

ht_impl_cc_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable calories counting service data flow: HT_IMPL_DISABLE, HT_IMPL_ENABLE

ht_impl_cc_t

Type Definition Name	Description
ht_impl_cc_t	Configuration state structure for calories counting service data flow: ht_impl_cc_config_t

ht_impl_cc_attrs_t

Data Structure Fields	Type	Description
mode	uint8_t	Calories counting mode. HT_IMPL_CC_OPMODE_REGULAR, HT_IMPL_CC_OPMODE_ROBUST
rate	uint32_t	Calories counting data rate. HT_IMPL_CC_RATE_1_PER_5MIN_HZ, HT_IMPL_CC_RATE_1_PER_4MIN_HZ, HT_IMPL_CC_RATE_1_PER_3MIN_HZ, HT_IMPL_CC_RATE_1_PER_2MIN_HZ, HT_IMPL_CC_RATE_1_PER_1MIN_HZ, HT_IMPL_CC_RATE_1_PER_50SEC_HZ, HT_IMPL_CC_RATE_1_PER_40SEC_HZ, HT_IMPL_CC_RATE_1_PER_30SEC_HZ, HT_IMPL_CC_RATE_1_PER_20SEC_HZ, HT_IMPL_CC_RATE_1_PER_10SEC_HZ

ht_impl_data_cc_t

Data Structure Fields	Type	Description
calories	int32_t	Amount of calories burned.
distance	int32_t	Distance covered.
num_of_epochs	uint32_t	Number of calories epochs included.
epoch_index	uint32_t	First epoch number calories counting data refer to.

© 2017 Dialog Semiconductor

ht_impl_cc_handle_t

Data Structure Fields	Type	Description
cc_data_cb	ht_impl_data_rdy_cb_t	Calories counting service data ready function pointer.
cct_data_cb	ht_impl_data_rdy_cb_t	Calories counting service total data ready function pointer.
config	ht_impl_cc_config_t	Calories counting service data flow instance configuration.
next	ht_impl_cc_handle_t	Next calories counting service data flow instance handle in the list.

© 2017 Dialog Semiconductor

8.2.8.2 Dependency Functions

ht_impl_cc_handle_t -> cc_data_cb()

Function Name	`ht_impl_error_t (*cc_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when calories counting service data need to be sent over a defined data flow instance of type `ht_impl_cc_handle_t`.
Parameters	`data` the calories counting service data samples of the following structure type: `ht_impl_data_cc_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_cc_handle_t -> cct_data_cb()

Function Name	`ht_impl_error_t (*cct_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when calories counting service total data need to be sent over a defined data flow instance of type `ht_impl_cc_handle_t`.
Parameters	`data` the calories counting service (total) data samples of the following structure type: `ht_impl_data_cc_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.8.3 Functions

ht_impl_init_cc()

Function Name	`ht_impl_error_t ht_impl_init_cc(ht_impl_cc_handle_t* handle, ht_impl_data_rdy_cb_t cc_data_cb, ht_impl_data_rdy_cb_t cct_data_cb)`
Function Description	Initializes calories counting service data flow instance, by setting initial values to the internal variables and structures of the particular Health Toolbox implementation. Callback functions may be set to the `ht_impl_cc_handle_t` instance handle, so that data indications can be sent.
Parameters	`handle` the handle structure of the data flow instance `cc_data_cb` the callback function indicating (per epoch/interval) calories counting service data `cct_data_cb` the callback function indicating total calories counting service data, since last activation or reset of calories counting service
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_clear_cc()

© 2017 Dialog Semiconductor

Function Name	`ht_impl_error_t ht_impl_clear_cc(ht_impl_cc_handle_t* handle)`
Function Description	Clears calories counting service data flow instance. The configuration of calories counting service is determined by the rest of the data flow instances (if any), and it is updated accordingly.
Parameters	`handle` the handle structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_set_cc_attrs()

Function Name	`ht_impl_error_t ht_impl_set_cc_attrs(ht_impl_cc_attrs_t* attrs)`
Function Description	Sets calories counting service attributes, which are applied to every active data flow instance.
Parameters	`attrs` the structure of the given calories counting service attributes
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_start_cc()

Function Name	`ht_impl_error_t ht_impl_start_cc(void)`
Function Description	Starts calories counting service based on the configured data flow instances.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_stop_cc()

Function Name	`ht_impl_error_t ht_impl_stop_cc(void)`
Function Description	Stops calories counting service, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_cc()

Function Name	`ht_impl_error_t ht_impl_config_cc(ht_impl_cc_handle_t* handle, ht_impl_cc_config_t* config)`
Function Description	Configures calories counting service data flow instance, i.e. enable/disable calories counting service data indication for the particular instance. Data indications are sent via `cc_data_cb()` and `cct_data_cb()` callback functions, indicating per-epoch/interval calories counting service data and total ones, respectively.
Parameters	`handle` the handle structure of the data flow instance `config` the calories counting service configuration structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

ht_impl_state_get_cc()

Function Name	`ht_impl_error_t ht_impl_state_get_cc(ht_impl_cc_t* state)`
Function Description	Gets configuration status of calories counting service, i.e. the calories counting service configuration determined by the initialized data flow instances..
Parameters	`state` the configuration status structure of the calories counting service data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_cc()

Function Name	`ht_impl_error_t ht_impl_data_read_cc(ht_impl_cc_handle_t* handle, ht_impl_data_cc_t* data)`
Function Description	Reads calories counting service data. If calories counting service is enabled for the particular data flow instance, total calories counting service data values (i.e. total calories burned and distance covered since last reset of calories counting service) are returned (through `data` parameter). Total calories counting service data is updated based on the data rate which is set using `ht_impl_set_cc_attrs()`.
Parameters	`handle` the handle structure of the data flow instance `data` the address to store calories counting service data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_reset_cc()

Function Name	`ht_impl_error_t ht_impl_data_reset_cc(void)`
Function Description	Resets total calories counting service data values.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_pushing_and_indication_cc_event()

Function Name	`ht_impl_error_t ht_impl_data_pushing_and_indication_cc_event( ht_impl_data_cc_t* data, uint16_t size)`
Function Description	Pushes calories counting service data and indicate (i.e. trigger a data-indication event) data over the initialized data flows, based on their configuration..
Parameters	`data` the calories counting service data being pushed `size` the size of data (i.e. number of data samples) being pushed
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.9 Sleep Quality Monitoring Service HT-Implementation – API Specification

All functions and respective data structures/types being defined in the API are summarized below:

 © 2017 Dialog Semiconductor

Table 26: Sleep Quality Monitoring Service HT-Implementation API

Data Structures and Types	
ht_impl_sq_config_t ht_impl_data_sq_t	ht_impl_sq_t ht_impl_sq_handle_t
Dependency Functions	
ht_impl_sq_handle_t -> sq_data_cb() ht_impl_sq_handle_t -> sqt_data_cb()	ht_impl_sq_handle_t -> squ_data_cb()
Functions	
ht_impl_init_sq() ht_impl_start_sq() ht_impl_config_sq() ht_impl_data_read_sq() ht_impl_data_pushing_and_indication_sq_event()	ht_impl_clear_sq() ht_impl_stop_sq() ht_impl_state_get_sq() ht_impl_data_reset_sq() ht_impl_data_pushing_and_indication_squ_event()

8.2.9.1 Data Structures and Types

ht_impl_sq_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable sleep quality monitoring service data flow: HT_IMPL_DISABLE, HT_IMPL_ENABLE

ht_impl_sq_t

Type Definition Name	Description
ht_impl_sq_t	Configuration state structure for sleep quality monitoring service data flow: ht_impl_sq_config_t

ht_impl_data_sq_t

Data Structure Fields	Type	Description
invalid	int32_t	Number of consecutive epochs classified as INVALID.
awake	int32_t	Number of consecutive epochs classified as AWAKE.
light_sleep	int32_t	Number of consecutive epochs classified as LIGHT SLEEP.
deep_sleep	int32_t	Number of consecutive epochs classified as DEEP SLEEP.
rem	int32_t	Number of consecutive epochs classified as REM.
error	int32_t	Number of consecutive epochs classified as ERROR.
epoch_index	uint32_t	First epoch number sleep quality monitoring data refer to.

© 2017 Dialog Semiconductor

ht_impl_sq_handle_t

Data Structure Fields	Type	Description
sq_data_cb	ht_impl_data_rdy_cb_t	Sleep quality monitoring service data ready function pointer.
squ_data_cb	ht_impl_data_rdy_cb_t	Sleep quality monitoring service update data ready function pointer.
sqt_data_cb	ht_impl_data_rdy_cb_t	Sleep quality monitoring service total data ready function pointer.
config	ht_impl_sq_config_t	Sleep quality monitoring service data flow instance configuration.
next	ht_impl_sq_handle_t	Next sleep quality monitoring service data flow instance handle in the list.

© 2017 Dialog Semiconductor

8.2.9.2 Dependency Functions

ht_impl_sq_handle_t -> sq_data_cb()

Function Name	`ht_impl_error_t (*sq_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when sleep quality monitoring service data need to be sent over a defined data flow instance of type `ht_impl_sq_handle_t`.
Parameters	`data` the sleep quality monitoring service data samples of the following structure type: `ht_impl_data_sq_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_sq_handle_t -> squ_data_cb()

Function Name	`ht_impl_error_t (*squ_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when sleep quality monitoing service update data need to be sent over a defined data flow instance of type `ht_impl_sq_handle_t`.
Parameters	`data` the sleep quality monitoring service (update) data samples of the following structure type: `ht_impl_data_sq_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_sq_handle_t -> sqt_data_cb()

Function Name	`ht_impl_error_t (*sqt_data_cb)(void* data, uint16_t size, ht_impl_data_ref_released_cb_t cb)`
Function Description	(Optional) Callback function which is called when sleep quality monitoing service total data need to be sent over a defined data flow instance of type `ht_impl_sq_handle_t`.
Parameters	`data` the sleep quality monitoring service (total) data samples of the following structure type: `ht_impl_data_sq_t` `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.2.9.3 Functions

ht_impl_init_sq()

 © 2017 Dialog Semiconductor

Function Name	`ht_impl_error_t ht_impl_init_sq(ht_impl_sq_handle_t* handle, ht_impl_data_rdy_cb_t sq_data_cb, ht_impl_data_rdy_cb_t squ_data_cb, ht_impl_data_rdy_cb_t sqt_data_cb)`
Function Description	Initializes sleep quality monitoring service data flow instance, by setting initial values to the internal variables and structures of the particular Health Toolbox implementation. Callback functions may be set to the `ht_impl_sq_handle_t` instance handle, so that data indications can be sent.
Parameters	`handle` the handle structure of the data flow instance `sq_data_cb` the callback function indicating (per epoch) sleep quality monitoring service data `squ_data_cb` the callback function indicating total sleep quality monitoring service data (epoch) update `sqt_data_cb` the callback function indicating total sleep quality monitoring service data, since last activation or reset of sleep quality monitoing service
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_clear_sq()

Function Name	`ht_impl_error_t ht_impl_clear_sq(ht_impl_sq_handle_t* handle)`
Function Description	Clears sleep quality monitoring service data flow instance. The configuration of sleep quality monitoring service is determined by the rest of the data flow instances (if any), and it is updated accordingly.
Parameters	`handle` the handle structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_start_sq()

Function Name	`ht_impl_error_t ht_impl_start_sq(void)`
Function Description	Starts sleep_quality monitoring service based on the configured data flow instances.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_stop_sq()

Function Name	`ht_impl_error_t ht_impl_stop_sq(void)`
Function Description	Stops sleep quality monitoring service, and the respective data indication process driven by the active data flow instances, as well.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_config_sq()

Function Name	`ht_impl_error_t ht_impl_config_sq(ht_impl_sq_handle_t* handle, ht_impl_sq_config_t* config)`

© 2017 Dialog Semiconductor

Function Description	Configures sleep quality monitoring service data flow instance, i.e. enable/disable sleep quality monitoring service data indication for the particular instance. Data indications are sent via `sq_data_cb()`, `squ_data_cb()` and `sqt_data_cb()` callback functions, indicating per-epoch sleep quality monitoring service data, update and total ones, respectively..
Parameters	`handle` the handle structure of the data flow instance `config` the sleep quality monitoring service configuration structure of the data flow instance
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_state_get_sq()

Function Name	`ht_impl_error_t ht_impl_state_get_sq(ht_impl_sq_t* state)`
Function Description	Gets configuration status of sleep quality monitoring service, i.e. the sleep quality monitoring service configuration determined by the initialized data flow instances.
Parameters	`state` the configuration status structure of the sleep quality monitoring service data flow
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_read_sq()

Function Name	`ht_impl_error_t ht_impl_data_read_sq(ht_impl_sq_handle_t* handle, ht_impl_data_sq_t* data)`
Function Description	Reads sleep quality monitoring service data. If sleep quality monitoring service is enabled for the particular data flow instance, total sleep quality monitoring service data values are returned (through `data` parameter). Total sleep quality monitoring service data is updated based on an 1-sample-per-1-min data rate.
Parameters	`handle` the handle structure of the data flow instance `data` the address to store sleep quality monitoring service data being read
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_reset_sq()

Function Name	`ht_impl_error_t ht_impl_data_reset_sq(void)`
Function Description	Resets total sleep quality monitoring service data values.
Parameters	None
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

ht_impl_data_pushing_and_indication_sq_event()

Function Name	`ht_impl_error_t ht_impl_data_pushing_and_indication_sq_event( ht_impl_data_sq_t* data, uint16_t size)`
Function Description	Pushes sleep quality monitoring service (epoch) data and indicate (i.e. trigger a data-indication event) data over the initialized data flows, based on their configuration.

© 2017 Dialog Semiconductor

Parameters	`data`	the sleep quality monitoring service (epoch) data being pushed
	`size`	the size of data (i.e. number of data samples) being pushed
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error	
Notes		

ht_impl_data_pushing_and_indication_squ_event()

Function Name	`ht_impl_error_t ht_impl_data_pushing_and_indication_squ_event( ht_impl_data_sq_t* data, uint16_t size)`
Function Description	Pushes sleep quality monitoring service (epoch update) data and indicate (i.e. trigger a data-indication event) data over the initialized data flows, based on their configuration.
Parameters	`data` the sleep quality monitoring service (epoch update) data being pushed `size` the size of data (i.e. number of data samples) being pushed
Return Values	`HT_IMPL_SUCCESS` for successful execution; ≥ `HT_IMPL_FAIL` for error
Notes	

8.3 Timer Component

Health Toolbox uses a timer abstruction for creating timer events, called “Timer”. The latter implements the creation of timer instances over the supported by the operating system timer-provision framework.

The API provided by Timer component includes functions for creating timer instances, configuring their periodicity and controlling in general their operation. Those functions and respective data structures/types being defined in the API are summarized below:

Table 27: Timer Component API

Data Structures and Types	
tmr_error_t	tmr_handle_t
Dependency Functions	
tmr_handle_t -> cb()	
Functions	
tmr_init() tmr_clear() tmr_start() tmr_startAt() tmr_restart() tmr_stop() tmr_isrunning()	tmr_create() tmr_clearAt() tmr_start_ms() tmr_startAt_ms() tmr_restartAt() tmr_stopAt() tmr_getNow()

© 2017 Dialog Semiconductor

8.3.1 Data Structures and Types

tmr_error_t

Type Definition Name	Description
tmr_error_t	Execution status of a function: TMR_SUCCESS, TMR_FAIL

tmr_handle_t

Data Structure Fields	Type	Description
timer	OS_TIMER	The OS_TIMER instance the abstraction refers to.
cb	TimerCallbackFunction_t	Timer-fired event function pointer.
next	tmr_handle_t	Pointer to the next timer instance in the list.

© 2017 Dialog Semiconductor

8.3.2 Dependency Functions

tmr_handle_t -> cb()

Function Name	`void (*cb)(TimerHandle_t xTimer)`
Function Description	(Optional) Callback function for indicatinf a “timer-fired” event.
Parameters	`xTimer` the OS timer instance which triggers the “timer-fired” event
Return Values	None
Notes	

8.3.3 Functions

tmr_init()

Function Name	`tmr_error_t tmr_init(void)`
Function Description	Initializes Timer component, setting initial values to the internal variables and structures and creating a FreeRTOS task for OS timers control.
Parameters	None
Return Values	`TMR_SUCCESS` for successful execution; ≥ `TMR_FAIL` for error
Notes	

tmr_create()

Function Name	`tmr_error_t tmr_create(tmr_handle_t* handle, bool periodic, TimerCallbackFunction_t cb)`
Function Description	Creates a new Timer instance defined by the given handle, appending it to the internal Timer instance list and setting its periodic/one-shot mode and callback function when the timer is fired.
Parameters	`handle` the handle structure for the Timer instance `periodic` indication of a periodic timer (not one-shot) `cb` the function called when the timer is fired
Return Values	`TMR_SUCCESS` for successful execution; ≥ `TMR_FAIL` for error
Notes	

tmr_clear()

Function Name	`tmr_error_t tmr_clear(tmr_handle_t* handle)`
Function Description	Clears Timer instance defined by the given handle, removing it from the internal Timer instance list and freeing previously allocated resources. There is no wait time for the command to be sent.
Parameters	`handle` the handle structure for the Timer instance
Return Values	`TMR_SUCCESS` for successful execution; ≥ `TMR_FAIL` for error
Notes	

tmr_clearAt()

Function Name	`tmr_error_t tmr_clearAt(tmr_handle_t* handle, const uint32_t waitTime)`

© 2017 Dialog Semiconductor

Function Description	Clears Timer instance defined by the given handle, removing it from the internal Timer instance list and freeing previously allocated resources. There is a maximum wait time until command is sent.
Parameters	`handle` the handle structure for the Timer instance `waitTime` max time in ticks to wait until command is sent
Return Values	`TMR_SUCCESS` for successful execution; ≥ `TMR_FAIL` for error
Notes	

tmr_start()

Function Name	`tmr_error_t tmr_start(tmr_handle_t* handle, uint32_t interval)`
Function Description	Starts a Timer instance configured with a specific time interval (in ticks). There is no wait time for the command to be sent.
Parameters	`handle` the handle structure for the Timer instance `interval` the time interval (in ticks) of the Timer instance
Return Values	`TMR_SUCCESS` for successful execution; ≥ `TMR_FAIL` for error
Notes	

tmr_start_ms()

Function Name	`tmr_error_t tmr_start_ms(tmr_handle_t* handle, uint32_t interval)`
Function Description	Starts a Timer instance configured with a specific time interval (in msec). There is no wait time for the command to be sent.
Parameters	`handle` the handle structure for the Timer instance `interval` the time interval (in msec) of the Timer instance
Return Values	`TMR_SUCCESS` for successful execution; ≥ `TMR_FAIL` for error
Notes	

tmr_startAt()

Function Name	`tmr_error_t tmr_startAt(tmr_handle_t* handle, uint32_t interval, const uint32_t waitTime)`
Function Description	Starts a Timer instance configured with a specific time interval (in ticks). There is a maximum wait time until command is sent.
Parameters	`handle` the handle structure for the Timer instance `interval` the time interval (in ticks) of the Timer instance `waitTime` max time in ticks to wait until command is sent
Return Values	`TMR_SUCCESS` for successful execution; ≥ `TMR_FAIL` for error
Notes	

tmr_startAt_ms()

Function Name	`tmr_error_t tmr_startAt_ms(tmr_handle_t* handle, uint32_t interval, const uint32_t waitTime)`
Function Description	Starts a Timer instance configured with a specific time interval (in msec). There is a maximum wait time until command is sent.
Parameters	`handle` the handle structure for the Timer instance `interval` the time interval (in msec) of the Timer instance `waitTime` max time in ticks to wait until command is sent

© 2017 Dialog Semiconductor

Return Values	TMR_SUCCESS for successful execution; ≥ TMR_FAIL for error
Notes	

tmr_restart()

Function Name	tmr_error_t tmr_restart(tmr_handle_t* handle)
Function Description	Restarts a Timer instance having been previously configured with a specific time interval. There is no wait time for the command to be sent.
Parameters	handle the handle structure for the Timer instance
Return Values	TMR_SUCCESS for successful execution; ≥ TMR_FAIL for error
Notes	

tmr_restartAt()

Function Name	tmr_error_t tmr_restartAt(tmr_handle_t* handle, const uint32_t waitTime)
Function Description	Restarts a Timer instance having been previously configured with a specific time interval. There is a maximum wait time until command is sent.
Parameters	handle the handle structure for the Timer instance waitTime max time in ticks to wait until command is sent
Return Values	TMR_SUCCESS for successful execution; ≥ TMR_FAIL for error
Notes	

tmr_stop()

Function Name	tmr_error_t tmr_stop(tmr_handle_t* handle)
Function Description	Stops a Timer instance having been previously started with a specific time interval. There is no wait time for the command to be sent.
Parameters	handle the handle structure for the Timer instance
Return Values	TMR_SUCCESS for successful execution; ≥ TMR_FAIL for error
Notes	

tmr_stopAt()

Function Name	tmr_error_t tmr_stopAt(tmr_handle_t* handle, const uint32_t waitTime)
Function Description	Stops a Timer instance having been previously started with a specific time interval. There is a maximum wait time until command is sent.
Parameters	handle the handle structure for the Timer instance waitTime max time in ticks to wait until command is sent
Return Values	TMR_SUCCESS for successful execution; ≥ TMR_FAIL for error
Notes	

tmr_isrunning()

Function Name	bool tmr_isrunning(tmr_handle_t* handle)
Function Description	Checks if a Timer instance is running.
Parameters	handle the handle structure for the Timer instance
Return Values	true for “running”; false for “not running”
Notes	

© 2017 Dialog Semiconductor

tmr_getNow()

Function Name	`uint32_t tmr_getNow(void)`
Function Description	Get current time in OS ticks.
Parameters	None
Return Values	The current time in OS ticks
Notes	

8.4 Sensor Control Component

The API provided by Sensor Control component includes functions for initializing, controlling and configuring the supported/integrated sensors in terms of their operation and data acquisition. All these functions and respective data structures/types being defined in the API are summarized below:

Table 28: Sensor Control API

Data Structures and Types	
sc_error_t	sc_acc_config_t
sc_gyr_config_t	sc_mag_config_t
sc_step_config_t	sc_temp_config_t
sc_pres_config_t	sc_hum_config_t
sc_opt_config_t	sc_acc_gyr_mag_config_t
sc_acc_data_t	sc_gyr_data_t
sc_mag_data_t	sc_acc_gyr_data_t
sc_acc_mag_data_t	sc_gyr_mag_data_t
sc_acc_gyr_mag_data_t	sc_temp_data_t
sc_pres_data_t	sc_hum_data_t
sc_env_data_t	sc_step_data_t
sc_opt_data_t	sc_calibr_offsets_t
sc_calibr_coeffs_t	sc_calibr_control_t
sc_handle_t	
Dependency Functions	
sc_handle_t -> sc_atomic_begin()	sc_handle_t -> sc_atomic_end()
sc_handle_t -> sc_acc_gyr_mag_fifo_cb()	sc_handle_t -> sc_calibr_acc_offsets_rdy_cb()
sc_handle_t -> sc_calibr_mag_control_new_cb()	sc_handle_t -> sc_calibr_mag_coeffs_rdy_cb()
sc_handle_t -> sc_intr_1_recatch()	

© 2017 Dialog Semiconductor

Functions	
sc_init() sc_acc_cal_init() sc_mag_cal_stop_auto_calibration() sc_acc_cal_set_offsets() sc_mag_cal_set_control() sc_mag_cal_get_coeffs() sc_sensor_set_state() sc_gyr_config() sc_acc_gyr_mag_config() sc_pres_config() sc_step_config() sc_acc_scnd_config()**Error! Reference source not ound.**sc_intr_1_enable() sc_sensors_disable_intrs() sc_acc_read() sc_mag_read() sc_pres_read() sc_env_read() sc_opt_read() sc_env_calc_data_update_latency() sc_time_get_now_ms()	sc_get_handle() sc_mag_cal_init() sc_mag_cal_isInitialized() sc_mag_cal_set_coeffs() sc_acc_cal_get_offsets() sc_mag_cal_get_control() sc_acc_config() sc_mag_config() sc_temp_config() sc_hum_config() sc_opt_config() sc_step_reset() sc_intr_1_catch() sc_sensors_suspend() sc_gyr_read() sc_temp_read() sc_hum_read() sc_step_read() sc_acc_scnd_read() sc_opt_adapt_operation()

8.4.1 Data Structures and Types

sc_error_t

Type Definition Name	Description
`sc_error_t`	Execution status of a function: `SC_SUCCESS, SC_FAIL`

sc_acc_config_t

Data Structure Fields	Type	Description
`enable`	`bool`	Enable/disable accelerometer sensor: `SC_DISABLE, SC_ENABLE`

`mode`	`uint8_t`	Accelerometer sensor operation mode:
		`SC_ACC_OPMODE_NORMAL,`
		`SC_ACC_OPMODE_LOWPOWER_LP1,`
		`SC_ACC_OPMODE_LOWPOWER_LP2,`
		`SC_ACC_OPMODE_LOWPOWER_LP3,`
		`SC_ACC_OPMODE_LOWPOWER_LP4,`
		`SC_ACC_OPMODE_LOWPOWER_LP5,`
		`SC_ACC_OPMODE_LOWPOWER_LP6`

`range`	`uint8_t`	Accelerometer sensor data range:
		`SC_ACCEL_RANGE_2G,`
		`SC_ACCEL_RANGE_4G,`
		`SC_ACCEL_RANGE_8G,`
		`SC_ACCEL_RANGE_16G`

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
ind_rate	uint8_t	Accelerometer sensor indication data rate: SC_ACC_RATE_INVALID, SC_ACC_RATE_0_78HZ, SC_ACC_RATE_1_56HZ, SC_ACC_RATE_3_12HZ, SC_ACC_RATE_6_25HZ, SC_ACC_RATE_12_5HZ, SC_ACC_RATE_25HZ, SC_ACC_RATE_50HZ, SC_ACC_RATE_100HZ, SC_ACC_RATE_200HZ, SC_ACC_RATE_400HZ, SC_ACC_RATE_800HZ, SC_ACC_RATE_1600HZ, SC_ACC_RATE_3200HZ
op_rate	uint8_t	Accelerometer sensor operation data rate: SC_ACC_RATE_INVALID, SC_ACC_RATE_0_78HZ, SC_ACC_RATE_1_56HZ, SC_ACC_RATE_3_12HZ, SC_ACC_RATE_6_25HZ, SC_ACC_RATE_12_5HZ, SC_ACC_RATE_25HZ, SC_ACC_RATE_50HZ, SC_ACC_RATE_100HZ, SC_ACC_RATE_200HZ, SC_ACC_RATE_400HZ, SC_ACC_RATE_800HZ, SC_ACC_RATE_1600HZ, SC_ACC_RATE_3200HZ

sc_gyr_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable gyroscope sensor: SC_DISABLE, SC_ENABLE
mode	uint8_t	Gyroscope sensor operation mode: SC_GYR_OPMODE_NORMAL, SC_GYR_OPMODE_LOWPOWER_LP1, SC_GYR_OPMODE_LOWPOWER_LP2, SC_GYR_OPMODE_LOWPOWER_LP3, SC_GYR_OPMODE_LOWPOWER_LP4, SC_GYR_OPMODE_LOWPOWER_LP5, SC_GYR_OPMODE_LOWPOWER_LP6
range	uint8_t	Gyroscope sensor data range: SC_GYRO_RANGE_2000_DEG_SEC, SC_GYRO_RANGE_1000_DEG_SEC, SC_GYRO_RANGE_500_DEG_SEC, SC_GYRO_RANGE_250_DEG_SEC, SC_GYRO_RANGE_125_DEG_SEC, SC_GYRO_RANGE_INVALID

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
ind_rate	uint8_t	Gyroscope sensor indication data rate: SC_GYR_RATE_INVALID, SC_GYR_RATE_0_78HZ, SC_GYR_RATE_1_56HZ, SC_GYR_RATE_3_12HZ, SC_GYR_RATE_6_25HZ, SC_GYR_RATE_12_5HZ, SC_GYR_RATE_25HZ, SC_GYR_RATE_50HZ, SC_GYR_RATE_100HZ, SC_GYR_RATE_200HZ, SC_GYR_RATE_400HZ, SC_GYR_RATE_800HZ, SC_GYR_RATE_1600HZ, SC_GYR_RATE_3200HZ
op_rate	uint8_t	Gyroscope sensor operation data rate: SC_GYR_RATE_INVALID, SC_GYR_RATE_0_78HZ, SC_GYR_RATE_1_56HZ, SC_GYR_RATE_3_12HZ, SC_GYR_RATE_6_25HZ, SC_GYR_RATE_12_5HZ, SC_GYR_RATE_25HZ, SC_GYR_RATE_50HZ, SC_GYR_RATE_100HZ, SC_GYR_RATE_200HZ, SC_GYR_RATE_400HZ, SC_GYR_RATE_800HZ, SC_GYR_RATE_1600HZ, SC_GYR_RATE_3200HZ

sc_mag_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable magnetometer sensor: SC_DISABLE, SC_ENABLE
mode	uint8_t	Magnetometer sensor operation mode: SC_MAG_OPMODE_NORMAL, SC_MAG_OPMODE_LOWPOWER_LP1, SC_MAG_OPMODE_LOWPOWER_LP2, SC_MAG_OPMODE_LOWPOWER_LP3, SC_MAG_OPMODE_LOWPOWER_LP4, SC_MAG_OPMODE_LOWPOWER_LP5, SC_MAG_OPMODE_LOWPOWER_LP6

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
ind_rate	uint8_t	Magnetometer sensor indication data rate: SC_MAG_RATE_INVALID, SC_MAG_RATE_0_78HZ, SC_MAG_RATE_1_56HZ, SC_MAG_RATE_3_12HZ, SC_MAG_RATE_6_25HZ, SC_MAG_RATE_12_5HZ, SC_MAG_RATE_25HZ, SC_MAG_RATE_50HZ, SC_MAG_RATE_100HZ, SC_MAG_RATE_200HZ, SC_MAG_RATE_400HZ, SC_MAG_RATE_800HZ, SC_MAG_RATE_1600HZ, SC_MAG_RATE_3200HZ
op_rate	uint8_t	Magnetometer sensor operation data rate: SC_MAG_RATE_INVALID, SC_MAG_RATE_0_78HZ, SC_MAG_RATE_1_56HZ, SC_MAG_RATE_3_12HZ, SC_MAG_RATE_6_25HZ, SC_MAG_RATE_12_5HZ, SC_MAG_RATE_25HZ, SC_MAG_RATE_50HZ, SC_MAG_RATE_100HZ, SC_MAG_RATE_200HZ, SC_MAG_RATE_400HZ, SC_MAG_RATE_800HZ, SC_MAG_RATE_1600HZ, SC_MAG_RATE_3200HZ

sc_step_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable step sensor (pedometer): SC_DISABLE, SC_ENABLE
mode	uint8_t	Step sensor (pedometer) operation mode: SC_STEP_OPMODE_SENSITIVE, SC_STEP_OPMODE_REGULAR, SC_STEP_OPMODE_ROBUST

sc_temp_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable temperature sensor: SC_DISABLE, SC_ENABLE
mode	uint8_t	Temperature sensor operation mode: SC_TEMP_OPMODE_NORMAL, SC_TEMP_OPMODE_LOWPOWER

sc_pres_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable pressure sensor: SC_DISABLE, SC_ENABLE

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
mode	uint8_t	Pressure sensor operation mode: SC_PRES_OPMODE_NORMAL, SC_PRES_OPMODE_LOWPOWER

sc_hum_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable humidity sensor: SC_DISABLE, SC_ENABLE
mode	uint8_t	Humidity sensor operation mode: SC_HUM_OPMODE_NORMAL, SC_HUM_OPMODE_LOWPOWER

sc_opt_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable optical sensor: SC_DISABLE, SC_ENABLE
mode	uint8_t	Optical sensor operation mode: SC_OPT_OPMODE_NORMAL, SC_OPT_OPMODE_LOWPOWER

sc_acc_gyr_mag_config_t

Data Structure Fields	Type	Description
acc_config	sc_acc_config_t	Accelerometer sensor configuration.
gyr_config	sc_gyr_config_t	Gyroscope sensor configuration.
mag_config	sc_mag_config_t	Magnetometer sensor configuration.

sc_acc_data_t

Data Structure Fields	Type	Description
xyz.x	int16_t	Accelerometer x-axis sensor data value.
xyz.y	int16_t	Accelerometer y-axis sensor data value.
xyz.z	int16_t	Accelerometer z-axis sensor data value.
calibr_state. uncalibr_data_valid	uint8_t:1	Pre-calibrated data valid flag.
calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
calibr_state. cal_enabled	uint8_t:1	Calibration enabled flag.
calibr_state. cal_settled	uint8_t:1	Calibration settled flag.
calibr_state. reserved4	uint8_t:1	Reserved flag.
calibr_state. cal_mode	uint8_t:3	Calibration mode: SC_ACC_CAL_MODE_NONE, SC_ACC_CAL_MODE_STATIC

© 2017 Dialog Semiconductor

sc_gyr_data_t

Data Structure Fields	Type	Description
xyz.x	int16_t	Gyroscope x-axis sensor data value.
xyz.y	int16_t	Gyroscope y-axis sensor data value.
xyz.z	int16_t	Gyroscope z-axis sensor data value.

sc_mag_data_t

Data Structure Fields	Type	Description
xyz.x	int16_t	Magnetometer x-axis sensor data value.
xyz.y	int16_t	Magnetometer y-axis sensor data value.
xyz.z	int16_t	Magnetometer z-axis sensor data value.
calibr_state. uncalibr_data_valid	uint8_t:1	Pre-calibrated data valid flag.
calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
calibr_state. cal_enabled	uint8_t:1	Calibration enabled flag.
calibr_state. cal_settled	uint8_t:1	Calibration settled flag.
calibr_state. cal_converged	uint8_t:1	Calibration converged flag.
calibr_state. cal_mode	uint8_t:3	Calibration mode: SC_MAG_CAL_MODE_NONE. SC_MAG_CAL_MODE_STATIC, SC_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC, SC_MAG_CAL_MODE_CONT_AUTO_BASIC, SC_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART, SC_MAG_CAL_MODE_CONT_AUTO_SMART

sc_acc_gyr_data_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: SC_VALID_ACC | SC_VALID_GYR
acc_data	sc_acc_data_t	Accelerometer xyz-axis sensor data.
gyr_data	sc_gyr_data_t	Gyroscope xyz-axis sensor data.

sc_acc_mag_data_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: SC_VALID_ACC | SC_VALID_MAG
acc_data	sc_acc_data_t	Accelerometer xyz-axis sensor data.
mag_data	sc_mag_data_t	Magnetometer xyz-axis sensor data.

© 2017 Dialog Semiconductor

sc_gyr_mag_data_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: SC_VALID_GYR | SC_VALID_MAG
gyr_data	sc_gyr_data_t	Gyroscope xyz-axis sensor data.
mag_data	sc_mag_data_t	Magnetometer xyz-axis sensor data.

sc_acc_gyr_mag_data_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: SC_VALID_ACC | SC_VALID_GYR | SC_VALID_MAG
acc_data	sc_acc_data_t	Accelerometer xyz-axis sensor data.
gyr_data	sc_gyr_data_t	Gyroscope xyz-axis sensor data.
mag_data	sc_mag_data_t	Magnetometer xyz-axis sensor data.

sc_temp_data_t

Data Structure Fields	Type	Description
temperature	int32_t	Temperature sensor data (in 0.01 Celsius degrees).

sc_pres_data_t

Data Structure Fields	Type	Description
pressure	uint32_t	Pressure sensor data (in Pa).

sc_hum_data_t

Data Structure Fields	Type	Description
humidity	uint32_t	Humidity sensor data (in % RH as unsigned 32-bit integer in Q22.10 format (22 integer, 10 fractional bits)).

sc_env_data_t

Data Structure Fields	Type	Description
temperature	sc_temp_data_t	Temperature sensor data.
pressure	sc_pres_data_t	Pressure sensor data.
humidity	sc_hum_data_t	Humidity sensor data.

sc_step_data_t

Data Structure Fields	Type	Description
steps	uint32_t	Step sensor (pedometer) data (in steps).

sc_opt_data_t

Data Structure Fields	Type	Description
green	uint16_t	Optical sensor green-light-intensity data.

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
ir	uint16_t	Optical sensor ir-light-intensity data.
valid_data	Bool	Optical sensor data validity indication.

sc_calibr_offsets_t

Data Structure Fields	Type	Description
offset_x	int16_t	x-axis sensor data offset value.
offset_y	int16_t	y-axis sensor data offset value.
offset_z	int16_t	z-axis sensor data offset value

sc_calibr_coeffs_t

Data Structure Fields	Type	Description
q_format	uint8_t	Q format of matrix.
offset_vector	int16_t[3]	Offset vectors for sensor data.
matrix	int16_t[3][3]	Matrix of coefficients to apply to sensor data.

sc_calibr_control_t

Data Structure Fields	Type	Description
flags	uint16_t	Control flags.
ref_mag	uint16_t	Control parameter 1.
mag_range	uint16_t	Control parameter 2.
mag_alpha	uint16_t	Control parameter 3.
mag_delta_thresh	uint16_t	Control parameter 4.
mu_offset	uint16_t	Control parameter 5.
mu_matrix	uint16_t	Control parameter 6.
err_alpha	uint16_t	Control parameter 7.
err_thresh	uint16_t	Control parameter 8.

sc_handle_t

Data Structure Fields	Type	Description
sc_atomic_begin	Function pointer	Function for beginning an atomic section in Sensor Control.
sc_atomic_end	Function pointer	Function for ending an atomic section in Sensor Control.
sc_gpio_intr_1_cb	Function pointer	Function for handling INT1 interrupt.
sc_acc_gyr_mag_fifo_cb	Function pointer	Function for handling indications of multiple new accelerometer, gyroscope and magnetometer sensor data (stored in a sensor module's FIFO).
sc_calibr_acc_offsets_rdy_cb	Function pointer	Function for handling indications of accelerometer sensor calibration completion.
sc_calibr_mag_control_new_cb	Function pointer	Function for handling indications of magnetometer sensor calibration control change.

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
sc_calibr_mag_coeffs_rdy_cb	Function pointer	Function for handling indications of magnetometer sensor calibration coefficients change.
sc_intr_1_recatch	Function pointer	Function for handling indications for still pending (i.e. produced during handling of a previous interrupt) INT1 interrupt.

© 2017 Dialog Semiconductor

8.4.2 Dependency Functions

sc_handle_t -> sc_atomic_begin()

Function Name	`void (*sc_atomic_begin)(void)`
Function Description	(Optional) Begins an atomic section in Sensor Control.
Parameters	None
Return Values	None
Notes	Must be used in combination with `sc_atomic_end()`

sc_handle_t -> sc_atomic_end()

Function Name	`void (*sc_atomic_end)(void)`
Function Description	(Optional) Ends an atomic section in Sensor Control.
Parameters	None
Return Values	None
Notes	

sc_handle_t -> sc_acc_gyr_mag_fifo_cb()

Function Name	`void (*sc_acc_gyr_mag_fifo_cb)(SC_INERTIAL_MULT_DATA_T* sc_data, uint16_t size)`
Function Description	Handles indications of multiple new accelerometer, gyroscope and magnetometer sensor data (stored in a sensor module's FIFO). `SC_INERTIAL_MULT_DATA_T` defines the structure for the combination of types of sensor data (frames) being indicated.
Parameters	`sc_data` the combined multiple samples (frames) of accelerometer, gyroscope and magnetometer sensors in a structure type: `sc_acc_data_t`, `sc_gyr_data_t`, `sc_mag_data_t`, `sc_acc_gyr_data_t`, `sc_acc_mag_data_t`, `sc_gyr_mag_data_t`, `sc_acc_gyr_mag_data_t` `size` the number of samples (frames)
Return Values	None
Notes	

sc_handle_t -> sc_calibr_acc_offsets_rdy_cb()

Function Name	`void (*sc_calibr_acc_offsets_rdy_cb)(sc_calibr_offsets_t* offsets)`
Function Description	Handles indications of accelerometer sensor calibration completion.
Parameters	`offsets` the calibration offsets after accelerometer calibration completion
Return Values	None
Notes	

sc_handle_t -> sc_calibr_mag_control_new_cb()

Function Name	`void (*sc_calibr_mag_control_new_cb)(uint8_t mode, sc_calibr_control_t* control)`
Function Description	Handles indications of magnetometer sensor calibration control change.
Parameters	`mode` the new calibration mode `control` the new calibration control

© 2017 Dialog Semiconductor

Return Values	None
Notes	

sc_handle_t -> sc_calibr_mag_coeffs_rdy_cb()

Function Name	`void (*sc_calibr_mag_coeffs_rdy_cb)(uint8_t mode, sc_calibr_coeffs_t* coeffs)`
Function Description	Handles indications of magnetometer sensor calibration coefficients change.
Parameters	`mode` the new calibration mode `control` the new calibration coefficients
Return Values	None
Notes	

sc_handle_t -> sc_intr_1_recatch()

Function Name	`void (*sc_intr_1_recatch)(void)`
Function Description	Handles indications for still pending (i.e. produced during handling of a previous interrupt) INT1 interrupt.
Parameters	None
Return Values	None
Notes	

8.4.3 Functions

sc_init()

Function Name	`sc_error_t sc_init(sc_handle_t* handle)`
Function Description	Initializes Sensor Control, by setting the appropriate initial values to the internal variables and structures, and initializes sensors.
Parameters	`handle` the handle structure of Sensor Control
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_get_handle()

Function Name	`sc_handle_t* sc_get_handle(void)`
Function Description	Gets handle structure of Sensor Control.
Parameters	None
Return Values	Pointer to the handle structure of Sensor Control.
Notes	

sc_acc_cal_init()

Function Name	`sc_error_t sc_acc_cal_init(void)`
Function Description	Initialize accelerometer sensor calibration.
Parameters	None
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

sc_mag_cal_init()

Function Name	`sc_error_t sc_mag_cal_init(uint8_t mode)`
Function Description	Initializes magnetometer sensor calibration with a specific calibration mode, applying the corresponding coefficients, offsets and control wherever appropriate.

Parameters	`mode` the magnetometer calibration mode
	`SC_MAG_CAL_MODE_NONE,`
	`SC_MAG_CAL_MODE_STATIC,`
	`SC_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC,`
	`SC_MAG_CAL_MODE_CONT_AUTO_BASIC,`
	`SC_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART,`
	`SC_MAG_CAL_MODE_CONT_AUTO_SMART`

Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_mag_cal_stop_auto_calibration()

Function Name	`sc_error_t sc_mag_cal_stop_auto_calibration(void)`
Function Description	Stops magnetometer sensor auto calibration and notify control and coefficients change through `sc_calibr_mag_control_new_cb()` and `sc_calibr_mag_coeffs_rdy_cb()` callback functions, respectively, set to `sc_handle_t` instance handle.
Parameters	None
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_mag_cal_isInitialized()

Function Name	`sc_error_t sc_mag_cal_isInitialized(void)`
Function Description	Notifies that magnetometer sensor calibration is finally initialized (after `sc_mag_cal_init()`). Therefore, magnetometer sensor data can be considered valid (if calibration is settled/converged).
Parameters	None
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_acc_cal_set_offsets()

Function Name	`sc_error_t sc_acc_cal_set_offsets(sc_calibr_offsets_t* offsets)`
Function Description	Sets new accelerometer sensor calibration offsets.
Parameters	`offsets` the calibration offsets structure for the accelerometer sensor
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_mag_cal_set_coeffs()

Function Name	`sc_error_t sc_mag_cal_set_coeffs(uint8_t mode, sc_calibr_coeffs_t* coeffs)`
Function Description	Sets new magnetometer sensor calibration coefficients for a specific calibration mode.

© 2017 Dialog Semiconductor

Parameters	`mode` the magnetometer calibration mode `SC_MAG_CAL_MODE_NONE`, `SC_MAG_CAL_MODE_STATIC`, `SC_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC`, `SC_MAG_CAL_MODE_CONT_AUTO_BASIC`, `SC_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART`, `SC_MAG_CAL_MODE_CONT_AUTO_SMART` `coeffs` the calibration coefficients structure for the magnetometer sensor
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_mag_cal_set_control()

Function Name	`sc_error_t sc_mag_cal_set_control(uint8_t mode, sc_calibr_control_t* control)`
Function Description	Sets new magnetometer sensor calibration control for a specific calibration mode.
Parameters	`mode` the magnetometer calibration mode `SC_MAG_CAL_MODE_NONE`, `SC_MAG_CAL_MODE_STATIC`, `SC_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC`, `SC_MAG_CAL_MODE_CONT_AUTO_BASIC`, `SC_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART`, `SC_MAG_CAL_MODE_CONT_AUTO_SMART` `control` the calibration control structure for the magnetometer sensor
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_acc_cal_get_offsets()

Function Name	`sc_error_t sc_acc_cal_get_offsets(sc_calibr_offsets_t* offsets)`
Function Description	Gets current accelerometer sensor calibration offsets.
Parameters	`offsets` the address to store calibration offsets structure for the accelerometer sensor
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_mag_cal_get_coeffs()

Function Name	`sc_error_t sc_mag_cal_get_coeffs(uint8_t mode, sc_calibr_coeffs_t* coeffs)`
Function Description	Gets magnetometer sensor calibration coefficients for a specific calibration mode.
Parameters	`mode` the magnetometer calibration mode `SC_MAG_CAL_MODE_NONE`, `SC_MAG_CAL_MODE_STATIC`, `SC_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC`, `SC_MAG_CAL_MODE_CONT_AUTO_BASIC`, `SC_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART`, `SC_MAG_CAL_MODE_CONT_AUTO_SMART` `coeffs` the address to store calibration coefficients structure for the magnetometer sensor
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

 © 2017 Dialog Semiconductor

sc_mag_cal_get_control()

Function Name	`sc_error_t sc_mag_cal_get_control(uint8_t mode, sc_calibr_control_t* control)`
Function Description	Gets magnetometer sensor calibration control for a specific calibration mode.

Parameters	`mode` the magnetometer calibration mode
	`SC_MAG_CAL_MODE_NONE,`
	`SC_MAG_CAL_MODE_STATIC,`
	`SC_MAG_CAL_MODE_ONE_SHOT_AUTO_BASIC,`
	`SC_MAG_CAL_MODE_CONT_AUTO_BASIC,`
	`SC_MAG_CAL_MODE_ONE_SHOT_AUTO_SMART,`
	`SC_MAG_CAL_MODE_CONT_AUTO_SMART`
	`control` the address to store calibration control structure for the magnetometer sensor

Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_acc_cal_get_state()

Function Name	`sc_error_t sc_acc_cal_get_state(uint8_t* cal_state)`
Function Description	Gets accelerometer sensor calibration state.
Parameters	`cal_state` the value of the accelerometer sensor calibration state (sc_acc_data_t->calibr_state)
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_mag_cal_get_state()

Function Name	`sc_error_t sc_mag_cal_get_state(uint8_t* cal_state)`
Function Description	Gets magnetometer sensor calibration state.
Parameters	`cal_state` the value of the magnetometer sensor calibration state (sc_mag_data_t->calibr_state)
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_sensor_set_state()

Function Name	`sc_error_t sc_sensor_set_state(uint8_t state)`
Function Description	Sets the power state mode of a sensor, i.e. accelerometer, gyroscope, magnetometer, environmental, step sensor (pedometer), optical/heart rate and secondary accelerometer.

Parameters	`state` the power state mode to set for the sensor
	`SC_STATE_ACC_SUSPEND , SC_STATE_ACC_ACTIVE,`
	`SC_STATE_GYR_SUSPEND , SC_STATE_GYR_ACTIVE,`
	`SC_STATE_MAG_SUSPEND , SC_STATE_MAG_ACTIVE,`
	`SC_STATE_STEP_SUSPEND , SC_STATE_STEP_ACTIVE,`
	`SC_STATE_ENV_SUSPEND , SC_STATE_ENV_FORCED,`
	`SC_STATE_OPT_SUSPEND , SC_STATE_OPT_ACTIVE,`
	`SC_STATE_SNCD_ACC_SUSPEND, SC_STATE_SCND_ACC_ACTIVE`

Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

 © 2017 Dialog Semiconductor

sc_acc_config()

Function Name	`sc_error_t sc_acc_config(sc_acc_config_t* config, uint8_t least_ind_rate)`
Function Description	Sets the operation mode and configuration of accelerometer sensor, by setting its power state mode, interrupt-based sensor data indication, and indication data rate, operation data rate and range. `NORMAL` and `LP1` to `LP6`, refer to decreasing power consumption levels.

Parameters	`config` the configuration structure of the sensor
	`least_ind_rate` least indication rate of new (multiple, combined) inertial sensor data samples through interrupt-based notifications
	`SC_IND_RATE_INVALID, SC_IND_RATE_0_78HZ , SC_IND_RATE_1_56HZ,`
	`SC_IND_RATE_3_12HZ , SC_IND_RATE_6_25HZ , SC_IND_RATE_12_5HZ,`
	`SC_IND_RATE_25HZ , SC_IND_RATE_50HZ , SC_IND_RATE_100HZ,`
	`SC_IND_RATE_200HZ , SC_IND_RATE_400HZ , SC_IND_RATE_800HZ,`
	`SC_IND_RATE_1600Z , SC_IND_RATE_3200HZ`

Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_gyr_config()

Function Name	`sc_error_t sc_gyr_config(sc_gyr_config_t* config, uint8_t least_ind_rate)`
Function Description	Sets the operation mode and configuration of gyroscope sensor, by setting its power state mode, interrupt-based sensor data indication, and indication data rate, operation data rate and range. `NORMAL` and `LP1` to `LP6`, refer to decreasing power consumption levels.

Parameters	`config` the configuration structure of the sensor
	`least_ind_rate` least indication rate of new (multiple, combined) inertial sensor data samples through interrupt-based notifications
	`SC_IND_RATE_INVALID, SC_IND_RATE_0_78HZ , SC_IND_RATE_1_56HZ,`
	`SC_IND_RATE_3_12HZ , SC_IND_RATE_6_25HZ , SC_IND_RATE_12_5HZ,`
	`SC_IND_RATE_25HZ , SC_IND_RATE_50HZ , SC_IND_RATE_100HZ,`
	`SC_IND_RATE_200HZ , SC_IND_RATE_400HZ , SC_IND_RATE_800HZ,`
	`SC_IND_RATE_1600Z , SC_IND_RATE_3200HZ`

Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_mag_config()

Function Name	`sc_error_t sc_mag_config(sc_mag_config_t* config, uint8_t least_ind_rate)`
Function Description	Sets the operation mode and configuration of magnetometer sensor, by setting its power state mode, interrupt-based sensor data indication, and indication data rate and operation data rate. `NORMAL` and `LP1` to `LP6`, refer to decreasing power consumption levels.

Parameters	`config` the configuration structure of the sensor
	`least_ind_rate` least indication rate of new (multiple, combined) inertial sensor data samples through interrupt-based notifications
	`SC_IND_RATE_INVALID, SC_IND_RATE_0_78HZ , SC_IND_RATE_1_56HZ,`
	`SC_IND_RATE_3_12HZ , SC_IND_RATE_6_25HZ , SC_IND_RATE_12_5HZ,`
	`SC_IND_RATE_25HZ , SC_IND_RATE_50HZ , SC_IND_RATE_100HZ,`
	`SC_IND_RATE_200HZ , SC_IND_RATE_400HZ , SC_IND_RATE_800HZ,`
	`SC_IND_RATE_1600Z , SC_IND_RATE_3200HZ`

Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

sc_acc_gyr_mag_config()

Function Name	`sc_error_t sc_acc_gyr_mag_config(sc_acc_gyr_mag_config_t* config, uint8_t least_ind_rate)`
Function Description	Sets the operation mode and configuration of accelerometer, gyroscope and magnetometer sensors, by setting their power state mode, interrupt-based sensor data indication, and indication data rates, operation data rates and ranges (if any). `NORMAL` and `LP1` to `LP6`, refer to decreasing power consumption levels, based on the enabled sensors and the applied configuration.

Parameters	`config` the configuration structure of the sensors
	`least_ind_rate` least indication rate of new (multiple, combined) inertial sensor data samples through interrupt-based notifications
	`SC_IND_RATE_INVALID, SC_IND_RATE_0_78HZ , SC_IND_RATE_1_56HZ,`
	`SC_IND_RATE_3_12HZ , SC_IND_RATE_6_25HZ , SC_IND_RATE_12_5HZ,`
	`SC_IND_RATE_25HZ , SC_IND_RATE_50HZ , SC_IND_RATE_100HZ,`
	`SC_IND_RATE_200HZ , SC_IND_RATE_400HZ , SC_IND_RATE_800HZ,`
	`SC_IND_RATE_1600Z , SC_IND_RATE_3200HZ`

Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_temp_config()

Function Name	`sc_error_t sc_temp_config(sc_temp_config_t* config)`
Function Description	Sets the operation mode and configuration of temperature sensor. `NORMAL` and `LOWPOWER`, refer to different power consumption levels when the sensor produces data samples.
Parameters	`config` the configuration structure of the sensor
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_pres_config()

Function Name	`sc_error_t sc_pres_config(sc_pres_config_t* config)`
Function Description	Sets the operation mode and configuration of pressure sensor. `NORMAL` and `LOWPOWER`, refer to different power consumption levels when the sensor produces data samples.
Parameters	`config` the configuration structure of the sensor
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_hum_config()

Function Name	`sc_error_t sc_hum_config(sc_hum_config_t* config)`
Function Description	Sets the operation mode and configuration of humidity sensor. `NORMAL` and `LOWPOWER`, refer to different power consumption levels when the sensor produces data samples.
Parameters	`config` the configuration structure of the sensor
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_step_config()

Function Name	`sc_error_t sc_step_config(sc_step_config_t* config)`

© 2017 Dialog Semiconductor

Function Description	Sets the operation mode and configuration of step sensor (pedometer). Operation mode refers to the sensitivity on counting steps.
Parameters	`config` the configuration structure of the step sensor (pedometer)
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_opt_config()

Function Name	`sc_error_t sc_opt_config(sc_opt_config_t* config)`
Function Description	Sets the operation mode and configuration of optical sensor. `NORMAL` and `LOWPOWER`, refer to different power consumption levels.
Parameters	`config` the configuration structure of the sensor
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_acc_scnd_config()

Function Name	`sc_error_t sc_acc_scnd_config(sc_acc_config_t* config)`
Function Description	Sets the operation mode and configuration of secondary accelerometer sensor. `NORMAL` and `LOWPOWER`, refer to different power consumption levels.
Parameters	`config` the configuration structure of the sensor
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_step_reset()

Function Name	`sc_error_t sc_step_reset(void)`
Function Description	Resets step sensor's (pedometer) number of accumulated steps.
Parameters	None
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_intr_1_enable()

Function Name	`sc_error_t sc_intr_1_enable(bool enable)`
Function Description	Enables/disables INT1 interrupt, which is related to a sensor module's interrupt-driven operation.
Parameters	`enable` the operation status of INT1 interrupt `SC_DISABLE, SC_ENABLE`
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_intr_1_catch()

Function Name	`sc_error_t sc_intr_1_catch(void)`
Function Description	Catches/handles INT1 interrupt, which is related to a sensor module's interrupt-driven operation.
Parameters	None

 © 2017 Dialog Semiconductor

Return Values	SC_SUCCESS for successful execution; ≥ SC_FAIL for error
Notes	

sc_sensors_disable_intrs()

Function Name	sc_error_t sc_sensors_disable_intrs(void)
Function Description	Disables any involved interrupt-based or -related operation in Sensor Control implementation.
Parameters	None
Return Values	SC_SUCCESS for successful execution; ≥ SC_FAIL for error
Notes	

sc_sensors_suspend()

Function Name	sc_error_t sc_sensors_suspend(void)
Function Description	Put all sensors to the lowest power state mode.
Parameters	None
Return Values	SC_SUCCESS for successful execution; ≥ SC_FAIL for error
Notes	

sc_acc_read()

Function Name	sc_error_t sc_acc_read(sc_acc_data_t* acc_data)
Function Description	Reads accelerometer sensor data, along with accelerometer sensor calibration status.
Parameters	acc_data the address to store accelerometer sensor data being read
Return Values	SC_SUCCESS for successful execution; ≥ SC_FAIL for error
Notes	

sc_gyr_read()

Function Name	sc_error_t sc_gyr_read(sc_gyr_data_t* gyr_data)
Function Description	Reads gyroscope sensor data.
Parameters	gyr_data the address to store gyroscope sensor data being read
Return Values	SC_SUCCESS for successful execution; ≥ SC_FAIL for error
Notes	

sc_mag_read()

Function Name	sc_error_t sc_mag_read(sc_mag_data_t* mag_data)
Function Description	Reads magnetometer compensated sensor data, along with magnetometer sensor calibration status.
Parameters	mag_data the address to store magnetometer sensor data being read
Return Values	SC_SUCCESS for successful execution; ≥ SC_FAIL for error
Notes	

sc_temp_read()

Function Name	sc_error_t sc_temp_read(sc_temp_data_t* temp_data)

© 2017 Dialog Semiconductor

Function Description	Reads temperature sensor data.
Parameters	`temp_data` the address to store temperature sensor data being read
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_pres_read()

Function Name	`sc_error_t sc_pres_read(sc_pres_data_t* pres_data)`
Function Description	Reads pressure sensor data.
Parameters	`pres_data` the address to store pressure sensor data being read
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_hum_read()

Function Name	`sc_error_t sc_hum_read(sc_hum_data_t* hum_data)`
Function Description	Reads humidity sensor data.
Parameters	`hum_data` the address to store humidity sensor data being read
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_env_read()

Function Name	`sc_error_t sc_env_read(sc_env_data_t* env_data)`
Function Description	Reads environmental sensor data.
Parameters	`env_data` the address to store environmental sensor data being read
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_step_read()

Function Name	`sc_error_t sc_step_read(sc_step_data_t* step_data)`
Function Description	Reads step sensor (pedometer) data, referring to the number of steps counted since sensors' initialization (i.e. `sc_init()`) or a previously performed 'reset' (i.e. `sc_step_reset()`).
Parameters	`step_data` the address to store step sensor (pedometer) data being read
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_opt_read()

Function Name	`sc_error_t sc_opt_read(sc_opt_data_t* opt_data)`
Function Description	Reads optical sensor data. It is also indicated whether data are valid or not.
Parameters	`opt_data` the address to store optical sensor data being read
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

sc_acc_scnd_read()

Function Name	`sc_error_t sc_acc_scnd_read(sc_acc_data_t* acc_data)`
Function Description	Reads secondary accelerometer sensor data, along with accelerometer sensor calibration status.
Parameters	`acc_data` the address to store accelerometer sensor data being read
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_env_calc_data_update_latency()

Function Name	`sc_error_t sc_env_calc_data_update_latency(uint16_t* latency)`
Function Description	Calculates environmental data update latency, i.e. the time needed in order for new samples to be ready to be read after setting environmental sensors to force power state mode (i.e. `sc_sensor_set_state(SC_STATE_ENV_FORCED)`), based on their configuration (`sc_temp_config()`, `sc_pres_config()`, and `sc_hum_config()`).
Parameters	`latency` the address to store the update latency of environmental sensor data
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_opt_adapt_operation()

Function Name	`sc_error_t sc_opt_adapt_operation(sc_opt_data_t* opt_data, bool* adapt_done)`
Function Description	Adapts (e.g. to the type and state of the skin) the operation of the employed LEDs, based on the acquired light intensity data. It is also indicated whether adaptation has been completed.
Parameters	`opt_data` the acquired optical sensor data `adapt_done` the address to store indication that adaptation is completed
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

sc_time_get_now_ms()

Function Name	`sc_error_t sc_time_get_now_ms(uint32_t* now)`
Function Description	Gets local time (in msec) produced by a sensor module clock.
Parameters	`now` the address to store local time
Return Values	`SC_SUCCESS` for successful execution; ≥ `SC_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

8.5 Health Care Services Task

The API provided by the Health Care Services Task includes functions for initializing and controlling its operation, configuring the supported/integrated services and the 'input data pushing' mechanism, and finally acquiring the services' configuration state and data. All these functions and respective data structures/types being defined in the API are summarized below.

Table 29: Health Care Services Task API

Data Structures and Types	
hc_svcs_error_t	hc_svcs_upf_config_t
hc_svcs_sf_config_t	hc_svcs_hr_config_t
hc_svcs_sq_config_t	hc_svcs_cc_config_t
hc_svcs_dpsh_src_config_t	hc_svcs_upf_t
hc_svcs_sf_t	hc_svcs_hr_t
hc_svcs_sq_t	hc_svcs_cc_t
hc_svcs_dpsh_src_t	hc_svcs_data_ref_released_cb_t
hc_svcs_handle_t	hc_svcs_data_sf_t
hc_svcs_data_hr_t	hc_svcs_data_sq_t
hc_svcs_data_cc_t	hc_svcs_data_acc_t
hc_svcs_data_gyr_t	hc_svcs_data_mag_t
hc_svcs_data_step_t	hc_svcs_data_pres_t
hc_svcs_data_opt_t	hc_svcs_data_acc_gyr_mag_t
hc_svcs_data_opt_acc_t	hc_svcs_data_step_pres_t
Dependency Functions	
hc_svcs_handle_t -> hc_svcs_clear_done()	hc_svcs_handle_t -> hc_svcs_start_done()
hc_svcs_handle_t -> hc_svcs_stop_done()	hc_svcs_handle_t -> hc_svcs_config_done()
hc_svcs_handle_t -> hc_svcs_state_get_done()	hc_svcs_handle_t -> hc_svcs_data_read_done()
hc_svcs_handle_t -> hc_svcs_data_indication()	hc_svcs_handle_t -> hc_svcs_data_push_done()
hc_svcs_handle_t -> hc_svcs_error()	hc_svcs_handle_t -> hc_svcs_get_rtc_notifs_config()
hc_svcs_handle_t -> hc_svcs_dpsh_src_config()	
Initialization and Control Functions	
hc_svcs_init()	hc_svcs_get_handle()
hc_svcs_clear()	hc_svcs_start()
hc_svcs_stop()	hc_svcs_rtc_send_notif()
Configuration Functions	
hc_svcs_config_upf()	hc_svcs_config_sf()
hc_svcs_config_hr()	hc_svcs_config_sq()
hc_svcs_config_cc()	
State Functions	
hc_svcs_isrunning()	hc_svcs_state_get_upf()
hc_svcs_state_get_sf()	hc_svcs_state_get_hr()
hc_svcs_state_get_sq()	hc_svcs_state_get_cc()
Data Acquisition Functions	
hc_svcs_data_read_sf()	hc_svcs_data_read_hr()
hc_svcs_data_read_sq()	hc_svcs_data_read_cc()

© 2017 Dialog Semiconductor

Data-Push Functions	
hc_svcs_data_push_acc_gyr_mag()	hc_svcs_data_push_acc_gyr_mag_no_copy()
hc_svcs_data_push_opt_acc()	hc_svcs_data_push_opt_acc_no_copy()
hc_svcs_data_push_step_pres()	hc_svcs_data_push_step_pres_no_copy()

8.5.1 Data Structures and Types

hc_svcs_error_t

Type Definition Name	Description
hc_svcs_error_t	Execution status of Health Care Services functions: HC_SVCS_SUCCESS, HC_SVCS_FAIL

hc_svcs_upf_config_t

Data Structure Fields	Type	Description
gender	uint8_t	User's gender type (m/f): HC_SVCS_UPF_GENDER_TYPE_FEMALE, HC_SVCS_UPF_GENDER_TYPE_MALE
age	uint8_t	User's age (in years).
height	uint8_t	User's height (in cm).
weight	uint8_t	User's weight (in kg).
mv_source	uint8_t	User's wrist as source of movement (right/left): HC_SVCS_UPF_MV_SOURCE_LEFT_WRIST, HC_SVCS_UPF_MV_SOURCE_RIGHT_WRIST

hc_svcs_sf_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable sensor fusion service: HC_SVCS_DISABLE, HC_SVCS_ENABLE
mode	uint8_t	Sensor combination mode: HC_SVCS_SF_MODE_GYR, HC_SVCS_SF_MODE_ACC_GYR, HC_SVCS_SF_MODE_ACC_MAG, HC_SVCS_SF_MODE_ACC_GYR_MAG
sf_rate	uint8_t	Sensor fusion service data rate: HC_SVCS_SF_DATA_RATE_INVALID, HC_SVCS_SF_DATA_RATE_0_78HZ, HC_SVCS_SF_DATA_RATE_1_56HZ, HC_SVCS_SF_DATA_RATE_3_12HZ, HC_SVCS_SF_DATA_RATE_6_25HZ, HC_SVCS_SF_DATA_RATE_12_5HZ, HC_SVCS_SF_DATA_RATE_25HZ, HC_SVCS_SF_DATA_RATE_50HZ
acc_range	uint8_t	Accelerometer sensor data range: HC_SVCS_ACCEL_RANGE_2G, HC_SVCS_ACCEL_RANGE_4G, HC_SVCS_ACCEL_RANGE_8G, HC_SVCS_ACCEL_RANGE_16G

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
acc_rate	uint8_t	Accelerometer sensor data rate: HC_SVCS_ACCEL_DATA_RATE_INVALID, HC_SVCS_ACCEL_DATA_RATE_0_78HZ, HC_SVCS_ACCEL_DATA_RATE_1_56HZ, HC_SVCS_ACCEL_DATA_RATE_3_12HZ, HC_SVCS_ACCEL_DATA_RATE_6_25HZ, HC_SVCS_ACCEL_DATA_RATE_12_5HZ, HC_SVCS_ACCEL_DATA_RATE_25HZ, HC_SVCS_ACCEL_DATA_RATE_50HZ, HC_SVCS_ACCEL_DATA_RATE_100HZ
gyr_range	uint8_t	Gyroscope sensor data range: HC_SVCS_GYRO_RANGE_2000_DEG_SEC, HC_SVCS_GYRO_RANGE_1000_DEG_SEC, HC_SVCS_GYRO_RANGE_500_DEG_SEC, HC_SVCS_GYRO_RANGE_250_DEG_SEC, HC_SVCS_GYRO_RANGE_125_DEG_SEC
gyr_rate	uint8_t	Gyroscope sensor data rate: HC_SVCS_GYRO_DATA_RATE_INVALID, HC_SVCS_GYRO_DATA_RATE_25HZ, HC_SVCS_GYRO_DATA_RATE_50HZ, HC_SVCS_GYRO_DATA_RATE_100HZ
mag_rate	uint8_t	Magnetometer sensor data rate: HC_SVCS_MAG_DATA_RATE_INVALID, HC_SVCS_MAG_DATA_RATE_0_78HZ, HC_SVCS_MAG_DATA_RATE_1_56HZ, HC_SVCS_MAG_DATA_RATE_3_12HZ, HC_SVCS_MAG_DATA_RATE_6_25HZ, HC_SVCS_MAG_DATA_RATE_12_5HZ, HC_SVCS_MAG_DATA_RATE_25HZ, HC_SVCS_MAG_DATA_RATE_50HZ, HC_SVCS_MAG_DATA_RATE_100HZ
beta_a	uint16_t	Control value for relative weight of accelerometer sensor data.
beta_m	uint16_t	Control value for relative weight of magnetometer sensor data.

hc_svcs_hr_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable heart rate estimation service: HC_SVCS_DISABLE, HC_SVCS_ENABLE

hc_svcs_sq_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable sleep quality monitoring service: HC_SVCS_DISABLE, HC_SVCS_ENABLE
acc_rate	uint8_t	Accelerometer sensor data rate at which sleep quality monitoring operates: HC_SVCS_ACCEL_DATA_RATE_25HZ

© 2017 Dialog Semiconductor

hc_svcs_cc_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable calories counting service: HC_SVCS_DISABLE, HC_SVCS_ENABLE
mode	uint8_t	Calories counting mode: HC_SVCS_CC_MODE_REGULAR, HC_SVCS_CC_MODE_ROBUST
epoch_interval	uint32_t	Calories counting service data rate: HC_SVCS_CC_EPOCH_5MIN, HC_SVCS_CC_EPOCH_4MIN, HC_SVCS_CC_EPOCH_3MIN, HC_SVCS_CC_EPOCH_2MIN, HC_SVCS_CC_EPOCH_1MIN, HC_SVCS_CC_EPOCH_50SEC, HC_SVCS_CC_EPOCH_40SEC, HC_SVCS_CC_EPOCH_30SEC, HC_SVCS_CC_EPOCH_20SEC, HC_SVCS_CC_EPOCH_10SEC, HC_SVCS_CC_EPOCH_INVALID

hc_svcs_dpsh_src_config_t

Data Structure Fields	Type	Description
acc_config. enable	bool	Enable/disable accelerometer sensor: HC_SVCS_DISABLE, HC_SVCS_ENABLE
acc_config. range	uint8_t	Accelerometer sensor data range: HC_SVCS_ACCEL_RANGE_2G, HC_SVCS_ACCEL_RANGE_4G, HC_SVCS_ACCEL_RANGE_8G, HC_SVCS_ACCEL_RANGE_16G
acc_config. rate	uint8_t	Accelerometer sensor data rate: HC_SVCS_ACCEL_DATA_RATE_INVALID, HC_SVCS_ACCEL_DATA_RATE_0_78HZ, HC_SVCS_ACCEL_DATA_RATE_1_56HZ, HC_SVCS_ACCEL_DATA_RATE_3_12HZ, HC_SVCS_ACCEL_DATA_RATE_6_25HZ, HC_SVCS_ACCEL_DATA_RATE_12_5HZ, HC_SVCS_ACCEL_DATA_RATE_25HZ, HC_SVCS_ACCEL_DATA_RATE_50HZ, HC_SVCS_ACCEL_DATA_RATE_100HZ
gyr_config. enable	bool	Enable/disable gyroscope sensor: HC_SVCS_DISABLE, HC_SVCS_ENABLE
gyr_config. range	uint8_t	Gyroscope sensor data range: HC_SVCS_GYRO_RANGE_2000_DEG_SEC, HC_SVCS_GYRO_RANGE_1000_DEG_SEC, HC_SVCS_GYRO_RANGE_500_DEG_SEC, HC_SVCS_GYRO_RANGE_250_DEG_SEC, HC_SVCS_GYRO_RANGE_125_DEG_SEC
gyr_config. rate	uint8_t	Gyroscope sensor data rate: HC_SVCS_GYRO_DATA_RATE_INVALID, HC_SVCS_GYRO_DATA_RATE_25HZ, HC_SVCS_GYRO_DATA_RATE_50HZ, HC_SVCS_GYRO_DATA_RATE_100HZ

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
mag_config. enable	bool	Enable/disable magnetometer sensor: HC_SVCS_DISABLE, HC_SVCS_ENABLE
mag_config. rate	uint8_t	Magnetometer sensor data rate: HC_SVCS_MAG_DATA_RATE_INVALID, HC_SVCS_MAG_DATA_RATE_0_78HZ, HC_SVCS_MAG_DATA_RATE_1_56HZ, HC_SVCS_MAG_DATA_RATE_3_12HZ, HC_SVCS_MAG_DATA_RATE_6_25HZ, HC_SVCS_MAG_DATA_RATE_12_5HZ, HC_SVCS_MAG_DATA_RATE_25HZ, HC_SVCS_MAG_DATA_RATE_50HZ, HC_SVCS_MAG_DATA_RATE_100HZ
step_config. enable	bool	Enable/disable step counting service: HC_SVCS_DISABLE, HC_SVCS_ENABLE
step_config. rate	uint32_t	Step counting service data rate: HC_SVCS_STEP_DATA_RATE_0_125HZ, HC_SVCS_STEP_DATA_RATE_0_25HZ, HC_SVCS_STEP_DATA_RATE_0_5HZ, HC_SVCS_STEP_DATA_RATE_1HZ, HC_SVCS_STEP_DATA_RATE_2HZ, HC_SVCS_STEP_DATA_RATE_4HZ, HC_SVCS_STEP_DATA_RATE_8HZ, HC_SVCS_STEP_DATA_RATE_DISABLE
pres_config. enable	bool	Enable/disable pressure sensor: HC_SVCS_DISABLE, HC_SVCS_ENABLE
pres_config. rate	uint32_t	Pressure sensor data rate: HC_SVCS_PRES_DATA_RATE_0_125HZ, HC_SVCS_PRES_DATA_RATE_0_25HZ, HC_SVCS_PRES_DATA_RATE_0_5HZ, HC_SVCS_PRES_DATA_RATE_1HZ, HC_SVCS_PRES_DATA_RATE_2HZ, HC_SVCS_PRES_DATA_RATE_4HZ, HC_SVCS_PRES_DATA_RATE_8HZ, HC_SVCS_PRES_DATA_RATE_DISABLE
hc_opt_config. enable	bool	Enable/disable (health care) optical sensor: HC_SVCS_DISABLE, HC_SVCS_ENABLE
hc_opt_config. rate	uint32_t	Optical sensor data rate (in msec).
hc_acc_config. enable	bool	Enable/disable (health care) accelerometer sensor (used for heart rate estimation): HC_SVCS_DISABLE, HC_SVCS_ENABLE
hc_acc_config. range	uint8_t	Accelerometer sensor data range: HC_SVCS_ACCEL_RANGE_2G, HC_SVCS_ACCEL_RANGE_4G, HC_SVCS_ACCEL_RANGE_8G, HC_SVCS_ACCEL_RANGE_16G
hc_acc_config. rate	uint32_t	Accelerometer sensor data rate (in msec).

hc_svcs_upf_t

Type Definition Name	Description
hc_svcs_upf_t	Configuration state structure for user profile attributes: hc_svcs_upf_config_t

© 2017 Dialog Semiconductor

hc_svcs_sf_t

Type Definition Name	Description
hc_svcs_sf_t	Configuration state structure for sensor fusion service: hc_svcs_sf_config_t

hc_svcs_hr_t

Type Definition Name	Description
hc_svcs_hr_t	Configuration state structure for heart rate estimation service: hc_svcs_hr_config_t

hc_svcs_sq_t

Type Definition Name	Description
hc_svcs_sq_t	Configuration state structure for sleep quality monitoring service: hc_svcs_sq_config_t

hc_svcs_cc_t

Type Definition Name	Description
hc_svcs_cc_t	Configuration state structure for calories counting service: hc_svcs_cc_config_t

hc_svcs_dpsh_src_t

Type Definition Name	Description
hc_svcs_dpsh_src_t	Configuration state structure for the source of the pushed (input) data: hc_svcs_dpsh_src_config_t

hc_svcs_data_ref_released_cb_t

Type Definition Name	Description
hc_svcs_data_ref_released_cb_t	Function pointer type for callback functions indicating that a data reference has been processed, and thus released by the “consumer” process.

hc_svcs_handle_t

Data Structure Fields	Type	Description
hc_svcs_clear_done	Function pointer	Function for clear-done indication.
hc_svcs_start_done	Function pointer	Function for start-done indication.
hc_svcs_stop_done	Function pointer	Function for stop-done indication.
hc_svcs_config_done	Function pointer	Function for config-done indication.
hc_svcs_state_get_done	Function pointer	Function for state-get-done indication.
hc_svcs_data_read_done	Function pointer	Function for data-read-done indication.
hc_svcs_data_indication	Function pointer	Function for data indication.
hc_svcs_data_push_done	Function pointer	Function for pushing-data-done indication.
hc_svcs_error	Function pointer	Function for error indication.

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
hc_svcs_get_rtc_notifs_config	Function pointer	Function for acquiring RTC availability and interval.
hc_svcs_dpsh_src_config	Function pointer	Function for requesting new configuration for the source of the pushed (input) data.

hc_svcs_data_sf_t

Data Structure Fields	Type	Description
quaternion_w	int32_t	Sensor fusion service data quartenion w.
quaternion_x	int32_t	Sensor fusion service data quartenion x.
quaternion_y	int32_t	Sensor fusion service data quartenion y.
quaternion_z	int32_t	Sensor fusion service data quartenion z.

hc_svcs_data_hr_t

Data Structure Fields	Type	Description
hr	uint8_t	Heart rate (in bpm).

hc_svcs_data_sq_t

Data Structure Fields	Type	Description
invalid	int32_t	Number of consecutive epochs classified as INVALID.
awake	int32_t	Number of consecutive epochs classified as AWAKE.
light_sleep	int32_t	Number of consecutive epochs classified as LIGHT SLEEP.
deep_sleep	int32_t	Number of consecutive epochs classified as DEEP SLEEP.
rem	int32_t	Number of consecutive epochs classified as REM.
error	int32_t	Number of consecutive epochs classified as ERROR.
epoch_index	uint32_t	First epoch number sleep quality monitoring data refer to.

hc_svcs_data_cc_t

Data Structure Fields	Type	Description
calories	int32_t	Amount of calories burned (in cal).
distance	int32_t	Distance covered (in cm).
num_of_epochs	uint8_t	Number of epochs included.
epoch_index	uint32_t	First epoch number calories counting data refer to.

hc_svcs_data_acc_t

Data Structure Fields	Type	Description
x	int16_t	Accelerometer x-axis data value.

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
y	int16_t	Accelerometer y-axis data value.
z	int16_t	Accelerometer z-axis data value.
calibr_state. reserved0	uint8_t:1	Reserved.
calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
calibr_state. reserved2	uint8_t:6	Reserved.

hc_svcs_data_gyr_t

Data Structure Fields	Type	Description
x	int16_t	Gyroscope x-axis data value.
y	int16_t	Gyroscope y-axis data value.
z	int16_t	Gyroscope z-axis data value.

hc_svcs_data_mag_t

Data Structure Fields	Type	Description
x	int16_t	Magnetometer x-axis data value.
y	int16_t	Magnetometer y-axis data value.
z	int16_t	Magnetometer z-axis data value.
calibr_state. reserved0	uint8_t:1	Reserved.
calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
calibr_state. reserved2	uint8_t:6	Reserved.

hc_svcs_data_step_t

Data Structure Fields	Type	Description
count	uint32_t	Step data value.

hc_svcs_data_pres_t

Data Structure Fields	Type	Description
val	uint32_t	Pressure sensor data value (in Pa).

hc_svcs_data_opt_t

Data Structure Fields	Type	Description
green	uint16_t	Optical sensor Green LED light intensity data value.
ir	uint16_t	Optical sensor IR LED light intensity data value.

hc_svcs_data_acc_gyr_mag_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HC_SVCS_VALID_ACC | HC_SVCS_VALID_GYR | HC_SVCS_VALID_MAG
acc_data	hc_svcs_data_acc_t	Accelerometer sensor xyz-axis data.

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
gyr_data	hc_svcs_data_gyr_t	Gyroscope sensor xyz-axis data.
mag_data	hc_svcs_data_mag_t	Magnetometer sensor xyz-axis data.

hc_svcs_data_opt_acc_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HC_SVCS_VALID_ACC | HC_SVCS_VALID_OPT
opt_data	hc_svcs_data_opt_t	Optical sensor Green and IR LED light intensity data.
acc_data	hc_svcs_data_acc_t	Accelerometer sensor xyz-axis data.

hc_svcs_data_step_pres_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: HC_SVCS_VALID_STEP | HC_SVCS_VALID_PRES
step_data	hc_svcs_data_step_t	Step data.
pres_data	hc_svcs_data_pres_t	Pressure sensor data.

© 2017 Dialog Semiconductor

8.5.2 Dependency Functions

hc_svcs_handle_t -> hc_svcs_clear_done()

Function Name	`void (*hc_svcs_clear_done)(hc_svcs_error_t error)`
Function Description	(Optional) Handles indication that Health Care Services Task has been cleared.
Parameters	`error` the execution status of 'clear' operation `HC_SVCS_SUCCESS, ≥ HC_SVCS_FAIL`
Return Values	None
Notes	

hc_svcs_handle_t -> hc_svcs_start_done()

Function Name	`void (*hc_svcs_start_done)(hc_svcs_error_t error)`
Function Description	(Optional) Handles indication that Health Care Services Task has been started.
Parameters	`error` the execution status of 'start' operation `HC_SVCS_SUCCESS, ≥ HC_SVCS_FAIL`
Return Values	None
Notes	

hc_svcs_handle_t -> hc_svcs_stop_done()

Function Name	`void (*hc_svcs_stop_done)(hc_svcs_error_t error)`
Function Description	(Optional) Handles indication that Health Care Services Task has been stopped.
Parameters	`error` the execution status of 'stop' operation `HC_SVCS_SUCCESS, ≥ HC_SVCS_FAIL`
Return Values	None
Notes	

hc_svcs_handle_t -> hc_svcs_config_done()

Function Name	`void (*hc_svcs_config_done)(uint8_t type, hc_svcs_error_t error)`
Function Description	(Optional) Handles indication that Health Care Services Task has been configured in terms of a specific attributes or service.

Parameters	`type` the configured attributes or service type
	`HC_SVCS_CONFIG_UPF, HC_SVCS_CONFIG_SF, HC_SVCS_CONFIG_HR, HC_SVCS_CONFIG_SQ , HC_SVCS_CONFIG_CC`
	`error` the execution status of 'config' operation
	`HC_SVCS_SUCCESS, ≥ HC_SVCS_FAIL`

Return Values	None
Notes	

hc_svcs_handle_t -> hc_svcs_state_get_done()

Function Name	`void (*hc_svcs_state_get_done)(uint8_t type, void* state, hc_svcs_error_t error)`
Function Description	(Optional) Handles indication that the configuration state of specific attributes or a service in Health Care Services component has been acquired.

© 2017 Dialog Semiconductor

Parameters	`type` the attributes or service type
	`HC_SVCS_STATE_UPF, HC_SVCS_STATE_SF, HC_SVCS_STATE_HR, HC_SVCS_STATE_SQ, HC_SVCS_STATE_CC`
	`state` the configuration state of the attributes or the service
	`error` the execution status of 'state-get' operation
	`HC_SVCS_SUCCESS, ≥ HC_SVCS_FAIL`

Return Values	None
Notes	

hc_svcs_handle_t -> hc_svcs_data_read_done()

Function Name	`void (*hc_svcs_data_read_done)(uint8_t type, void* data, hc_svcs_error_t error)`
Function Description	(Optional) Handles indication that reading data of a specific service in Health Care Services component has been completed.

Parameters	`type` the service type
	`HC_SVCS_SERVICE_TYPE_SF, HC_SVCS_SERVICE_TYPE_HR, HC_SVCS_SERVICE_TYPE_SQ, HC_SVCS_SERVICE_TYPE_CC`
	`data` the acquired service data
	`error` the execution status of 'data-read' operation
	`HC_SVCS_SUCCESS, ≥ HC_SVCS_FAIL`

Return Values	None
Notes	

hc_svcs_handle_t -> hc_svcs_data_indication()

Function Name	`void (*hc_svcs_data_indication)(uint8_t type, void* data, hc_svcs_error_t error)`
Function Description	(Optional) Handles indication that new data of a specific service in Health Care Services component have been produced.

Parameters	`type` the service type
	`HC_SVCS_SERVICE_TYPE_SF, HC_SVCS_SERVICE_TYPE_HR, HC_SVCS_SERVICE_TYPE_SQ, HC_SVCS_SERVICE_TYPE_SQU, HC_SVCS_SERVICE_TYPE_CC`
	`data` the produced/notified service data
	`error` the execution status of new data indication operation
	`HC_SVCS_SUCCESS, ≥ HC_SVCS_FAIL`

Return Values	None
Notes	

hc_svcs_handle_t -> hc_svcs_data_push_done()

Function Name	`void (*hc_svcs_data_push_done)(uint8_t type, hc_svcs_error_t error)`
Function Description	(Optional) Handles indication that the last input data have been pushed to Health Care Services component.

Parameters	`type` the input data type
	`HC_SVCS_DATA_PUSH_ACC_GYR_MAG, HC_SVCS_DATA_PUSH_OPT_ACC, HC_SVCS_DATA_PUSH_STEP_PRES`
	`error` the execution status of data-push operation
	`HC_SVCS_SUCCESS, ≥ HC_SVCS_FAIL`

© 2017 Dialog Semiconductor

Return Values	None
Notes	

hc_svcs_handle_t -> hc_svcs_error()

Function Name	`void (*hc_svcs_error)(uint8_t type, ht_error_t error)`
Function Description	(Optional) Handles indication that an error in Health Care Services component has been produced.

Parameters	`type` the error type
	`HC_SVCS_INIT_ERROR,`
	`HC_SVCS_CONFIG_TYPE_ERROR,`
	`HC_SVCS_STATE_GET_TYPE_ERROR,`
	`HC_SVCS_DATA_READ_SF_ERROR,`
	`HC_SVCS_DATA_READ_HR_ERROR,`
	`HC_SVCS_DATA_READ_SQ_ERROR,`
	`HC_SVCS_DATA_READ_CC_ERROR,`
	`HC_SVCS_OS_TASKNOTIFY_CONTROL_ERROR,`
	`HC_SVCS_OS_TASKNOTIFY_STATE_GET_ERROR,`
	`HC_SVCS_OS_TASKNOTIFY_DATA_PUSH_ACC_GYR_MAG_ERROR,`
	`HC_SVCS_OS_TASKNOTIFY_DATA_PUSH_OPT_ACC_ERROR,`
	`HC_SVCS_OS_TASKNOTIFY_DATA_PUSH_STEP_PRES_ERROR`
	`error` the execution status in Health Care Services component
	`HC_SVCS_SUCCESS, ≥ HC_SVCS_FAIL`

Return Values	None
Notes	

hc_svcs_handle_t -> hc_svcs_get_rtc_notifs_config()

Function Name	`void (*hc_svcs_get_rtc_notifs_config)(bool* enable, uint32_t* interval)`
Function Description	(Optional) Acquires RTC configuration, i.e. availability/status and interval.
Parameters	`enable` the status of RTC operation `HC_SVCS_DISABLE, HC_SVCS_ENABLE` `interval` the (periodic) interval of RTC ticks (in ms)
Return Values	None
Notes	

hc_svcs_handle_t -> hc_svcs_dpsh_src_config()

Function Name	`void (*hc_svcs_dpsh_src_config)(hc_svcs_dpsh_src_config_t* dpsh_config)`
Function Description	(Optional) Requests new configuration for the source of the pushed (input) data.
Parameters	`dpsh_config` the new configuration of the source of the pushed (input) data
Return Values	None
Notes	

8.5.3 Initialization and Control Functions

hc_svcs_init()

Function Name	`hc_svcs_error_t hc_svcs_init(hc_svcs_handle_t* handle)`

© 2017 Dialog Semiconductor

Function Description	Initializes Health Care Services component, setting initial values to the internal variables and structures, creating a FreeRTOS task for control coordination and data manipulation, and initializing services and other processes being supported. Callback function may be set to the `hc_svcs_handle_t` instance handle, so that control and data indications can be sent.
Parameters	`handle` the handle structure of Health Care Services
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	All callback functions defined in `hc_svcs_handle_t` structure are optional.

hc_svcs_get_handle()

Function Name	`hc_svcs_error_t hc_svcs_get_handle(void)`
Function Description	Gets handle structure of Health Care Services.
Parameters	None
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_clear()

Function Name	`hc_svcs_error_t hc_svcs_clear(void)`
Function Description	Clears Health Care Services component, freeing previously allocated resources. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a clear-done indication is sent via `hc_svcs_clear_done()` callback function.
Parameters	None
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_start()

Function Name	`hc_svcs_error_t hc_svcs_start(void)`
Function Description	Starts Health Care Services component, configuring with default parameter values the services being supported. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a start-done indication is sent via `hc_svcs_start_done()` callback function.
Parameters	None
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_stop()

Function Name	`hc_svcs_error_t hc_svcs_stop(void)`
Function Description	Stops Health Care Services component, disabling all services being supported. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a stop-done indication is sent via `hc_svcs_stop_done()` callback function.
Parameters	None
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_rtc_send_notif()

© 2017 Dialog Semiconductor

Function Name	`hc_svcs_error_t hc_svcs_rtc_send_notif(void)`
Function Description	Sends RTC tick notification to Health Care Services component. If RTC interval is not appropriate for Health Care Services operation, RTC tick notification is ignored.
Parameters	None
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

8.5.4 Configuration Functions

hc_svcs_config_upf()

Function Name	`hc_svcs_error_t hc_svcs_config_upf(hc_svcs_upf_config_t* config)`
Function Description	Configures user profile attributes. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a config-done indication is sent via `hc_svcs_config_done()` callback function, setting parameter type to `HC_SVCS_CONFIG_UPF`.
Parameters	`config` the configuration structure of user profile attributes
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_config_sf()

Function Name	`hc_svcs_error_t hc_svcs_config_sf(hc_svcs_sf_config_t* config)`
Function Description	Configures sensor fusion service. If the respective function pointers are defined in `hc_svcs_handle_t` instance handle, a config-done indication is sent via `hc_svcs_config_done()` callback function, setting parameter type to `HC_SVCS_CONFIG_SF`, while data indications via `hc_svcs_data_indication()` are sent based on the configured data rate, setting parameter type to `HC_SVCS_SERVICE_TYPE_SF`.
Parameters	`config` the configuration structure of sensor fusion service
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_config_hr()

Function Name	`hc_svcs_error_t hc_svcs_config_hr(hc_svcs_hr_config_t* config)`
Function Description	Configures heart rate estimation service. If the respective function pointers are defined in `hc_svcs_handle_t` instance handle, a config-done indication is sent via `hc_svcs_config_done()` callback function, setting parameter type to `HC_SVCS_CONFIG_HR`, while data indications via `hc_svcs_data_indication()` are sent based on the configured data rate, setting parameter type to `HC_SVCS_SERVICE_TYPE_HR`.
Parameters	`config` the configuration structure of heart rate estimation service
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_config_sq()

Function Name	`hc_svcs_error_t hc_svcs_config_sq(hc_svcs_sq_config_t* config)`

© 2017 Dialog Semiconductor

Function Description	Configures sleep quality monitoring service. If the respective function pointers are defined in `hc_svcs_handle_t` instance handle, a config-done indication is sent via `hc_svcs_config_done()` callback function, setting parameter type to `HC_SVCS_CONFIG_SQ`, while data indications via `hc_svcs_data_indication()` are sent based on the configured data rate, setting parameter type to `HC_SVCS_SERVICE_TYPE_SQ`.
Parameters	`config` the configuration structure of sleep quality monitoring service
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_config_cc()

Function Name	`hc_svcs_error_t hc_svcs_config_cc(hc_svcs_cc_config_t* config)`
Function Description	Configures calories counting service. If the respective function pointers are defined in `hc_svcs_handle_t` instance handle, a config-done indication is sent via `hc_svcs_config_done()` callback function, setting parameter type to `HC_SVCS_CONFIG_CC`, while data indications via `hc_svcs_data_indication()` are sent based on the configured data rate, setting parameter type to `HC_SVCS_SERVICE_TYPE_CC`.
Parameters	`config` the configuration structure of calories counting service
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

8.5.5 State Functions

hc_svcs_isrunning()

Function Name	`bool hc_svcs_isrunning(void)`
Function Description	Checks if Health Care Service task is running.
Parameters	None
Return Values	`true` if running; false if not running
Notes	

hc_svcs_state_get_upf()

Function Name	`hc_svcs_error_t hc_svcs_state_get_upf(void)`
Function Description	Gets configuration status of user profile attributes. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a state-get-done indication is sent via `hc_svcs_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HC_SVCS_STATE_UPF`.
Parameters	None
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_state_get_sf()

Function Name	`hc_svcs_error_t hc_svcs_state_get_sf(void)`
Function Description	Gets configuration status of sensor fusion service. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a state-get-done indication is sent via `hc_svcs_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `HC_SVCS_STATE_SF`.

© 2017 Dialog Semiconductor

Parameters	None
Return Values	HC_SVCS_SUCCESS for successful execution; ≥ HC_SVCS_FAIL for error
Notes	

hc_svcs_state_get_hr()

Function Name	hc_svcs_error_t hc_svcs_state_get_hr(void)
Function Description	Gets configuration status of heart rate estimation service. If the respective function pointer is defined in hc_svcs_handle_t instance handle, a state-get-done indication is sent via hc_svcs_state_get_done() callback function, carrying the configuration status structure and setting parameter type to HC_SVCS_STATE_HR.
Parameters	None
Return Values	HC_SVCS_SUCCESS for successful execution; ≥ HC_SVCS_FAIL for error
Notes	

hc_svcs_state_get_sq()

Function Name	hc_svcs_error_t hc_svcs_state_get_sq(void)
Function Description	Gets configuration status of sleep quality monitoring service. If the respective function pointer is defined in hc_svcs_handle_t instance handle, a state-get-done indication is sent via hc_svcs_state_get_done() callback function, carrying the configuration status structure and setting parameter type to HC_SVCS_STATE_SQ.
Parameters	None
Return Values	HC_SVCS_SUCCESS for successful execution; ≥ HC_SVCS_FAIL for error
Notes	

hc_svcs_state_get_cc()

Function Name	hc_svcs_error_t hc_svcs_state_get_cc(void)
Function Description	Gets configuration status of calories counting service. If the respective function pointer is defined in hc_svcs_handle_t instance handle, a state-get-done indication is sent via hc_svcs_state_get_done() callback function, carrying the configuration status structure and setting parameter type to HC_SVCS_STATE_CC.
Parameters	None
Return Values	HC_SVCS_SUCCESS for successful execution; ≥ HC_SVCS_FAIL for error
Notes	

8.5.6 Data Acquisition Functions

hc_svcs_data_read_sf()

Function Name	hc_svcs_error_t hc_svcs_data_read_sf(void)
Function Description	Reads last sensor fusion data. If sensor fusion service is enabled and the respective function pointer is defined in hc_svcs_handle_t instance handle, a data-read-done indication is sent via hc_svcs_data_read_done() callback function, carrying last sensor fusion data and setting parameter type to HC_SVCS_SERVICE_TYPE_SF.
Parameters	None
Return Values	HC_SVCS_SUCCESS for successful execution; ≥ HC_SVCS_FAIL for error
Notes	

 © 2017 Dialog Semiconductor

hc_svcs_data_read_hr()

Function Name	`hc_svcs_error_t hc_svcs_data_read_hr(void)`
Function Description	Reads last heart rate estimation data. If heart rate estimation service is enabled and the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a data-read-done indication is sent via `hc_svcs_data_read_done()` callback function, carrying last heart rate estimation data and setting parameter type to `HC_SVCS_SERVICE_TYPE_HR`.
Parameters	None
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_data_read_sq()

Function Name	`hc_svcs_error_t hc_svcs_data_read_sq(void)`
Function Description	Reads last sleep quality monitoring data. If sleep quality monitoring service is enabled and the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a data-read-done indication is sent via `hc_svcs_data_read_done()` callback function, carrying last sleep quality monitoring data and setting parameter type to `HC_SVCS_SERVICE_TYPE_SQ`.
Parameters	None
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_data_read_cc()

Function Name	`hc_svcs_error_t hc_svcs_data_read_cc(void)`
Function Description	Reads last calories counting data. If calories counting service is enabled and the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a data-read-done indication is sent via `hc_svcs_data_read_done()` callback function, carrying last calories counting data and setting parameter type to `HC_SVCS_SERVICE_TYPE_CC`.
Parameters	None
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

8.5.7 Data Push Functions

hc_svcs_data_push_acc_gyr_mag()

Function Name	`hc_svcs_error_t hc_svcs_data_push_acc_gyr_mag(` `hc_svcs_data_acc_gyr_mag_t* data, uint16_t size)`
Function Description	Pushes multiple combined accelerometer, gyroscope and magnetometer sensor data. Data are copied internally. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a data-push-done indication is sent via `hc_svcs_data_push_done()` callback function, setting parameter type to `HC_SVCS_DATA_PUSH_ACC_GYR_MAG`.
Parameters	`data` the multiple combined acc/gyr/mag sensor data samples `size` the number of samples
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

hc_svcs_data_push_acc_gyr_mag_no_copy()

Function Name	`hc_svcs_error_t hc_svcs_data_push_acc_gyr_mag_no_copy(`
	`hc_svcs_data_acc_gyr_mag_t* data, uint16_t size,`
	`hc_svcs_data_ref_released_cb_t cb)`

Function Description	Pushes multiple combined accelerometer, gyroscope and magnetometer sensor data with zero copy. The provided callback function is called upon the release of the data reference by Health Care Services component. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a data-push-done indication is sent via `hc_svcs_data_push_done()` callback function, setting parameter type to `HC_SVCS_DATA_PUSH_ACC_GYR_MAG`.
Parameters	`data` the multiple combined acc/gyr/mag sensor data samples `size` the number of samples `cb` data-reference-released fuction pointer
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_data_push_opt_acc()

Function Name	`hc_svcs_error_t hc_svcs_data_push_opt_acc(` `hc_svcs_data_opt_acc_t* data, uint16_t size)`
Function Description	Pushes multiple combined optical sensor (Green and IR) LED light intensity and accelerometer sensor data. Data are copied internally. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a data-push-done indication is sent via `hc_svcs_data_push_done()` callback function, setting parameter type to `HC_SVCS_DATA_PUSH_OPT_ACC`.
Parameters	`data` the multiple combined optical sensor and accelerometer sensor data samples `size` the number of samples
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_data_push_opt_acc_no_copy()

Function Name	`hc_svcs_error_t hc_svcs_data_push_opt_acc_no_copy(`
	`hc_svcs_data_opt_acc_t* data, uint16_t size,`
	`hc_svcs_data_ref_released_cb_t cb)`

Function Description	Pushes multiple combined optical sensor (Green and IR) LED light intensity and accelerometer sensor data with zero copy. The provided callback function is called upon the release of the data reference by Health Care Services component. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a data-push-done indication is sent via `hc_svcs_data_push_done()` callback function, setting parameter type to `HC_SVCS_DATA_PUSH_OPT_ACC`.
Parameters	`data` the multiple combined optical sensor and accelerometer sensor data samples `size` the number of samples `cb` data-reference-released function pointer
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_data_push_step_pres()

© 2017 Dialog Semiconductor

Function Name	`hc_svcs_error_t hc_svcs_data_push_step_pres(` `hc_svcs_data_step_pres_t* data, uint16_t size)`
Function Description	Pushes multiple combined step and pressure sensor data. Data are copied internally. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a data-push-done indication is sent via `hc_svcs_data_push_done()` callback function, setting parameter type to `HC_SVCS_DATA_PUSH_STEP_PRES`.
Parameters	`data` the multiple combined step and pressure sensor data samples `size` the number of samples
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

hc_svcs_data_push_step_pres_no_copy()

Function Name	`hc_svcs_error_t hc_svcs_data_push_step_pres_no_copy(`
	`hc_svcs_data_step_pres_t* data, uint16_t size,`
	`hc_svcs_data_ref_released_cb_t cb)`

Function Description	Pushes multiple ccombined step and pressure sensor data with zero copy. The provided callback function is called upon the release of the data reference by Health Care Services component. If the respective function pointer is defined in `hc_svcs_handle_t` instance handle, a data-push-done indication is sent via `hc_svcs_data_push_done()` callback function, setting parameter type to `HC_SVCS_DATA_PUSH_STEP_PRES`.
Parameters	`data` the multiple combined step and pressure sensor data samples `size` the number of samples `cb` data-reference-released function pointer
Return Values	`HC_SVCS_SUCCESS` for successful execution; ≥ `HC_SVCS_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

8.6 KIWI Services Task

The KIWI Services Task follows the Health Care Services Task code structure and operation paradigm for the integration in the Wearable DK system of the following services, which are supported by the KIWI basic activity state tracking library:

- Step Counting
- Calories Counting
- Sleep Quality Estimation
- Body State/Activity Classification

As a services provision component, using its API functions the KIWI Services Task can be:

- Initialized, cleared, started, stopped
- Configured in terms of:
  - Common attributes affecting the operation of the supported services: required/pushed accelerometer data range and rate, gyroscope data range and rate (for future use), and magnetometer rate (for future use), using the function `kiwi_svcs_config_common()`.
  - User profile attributes related to anthropometric variables: gender, age, height, weight, and source of movement (right or left wrist), using the function `kiwi_svcs_config_upf()`.
  - Which service (step counting, sleep quality etc.) is enabled or disabled and at which epoch notification interval (if any), using the `kiwi_svcs_conifg_[sc/sq/cc/bs]()` set of functions.

The API provided by the KIWI Services Task includes functions for initializing and controlling its operation, configuring the supported/integrated services and the ‘input data pushing’ mechanism, and finally acquiring the services' configuration state and last data. All these functions and respective data structures/types being defined in the API are summarized below.

Table 30: KIWI Services Task API

Data Structures and Types	
kiwi_svcs_error_t	kiwi_svcs_common_config_t
kiwi_svcs_upf_config_t	kiwi_svcs_sc_config_t
kiwi_svcs_sq_config_t	kiwi_svcs_cc_config_t
kiwi_svcs_bs_config_t	kiwi_svcs_dpsh_src_config_t
kiwi_svcs_common_t	kiwi_svcs_upf_t
kiwi_svcs_sc_t	kiwi_svcs_sq_t
kiwi_svcs_cc_t	kiwi_svcs_bs_t
kiwi_svcs_dpsh_src_t	kiwi_svcs_data_ref_released_cb_t
kiwi_svcs_handle_t	kiwi_svcs_data_sc_t
kiwi_svcs_data_sq_t	kiwi_svcs_data_cc_t
kiwi_svcs_data_bs_t	kiwi_svcs_data_acc_t
kiwi_svcs_data_gyr_t	kiwi_svcs_data_mag_t
kiwi_svcs_data_acc_gyr_mag_t	
Dependency Functions	
kiwi_svcs_handle_t -> kiwi_svcs_clear_done()	kiwi_svcs_handle_t -> kiwi_svcs_start_done()
kiwi_svcs_handle_t -> kiwi_svcs_stop_done()	kiwi_svcs_handle_t -> kiwi_svcs_config_done()
kiwi_svcs_handle_t -> kiwi_svcs_state_get_done()	kiwi_svcs_handle_t -> kiwi_svcs_data_read_done()
kiwi_svcs_handle_t -> kiwi_svcs_data_reset_done()	kiwi_svcs_handle_t -> kiwi_svcs_data_indication()
kiwi_svcs_handle_t -> kiwi_svcs_data_push_done()	kiwi_svcs_handle_t -> kiwi_svcs_error()
kiwi_svcs_handle_t -> kiwi_svcs_get_rtc_notifs_config()	kiwi_svcs_handle_t -> kiwi_svcs_dpsh_src_config()

© 2017 Dialog Semiconductor

Initialization and Control Functions	
kiwi_svcs_init() kiwi_svcs_clear() kiwi_svcs_stop()	kiwi_svcs_get_handle() kiwi_svcs_start() kiwi_svcs_rtc_send_notif()
Configuration Functions	
kiwi_svcs_config_common() kiwi_svcs_config_sc() kiwi_svcs_config_cc()	kiwi_svcs_config_upf() kiwi_svcs_config_sq() kiwi_svcs_config_bs()
State Functions	
kiwi_svcs_isrunning() kiwi_svcs_state_get_upf() kiwi_svcs_state_get_sq() kiwi_svcs_state_get_bs()	kiwi_svcs_state_get_common() kiwi_svcs_state_get_sc() kiwi_svcs_state_get_cc()
Data Acquisition Functions	
kiwi_svcs_data_read_sc() kiwi_svcs_data_read_cc()	kiwi_svcs_data_read_sq() kiwi_svcs_data_read_bs()
Data Reset Functions	
kiwi_svcs_data_reset_sc()	
Data-Push Functions	
kiwi_svcs_data_push_acc_gyr_mag()	kiwi_svcs_data_push_acc_gyr_mag_no_copy()

8.6.1 Data Structures and Types

kiwi_svcs_error_t

Type Definition Name	Description
kiwi_svcs_error_t	Execution status of a function: KIWI_SVCS_SUCCESS, KIWI_SVCS_FAIL

kiwi_svcs_common_config_t

Data Structure Fields	Type	Description
acc_range	uint8_t	Accelerometer sensor data range: KIWI_SVCS_ACCEL_RANGE_2G, KIWI_SVCS_ACCEL_RANGE_4G, KIWI_SVCS_ACCEL_RANGE_8G, KIWI_SVCS_ACCEL_RANGE_16G
acc_rate	uint8_t	Accelerometer sensor data rate: KIWI_SVCS_ACCEL_DATA_RATE_INVALID, KIWI_SVCS_ACCEL_DATA_RATE_0_78HZ, KIWI_SVCS_ACCEL_DATA_RATE_1_56HZ, KIWI_SVCS_ACCEL_DATA_RATE_3_12HZ, KIWI_SVCS_ACCEL_DATA_RATE_6_25HZ, KIWI_SVCS_ACCEL_DATA_RATE_12_5HZ, KIWI_SVCS_ACCEL_DATA_RATE_25HZ, KIWI_SVCS_ACCEL_DATA_RATE_50HZ, KIWI_SVCS_ACCEL_DATA_RATE_100HZ

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
gyr_range	uint8_t	Gyroscope sensor data range: KIWI_SVCS_GYRO_RANGE_2000_DEG_SEC, KIWI_SVCS_GYRO_RANGE_1000_DEG_SEC, KIWI_SVCS_GYRO_RANGE_500_DEG_SEC, KIWI_SVCS_GYRO_RANGE_250_DEG_SEC, KIWI_SVCS_GYRO_RANGE_125_DEG_SEC
gyr_rate	uint8_t	Gyroscope sensor data rate: KIWI_SVCS_GYRO_DATA_RATE_INVALID, KIWI_SVCS_GYRO_DATA_RATE_25HZ, KIWI_SVCS_GYRO_DATA_RATE_50HZ, KIWI_SVCS_GYRO_DATA_RATE_100HZ
mag_rate	uint8_t	Magnetometer sensor data rate: KIWI_SVCS_MAG_DATA_RATE_INVALID, KIWI_SVCS_MAG_DATA_RATE_0_78HZ, KIWI_SVCS_MAG_DATA_RATE_1_56HZ, KIWI_SVCS_MAG_DATA_RATE_3_12HZ, KIWI_SVCS_MAG_DATA_RATE_6_25HZ, KIWI_SVCS_MAG_DATA_RATE_12_5HZ, KIWI_SVCS_MAG_DATA_RATE_25HZ, KIWI_SVCS_MAG_DATA_RATE_50HZ, KIWI_SVCS_MAG_DATA_RATE_100HZ

kiwi_svcs_upf_config_t

Data Structure Fields	Type	Description
gender	uint8_t	User's gender type (m/f): KIWI_SVCS_UPF_GENDER_TYPE_FEMALE, KIWI_SVCS_UPF_GENDER_TYPE_MALE
age	uint8_t	User's age (in years).
height	uint8_t	User's height (in cm).
weight	uint8_t	User's weight (in kg).
mv_source	uint8_t	User's wrist as source of movement (right/left): KIWI_SVCS_UPF_MV_SOURCE_LEFT_WRIST, KIWI_SVCS_UPF_MV_SOURCE_RIGHT_WRIST

kiwi_svcs_sc_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable step counting service: KIWI_SVCS_DISABLE, KIWI_SVCS_ENABLE
epoch_interval	uint32_t	Step counting service epoch interval: KIWI_SVCS_SC_EPOCH_5MIN, KIWI_SVCS_SC_EPOCH_4MIN, KIWI_SVCS_SC_EPOCH_3MIN, KIWI_SVCS_SC_EPOCH_4MIN, KIWI_SVCS_SC_EPOCH_1MIN, KIWI_SVCS_SC_EPOCH_50SEC, KIWI_SVCS_SC_EPOCH_40SEC, KIWI_SVCS_SC_EPOCH_30SEC, KIWI_SVCS_SC_EPOCH_20SEC, KIWI_SVCS_SC_EPOCH_10SEC, KIWI_SVCS_SC_EPOCH_DISABLE, KIWI_SVCS_SC_EPOCH_INVALID

© 2017 Dialog Semiconductor

kiwi_svcs_sq_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable sleep quality monitoring service: KIWI_SVCS_DISABLE, KIWI_SVCS_ENABLE
epoch_interval	uint32_t	Sleep quality monitoring service epoch interval: KIWI_SVCS_SQ_EPOCH_5MIN, KIWI_SVCS_SQ_EPOCH_4MIN, KIWI_SVCS_SQ_EPOCH_3MIN, KIWI_SVCS_SQ_EPOCH_4MIN, KIWI_SVCS_SQ_EPOCH_1MIN, KIWI_SVCS_SQ_EPOCH_INVALID

kiwi_svcs_cc_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable calories counting service: KIWI_SVCS_DISABLE, KIWI_SVCS_ENABLE
epoch_interval	uint32_t	Calories counting epoch interval: KIWI_SVCS_CC_EPOCH_5MIN, KIWI_SVCS_CC_EPOCH_4MIN, KIWI_SVCS_CC_EPOCH_3MIN, KIWI_SVCS_CC_EPOCH_4MIN, KIWI_SVCS_CC_EPOCH_1MIN, KIWI_SVCS_CC_EPOCH_50SEC, KIWI_SVCS_CC_EPOCH_40SEC, KIWI_SVCS_CC_EPOCH_30SEC, KIWI_SVCS_CC_EPOCH_20SEC, KIWI_SVCS_CC_EPOCH_10SEC, KIWI_SVCS_CC_EPOCH_INVALID

kiwi_svcs_bs_config_t

Data Structure Fields	Type	Description
enable	bool	Enable/disable body state classification service: KIWI_SVCS_DISABLE, KIWI_SVCS_ENABLE

kiwi_svcs_dpsh_src_config_t

Data Structure Fields	Type	Description
acc_config. enable	bool	Enable/disable accelerometer sensor: KIWI_SVCS_DISABLE, KIWI_SVCS_ENABLE
acc_config. range	uint8_t	Accelerometer sensor data range: KIWI_SVCS_ACCEL_RANGE_2G, KIWI_SVCS_ACCEL_RANGE_4G, KIWI_SVCS_ACCEL_RANGE_8G, KIWI_SVCS_ACCEL_RANGE_16G

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
acc_config. rate	uint8_t	Accelerometer sensor data rate: KIWI_SVCS_ACCEL_DATA_RATE_INVALID, KIWI_SVCS_ACCEL_DATA_RATE_0_78HZ, KIWI_SVCS_ACCEL_DATA_RATE_1_56HZ, KIWI_SVCS_ACCEL_DATA_RATE_3_12HZ, KIWI_SVCS_ACCEL_DATA_RATE_6_25HZ, KIWI_SVCS_ACCEL_DATA_RATE_12_5HZ, KIWI_SVCS_ACCEL_DATA_RATE_25HZ, KIWI_SVCS_ACCEL_DATA_RATE_50HZ, KIWI_SVCS_ACCEL_DATA_RATE_100HZ
gyr_config. enable	bool	Enable/disable gyroscope sensor: KIWI_SVCS_DISABLE, KIWI_SVCS_ENABLE
gyr_config. range	uint8_t	Gyroscope sensor data range: KIWI_SVCS_GYRO_RANGE_2000_DEG_SEC, KIWI_SVCS_GYRO_RANGE_1000_DEG_SEC, KIWI_SVCS_GYRO_RANGE_500_DEG_SEC, KIWI_SVCS_GYRO_RANGE_250_DEG_SEC, KIWI_SVCS_GYRO_RANGE_125_DEG_SEC
gyr_config. rate	uint8_t	Gyroscope sensor data rate: KIWI_SVCS_GYRO_DATA_RATE_INVALID, KIWI_SVCS_GYRO_DATA_RATE_25HZ, KIWI_SVCS_GYRO_DATA_RATE_50HZ, KIWI_SVCS_GYRO_DATA_RATE_100HZ
mag_config. enable	bool	Enable/disable magnetometer sensor: KIWI_SVCS_DISABLE, KIWI_SVCS_ENABLE
mag_config. rate	uint8_t	Magnetometer sensor data rate: KIWI_SVCS_MAG_DATA_RATE_INVALID, KIWI_SVCS_MAG_DATA_RATE_0_78HZ, KIWI_SVCS_MAG_DATA_RATE_1_56HZ, KIWI_SVCS_MAG_DATA_RATE_3_12HZ, KIWI_SVCS_MAG_DATA_RATE_6_25HZ, KIWI_SVCS_MAG_DATA_RATE_12_5HZ, KIWI_SVCS_MAG_DATA_RATE_25HZ, KIWI_SVCS_MAG_DATA_RATE_50HZ, KIWI_SVCS_MAG_DATA_RATE_100HZ

kiwi_svcs_common_t

Type Definition Name	Description
kiwi_svcs_common_t	Configuration state structure for common attributes of supported services: kiwi_svcs_common_config_t

kiwi_svcs_upf_t

Type Definition Name	Description
kiwi_svcs_upf_t	Configuration state structure for user profile attributes: kiwi_svcs_upf_config_t

kiwi_svcs_sc_t

Type Definition Name	Description
kiwi_svcs_sc_t	Configuration state structure for step counting service: kiwi_svcs_sc_config_t

© 2017 Dialog Semiconductor

kiwi_svcs_sq_t

Type Definition Name	Description
kiwi_svcs_sq_t	Configuration state structure for sleep quality monitoring service: kiwi_svcs_sq_config_t

kiwi_svcs_cc_t

Type Definition Name	Description
kiwi_svcs_cc_t	Configuration state structure for calories counting service: kiwi_svcs_cc_config_t

kiwi_svcs_bs_t

Type Definition Name	Description
kiwi_svcs_bs_t	Configuration state structure for body state classification service: kiwi_svcs_bs_config_t

kiwi_svcs_dpsh_src_t

Type Definition Name	Description
kiwi_svcs_dpsh_src_t	Configuration state structure for the source of the pushed (input) data: kiwi_svcs_dpsh_src_config_t

kiwi_svcs_data_ref_released_cb_t

Type Definition Name	Description
kiwi_svcs_data_ref_released_cb_t	Function pointer type for callback functions indicating that a data reference has been processed, and thus released by the “consumer” process.

kiwi_svcs_handle_t

Data Structure Fields	Type	Description
kiwi_svcs_clear_done	Function pointer	Function for clear-done indication.
kiwi_svcs_start_done	Function pointer	Function for start-done indication.
kiwi_svcs_stop_done	Function pointer	Function for stop-done indication.
kiwi_svcs_config_done	Function pointer	Function for config-done indication.
kiwi_svcs_state_get_done	Function pointer	Function for state-get-done indication.
kiwi_svcs_data_read_done	Function pointer	Function for data-read-done indication.
kiwi_svcs_data_reset_done	Function pointer	Function for data-reset-done indication.
kiwi_svcs_data_indication	Function pointer	Function for data indication.
kiwi_svcs_data_push_done	Function pointer	Function for pushing-data-done indication.
kiwi_svcs_error	Function pointer	Function for error indication.
kiwi_svcs_get_rtc_notifs_config	Function pointer	Function for acquiring RTC availability and interval.
kiwi_svcs_dpsh_src_config	Function pointer	Function for requesting new configuration for the source of the pushed (input) data.

© 2017 Dialog Semiconductor

kiwi_svcs_data_sc_t

Data Structure Fields	Type	Description
steps	int32_t	Steps counted.
epoch_index	uint32_t	First epoch number step counting data refer to.

kiwi_svcs_data_sq_t

Data Structure Fields	Type	Description
invalid	int32_t	Number of consecutive epochs classified as INVALID.
awake	int32_t	Number of consecutive epochs classified as AWAKE.
light_sleep	int32_t	Number of consecutive epochs classified as LIGHT SLEEP.
deep_sleep	int32_t	Number of consecutive epochs classified as DEEP SLEEP.
epoch_index	uint32_t	First epoch number sleep quality monitoring data refer to.

kiwi_svcs_data_cc_t

Data Structure Fields	Type	Description
calories	int32_t	Amount of calories burned (in cal).
epoch_index	uint32_t	First epoch number calories counting data refer to.

kiwi_svcs_data_bs_t

Data Structure Fields	Type	Description
body_state	uint8_t	Body state classification: KIWI_SVCS_BODY_STATE_OTHER, KIWI_SVCS_BODY_STATE_WALK, KIWI_SVCS_BODY_STATE_RUN, KIWI_SVCS_BODY_STATE_SIT, KIWI_SVCS_BODY_STATE_STAND, KIWI_SVCS_BODY_STATE_LAYING_DOWN

kiwi_svcs_data_acc_t

Data Structure Fields	Type	Description
x	int16_t	Accelerometer x-axis data value.
y	int16_t	Accelerometer y-axis data value.
z	int16_t	Accelerometer z-axis data value.
calibr_state. reserved0	uint8_t:1	Reserved.
calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
calibr_state. reserved2	uint8_t:6	Reserved.

kiwi_svcs_data_gyr_t

Data Structure Fields	Type	Description
x	int16_t	Gyroscope x-axis data value.

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
y	int16_t	Gyroscope y-axis data value.
z	int16_t	Gyroscope z-axis data value.

kiwi_svcs_data_mag_t

Data Structure Fields	Type	Description
x	int16_t	Magnetometer x-axis data value.
y	int16_t	Magnetometer y-axis data value.
z	int16_t	Magnetometer z-axis data value.
calibr_state. reserved0	uint8_t:1	Reserved.
calibr_state. calibr_data_valid	uint8_t:1	Post-calibrated data valid flag.
calibr_state. reserved2	uint8_t:6	Reserved.

kiwi_svcs_data_acc_gyr_mag_t

Data Structure Fields	Type	Description
header	uint8_t	Header indicating sensor data validity/presence: KIWI_SVCS_VALID_ACC | KIWI_SVCS_VALID_GYR | KIWI_SVCS_VALID_MAG
acc_data	kiwi_svcs_data_acc_t	Accelerometer sensor xyz-axis data.
gyr_data	kiwi_svcs_data_gyr_t	Gyroscope sensor xyz-axis data.
mag_data	kiwi_svcs_data_mag_t	Magnetometer sensor xyz-axis data.

© 2017 Dialog Semiconductor

8.6.2 Dependency Functions

kiwi_svcs_handle_t -> kiwi_svcs_clear_done()

Function Name	`void (*kiwi_svcs_clear_done)(kiwi_svcs_error_t error)`
Function Description	(Optional) Handles indication that KIWI Services Task has been cleared.
Parameters	`error` the execution status of 'clear' operation `KIWI_SVCS_SUCCESS, ≥ KIWI_SVCS_FAIL`
Return Values	None
Notes	

kiwi_svcs_handle_t -> kiwi_svcs_start_done()

Function Name	`void (*kiwi_svcs_start_done)(kiwi_svcs_error_t error)`
Function Description	(Optional) Handles indication that KIWI Services Task has been started.
Parameters	`error` the execution status of 'start' operation `KIWI_SVCS_SUCCESS, ≥ KIWI_SVCS_FAIL`
Return Values	None
Notes	

kiwi_svcs_handle_t -> kiwi_svcs_stop_done()

Function Name	`void (*kiwi_svcs_stop_done)(kiwi_svcs_error_t error)`
Function Description	(Optional) Handles indication that KIWI Services Task has been stopped.
Parameters	`error` the execution status of 'stop' operation `KIWI_SVCS_SUCCESS, ≥ KIWI_SVCS_FAIL`
Return Values	None
Notes	

kiwi_svcs_handle_t -> kiwi_svcs_config_done()

Function Name	`void (*kiwi_svcs_config_done)(uint8_t type, kiwi_svcs_error_t error)`
Function Description	(Optional) Handles indication that KIWI Services Task has been configured in terms of specific attributes or a service.

Parameters	`type` the configured attributes or service type
	`KIWI_SVCS_CONFIG_COMMON, KIWI_SVCS_CONFIG_UPF,`
	`KIWI_SVCS_CONFIG_SC , KIWI_SVCS_CONFIG_SQ,`
	`KIWI_SVCS_CONFIG_CC , KIWI_SVCS_CONFIG_BS`
	`error` the execution status of 'config' operation
	`KIWI_SVCS_SUCCESS, ≥ KIWI_SVCS_FAIL`

Return Values	None
Notes	

kiwi_svcs_handle_t -> kiwi_svcs_state_get_done()

Function Name	`void (*kiwi_svcs_state_get_done)(uint8_t type, void* state, kiwi_svcs_error_t error)`
Function Description	(Optional) Handles indication that the configuration state of specific attributes or a service in KIWI Services component has been acquired.

© 2017 Dialog Semiconductor

Parameters	`type` the attributes or service type
	`KIWI_SVCS_STATE_COMMON, KIWI_SVCS_STATE_UPF, KIWI_SVCS_STATE_SC , KIWI_SVCS_STATE_SQ, KIWI_SVCS_STATE_CC , KIWI_SVCS_STATE_BS`
	`state` the configuration state of the attributes or the service
	`error` the execution status of 'state-get' operation
	`KIWI_SVCS_SUCCESS, ≥ KIWI_SVCS_FAIL`

Return Values	None
Notes	

kiwi_svcs_handle_t -> kiwi_svcs_data_read_done()

Function Name	`void (*kiwi_svcs_data_read_done)(uint8_t type, void* data, kiwi_svcs_error_t error)`
Function Description	(Optional) Handles indication that reading data of a specific service in KIWI Services component has been completed.

Parameters	`type` the service type
	`KIWI_SVCS_SERVICE_TYPE_SCT, KIWI_SVCS_SERVICE_TYPE_SQ, KIWI_SVCS_SERVICE_TYPE_CC, KIWI_SVCS_SERVICE_TYPE_BS`
	`data` the acquired service data
	`error` the execution status of 'data-read' operation
	`KIWI_SVCS_SUCCESS, ≥ KIWI_SVCS_FAIL`

Return Values	None
Notes	

kiwi_svcs_handle_t -> kiwi_svcs_data_reset_done()

Function Name	`void (*kiwi_svcs_data_reset_done)(uint8_t type, kiwi_svcs_error_t error)`
Function Description	(Optional) Handles indication that resetting data of a specific service in KIWI Services component has been completed.

Parameters	`type` the service type
	`KIWI_SVCS_SERVICE_TYPE_SC`
	`error` the execution status of 'data-reset' operation
	`KIWI_SVCS_SUCCESS, ≥ KIWI_SVCS_FAIL`

Return Values	None
Notes	

kiwi_svcs_handle_t -> kiwi_svcs_data_indication()

Function Name	`void (*kiwi_svcs_data_indication)(uint8_t type, void* data, kiwi_svcs_error_t error)`
Function Description	(Optional) Handles indication that new data of a specific service in KIWI Services component have been produced.

Parameters	`type` the service type
	`KIWI_SVCS_SERVICE_TYPE_SC, KIWI_SVCS_SERVICE_TYPE_SQ, KIWI_SVCS_SERVICE_TYPE_CC, KIWI_SVCS_SERVICE_TYPE_BS`
	`data` the produced/notified service data
	`error` the execution status of new data indication operation
	`KIWI_SVCS_SUCCESS, ≥ KIWI_SVCS_FAIL`

Return Values	None

© 2017 Dialog Semiconductor

Notes	

kiwi_svcs_handle_t -> kiwi_svcs_data_push_done()

Function Name	`void (*kiwi_svcs_data_push_done)(uint8_t type, kiwi_svcs_error_t error)`
Function Description	(Optional) Handles indication that the last input data have been pushed to KIWI Services component.

Parameters	`type` the input data type
	`KIWI_SVCS_DATA_PUSH_ACC_GYR_MAG`
	`error` the execution status of data-push operation
	`KIWI_SVCS_SUCCESS, ≥ KIWI_SVCS_FAIL`

Return Values	None
Notes	

kiwi_svcs_handle_t -> kiwi_svcs_error()

Function Name	`void (*kiwi_svcs_error)(uint8_t type, kiwi_error_t error)`
Function Description	(Optional) Handles indication that an error in KIWI Services component has been produced.

Parameters	`type` the error type
	`KIWI_SVCS_INIT_ERROR,`
	`KIWI_SVCS_CONFIG_TYPE_ERROR,`
	`KIWI_SVCS_STATE_GET_TYPE_ERROR,`
	`KIWI_SVCS_DATA_READ_SC_ERROR,`
	`KIWI_SVCS_DATA_READ_SQ_ERROR,`
	`KIWI_SVCS_DATA_READ_CC_ERROR,`
	`KIWI_SVCS_DATA_READ_BS_ERROR,`
	`KIWI_SVCS_OS_TASKNOTIFY_CONTROL_ERROR,`
	`KIWI_SVCS_OS_TASKNOTIFY_STATE_GET_ERROR,`
	`KIWI_SVCS_OS_TASKNOTIFY_DATA_PUSH_ACC_GYR_MAG_ERROR`
	`error` the execution status in KIWI Services component
	`KIWI_SVCS_SUCCESS, ≥ KIWI_SVCS_FAIL`

Return Values	None
Notes	

kiwi_svcs_handle_t -> kiwi_svcs_get_rtc_notifs_config()

Function Name	`void (*kiwi_svcs_get_rtc_notifs_config)(bool* enable, uint32_t* interval)`
Function Description	(Optional) Acquires RTC configuration, i.e. availability/status and interval.
Parameters	`enable` the status of RTC operation `KIWI_SVCS_DISABLE, KIWI_SVCS_ENABLE` `interval` the (periodic) interval of RTC ticks (in ms)
Return Values	None
Notes	

kiwi_svcs_handle_t -> kiwi_svcs_dpsh_src_config()

Function Name	`void (*kiwi_svcs_dpsh_src_config)(` `kiwi_svcs_dpsh_src_config_t* dpsh_config)`
Function Description	(Optional) Requests new configuration for the source of the pushed (input) data.
Parameters	`dpsh_config` the new configuration of the source of the pushed (input) data

© 2017 Dialog Semiconductor

Return Values	None
Notes	

8.6.3 Initialization and Control Functions

kiwi_svcs_init()

Function Name	`kiwi_svcs_error_t kiwi_svcs_init(kiwi_svcs_handle_t* handle)`
Function Description	Initializes KIWI Services, setting initial values to the internal variables and structures, creating a FreeRTOS task for control coordination and data manipulation, and initializing services and other processes being supported. Callback functions may be set to the `kiwi_svcs_handle_t` instance handle, so that control and data indications can be sent.
Parameters	`handle` the handle structure of KIWI Services
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	All callback functions defined in `kiwi_svcs_handle_t` structure are optional.

kiwi_svcs_get_handle()

Function Name	`kiwi_svcs_error_t kiwi_svcs_get_handle(void)`
Function Description	Gets handle structure of Kiwi Services.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_clear()

Function Name	`kiwi_svcs_error_t kiwi_svcs_clear(void)`
Function Description	Clears KIWI Services, freeing previously allocated resources. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a clear-done indication is sent via `kiwi_svcs_clear_done()` callback function.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_start()

Function Name	`kiwi_svcs_error_t kiwi_svcs_start(void)`
Function Description	Starts KIWI Services, configuring with default parameter values the services being supported. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a start-done indication is sent via `kiwi_svcs_start_done()` callback function.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_stop()

Function Name	`kiwi_svcs_error_t kiwi_svcs_stop(void)`

© 2017 Dialog Semiconductor

Function Description	Stops KIWI Services, disabling all services being supported. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a stop-done indication is sent via `kiwi_svcs_stop_done()` callback function.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_rtc_send_notif()

Function Name	`kiwi_svcs_error_t kiwi_svcs_rtc_send_notif(void)`
Function Description	Sends RTC tick notification to KIWI Services. If RTC interval is not appropriate for KIWI Services operation, RTC tick notification is ignored.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

8.6.4 Configuration Functions

kiwi_svcs_config_common()

Function Name	`kiwi_svcs_error_t kiwi_svcs_config_common(` `kiwi_svcs_common_config_t* config)`
Function Description	Configure common attributes for supported services. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a config-done indication is sent via `kiwi_svcs_config_done()` callback function, setting parameter type to `KIWI_SVCS_CONFIG_COMMON`.
Parameters	`config` the configuration structure of kiwi services common attributes
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_config_upf()

Function Name	`kiwi_svcs_error_t kiwi_svcs_config_upf(kiwi_svcs_upf_config_t* config)`
Function Description	Configures user profile attributes. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a config-done indication is sent via `kiwi_svcs_config_done()` callback function, setting parameter type to `KIWI_SVCS_CONFIG_UPF`.
Parameters	`config` the configuration structure of user profile attributes
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_config_sc()

Function Name	`kiwi_svcs_error_t kiwi_svcs_config_sc(kiwi_svcs_sc_config_t* config)`
Function Description	Configures step counting service. If the respective function pointers are defined in `kiwi_svcs_handle_t` instance handle, a config-done indication is sent via `kiwi_svcs_config_done()` callback function, setting parameter type to `KIWI_SVCS_CONFIG_SC`, while data indications via `kiwi_svcs_data_indication()` are sent based on the configured data rate (epoch interval), setting parameter type to `KIWI_SVCS_SERVICE_TYPE_SC`.

© 2017 Dialog Semiconductor

Parameters	`config` the configuration structure of step counting service
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_config_sq()

Function Name	`kiwi_svcs_error_t kiwi_svcs_config_sq(kiwi_svcs_sq_config_t* config)`
Function Description	Configures sleep quality monitoring service. If the respective function pointers are defined in `kiwi_svcs_handle_t` instance handle, a config-done indication is sent via `kiwi_svcs_config_done()` callback function, setting parameter type to `KIWI_SVCS_CONFIG_SQ`, while data indications via `kiwi_svcs_data_indication()` are sent based on the configured data rate (epoch interval), setting parameter type to `KIWI_SVCS_SERVICE_TYPE_SQ`.
Parameters	`config` the configuration structure of sleep quality monitoring service
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_config_cc()

Function Name	`kiwi_svcs_error_t kiwi_svcs_config_cc(kiwi_svcs_cc_config_t* config)`
Function Description	Configures calories counting service. If the respective function pointers are defined in `kiwi_svcs_handle_t` instance handle, a config-done indication is sent via `kiwi_svcs_config_done()` callback function, setting parameter type to `KIWI_SVCS_CONFIG_CC`, while data indications via `kiwi_svcs_data_indication()` are sent based on the configured data rate (epoch interval), setting parameter type to `KIWI_SVCS_SERVICE_TYPE_CC`.
Parameters	`config` the configuration structure of calories counting service
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_config_bs()

Function Name	`kiwi_svcs_error_t kiwi_svcs_config_bs(kiwi_svcs_bs_config_t* config)`
Function Description	Configures body state classification service. If the respective function pointers are defined in `kiwi_svcs_handle_t` instance handle, a config-done indication is sent via `kiwi_svcs_config_done()` callback function, setting parameter type to `KIWI_SVCS_CONFIG_BS`, while data indications via `kiwi_svcs_data_indication()` are sent on every change of body state, setting parameter type to `KIWI_SVCS_SERVICE_TYPE_BS`.
Parameters	`config` the configuration structure of body state classification service
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

8.6.5 State Functions

kiwi_svcs_isrunning()

Function Name	`bool kiwi_svcs_isrunning(void)`
Function Description	Checks if KIWI Service task is running.
Parameters	None

© 2017 Dialog Semiconductor

Return Values	`true` if running; false if not running
Notes	

kiwi_svcs_state_get_common()

Function Name	`kiwi_svcs_error_t kiwi_svcs_state_get_common(void)`
Function Description	Gets configuration status of supported services common attributes. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a state-get-done indication is sent via `kiwi_svcs_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `KIWI_SVCS_STATE_COMMON`.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_state_get_upf()

Function Name	`kiwi_svcs_error_t kiwi_svcs_state_get_upf(void)`
Function Description	Gets configuration status of user profile attributes. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a state-get-done indication is sent via `kiwi_svcs_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `KIWI_SVCS_STATE_UPF`.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_state_get_sc()

Function Name	`kiwi_svcs_error_t kiwi_svcs_state_get_sc(void)`
Function Description	Gets configuration status of step counting service. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a state-get-done indication is sent via `kiwi_svcs_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `KIWI_SVCS_STATE_SC`.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_state_get_sq()

Function Name	`kiwi_svcs_error_t kiwi_svcs_state_get_sq(void)`
Function Description	Gets configuration status of sleep quality monitoring service. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a state-get-done indication is sent via `kiwi_svcs_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `KIWI_SVCS_STATE_SQ`.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_state_get_cc()

Function Name	`kiwi_svcs_error_t kiwi_svcs_state_get_cc(void)`

© 2017 Dialog Semiconductor

Function Description	Gets configuration status of calories counting service. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a state-get-done indication is sent via `kiwi_svcs_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `KIWI_SVCS_STATE_CC`.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_state_get_bs()

Function Name	`kiwi_svcs_error_t kiwi_svcs_state_get_cc(void)`
Function Description	Gets configuration status of body state classification service. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a state-get-done indication is sent via `kiwi_svcs_state_get_done()` callback function, carrying the configuration status structure and setting parameter type to `KIWI_SVCS_STATE_BS`.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

8.6.6 Data Acquisition Functions

kiwi_svcs_data_read_sc()

Function Name	`kiwi_svcs_error_t kiwi_svcs_data_read_sc(void)`
Function Description	Reads total step counter. If step counting service is enabled and the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a data-read-done indication is sent via `kiwi_svcs_data_read_done()` callback function, carrying total step counter value and setting parameter type to `KIWI_SVCS_SERVICE_TYPE_SCT`.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_data_read_sq()

Function Name	`kiwi_svcs_error_t kiwi_svcs_data_read_sq(void)`
Function Description	Reads last sleep quality monitoring data. If sleep quality monitoring service is enabled and the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a data-read-done indication is sent via `kiwi_svcs_data_read_done()` callback function, carrying last sleep quality monitoring data and setting parameter type to `KIWI_SVCS_SERVICE_TYPE_SQ`.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_data_read_cc()

Function Name	`kiwi_svcs_error_t kiwi_svcs_data_read_cc(void)`

© 2017 Dialog Semiconductor

Function Description	Reads last calories counting data. If calories counting service is enabled and the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a data-read-done indication is sent via `kiwi_svcs_data_read_done()` callback function, carrying last calories counting data and setting parameter type to `KIWI_SVCS_SERVICE_TYPE_CC`.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_data_read_bs()

Function Name	`kiwi_svcs_error_t kiwi_svcs_data_read_bs(void)`
Function Description	Reads last body state classification data. If body state classification service is enabled and the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a data-read-done indication is sent via `kiwi_svcs_data_read_done()` callback function, carrying last body state classification data and setting parameter type to `KIWI_SVCS_SERVICE_TYPE_BS`.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

8.6.7 Data Reset Functions

kiwi_svcs_data_reset_sc()

Function Name	`kiwi_svcs_error_t kiwi_svcs_data_reset_sc(void)`
Function Description	Resets total step counter value. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a data-reset-done indication is sent via `kiwi_svcs_data_reset_done()` callback function, setting parameter type to `KIWI_SVCS_SERVICE_TYPE_SC`.
Parameters	None
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

8.6.8 Data Push Functions

kiwi_svcs_data_push_acc_gyr_mag()

Function Name	`kiwi_svcs_error_t kiwi_svcs_data_push_acc_gyr_mag(` `kiwi_svcs_data_acc_gyr_mag_t* data, uint16_t size)`
Function Description	Pushes multiple combined accelerometer, gyroscope and magnetometer sensor data. Data are copied internally. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a data-push-done indication is sent via `kiwi_svcs_data_push_done()` callback function, setting parameter type to `KIWI_SVCS_DATA_PUSH_ACC_GYR_MAG`.
Parameters	`data` the multiple combined acc/gyr/mag sensor data samples `size` the number of samples
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

kiwi_svcs_data_push_acc_gyr_mag_no_copy()

© 2017 Dialog Semiconductor

Function Name	`kiwi_svcs_error_t kiwi_svcs_data_push_acc_gyr_mag_no_copy(`
	`kiwi_svcs_data_acc_gyr_mag_t* data, uint16_t size,`
	`kiwi_svcs_data_ref_released_cb_t cb)`

Function Description	Pushes multiple combined accelerometer, gyroscope and magnetometer sensor data with zero copy. The provided callback function is called upon the release of the data reference by Kiwi Services component. If the respective function pointer is defined in `kiwi_svcs_handle_t` instance handle, a data-push-done indication is sent via `kiwi_svcs_data_push_done()` callback function, setting parameter type to `KIWI_SVCS_DATA_PUSH_ACC_GYR_MAG`.
Parameters	`data` the multiple combined acc/gyr/mag sensor data samples `size` the number of samples `cb` data-reference-released function pointer
Return Values	`KIWI_SVCS_SUCCESS` for successful execution; ≥ `KIWI_SVCS_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

9 Health Toolbox Libraries

9.1 Heart Rate Estimation Library

The Heart Rate Estimation Library provides functionality for determining the heartbeat of a person from Green LED light intensity data produced by the optical sensor which is integrated in DI5115 sensor module, as well as from accelerometer data produced by the integrated accelerometer sensor in the DI5115 sensor module for compensating movement artifacts. It provides also functionality for estimating SPO2 (i.e. Oximeter), based on IR LED light intensity data that can also be produced by DI5115's optical sensor. A more detailed description of the library can be found in the user manual of the DA14681 wearable development kit [1].

The API provided by the Heart Rate Estimation Library includes functions for initializing, executing and configuring the heart rate estimation process. All these functions are summarized below.

Table 31: Heart Rate Estimation Library API

Initialization Functions	
HealthCare_Init()	
Main Process Functions	
HealthCare_Run()	
Configuration Functions	
HealthCare_Config()	

© 2017 Dialog Semiconductor

9.1.1 Initialization Functions

HealthCare_Init()

Function Name	`void HealthCare_Init(void)`
Function Description	Initializes HealthCare algorithm and its data.
Parameters	None
Return Values	None
Notes	

9.1.2 Main Process Functions

HealthCare_Run()

Function Name	`void HealthCare_Run(unsigned short* data, short* accele_data, unsigned char* val)`
Function Description	Calculates heartbeat and SPO2, as output of HealthCare algorithm, based on the LED light intensity and accelerometer data provided by DI5115 sensor module.
Parameters	`data` pointer to (`uint16_t`) LED light intensity data, i.e. Green LED (data[0]) and IR LED (data[1]) `accele_data` pointer to (`int16_t`) 8-bit resolution and ±2g range accelerometer xyz-axis data, i.e. x-axis (data[0]), y-axis (data[1]) and z-axis (data[2]) `val` pointer to the returned (uint8_t) health care data, i.e. heartbeat (data[0]) and SPO2 (data[1])
Return Values	None
Notes	

9.1.3 Configuration Functions

HealthCare_Config()

Function Name	`void HealthCare_Config(float HoldTime, unsigned short PersistNum, unsigned char OXM_EN)`
Function Description	Configures HealthCare algorithm operation parameters.
Parameters	`HoldTime` the value of holding time for the produced health care data when no object is detected in the proximity of the optical sensor (in sec) `PersistNum` the number of successive heartbeat samples to be checked, so that a valid heartbeat value can be produced `OXM_EN` the value for the Oximeter enable mode, i.e. disable = 0, enable = 1
Return Values	None
Notes	Default value for `HoldTime` is 5000 ms. Default value for `PersistNum` is 3 heartbeat samples. Default value for `OXM_EN` is 1 (Oximeter enabled).

© 2017 Dialog Semiconductor

9.2 Sleep Quality Monitoring Library

The Sleep Quality Monitoring Library provides functionality for determining the sleep quality of a person based on the physical activity, which is indicated by accelerometer data produced due to wrist movement. The sleep states being tracked are ‘awake’, ‘light sleep’ and ‘deep sleep’’ (‘REM’ sleep state detection is not supported). A more detailed description of the library can be found in the user manual of the DA14681 wearable development kit [1].

The API provided by the Sleep Quality Monitoring Library includes functions for initializing or clearing the allocated resources, providing accelerometer data as input to sleep quality monitoring process, executing the sleep state classification at the end of an epoch, and finally acquiring sleep state data or the state of sleep quality monitoring process. All these functions and respective data structures/types being defined in the API are summarized below.

Table 32: Sleep Quality Monitoring Library API

Data Structures and Types	
sq_error_t sq_handle_t sq_data_sleep_state_t	sq_config_t sq_data_acc_t sq_data_sleep_state_update_t
Dependency Functions	
sq_handle_t -> sq_get_buffer() sq_handle_t -> sq_epoch_completed()	sq_handle_t -> sq_free_buffer() sq_handle_t -> sq_epoch_updated()
Initialization and Basic Configuration Functions	
sq_init()	sq_clear()
Main Process Functions	
sq_data_update_acc()	sq_calc_epoch_state()
Service Data Acquisition Functions	
sq_data_read_sleep_state_till()	sq_data_read_sleep_state()
State Functions	
sq_state_get_sleep_state_epoch_last()	sq_state_get_sleep_state_epochs_num()

9.2.1 Data Structures and Types

sq_error_t

Type Definition Name	Description
sq_error_t	Execution status of a function: SQ_SUCCESS, SQ_FAIL

sq_config_t

Data Structure Fields	Type	Description
sleep_classif_model	uint8_t	Sleep/awake classification model (only one supported in this version): SQ_SLEEP_CLASSIF_MODEL_CUSTOM, SQ_SLEEP_CLASSIF_MODEL_SADEH. SQ_SLEEP_CLASSIF_MODEL_JEAN_LOUIS, SQ_SLEEP_CLASSIF_MODEL_OAKLEY
sleep_onset_time	uint8_t	Time (in min) to pass with low activity, so that sleep/awake classification can be activated, otherwise ‘AWAKE’ state is considered.

© 2017 Dialog Semiconductor

Data Structure Fields	Type	Description
wake_onset_act_time_per_epoch	uint8_t	Total activity duration (in sec) per epoch, so that sleep/awake classification can be deactivated and 'AWAKE' state is considered.
not_attached_onset_time	uint8_t	Time (in min) to pass with no activity, so that 'NOT ATTACHED' state can be considered, that is, no subject is detected.
acc_rate	uint8_t	Accelerometer sensor data rate at which Sleep Quality Monitoring is configured to operate (only 25 Hz is supported in this version): SQ_ACCEL_DATA_RATE_INVALID, SQ_ACCEL_DATA_RATE_25HZ
physical_activity_thres_sleep_onset	uint16_t	Accelerometer sensor data magnitude threshold (in mg) used for tracking sleep-on-set time.
physical_activity_thres_model	uint16_t	Accelerometer sensor data magnitude threshold (in mg) used by the sleep/awake classification model.
movement_index_deep_sleep	uint8_t	Index value defining the level of movement discriminating light from deep sleep.

sq_handle_t

Data Structure Fields	Type	Description
sq_get_buffer	Function pointer	Function for memory space allocation.
sq_free_buffer	Function pointer	Function for memory space deallocation.
sq_epoch_completed	Function pointer	Function for epoch change indication (not supported in this version).
sq_epoch_updated	Function pointer	Function for epoch update indication.

sq_data_acc_t

Data Structure Fields	Type	Description
x	int32_t	Accelerometer x-axis data value (in -2G to +2G range over 16-bit res.: 2/(1<<(16-1)) G).
y	int32_t	Accelerometer y-axis data value (in -2G to +2G range over 16-bit res.: 2/(1<<(16-1)) G).
z	int32_t	Accelerometer z-axis data value (in -2G to +2G range over 16-bit res.: 2/(1<<(16-1)) G).

sq_data_sleep_state_t

Data Structure Fields	Type	Description
state	uint8_t	Sleep state value: SQ_SLEEP_STATE_INVALID, SQ_SLEEP_STATE_AWAKE, SQ_SLEEP_STATE_LIGHT_SLEEP, SQ_SLEEP_STATE_DEEP_SLEEP, SQ_SLEEP_STATE_REM, SQ_SLEEP_STATE_ERROR
epoch_index	uint32_t	The epoch number sleep state classification refers to.

© 2017 Dialog Semiconductor

sq_data_sleep_state_update_t

Data Structure Fields	Type	Description
state	uint8_t	Sleep state value to update epochs with: SQ_SLEEP_STATE_INVALID, SQ_SLEEP_STATE_AWAKE, SQ_SLEEP_STATE_LIGHT_SLEEP, SQ_SLEEP_STATE_DEEP_SLEEP, SQ_SLEEP_STATE_REM, SQ_SLEEP_STATE_ERROR
length	uint8_t	The number of consecutive epochs to be updated.
epoch_index	uint32_t	The epoch number of the first epoch being updated.

9.2.2 Dependency Functions

sq_handle_t -> sq_get_buffer()

Function Name	sq_error_t (*sq_get_buffer)(uint8_t** buf, uint16_t size)
Function Description	Implements memory space allocation for variables and structures used internally in Sleep Quality Monitoring Library.
Parameters	buf the address of the pointer variable to put the address of the allocated memory space size the size of the memory space to be allocated (in bytes)
Return Values	SQ_SUCCESS for successful execution; ≥ SQ_FAIL for error
Notes	

sq_handle_t -> sq_free_buffer()

Function Name	sq_error_t (*sq_free_buffer)(uint8_t* buf)
Function Description	Implements memory space deallocation for variables and structures having been used internally in Sleep Quality Monitoring Library.
Parameters	buf the address of the allocated memory space
Return Values	SQ_SUCCESS for successful execution; ≥ SQ_FAIL for error
Notes	

sq_handle_t -> sq_epoch_completed()

Function Name	sq_error_t (*sq_epoch_completed)(void)
Function Description	(Optional) Handles epoch change indication sent when the number of accelerometer data samples that have been provided as input is such that, based on the given rate (sq_config_t -> acc_rate), an epoch is detected that has been completed (not supported in this version).
Parameters	None
Return Values	SQ_SUCCESS for successful execution; ≥ SQ_FAIL for error
Notes	

© 2017 Dialog Semiconductor

sq_handle_t -> sq_epoch_updated()

Function Name	`sq_epoch_t (*sq_epoch_updated)(sq_data_sleep_state_update_t* update)`
Function Description	(Optional) Handles indication that several successive epochs are to be updated in terms of the classified sleep state from a specific epoch number in tha past and then.
Parameters	`update` the epoch update
Return Values	`SQ_SUCCESS` for successful execution; ≥ `SQ_FAIL` for error
Notes	

9.2.3 Initialization and Basic Configuration Functions

sq_init()

Function Name	`sq_error_t sq_init(sq_handle_t* handle, sq_config_t* config)`
Function Description	Initializes Sleep Quality Monitoring with the appropriate configuration in terms of the employed sleep/awake classification model, activation/deactivation modes, accelerometer sensor input data rate, physical activity thresholds and movement index for discriminating light and deep sleep. Memory space is requested via `sq_get_buffer()` callback function set to `sq_handle_t` instance handle.
Parameters	`handle` the handle structure for Sleep Quality Monitoring `config` the configuration structure for Sleep Quality Monitoring operation
Return Values	`SQ_SUCCESS` for successful execution; ≥ `SQ_FAIL` for error
Notes	`sq_get_buffer()` and `sq_free_buffer()` must be set. `sleep_onset_time` must be greater than 1min. `wake_onset_act_time_per_epoch` must be greater than 1min and less than 56sec. `not_attached_onset_time` must be 0 (disable) or greater than 4min, and grater or equal to `sleep_onset_time`. Only `SQ_ACCEL_DATA_RATE_25HZ` accelerometer input data rate is supported in current version. 'REM' sleep state detection is not supported in current version.

sq_clear()

Function Name	`sq_error_t sq_clear(void)`
Function Description	Clears Sleep Quality Monitoring, freeing previously allocated resources via `sq_gree_buffer()` callback function set to `sq_handle_t` instance handle.
Parameters	None
Return Values	`SQ_SUCCESS` for successful execution; ≥ `SQ_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

9.2.4 Main Process Functions

sq_data_update_acc()

Function Name	`sq_error_t sq_data_update_acc(sq_data_acc_t* data, uint16_t size)`
Function Description	Provides new accelerometer data samples to Sleep Quality Monitoring, based on the data rate with which the latter has been previously configured. Accelerometer data must be converted to (+2g,-2g) range over 16-bit resolution (i.e. 2/(1<<(16-1)) = 0.061 mg).
Parameters	`data` the accelerometer xyz-axis data samples `size` the number of samples
Return Values	`SQ_SUCCESS` for successful execution; ≥ `SQ_FAIL` for error
Notes	

sq_calc_epoch_state()

Function Name	`sq_error_t sq_calc_epoch_state(void)`
Function Description	Executes sleep state classification. This function must be called at the end of an epoch, the time interval of which must be 1 minute. Upon execution an epoch update may be notified through `sq_epoch_updated()` callback function set to `sq_handle_t` instance handle.
Parameters	None
Return Values	`SQ_SUCCESS` for successful execution; ≥ `SQ_FAIL` for error
Notes	

9.2.5 Service Data Acquisition Functions

sq_data_read_sleep_state_till()

Function Name	`sq_error_t sq_data_read_sleep_state_till(sq_data_sleep_state_t* data, uint16_t epoch_last, uint16_t num_of_epochs)`
Function Description	Reads a number of classified epochs till a specific epoch number since Sleep Quality Monitoring initialization (i.e. `sq_init()`).
Parameters	`data` the buffer to store sleep state data being read `epoch_last` the last epoch (the last epoch number) `num_of_epochs` the number of consecutive epochs to read
Return Values	`SQ_SUCCESS` for successful execution; ≥ `SQ_FAIL` for error
Notes	

sq_data_read_sleep_state()

Function Name	`sq_error_t sq_data_read_sleep_state(sq_data_sleep_state_t* data, uint16_t num_of_epochs)`
Function Description	Reads a number of last classified epochs since Sleep Quality Monitoring initialization (i.e. `sq_init()`).
Parameters	`data` the buffer to store sleep state data being read `num_of_epochs` the number of consecutive last epochs to read
Return Values	`SQ_SUCCESS` for successful execution; ≥ `SQ_FAIL` for error
Notes	

 © 2017 Dialog Semiconductor

9.2.6 State Functions

sq_state_get_sleep_state_epoch_last()

Function Name	`sq_error_t sq_state_get_sleep_state_epoch_last(uint32_t* epoch_index)`
Function Description	Get the last epoch number having been classified in terms of a sleep state.
Parameters	`epoch_index` the last classified epoch number
Return Values	`SQ_SUCCESS` for successful execution; ≥ `SQ_FAIL` for error
Notes	

sq_state_get_sleep_state_epochs_num()

Function Name	`sq_error_t sq_state_get_sleep_state_epochs_num(uint16_t* num_of_epochs)`
Function Description	Get the number of epochs stored at the internally maintained sequence of the last epoch classifications.
Parameters	`num_of_epochs` the number of internally maintained last classified epochs
Return Values	`SQ_SUCCESS` for successful execution; ≥ `SQ_FAIL` for error
Notes	

9.3 Calories Counting Library

The Calories Counting Library provides functionality for determining the calories burned by a person based on user profile attributes (i.e. gender, age, height and weight), the steps being performed and the inclination being detected (based on atmospheric pressure data). A more detailed description of the library can be found in the user manual of the DA14681 wearable development kit [1].

The API provided by the Calories Counting Library includes functions for initializing or clearing the allocated resources, updating user profile attributes, providing step and (atmospheric) pressure sensor data as input to calories counting process, executing the calculation of calories burned at the end of an epoch, and finally acquiring calories (and distance covered) data or the state of calories counting process. All these functions and respective data structures/types being defined in the API are summarized below.

Table 33: Calories Counting Library API

Data Structures and Types	
cc_error_t cc_config_t cc_data_step_t cc_data_step_press_t	cc_upf_config_t cc_handle_t cc_data_pres_t cc_data_calories_t
Dependency Functions	
cc_handle_t -> cc_get_buffer() cc_handle_t -> cc_calories_drdy()	cc_handle_t -> cc_free_buffer()
Initialization and Basic Configuration Functions	
cc_init() cc_update_user_profile()	cc_clear()
Main Process Functions	
cc_data_update_step_pres()	cc_calc_calories()
Service Data Acquisition Functions	
cc_data_read_calories()	

© 2017 Dialog Semiconductor

Command Functions	
cc_data_reset_calories()	
State Functions	
cc_state_get_calories_epochs_num()	

9.3.1 Data Structures and Types

cc_error_t

Type Definition Name	Description
cc_error_t	Execution status of a function: CC_SUCCESS, CC_FAIL

cc_upf_config_t

Data Structure Fields	Type	Description
gender	uint8_t	User's gender type (m/f): CC_GENDER_TYPE_FEMALE, CC_GENDER_TYPE_MALE
age	uint8_t	User's age (in years).
height	uint8_t	User's height (in cm).
weight	uint8_t	User's weight (in kg).

cc_config_t

Data Structure Fields	Type	Description
user_profile	cc_upf_config_t	User profile with which Calories Counting is configured to operate.
step_pres_rate	uint16_t	Step and pressure sensor data rate at which Calories Counting is configured to operate: CC_STEP_PRES_DATA_RATE_INVALID, CC_STEP_PRES_DATA_RATE_1_PER_2SEC_HZ, CC_STEP_PRES_DATA_RATE_1_PER_10SEC_HZ, CC_STEP_PRES_DATA_RATE_1_PER_20SEC_HZ, CC_STEP_PRES_DATA_RATE_1_PER_30SEC_HZ, CC_STEP_PRES_DATA_RATE_1_PER_1MIN_HZ, CC_STEP_PRES_DATA_RATE_1_PER_2MIN_HZ, CC_STEP_PRES_DATA_RATE_1_PER_3MIN_HZ, CC_STEP_PRES_DATA_RATE_1_PER_4MIN_HZ, CC_STEP_PRES_DATA_RATE_1_PER_5MIN_HZ
epoch_interval	uint16_t	The epoch interval at which Calories Counting is configured to operate: CC_CALORIES_EPOCH_INVALID, CC_CALORIES_EPOCH_10SEC, CC_CALORIES_EPOCH_20SEC, CC_CALORIES_EPOCH_30SEC, CC_CALORIES_EPOCH_1MIN, CC_CALORIES_EPOCH_2MIN, CC_CALORIES_EPOCH_3MIN, CC_CALORIES_EPOCH_4MIN, CC_CALORIES_EPOCH_5MIN

© 2017 Dialog Semiconductor

cc_handle_t

Data Structure Fields	Type	Description
cc_get_buffer	Function pointer	Function for memory space allocation.
cc_free_buffer	Function pointer	Function for memory space deallocation.
cc_calories_drdy	Function pointer	Function for epoch change indication (calories data ready).

cc_data_step_t

Data Structure Fields	Type	Description
count	uint16_t	The step counter.

cc_data_pres_t

Data Structure Fields	Type	Description
val	uint32_t	The value of pressure sensor data (in Pa).

cc_data_step_press_t

Data Structure Fields	Type	Description
step_data	cc_data_step_t	The value of step data.
pres_data	cc_data_pres_t	The value of pressure sensor data (in Pa).

cc_data_calories_t

Data Structure Fields	Type	Description
count	double	The calories counter.
distance	double	The distance covered (in m).
epoch_index	uint32_t	The epoch number calories counter refers to.

© 2017 Dialog Semiconductor

9.3.2 Dependency Functions

cc_handle_t -> cc_get_buffer()

Function Name	`cc_error_t (*cc_get_buffer)(uint8_t** buf, uint16_t size)`
Function Description	Implements memory space allocation for variables and structures used internally in Calories Counting Library.
Parameters	`buf` the address of the pointer variable to put the address of the allocated memory space `size` the size of the memory space to be allocated (in bytes)
Return Values	`CC_SUCCESS` for successful execution; ≥ `CC_FAIL` for error
Notes	

cc_handle_t -> cc_free_buffer()

Function Name	`cc_error_t (*cc_free_buffer)(uint8_t* buf)`
Function Description	Implements memory space deallocation for variables and structures having been used internally in Calories Counting Library.
Parameters	`buf` the address of the allocated memory space
Return Values	`CC_SUCCESS` for successful execution; ≥ `CC_FAIL` for error
Notes	

cc_handle_t -> cc_calories_drdy()

Function Name	`cc_error_t (*cc_calories_drdy)(void)`
Function Description	Handles epoch change indication sent when the number of step/pressure data samples that have been provided as input is such that, based on the given rate (`cc_config_t -> step_pres_rate`), an epoch is detected that has been completed.
Parameters	None
Return Values	`CC_SUCCESS` for successful execution; ≥ `CC_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

9.3.3 Initialization and Basic Configuration Functions

cc_init()

Function Name	`cc_error_t cc_init(cc_handle_t* handle, cc_config_t* config)`
Function Description	Initializes Calories Counting with the appropriate configuration in terms of user profile data (i.e. gender, age, height and weight), step and pressure sensor input data rate and epoch interval of calories estimation. Memory space is requested via `cc_get_buffer()` callback function set to `cc_handle_t` instance handle.
Parameters	`handle` the handle structure for Calories Counting `config` the configuration structure for Calories Counting operation
Return Values	`CC_SUCCESS` for successful execution; ≥ `CC_FAIL` for error
Notes	`cc_get_buffer()` and `cc_free_buffer()` must be set.

cc_clear()

Function Name	`cc_error_t cc_clear(void)`
Function Description	Clears Calories Counting, freeing previously allocated resources via `cc_gree_buffer()` callback function set to `cc_handle_t` instance handle.
Parameters	None
Return Values	`CC_SUCCESS` for successful execution; ≥ `CC_FAIL` for error
Notes	

cc_update_user_profile()

Function Name	`cc_error_t cc_update_user_profile(cc_upf_config_t* config)`
Function Description	Updates user profile attributes.
Parameters	None
Return Values	`CC_SUCCESS` for successful execution; ≥ `CC_FAIL` for error
Notes	

9.3.4 Main Process Functions

cc_data_update_step_pres()

Function Name	`cc_error_t cc_data_update_step_pres(cc_data_step_pres_t* data)`
Function Description	Provides new combined step and pressure sensor data sample to Calories Counting, based on the data rate with which the latter has been previously configured. Pressure sensor data must be converted to Pa.
Parameters	`data` the combined step and pressure sensor data sample
Return Values	`CC_SUCCESS` for successful execution; ≥ `CC_FAIL` for error
Notes	

cc_calc_calories()

Function Name	`cc_error_t cc_calc_calories(void)`
Function Description	Executes calories estimation for the current epoch. This function must be called when an epoch change is notified through `cc_calories_drdy()` callback function set to `cc_handle_t` instance handle, i.e. at the end of an epoch.
Parameters	None

© 2017 Dialog Semiconductor

Return Values	CC_SUCCESS for successful execution; ≥ CC_FAIL for error
Notes	

9.3.5 Service Data Acquisition Functions

cc_data_read_calories()

Function Name	cc_error_t cc_data_read_calories(cc_data_calories_t* data)
Function Description	Reads the total calories (and distance) counter since Calories Counting initialization (i.e. cc_init()) or a previously performed 'reset' (i.e. cc_data_read_calories()).
Parameters	data the buffer to store calories counting related total counters being read
Return Values	CC_SUCCESS for successful execution; ≥ CC_FAIL for error
Notes	

9.3.6 Command Functions

cc_data_reset_calories()

Function Name	cc_error_t cc_data_reset_calories(void)
Function Description	Resets the total calories (and distance) counter.
Parameters	None
Return Values	CC_SUCCESS for successful execution; ≥ CC_FAIL for error
Notes	

9.3.7 State Functions

cc_state_get_calories_epochs_num()

Function Name	cc_error_t cc_state_get_calories_epochs_num(uint16_t* num_of_epochs)
Function Description	Gets the number of epochs for which Calories Counting has been performed since its initialization (i.e. since the last cc_init()).
Parameters	num_of_epochs the number of epochs for which Calories Counting has been performed
Return Values	CC_SUCCESS for successful execution; ≥ CC_FAIL for error
Notes	

© 2017 Dialog Semiconductor

9.4 KIWI Basic Activity Tracking

The KIWI Movement AI is the software at the core of KIWI's sensor pattern recognition and intelligence solutions, designed to work seamlessly in both low-level embedded (M0, M3 and M4 chipsets), mobile and cloud-based environments. A more detailed description of the library can be found in the user manual of the DA14681 wearable development kit [1].

kiwi_init()

Function Name	`void kiwi_init(void)`
Function Description	Initializes KIWI basic activity state tracking internal variables, data structures and buffers.
Parameters	None
Return Values	None
Notes	It needs to be called once before the program's main loop.

 © 2017 Dialog Semiconductor

kiwi_main()

Function Name	`int kiwi_main(float rawAccelX, float rawAccelY, float rawAccelZ, int isRightWrist)`
Function Description	This function is the main loop for the kiwi classification library. One accelerometer sample (x, y, z) is passed and stored internally. As new data samples are passed in, data is stored internally and features are computed iteratively as well as in a batch once the internal buffer is filled.
Parameters	`rawAccelX` the raw accelerometer x-axis data (in 'g's) `rawAccelY` the raw accelerometer y-axis data (in 'g's) `rawAccelZ` the raw accelerometer z-axis data (in 'g's) `isRightWrist` the value for the wearable/accelerometer placement selection on the right wrist (i.e. 'right wrist' = 1, 'left wrist' = 0). Default is set to right wrist
Return Values	0 for no new classification; 1 for new classification
Notes	If there is no new classification indicated, the previous activity state should be used. Activity state classification result can be acquired using `kiwi_get_state()` Given the watch is on the right wrist with a fist pointing forwards infront of the body and palms down, the raw accelerometer input to `kiwi_main` need to be in the following orientation - positive x pointing up the arm; negative y pointing away from the thumb; positive z pointing down/away from the palm. When wearing the watch on the left wrist, this orientation remains the same and the 'isRightWrist' parameter is set to 0.

kiwi_get_state()

Function Name	`int kiwi_get_state(void)`
Function Description	Returns the enumeration of the last detected activity state (i.e. WALK, RUN, SIT, STAND, SLEEP or OTHER state). Should be called when `kiwi_main` () returns 1. If called when `kiwi_main` () returns 0, the returned value will be the previously classified activity.
Parameters	None
Return Values	Enumeration of the last detected activity state `WALK_STATE , RUN_STATE, SIT_STATE, STAND_STATE, SLEEP_STATE, OTHER_STATE`
Notes	

kiwi_get_step_count()

Function Name	`int kiwi_get_step_count(void)`
Function Description	This function returns the accumulated step count while in walking and running states since the initialization of KIWI basic activity state tracking or the last performed 'reset' of the implemented step counting process using `kiwi_reset_step_count()`.
Parameters	None
Return Values	The number of steps counted during processing time.
Notes	

© 2017 Dialog Semiconductor

kiwi_reset_step_count()

Function Name	`int kiwi_reset_step_count(void)`
Function Description	Resets the number of steps counted.
Parameters	None
Return Values	None
Notes	

kiwi_set_bodytype_vars()

Function Name	`void kiwi_set_bodytype_vars(uint16_t gender, uint16_t age, uint16_t height, uint16_t weight)`
Function Description	Configures KIWI basic activity state tracking with the anthropometric variables defining a user profile (i.e. body type), which are taken into consideration when calories counting data metrics are produced.
Parameters	`gender` the ASCII value of the gender type (i.e. male = 'm' (0x6d) and female = 'f' (0x66)) `age` the value of the age (in years) `height` the value of the height (in inches) `weight` the value of the weight (in lbs.)
Return Values	None
Notes	

kiwi_get_sleep_stage()

Function Name	`int kiwi_get_sleep_stage(void)`
Function Description	Returns the enumeration of the last detected sleep stage (i.e. NONE, DEEP, LIGHT, or AWAKE sleep stage).
Parameters	None
Return Values	Enumeration of the last detected sleep stage: `NONE = 0; DEEP = 1, LIGHT = 2, AWAKE = 3`
Notes	

kiwi_get_calorie_count()

Function Name	`float kiwi_get_calorie_count(void)`
Function Description	Returns an estimate of ‘net/active’ calories having been burned since the initialization of KIWI basic activity state tracking.
Parameters	None
Return Values	An estimate of ‘net/active’ calories burned during processing time.
Notes	Resting activities (i.e. Basal Metabolic Rate - BMR) do not accumulate calories in the respective calories counter Calories counting is affected by the user's gender and body type being set using `kiwi_set_bodytype_vars()`.

© 2017 Dialog Semiconductor

10 Peripheral Device Drivers

The system implements different interfaces for the configuration, control and data acquisition from the sensor modules (BMI160, BMM150, BME280 and DI5115) and the rest of the peripheral devices (SX9300, AB08X5, LS012B7DH03 and FXL6408) controlled by the DA14681, which are used to implement the upper layers of application functionality. The driver architecture provided in the Wearable DK software is presented in Figure 6.

Regarding the BMI160, BMM150 and BME280 sensor modules and FXL6408 peripheral module, the corresponding drivers provide the necessary abstraction to the upper layers, when access for control and data acquisition is required. Although the modules have different functionality and corresponding options for the type of data as well as the way data is acquired from their registers, all drivers provide the following common functionality:

- **INIT-CONFIG:** Initializing, setting the power mode state and configuring the operation mode of the module.
- **SAMPLE:** Reading (sensor) data produced by the module (only for sensor module drivers).
- **CMD:** Sending commands to the module.
- **STATE:** Getting the status of the module.
- **CONFIG:** Setting/getting specific setting options for the module.
- **LOW-ACCESS:** Setting/getting register values provided by the module.

Drivers for DI5115, SX9300, AB08X5, FXL6408 and LS013B7DH03 adopt a different structure for their API due to the lower complexity and more simplistic functionality. Generally, these drivers should be initialized, configured (if needed, since initialization includes configuration) and then use the provided API to acquire/set data/configurations.

An important aspect of the whole DA14680 driver configuration, which is also depicted in Figure 6, is that the FXL6408 driver is not only used in communication with a task (wearable task) but also for communication with another driver (LS013B7DH03). The LS013B7DH03 driver uses it to control the display enable pin, since it is directly controlled through the FXL6408 GPIO expander module.

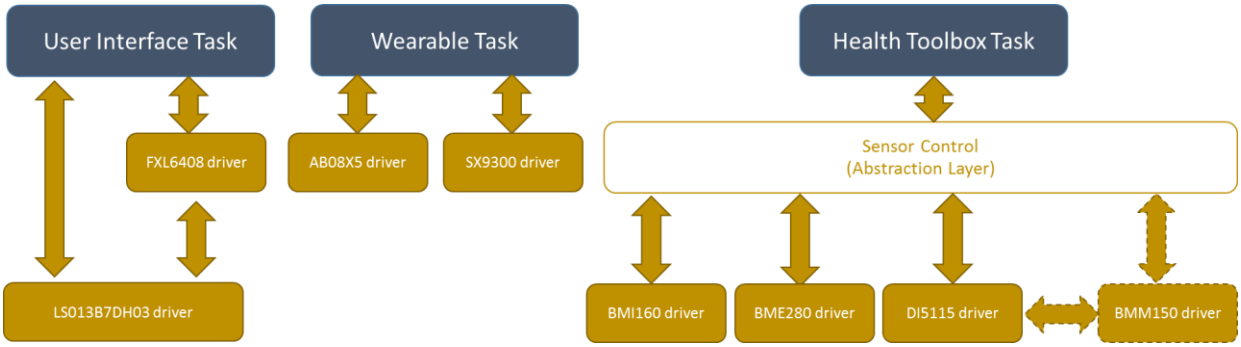

Figure 6: Peripheral Device Driver Architecture

© 2017 Dialog Semiconductor

10.1.1 BMI160 Sensor Driver

The BMI160 sensor driver API provides functions for the configuration, control and sensor data acquisition from the accelerometer and gyroscope sensors integrated in the corresponding modules, as well as from the magnetometer sensor integrated in the BMM150 sensor module, which in turn is controlled by the BMI160, as shown in [9]. Below mainly the functions are presented which are used by the Wearable project, while a complete list of the functions is presented in file `bmi160.h`, based on the datasheet [10] from Bosch Sensortec.

General guidelines for using the BMI160 sensor driver are as follows:

- Initialize sensor module using the function `bmi160_initialize_sensor()`. The device address for the sensor module must be set, along with pointers to functions which implement the SPI/I2C bus access and introduce delay.
- Set the power mode state of the module by subsetting the power mode state of each included sensor (i.e. accelerometer, gyroscope) using the function `bmi160_set_sensor_state()`.
- Configure the sensor module in terms of specific setting/operation features (`setting_feature = accel_out_data_rate`, `accel_bandwidth`, `accel_range`, etc.) using the `bmi160_set_[setting_feature]()` set of functions.
- When the current configuration state of the sensor module is to be checked in terms of specific setting/operation features, use the `bmi160_get_[setting_feature]()` set of functions.
- Make appropriate decisions by getting the status of the sensor module (`state_feature = accel_state`, `gyro_state`, `drdy_accel_stat`, `drdy_gyro_stat`, etc.) using the `bmi160_get_[state_feature]()` set of functions.
- Access the sensor module's registers (`reg_name`) when low level control and data handling is required, using the `bmi160_[set/get]_reg_[reg_name]()` set of functions.
- When the power mode state of the module is SUSPEND (or LOWPOWER), between two successive register write accesses a delay shall be introduced (~400 μs), while burst writes and FIFO burst reads are not supported. In order for the latter to be performed, first change the power mode state to NORMAL.
- Read inertial sensor (`sensor = accel`, `gyro`, `mag`) data using the `bmi160_read_[sensor]_[x|y|z]()` set of functions.
- When the sensor module is configured to use the internal FIFO to store (accelerometer, gyroscope and magnetometer frame) data, read a set of sensor data using the function `bmi160_read_fifo_data()`. If all sensors are in either SUSPEND or LOWPOWER power mode state, the FIFO must not be read.
- Read step counter data using `bmi160_read_step_counter()` and the clear step counter using the function `bmi160_clear_step_counter()`. If the power mode state of the accelerometer is LOWPOWER, it is recommended first to change the accelerometer power mode state to NORMAL.
- Read temperature sensor data using the function `bmi160_read_temperature()`.

A list of all API data structures/types and functions provided by the BMI160 sensor driver and presented in this section follows.

Table 34: BMI160 Driver API

Data Structures and Types	
bmi160_t bmi160_gyro_t	bmi160_mag_t bmi160_accel_t
Dependency Functions	
bmi160_t -> bus_write() bmi160_t -> burst_read()	bmi160_t -> bus_read() bmi160_t -> delay_msec()

© 2017 Dialog Semiconductor

Initialization and Basic Configuration Functions	
bmi160_initialize_sensor()	bmi160_set_sensor_state()
Sensor Data Acquisition Functions	
bmi160_read_fifo_data() bmi160_get_sensor_time() bmi160_read_mag_r() bmi160_read_gyro_xyz() bmi160_read_temperature()	bmi160_get_fifo_length() bmi160_read_mag_xyz() bmi160_compensate_bmm150_mag_[x/y/z]() bmi160_read_accel_xyz() bmi160_read_step_counter()
Command Functions	
bmi160_flush_fifo() bmi160_clear_step_counter()	bmi160_reset_intrs() bmi160_soft_reset()
State Functions	
bmi160_get_chip_id() bmi160_get_gyro_state() bmi160_get_drdy_mag_stat() bmi160_get_drdy_accel_stat() bmi160_get_significant_motion_stat() bmi160_get_pmu_trigger_stat() bmi160_get_single_tap_stat() bmi160_get_fifo_full_stat() bmi160_get_no_motion_stat()	bmi160_get_mag_state() bmi160_get_accel_state() bmi160_get_drdy_gyro_stat() bmi160_get_step_stat() bmi160_get_any_motion_stat() bmi160_get_double_tap_stat() bmi160_get_drdy_stat() bmi160_get_fifo_watermark_stat()

© 2017 Dialog Semiconductor

Configuration Functions	
bmi160_[set/get]_accel_out_data_rate()	bmi160_[set/get]_accel_bandwidth()
bmi160_[set/get]_accel_undersampling_enable()	bmi160_[set/get]_accel_range()
bmi160_[set/get]_gyro_out_data_rate()	bmi160_[set/get]_gyro_bandwidth()
bmi160_[set/get]_gyro_range()	bmi160_[set/get]_mag_out_data_rate()
bmi160_[set/get]_fifo_gyro_downsampling_enable()	bmi160_[set/get]_fifo_gyro_filter_data_enable()
bmi160_[set/get]_fifo_accel_downsampling_enable()	bmi160_[set/get]_fifo_accel_filter_data_enable()
bmi160_[set/get]_fifo_watermark_level()	bmi160_[set/get]_fifo_stop_on_full_enable()
bmi160_[set/get]_fifo_time_enable()	bmi160_[set/get]_fifo_header_enable()
bmi160_[set/get]_fifo_mag_enable()	bmi160_[set/get]_fifo_accel_enable()
bmi160_[set/get]_fifo_gyro_enable()	bmi160_[set/get]_i2c_device_address()
bmi160_[set/get]_mag_manual_enable()	bmi160_[set/get]_mag_read_address()
bmi160_[set/get]_mag_write_address()	bmi160_[set/get]_mag_write_data()
bmi160_[set/get]_any_motion_[x/y/z]_enable()	bmi160_[set/get]_double_tap_enable()
bmi160_[set/get]_single_tap_enable()	bmi160_[set/get]_drdy_enable()
bmi160_[set/get]_fifo_full_enable()	bmi160_[set/get]_fifo_watermark_enable()
bmi160_[set/get]_no_motion_[x/y/z]_enable()	bmi160_[set/get]_step_detector_enable()
bmi160_[set/get]_intr_1_output_edge_level()	bmi160_[set/get]_intr_1_output_level()
bmi160_[set/get]_intr_1_output_odrn_pshpll()	bmi160_[set/get]_intr_1_output_enable()
bmi160_[set/get]_intrs_latched()	bmi160_[set/get]_any_motion_intr_1()
bmi160_[set/get]_no_motion_intr_1()	bmi160_[set/get]_double_tap_intr_1()
bmi160_[set/get]_single_tap_intr_1()	bmi160_[set/get]_pmu_trigger_intr_1()
bmi160_[set/get]_fifo_full_intr_1()	bmi160_[set/get]_fifo_watermark_intr_1()
bmi160_[set/get]_drdy_intr_1()	bmi160_[set/get]_any_motion_duration()
bmi160_[set/get]_slow_no_motion_duration()	bmi160_[set/get]_any_motion_threshold()
bmi160_[set/get]_slow_no_motion_threshold()	bmi160_[set/get]_slow_no_motion_select()
bmi160_[set/get]_significant_motion_select()	bmi160_[set/get]_significant_motion_skip()
bmi160_[set/get]_significant_motion_proof()	bmi160_[set/get]_tap_duration()
bmi160_[set/get]_tap_shock_duration()	bmi160_[set/get]_tap_quiet_duration()
bmi160_[set/get]_tap_threshold()	bmi160_[set/get]_if_mode()
bmi160_[set/get]_gyro_sleep_trigger()	bmi160_[set/get]_gyro_wakeup_trigger()
bmi160_[set/get]_gyro_sleep_trigger_state()	bmi160_[set/get]_gyro_wakeup_enable()
bmi160_[set/get]_step_min_steptime()	bmi160_[set/get]_step_min_threshold()
bmi160_[set/get]_step_alpha()	bmi160_[set/get]_step_min_buffer_size()
bmi160_[set/get]_step_counter_enable()	bmi160_[set/get]_target_page()
bmi160_[set/get]_paging_enable()	bmi160_[set/get]_pullup_configuration()
Low Level Register Access Functions	
bmi160_[set/get]_reg_[reg_name]()	
Magnetometer Control and Configuration Functions (Indirect Access)	
bmi160_init_bmm150_mag_interface()	bmi160_get_bmm150_mag_interface()
bmi160_wakeup_bmm150_mag()	bmi160_suspend_bmm150_mag()
bmi160_init_bmm150_mag_trim_registers()	bmi160_set_bmm150_mag_preset_mode()
bmi160_init_bmm150_mag_trim_registers()	

© 2017 Dialog Semiconductor

10.1.1.1 Data Structures and Types

bmi160_t

Data Structure Fields	Type	Description
chip_id	uint8_t	Chip ID of the BMI160 sensor module.
dev_addr	uint8_t	Device address of the BMI160 sensor module.
bus_write	Function pointer	Function for SPI/I2C bus write-access.
bus_read	Function pointer	Function for SPI/I2C bus read-access.
burst_read	Function pointer	Function for SPI/I2C bus burst-read-access.
delay_msec	Function pointer	Function for introducing delay (in ms).

bmi160_mag_t

Data Structure Fields	Type	Description
x	int16_t	Magnetometer x-axis raw data value.
y	int16_t	Magnetometer y-axis raw data value.
z	int16_t	Magnetometer z-axis raw data value.

bmi160_gyro_t

Data Structure Fields	Type	Description
x	int16_t	Gyroscope x-axis raw data value.
y	int16_t	Gyroscope y-axis raw data value.
z	int16_t	Gyroscope z-axis raw data value.

bmi160_accel_t

Data Structure Fields	Type	Description
x	int16_t	Accelerometer x-axis raw data value.
y	int16_t	Accelerometer y-axis raw data value.
z	int16_t	Accelerometer z-axis raw data value.

10.1.1.2 Dependency Functions

bmi160_t -> bus_write()

Function Name	`int8_t (*bus_write)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data, uint8_t cnt)`
Function Description	Implements write-access over the SPI / I2C bus for writing data to BMI160 sensor module registers.
Parameters	`dev_addr` the device address `reg_addr` the address of the register to write `reg_data` the data to write `cnt` the size of data (in bytes)
Return Values	0 for successful execution; ≠0 for error
Notes	

 © 2017 Dialog Semiconductor

bmi160_t -> bus_read()

Function Name	`int8_t (*bus_read)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data, uint8_t cnt)`
Function Description	Implements read-access over the SPI / I2C bus for reading data from BMI160 sensor module registers.
Parameters	`dev_addr` the device address `reg_addr` the address of the register to read `reg_data` the buffer to put register data being read to `cnt` the size of data (in bytes)
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_t -> burst_read()

Function Name	`int8_t (*burst_read)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data, uint32_t len)`
Function Description	Implements burst-read-access over the SPI / I2C bus for reading data from BMI160 sensor module registers.
Parameters	`dev_addr` the device address `reg_addr` the address of the register to read `reg_data` the buffer to put register data being read to `len` the size of data (in bytes)
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_t -> delay_msec()

Function Name	`void (*delay_msec)(uint32_t ms)`
Function Description	Implements a delay in ms that may be necessary for the BMI160 sensor module driver functionality implementation.
Parameters	`ms` the delay (in ms) to be introduced
Return Values	None
Notes	

10.1.1.3 Initialization and Basic Configuration Functions

bmi160_initialize_sensor()

Function Name	`int8_t bmi160_initialize_sensor(bmi160_t* s_bmi160)`
Function Description	(INIT-CONFIG) Initializes the BMI160 sensor module, creating the appropriate instance handle for the module which includes references to the SPI / I2C bus access functions used for reading/writing values to the registers, soft-resetting its state, flushing its internal FIFO and initializing the secondary interface with BMM150, which is connected to BMI160 in cascade mode.
Parameters	`s_bmi160` the handle structure for BMI160 driver
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_set_sensor_state()

Function Name	`int8_t bmi160_set_sensor_state(uint8_t state)`
Function Description	(INIT-CONFIG) Sets the state of a sensor integrated in BMI160 sensor module.

Parameters	`state` the state to set for a sensor
	`ACC_SUSPEND , ACC_NORMAL , ACC_LOWPOWER,`
	`GYR_SUSPEND , GYR_NORMAL , GYR_FASTSTARTUP,`
	`MAG_SLV_SUSPEND, MAG_SLV_NORMAL, MAG_SLV_LOWPOWER`

Return Values	0 for successful execution; ≠0 for error
Notes	On successful execution of the corresponding register access it is introduced a delay of `[ACC/GYR/MAG_SLV]_MODE_TRANSITION_DELAY`.

10.1.1.4 Sensor Data Acquisition Functions

bmi160_read_fifo_data()

Function Name	`int8_t bmi160_read_fifo_data(uint8_t *v_fifo_data_u8, uint32_t v_len_u32)`
Function Description	(SAMPLE) Reads sensor data stored in the FIFO.
Parameters	`v_fifo_data_u8` the buffer to transfer the FIFO sensor data `v_len_u32` the size of data (in bytes) to transfer
Return Values	0 for successful execution; ≠0 for error
Notes	The maximum size of FIFO in bytes is 1024 (`FIFO_FRAME`)

bmi160_get_fifo_length()

Function Name	`int8_t bmi160_get_fifo_length(uint16_t* v_fifo_length_u16)`
Function Description	(SAMPLE) Reads the size of data in bytes (length) stored in the FIFO
Parameters	`v_fifo_length_u16` the length of the FIFO
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_sensor_time()

Function Name	`int8_t bmi160_get_sensor_time(uint32_t* v_sensor_time_u32)`
Function Description	(SAMPLE) Reads sensor time provided by the BMI160 sensor module in units of 0.039 ms.
Parameters	`v_sensor_time_u32` the sensor time
Return Values	0 for successful execution; ≠0 for error
Notes	1024 is 40 ms, 256 is 10 ms, 64 is 2.5 ms, 16 is 0.625 ms etc.

bmi160_read_mag_xyz()

Function Name	`int8_t bmi160_read_mag_xyz(bmi160_mag_t *mag_xyz)`
Function Description	(SAMPLE) Reads magnetometer xyz-axis data.
Parameters	`mag_xyz` the magnetometer xyz-axis data
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_read_mag_r()

Function Name	`int8_t bmi160_read_mag_xyz(uint16_t *v_mag_r_u16)`
Function Description	(SAMPLE) Reads magnetometer RHALL (r) data.
Parameters	`v_mag_r_u16` the magnetometer RHALL (r) data
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_compensate_bmm150_mag_[x/y/z]()

Function Name	`int16_t bmi160_compensate_bmm150_mag_x(int16_t v_mag_x_s16, uint16_t v_mag_r_u16)`
	`int16_t bmi160_compensate_bmm150_mag_y(int16_t v_mag_y_s16, uint16_t v_mag_r_u16)`
	`int16_t bmi160_compensate_bmm150_mag_z(int16_t v_mag_z_s16, uint16_t v_mag_r_u16)`

Function Description	(SAMPLE) Converts uncompensated magnetometer x/y/z-axis data values to true ones.
Parameters	`v_mag_[x/y/z]_s16` the magnetometer x/y/z-axis data value `v_mag_r_u16` the RHALL (r) data value
Return Values	The true-compensated magnetometer x/y/z-axis value
Notes	

bmi160_read_gyro_xyz()

Function Name	`int8_t bmi160_read_gyro_xyz(bmi160_gyro_t *gyro_xyz)`
Function Description	(SAMPLE) Reads gyroscope xyz-axis data.
Parameters	`gyro_xyz` the gyroscope xyz-axis data
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_read_accel_xyz()

Function Name	`int8_t bmi160_read_accel_xyz(bmi160_accel_t *accel_xyz)`
Function Description	(SAMPLE) Reads accelerometer xyz-axis data.
Parameters	`accel_xyz` the accelerometer xyz-axis data
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_read_temperature()

Function Name	`int8_t bmi160_read_temperature(int16_t *v_temp_s16)`
Function Description	(SAMPLE) Reads temperature data.
Parameters	`v_temp_s16` the temperature data
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_read_step_counter()

Function Name	`int8_t bmi160_read_step_counter(int16_t *v_step_cnt_s16)`
Function Description	(SAMPLE) Reads the number of detected steps.
Parameters	`v_step_cnt_s16` the step counter data
Return Values	0 for successful execution; ≠0 for error
Notes	

10.1.1.5 Command Functions

bmi160_flush_fifo()

Function Name	`int8_t bmi160_flush_fifo(void)`
Function Description	(CMD) Flushes FIFO content.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_reset_intrs()

Function Name	`int8_t bmi160_reset_intrs(void)`
Function Description	(CMD) Resets interrupts.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_clear_step_counter()

Function Name	`int8_t bmi160_clear_step_counter(void)`
Function Description	(CMD) Clears step counter data.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_soft_reset()

Function Name	`int8_t bmi160_soft_reset(void)`
Function Description	(CMD) Performs a soft reset to the BMI160 sensor module.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	On successful execution it is introduced a `GYR_MODE_TRANSITION_MAX_DELAY` delay.

 © 2017 Dialog Semiconductor

10.1.1.6 State Functions

bmi160_get_chip_id()

Function Name	`int8_t bmi160_get_chip_id(uint8_t* v_chip_id_u8)`
Function Description	(STATE) Reads the chip id of the BMI160 sensor module.
Parameters	`v_chip_id_u8` the chip id
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_mag_state()

Function Name	`int8_t bmi160_get_mag_state(uint8_t *v_mag_power_mode_state_u8)`
Function Description	(STATE) Reads the power mode state of the magnetometer.
Parameters	`v_mag_power_mode_state_u8` the magnetometer power mode state `MAG_SLV_SUSPEND, MAG_SLV_NORMAL, MAG_SLV_LOWPOWER`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_gyro_state()

Function Name	`int8_t bmi160_get_gyro_state(uint8_t *v_gyro_power_mode_state_u8)`
Function Description	(STATE) Reads the power mode state of the gyroscope.
Parameters	`v_gyro_power_mode_state_u8` : the gyroscope power mode state `GYR_SUSPEND, GYR_NORMAL, GYR_FASTSTARTUP`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_accel_state()

Function Name	`int8_t bmi160_get_accel_state(uint8_t *v_accel_power_mode_state_u8)`
Function Description	(STATE) Reads the power mode state of the accelerometer.
Parameters	`v_accel_power_mode_state_u8` the accelerometer power mode state `ACC_SUSPEND, ACC_NORMAL, ACC_LOWPOWER`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_drdy_mag_stat()

Function Name	`int8_t bmi160_get_drdy_mag_stat(uint8_t* v_data_rdy_u8)`
Function Description	(STATE) Reads the status of data ready (drdy) status bit, indicating whether a new magnetometer sample value has been set to the corresponding data register.
Parameters	`v_data_rdy_u8` the value of the magnetometer drdy interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_get_drdy_gyro_stat()

Function Name	`int8_t bmi160_get_drdy_gyro_stat(uint8_t* v_data_rdy_u8)`
Function Description	(STATE) Reads the status of data ready (drdy) status bit, indicating whether a new gyroscope sample value has been set to the corresponding data register.
Parameters	`v_data_rdy_u8` the value of the gyroscope drdy interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_drdy_accel_stat()

Function Name	`int8_t bmi160_get_drdy_accel_stat(uint8_t* v_data_rdy_u8)`
Function Description	(STATE) Reads the status of data ready (drdy) status bit, indicating whether a new accelerometer sample value has been set to the corresponding data register.
Parameters	`v_data_rdy_u8` the value of the accelerometer drdy interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_step_stat()

Function Name	`int8_t bmi160_get_step_stat(uint8_t* v_step_intr_u8)`
Function Description	(STATE) Reads the status of the step detection interrupt status bit, indicating the detection of a step.
Parameters	`v_step_intr_u8` the value of the step detection interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_significant_motion_stat()

Function Name	`int8_t bmi160_get_significant_motion_stat( uint8_t* v_significant_motion_intr_u8)`
Function Description	(STATE) Reads the status of the significant motion interrupt status bit, indicating the detection of a significant motion event.
Parameters	`v_significant_motion_intr_u8` the value of the significant motion interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_any_motion_stat()

Function Name	`int8_t bmi160_get_any_motion_stat(uint8_t* v_any_motion_intr_u8)`
Function Description	(STATE) Reads the status of the any-motion interrupt status bit, indicating the detection of an ‘any-motion’ event.
Parameters	`v_any_motion_intr_u8` the value of the any-motion interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_get_pmu_trigger_stat()

Function Name	`int8_t bmi160_get_pmu_trigger_stat(uint8_t* v_pmu_trigger_intr_u8)`
Function Description	(STATE) Reads the status of the power mode (pmu) trigger interrupt status bit, indicating the transition of the gyroscope to a different power mode.
Parameters	`v_pmu_trigger_intr_u8` the value of the pmu-trigger interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_double_tap_stat()

Function Name	`int8_t bmi160_get_double_tap_stat(uint8_t* v_double_tap_intr_u8)`
Function Description	(STATE) Reads the status of the double-tap interrupt status bit, indicating the detection of a ‘double-tap’ event.
Parameters	`v_double_tap_intr_u8` the value of the double-tap interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_single_tap_stat()

Function Name	`int8_t bmi160_get_single_tap_stat(uint8_t* v_single_tap_intr_u8)`
Function Description	(STATE) Reads the status of the single-tap interrupt status bit, indicating the detection of a ‘single-tap’ event.
Parameters	`v_single_tap_intr_u8` the value of the single-tap interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_drdy_stat()

Function Name	`int8_t bmi160_get_drdy_stat(uint8_t* v_data_rdy_intr_u8)`
Function Description	(STATE) Reads the status of the data ready (drdy) status bit, indicating whether a new sensor sample value has been set to the corresponding data register.
Parameters	`v_data_rdy_intr_u8` the value of the drdy interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_fifo_full_stat()

Function Name	`int8_t bmi160_get_fifo_full_stat(uint8_t* v_fifo_full_intr_u8)`
Function Description	(STATE) Reads the status of the FIFO full interrupt status bit, indicating whether data in FIFO have reached the maximum size.
Parameters	`v_fifo_full_intr_u8` the value of the FIFO full interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_get_fifo_watermark_stat()

Function Name	`int8_t bmi160_get_fifo_watermark_stat(uint8_t* v_fifo_wm_intr_u8)`
Function Description	(STATE) Reads the status of the FIFO watermark (wm) interrupt status bit, indicating whether data in FIFO have reached a specific level, which is set using bmi160_set_fifo_watermark_level() function.
Parameters	`v_fifo_wm_intr_u8` the value of the FIFO wm interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_no_motion_stat()

Function Name	`int8_t bmi160_get_no_motion_stat(uint8_t* v_no_motion_intr_u8)`
Function Description	(STATE) Reads the status of the no-motion interrupt status bit, indicating the detection of a 'no-motion' event.
Parameters	`v_no_motion_intr_u8` the value of the no-motion interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

10.1.1.7 Configuration Functions

bmi160_[set/get]_accel_out_data_rate()

Function Name	`int8_t bmi160_set_accel_out_data_rate(uint8_t v_output_data_rate_u8)` `int8_t bmi160_get_accel_out_data_rate(uint8_t* v_output_data_rate_u8)`
Function Description	(CONFIG) Sets/gets the accelerometer output data rate (odr), that is, the sampling data rate.

Parameters	`v_output_data_rate_u8` the value of the accelerometer output data rate
	`BMI160_ACCEL_OUTPUT_DATA_RATE_0_78HZ,`
	`BMI160_ACCEL_OUTPUT_DATA_RATE_1_56HZ,`
	`BMI160_ACCEL_OUTPUT_DATA_RATE_3_12HZ,`
	`BMI160_ACCEL_OUTPUT_DATA_RATE_6_25HZ,`
	`BMI160_ACCEL_OUTPUT_DATA_RATE_12_5HZ,`
	`BMI160_ACCEL_OUTPUT_DATA_RATE_25HZ,`
	`BMI160_ACCEL_OUTPUT_DATA_RATE_50HZ,`
	`BMI160_ACCEL_OUTPUT_DATA_RATE_100HZ,`
	`BMI160_ACCEL_OUTPUT_DATA_RATE_200HZ,`
	`BMI160_ACCEL_OUTPUT_DATA_RATE_400HZ,`
	`BMI160_ACCEL_OUTPUT_DATA_RATE_800HZ,`
	`BMI160_ACCEL_OUTPUT_DATA_RATE_1600HZ`

Return Values	0 for successful execution; ≠0 for error
Notes	For output data rates lower than 12.5 Hz, the accelerometer must operate with under-sampling mode enabled (`acc_us` = 1) (see `bmi160_[set/get]_accel_undersampling_enable()`)

 © 2017 Dialog Semiconductor

bmi160_[set/get]_accel_bandwidth()

Function Name	`int8_t bmi160_set_accel_bandwidth(uint8_t v_bw_u8)` `int8_t bmi160_get_accel_bandwidth(uint8_t* v_bw_u8)`
Function Description	(CONFIG) Sets/gets the accelerometer bandwidth, that is, either the Over-Sampling Rate (OSR) using the integrated accelerometer digital filter (`acc_us` under-sampling parameter is disabled) or the under-sampling rate/AVeraGing (AVG) (`acc_us` under-sampling parameter is enabled).
Parameters	`v_bw_u8` the value of the accelerometer bandwidth `BMI160_ACCEL_OSR4_AVG1` , `BMI160_ACCEL_OSR2_AVG2`, `BMI160_ACCEL_NORMAL_AVG4`, `BMI160_ACCEL_CIC_AVG8`, `BMI160_ACCEL_RES_AVG16` , `BMI160_ACCEL_RES_AVG32`, `BMI160_ACCEL_RES_AVG64` , `BMI160_ACCEL_RES_AVG128`
Return Values	0 for successful execution; ≠0 for error
Notes	If digital filter configuration is enabled (`acc_us` = 0), the options are: OSR4 mode, OSR2 mode, normal mode, CIS mode If under-sampling mode is enabled (`acc_us`=1), the options are: averaging 1 (AVG1 or no-averaging), 2 (AVG2), 4 (AVG4), 8 (AVG8), 16 (AVG16), 32 (AVG32), 64 (AVG64), 128 (AVG128) samples

bmi160_[set/get]_accel_undersampling_enable()

Function Name	`int8_t bmi160_set_accel_undersampling_enable(`
	`uint8_t v_accel_under_sampling_u8)`
	`int8_t bmi160_get_accel_undersampling_enable(`
	`uint8_t* v_accel_under_sampling_u8)`

Function Description	(CONFIG) Sets/gets the accelerometer under-sampling mode (`acc_us`). If under-sampling is disabled the filter accelerometer digital filter configuration is enabled, and vice versa.
Parameters	`v_accel_under_sampling_u8` the value of the accelerometer under-sampling mode `BMI160_DISABLE`, `BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	If digital filter configuration is enabled (`acc_us` = 0), the options are: OSR4 mode, OSR2 mode, normal mode, CIS mode If under-sampling mode is enabled (`acc_us` = 1), the options are: averaging 1 (AVG1 or no-averaging), 2 (AVG2), 4 (AVG4), 8 (AVG8), 16 (AVG16), 32 (AVG32), 64 (AVG64), 128 (AVG128) samples

bmi160_[set/get]_accel_range()

Function Name	`int8_t bmi160_set_accel_range(uint8_t v_range_u8)` `int8_t bmi160_get_accel_range(uint8_t* v_range_u8)`
Function Description	(CONFIG) Sets/gets the accelerometer g-range.
Parameters	`v_range_u8` the value of the accelerometer range `BMI160_ACCEL_RANGE_2G`, `BMI160_ACCEL_RANGE_4G`, `BMI160_ACCEL_RANGE_8G`, `BMI160_ACCEL_RANGE_16G`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_gyro_out_data_rate()

Function Name	`int8_t bmi160_set_gyro_out_data_rate(`
	`uint8_t v_output_data_rate_u8)`
	`int8_t bmi160_get_gyro_out_data_rate(`
	`uint8_t* v_output_data_rate_u8)`

Function Description	(CONFIG) Sets/gets the gyroscope Output Data Rate (ODR), that is, the sampling data rate.

Parameters	`v_output_data_rate_u8` the value of the gyroscope output data rate
	`BMI160_GYRO_OUTPUT_DATA_RATE_25HZ,`
	`BMI160_GYRO_OUTPUT_DATA_RATE_50HZ,`
	`BMI160_GYRO_OUTPUT_DATA_RATE_100HZ,`
	`BMI160_GYRO_OUTPUT_DATA_RATE_200HZ,`
	`BMI160_GYRO_OUTPUT_DATA_RATE_400HZ,`
	`BMI160_GYRO_OUTPUT_DATA_RATE_800HZ,`
	`BMI160_GYRO_OUTPUT_DATA_RATE_1600HZ,`
	`BMI160_GYRO_OUTPUT_DATA_RATE_3200HZ`

Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_gyro_bandwidth()

Function Name	`int8_t bmi160_set_gyro_bandwidth(uint8_t v_bw_u8)` `int8_t bmi160_get_gyro_bandwidth(uint8_t* v_bw_u8)`
Function Description	(CONFIG) Sets/gets the gyroscope bandwidth, that is, the Over-Sampling Rate (OSR) using the integrated gyroscope digital filter.

Parameters	`v_bw_u8` the value of the gyroscope bandwidth
	`BMI160_GYRO_OSR4_MODE , BMI160_GYRO_OSR2_MODE,`
	`BMI160_GYRO_NORMAL_MODE, BMI160_GYRO_CIC_MODE`

Return Values	0 for successful execution; ≠0 for error
Notes	The options for the digital filter configuration are: OSR4 mode, OSR2 mode, normal mode, CIS mode

bmi160_[set/get]_gyro_range()

Function Name	`int8_t bmi160_set_gyro_range(uint8_t v_range_u8)` `int8_t bmi160_get_gyro_range(uint8_t* v_range_u8)`
Function Description	(CONFIG) Sets/gets the gyroscope angular rate measurement range or resolution.

Parameters	`v_range_u8` the value of the gyroscope range
	`BMI160_GYRO_RANGE_2000_DEG_SEC,`
	`BMI160_GYRO_RANGE_1000_DEG_SEC,`
	`BMI160_GYRO_RANGE_500_DEG_SEC,`
	`BMI160_GYRO_RANGE_250_DEG_SEC,`
	`BMI160_GYRO_RANGE_125_DEG_SEC`

Return Values	0 for successful execution; ≠0 for error
Notes	Angular rate measurement range results in specific resolution, as follows: ±2000 deg/s range → 16.4 LSB / deg/s ⇔ 61.0 mdeg/s / LSB resolution ±1000 deg/s range → 32.8 LSB / deg/s ⇔ 30.5 mdeg/s / LSB resolution ±500 deg/s range → 65.6 LSB / deg/s ⇔ 15.3 mdeg/s / LSB resolution ±250 deg/s range → 131.2 LSB / deg/s ⇔ 7.6 mdeg/s / LSB resolution ±125 deg/s range → 262.4 LSB / deg/s ⇔ 3.8 mdeg/s / LSB resolution

© 2017 Dialog Semiconductor

bmi160_[set/get]_mag_out_data_rate()

Function Name	`int8_t bmi160_set_mag_out_data_rate(uint8_t v_output_data_rate_u8)` `int8_t bmi160_get_mag_out_data_rate(uint8_t* v_output_data_rate_u8)`
Function Description	(CONFIG) Sets/gets the magnetometer output-poll data rate (odr), that is, the sampling data rate of the magnetometer xyz-axis data values through the respective interface.

Parameters	`v_output_data_rate_u8` the value of the magnetometer output data rate
	`BMI160_MAG_OUTPUT_DATA_RATE_0_78HZ,`
	`BMI160_MAG_OUTPUT_DATA_RATE_1_56HZ,`
	`BMI160_MAG_OUTPUT_DATA_RATE_3_12HZ,`
	`BMI160_MAG_OUTPUT_DATA_RATE_6_25HZ,`
	`BMI160_MAG_OUTPUT_DATA_RATE_12_5HZ,`
	`BMI160_MAG_OUTPUT_DATA_RATE_25HZ,`
	`BMI160_MAG_OUTPUT_DATA_RATE_50HZ,`
	`BMI160_MAG_OUTPUT_DATA_RATE_100HZ,`
	`BMI160_MAG_OUTPUT_DATA_RATE_200HZ,`
	`BMI160_MAG_OUTPUT_DATA_RATE_400HZ,`
	`BMI160_MAG_OUTPUT_DATA_RATE_800HZ`
	`BMI160_MAG_OUTPUT_DATA_RATE_1600HZ`

Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_fifo_gyro_downsampling_enable()

Function Name	`int8_t bmi160_set_fifo_gyro_downsampling_enable(uint8_t v_fifo_down_u8)` `int8_t bmi160_get_fifo_gyro_downsampling_enable(uint8_t* v_fifo_down_u8)`
Function Description	(CONFIG) Sets/gets the gyroscope down-sampling rate for the data stored to the FIFO (gyr_fifo_downs_rate), that is, the rate of samples that are dropped.
Parameters	`v_fifo_down_u8` the value of the gyroscope FIFO down-sampling rate
Return Values	0 for successful execution; ≠0 for error
Notes	FIFO down-sampling rate for the gyroscope is given by $2^{Val(gyr\_fifo\_downs\_rate)}$

bmi160_[set/get]_fifo_gyro_filter_data_enable()

Function Name	`int8_t bmi160_set_fifo_gyro_filter_data_enable(`
	`uint8_t v_gyro_fifo_filter_u8)`
	`int8_t bmi160_get_fifo_gyro_filter_data_enable(`
	`uint8_t* v_gyro_fifo_filter_u8)`

Function Description	(CONFIG) Sets/gets filtered/pre-filtered mode for the gyroscope data stored to the FIFO.
Parameters	`v_gyro_fifo_filter_u8` the value of the gyroscope FIFO data filter mode `FILTER_DATA, UNFILTER_DATA`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_fifo_accel_downsampling_enable()

Function Name	`int8_t bmi160_set_fifo_accel_downsampling_enable(`
	`uint8_t v_fifo_down_u8)`
	`int8_t bmi160_get_fifo_accel_downsampling_enable(`
	`uint8_t* v_fifo_down_u8)`

Function Description	(CONFIG) Sets/gets the accelerometer down-sampling rate for the data stored to the FIFO (acc_fifo_downs_rate), that is, the rate of samples that are dropped.
Parameters	`v_fifo_down_u8` the value of the accelerometer FIFO down-sampling rate
Return Values	0 for successful execution; ≠0 for error
Notes	FIFO down-sampling rate for the accelerometer is given by $2^{Val(acc\_fifo\_downs\_rate)}$

bmi160_[set/get]_fifo_accel_filter_data_enable()

Function Name	`int8_t bmi160_set_fifo_accel_filter_data_enable(`
	`uint8_t v_accel_fifo_filter_u8)`
	`int8_t bmi160_get_fifo_accel_filter_data_enable(`
	`uint8_t* v_accel_fifo_filter_u8)`

Function Description	(CONFIG) Sets/gets filtered/pre-filtered mode for the accelerometer data stored to the FIFO.
Parameters	`v_accel_fifo_filter_u8` the value of the accelerometer FIFO data filter mode `FILTER_DATA, UNFILTER_DATA`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_fifo_watermark_level()

Function Name	`int8_t bmi160_set_fifo_watermark_level(uint8_t v_fifo_wm_u8)` `int8_t bmi160_get_fifo_watermark_level(uint8_t* v_fifo_wm_u8)`
Function Description	(CONFIG) Sets/gets the FIFO watermark level (wm), that is, the level in bytes of the stored data in the FIFO that triggers the respective FIFO watermark interrupt.
Parameters	`v_fifo_wm_u8` the value of the FIFO watermark level (in 4-byte units)
Return Values	0 for successful execution; ≠0 for error
Notes	FIFO watermark level is measured in 4 bytes units.

bmi160_[set/get]_fifo_stop_on_full_enable()

Function Name	`int8_t bmi160_set_fifo_stop_on_full_enable(`
	`uint8_t v_fifo_stop_on_full_u8)`
	`int8_t bmi160_get_fifo_stop_on_full_enable(`
	`uint8_t* v_fifo_stop_on_full_u8)`

Function Description	(CONFIG) Sets/gets the mode of stopping writing samples into FIFO when it is full.
Parameters	`v_fifo_stop_on_full_u8` the value of the FIFO stop-on-full behavior mode `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_fifo_time_enable()

Function Name	`int8_t bmi160_set_fifo_time_enable(uint8_t v_fifo_time_enable_u8)` `int8_t bmi160_get_fifo_time_enable(uint8_t* v_fifo_time_enable_u8)`
Function Description	(CONFIG) Sets/gets the mode of setting a FIFO sensor time frame after the last valid data frame.
Parameters	`v_fifo_time_enable_u8` the value of the FIFO sensor time setting mode `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_fifo_header_enable()

Function Name	`int8_t bmi160_set_fifo_header_enable(uint8_t v_fifo_header_u8)` `int8_t bmi160_get_fifo_header_enable(uint8_t* v_fifo_header_u8)`
Function Description	(CONFIG) Sets/gets the frame header FIFO data storing mode. If disabled, FIFO operated in frame header-less mode.
Parameters	`v_fifo_header_u8` the value of the frame header FIFO data storing mode `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_fifo_mag_enable()

Function Name	`int8_t bmi160_set_fifo_mag_enable(uint8_t v_fifo_mag_u8)` `int8_t bmi160_get_fifo_mag_enable(uint8_t* v_fifo_mag_u8)`
Function Description	(CONFIG) Sets/gets the magnetometer FIFO data storing mode. If enabled, magnetometer xyz-axis and RHALL (r) data are stored to the FIFO.
Parameters	`v_fifo_mag_u8` the value of the magnetometer FIFO data storing mode `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_fifo_accel_enable()

Function Name	`int8_t bmi160_set_fifo_accel_enable(uint8_t v_fifo_accel_u8)` `int8_t bmi160_get_fifo_accel_enable(uint8_t* v_fifo_accel_u8)`
Function Description	(CONFIG) Sets/gets the accelerometer FIFO data storing mode. If enabled, accelerometer xyz-axis data are stored to the FIFO.
Parameters	`v_fifo_accel_u8` the value of the accelerometer FIFO data storing mode `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_fifo_gyro_enable()

Function Name	`int8_t bmi160_set_fifo_gyro_enable(uint8_t v_fifo_gyro_u8)` `int8_t bmi160_get_fifo_gyro_enable(uint8_t* v_fifo_gyro_u8)`
Function Description	(CONFIG) Sets/gets the gyroscope FIFO data storing mode. If enabled, gyroscope xyz-axis data are stored to the FIFO.
Parameters	`v_fifo_gyro_u8` the value of the gyroscope FIFO data storing mode `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_i2c_device_address()

Function Name	`int8_t bmi160_set_i2c_device_address(uint8_t v_i2c_device_addr_u8)` `int8_t bmi160_get_i2c_device_address(uint8_t* v_i2c_device_addr_u8)`
Function Description	(CONFIG) Sets/gets the I2C device address for the connected through the secondary I2C interface BMM150 module and its integrated magnetometer sensor.
Parameters	`v_i2c_device_addr_u8` the value of the magnetometer I2C device address
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_mag_manual_enable()

Function Name	`int8_t bmi160_set_mag_manual_enable(uint8_t v_mag_manual_u8)` `int8_t bmi160_get_mag_manual_enable(uint8_t* v_mag_manual_u8)`
Function Description	(CONFIG) Sets/gets the status of enabling indirect addressing of the connected (BMM150) magnetometer (manual control) through the secondary I2C interface, allowing read and write operations on the magnetometer register map.
Parameters	`v_mag_manual_u8` the value of the manual magnetometer control mode `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	DATA registers are not updated with magnetometer data values. Accessing magnetometer registers requires the magnetometer to be set to normal mode in PMU_STATUS. Magnetometer-interface setup and read loops are exclusive to each other, i.e. during the read loop no registers in the magnetometer may be accessed.

bmi160_[set/get]_mag_read_address()

Function Name	`int8_t bmi160_set_mag_read_address(uint8_t v_mag_read_addr_u8)` `int8_t bmi160_get_mag_read_address(uint8_t* v_mag_read_addr_u8)`
Function Description	(CONFIG) Sets/gets the address to perform an indirect read of a register in the magnetometer-BMM150 register map.
Parameters	`v_mag_read_addr_u8` the value of the magnetometer register address to read
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_mag_write_address()

Function Name	`int8_t bmi160_set_mag_write_address(uint8_t v_mag_write_addr_u8)` `int8_t bmi160_get_mag_write_address(uint8_t* v_mag_write_addr_u8)`
Function Description	(CONFIG) Sets/gets the address to perform an indirect write of a register in the magnetometer-BMM150 register map.
Parameters	`v_mag_write_addr_u8` the value of the magnetometer register address to write
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_mag_write_data()

Function Name	`int8_t bmi160_set_mag_write_data(uint8_t v_mag_write_data_u8)` `int8_t bmi160_get_mag_write_data(uint8_t* v_mag_write_data_u8)`
Function Description	(CONFIG) Sets/gets the data that is written with an indirect write of a register in the magnetometer-BMM150 register map.
Parameters	`v_mag_write_data_u8` the value of the magnetometer register data to write
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_any_motion_[x/y/z]_enable()

Function Name	`int8_t bmi160_set_any_motion_[x/y/z]_enable(uint8_t v_intr_enable_u8)` `int8_t bmi160_get_any_motion_[x/y/z]_enable(uint8_t* v_intr_enable_u8)`
Function Description	(CONFIG) Sets/gets the state of the interrupt of any-motion on x/y/z-axis.
Parameters	`v_intr_enable_u8` the value of any-motion on x/y/z-axis interrupt enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_double_tap_enable()

Function Name	`int8_t bmi160_set_double_tap_enable(uint8_t v_intr_enable_u8)` `int8_t bmi160_get_double_tap_enable(uint8_t* v_intr_enable_u8)`
Function Description	(CONFIG) Sets/gets the state of the double-tap interrupt.
Parameters	`v_intr_enable_u8` the value of the double-tap interrupt enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_single_tap_enable()

Function Name	`int8_t bmi160_set_single_tap_enable(uint8_t v_intr_enable_u8)` `int8_t bmi160_get_single_tap_enable(uint8_t* v_intr_enable_u8)`
Function Description	(CONFIG) Sets/gets the state of the single-tap interrupt.
Parameters	`v_intr_enable_u8` the value of the single-tap interrupt enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_drdy_enable()

Function Name	`int8_t bmi160_set_drdy_enable(uint8_t v_intr_enable_u8)` `int8_t bmi160_get_drdy_enable(uint8_t* v_intr_enable_u8)`
Function Description	(CONFIG) Sets/gets the state of the data ready (drdy) interrupt.
Parameters	`v_intr_enable_u8` the value of the data ready interrupt enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_fifo_full_enable()

Function Name	`int8_t bmi160_set_fifo_full_enable(uint8_t v_intr_enable_u8)` `int8_t bmi160_get_fifo_full_enable(uint8_t* v_intr_enable_u8)`
Function Description	(CONFIG) Sets/gets the state of the FIFO full interrupt.
Parameters	`v_intr_enable_u8` the value of the FIFO full interrupt enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_fifo_watermark_enable()

Function Name	`int8_t bmi160_set_fifo_watermark_enable(uint8_t v_intr_enable_u8)` `int8_t bmi160_get_fifo_watermark_enable(uint8_t* v_intr_enable_u8)`
Function Description	(CONFIG) Sets/gets the state of the FIFO watermark (wm) interrupt.
Parameters	`v_intr_enable_u8` the value of the FIFO watermark interrupt enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_no_motion_[x/y/z]_enable()

Function Name	`int8_t bmi160_set_no_motion_[x/y/z]_enable(uint8_t v_intr_enable_u8)` `int8_t bmi160_get_no_motion_[x/y/z]_enable(uint8_t* v_intr_enable_u8)`
Function Description	(CONFIG) Sets/gets the state of the interrupt of no-motion on x/y/z-axis.
Parameters	`v_intr_enable_u8` the value of no-motion on x/y/z-axis interrupt enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_step_detector_enable()

Function Name	`int8_t bmi160_set_step_detector_enable(uint8_t v_step_intr_u8)` `int8_t bmi160_get_step_detector_enable(uint8_t* v_step_intr_u8)`
Function Description	(CONFIG) Sets/gets the state of the step detection interrupt.
Parameters	`v_intr_enable_u8` the value of the step detection interrupt enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_intr_1_output_edge_level()

Function Name	`int8_t bmi160_set_intr_1_output_edge_level(uint8_t v_intr_edge_ctrl_u8)` `int8_t bmi160_get_intr_1_output_edge_level(uint8_t* v_intr_edge_ctrl_u8)`
Function Description	(CONFIG) Sets/gets the trigger condition, edge or level, for interrupts mapped to pin INT1 of BMI160 sensor module.
Parameters	`v_intr_edge_ctrl_u8` the value of the trigger condition for INT1 `BMI160_LEVEL, BMI160_EDGE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_intr_1_output_level()

Function Name	`int8_t bmi160_set_intr_1_output_level(uint8_t v_intr_level_u8)` `int8_t bmi160_get_intr_1_output_level(uint8_t* v_intr_level_u8)`
Function Description	(CONFIG) Sets/gets the level condition - polarity, high or low, for interrupts mapped to pin INT1 of BMI160 sensor module.
Parameters	`v_intr_level_u8` the value of the level condition - polarity for INT1 `BMI160_LEVEL_LOW, BMI160_LEVEL_HIGH`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_intr_1_output_odrn_pshpll()

Function Name	int8_t bmi160_set_intr_1_output_odrn_pshpll(uint8_t v_intr_output_type_u8) int8_t bmi160_get_intr_1_output_odrn_pshpll(uint8_t* v_intr_output_type_u8)
Function Description	(CONFIG) Sets/gets the type, open-drain or push-pull, for interrupts mapped to pin INT1 of BMI160 sensor module.
Parameters	v_intr_level_u8 the value of the interrupt type for INT1 BMI160_PUSH_PULL, BMI160_OPEN_DRAIN
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_intr_1_output_enable()

Function Name	int8_t bmi160_set_intr_1_output_enable(uint8_t v_output_enable_u8) int8_t bmi160_get_intr_1_output_enable(uint8_t* v_output_enable_u8)
Function Description	(CONFIG) Sets/gets the operation mode of pin INT1 of BMI160 sensor module as output.
Parameters	v_output_enable_u8 the value of the INT1 output mode enable bit BMI160_DISABLE, BMI160_ENABLE
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_intrs_latched()

Function Name	int8_t bmi160_set_intrs_latched(uint8_t v_latch_intr_u8) int8_t bmi160_get_intrs_latched(uint8_t* v_latch_intr_u8)
Function Description	(CONFIG) Sets/gets BMI160 sensor module interrupts' latch duration.
Parameters	v_latch_intr_u8 the value of the interrupt latch duration BMI160_LATCH_DUR_NONE, BMI160_LATCH_DUR_312_5_MICRO_SEC, BMI160_LATCH_DUR_625_MICRO_SEC, BMI160_LATCH_DUR_1_25_MILLI_SEC, BMI160_LATCH_DUR_2_5_MILLI_SEC, BMI160_LATCH_DUR_5_MILLI_SEC, BMI160_LATCH_DUR_10_MILLI_SEC, BMI160_LATCH_DUR_20_MILLI_SEC, BMI160_LATCH_DUR_40_MILLI_SEC, BMI160_LATCH_DUR_80_MILLI_SEC, BMI160_LATCH_DUR_160_MILLI_SEC, BMI160_LATCH_DUR_320_MILLI_SEC, BMI160_LATCH_DUR_640_MILLI_SEC, BMI160_LATCH_DUR_1_28_SEC, BMI160_LATCH_DUR_2_56_SEC, BMI160_LATCHED
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_any_motion_intr_1()

Function Name	`int8_t bmi160_set_any_motion_intr_1(uint8_t v_intr_any_motion_u8)` `int8_t bmi160_get_any_motion_intr_1(uint8_t* v_intr_any_motion_u8)`
Function Description	(CONFIG) Sets/gets the state of mapping any-motion interrupt to BMI160 sensor module pin INT1.
Parameters	`v_intr_any_motion_u8` the value of the any-motion interrupt mapping enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_no_motion_intr_1()

Function Name	`int8_t bmi160_set_no_motion_intr_1(uint8_t v_intr_nomotion_u8)` `int8_t bmi160_get_no_motion_intr_1(uint8_t* v_intr_nomotion_u8)`
Function Description	(CONFIG) Sets/gets the state of mapping no-motion interrupt to BMI160 sensor module pin INT1.
Parameters	`v_intr_nomotion_u8` the value of the no-motion interrupt mapping enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_double_tap_intr_1()

Function Name	`int8_t bmi160_set_double_tap_intr_1(uint8_t v_intr_double_tap_u8)` `int8_t bmi160_get_double_tap_intr_1(uint8_t* v_intr_double_tap_u8)`
Function Description	(CONFIG) Sets/gets the state of mapping double-tap interrupt to BMI160 sensor module pin INT1.
Parameters	`v_intr_double_tap_u8` the value of the double-tap interrupt mapping enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_single_tap_intr_1()

Function Name	`int8_t bmi160_set_single_tap_intr_1(uint8_t v_intr_single_tap_u8)` `int8_t bmi160_get_single_tap_intr_1(uint8_t* v_intr_single_tap_u8)`
Function Description	(CONFIG) Sets/gets the state of mapping single-tap interrupt to BMI160 sensor module pin INT1.
Parameters	`v_intr_single_tap_u8` the value of the single-tap interrupt mapping enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_pmu_trigger_intr_1()

Function Name	`int8_t bmi160_set_pmu_trigger_intr_1(uint8_t v_intr_pmu_trig_u8)` `int8_t bmi160_get_pmu_trigger_intr_1(uint8_t* v_intr_pmu_trig_u8)`
Function Description	(CONFIG) Sets/gets the state of mapping pmu trigger interrupt to BMI160 sensor module pin INT1.
Parameters	`v_intr_pmu_trig_u8` the value of the pmu-trigger interrupt mapping enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_fifo_full_intr_1()

Function Name	`int8_t bmi160_set_fifo_full_intr_1(uint8_t v_intr_fifo_full_u8)` `int8_t bmi160_get_fifo_full_intr_1(uint8_t* v_intr_fifo_full_u8)`
Function Description	(CONFIG) Sets/gets the state of mapping FIFO full interrupt to BMI160 sensor module pin INT1.
Parameters	`v_intr_fifo_full_u8` the value of the FIFO full interrupt mapping enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_fifo_watermark_intr_1()

Function Name	`int8_t bmi160_set_fifo_watermark_intr_1(uint8_t v_intr_fifo_wm_u8)` `int8_t bmi160_get_fifo_watermark_intr_1(uint8_t* v_intr_fifo_wm_u8)`
Function Description	(CONFIG) Sets/gets the state of mapping FIFO watermark (wm) interrupt to BMI160 sensor module pin INT1.
Parameters	`v_intr_fifo_wm_u8` the value of the FIFO watermark interrupt mapping enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_drdy_intr_1()

Function Name	`int8_t bmi160_set_drdy_intr_1(uint8_t v_intr_data_rdy_u8)` `int8_t bmi160_get_drdy_intr_1(uint8_t* v_intr_data_rdy_u8)`
Function Description	(CONFIG) Sets/gets the state of mapping data ready (drdy) interrupt to BMI160 sensor module pin INT1.
Parameters	`v_intr_data_rdy_u8` the value of the drdy interrupt mapping enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

 © 2017 Dialog Semiconductor

bmi160_[set/get]_any_motion_duration()

Function Name	`int8_t bmi160_set_any_motion_duration(uint8_t v_any_motion_durn_u8)` `int8_t bmi160_get_any_motion_duration(uint8_t* v_any_motion_durn_u8)`
Function Description	(CONFIG) Sets/gets the any-motion interrupt trigger delay duration.
Parameters	`v_any_motion_durn_u8` the value of the any-motion interrupt duration
Return Values	0 for successful execution; ≠0 for error
Notes	Any-motion duration can be given by (`v_any_motion_durn_u8` + 1)

bmi160_[set/get]_slow_no_motion_duration()

Function Name	`int8_t bmi160_set_slow_no_motion_duration(uint8_t v_slow_no_motion_u8)` `int8_t bmi160_get_slow_no_motion_duration(uint8_t* v_slow_no_motion_u8)`
Function Description	(CONFIG) Sets/gets the slow/no-motion interrupt trigger delay duration.
Parameters	`v_slow_no_motion_u8` the value of the slow/no-motion interrupt duration
Return Values	0 for successful execution; ≠0 for error
Notes	Slow/no-motion duration can be given by: if `v_slow_no_motion_u8` < 16 → (`v_slow_no_motion_u8`%16 + 1) * 1.28 s (1.28 s to 20.48 s duration range) if 16 ≤ `v_slow_no_motion_u8` < 32 → (`v_slow_no_motion_u8`%16 + 5) * 5.12 s (25.6 s to 102.4 s duration range) if 32 ≤ `v_slow_no_motion_u8` → (`v_slow_no_motion_u8`%32 + 11) * 10.24 s (112.64 s to 430.08 s duration range)

bmi160_[set/get]_any_motion_threshold()

Function Name	`int8_t bmi160_set_any_motion_threshold(uint8_t v_any_motion_thres_u8)` `int8_t bmi160_get_any_motion_threshold(uint8_t* v_any_motion_thres_u8)`
Function Description	(CONFIG) Sets/gets the any-motion interrupt threshold. It is accelerometer range-dependent.
Parameters	`v_any_motion_thres_u8` : the value of the any-motion interrupt threshold
Return Values	0 for successful execution; ≠0 for error
Notes	Any-motion interrupt threshold can be given by: if `v_any_motion_thres_u8` = 0 → threshold is 1.95 mG (2G range), 3.91 mG (4G range), 7.81 mG (8G range), 15.63 mG (16G range) if `v_any_motion_thres_u8` ≠ 0 → threshold is `v_any_motion_thres_u8` * 3.91 mG (2G range), `v_any_motion_thres_u8` * 7.81 mG (4G range), `v_any_motion_thres_u8` * 15.63 mG (8G range), `v_any_motion_thres_u8` * 31.25 mG (16G range)

© 2017 Dialog Semiconductor

bmi160_[set/get]_slow_no_motion_threshold()

Function Name	`int8_t bmi160_set_slow_no_motion_threshold(`
	`uint8_t v_slow_no_motion_thres_u8)`
	`int8_t bmi160_get_slow_no_motion_threshold(`
	`uint8_t* v_slow_no_motion_thres_u8)`

Function Description	(CONFIG) Sets/gets the slow/no-motion interrupt threshold. It is accelerometer range-dependent.
Parameters	`v_slow_no_motion_thres_u8` the value of the slow/no-motion interrupt threshold
Return Values	0 for successful execution; ≠0 for error
Notes	Slow/no-motion interrupt threshold can be given by: if `v_slow_no_motion_thres_u8` = 0 → threshold is 1.95 mG (2G range), 3.91 mG (4G range), 7.81 mG (8G range), 15.63 mG (16G range) if `v_slow_no_motion_thres_u8` ≠ 0 → threshold is v_slow_no_motion_thres_u8 * 3.91 mG (2G range), v_slow_no_motion_thres_u8 * 7.81 mG (4G range), v_slow_no_motion_thres_u8 * 15.63 mG (8G range), v_slow_no_motion_thres_u8 * 31.25 mG (16G range)

bmi160_[set/get]_slow_no_motion_select()

Function Name	`int8_t bmi160_set_slow_no_motion_select(`
	`uint8_t v_intr_slow_no_motion_select_u8)`
	`int8_t bmi160_get_slow_no_motion_select(`
	`uint8_t* v_intr_slow_no_motion_select_u8)`

Function Description	(CONFIG) Sets/gets the slow/no-motion interrupt type, that is, slow or no motion as trigger condition for the slow/no motion interrupt.
Parameters	`v_intr_slow_no_motion_select_u8` the value of the slow/no-motion interrupt type `SLOW_MOTION, NO_MOTION`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_significant_motion_select()

Function Name	`int8_t bmi160_set_significant_motion_select(`
	`uint8_t v_intr_significant_motion_select_u8)`
	`int8_t bmi160_get_significant_motion_select(`
	`uint8_t* v_intr_significant_motion_select_u8)`

Function Description	(CONFIG) Sets/gets the significant/any-motion interrupt type, that is, significant or any-motion as trigger condition for the significant/any-motion interrupt.
Parameters	`v_intr_significant_motion_select_u8` the value of the significant/any-motion interrupt type `ANY_MOTION, SIGNIFICANT_MOTION`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_significant_motion_skip()

Function Name	`int8_t bmi160_set_significant_motion_skip(`
	`uint8_t v_int_sig_mot_skip_u8)`
	`int8_t bmi160_get_significant_motion_skip(`
	`uint8_t* v_int_sig_mot_skip_u8)`

Function Description	(CONFIG) Sets/gets the significant-motion interrupt skip time (t_skip), that is, the time that needs to pass, so that a new detection of significant motion can begin (of t_proof duration) and finally trigger the significant/any-motion interrupt.

Parameters	`v_int_sig_mot_skip_u8` the value of the significant/any-motion interrupt skip time
	`BMI160_SIGNIFICANT_MOTION_SKIP_1_5_SEC,`
	`BMI160_SIGNIFICANT_MOTION_SKIP_3_SEC,`
	`BMI160_SIGNIFICANT_MOTION_SKIP_6_SEC,`
	`BMI160_SIGNIFICANT_MOTION_SKIP_12_SEC`

Return Values	0 for successful execution ; ≠0 for error
Notes	

bmi160_[set/get]_significant_motion_proof()

Function Name	`int8_t bmi160_set_significant_motion_proof(`
	`uint8_t v_significant_motion_proof_u8)`
	`int8_t bmi160_get_significant_motion_proof(`
	`uint8_t* v_significant_motion_proof_u8)`

Function Description	(CONFIG) Sets/gets the significant-motion interrupt proof time (t_proof), that is, the time during which a new detection of significant motion has to happen (after t_skip time of the first detection), so that a significant/any-motion interrupt can be triggered.

Parameters	`v_significant_motion_proof_u8` the value of the significant/any-motion interrupt skip time
	`BMI160_SIGNIFICANT_MOTION_PROOF_0_25_SEC,`
	`BMI160_SIGNIFICANT_MOTION_PROOF_0_5_SEC,`
	`BMI160_SIGNIFICANT_MOTION_PROOF_1_SEC,`
	`BMI160_SIGNIFICANT_MOTION_PROOF_2_SEC`

Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_tap_duration()

Function Name	`int8_t bmi160_set_tap_duration(uint8_t v_tap_durn_u8)` `int8_t bmi160_get_tap_duration(uint8_t* v_tap_durn_u8)`
Function Description	(CONFIG) Sets/gets the length of the time window for the second shock event to happen in the case of double tap detection, triggering the double-tap interrupt.

Parameters	`v_tap_durn_u8` the value of the tap duration
	`BMI160_TAP_DURN_50MS , BMI160_TAP_DURN_100MS,`
	`BMI160_TAP_DURN_150MS, BMI160_TAP_DURN_200MS,`
	`BMI160_TAP_DURN_250MS, BMI160_TAP_DURN_375MS,`
	`BMI160_TAP_DURN_700MS`

Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_tap_shock_duration()

Function Name	`int8_t bmi160_set_tap_shock_duration(uint8_t v_tap_shock_u8)` `int8_t bmi160_get_tap_shock_duration(uint8_t* v_tap_shock_u8)`
Function Description	(CONFIG) Sets/gets the tap interrupt shock duration.
Parameters	`v_tap_shock_u8` the value of the tap shock duration `BMI160_TAP_SHOCK_50MS, BMI160_TAP_SHOCK_75MS`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_tap_quiet_duration()

Function Name	`int8_t bmi160_set_tap_quiet_duration(uint8_t v_tap_quiet_u8)` `int8_t bmi160_get_tap_quiet_duration(uint8_t* v_tap_quiet_u8)`
Function Description	(CONFIG) Sets/gets the tap interrupt quiet duration.
Parameters	`v_tap_shock_u8` the value of the tap quiet duration `BMI160_TAP_QUIET_30MS, BMI160_TAP_QUIET_20MS`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_tap_threshold()

Function Name	`int8_t bmi160_set_tap_threshold(uint8_t v_tap_thres_u8)` `int8_t bmi160_get_tap_threshold(uint8_t* v_tap_thres_u8)`
Function Description	(CONFIG) Sets/gets the single/double-tap interrupt threshold. It is accelerometer range-dependent.
Parameters	`v_tap_thres_u8` the value of the single/double-tap interrupt threshold
Return Values	0 for successful execution; ≠0 for error
Notes	Single/double-tap interrupt threshold can be given by: if `v_tap_thres_u8` = 0 → threshold is 31.25 mG (2G range), 62.5 mG (4G range), 125 mG (8G range), 250 mG (16G range) if `v_tap_thres_u8` ≠ 0 → threshold is (`v_tap_thres_u8` + 1) * 62.5 mG (2G range), (`v_tap_thres_u8` + 1) * 125 mG (4G range), (`v_tap_thres_u8` + 1) * 250 mG (8G range), (`v_tap_thres_u8` + 1) * 500 mG (16G range)

© 2017 Dialog Semiconductor

bmi160_[set/get]_if_mode()

Function Name	`int8_t bmi160_set_if_mode(uint8_t v_if_mode_u8)` `int8_t bmi160_get_if_mode(uint8_t* v_if_mode_u8)`
Function Description	(CONFIG) Sets/gets the digital interface (if) configuration mode.
Parameters	`v_if_mode_u8` the value of the interface configuration mode
Return Values	0 for successful execution; ≠0 for error
Notes	Interface configuration mode options (in `v_if_mode_u8`) are: 0x00 → Primary interface:autoconfig / secondary interface:off 0x01 → Primary interface:I2C / secondary interface:OIS 0x02 → Primary interface:autoconfig / secondary interface:Magnetometer 0x03 → Reserved

bmi160_[set/get]_gyro_sleep_trigger()

Function Name	`int8_t bmi160_set_gyro_sleep_trigger(uint8_t v_gyro_sleep_trigger_u8)` `int8_t bmi160_get_gyro_sleep_trigger(uint8_t* v_gyro_sleep_trigger_u8)`
Function Description	(CONFIG) Sets/gets the trigger condition for the gyroscope transition to sleep power mode.

Parameters	`v_gyro_sleep_trigger_u8` the value of the gyroscope sleep transition trigger condition
	`BMI160_GYRO_SLEEP_TRIGGER_DISABLE,`
	`BMI160_GYRO_SLEEP_TRIGGER_INT2,`
	`BMI160_GYRO_SLEEP_TRIGGER_NOTINT1,`
	`BMI160_GYRO_SLEEP_TRIGGER_NOTINT1_OR_INT2,`
	`BMI160_GYRO_SLEEP_TRIGGER_NOMOTION,`
	`BMI160_GYRO_SLEEP_TRIGGER_ANYMOTION_OR_INT2,`
	`BMI160_GYRO_SLEEP_TRIGGER_ANYMOTION_OR_NOTINT1,`
	`BMI160_GYRO_SLEEP_TRIGGER_ANYMOTION_OR_NOTINT1_OR_INT2`

Return Values	0 for successful execution; ≠0 for error
Notes	If a gyroscope wakeup condition (see `bmi160_set_gyro_wakeup_trigger()`) holds at the same time, then sleep transition is cancelled.

bmi160_[set/get]_gyro_wakeup_trigger()

Function Name	`int8_t bmi160_set_gyro_wakeup_trigger(uint8_t v_gyro_wakeup_trigger_u8)` `int8_t bmi160_get_gyro_wakeup_trigger(uint8_t* v_gyro_wakeup_trigger_u8)`
Function Description	(CONFIG) Sets/gets the trigger condition for the gyroscope transition to normal power mode.

Parameters	`v_gyro_wakeup_trigger_u8` the value of the gyroscope wakeup transition trigger condition
	`BMI160_GYRO_WAKEUP_TRIGGER_DISABLE,`
	`BMI160_GYRO_WAKEUP_TRIGGER_INT1,`
	`BMI160_GYRO_WAKEUP_TRIGGER_ANYMOTION,`
	`BMI160_GYRO_WAKEUP_TRIGGER_ANYMOTION_AND_INT1`

Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_gyro_sleep_trigger_state()

Function Name	`int8_t bmi160_set_gyro_sleep_trigger_state(`
	`uint8_t v_gyro_sleep_state_u8)`
	`int8_t bmi160_get_gyro_sleep_trigger_state(`
	`uint8_t* v_gyro_sleep_state_u8)`

Function Description	(CONFIG) Sets/gets the sleep state of the gyroscope when a sleep transition is performed, i.e. `GYR_SUSPEND` or `GYR_FASTSTARTUP`.
Parameters	`v_gyro_sleep_state_u8` the value of the gyroscope sleep state on sleep transition `BMI160_GYRO_SLEEP_TRIGGER_STATE_FAST_STARTUP`, `BMI160_GYRO_SLEEP_TRIGGER_STATE_SUSPEND`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_gyro_wakeup_enable()

Function Name	`int8_t bmi160_set_gyro_wakeup_enable(uint8_t v_gyro_wakeup_intr_u8)` `int8_t bmi160_get_gyro_wakeup_enable(uint8_t* v_gyro_wakeup_intr_u8)`
Function Description	(CONFIG) Sets/gets the state of the gyro wakeup interrupt (triggering PMU trigger interrupt).
Parameters	`v_gyro_wakeup_intr_u8` the value of the gyro wakeup interrupt enable bit `BMI160_DISABLE`, `BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_step_min_steptime()

Function Name	`int8_t bmi160_set_step_min_steptime(uint8_t v_step_min_steptime_u8)` `int8_t bmi160_get_step_min_steptime(uint8_t* v_step_min_steptime_u8)`
Function Description	(CONFIG) Sets/gets the minimal detectable step time with offset 1.
Parameters	`v_step_min_steptime_u8` the value of the min detectable step time (range 0 to 7)
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_step_min_threshold()

Function Name	`int8_t bmi160_set_step_min_threshold(uint8_t v_step_min_threshold_u8)` `int8_t bmi160_get_step_min_threshold(uint8_t* v_step_min_threshold_u8)`
Function Description	(CONFIG) Sets/gets the minimum threshold value allowing the detection of a step.
Parameters	`v_step_min_threshold_u8` the value of the minimum step threshold (range 0 to 3)
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_step_alpha()

Function Name	`int8_t bmi160_set_step_alpha(uint8_t v_step_alpha_u8)` `int8_t bmi160_get_step_alpha(uint8_t* v_step_alpha_u8)`
Function Description	(CONFIG) Sets/gets the alpha parameter for step detection.
Parameters	`v_step_alpha_u8` the value of the alpha parameter (range 0 to 7)
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_step_min_buffer_size()

Function Name	`int8_t bmi160_set_step_min_buffer_size(`
	`uint8_t v_step_min_buffer_size_u8)`
	`int8_t bmi160_get_step_min_buffer_size(`
	`uint8_t* v_step_min_buffer_size_u8)`

Function Description	(CONFIG) Sets/gets the minimum step buffer size, that is, the number of steps to suppress false positives at the beginning of the walking activity. After walking activity has been detected, these steps will be added to the total number of steps counted.
Parameters	`v_step_min_buffer_size_u8` the value of the minimum step buffer size
Return Values	0 for successful execution; ≠0 for error
Notes	As a consequence the step counter output is suppressed during the first steps of the walking activity.

bmi160_[set/get]_step_counter_enable()

Function Name	`int8_t bmi160_set_step_counter_enable(uint8_t v_step_counter_u8)` `int8_t bmi160_get_step_counter_enable(uint8_t* v_step_counter_u8)`
Function Description	(CONFIG) Sets/gets the state of the step counter.
Parameters	`v_step_counter_u8` the value of the step counter enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_target_page()

Function Name	`int8_t bmi160_set_target_page(uint8_t v_target_page_u8)` `int8_t bmi160_get_target_page(uint8_t* v_target_page_u8)`
Function Description	(CONFIG) Sets/gets the target page.

Parameters	`v_target_page_u8` the value of the target page
	`BMI160_TARGET_PAGE_USER_DATA_CONFIGURE,`
	`BMI160_TARGET_PAGE_CHIP_LEVEL_TRIM_TEST`

Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_[set/get]_paging_enable()

Function Name	`int8_t bmi160_set_paging_enable(uint8_t v_page_enable_u8)` `int8_t bmi160_get_paging_enable(uint8_t* v_page_enable_u8)`
Function Description	(CONFIG) Sets/gets the state of paging.
Parameters	`v_page_enable_u8` the value of the paging enable bit `BMI160_DISABLE, BMI160_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_[set/get]_pullup_configuration()

Function Name	`int8_t bmi160_set_pullup_configuration(uint8_t v_control_pullup_u8)` `int8_t bmi160_get_pullup_configuration(uint8_t* v_control_pullup_u8)`
Function Description	(CONFIG) Sets/gets the pullup configuration.
Parameters	`v_control_pullup_u8` the value of the pullup configuration `BMI160_DISABLE = 0x00, BMI160_ENABLE = 0x03`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

10.1.1.8 Low Level Register Access Functions

bmi160_[set/get]_reg_[reg_name]()

Function Name	int8_t bmi160_set_reg_cmd(uint8_t v_reg_value_u8) int8_t bmi160_get_reg_cmd(uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_step_conf_[0/1](uint8_t v_reg_value_u8) int8_t bmi160_get_reg_step_conf_[0/1](uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_pmu_trigger(uint8_t v_reg_value_u8) int8_t bmi160_get_reg_pmu_trigger(uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_int_tap_[0/1](uint8_t v_reg_value_u8) int8_t bmi160_get_reg_int_tap_[0/1](uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_int_motion_[0/1/2/3](uint8_t v_reg_value_u8) int8_t bmi160_get_reg_int_motion_[0/1/2/3](uint8_t*v_reg_value_u8) int8_t bmi160_set_reg_int_map_[0/1/2](uint8_t v_reg_value_u8) int8_t bmi160_get_reg_int_map_[0/1/2](uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_int_latch(uint8_t v_reg_value_u8) int8_t bmi160_get_reg_int_latch(uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_int_out_ctrl(uint8_t v_reg_value_u8) int8_t bmi160_get_reg_int_out_ctrl(uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_int_en_[0/1/2](uint8_t v_reg_value_u8) int8_t bmi160_get_reg_int_en_[0/1/2](uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_mag_if_[0/1/2/3/4](uint8_t v_reg_value_u8) int8_t bmi160_get_reg_mag_if_[0/1/2/3/4](uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_fifo_config_[0/1](uint8_t v_reg_value_u8) int8_t bmi160_get_reg_fifo_config_[0/1](uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_mag_conf(uint8_t v_reg_value_u8) int8_t bmi160_get_reg_mag_conf(uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_gyr_range(uint8_t v_reg_value_u8) int8_t bmi160_get_reg_gyr_range(uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_gyr_conf(uint8_t v_reg_value_u8) int8_t bmi160_get_reg_gyr_conf(uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_acc_range(uint8_t v_reg_value_u8) int8_t bmi160_get_reg_acc_range(uint8_t *v_reg_value_u8) int8_t bmi160_set_reg_acc_conf(uint8_t v_reg_value_u8) int8_t bmi160_get_reg_acc_conf(uint8_t *v_reg_value_u8) int8_t bmi160_get_reg_status(uint8_t *v_reg_value_u8) int8_t bmi160_get_reg_int_status_0(uint8_t *v_reg_value_u8) int8_t bmi160_get_reg_int_status_1(uint8_t *v_reg_value_u8) int8_t bmi160_get_reg_int_status_2(uint8_t *v_reg_value_u8) int8_t bmi160_get_reg_int_status_3(uint8_t *v_reg_value_u8)
Function Description	(LOW-ACCESS) Sets/gets the register [reg_name] value based on the register names presented in BMI160 datasheet Register Map
Parameters	v_reg_value_u8 the value of the register [reg_name]
Return Values	0 for successful execution; ≠0 for error
Notes	Direct write- or read-access to BMI160 registers, without any further processing (i.e. bit- masking and shifting) or delay (if any)

© 2017 Dialog Semiconductor

10.1.1.9 Magnetometer Control and Configuration Functions (Indirect Access)

bmi160_init_bmm150_mag_interface()

Function Name	`int8_t bmi160_init_bmm150_mag_interface(void)`
Function Description	(INIT-CONFIG) Initializes the I2C interface between BMI160 and BMM150 sensor modules, setting the I2C address, reading the magnetometer trim values, setting the magnetometer to preset mode `BMI160_MAG_PRESETMODE_REGULAR` (i.e. 9 and 15 repetitions for x/y-axis and z-axis, respectively) and to SUSPEND power mode state.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_get_bmm150_mag_interface()

Function Name	`int8_t bmi160_get_bmm150_mag_interface(uint8_t* v_mag_interface_u8)`
Function Description	(CONFIG) Gets the type of interface established between BMI160 and BMM150 sensor modules.
Parameters	`v_mag_interface_u8` the value of the interface type `BMI160_BMM150_INTERFACE_NONE`, `BMI160_BMM150_INTERFACE_I2C`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_wakeup_bmm150_mag()

Function Name	`int8_t bmi160_wakeup_bmm150_mag(void)`
Function Description	(INIT-CONFIG) Wakes up the BMM150 sensor module (putting magnetometer to SLEEP power mode state).
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	After some maximum number of retries (`BMM150_MAX_RETRY_WAKEUP`) to power-on the device, it fails, clearing I2C address (0x00).

bmi160_suspend_bmm150_mag()

Function Name	`int8_t bmi160_suspend_bmm150_mag(void)`
Function Description	(INIT-CONFIG) Suspends the BMM150 sensor module (putting magnetometer to SUSPEND power mode state).
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmi160_init_bmm150_mag_trim_registers()

Function Name	`int8_t bmi160_init_bmm150_mag_trim_registers(void)`
Function Description	(INIIT-CONFIG) Reads trim values of the BMM150 magnetometer and stores them to the instance handle of the BMI160 module.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

bmi160_set_bmm150_mag_preset_mode()

Function Name	`int8_t bmi160_set_bmm150_mag_preset_mode(uint8_t v_preset_mode_u8)`
Function Description	(INIT-CONFIG) Sets a predefined operation mode for the BMM150 sensor module.

Parameters	`v_preset_mode_u8` the value of the magnetometer preset mode
	`BMI160_MAG_PRESETMODE_LOWPOWER,`
	`BMI160_MAG_PRESETMODE_REGULAR,`
	`BMI160_MAG_PRESETMODE_HIGHACCURACY,`
	`BMI160_MAG_PRESETMODE_ENHANCED`

Return Values	0 for successful execution; ≠0 for error
Notes	The performed number of repetitions for each preset-mode are: `BMI160_MAG_PRESETMODE_LOWPOWER` → num. of xy-repetitions = 3 num. of z-repetitions = 3 `BMI160_MAG_PRESETMODE_REGULAR` → num. of xy-repetitions = 9 num. of z-repetitions = 15 `BMI160_MAG_PRESETMODE_HIGHACCURACY` → num. of xy-repetitions = 47 num. of z-repetitions = 83 `BMI160_MAG_PRESETMODE_ENHANCED` → num. of xy-repetitions = 15 num. of z-repetitions = 27

© 2017 Dialog Semiconductor

10.1.2 BMM150 Sensor Driver

The BMM150 sensor driver API provides functions for the configuration, control and sensor data acquisition from the integrated by the corresponding module magnetometer sensor. In the Wearable DK platform the latter is controlled by the BMI160 via a secondary I2C interface, as described earlier. Therefore, the appropriate control and configuration of the magnetometer for the support of the respective Wearable DK application operational features is done using special functions included in the BMI160 driver API (see Section 0), implementing an indirect register access mechanism.

This basic set of interface functions for the control and configuration of the magnetometer may be extended with functions provided by the BMM150 driver API, applying the implemented indirect register access mechanism for each of those functions. Below a part of the BMM150 driver API functions is presented, while a complete list of the functions is presented in file `bmm150.h`, based on the datasheet [11] provided by Bosch Sensortec.

General guidelines for using the BMM150 sensor driver are as follows:

- Initialize sensor module using the function `bmm150_initialize_sensor()`. The device address for the sensor module must be set, along with pointers to functions which implement SPI/I2C bus access and introduce delay.
- Set the power mode state of the module by setting the power mode state of the magnetometer sensor using the function `bmm150_set_sensor_state()`.
- Configure the sensor module to operate in a predefined mode using the function `bmm150_config_preset_mode()`.
- Configure sensor module in terms of specific setting/operation features (`setting_feature = mag_out_data_rate`, `mag_repetion_xy`, `mag_repetition_z`, etc.) using the `bmm150_set_[setting_feature]()` set of functions.
- If output data rates for the magnetometer data higher that 30 Hz need to be achieved (maximum for NORMAL power mode), then periodically change the power mode state of the magnetometer to FORCED, based on the particular rate.
- When the current configuration state of the sensor module is to be checked in terms of specific setting/operation features, use the `bmm150_get_[setting_feature]()` set of functions.
- Make appropriate decisions by getting the status of the sensor module (`state_feature = drdy_mag_stat`, `low_thres_z_stat`, etc.) using the `bmm150_get_[state_feature]()` set of functions.
- Access the sensor module's registers (`reg_name`) when low level control and data handling is required, using the `bmm150_[set/get]_reg_[reg_name]()` set of functions.
- When the power mode state of the module is SUSPEND (or POWER_OFF), no register can be accessed, except for the register which defines the power mode state. To perform a register access or generally use a function to configure or read data from the BMM150 sensor module, first change the power mode state to SLEEP.
- Read magnetometer sensor data using the `bmm150_read_[sensor]_[x/y/z/r]()` set of functions.
- Use the `bmm150_compensate_mag_[x/y/z]()` set of functions in order to convert uncompensated magnetometer xyz-axis values to true ones.

A list of all API data structures/types and functions provided by the BMM150 sensor driver and presented in this section follows.

Table 35: BMM150 Driver API

Data Structures and Types	
bmm150_t	bmm150_mag_data_s16_t
Dependency Functions	
bmm150_t -> bus_write() bmm150_t -> delay_msec()	bmm150_t -> bus_read()

 © 2017 Dialog Semiconductor

Initialization and Basic Configuration Functions	
bmm150_initialize_sensor() bmm150_config_preset_mode()	bmm150_set_sensor_state() bmi160_set_sensor_state()
Sensor Data Acquisition Functions	
bmm150_read_mag_xyz() bmm150_compensate_mag_[x/y/z]()	bmm150_read_mag_r()
Command Functions	
bmm150_soft_reset()	
State Functions	
bmm150_get_chip_id() bmm150_get_low_thres_[x/y/z]_stat()	bmm150_get_drdy_mag_stat() bmm150_get_high_thres_[x/y/z]_stat()
Configuration Functions	
bmm150_[set/get]_mag_out_data_rate() bmm150_[set/get]_high_thres_[x/y/z]_enable() bmm150_[set/get]_int_latched() bmm150_[set/get]_channel_[x/y/z]_enable() bmm150_[set/get]_drdy_enable() bmm150_[set/get]_mag_high_thres() bmm150_[set/get]_mag_repetition_z()	bmm150_[set/get]_low_thres_[x/y/z]_enable() bmm150_[set/get]_int_polarity() bmm150_[set/get]_drdy_polarity() bmm150_[set/get]_int_enable() bmm150_[set/get]_mag_low_thres() bmm150_[set/get]_mag_repetition_xy()bmm150_[set/get]_mag_repetition_xy()
Low Level Register Access Functions	
bmm150_[set/get]_reg_[reg_name]()	

10.1.2.1 Data Structures and Types

bmm150_t

Data Structure Fields	Type	Description
company_id	uint8_t	Chip ID of the BMM150 sensor module.
dev_addr	uint8_t	Device address of the BMM150 sensor module.
bus_write	Function pointer	Function for SPI/I2C bus write-access.
bus_read	Function pointer	Function for SPI/I2C bus read-access.
delay_msec	Function pointer	Function for introducing delay (in ms).

bmm150_mag_data_s16_t

Data Structure Fields	Type	Description
datax	int16_t	Magnetometer x-axis raw data value.
datay	int16_t	Magnetometer y-axis raw data value.
dataz	int16_t	Magnetometer z-axis raw data value.
resistance	uint16_t	Internal resistance raw data value.

© 2017 Dialog Semiconductor

10.1.2.2 Dependency Functions

bmm150_t -> bus_write()

Function Name	`int8_t (*bus_write)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data, uint8_t cnt)`
Function Description	Implements write-access over the SPI / I2C bus for writing data to BMM150 sensor module registers.
Parameters	`dev_addr` the device address `reg_addr` the address of the register to write `reg_data` the data to write `cnt` the size of data (in bytes)
Return Values	0 for successful execution; ≠0 for error
Notes	

bmm150_t -> bus_read()

Function Name	`int8_t (*bus_read)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data, uint8_t cnt)`
Function Description	Implements read-access over the SPI / I2C bus for reading data from BMM150 sensor module registers.
Parameters	`dev_addr` the device address `reg_addr` the address of the register to read `reg_data` the buffer to put register data being read to `cnt` the size of data (in bytes)
Return Values	0 for successful execution; ≠0 for error
Notes	

bmm150_t -> delay_msec()

Function Name	`void (*delay_msec)(uint32_t ms)`
Function Description	Implements a delay in ms that may be necessary for the BMM150 sensor module driver functionality implementation.
Parameters	`ms` the delay (in ms) to be introduced
Return Values	None
Notes	

10.1.2.3 Initialization and Basic Configuration Functions

bmm150_initialize_sensor()

Function Name	`int8_t bmm150_initialize_sensor(bmm150_t* s_bmm150)`
Function Description	(INIT-CONFIG) Initializes the BMM150 sensor module, creating the appropriate instance handle for the module which includes references to the SPI / I2C bus access functions used for reading/writing values to the registers, setting SLEEP power mode state for the magnetometer, getting its trim values and soft-resetting the sensor module.
Parameters	`s_bmm150` the handle structure for BMM150 driver
Return Values	0 for successful execution; ≠0 for error
Notes	

 © 2017 Dialog Semiconductor

bmm150_set_sensor_state()

Function Name	`int8_t bmm150_set_sensor_state(uint8_t state)`
Function Description	(INIT-CONFIG) Sets the state of the magnetometer.
Parameters	`state` the state to set for a sensor `MAG_NORMAL, MAG_FORCED, MAG_SUSPEND, MAG_SLEEP`
Return Values	0 for successful execution; ≠0 for error
Notes	On successful execution of the corresponding register access it is introduced a delay of `[ACC/GYR/MAG_SLV]_MODE_TRANSITION_DELAY`.

bmm150_config_preset_mode()

Function Name	`int8_t bmm150_comfig_preset_mode(uint8_t v_preset_mode_u8)`
Function Description	(INIT-CONFIG) Sets a predefined operation mode for the BMM150 sensor module.

Parameters	`v_preset_mode_u8` the value of the magnetometer preset mode
	`MAG_PRESETMODE_LOWPOWER , MAG_PRESETMODE_REGULAR,`
	`MAG_PRESETMODE_HIGHACCURACY, MAG_PRESETMODE_ENHANCED`

Return Values	0 for successful execution; ≠0 for error
Notes	The performed number of repetitions for each preset-mode are: `MAG_PRESETMODE_LOWPOWER` → num. of xy-repetitions = 3 num. of z-repetitions = 3 output data rate = 10 Hz `MAG_PRESETMODE_REGULAR` → num. of xy-repetitions = 9 num. of z-repetitions = 15 output data rate = 10 Hz `MAG_PRESETMODE_HIGHACCURACY` → num. of xy-repetitions = 47 num. of z-repetitions = 83 output data rate = 20 Hz `MAG_PRESETMODE_ENHANCED` → num. of xy-repetitions = 15 num. of z-repetitions = 27 output data rate = 10 Hz

bmm150_init_trim_registers()

Function Name	`int8_t bmm150_init_trim_registers(void)`
Function Description	(INIT-CONFIG) Reads trim values of the BMM150 magnetometer and stores them to the instance handle of the module.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

10.1.2.4 Sensor Data Acquisition Functions

bmm150_read_mag_xyz()

Function Name	`int8_t bmm150_read_mag_xyz(bmm150_mag_data_s16_t *mag_xyz)`
Function Description	(SAMPLE) Reads magnetometer xyz-axis data.
Parameters	`mag_xyz` the magnetometer xyz-axis data
Return Values	0 for successful execution; ≠0 for error
Notes	Magnetometer axis sensor data values are measured in μT (~0.3 μT resolution).

© 2017 Dialog Semiconductor

bmm150_read_mag_r()

Function Name	`int8_t bmm150_read_mag_r(uint16_t *v_mag_r_u16)`
Function Description	(SAMPLE) Reads magnetometer RHALL (r) data.
Parameters	`v_mag_r_u16` the magnetometer RHALL (r) data
Return Values	0 for successful execution; ≠0 for error
Notes	

bmm150_compensate_mag_[x/y/z]()

Function Name	`int16_t bmm150_compensate_mag_x(int16_t v_mag_x_s16, uint16_t v_mag_r_u16)`
	`int16_t bmm150_compensate_mag_y(int16_t v_mag_y_s16, uint16_t v_mag_r_u16)`
	`int16_t bmm150_compensate_mag_z(int16_t v_mag_z_s16, uint16_t v_mag_r_u16)`

Function Description	(SAMPLE) Converts uncompensated magnetometer x/y/z-axis data values to true ones.
Parameters	`v_mag_[x/y/z]_s16` the magnetometer x/y/z-axis data value `v_mag_r_u16` the RHALL (r) data value
Return Values	The true-compensated magnetometer x/y/z-axis value
Notes	

10.1.2.5 Command Functions

bmm150_soft_reset()

Function Name	`int8_t bmm150_soft_reset(void)`
Function Description	(CMD) Performs a soft reset to the BMM150 sensor module.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	On successful execution it is introduced a `BMM050_DELAY_SOFTRESET` delay.

10.1.2.6 State Functions

bmm150_get_chip_id()

Function Name	`int8_t bmm150_get_chip_id(uint8_t* v_chip_id_u8)`
Function Description	(STATE) Reads the chip id of the BMM150 sensor module.
Parameters	`v_chip_id_u8` the chip id
Return Values	0 for successful execution; ≠0 for error
Notes	

bmm150_get_drdy_mag_stat()

Function Name	`int8_t bmm150_get_drdy_mag_stat(uint8_t* v_data_rdy_u8)`
Function Description	(STATE) Reads the status of data ready (drdy) status bit, indicating whether a new magnetometer sample value has been set to the corresponding data register.
Parameters	`v_data_rdy_u8` the value of the magnetometer drdy interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmm150_get_low_thres_[x/y/z]_stat()

Function Name	`int8_t bmm150_get_low_thres_x_stat(uint8_t* v_low_thres_u8)`
	`int8_t bmm150_get_low_thres_y_stat(uint8_t* v_low_thres_u8)`
	`int8_t bmm150_get_low_thres_z_stat(uint8_t* v_low_thres_u8)`

Function Description	(STATE) Reads the status of low threshold magnetometer data on [x/y/z]-axis status bit, indicating whether a magnetometer sample value has been less than the defined low threshold (see `bmm150_set_mag_low_thres()`).
Parameters	`v_low_thres_u8` the value of the magnetometer low threshold on [x/y/z]-axis interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

bmm150_get_high_thres_[x/y/z]_stat()

Function Name	`int8_t bmm150_get_high_thres_x_stat(uint8_t* v_low_thres_u8)`
	`int8_t bmm150_get_high_thres_y_stat(uint8_t* v_low_thres_u8)`
	`int8_t bmm150_get_high_thres_z_stat(uint8_t* v_low_thres_u8)`

Function Description	(STATE) Reads the status of high threshold magnetometer data on [x/y/z]-axis status bit, indicating whether a magnetometer sample value has been more than the defined high threshold (see `bmm150_set_mag_high_thres()`).
Parameters	`v_high_thres_u8` the value of the magnetometer high threshold on [x/y/z]-axis interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

10.1.2.7 Configuration Functions

bmm150_[set/get]_mag_out_data_rate()

Function Name	`int8_t bmm150_set_mag_out_data_rate(uint8_t v_output_data_rate_u8)` `int8_t bmm150_get_mag_out_data_rate(uint8_t* v_output_data_rate_u8)`
Function Description	(CONFIG) Sets/gets the magnetometer output data rate (odr), that is, the sampling data rate.

Parameters	`v_output_data_rate_u8` the value of the magnetometer output data rate
	`BMM050_DATA_RATE_10HZ, BMM050_DATA_RATE_02HZ,`
	`BMM050_DATA_RATE_06HZ, BMM050_DATA_RATE_08HZ,`
	`BMM050_DATA_RATE_15HZ, BMM050_DATA_RATE_20HZ,`
	`BMM050_DATA_RATE_25HZ, BMM050_DATA_RATE_30HZ`

Return Values	0 for successful execution; ≠0 for error
Notes	The above output data rates refer to NORMAL power state mode of the magnetometer. In order to be achieved higher or generally different output data rates, the magnetometer must be periodically changed to FORCED power state mode. (i.e. `bmm150_set_sensor_state(MAG_FORCED)`)

© 2017 Dialog Semiconductor

bmm150_[set/get]_low_thres_[x/y/z]_enable()

Function Name	`int8_t bmm150_set_low_thres_x_enable(uint8_t v_intr_enable_u8)`
	`int8_t bmm150_get_low_thres_x_enable(uint8_t* v_intr_enable_u8)`
	`int8_t bmm150_set_low_thres_y_enable(uint8_t v_intr_enable_u8)`
	`int8_t bmm150_get_low_thres_y_enable(uint8_t* v_intr_enable_u8)`
	`int8_t bmm150_set_low_thres_z_enable(uint8_t v_intr_enable_u8)`
	`int8_t bmm150_get_low_thres_z_enable(uint8_t* v_intr_enable_u8)`

Function Description	(CONFIG) Sets/gets the state of the interrupt of low threshold magnetometer sensor data on [x/y/z]-axis.
Parameters	`v_intr_enable_u8` the value of the low threshold on [x/y/z]-axis interrupt enable bit `BMM050_DISABLE, BMM050_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmm150_[set/get]_high_thres_[x/y/z]_enable()

Function Name	`int8_t bmm150_set_high_thres_x_enable(uint8_t v_intr_enable_u8)`
	`int8_t bmm150_get_high_thres_x_enable(uint8_t* v_intr_enable_u8)`
	`int8_t bmm150_set_high_thres_y_enable(uint8_t v_intr_enable_u8)`
	`int8_t bmm150_get_high_thres_y_enable(uint8_t* v_intr_enable_u8)`
	`int8_t bmm150_set_high_thres_z_enable(uint8_t v_intr_enable_u8)`
	`int8_t bmm150_get_high_thres_z_enable(uint8_t* v_intr_enable_u8)`

Function Description	(CONFIG) Sets/gets the state of the interrupt of high threshold magnetometer sensor data on [x/y/z]-axis.
Parameters	`v_intr_enable_u8` the value of the high threshold on [x/y/z]-axis interrupt enable bit `BMM050_DISABLE, BMM050_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmm150_[set/get]_int_polarity()

Function Name	`int8_t bmm150_set_int_polarity(uint8_t v_intr_polarity_u8)` `int8_t bmm150_get_int_polarity(uint8_t* v_intr_polarity_u8)`
Function Description	(CONFIG) Sets/gets the INT interrupt pin polarity.
Parameters	`v_intr_polarity_u8` the value of the INT pin polarity `BMM050_POLARITY_LOW, BMM050_POLARITY_HIGH`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmm150_[set/get]_int_latched()

Function Name	`int8_t bmm150_set_int_latched(uint8_t v_intr_latch_u8)` `int8_t bmm150_get_int_latched(uint8_t* v_intr_latch_u8)`
Function Description	(CONFIG) Sets/gets the state of the INT interrupt pin as latched.
Parameters	`v_intr_latch_u8` the value of the interrupt latching for the INT pin `BMM050_DISABLE, BMM050_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmm150_[set/get]_drdy_polarity()

Function Name	`int8_t bmm150_set_drdy_polarity(uint8_t v_intr_polarity_u8)` `int8_t bmm150_get_drdy_polarity(uint8_t* v_intr_polarity_u8)`
Function Description	(CONFIG) Sets/gets the DRDY interrupt pin polarity.
Parameters	`v_intr_polarity_u8` the value of the DRDY pin polarity `BMM050_POLARITY_LOW, BMM050_POLARITY_HIGH`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmm150_[set/get]_channel_[x/y/z]_enable()

Function Name	`int8_t bmm150_set_channel_x_enable(uint8_t v_meas_enable_u8)`
	`int8_t bmm150_get_channel_x_enable(uint8_t* v_meas_enable_u8)`
	`int8_t bmm150_set_channel_y_enable(uint8_t v_meas_enable_u8)`
	`int8_t bmm150_get_channel_y_enable(uint8_t* v_meas_enable_u8)`
	`int8_t bmm150_set_channel_z_enable(uint8_t v_meas_enable_u8)`
	`int8_t bmm150_get_channel_z_enable(uint8_t* v_meas_enable_u8)`

Function Description	(CONFIG) Sets/gets the state of [x/y/z]-axis measurement channel.
Parameters	`v_meas_enable_u8` the value of the [x/y/z]-axis measurement channel state `BMM050_DISABLE, BMM050_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	If z-axis measurement channel is enabled, then it is also enabled the resistance measurement.

bmm150_[set/get]_int_enable()

Function Name	`int8_t bmm150_set_int_enable(uint8_t v_intr_enable_u8)` `int8_t bmm150_get_int_enable(uint8_t* v_intr_enable_u8)`
Function Description	(CONFIG) Sets/gets the state of mapping interrupts' status to the INT pin of the BMM150 sensor module.
Parameters	`v_intr_enable_u8` the value of the interrupts' mapping to INT pin enable bit `BMM050_DISABLE, BMM050_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bmm150_[set/get]_drdy_enable()

Function Name	`int8_t bmm150_set_drdy_enable(uint8_t v_drdy_enable_u8)` `int8_t bmm150_get_drdy_enable(uint8_t* v_drdy_enable_u8)`
Function Description	(CONFIG) Sets/gets the state of mapping the data ready (drdy) interrupt status to the DRDY pin of the BMM150 sensor module.
Parameters	`v_drdy_enable_u8` the value of the data ready interrupt mapping to DRDY pin enable bit `BMM050_DISABLE, BMM050_ENABLE`
Return Values	0 for successful execution; ≠0 for error
Notes	

bmm150_[set/get]_mag_low_thres()

Function Name	`int8_t bmm150_set_mag_low_thres(uint8_t v_thres_u8)` `int8_t bmm150_get_mag_low_thres(uint8_t* v_thres_u8)`
Function Description	(CONFIG) Sets/gets the low threshold level for magnetometer sensor data, which defines the condition for triggering low threshold interrupt events.
Parameters	`v_thres_u8` the value of the magnetometer data low threshold level
Return Values	0 for successful execution; ≠0 for error
Notes	1 bit corresponds to roughly 6 μT.

bmm150_[set/get]_mag_high_thres()

Function Name	`int8_t bmm150_set_mag_high_thres(uint8_t v_thres_u8)` `int8_t bmm150_get_mag_high_thres(uint8_t* v_thres_u8)`
Function Description	(CONFIG) Sets/gets the high threshold level for magnetometer sensor data, which defines the condition for triggering high threshold interrupt events.
Parameters	`v_thres_u8` the value of the magnetometer data high threshold level
Return Values	0 for successful execution; ≠0 for error
Notes	1 bit corresponds to roughly 6 μT.

bmm150_[set/get]_mag_repetition_xy()

Function Name	`int8_t bmm150_set_mag_repetition_xy(uint8_t v_rep_xy_u8)` `int8_t bmm150_get_mag_repetition_xy(uint8_t* v_rep_xy_u8)`
Function Description	(CONFIG) Sets/gets the number of repetitions for magnetometer xy-axis data.
Parameters	`v_rep_xy_u8` the value of the magnetometer xy-axis repetitions
Return Values	0 for successful execution; ≠0 for error
Notes	The performed number of repetitions are given by: num. of xy-repetitions = 1 + 2 × `v_rep_xy_u8`

© 2017 Dialog Semiconductor

bmm150_[set/get]_mag_repetition_z()

Function Name	`int8_t bmm150_set_mag_repetition_z(uint8_t v_rep_z_u8)` `int8_t bmm150_get_mag_repetition_z(uint8_t* v_rep_z_u8)`
Function Description	(CONFIG) Sets/gets the number of repetitions for magnetometer z-axis data.
Parameters	`v_rep_z_u8` the value of the magnetometer z-axis repetitions
Return Values	0 for successful execution; ≠0 for error
Notes	The performed number of repetitions are given by: num. of z-repetitions = 1 + `v_rep_z_u8`

10.1.2.8 Low Level Register Access Functions

bmm150_[set/get]_reg_[reg_name]()

Function Name	`int8_t bmm150_set_reg_power_control(uint8_t v_reg_value_u8)`
	`int8_t bmm150_get_reg_power_control(uint8_t *v_reg_value_u8)`
	`int8_t bmm150_set_reg_control(uint8_t v_reg_value_u8)`
	`int8_t bmm150_get_reg_control(uint8_t *v_reg_value_u8)`
	`int8_t bmm150_set_reg_int_control(uint8_t v_reg_value_u8)`
	`int8_t bmm150_get_reg_int_control(uint8_t *v_reg_value_u8)`
	`int8_t bmm150_set_reg_sens_control(uint8_t v_reg_value_u8)`
	`int8_t bmm150_get_reg_sens_control(uint8_t *v_reg_value_u8)`
	`int8_t bmm150_set_reg_low_thres(uint8_t v_reg_value_u8)`
	`int8_t bmm150_get_reg_low_thres(uint8_t *v_reg_value_u8)`
	`int8_t bmm150_set_reg_high_thres(uint8_t v_reg_value_u8)`
	`int8_t bmm150_get_reg_high_thres(uint8_t *v_reg_value_u8)`
	`int8_t bmm150_set_reg_rep_xy(uint8_t v_reg_value_u8)`
	`int8_t bmm150_get_reg_rep_xy(uint8_t *v_reg_value_u8)`
	`int8_t bmm150_set_reg_rep_z(uint8_t v_reg_value_u8)`
	`int8_t bmm150_get_reg_rep_z(uint8_t *v_reg_value_u8)`
	`int8_t bmm150_get_reg_int_stat(uint8_t *v_reg_value_u8)`

Function Description	(LOW-ACCESS) Sets/gets the register [reg_name] value based on the register names presented in BMM150 datasheet Register Map
Parameters	`v_reg_value_u8` the value of the register [reg_name]
Return Values	0 for successful execution; ≠0 for error
Notes	Direct write- or read-access to BMM150 registers, without any further processing (i.e. bit- masking and shifting) or delay (if any)

© 2017 Dialog Semiconductor

10.1.3 BME280 Sensor Driver

The BME280 sensor driver API provides functions for the configuration, control and sensor data acquisition from the integrated by the corresponding module environmental sensors: pressure, temperature and humidity. Below mainly the functions are presented which are used by the Wearable project, while a complete list of the functions is presented in file `bme280.h`, based on the datasheet [12] provided by Bosch Sensortec.

General guidelines for using the BME280 sensor driver are as follows:

- Initialize the sensor module using the function `bme280_initialize_sensor()`. The device address for the sensor module must be set, along with pointers to functions which implement SPI/I2C bus access and introduce delay.
- Set the power mode state of the module by setting the power mode state of the environmental sensors using the function `bme280_set_sensor_state()`.
- Configure the sensor module in terms of specific setting/operation features (`setting_feature = oversamp_pressure`, `oversamp_temperature`, `oversamp_humidity`, etc.) using the `bme280_set_[setting_feature]()` set of functions.
- When the current configuration state of the sensor module is to be checked in terms of specific setting/operation features, use the `bme280_get_[setting_feature]()` set of functions.
- Use either NORMAL power mode state, so that the sensor data registers can be automatically and periodically updated with new sample data, based on a specific-configured output data rate, or periodically change the power mode state of the sensors to FORCED, so that the output data rate can be controlled 'manually'. When the second case is considered, use `bme280_calc_force_read_wait_time()` to calculate the time needed for all enabled sensors data registers to be updated with new sample data (in essence, determining the maximum data rate), based on the current configuration.
- Access the sensor module's registers (`reg_name`) when low level control and data handling is required, using the `bme280_[set/get]_reg_[reg_name]()` set of functions.
- Read environmental sensor data using the `bme280_read_[pressure/temperature/humidity]()` set of functions.

A list of all API data structures/types and functions provided by the BME280 sensor driver and presented in this section follows:

Table 36: BME280 Driver API

Data Structures and Types	
bme280_t	
Dependency Functions	
bme280_t -> bus_write() bme280_t -> delay_msec()	bme280_t -> bus_read()
Initialization and Basic Configuration Functions	
bme280_initialize_sensor()	bme280_set_sensor_state()
Sensor Data Acquisition Functions	
bme280_read_pressure_temperature_humidity() bme280_read_temperature()	bme280_read_pressure() bme280_read_humidity()
Command Functions	
bme280_soft_reset()	
State Functions	
bme280_get_chip_id() bme280_get_measuring_stat()	bme280_calc_force_read_wait_time()

© 2017 Dialog Semiconductor

Configuration Functions	
bme280_[set/get]_hum_oversamp() bme280_[set/get]_temp_oversamp() bme280_[set/get]_standby_time()	bme280_[set/get]_pres_oversamp() bme280_[set/get]_filter_coefficient()
Low Level Register Access Functions	
bme280_[set/get]_reg_[reg_name]()	

10.1.3.1 Data Structures and Types

bme280_t

Data Structure Fields	Type	Description
chip_id	uint8_t	Chip ID of the BME280 sensor module.
dev_addr	uint8_t	Device address of the BME280 sensor module.
bus_write	Function pointer	Function for SPI/I2C bus write-access.
bus_read	Function pointer	Function for SPI/I2C bus read-access.
delay_msec	Function pointer	Function for introducing delay (in ms).

10.1.3.2 Dependency Functions

bme280_t -> bus_write()

Function Name	`int8_t (*bus_write)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data, uint8_t cnt)`
Function Description	Implements write-access over the SPI / I2C bus for writing data to BME280 sensor module registers.
Parameters	`dev_addr` the device address `reg_addr` the address of the register to write `reg_data` the data to write `cnt` the size of data (in bytes)
Return Values	0 for successful execution; ≠0 for error
Notes	

bme280_t -> bus_read()

Function Name	`int8_t (*bus_read)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data, uint8_t cnt)`
Function Description	Implements read-access over the SPI / I2C bus for reading data from BME280 sensor module registers.
Parameters	`dev_addr` the device address `reg_addr` the address of the register to read `reg_data` the buffer to put register data being read to `cnt` the size of data (in bytes)
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

bme280_t -> delay_msec()

Function Name	`void (*delay_msec)(uint32_t ms)`
Function Description	Implements a delay in ms that may be necessary for the BME280 sensor module driver functionality implementation.
Parameters	`ms` the delay (in ms) to be introduced
Return Values	None
Notes	

10.1.3.3 Initialization and Basic Configuration Functions

bme280_initialize_sensor()

Function Name	`int8_t bme280_initialize_sensor(bme280_t* s_bme280)`
Function Description	(INIT-CONFIG) Initializes the BME280 sensor module, creating the appropriate instance handle for the module which includes references to the SPI / I2C bus access functions used for reading/writing values to the registers, setting SLEEP power mode state for the environmental sensors, getting their calibration parameter values and soft-resetting the sensor module.
Parameters	`s_bme280` the handle structure for BME280 driver
Return Values	0 for successful execution; ≠0 for error
Notes	

bme280_set_sensor_state()

Function Name	`int8_t bme280_set_sensor_state(uint8_t state)`
Function Description	(INIT-CONFIG) Sets the state of the environmental sensors.
Parameters	`state` the state to set for the environmental sensors `ENV_SLEEP, ENV_FORCED, ENV_NORMAL`
Return Values	0 for successful execution; ≠0 for error
Notes	

10.1.3.4 Sensor Data Acquisition Functions

bme280_read_pressure_temperature_humidity()

Function Name	`int8_t bme280_read_pressure_temperature_humidity(uint32_t *v_pressure_u32, int32_t *v_temperature_s32, uint32_t *v_humidity_u32)`
Function Description	(SAMPLE) Reads pressure, temperature and humidity sensor data.
Parameters	`v_pressure_u32` the pressure sensor data `v_temperature_s32` the temperature sensor data `v_humidity_u32` the humidity sensor data
Return Values	0 for successful execution; ≠0 for error
Notes	Pressure, temperature and humidity sensor data values are measured in 1 Pa, 0.01 ºC and 1/1024 %RH units, respectively.

© 2017 Dialog Semiconductor

bme280_read_pressure()

Function Name	`int8_t bme280_read_pressure(uint32_t *v_pressure_u32)`
Function Description	(SAMPLE) Reads pressure sensor data.
Parameters	`v_pressure_u32` the pressure sensor data
Return Values	0 for successful execution; ≠0 for error
Notes	Pressure sensor data values are measured in 1 Pa units.

bme280_read_temperature()

Function Name	`int8_t bme280_read_temperature(int32_t *v_temperature_s32)`
Function Description	(SAMPLE) Reads temperature sensor data.
Parameters	`v_temperature_s32` the temperature sensor data
Return Values	0 for successful execution; ≠0 for error
Notes	Temperature sensor data values are measured in 0.01 ºC units.

bme280_read_humidity()

Function Name	`int8_t bme280_read_humidity(uint32_t *v_humidity_u32)`
Function Description	(SAMPLE) Reads humidity sensor data.
Parameters	`v_humidity_u32` the humidity sensor data
Return Values	0 for successful execution; ≠0 for error
Notes	Humidity sensor data values are measured in 1/1024 %RH units.

10.1.3.5 Command Functions

bme280_soft_reset()

Function Name	`int8_t bme280_soft_reset(void)`
Function Description	(CMD) Performs a soft reset to the BME280 sensor module.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	On successful execution it is introduced a `BME280_3MS_DELAY` delay.

10.1.3.6 State Functions

bme280_get_chip_id()

Function Name	`int8_t bme280_get_chip_id(uint8_t* v_chip_id_u8)`
Function Description	(STATE) Reads the chip id of the BME280 sensor module.
Parameters	`v_chip_id_u8` the chip id
Return Values	0 for successful execution; ≠0 for error
Notes	

 © 2017 Dialog Semiconductor

bme280_calc_force_read_wait_time()

Function Name	`int8_t bme280_calc_force_read_wait_time(uint8_t* wait_time)`
Function Description	(STATE) Calculates wait time needed (in ms) for the sensor data registers to be updated after a change of the power state mode of the environmental sensors to FORCED is performed (see `bme280_set_sensor_state(ENV_FORCED)`).
Parameters	`wait_time` the value of the wait time (ms)
Return Values	0 for successful execution; ≠0 for error
Notes	

bme280_get_measuring_stat()

Function Name	`int8_t bme280_get_measuring_stat(uint8_t* v_measuring_u8)`
Function Description	(STATE) Gets the measuring status of the BME280 module. Measuring status bit is automatically set to '1' whenever a conversion is running and back to '0' when the sensor data registers have been updated.
Parameters	`v_measuring_u8` the value of the measuring status (range 0 to 1)
Return Values	0 for successful execution; ≠0 for error
Notes	

10.1.3.7 Configuration Functions

bme280_[set/get]_hum_oversamp()

Function Name	`int8_t bme280_set_hum_oversamp(uint8_t v_oversamp_u8)` `int8_t bme280_get_hum_oversamp(uint8_t* v_oversamp_u8)`
Function Description	(CONFIG) Sets/gets the oversampling mode for humidity sensor data.
Parameters	`v_oversamp_u8` the value of the humidity oversampling mode `BME280_OVERSAMP_SKIPPED, BME280_OVERSAMP_1X, BME280_OVERSAMP_2X, BME280_OVERSAMP_4X, BME280_OVERSAMP_8X, BME280_OVERSAMP_16X`
Return Values	0 for successful execution; ≠0 for error
Notes	Setting the oversampling mode reduces noise in sensor data. Changes to humidity oversampling mode only become effective after setting either the oversampling mode for temperature sensor data or the oversampling mode for pressure sensor data or changing environmental sensors power state mode.

bme280_[set/get]_pres_oversamp()

Function Name	`int8_t bme280_set_pres_oversamp(uint8_t v_oversamp_u8)` `int8_t bme280_get_pres_oversamp(uint8_t* v_oversamp_u8)`
Function Description	(CONFIG) Sets/gets the oversampling mode for pressure sensor data.
Parameters	`v_oversamp_u8` the value of the pressure oversampling mode `BME280_OVERSAMP_SKIPPED, BME280_OVERSAMP_1X, BME280_OVERSAMP_2X, BME280_OVERSAMP_4X, BME280_OVERSAMP_8X, BME280_OVERSAMP_16X`
Return Values	0 for successful execution; ≠0 for error
Notes	Setting the oversampling mode reduces noise in sensor data.

© 2017 Dialog Semiconductor

bme280_[set/get]_temp_oversamp()

Function Name	`int8_t bme280_set_temp_oversamp(uint8_t v_oversamp_u8)` `int8_t bme280_get_temp_oversamp(uint8_t* v_oversamp_u8)`
Function Description	(CONFIG) Sets/gets the oversampling mode for temperature sensor data.

Parameters	`v_oversamp_u8` the value of the temperature oversampling mode
	`BME280_OVERSAMP_SKIPPED, BME280_OVERSAMP_1X,`
	`BME280_OVERSAMP_2X, BME280_OVERSAMP_4X,`
	`BME280_OVERSAMP_8X, BME280_OVERSAMP_16X`

Return Values	0 for successful execution; ≠0 for error
Notes	Setting the oversampling mode reduces noise in sensor data.

bme280_[set/get]_filter_coefficient()

Function Name	`int8_t bme280_set_filter_coefficient(uint8_t v_filter_coef_u8)` `int8_t bme280_get_filter_coefficient(uint8_t* v_filter_coef_u8)`
Function Description	(CONFIG) Sets/gets the coefficient value for BME280's integrated IIR filter.

Parameters	`v_filter_coef_u8` the value of the filter coefficient
	`BME280_FILTER_COEFF_OFF, BME280_FILTER_COEFF_2,`
	`BME280_FILTER_COEFF_4, BME280_FILTER_COEFF_8,`
	`BME280_FILTER_COEFF_16`

Return Values	0 for successful execution; ≠0 for error
Notes	

bme280_[set/get]_standby_time()

Function Name	`int8_t bme280_set_standby_time(uint8_t v_standby_durn_u8)` `int8_t bme280_get_standby_time(uint8_t* v_standby_durn_u8)`
Function Description	(CONFIG) Sets/gets the standby time duration in a measurement cycle, that is, the time interval in NORMAL power mode state between two subsequent sensor (temperature or/and pressure or/and humidity) data measurements.

Parameters	`v_standby_durn_u8` the value of the standby time duration
	`BME280_STANDBY_TIME_1_MS , BME280_STANDBY_TIME_63_MS,`
	`BME280_STANDBY_TIME_125_MS, BME280_STANDBY_TIME_250_MS,`
	`BME280_STANDBY_TIME_500_MS, BME280_STANDBY_TIME_1000_MS,`
	`BME280_STANDBY_TIME_10_MS , BME280_STANDBY_TIME_20_MS`

Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

10.1.3.8 Low Level Register Access Functions

bme280_[set/get]_reg_[reg_name]()

Function Name	`int8_t bme280_set_reg_ctrl_hum(uint8_t v_reg_value_u8)`
	`int8_t bme280_get_reg_ctrl_hum(uint8_t *v_reg_value_u8)`
	`int8_t bme280_get_reg_status(uint8_t *v_reg_value_u8)`
	`int8_t bme280_set_reg_ctrl_meas(uint8_t v_reg_value_u8)`
	`int8_t bme280_get_reg_ctrl_meas(uint8_t *v_reg_value_u8)`
	`int8_t bme280_set_reg_config(uint8_t v_reg_value_u8)`
	`int8_t bme280_get_reg_config(uint8_t *v_reg_value_u8)`

Function Description	(LOW-ACCESS) Sets/gets the register [reg_name] value based on the register names presented in BME280 datasheet Register Map.
Parameters	`v_reg_value_u8` the value of the register [reg_name]
Return Values	0 for successful execution; ≠0 for error
Notes	Direct write- or read-access to BME280 registers, without any further processing (i.e. bit- masking and shifting) or delay (if any)

© 2017 Dialog Semiconductor

10.1.4 DI5115 Sensor Driver

The DI5115 sensor high-level driver API provides functions for the configuration, control and sensor data acquisition from the integrated by the corresponding module accelerometer and optical (Health Care - HC) sensors. The term 'high level' means that the API functions do not refer to direct register accesses (as in the Bosch Sensortec sensor drivers), but to a more complex functionality, which implements the configuration of the integrated sensors and sensor data manipulation, based on the needs of the Wearable DK for the implementation of the supported heart rate estimation service.

The DI5115 sensor module also drives two LEDs (Green and IR light) needed to perform heart rate estimation. Thus the functionality for configuring the operation of the LEDs is being provided/hidden by the high level DI5115 sensor driver API. In this particular implementation the IR LED light is not used and can be ignored. The rate at which the LED light intensity and accelerometer data are updated are 50 Hz and 64 Hz, respectively. Below the functions are presented which are used by the Wearable project (refer also to file `di5115_hld.h`).

General guidelines for using the DI5115 sensor driver are as follows:

- Initialize sensor module using the function `di5115_hld_opt_init()`. It initializes the DI5115 sensor module and initially-configures the core sensor, i.e. optical-Health Care (HC) sensor. Different I2C device addresses for the optical-HC and accelerometer sensors must be set, respectively, along with pointers to functions that implement I2C bus access and introduce delay.
- Initialize the accelerometer sensor using the function `di5115_hld_acc_init()`.
- Set the power mode state of the optical-HC sensor using the function `di5115_hld_set_opt_state()`.
- Set the power mode state of the accelerometer sensor using the function `di5115_hld_set_acc_state()`.
- Disable any involved interrupt-based or -related operation in the sensor module high level driver implementation using the function `di5115_hld_opt_disable_intrs()`.
- Read optical-HC sensor LED light intensity data (i.e. Green and IR) using the function `di5115_hld_opt_hr_leds_intensity_read()`.
- Read accelerometer sensor xyz-axis data using the function `di5115_hld_accel_read()`.
- Use the function `di5115_hld_opt_hr_adapt_optical_gain()` to adapt (e.g. to the type and state of the skin) the operation of the employed LEDs for the heart rate estimation based on the acquired light intensity.

A list of all API data structure/types and functions provided by the DI5115 sensor driver and presented in this section follows:

Table 37: DI5115 Driver API

Data Structures and Types	
di5115_hld_error_t di5115_hld_hr_ir_led_intens_data_t di5115_hld_acc_data_t	di5115_hld_hr_gr_led_intens_data_t di5115_hld_handle_t
Dependency Functions	
di5115_hld_handle_t -> bus_write() di5115_hld_handle_t -> burst_read()	di5115_hld_handle_t -> bus_read() di5115_hld_handle_t -> delay_msec()
Initialization and Basic Configuration Functions	
di5115_hld_opt_init() di5115_hld_set_opt_state()	di5115_hld_acc_init() di5115_hld_set_acc_state()
Sensor Data Acquisition Functions	
di5115_hld_opt_hr_leds_intensity_read()	di5115_hld_accel_read()

© 2017 Dialog Semiconductor

Command Functions	
di5115_hld_opt_disable_intrs()	
Configuration Functions	
di5115_hld_opt_hr_adapt_optical_gain()	

10.1.4.1 Data Structures and Types

di5115_hld_error_t

Type Definition Name	Description
di5115_hld_error_t	Execution status of a function: DI5115_HLD_SUCCESS, DI5115_HLD_FAIL

di5115_hld_hr_gr_led_intens_data_t

Type Definition Name	Description
di5115_hld_hr_gr_led_intens_data_t	Green LED light intensity raw data value representation: uint16_t

di5115_hld_hr_ir_led_intens_data_t

Type Definition Name	Description
di5115_hld_hr_ir_led_intens_data_t	IR LED light intensity raw data value representation: uint16_t

di5115_hld_handle_t

Data Structure Fields	Type	Description
acc_addr	uint8_t	Device (I2C) address of the accelerometer sensor.
opt_addr	uint8_t	Device (I2C) address of the optical-HC sensor.
bus_write	Function pointer	Function for I2C bus write-access.
bus_read	Function pointer	Function for I2C bus read-access.
burst_read	Function pointer	Function for I2C bus burst-read-access.
delay_msec	Function pointer	Function for introducing delay (in ms).

di5115_hld_acc_data_t

Data Structure Fields	Type	Description
x	int16_t	Accelerometer x-axis raw data value.
y	int16_t	Accelerometer y-axis raw data value.
z	int16_t	Accelerometer z-axis raw data value.

© 2017 Dialog Semiconductor

10.1.4.2 Dependency Functions

di5115_hld_handle_t -> bus_write()

Function Name	`int8_t (*bus_write)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data, uint8_t cnt)`
Function Description	Implements write-access over the I2C bus for writing data to DI5115 sensor module registers.
Parameters	`dev_addr` the device (I2C) address `reg_addr` the address of the register to write `reg_data` the data to write `cnt` the size of data (in bytes)
Return Values	0 for successful execution; ≠0 for error
Notes	

di5115_hld_handle_t -> bus_read()

Function Name	`int8_t (*bus_read)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data, uint8_t cnt)`
Function Description	Implements read-access over the I2C bus for reading data from DI5115 sensor module registers.
Parameters	`dev_addr` the device (I2C) address `reg_addr` the address of the register to read `reg_data` the buffer to put register data being read to `cnt` the size of data (in bytes)
Return Values	0 for successful execution; ≠0 for error
Notes	

di5115_hld_handle_t -> burst_read()

Function Name	`int8_t (*burst_read)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data, uint32_t len)`
Function Description	Implements burst-read-access over the I2C bus for reading data from DI5115 sensor module registers.
Parameters	`dev_addr` the device (I2C) address `reg_addr` the address of the register to read `reg_data` the buffer to put register data being read to `len` the size of data (in bytes)
Return Values	0 for successful execution; ≠0 for error
Notes	

di5115_hld_handle_t -> delay_msec()

Function Name	`void (*delay_msec)(uint32_t ms)`
Function Description	Implements a delay in ms that may be necessary for the DI5115 sensor module driver functionality implementation.
Parameters	`ms` the delay (in ms) to be introduced
Return Values	None
Notes	

© 2017 Dialog Semiconductor

10.1.4.3 Initialization and Basic Configuration Functions

di5115_hld_opt_init()

Function Name	`di5115_hld_error_t di5115_hld_opt_init(di5115_hld_handle_t* handle)`
Function Description	Initializes the DI5115 sensor module, creating the appropriate instance handle for the module which includes references to the I2C bus access functions used for reading/writing values to the registers, and soft-resetting its state. The integrated sensors, that is, accelerometer and optical, are initialized to STANDBY (SUSPEND) and POWER_DOWN (SUSPEND) power mode states, respectively, after soft-resetting and powering-on.
Parameters	`handle` the handle structure for DI5115 high level driver
Return Values	`DI5115_HLD_SUCCESS` for successful execution; ≥ `DI5115_HLD_FAIL` for error
Notes	

di5115_hld_acc_init()

Function Name	`di5115_hld_error_t di5115_hld_acc_init(void)`
Function Description	Initializes the accelerometer sensor with the required for the proper operation of the heart rate estimation algorithm configuration (i.e. 64 Hz output data rate, 8-bit resolution and ±2G range).
Parameters	None
Return Values	`DI5115_HLD_SUCCESS` for successful execution; ≥ `DI5115_HLD_FAIL` for error
Notes	

di5115_hld_set_opt_state()

Function Name	`di5115_hld_error_t di5115_hld_set_opt_state(uint8_t state)`
Function Description	Sets the state of the optical-HC sensor and the operation of LEDs.
Parameters	`state` the state to set for the optical-HC sensor and LEDs' operation `DI5115_HLD_STATE_OPT_SUSPEND`, `DI5115_HLD_STATE_OPT_NORMAL`
Return Values	`DI5115_HLD_SUCCESS` for successful execution; ≥ `DI5115_HLD_FAIL` for error
Notes	In NORMAL mode LED light intensity data are updated every `DI5115_HLD_HR_LED_INTENS_SAMPLING_RATE` msec time interval (i.e. 20 ms or 50 Hz)

di5115_hld_set_acc_state()

Function Name	`di5115_hld_error_t di5115_hld_set_acc_state(uint8_t state)`
Function Description	Sets the state of the accelerometer sensor.
Parameters	`state` the state to set for the accelerometer sensor `DI5115_HLD_STATE_ACC_SUSPEND`, `DI5115_HLD_STATE_ACC_NORMAL`
Return Values	`DI5115_HLD_SUCCESS` for successful execution; ≥ `DI5115_HLD_FAIL` for error
Notes	In NORMAL mode accelerometer xyz-axis data are updated with an output data rate 64 Hz.

© 2017 Dialog Semiconductor

10.1.4.4 Sensor Data Acquisition Functions

di5115_hld_opt_hr_leds_intensity_read()

Function Name	`di5115_hld_error_t di5115_hld_opt_hr_leds_intensity_read(`
	`  di5115_hld_hr_gr_led_intens_data_t* hgr_data,`
	`  di5115_hld_hr_ir_led_intens_data_t* hir_data,`
	`  bool* valid_data)`

Function Description	Reads optical sensor data related to Green and IR LED light intensity.
Parameters	`hgr_data` the Green LED light intensity data `hir_data` the IR LED light intensity data `valid_data` indication that sensor data are valid
Return Values	`DI5115_HLD_SUCCESS` for successful execution; ≥ `DI5115_HLD_FAIL` for error
Notes	

di5115_hld_accel_read()

Function Name	`di5115_hld_error_t di5115_hld_accel_read(di5115_hld_acc_data_t* acc_data)`
Function Description	Reads accelerometer xyz-axis data.
Parameters	`acc_data` the accelerometer xyz-axis data
Return Values	`DI5115_HLD_SUCCESS` for successful execution; ≥ `DI5115_HLD_FAIL` for error
Notes	

10.1.4.5 Command Functions

di5115_hld_opt_disable_intrs()

Function Name	`di5115_hld_error_t di5115_hld_opt_disable_intrs(void)`
Function Description	Disables any involved interrupt-based or -related operation in the sensor module high level driver implementation.
Parameters	None
Return Values	`DI5115_HLD_SUCCESS` for successful execution; ≥ `DI5115_HLD_FAIL` for error
Notes	

10.1.4.6 Configuration Functions

di5115_hld_opt_hr_adapt_optical_gain()

Function Name	`di5115_hld_error_t di5115_hld_opt_hr_adapt_optical_gain(`
	`  di5115_hld_hr_gr_led_intens_data_t hgr_data,`
	`  di5115_hld_hr_ir_led_intens_data_t hir_data,`
	`  bool* adapt_done)`

Function Description	Adapt (e.g. to the type and state of the skin) the operation of the employed LEDs for heart rate estimation based on the acquired light intensity.
Parameters	`hgr_data` the Green LED light intensity data `hir_data` the IR LED light intensity data `adapt_done` indication that adaptation is completed
Return Values	`DI5115_HLD_SUCCESS` for successful execution; ≥ `DI5115_HLD_FAIL` for error
Notes	

© 2017 Dialog Semiconductor

10.1.5 SX9300 Sensor Driver

The SX9300 sensor driver API provides functions for communication with the SX9300 touch sensor module. The wearable task gets information from the touch sensor via the SX9300 driver and uses it to detect finger touches on the two touch pads below the wearable screen. Configurations and registers of the sensor can be set easily using the provided API functions.

A list of all API data structures/types and functions provided by the SX9300 driver and presented in this section follows:

Table 38: SX9300 Driver API

Data Structures and Types	
sx9300_scan_period_t	sx9300_shield_t
sx9300_input_cap_t	sx9300_gain_t
sx9300_sampling_freq_t	sx9300_doze_en_period_t
sx9300_prox_raw_filter_t	sx9300_deb_samples_t
sx9300_prox_threshold_t	sx9300_prox_hysteresis_t
sx9300_comp_method_t	sx9300_sar_threshold_t
sx9300_prox_stuck_timeout_t	sx9300_t
Functions	
sx9300_init()	sx9300_poll()
sx9300_set_irq_mask()	sx9300_set_sensor_enable()
sx9300_set_[register field]()	sx9300_shield_config()
sx9300_config_dozen()	sx9300_config_average()
sx9300_config_autocompensation()	sx9300_config_close_far_deb()
sx9300_config_sar()	sx9300_soft_reset()
Dependency Functions	
bus_write()	bus_read()
cs0_touch()	cs1_touch()

10.1.5.1 Data Structures and Types

sx9300_scan_period_t

Type Definition Name	Description
sx9300_scan_period_t	Scan period enumeration type defined as: SX9300_SCAN_PERIOD_30_MS, SX9300_SCAN_PERIOD_60_MS, SX9300_SCAN_PERIOD_90_MS, SX9300_SCAN_PERIOD_120_MS, SX9300_SCAN_PERIOD_150_MS, SX9300_SCAN_PERIOD_200_MS, SX9300_SCAN_PERIOD_300_MS, SX9300_SCAN_PERIOD_400_MS

sx9300_shield_t

Type Definition Name	Description
sx9300_shield_t	Shield state enumeration type defined as: SX9300_SHIELD_OFF, SX9300_SHIELD_ON

 © 2017 Dialog Semiconductor

sx9300_input_cap_t

Type Definition Name	Description
sx9300_input_cap_t	Input capacitance range enumeration type defined as: SX9300_CAP_LARGE, SX9300_CAP_MEDIUM_LARGE, SX9300_CAP_MEDIUM_SMALL, SX9300_CAP_SMALL

sx9300_gain_t

Type Definition Name	Description
sx9300_gain_t	Digital gain factor enumeration type defined as: SX9300_GAIN_OFF, SX9300_GAIN_2, SX9300_GAIN_4, SX9300_GAIN_8

sx9300_sampling_freq_t

Type Definition Name	Description
sx9300_sampling_freq_t	Sampling frequency enumeration type defined as: SX9300_FREQ_83K, SX9300_FREQ_125K, SX9300_FREQ_167K

sx9300_doze_en_period_t

Type Definition Name	Description
sx9300_dozen_period_t	Doze scan period enumeration type defined as: SX9300_DOZE_EN_PERIOD_2, SX9300_DOZE_EN_PERIOD_4, SX9300_DOZE_EN_PERIOD_8, SX9300_DOZE_EN_PERIOD_16

sx9300_prox_raw_filter_t

Type Definition Name	Description
sx9300_proxraw_filter_t	PROXRAW filter strength enumeration type defined as: SX9300_PROX_RAW_FILTER_OFF, SX9300_PROX_RAW_FILTER_LOW, SX9300_PROX_RAW_FILTER_MEDIUM, SX9300_PROX_RAW_FILTER_HIGH

sx9300_deb_samples_t

Type Definition Name	Description
sx9300_deb_samples_t	Average debouncer applied to AVGTHRESH enumeration type defined as: SX9300_DEB_OFF, SX9300_DEB_SAMPLES_2, SX9300_DEB_SAMPLES_4, SX9300_DEB_SAMPLES_8

© 2017 Dialog Semiconductor

sx9300_prox_threshold_t

Type Definition Name	Description
sx9300_proxthreshold_t	Proximity detection threshold enumeration type defined as: SX9300_PROX_THRESH_0, SX9300_PROX_THRESH_20, SX9300_PROX_THRESH_40, SX9300_PROX_THRESH_60, SX9300_PROX_THRESH_80, SX9300_PROX_THRESH_100, SX9300_PROX_THRESH_120, SX9300_PROX_THRESH_140, SX9300_PROX_THRESH_160, SX9300_PROX_THRESH_180, SX9300_PROX_THRESH_200, SX9300_PROX_THRESH_220, SX9300_PROX_THRESH_240, SX9300_PROX_THRESH_260, SX9300_PROX_THRESH_280, SX9300_PROX_THRESH_300, SX9300_PROX_THRESH_350, SX9300_PROX_THRESH_400, SX9300_PROX_THRESH_450, SX9300_PROX_THRESH_500, SX9300_PROX_THRESH_600, SX9300_PROX_THRESH_700, SX9300_PROX_THRESH_800, SX9300_PROX_THRESH_900, SX9300_PROX_THRESH_1000, SX9300_PROX_THRESH_1100, SX9300_PROX_THRESH_1200, SX9300_PROX_THRESH_1300, SX9300_PROX_THRESH_1400, SX9300_PROX_THRESH_1500, SX9300_PROX_THRESH_1600, SX9300_PROX_THRESH_1700

sx9300_prox_hysteresis_t

Type Definition Name	Description
sx9300_prox_hysteresis_t	Proximity detection hysteresis applied to PROXTHRESH enumeration type defined as: SX9300_PROX_HYSTERESIS_32, SX9300_PROX_HYSTERESIS_64, SX9300_PROX_HYSTERESIS_128, SX9300_PROX_HYSTERESIS_256

sx9300_comp_method_t

Type Definition Name	Description
sx9300_comp_method_t	Compensation method enumeration type defined as: SX9300_COMP_METHOD_INDEPENDENT, SX9300_COMP_METHOD_TOGETHER

© 2017 Dialog Semiconductor

sx9300_sar_threshold_t

Type Definition Name	Description
sx9300_sar_threshold_t	SAR delta threshold enumeration type defined as: SX9300_SAR_THRESH_OFF, SX9300_SAR_THRESH_1, SX9300_SAR_THRESH_3, SX9300_SAR_THRESH_5, SX9300_SAR_THRESH_10, SX9300_SAR_THRESH_15, SX9300_SAR_THRESH_20, SX9300_SAR_THRESH_25, SX9300_SAR_THRESH_30, SX9300_SAR_THRESH_35, SX9300_SAR_THRESH_40, SX9300_SAR_THRESH_45, SX9300_SAR_THRESH_50, SX9300_SAR_THRESH_55, SX9300_SAR_THRESH_60, SX9300_SAR_THRESH_70

sx9300_prox_stuck_timeout_t

Type Definition Name	Description
sx9300_prox_stuck_timeout_t	Proximity ‘stuck’ timeout enumeration type defined as: SX9300_PROX_STUCK_TIMEOUT_OFF = 0x00, SX9300_PROX_STUCK_TIMEOUT_64_SAMPLES = 0x02, SX9300_PROX_STUCK_TIMEOUT_128_SAMPLES = 0x04, SX9300_PROX_STUCK_TIMEOUT_256_SAMPLES = 0x08

sx9300_t

Field	Type	Description
bus_access_t	bus_write	Callback function to write an I2C register set
bus_access_t	bus_read	Callback function to read an I2C register set
cs0_touch	sx9300_touch_cb_t	Callback called when capacitive sensor 0 is touched
cs1_touch	sx9300_touch_cb_t	Callback called when capacitive sensor 1 is touched

© 2017 Dialog Semiconductor

10.1.5.2 Functions

sx9300_init()

Function Name	`void sx9300_init(const sx9300_t *handle);`
Function Description	Initializes SX9300 driver, allocates the memory, initializes I2C for communication with sensor, sets up sensitivities and other sensor configurations and configures interrupts.
Parameters	`handle` Handle of the SX9300 driver
Return Values	SX9300 driver instance.
Notes	

sx9300_poll()

Function Name	`void sx9300_poll();`
Function Description	Polls status register to check if a touch pad has been touched and calls the corresponding callback. Additionally, reads pending interrupts register to clear it.
Parameters	None
Return Values	None
Notes	

sx9300_set_irq_mask()

Function Name	`void sx9300_set_irq_mask(uint8_t mask);`
Function Description	Sets IRQ mask value to corresponding register.
Parameters	`mask` Mask value to set
Return Values	None
Notes	

sx9300_set_sensor_enable()

Function Name	`void sx9300_set_sensor_enable(uint8_t enable);`
Function Description	Sets state of sensor (enabled / disabled).
Parameters	`enable` State to set
Return Values	None
Notes	

sx9300_set_[register field]()

Function Name	`void sx9300_set_scan_period(sx9300_scan_period_t period);`
	`void sx9300_set_input_cap_range(sx9300_input_cap_t range);`
	`void sx9300_set_digital_gain(sx9300_gain_t gain);`
	`void sx9300_set_sampling_freq(sx9300_sampling_freq_t freq);`
	`void sx9300_set_cap_measure_res(uint8_t res);`
	`void sx9300_set_prox_raw_filter(sx9300_prox_raw_filter_t filter);`
	`void sx9300_set_average_threshold(uint8_t threshold);`
	`void sx9300_set_prox_threshold(sx9300_proxthreshold_t threshold);`
	`void sx9300_set_prox_hysteresis(sx9300_prox_hysteresis_t hysteresis);`
	`void sx9300_set_sar_ratio_threshold(uint8_t ratio_threshold);`
	`void sx9300_set_proximity_stuck_timeout(`
	`sx9300_prox_stuck_timeout_t timeout);`
	`void sx9300_set_periodic_compensation_interval(uint8_t time);`

Function Description	Sets the corresponding register field of the SX9300 touch sensor module.

© 2017 Dialog Semiconductor

Parameters	Corresponding value of `period` Scan period `range` Input capacitance range `gain` Digital gain factor `freq` Sampling frequency `res` Capacitance measurement resolution / precision `filter` PROXRAW filter strength `threshold` Average threshold that will trigger compensation `threshold` Proximity detection threshold `hysteresis` Proximity detection hysteresis applied to PROXTHRESH `ratio_threshold` SAR ratio threshold `timeout` Proximity 'stuck' timeout `time` Periodic compensation interval
Return Values	None
Notes	

sx9300_shield_config()

Function Name	`void sx9300_shield_config(sx9300_shield_t shield);`
Function Description	Sets the shield state register.
Parameters	`shield` Value to set
Return Values	None
Notes	

sx9300_config_doze_en()

Function Name	`void sx9300_config_doze_en(bool enable, sx9300_doze_en_period_t doze_en_period);`
Function Description	Configures the doze mode settings register.
Parameters	`sx` SX9300 driver instance `enable` State to set `doze_en_period` Doze scan period to set
Return Values	None
Notes	

sx9300_config_average()

Function Name	`void sx9300_config_average(sx9300_deb_samples_t samples, uint8_t neg_filter, uint8_t pos_filter);`
Function Description	Configures the average debounce filter.
Parameters	`samples` Average debouncer applied to AVGTHRESH `neg_filter` Average negative filter strength `pos_filter` Average positive filter strength
Return Values	None
Notes	

sx9300_config_autocompensation()

© 2017 Dialog Semiconductor

Function Name	`void sx9300_config_autocompensation(bool enable, sx9300_comp_method_t mode);`
Function Description	Configures the automatic compensation settings.
Parameters	`enable` True disables the automatic compensation `mode` Compensation method
Return Values	None
Notes	

sx9300_config_close_far_deb()

Function Name	`void sx9300_config_close_far_deb(sx9300_deb_samples_t close, sx9300_deb_samples_t far);`
Function Description	Configures close and far debouncer settings.
Parameters	`close` Close debouncer applied to PROXTHRESH `far` Far debouncer applied to PROXTHRESH
Return Values	None
Notes	

sx9300_config_sar()

Function Name	`void sx9300_config_sar(sx9300_deb_samples_t sar_deb, sx9300_sar_threshold_t threshold);`
Function Description	Configures Specific Absorption Rare settings register.
Parameters	`sar_deb` SAR engine debouncer applied to human body reporting `threshold` SAR delta threshold (for both sensor pairs)
Return Values	None
Notes	

sx9300_soft_reset()

Function Name	`void sx9300_soft_reset();`
Function Description	Performs chip software reset.
Parameters	None
Return Values	None
Notes	

10.1.5.3 Dependency Functions

bus_write()

Function Name	`void bus_write(uint8_t reg, uint8_t *value, size_t length);`
Function Description	Callback function to write an I2C register set
Parameters	`reg` Register to write `value` Data to set `length` Lenth of data to set
Return Values	None
Notes	

© 2017 Dialog Semiconductor

bus_read()

Function Name	`void bus_read(uint8_t reg, uint8_t *value, size_t length);`	
Function Description	Callback function to read an I2C register set	
Parameters	`reg`	Register to read
	`value`	Data to get
	`length`	Lenth of data to get
Return Values	None	
Notes		

cs0_touch()

Function Name	`void cs0_touch(bool both_touched);`	
Function Description	Informs that the capacitive sensor 0 touch pad was pressed.	
Parameters	`both_touched`	Indication if both touch pads have been detected as pressed.
Return Values	None	
Notes		

cs1_touch()

Function Name	`void cs1_touch(bool both_touched);`	
Function Description	Informs that the capacitive sensor 1 touch pad was pressed.	
Parameters	`both_touched`	Indication if both touch pads have been detected as pressed.
Return Values	None	
Notes		

© 2017 Dialog Semiconductor

10.1.6 AB08X5 Real Time Clock Driver

The AB08X5 RTC driver API provides functions for communication with the AB08X5 Real Time Clock module. The wearable task gets information from the AB08X5 RTC via the RTC driver and uses it in communication with other modules/tasks (e.g. display values in LCD driver). Time, alarm and date values can be set/changed according to input. Furthermore, the AB08X5 RTC device is used as a square wave generator for the LCD module (provided directly through SQW line) at 64 Hz and also as an event generator for the DA1468x device providing interrupts at a 2 s interval.

A list of all API data structures/types and functions provided by the AB08X5 driver and presented in this section follows:

Table 39: AB08X5 Driver API

Data Structures and Types	
ab08x5_alarm_mode_t AB08X5_err_t	ab08x5_airq_rpt_t ab08x5_t
Functions	
ab08x5_init() ab08x5_get_rtc_tick_intr_interval_ms ab08x5_get_[time/date]() ab08x5_set_full_time() ab08x5_get_[alarm field]_alarm() ab08x5_set_[alarm field]_alarm() ab08x5_set_full_date_alarm() ab08x5_get_status_register() ab08x5_[SQW/rtc_tick]_[enable/disable]() ab08x5_alarm_[rpt/mode]_config() ab08x5_get_timestamp()	ab08x5_set_rtc_tick_intr_interval_ms ab08x5_get_[field]() ab08x5_set_[field]() ab08x5_set_full_date() ab08x5_get_[time/date]_alarm() ab08x5_set_full_time_alarm() ab08x5_minute_leap() ab08x5_get_alarm_status() ab08x5_rtc_tick_is_enabled ab08x5_get_alarm_[rpt/mode]_config()

10.1.6.1 Data Structures and Types

ab08x5_alarm_mode_t

Type Definition Name	Description
ab08x5_alarm_mode_t	Alarm mode enumeration type defined as: AB08X5_ALARM_ONE_SHOT, AB08X5_ALARM_REPEATED

ab08x5_airq_rpt_t

Type Definition Name	Description
ab08x5_airq_rpt_t	Alarm interrupt frequency enumeration type defined as: AB08X5_AIRQ_RPT_DISABLED, AB08X5_AIRQ_RPT_ONCE_PER_YEAR, AB08X5_AIRQ_RPT_ONCE_PER_MONTH, AB08X5_AIRQ_RPT_ONCE_PER_WEEK, AB08X5_AIRQ_RPT_ONCE_PER_DAY, AB08X5_AIRQ_RPT_ONCE_PER_HOUR, AB08X5_AIRQ_RPT_ONCE_PER_MINUTE, AB08X5_AIRQ_RPT_ONCE_PER_SECOND Alarm interrupt AIRQ is generated when the values in the Time and Date registers match the values in the Alarm registers. Which register comparisons are required to generate AIRQ is controlled by this field.

© 2017 Dialog Semiconductor

ab08x5_err_t

Type Definition Name	Description
ab08x5_err_t	Error value enumeration type defined as: AB08X5_OK, AB08X5_NOT_FOUND, AB08X5_ERROR, AB08X5_BAD_VALUE = 0x04

ab08x5_t

Field	Type	Description
read_register	reg_access_callback	Read register callback
write_reister	reg_access_callback	Write register callback

© 2017 Dialog Semiconductor

10.1.6.2 Functions

ab08x5_init()

Function Name	`void ab08x5_init(const ab08x5_t *cb);`
Function Description	Initializes AB08X5 driver, sets hour mode to 24-hour mode, enables trickle charger and configures interrupts.
Parameters	None
Return Values	None
Notes	

ab08x5_set_rtc_tick_intr_interval_ms

Function Name	`bool ab08x5_set_rtc_tick_intr_interval_ms(uint32_t ms);`
Function Description	Set value of RTC tick interval in msec
Parameters	ms Interval to set in milliseconds
Return Values	True if interval was set successfully
Notes	

ab08x5_get_rtc_tick_intr_interval_ms

Function Name	`uint32_t ab08x5_get_rtc_tick_intr_interval_ms();`
Function Description	Gets the value of RTC tick interval in ms.
Parameters	None
Return Values	Value of tick interval in ms.
Notes	

ab08x5_get_[field]()

Function Name	`uint8_t ab08x5_get_hundredths(bool update);`
	`uint8_t ab08x5_get_seconds(bool update);`
	`uint8_t ab08x5_get_minutes(bool update);`
	`uint8_t ab08x5_get_hours(bool update);`
	`uint8_t ab08x5_get_days(bool update);`
	`uint8_t ab08x5_get_months(bool update);`
	`uint8_t ab08x5_get_years(bool update);`
	`uint8_t ab08x5_get_weekdays(bool update);`

Function Description	Gets value of hundredths / seconds / minutes / hours / days / months / years / weekdays.
Parameters	update true if value is read from RTC
Return Values	Corresponding value of `hundredths` (range <00 , 99>), `seconds` (range <00 , 59>), `minutes` (range <00 , 59>), `hours` (range <00 , 23>), `days` (range <01 , 31>), `months` (range <01, 12>), `years` (range <100 , 255>, since 1900), `weekdays` (0 - unknown, 1 - Monday, 7 - Sunday)
Notes	

ab08x5_get_[time/date]()

© 2017 Dialog Semiconductor

Function Name	`void ab08x5_get_time();` `void ab08x5_get_date();`
Function Description	Updates the time/date values in RTC structure
Parameters	None
Return Values	None
Notes	

ab08x5_set_[field]()

Function Name	`ab08x5_err_t ab08x5_set_hundredths(uint8_t hundredths);`
	`ab08x5_err_t ab08x5_set_seconds(uint8_t seconds);`
	`ab08x5_err_t ab08x5_set_minutes(uint8_t minutes);`
	`ab08x5_err_t ab08x5_set_hours(uint8_t hours);`
	`ab08x5_err_t ab08x5_set_days(uint8_t days);`
	`ab08x5_err_t ab08x5_set_months(uint8_t months);`
	`ab08x5_err_t ab08x5_set_years(uint8_t years);`
	`ab08x5_err_t ab08x5_set_weekdays(uint8_t weekdays);`

Function Description	Sets value of hundredths / seconds / minutes / hours / days / months / years / weekdays.
Parameters	Corresponding value of `hundredths` value of hundredths to be set (range <00 , 99>) `seconds` value of seconds to be set (range <00 , 59>) `minutes` value of minutes to be set (range <00 , 59>) `hours` value of hours to be set (range <00, 23>) `days` value of days to be set (range <00, 31>) `months` value of months to be set (range <01, 12>) `years` value of years to be set (range <00, 255>, since 1900) `weekdays` value of weekdays to be set (0 - unknown, 1 - Monday, 7 - Sunday)
Return Values	Error number `ab08x5_err_t`
Notes	

ab08x5_set_full_time()

Function Name	`ab08x5_err_t ab08x5_set_full_time(uint8_t hours, uint8_t minutes,` `uint8_t seconds, uint8_t hundredths);`
Function Description	Sets all time values.
Parameters	`hours` Hours value to be set `minutes` Minutes value to be set `seconds` Seconds value to be set `hundredths` Hundredths value to be set
Return Values	Error number `ab08x5_err_t`
Notes	

ab08x5_set_full_date()

Function Name	`ab08x5_err_t ab08x5_set_full_date(uint8_t years, uint8_t months,` `uint8_t days, uint8_t weekdays);`
Function Description	Sets all date values.

© 2017 Dialog Semiconductor

Parameters	`years`
	`months`
	`days`
	`weekday`
		Years value to be set
		Months value to be set
		Days value to be set
		Weekday value to be set

Return Values	Error number `ab08x5_err_t`	
Notes		

ab08x5_get_[alarm field]_alarm()

Function Name	`uint8_t ab08x5_get_hundredths_alarm(bool update);`
	`uint8_t ab08x5_get_seconds_alarm(bool update);`
	`uint8_t ab08x5_get_minutes_alarm(bool update);`
	`uint8_t ab08x5_get_hours_alarm(bool update);`
	`uint8_t ab08x5_get_days_alarm(bool update);`
	`uint8_t ab08x5_get_months_alarm(bool update);`
	`uint8_t ab08x5_get_weekdays_alarm(bool update);`

Function Description	Gets value of hundredths / seconds / minutes / hours / days / months / weekdays.
Parameters	update — true if value is read from RTC
Return Values	Corresponding value of `hundredths` (range <00 , 99>), `seconds` (range <00 , 59>), `minutes` (range <00 , 59>), `hours` (range <00 , 23>), `days` (range <01 , 31>), `months` (range <01, 12>), `weekdays` (0 - unknown, 1 - Monday, 7 - Sunday)
Notes	

ab08x5_get_[time/date]_alarm()

Function Name	`void ab08x5_get_time_alarm();` `void ab08x5_get_date_alarm();`
Function Description	Updates the time/date alarm values in RTC structure.
Parameters	None
Return Values	None
Notes	

ab08x5_set_[alarm field]_alarm()

Function Name	`ab08x5_err_t ab08x5_set_hundredths_alarm(uint8_t hundredths_alarm);`
	`ab08x5_err_t ab08x5_set_seconds_alarm(uint8_t seconds_alarm);`
	`ab08x5_err_t ab08x5_set_minutes_alarm(uint8_t minutes_alarm);`
	`ab08x5_err_t ab08x5_set_hours_alarm(uint8_t hours_alarm);`
	`ab08x5_err_t ab08x5_set_days_alarm(uint8_t days_alarm);`
	`ab08x5_err_t ab08x5_set_months_alarm(uint8_t months_alarm);`
	`ab08x5_err_t ab08x5_set_weekdays_alarm(uint8_t weekdays_alarm);`

Function Description	Sets value of hundredths / seconds / minutes / hours / days / months / weekdays.

© 2017 Dialog Semiconductor

Parameters	Corresponding value of `hundredths_alarm` value of hundredths to be set (range <00 , 99>) `seconds_alarm` value of seconds to be set (range <00 , 59>) `minutes_alarm` value of minutes to be set (range <00 , 59>) `hours_alarm` value of hours to be set (range <00, 23>) `days_alarm` value of days to be set (range <00, 31>) `months_alarm` value of months to be set (range <01, 12>) `weekdays_alarm` value of weekdays to be set (0 - unknown, 1 - Monday, 7 - Sunday)
Return Values	Error number `ab08x5_err_t`
Notes	

ab08x5_set_full_time_alarm()

Function Name	`ab08x5_err_t ab08x5_set_full_time_alarm(uint8_t hours_alarm, uint8_t minutes_alarm, uint8_t seconds_alarm, uint8_t hundredths_alarm);`
Function Description	Sets all time of alarm values.
Parameters	`hours_alarm` Hours value of alarm to be set `minutes_alarm` Minutes value of alarm to be set `seconds_alarm` Seconds value of alarm to be set `hundredths_alarm` Hundredths value of alarm to be set
Return Values	Error number `ab08x5_err_t`
Notes	

ab08x5_set_full_date_alarm()

Function Name	`ab08x5_err_t ab08x5_set_full_date_alarm(uint8_t months_alarm, uint8_t days_alarm, uint8_t weekdays_alarm);`
Function Description	Sets all date of alarm values.
Parameters	`years_alarm` Years value of alarm to be set `months_alarm` Months value of alarm to be set `days_alarm` Days value of alarm to be set `weekday_alarm` Weekday value of alarm to be set
Return Values	Error number `ab08x5_err_t`
Notes	

ab08x5_minute_leap()

Function Name	`bool ab08x5_minute_leap();`
Function Description	Checks if minutes have changed.
Parameters	None
Return Values	True if minutes changed
Notes	

ab08x5_get_status_register()

Function Name	`uint8_t ab08x5_get_status_register();`
Function Description	Gets the status register value.
Parameters	None
Return Values	Value of status register.

© 2017 Dialog Semiconductor

Notes	Status register bits:
	`RTC_STAT_REG_EX1 (1 << 0)`
	`RTC_STAT_REG_EX2 (1 << 1)`
	`RTC_STAT_REG_ALM (1 << 2)`
	`RTC_STAT_REG_TIM (1 << 3)`
	`RTC_STAT_REG_BL (1 << 4)`
	`RTC_STAT_REG_WDT (1 << 5)`
	`RTC_STAT_REG_BAT (1 << 6)`
	`RTC_STAT_REG_CB (1 << 7)`

ab08x5_get_alarm_status()

Function Name	`bool ab08x5_get_alarm_status(void);`
Function Description	Get the status of the alarm from status register
Parameters	None
Return Values	True if alarm event occurred
Notes	

ab08x5_[SQW/rtc_tick]_[enable/disable]()

Function Name	`void ab08x5_SQW_enable();`
	`void ab08x5_SQW_disable();`
	`void ab08x5_rtc_tick_enable();`
	`void ab08x5_rtc_tick_disable();`

Function Description	Enables / disables the SQW signal for toggling LCD or the 2 s interrupt from RTC.
Parameters	None
Return Values	None
Notes	

ab08x5_rtc_tick_is_enabled

Function Name	`bool ab08x5_rtc_tick_is_enabled();`
Function Description	Gets the state of the 2 s interrupt from RTC.
Parameters	None
Return Values	True if RTC tick interrupt is enabled.
Notes	

ab08x5_alarm_[rpt/mode]_config()

Function Name	`void ab08x5_alarm_rpt_config(ab08x5_airq_rpt_t alarm_rpt);` `void ab08x5_alarm_mode_config(ab08x5_alarm_mode_t mode);`
Function Description	Configure the matching of the alarm to the current time/date
Parameters	`Coresponding value of` `alarm_rpt` Repeat of alarm `mode` Select if repeated or one shot
Return Values	None
Notes	

ab08x5_get_alarm_[rpt/mode]_config()

© 2017 Dialog Semiconductor

Function Name	`ab08x5_airq_rpt_t ab08x5_get_alarm_rpt_config(void);` `ab08x5_alarm_mode_t ab08x5_get_alarm_mode_config(void);`
Function Description	Get configuration of alarm matching the current time/date
Parameters	None

Return Values	Coresponding configuration
	`ab08x5_airq_rpt_t`
	`ab08x5_alarm_mode_t`

Notes	

ab08x5_get_timestamp()

Function Name	`uint32_t ab08x5_get_timestamp();`
Function Description	Creates the UNIX timestamp from RTC values.
Parameters	None
Return Values	UNIX timestamp
Notes	

© 2017 Dialog Semiconductor

10.1.7 LS013B7DH03 Display Driver

The LS013B7DH03 display driver module is the bottom layer of communication with the LCD display. The module contains a set of API functions that allow full control of the display. Brief descriptions for each function can be found in the next section. The display driver is used by the user interface task via the graphics display interface (GDI), which directly controls it and uses the driver's functionality.

For more details about the LS013B7DH03 LCD display please see the LS013B7DH03 Datasheet [8].

A list of all API functions provided by the LS013B7DH03 driver and presented in this section follows:

Table 40: LS013B7DH03 driver API

Functions	
ls013b7dh03_init() ls013b7dh03_clear()	ls013b7dh03_[enable/disable]() ls013b7dh03_refresh()
Callback Functions	
bus_write()	set_enable_gpio()

© 2017 Dialog Semiconductor

10.1.7.1 Functions

ls013b7dh03_init()

Function Name	`void ls013b7dh03_init(const ls013b7dh03_t *handle);`
Function Description	Initializes display driver.
Parameters	handle — Instance of the ls013b7dh03 driver
Return Values	None
Notes	

ls013b7dh03_[enable/disable]()

Function Name	`void ls013b7dh03_enable(void);` `void ls013b7dh03_disable(void);`
Function Description	Turns display on/off
Parameters	None
Return Values	None
Notes	

ls013b7dh03_clear()

Function Name	`void ls013b7dh03_clear(void);`
Function Description	Clears all data from pixels memory. Display will revert to its normal white color.
Parameters	None
Return Values	None
Notes	

ls013b7dh03_refresh()

Function Name	`void ls013b7dh03_refresh(uint8_t *pixel_data, uint8_t startRow, uint8_t height);`
Function Description	Transfers frame buffer data to display memory
Parameters	`pixel_data` — Pointer to frame buffer data `start_row` — Start line of the display `height` — Number of lines to transfer
Return Values	None
Notes	

10.1.7.2 Callback Functions

bus_write()

Function Name	`void bus_write(uint8_t reg, uint8_t *value, size_t length);`
Function Description	Transfers frame buffer data to display memory
Parameters	`reg` — Pointer to frame buffer data `value` — Start line of the display `length` — Number of lines to transfer
Return Values	None

© 2017 Dialog Semiconductor

Notes	

set_enable_gpio()

Function Name	`void ls013b7dh03_refresh(uint8_t *pixel_data, uint8_t startRow, uint8_t height);`	
Function Description	Transfers frame buffer data to display memory	
Parameters	`pixel_data`	Pointer to frame buffer data
	`start_row`	Start line of the display
	`height`	Number of lines to transfer
Return Values	None	
Notes		

10.1.8 FXL6408 GPIO Expander Driver

The FXL6408 GPIO expander driver API provides functions for the configuration and control of the GPIO expansion provided to the DA14681 chip for the connection and control of more peripheral devices than the DA14681 can support: NFC, MIC (enable/disable), Display (enable/disable) and HRM option 2 (enable/disable) based on the schematic in [9]). Below the functions are presented which are used by the Wearable project, while a complete list of the functions is presented in file `fxl6408.h` based on the datasheet [13] provided by Fairchild Semiconductor.

General guidelines for using the FXL6408 GPIO expander driver are as follows:

- Initialize the driver and the peripheral device module using the function `fxl6408_init()`. Contrary to Bosch sensor module drivers, this function internally creates the appropriate instance handle for the module, which includes references to the I2C bus access functions used for reading/writing values to the registers. Also internally defined are the device address and the functions which implement I2C bus access and introduce delay. Alternatively, the function `fxl6408_initialize_module()` can be used for initializing the instance handle for the module in the same way as it is done in the case of Bosch sensor module drivers presented earlier.
- Use the functions `fxl6408_dev_open()` and `fxl6408_dev_close()` to initiate and terminate FXL6408 device access, respectively, each time a set of the driver API functions needs to be called. The driver uses I2C bus adapter API functions provided by the SDK accompanying the DA14681 chip to implement I2C bus access, so that the latter can be shared with other devices (i.e. DI5115, SX9300 and AB09X5).
- Configure the peripheral device module in terms of specific setting/operation features (`setting_feature = gpio_input_pull_up_down_mode, gpio_io_direction, gpio_output_state`, etc.) using the `fxl6408_set_[setting_feature]()` set of functions.
- When the current configuration state of the peripheral device module is to be checked in terms of specific setting/operation features, use the `fxl6408_get_[setting_feature]()` set of functions.
- Access the peripheral device module's registers (`reg_name`) when low level control and data handling is required, using the `fxl6408_[set/get]_reg_[reg_name]()` set of functions.

A list of all API data structures/types and functions provided by the FXL6408 GPIO expander driver and presented in this section follows:

Table 41: FXL6408 Driver API

Data Structures and Types	
fxl6408_t	
Dependency Functions	
fxl6408_t -> bus_write() fxl6408_t -> delay_msec()	fxl6408_t -> bus_read()

© 2017 Dialog Semiconductor

Initialization and Basic Configuration Functions	
fxl6408_initialize_module() fxl6408_dev_open()	fxl6408_init() fxl6408_dev_close()
Command Functions	
fxl6408_soft_reset() fxl6408_clear_gpio_intr()	fxl6408_clear_rst_intr()
State Functions	
fxl6408_get_mf_id() fxl6408_get_gpio_intr_stat()	fxl6408_get_rst_intr_stat()
Configuration Functions	
fxl6408_set_gpio_io_direction() fxl6408_set_gpio_output_high_z_enable() fxl6408_set_gpio_input_pull_up_down_enable() fxl6408_set_gpio_configuration()	fxl6408_set_gpio_output_state() fxl6408_set_gpio_input_default_state() fxl6408_set_gpio_input_pull_up_down_mode()
Low Level Register Access Functions	
fxl6408_[set/get]_reg_[reg_name]()	

10.1.8.1 Data Structures and Types

fxl6408_t

Data Structure fields	Type	Description
mf_id	uint8_t	Manufacturer ID of the FXL6408 peripheral device module.
dev_addr	uint8_t	Device address of the FXL6408 peripheral device module.
dev	i2c_device	The device handle to use with I2C adapter API functions (ad_12c.h) provided by the SDK.
bus_write	Function pointer	Function for I2C bus write-access.
bus_read	Function pointer	Function for I2C bus read-access.
delay_msec	Function pointer	Function for introducing delay (in ms).

© 2017 Dialog Semiconductor

10.1.8.2 Dependency Functions

fxl6408_t -> bus_write()

Function Name	`int8_t (*bus_write)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data)`
Function Description	Implements write-access over the I2C bus for writing data to FXL6408 peripheral device module registers.
Parameters	`dev_addr` the device address `reg_addr` the address of the register to write `reg_data` the data to write
Return Values	0 for successful execution; ≠0 for error
Notes	Writing 1 byte.

fxl6408_t -> bus_read()

Function Name	`int8_t (*bus_read)(uint8_t dev_addr, uint8_t reg_addr, uint8_t* reg_data)`
Function Description	Implements read-access over the I2C bus for reading data from FXL6408 peripheral device module registers.
Parameters	`dev_addr` the device address `reg_addr` the address of the register to read `reg_data` the buffer to put register data being read to
Return Values	0 for successful execution; ≠0 for error
Notes	Reading 1 byte.

© 2017 Dialog Semiconductor

fxl6408_t -> delay_msec()

Function Name	`void (*delay_msec)(uint32_t ms)`
Function Description	Implements a delay in ms that may be necessary for the FXL6408 peripheral device module driver functionality implementation.
Parameters	`ms` the delay (in ms) to be introduced
Return Values	None
Notes	

10.1.8.3 Initialization and Basic Configuration Functions

fxl6408_initialize_module()

Function Name	`int8_t fxl6408_initialize_module(fxl6408_t* s_fxl6408)`
Function Description	(INIT-CONFIG) Initializes the FXL6408 peripheral device module, creating the appropriate instance handle for the module which includes references to the I2C bus access functions used for reading/writing values to the registers, and soft-resetting the module.
Parameters	`s_fxl6408` the handle structure for FXL6408 driver
Return Values	0 for successful execution; ≠0 for error
Notes	

fxl6408_init()

Function Name	`int8_t` fxl6408_`init(void)`
Function Description	(INIT-CONFIG) Initializes the FXL6408 peripheral device module internally.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

fxl6408_dev_open()

Function Name	`int8_t fxl6408_dev_open(void)`
Function Description	(INIT-CONFIG) Initiates FXL6408 device access for the driver API functions to be called. initiate and terminate FXL6408 device access, respectively, each time a set of the driver API functions need to be called
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

fxl6408_dev_close()

Function Name	`int8_t fxl6408_dev_close(void)`
Function Description	(INIT-CONFIG) Terminates FXL6408 device access after calling a set of the driver API functions.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

10.1.8.4 Command Functions

fxl6408_soft_reset()

Function Name	`int8_t fxl6408_soft_reset(void)`
Function Description	(CMD) Performs a soft reset to the FXL6408 peripheral device module.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

fxl6408_clear_rst_intr()

Function Name	`int8_t fxl6408_clear_rst_intr(void)`
Function Description	(CMD) Clears reset interrupt.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

fxl6408_clear_gpio_intr()

Function Name	`int8_t fxl6408_clear_gpio_intr(void)`
Function Description	(CMD) Clears GPIO interrupt.
Parameters	None
Return Values	0 for successful execution; ≠0 for error
Notes	

10.1.8.5 State Functions

fxl6408_get_mf_id()

Function Name	`int8_t fxl6408_get_mf_id(uint8_t* v_mf_id_u8)`
Function Description	(STATE) Reads the manufacturer id of the FXL6408 peripheral device module.
Parameters	`v_mf_id_u8` the manufacturer id
Return Values	0 for successful execution; ≠0 for error
Notes	

fxl6408_get_rst_intr_stat()

Function Name	`int8_t fxl6408_get_rst_intr_stat(uint8_t* v_rst_intr_u8)`
Function Description	(STATE) Reads the status of the reset interrupt status bit, indicating whether the device has been reset and the default values are set.
Parameters	`v_rst_intr_u8` the value of the reset interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

fxl6408_get_gpio_intr_stat()

Function Name	`int8_t fxl6408_get_gpio_intr_stat(uint8_t v_gpio_u8, uint8_t* v_gpio_intr_u8)`
Function Description	(STATE) Reads the status of a specific GPIO interrupt status bit, indicating whether a GPIO input pin has changed state from default.
Parameters	`v_gpio_u8` the value of the GPIO pin `v_gpio_intr_u8` the value of the GPIO interrupt status bit
Return Values	0 for successful execution; ≠0 for error
Notes	

10.1.8.6 Configuration Functions

fxl6408_set_gpio_io_direction()

Function Name	`int8_t fxl6408_set_gpio_io_direction(uint8_t v_gpio_u8, uint8_t v_gpio_io_dir_u8)`
Function Description	(CONFIG) Sets input/output direction of a specific GPIO pin.

Parameters	`v_gpio_u8` the value of the GPIO pin
	`FXL6408_GPIO_0, FXL6408_GPIO_1, FXL6408_GPIO_2, FXL6408_GPIO_3, FXL6408_GPIO_4, FXL6408_GPIO_5, FXL6408_GPIO_6, FXL6408_GPIO_7`
	`v_gpio_io_dir_u8` the value of the GPIO pin input/output direction
	`FXL6408_GPIO_INPUT, FXL6408_GPIO_OUTPUT`

Return Values	0 for successful execution; ≠0 for error
Notes	

fxl6408_set_gpio_output_state()

Function Name	`int8_t fxl6408_set_gpio_output_state(uint8_t v_gpio_u8, uint8_t v_gpio_out_stat_u8)`
Function Description	(CONFIG) Sets the state of a specific GPIO pin configured as output.

Parameters	`v_gpio_u8` the value of the GPIO pin
	`FXL6408_GPIO_0, FXL6408_GPIO_1, FXL6408_GPIO_2, FXL6408_GPIO_3, FXL6408_GPIO_4, FXL6408_GPIO_5, FXL6408_GPIO_6, FXL6408_GPIO_7`
	`v_gpio_out_stat_u8` the value of the GPIO pin output state
	`FXL6408_GPIO_LOW, FXL6408_GPIO_HIGH`

Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

fxl6408_set_gpio_output_high_z_enable()

Function Name	`int8_t fxl6408_set_gpio_output_high_z_enable(uint8_t v_gpio_u8, uint8_t v_gpio_out_high_z_enable_u8)`
Function Description	(CONFIG) Sets the state of high-z enable status bit for a specific GPIO pin configured as output.

Parameters	`v_gpio_u8` the value of the GPIO pin
	`FXL6408_GPIO_0, FXL6408_GPIO_1, FXL6408_GPIO_2, FXL6408_GPIO_3, FXL6408_GPIO_4, FXL6408_GPIO_5, FXL6408_GPIO_6, FXL6408_GPIO_7`
	`v_gpio_out_high_z_enable_u8` the value of the GPIO pin output high-z state enable bit status
	`FXL6408_DISABLE, FXL6408_ENABLE`

Return Values	0 for successful execution; ≠0 for error
Notes	

fxl6408_set_gpio_input_default_state()

Function Name	`int8_t fxl6408_set_gpio_input_default_state(uint8_t v_gpio_u8, uint8_t v_gpio_in_dflt_stat_u8)`
Function Description	(CONFIG) Sets the default state for a specific GPIO pin configured as input.

Parameters	`v_gpio_u8` the value of the GPIO pin
	`FXL6408_GPIO_0, FXL6408_GPIO_1, FXL6408_GPIO_2, FXL6408_GPIO_3, FXL6408_GPIO_4, FXL6408_GPIO_5, FXL6408_GPIO_6, FXL6408_GPIO_7`
	`v_gpio_in_dflt_stat_u8` the value of the GPIO pin input default state
	`FXL6408_GPIO_LOW, FXL6408_GPIO_HIGH`

Return Values	0 for successful execution; ≠0 for error
Notes	

fxl6408_set_gpio_input_pull_up_down_enable()

Function Name	`int8_t fxl6408_set_gpio_input_pull_up_down_enable(uint8_t v_gpio_u8, uint8_t v_gpio_in_pull_up_down_enable_u8)`
Function Description	(CONFIG) Sets the state of pull-up or pull-down enable status bit for a specific GPIO pin configured as input.

Parameters	`v_gpio_u8` the value of the GPIO pin
	`FXL6408_GPIO_0, FXL6408_GPIO_1, FXL6408_GPIO_2, FXL6408_GPIO_3, FXL6408_GPIO_4, FXL6408_GPIO_5, FXL6408_GPIO_6, FXL6408_GPIO_7`
	`v_gpio_in_pull_up_down_enable_u8` the value of the GPIO pin input pull-up or pull-down enable bit status
	`FXL6408_DISABLE, FXL6408_ENABLE`

Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

fxl6408_set_gpio_input_pull_up_down_mode()

Function Name	`int8_t fxl6408_set_gpio_input_pull_up_down_mode(uint8_t v_gpio_u8, uint8_t v_gpio_in_pull_up_down_mode_u8)`
Function Description	(CONFIG) Sets pull-up or pull-down mode selection for a specific GPIO pin configured as input.

Parameters	`v_gpio_u8` the value of the GPIO pin
	`FXL6408_GPIO_0, FXL6408_GPIO_1, FXL6408_GPIO_2, FXL6408_GPIO_3, FXL6408_GPIO_4, FXL6408_GPIO_5, FXL6408_GPIO_6, FXL6408_GPIO_7`
	`v_gpio_in_pull_up_down_mode_u8` the value of the GPIO pin input pull-up or pull-down mode
	`FXL6408_GPIO_PULL_DOWN, FXL6408_GPIO_PULL_UP`

Return Values	0 for successful execution; ≠0 for error
Notes	

fxl6408_set_gpio_configuration()

Function Name	`int8_t fxl6408_set_gpio_configuration(uint8_t v_gpio_u8, uint8_t v_gpio_io_mode_u8, uint8_t v_gpio_io_stat_u8)`
Function Description	(CONFIG) Sets the configuration of a specific GPIO pin in terms of the input/output direction type, the mode and the default/initial state.

Parameters	`v_gpio_u8` the value of the GPIO pin
	`FXL6408_GPIO_0, FXL6408_GPIO_1, FXL6408_GPIO_2, FXL6408_GPIO_3, FXL6408_GPIO_4, FXL6408_GPIO_5, FXL6408_GPIO_6, FXL6408_GPIO_7`
	`v_gpio_io_mode_u8` the value of the GPIO pin mode
	`FXL6408_GPIO_INPUT, FXL6408_GPIO_OUTPUT, FXL6408_GPIO_INPUT_PULLDOWN, FXL6408_GPIO_INPUT_PULLUP`
	`v_gpio_io_stat_u8` the value of the GPIO pin default/initial state
	`FXL6408_GPIO_LOW, FXL6408_GPIO_HIGH, FXL6408_GPIO_HIGH_Z`

Return Values	0 for successful execution; ≠0 for error
Notes	

© 2017 Dialog Semiconductor

10.1.8.7 Low Level Register Access Functions

fxl6408_[set/get]_reg_[reg_name]()

Function Name	`int8_t fxl6408_get_reg_intr_stat(uint8_t *v_reg_value_u8)`
	`int8_t fxl6408_set_reg_intr_mask(uint8_t v_reg_value_u8)`
	`int8_t fxl6408_get_reg_intr_mask(uint8_t *v_reg_value_u8)`
	`int8_t fxl6408_get_reg_in_stat(uint8_t *v_reg_value_u8)`
	`int8_t fxl6408_set_reg_pull_dn_pull_up(uint8_t v_reg_value_u8)`
	`int8_t fxl6408_get_reg_pull_dn_pull_up(uint8_t *v_reg_value_u8)`
	`int8_t fxl6408_set_reg_pull_en(uint8_t v_reg_value_u8)`
	`int8_t fxl6408_get_reg_pull_en(uint8_t *v_reg_value_u8)`
	`int8_t fxl6408_set_reg_in_dflt_stat(uint8_t v_reg_value_u8)`
	`int8_t fxl6408_get_reg_in_dflt_stat(uint8_t *v_reg_value_u8)`
	`int8_t fxl6408_set_reg_out_high_z(uint8_t v_reg_value_u8)`
	`int8_t fxl6408_get_reg_out_high_z(uint8_t *v_reg_value_u8)`
	`int8_t fxl6408_set_reg_out_stat(uint8_t v_reg_value_u8)`
	`int8_t fxl6408_get_reg_out_stat(uint8_t *v_reg_value_u8)`
	`int8_t fxl6408_set_reg_io_dir(uint8_t v_reg_value_u8)`
	`int8_t fxl6408_get_reg_io_dir(uint8_t *v_reg_value_u8)`

Function Description	(LOW-ACCESS) Sets/gets the register [reg_name] value based on the register names presented in FXL6408 datasheet Register Map.
Parameters	`v_reg_value_u8` the value of the register [reg_name]
Return Values	0 for successful execution; ≠0 for error
Notes	Direct write- or read-access to FXL6408 registers, without any further processing (i.e. bit- masking and shifting) or delay (if any)

© 2017 Dialog Semiconductor

Revision History

Revision	Date	Description
1.0	08-Feb-2017	Initial version.

© 2017 Dialog Semiconductor

DA14681 Wearable Development Kit API

Status Definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

Disclaimer

Information in this document is believed to be accurate and reliable. However, Dialog Semiconductor does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information. Dialog Semiconductor furthermore takes no responsibility whatsoever for the content in this document if provided by any information source outside of Dialog Semiconductor.

Dialog Semiconductor reserves the right to change without notice the information published in this document, including without limitation the specification and the design of the related semiconductor products, software and applications.

Applications, software, and semiconductor products described in this document are for illustrative purposes only. Dialog Semiconductor makes no representation or warranty that such applications, software and semiconductor products will be suitable for the specified use without further testing or modification. Unless otherwise agreed in writing, such testing or modification is the sole responsibility of the customer and Dialog Semiconductor excludes all liability in this respect.

Customer notes that nothing in this document may be construed as a license for customer to use the Dialog Semiconductor products, software and applications referred to in this document. Such license must be separately sought by customer with Dialog Semiconductor.

All use of Dialog Semiconductor products, software and applications referred to in this document are subject to Dialog Semiconductor's Standard Terms and Conditions of Sale, available on the company website (www.dialog-semiconductor.com) unless otherwise stated.

Dialog and the Dialog logo are trademarks of Dialog Semiconductor plc or its subsidiaries. All other product or service names are the property of their respective owners.

© 2017 Dialog Semiconductor. All rights reserved.

Contacting Dialog Semiconductor

United Kingdom (Headquarters)
Dialog Semiconductor (UK) LTD
Phone: +44 1793 757700

Germany
Dialog Semiconductor GmbH
Phone: +49 7021 805-0

The Netherlands
Dialog Semiconductor B.V.
Phone: +31 73 640 8822

Email:
enquiry@diasemi.com

North America
Dialog Semiconductor Inc.
Phone: +1 408 845 8500

Japan
Dialog Semiconductor K. K.
Phone: +81 3 5425 4567

Taiwan
Dialog Semiconductor Taiwan
Phone: +886 281 786 222

Web site:
www.dialog-semiconductor.com

Singapore
Dialog Semiconductor Singapore
Phone: +65 64 8499 29

Hong Kong
Dialog Semiconductor Hong Kong
Phone: +852 3769 5200

Korea
Dialog Semiconductor Korea
Phone: +82 2 3469 8200

China (Shenzhen)
Dialog Semiconductor China
Phone: +86 755 2981 3669

China (Shanghai)
Dialog Semiconductor China
Phone: +86 21 5424 9058