AS7050 EVK Firmware

API Documentation
revision v3.2.0

Generated by Doxygen

Contents

1 AS7050 Core Firmware

1.1 Introduction

This document provides an overview on how the vital signs sensor family AS7050 can be controlled by a host PC.

The API provides direct access to the AS7050 Vital Signs Controller. Furthermore, it allows Bio Applications such as Heart Rate Monitor (HRM) and Pulse Rate Variability (PRV) to be controlled and read out.

Key features:

- Usage of ams standard software components
- Low-level access to Vital Signs Controller
- Direct configuration of Bio Apps
- Asynchronous sending of measurement results

The documentation is split into two sections:

- The overview section describes which modules are used and how they interact.
- The module section lists all functions and parameters which can be used.

1.2 Overview

This firmware provides low-level access to the AS7050 Vital Signs Controller and all supported Bio Apps. For easy code reuse and to reduce the integration effort for customers to a minimum, the interface is kept simple.

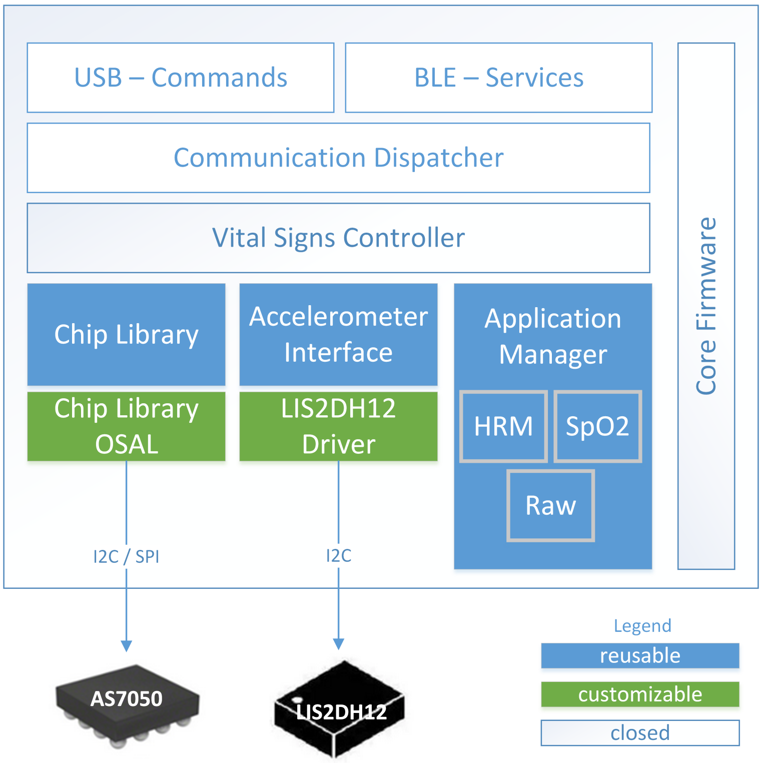

Figure 1 Structure Overview

- *USB - Commands:* All functions of the Vital Signs Controller can be called directly via USB
- *BLE - Services:* All functions of the Vital Signs Controller can be called directly via BLE
- *Communication Dispatcher:* This module manages the different interfaces and provides the interface to the Vital Signs Controller
- *Vital Signs Controller:* This software component connects the Chip Library, accelerometer interface and application manager
- *Accelerometer Interface:* Generic interface to connect any accelerometer to the Vital Signs Controller
- *LIS2DH12 Driver:* Example driver for a accelerometer which is used on the EVK
- *Application Manager:* This component provides a generic interface for all Bio Apps
- *Chip Library:* Chip Library for AS7050 which encapsulates all I2C register interactions to a simple interface
- *OSAL:* Operating System Abstraction Layer of the Chip Library which moves all platform-dependent functions to a separate layer, which can be exchanged without adaption of the application
- *Core Firmware:* Firmware base which provides lots of features like I2C/SPI interfaces, system initialisation and USB communication

1.3 Acronyms and Abbreviations

1.3.1 Acronyms

API = Application Programming Interface
I2C = Inter-Integrated Circuit
FIFO = First in, first out
HRM = Heart rate monitor
PRV = Pulse Rate Variability
OSAL = Operating System Abstraction Layer
SPI = Serial Peripheral Interface
USB = Universal Serial Bus

2 Module Index

2.1 Modules

Here is a list of all modules:

3 Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

4 Module Documentation

4.1 AS7050 Vital Signs Controller USB Commands

Description of the application dependent messages.

Enumerations

- enum E_CMD_ID {
  CMD_ID_VSC_INITIALIZE = 100,
  CMD_ID_VSC_SHUTDOWN = 101,
  CMD_ID_VSC_CL_SET_REG_GROUP = 102,
  CMD_ID_VSC_CL_GET_REG_GROUP = 103,
  CMD_ID_VSC_CL_SET_AGC_CONFIG = 125,
  CMD_ID_VSC_CL_GET_AGC_CONFIG = 126,
  CMD_ID_VSC_CL_WRITE_REGISTER = 106,
  CMD_ID_VSC_CL_READ_REGISTER = 107,
  CMD_ID_VSC_CL_GET_MEAS_CONFIG = 108,
  CMD_ID_VSC_START = 111,
  CMD_ID_VSC_STOP = 112,
  CMD_ID_VSC_AM_SET_SIGNAL_ROUTING = 127,
  CMD_ID_VSC_AM_ENABLE_APPS = 128,
  CMD_ID_VSC_AM_APP_CONFIG = 129,
  CMD_ID_VSC_AM_APP_OUTPUT = 130,
  CMD_ID_VSC_GET_VERSION = 115,
  CMD_ID_VSC_MEAS_ERROR = 122,
  CMD_ID_VSC_DAC_REF_CALC_CTRL = 131,
  CMD_ID_VSC_DAC_REF_CALC_RESULT = 132,
  CMD_ID_VSC_ACC_SET_SAMPLE_PERIOD = 133,
  CMD_ID_VSC_ACC_GET_SAMPLE_PERIOD = 134 }

Functions

- const prt_table_entry_t ∗ cmd_vital_signs_get_table (uint32_t ∗p_num)

  Returns the supported command table and entry number.
- err_code_t cmd_vital_signs_send_appmgr_output (as7050_appmgr_app_id_t app_id, err_code_t error, const uint8_t ∗p_data, uint16_t size)

  Sends application manager output to the host application.
- err_code_t cmd_vital_signs_send_dac_ref (err_code_t error, uint32_t dac_ref)

  Sends the result of a DAC reference calculation to the host application.
- err_code_t cmd_vital_signs_send_meas_error (err_code_t error)

  Sends measurement error to host application.

4.1.1 Detailed Description

Description of the application dependent messages.

These messages will be used to control the measurements remotely. Supports direct control of the vital signs controller.

4.1.2 Enumeration Type Documentation

4.1.2.1 E_CMD_ID `enum E_CMD_ID`

Supported Chip-Lib message IDs and further configuration for BIO Apps

Enumerator

CMD_ID_VSC_INITIALIZE	This command calls the function as7050_vsc_initialize directly. **Input** [Payload-Size: 0 - ...]: • *interface_description:* Optional parameter which will be commited to the OSAL of the ChipLib. **Output** [Payload-Size: 0 byte]
CMD_ID_VSC_SHUTDOWN	This command calls the function as7050_vsc_shutdown directly. **Input** [Payload-Size: 0 byte] **Output** [Payload-Size: 0 byte]

Enumerator

CMD_ID_VSC_CL_SET_REG_GROUP	This command calls the function as7050_vsc_cl_set_reg_group directly. **Input** [Payload-Size: 1 - ...]: • *reg_group_id:* Size uint8. See as7050_reg_group_ids • *reg_group_data:* Size uint8 ∗ (payload_size - 1) See parameter definition of as7050_vsc_cl_set_reg_group **Output** [Payload-Size: 0 byte]
CMD_ID_VSC_CL_GET_REG_GROUP	This command calls the function as7050_vsc_cl_get_reg_group directly. **Input** [Payload-Size: 2 byte]: • *reg_group_id:* Size uint8. See as7050_reg_group_ids • *reg_group_size:* Size uint8. See parameter definition of as7050_vsc_cl_get_reg_group **Output** [Payload-Size: ...]: • *reg_group_data:* Size uint8 ∗ (reg_group_size). For parameter format see as7050_vsc_cl_get_reg_group definition
CMD_ID_VSC_CL_SET_AGC_CONFIG	This command calls the function as7050_vsc_cl_set_agc_config directly. **Input** [Payload Size: Multiple of sizeof(::agc_configuration_t)]: • *agc_config:* Multiple of sizeof(::agc_configuration_t). Array of AGC configurations, see ::agc_configuration_t. All previous AGC configurations are discarded when sending this command. **Output** [Payload Size: 0 byte]

Enumerator

CMD_ID_VSC_CL_GET_AGC_CONFIG	This command calls the function as7050_vsc_cl_get_agc_config directly. **Input** [Payload Size: 0 byte] **Output** [Payload Size: Multiple of sizeof(::agc_configuration_t)]: • *agc_config:* Multiple of sizeof(::agc_configuration_t). Array of AGC configurations, see ::agc_configuration_t. A configuration is provided for each enabled AGC instance.
CMD_ID_VSC_CL_WRITE_REGISTER	This command calls the function as7050_vsc_cl_write_register directly. **Input** [Payload-Size: 2 byte]: • *reg_address:* Size uint8. Register address • *reg_value:* Size uint8. New register value **Output** [Payload-Size: 0 byte]
CMD_ID_VSC_CL_READ_REGISTER	This command calls the function as7050_vsc_cl_read_register directly. **Input** [Payload-Size: 1 byte]: • *reg_address:* Size uint8. Register address **Output** [Payload-Size: 1 byte]: • *reg_value:* Size uint8. Actual register value

Enumerator

CMD_ID_VSC_CL_GET_MEAS_CONFIG	This command calls the function as7050_vsc_cl_get_measurement_config directly. **Input** [Payload-Size: 0 byte] **Output** [Payload-Size: sizeof(as7050_meas_config_t)]: • *ppg_sample_period_us:* uint32, Sample period of PPG signals in microseconds • *ecg_sample_period_us:* uint32, Sample period of ECG signals in microseconds • *max_adc_count:* uint32, Maximum possible ADC count for the current configuration • *fifo_map:* uint16, Definition which channels are mapped inside FIFO. See as7050_channel_flags_t • *fifo_threshold:* uint16, FIFO threshold, when data shall be read • *sample_size:* uint8_t, 3 or 4 bytes for every sample • *max_adc_bit_width:* uint8_t, Maximum bit width of the ADC count for the current configuration • *reserved:* uint16, For alignment only, Do not use. Always set to 0
CMD_ID_VSC_START	This command calls the function as7050_vsc_start_measurement directly. **Input** [Payload-Size: 0 byte] **Output** [Payload-Size: 0 byte]
CMD_ID_VSC_STOP	This command calls the function as7050_vsc_stop_measurement directly. **Input** [Payload-Size: 0 byte] **Output** [Payload-Size: 0 byte]
CMD_ID_VSC_AM_SET_SIGNAL_ROUTING	This command calls the function as7050_vsc_am_set_signal_routing directly. **Input** [Payload Size: at least 1 byte]: • *app_id:* uint8. Identifier of the app to configure, see ::as7050_appmgr_app_id_t. • *sample_ids:* App-specific number of signals $*$ uint8. Signal routing for the given app, see AS7050_APPMGR_APPS and as7050_channel_t.

Enumerator

CMD_ID_VSC_AM_ENABLE_APPS	This command calls the function as7050_vsc_am_enable_apps directly. **Input** [Payload Size: 4 byte]: • *enabled_apps:* uint32. Flags of enabled Vital Signs Applications, see AS7050_APPMGR_APP_FLAG.
CMD_ID_VSC_AM_APP_CONFIG	This command calls the function ::as7056_57_vsc_am_configure_app directly. **Input** [Payload Size: at least 1 byte]: • *app_id:* uint8. Identifier of the app to configure, see ::as7050_appmgr_app_id_t. • *configuration:* App-specific structure, see AS7050_APPMGR_APPS.
CMD_ID_VSC_AM_APP_OUTPUT	This command is used to send output of the Vital Signs Applications asynchronously. Note This ID is only used to send data asynchronously. This command is never sent from the host to AS7050 hardware. **Output** [Payload Size: at least 1 byte]: • *app_id:* uint8. Identifier of the app that generated the output, see ::as7050_appmgr_app_id_t. • *output:* App-specific data, see AS7050_APPMGR_APPS.
CMD_ID_VSC_GET_VERSION	This command calls the function as7050_vsc_get_version directly. **Output** [Payload-Size: 3 byte]: See as7050_version_t
CMD_ID_VSC_MEAS_ERROR	It will be used to send measurement errors acyclic. Error code will be transferred inside the message header Note ID is not supported for a host request! **Output** [Payload-Size: 0 byte]

Enumerator

CMD_ID_VSC_DAC_REF_CALC_CTRL	This command starts or aborts the calculation of the DAC reference value, which is used for GSR measurement. This command calls as7050_vsc_dac_ref_calc_start or as7050_vsc_dac_ref_calc_abort. A message with command ID CMD_ID_VSC_DAC_REF_CALC_RESULT containing the DAC reference value is sent after the calculation finished successfully. **Input** [Payload-Size: 3 byte] • *control:* uint8. DAC reference value calculation is started when this value is 0. Calculation is aborted when this value is 1. All other values are illegal. • *sample_cnt:* uint16. Minimum number of samples that are acquired per DAC to calculate the DAC reference value. This argument is optional when a calculation is aborted. **Output** [Payload-Size: 0 byte]
CMD_ID_VSC_DAC_REF_CALC_RESULT	This command is sent after the calculation of the DAC reference value finished. **Note** ID is not supported for a host request! **Output** [Payload-Size: 4 byte] • *dac_ref:* uint32. DAC reference value.
CMD_ID_VSC_ACC_SET_SAMPLE_PERIOD	This command calls the function as7050_vsc_acc_set_sample_period directly. Supported sample periods of the LIS2HD12 driver: 1s (1hz), 100ms (10hz), 40ms (25hz), 20ms (50hz), 10ms (100hz) and 5ms (200hz) **Input** [Payload Size: 4 byte]: • *sample_period:* uint32. Sample period of the accelerometer in microseconds

Enumerator

CMD_ID_VSC_ACC_GET_SAMPLE_PERIOD	This command calls the function as7050_vsc_acc_get_sample_period directly. Supported sample periods of the LIS2HD12 driver: 1s (1hz), 100ms (10hz), 40ms (25hz), 20ms (50hz), 10ms (100hz) and 5ms (200hz) **Output** [Payload Size: 4 byte]: • *sample_period:* uint32. Sample period of the accelerometer in microseconds. Default 100000.

4.1.3 Function Documentation

4.1.3.1 cmd_vital_signs_get_table()

```
const prt_table_entry_t* cmd_vital_signs_get_table (
        uint32_t * p_num )
```

Returns the supported command table and entry number.

This table defines the callback functions, which will be used by the protocol handler

Parameters

in	*p_num*	Number of given table entries

Returns

Address, which points to the command table

4.1.3.2 cmd_vital_signs_send_appmgr_output()

```
err_code_t cmd_vital_signs_send_appmgr_output (
        as7050_appmgr_app_id_t app_id,
        err_code_t error,
        const uint8_t * p_data,
        uint16_t size )
```

Sends application manager output to the host application.

Parameters

in	*app_id*	Identification of the app. See ::as7050_appmgr_app_id_t.
in	*error*	Error message during data handling.
in	*p_data*	Pointer to the application-specific output.
in	*size*	Size of raw data in bytes.

Return values

ERR_SUCCESS	Function returns without error.

4.1.3.3 cmd_vital_signs_send_dac_ref()

```
err_code_t cmd_vital_signs_send_dac_ref (
        err_code_t error,
        uint32_t dac_ref )
```

Sends the result of a DAC reference calculation to the host application.

Parameters

in	*error*	Error that occurred during data handling
in	*dac_ref*	DAC reference value

Return values

ERR_SUCCESS	Function returns without error.

4.1.3.4 cmd_vital_signs_send_meas_error()

```
err_code_t cmd_vital_signs_send_meas_error (
        err_code_t error )
```

Sends measurement error to host application.

Parameters

in	*error*	Measurement error

Return values

ERR_SUCCESS	Function returns without error.

4.2 AS7050 Vital Signs Controller BLE Object IDs

Description of the objects which can be written/read by the BLE Object Access Service.

Data Structures

- struct obj_access_table_entry_t

Typedefs

- typedef enum E_OBJ_ID obj_id_t

Enumerations

- enum E_OBJ_ID {
  OBJ_ID_VSC_SIGNAL_ROUTING = 0x24,
  OBJ_ID_VSC_ENABLE_APPS = 0x25,
  OBJ_ID_VSC_REG_GROUP = 0x21,
  OBJ_ID_VSC_AGC_CONFIG = 0x26,
  OBJ_ID_VSC_REG_ACCESS = 0x04,
  OBJ_ID_VSC_MEAS_CONFIG = 0x23,
  OBJ_ID_VSC_VERSION = 0x06,
  OBJ_ID_VSC_RAW_APP_CONF = 0x07,
  OBJ_ID_VSC_HRM_APP_CONF = 0x08,
  OBJ_ID_VSC_SPO2_APP_CONF = 0x09,
  OBJ_ID_VSC_DAC_REF_CALC_CTRL = 0x27,
  OBJ_ID_VSC_GSR_APP_CONF = 0x28,
  OBJ_ID_VSC_ACC_SAMPLE_PERIOD = 0x29 }

Functions

- const obj_access_table_entry_t ∗ obj_vital_signs_get_table (uint32_t ∗p_num)

  returns the object table for vital signs

4.2.1 Detailed Description

Description of the objects which can be written/read by the BLE Object Access Service.

4.2.2 Typedef Documentation

4.2.2.1 obj_id_t `typedef enum E_OBJ_ID obj_id_t`

Supported BLE "Object Access Service" Object IDs

Note

> Objects that are identical on AS703x and AS7050 are assigned the same ID as on AS703x. Objects which differ from AS703x are assigned new IDs starting from 0x20.

4.2.3 Enumeration Type Documentation

4.2.3.1 E_OBJ_ID `enum E_OBJ_ID`

Supported BLE "Object Access Service" Object IDs

Note

> Objects that are identical on AS703x and AS7050 are assigned the same ID as on AS703x. Objects which differ from AS703x are assigned new IDs starting from 0x20.

Enumerator

OBJ_ID_VSC_SIGNAL_ROUTING	[Write-Only] Set the signal routing. Accessing this object calls the function ::as7050_vsc_set_signal_routing directly. **Index:** 0x24 **SubIndex:** Identifier of the app to configure, see ::as7050_appmgr_app_id_t. This command calls the function as7050_vsc_am_set_signal_routing directly. **Input** [Payload Size: variable]: • *sample_ids:* App-specific number of signals ∗ uint8. Signal routing for the given app, see AS7050_APPMGR_APPS and as7050_channel_t.
OBJ_ID_VSC_ENABLE_APPS	[Write-Only] Set the enabled BIO applications. Accessing this object calls the function ::as7050_vsc_enable_apps directly. **Index:** 0x25 **SubIndex:** Don't care **Data:** [Payload Size: 4 bytes] • *enable_apps:* Set flags for the enabled applications. See ::as7050_app_flags_t

Enumerator

OBJ_ID_VSC_REG_GROUP	Set/Get Register group. This command calls the function as7050_vsc_cl_set_reg_group or as7050_vsc_cl_get_reg_group directly, depending on whether it is a write or read request. **Index:** 0x21 **SubIndex:** Register group. See as7050_reg_group_ids **Data (Set):** [Payload Size: 1 - ... bytes] • *reg_group:* Data to set for register group. Payload depends on register group ID. **Data (Get):** [Payload Size: 0 - 1 byte] • *byte_count:* (optional) Amount of bytes to get from register group. If the value is 0 or omitted, the whole object is returned. **Answer (Get):** [Payload Size: byte_count if present and non-zero, up to AS7050_MAX_GROUP_SIZE otherwise] • *reg_group:* Data of the register group. See as7050_reg_group_ids

Enumerator

OBJ_ID_VSC_AGC_CONFIG	Set/Get AGC configuration. This command calls the functions as7050_vsc_cl_set_agc_config or as7050_vsc_cl_get_agc_config directly, depending on whether it is a write/or read request. **Index:** 0x26 **SubIndex:** Don't care **Data (Set):** [Payload Size: Multiple of sizeof(::agc_configuration_t)]: • *agc_config:* Multiple of sizeof(::agc_configuration_t). Array of AGC configurations, see ::agc_configuration_t. All previous AGC configurations are discarded when writing this object. **Data (Get):** [Payload Size: 0 - 1 byte] • *byte_count:* (optional) Maximum amount of bytes to read. If the value is 0 or omitted, the AGC configuration structures of all enabled AGC instances are returned. If the value is present and non-zero, it needs to be at least n ∗ sizeof(::agc_configuration_t) bytes, where n is the number of enabled AGC instances. **Answer (Get):** [Payload Size: Multiple of sizeof(::agc_configuration_t)]: • *agc_config:* Multiple of sizeof(::agc_configuration_t). Array of AGC configurations, see ::agc_configuration_t. A configuration is provided for each enabled AGC instance.
OBJ_ID_VSC_REG_ACCESS	Write/Read I2C Register. This command calls the function as7050_vsc_cl_write_register or as7050_vsc_cl_read_register directly, depending on whether it is a write or read request. **Index:** 0x04 **SubIndex:** Register address. See as7050_reg_addresses **Data (Write):** [Payload Size: 1 byte] • *reg_value:* Register value. **Data (Read):** [Payload Size: 0 - 1 byte] • *byte_count:* (optional) This value will be ignored, if set. **Answer (Read):** [Payload Size: 1 byte] • *reg_value:* Register value

Enumerator

OBJ_ID_VSC_MEAS_CONFIG	[Read-Only] Get the currently active measurement configuration. Accessing this object calls the function as7050_vsc_cl_get_measurement_config directly. **Index:** 0x23 **SubIndex:** Don't care **Data:** [Payload Size: 0 - 1 byte] • *byte_count:* (optional) Maximum amount of bytes to read. If the value is 0 or omitted, the whole measurement configuration structure is returned. If the value is present and non-zero, it needs to be at least sizeof(as7050_meas_config_t). **Answer:** [Payload Size: sizeof(as7050_meas_config_t)] • *meas_conf:* Currently active measurement configuration. See as7050_meas_config_t
OBJ_ID_VSC_VERSION	[Read-Only] Get the Version of the Vital Signs Controller. Accessing this object calls the function as7050_vsc_get_version directly. **Index:** 0x06 **SubIndex:** Don't care **Data:** [Payload Size: 0 - 1 byte] • *byte_count:* (optional) Maximum amount of bytes to read. If the value is 0 or omitted, the whole version structure is returned. If the value is present and non-zero, it needs to be at least sizeof(as7050_version_t). **Answer:** [Payload Size: sizeof(as7050_version_t)] • *version:* Version of the Vital Signs Controller. See as7050_version_t

Enumerator

OBJ_ID_VSC_RAW_APP_CONF	[Write-Only] Configuration of the raw data measurement application. Accessing this object calls the function ::as7050_vsc_raw_configure directly. **Index:** 0x07 **SubIndex:** Don't care **Data:** [Payload Size: 1 byte] • *enable_acc_streaming:* Enable streaming of accelerometer data (1 = on, 0 = off).
OBJ_ID_VSC_HRM_APP_CONF	[Write-Only] Configuration of the HRM data measurement application. Accessing this object calls the function ::as7050_vsc_hrm_configure directly. **Index:** 0x08 **SubIndex:** Don't care **Data:** [Payload Size: 1 byte] • *enable_hrv:* Enable streaming of additional HRV data (1 = on, 0 = off).
OBJ_ID_VSC_SPO2_APP_CONF	[Write-Only] Configuration of the SPO2 data measurement application. Accessing this object calls the function ::as7050_vsc_spo2_configure directly. **Index:** 0x09 **SubIndex:** Don't care **Data:** [Payload Size: sizeof(as7050_spo2_app_config_t)] • *spo2_config:* SpO2 configuration structure. See ::as7050_spo2_app_config_t

Enumerator

OBJ_ID_VSC_DAC_REF_CALC_CTRL	[Write-Only] Starts or aborts the calculation of the DAC reference, which is used for GSR measurement. Writing to this object triggers a call of as7050_vsc_dac_ref_calc_start or as7050_vsc_dac_ref_calc_abort. An indication is sent on the DAC Reference Value characteristic of the Galvanic Skin Resistance Measurement GATT service after the calculation finished successfully. **Index:** 0x27 **SubIndex:** DAC reference value calculation is started when this value is 0. Calculation is aborted when this value is 1. All other values are illegal. **Input** [Payload Size: 2 byte] • *sample_cnt:* uint16. Minimum number of samples that are acquired per DAC to calculate the DAC reference value. This argument is optional when a calculation is aborted.
OBJ_ID_VSC_GSR_APP_CONF	[Write-Only] Configures the GSR measurement application. Writing to this object triggers a call of ::as7050_vsc_gsr_configure. **Index:** 0x28 **SubIndex:** Don't care **Data:** [Payload Size: sizeof(gsr_app_configuration_t)] • *gsr_config:* GSR application configuration structure, see ::gsr_app_configuration_t.
OBJ_ID_VSC_ACC_SAMPLE_PERIOD	Set/Get the configuration of the accelerometer sample period Supported sample periods of the LIS2HD12 driver: 1s (1hz), 100ms (10hz), 40ms (25hz), 20ms (50hz), 10ms (100hz) and 5ms (200hz) **Index:** 0x29 **SubIndex:** Don't care **Data** (Write): [Payload Size: 4 bytes] • *sample_period:* uint32. Sample period of the accelerometer in microseconds. **Data (Read):** [Payload Size: 0 byte] **Answer (Read):** [Payload Size: 4 byte] • *sample_period:* uint32. Sample period of the accelerometer in microseconds. Default 100000.

4.2.4 Function Documentation

4.2.4.1 obj_vital_signs_get_table()

```
const obj_access_table_entry_t* obj_vital_signs_get_table (
        uint32_t * p_num )
```

returns the object table for vital signs

Parameters

out	*p_num*	number of elements in the table

Returns

pointer to the object table

4.3 AS7050 Vital Signs Controller Interface

The AS7050 Vital Signs Controller integrates the AS7050 Chip Library with the AS7050 Application Manager. This module can be used to configure the sensor and to perform measurements. The data acquired by the sensor is provided to the Application Manager automatically. Additionally, the Vital Signs Controller also interfaces with an accelerometer. The accelerometer data is also provided to the Application Manager.

Macros

- #define AS7050_VSC_CALC_DAC_REF_FLAG 0x8000000

Functions

- err_code_t as7050_vsc_initialize (const char ∗p_interface_descr)

  Initializes the Vital Signs Controller.
- err_code_t as7050_vsc_start_measurement (void)

  Starts a measurement.
- err_code_t as7050_vsc_execute (uint32_t ∗p_app_data_available, err_code_t ∗p_internal_error)

  Executes the Vital Signs Applications.
- err_code_t as7050_vsc_stop_measurement (void)

  Stops a measurement.
- err_code_t as7050_vsc_get_version (as7050_version_t ∗p_version)

  Gets the version of the Vital Signs Controller.
- err_code_t as7050_vsc_shutdown (void)

  De-initializes the Vital Signs Controller and powers down the sensor.
- err_code_t as7050_vsc_acc_set_sample_period (uint32_t sample_period_us)

  Configuration of the sample period for the accelerometer.
- err_code_t as7050_vsc_acc_get_sample_period (uint32_t ∗p_sample_period_us)

  Get the configured sample period for the accelerometer.
- err_code_t as7050_vsc_cl_set_reg_group (const enum as7050_reg_group_ids id, uint8_t ∗p_data, const uint8_t size)

  Writes to a register group.
- err_code_t as7050_vsc_cl_get_reg_group (const enum as7050_reg_group_ids id, uint8_t ∗p_data, uint8_t ∗p_size)

  Reads a register group.
- err_code_t as7050_vsc_cl_set_agc_config (const agc_configuration_t ∗p_agc_configs, uint8_t agc_config←_num)

  Sets the Automatic Gain Control (AGC) configuration.
- err_code_t as7050_vsc_cl_get_agc_config (agc_configuration_t ∗p_agc_configs, uint8_t ∗p_agc_config_←num)

  Gets the Automatic Gain Control (AGC) configuration.
- err_code_t as7050_vsc_cl_write_register (uint8_t reg_addr, uint8_t reg_val)

  Writes to an individual register.
- err_code_t as7050_vsc_cl_read_register (uint8_t reg_addr, uint8_t ∗p_reg_val)

  Reads an individual register.
- err_code_t as7050_vsc_cl_get_measurement_config (as7050_meas_config_t ∗p_meas_config)

  Reads the current measurement configuration of the sensor.

- err_code_t as7050_vsc_am_enable_apps (uint32_t enabled_apps)

  Sets the enabled Vital Signs Applications.
- err_code_t as7050_vsc_am_set_signal_routing (as7050_appmgr_app_id_t app, const as7050_appmgr_↩ channel_id_t ∗p_channels, uint8_t channels_num)

  Sets the signal routing for a Vital Signs Application.
- err_code_t as7050_vsc_am_configure_app (as7050_appmgr_app_id_t app, const void ∗p_config, uint8_↩ t size)

  Configures a Vital Signs Application.
- err_code_t as7050_vsc_am_get_output (as7050_appmgr_app_id_t app, void ∗p_dest, uint16_t ∗p_size)

  Writes output of a Vital Signs Application to a buffer provided by the caller.
- err_code_t as7050_vsc_dac_ref_calc_start (uint16_t sample_cnt)

  Starts the DAC reference calculation.
- err_code_t as7050_vsc_dac_ref_calc_abort (void)

  Aborts an ongoing DAC reference calculation.
- err_code_t as7050_vsc_get_dac_ref (uint32_t ∗p_dac_ref)

  Gets the calculated DAC reference value.

4.3.1 Detailed Description

The AS7050 Vital Signs Controller integrates the AS7050 Chip Library with the AS7050 Application Manager. This module can be used to configure the sensor and to perform measurements. The data acquired by the sensor is provided to the Application Manager automatically. Additionally, the Vital Signs Controller also interfaces with an accelerometer. The accelerometer data is also provided to the Application Manager.

4.3.2 Macro Definition Documentation

4.3.2.1 AS7050_VSC_CALC_DAC_REF_FLAG `#define AS7050_VSC_CALC_DAC_REF_FLAG 0x8000000`

Flag which is set in argument p_app_data_available of as7050_vsc_execute when DAC reference calculation finished.

4.3.3 Function Documentation

4.3.3.1 as7050_vsc_initialize()

```
err_code_t as7050_vsc_initialize (
        const char * p_interface_descr )
```

Initializes the Vital Signs Controller.

This function initializes the Chip Library, the Application Manager, and the accelerometer driver.

Note

> This function must be called first. All other functions of the Vital Signs Controller are blocked and return ERR_PERMISSION when the module is not initialized.

Parameters

in	*p_interface_descr*	Identifier of the AS7050 sensor. This value is provided to the Chip Library and its OSAL. The meaning of this value is dependent on the used OSAL implementation.

Return values

ERR_SUCCESS	Function returns without error.
ERR_IDENTIFICATION	The specified sensor was not found.
ERR_DATA_TRANSFER	Communication error with sensor.

4.3.3.2 as7050_vsc_start_measurement()

```
err_code_t as7050_vsc_start_measurement (
        void )
```

Starts a measurement.

Return values

ERR_SUCCESS	Function returns without error.
ERR_DATA_TRANSFER	Communication error with sensor.
ERR_CONFIG	Sensor configuration is incomplete.
ERR_PERMISSION	Access to the library is blocked, call as7050_vsc_initialize first.

4.3.3.3 as7050_vsc_execute()

```
err_code_t as7050_vsc_execute (
        uint32_t * p_app_data_available,
        err_code_t * p_internal_error )
```

Executes the Vital Signs Applications.

If enough input data is available, the applications are executed generate output data. It is indicated via the p_app_data_available argument which Vital Signs Applications have output data available. When output data is available, as7050_vsc_am_get_output can be called with the corresponding application identifier to obtain the generated output data. This function also provides information whether an error occurred in the Chip Library interrupt callback.

Parameters

out	*p_app_data_available*	Flags of Vital Signs Applications that have a output data available, see AS7050_APPMGR_APP_FLAG. An app has output data available when the corresponding bit is set.
out	*p_internal_error*	Error that occurred in the interrupt callback. If no error occurred, the variable pointed to by this argument is set to ERR_SUCCESS.

Return values

ERR_SUCCESS	Function returns without error.
ERR_POINTER	Invalid pointer argument value provided.
ERR_PERMISSION	Access to the library is blocked, call as7050_vsc_initialize first.

4.3.3.4 as7050_vsc_stop_measurement()

```
err_code_t as7050_vsc_stop_measurement (
        void )
```

Stops a measurement.

Return values

ERR_SUCCESS	Function returns without error.
ERR_DATA_TRANSFER	Communication error with sensor.
ERR_PERMISSION	Access to the library is blocked, call as7050_vsc_initialize first.

4.3.3.5 as7050_vsc_get_version()

```
err_code_t as7050_vsc_get_version (
        as7050_version_t * p_version )
```

Gets the version of the Vital Signs Controller.

Parameters

out	*p_version*	Version of the Vital Signs Controller, see as7050_version_t.

Return values

ERR_SUCCESS	Function returns without error.
ERR_POINTER	Invalid pointer argument value provided.
ERR_PERMISSION	Access to the library is blocked, call as7050_vsc_initialize first.

4.3.3.6 as7050_vsc_shutdown()

```
err_code_t as7050_vsc_shutdown (
        void )
```

De-initializes the Vital Signs Controller and powers down the sensor.

This functions performs the following tasks:

- Stopping running measurement, if required
- Powering down of the AS7050 sensor
- De-initializing the Chip Library, the Application Manager, and the accelerometer driver
- Blocking all functions of the Vital Signs Controller other than as7050_vsc_initialize.

Return values

ERR_SUCCESS	Function returns without error.
ERR_DATA_TRANSFER	Communication error to sensor.

4.3.3.7 as7050_vsc_acc_set_sample_period()

```
err_code_t as7050_vsc_acc_set_sample_period (
        uint32_t sample_period_us )
```

Configuration of the sample period for the accelerometer.

Parameters

in	*sample_period_us*	Sample period of the accelerometer in microseconds.

Return values

ERR_SUCCESS	Function returns without error.
ERR_ACCELEROMETER	Any error regarding the accelerometer.
ERR_PERMISSION	Access to driver is blocked.

4.3.3.8 as7050_vsc_acc_get_sample_period()

```
err_code_t as7050_vsc_acc_get_sample_period (
        uint32_t * p_sample_period_us )
```

Get the configured sample period for the accelerometer.

Parameters

out	*p_sample_period_us*	Sample period of the accelerometer in microseconds.

Return values

ERR_SUCCESS	Function returns without error.
ERR_ACCELEROMETER	Any error regarding the accelerometer.
ERR_PERMISSION	Access to driver is blocked.

Return values

ERR_POINTER	Null pointer detected.

4.3.3.9 as7050_vsc_cl_set_reg_group()

```
err_code_t as7050_vsc_cl_set_reg_group (
        const enum as7050_reg_group_ids id,
        uint8_t * p_data,
        const uint8_t size )
```

Writes to a register group.

This function calls ::as7050_set_reg_group. See the description of this function for more information.

4.3.3.10 as7050_vsc_cl_get_reg_group()

```
err_code_t as7050_vsc_cl_get_reg_group (
        const enum as7050_reg_group_ids id,
        uint8_t * p_data,
        uint8_t * p_size )
```

Reads a register group.

This function calls ::as7050_get_reg_group. See the description of this function for more information.

4.3.3.11 as7050_vsc_cl_set_agc_config()

```
err_code_t as7050_vsc_cl_set_agc_config (
        const agc_configuration_t * p_agc_configs,
        uint8_t agc_config_num )
```

Sets the Automatic Gain Control (AGC) configuration.

This function calls ::as7050_set_agc_config. See the description of this function for more information.

4.3.3.12 as7050_vsc_cl_get_agc_config()

```
err_code_t as7050_vsc_cl_get_agc_config (
        agc_configuration_t * p_agc_configs,
        uint8_t * p_agc_config_num )
```

Gets the Automatic Gain Control (AGC) configuration.

This function calls ::as7050_get_agc_config. See the description of this function for more information.

4.3.3.13 as7050_vsc_cl_write_register()

```
err_code_t as7050_vsc_cl_write_register (
        uint8_t reg_addr,
        uint8_t reg_val )
```

Writes to an individual register.

This function calls ::as7050_write_register. See the description of this function for more information.

4.3.3.14 as7050_vsc_cl_read_register()

```
err_code_t as7050_vsc_cl_read_register (
        uint8_t reg_addr,
        uint8_t * p_reg_val )
```

Reads an individual register.

This function calls ::as7050_read_register. See the description of this function for more information.

4.3.3.15 as7050_vsc_am_enable_apps()

```
err_code_t as7050_vsc_am_enable_apps (
        uint32_t enabled_apps )
```

Sets the enabled Vital Signs Applications.

This function calls ::as7050_appmgr_enable_apps. See the description of this function for more information.

4.3.3.16 as7050_vsc_am_set_signal_routing()

```
err_code_t as7050_vsc_am_set_signal_routing (
        as7050_appmgr_app_id_t app,
        const as7050_appmgr_channel_id_t * p_channels,
        uint8_t channels_num )
```

Sets the signal routing for a Vital Signs Application.

This function calls ::as7050_appmgr_set_signal_routing. See the description of this function for more information.

4.3.3.17 as7050_vsc_am_configure_app()

```
err_code_t as7050_vsc_am_configure_app (
        as7050_appmgr_app_id_t app,
        const void * p_config,
        uint8_t size )
```

Configures a Vital Signs Application.

This function calls ::as7050_appmgr_configure_app. See the description of this function for more information.

4.3.3.18 as7050_vsc_am_get_output()

```
err_code_t as7050_vsc_am_get_output (
        as7050_appmgr_app_id_t app,
        void * p_dest,
        uint16_t * p_size )
```

Writes output of a Vital Signs Application to a buffer provided by the caller.

This function calls ::as7050_appmgr_get_output. See the description of this function for more information.

4.3.3.19 as7050_vsc_dac_ref_calc_start()

```
err_code_t as7050_vsc_dac_ref_calc_start (
        uint16_t sample_cnt )
```

Starts the DAC reference calculation.

as7050_vsc_execute needs to be called repeatedly while the DAC reference value is calculated and indicates via its p_app_data_available argument when the DAC reference calculation finished. During DAC reference calculation sensor configuration or regular measurements cannot be performed.

When calling this function the GSR registers of the sensor must be configured. The ECG channel must be enabled and all PPG channels must be disabled.

Parameters

in	*sample_cnt*	Minimum number of samples that are acquired per DAC to calculate the DAC reference value.

Return values

ERR_SUCCESS	DAC reference calculation started.
ERR_PERMISSION	Library is not initialized or a regular measurement or a DAC reference calculation is in progress.
ERR_ARGUMENT	Invalid sample_cnt argument value.
ERR_CONFIG	Incompatible sensor configuration.
ERR_EVENT	Unexpected event while attempting to start DAC reference calculation.

4.3.3.20 as7050_vsc_dac_ref_calc_abort()

```
err_code_t as7050_vsc_dac_ref_calc_abort (
        void )
```

Aborts an ongoing DAC reference calculation.

Return values

ERR_SUCCESS	DAC reference calculation started.
ERR_PERMISSION	Library is not initialized or a regular measurement is in progress or a DAC reference calculation is not ongoing.
ERR_EVENT	Unexpected event while attempting to abort DAC reference calculation.

4.3.3.21 as7050_vsc_get_dac_ref()

```
err_code_t as7050_vsc_get_dac_ref (
        uint32_t * p_dac_ref )
```

Gets the calculated DAC reference value.

DAC reference calculation is started via as7050_vsc_dac_ref_calc_start.

Parameters

out	*p_dac_ref*	Pointer to variable where the calculated DAC reference value is written to.

Return values

ERR_SUCCESS	Function returns without error.
ERR_PERMISSION	Access to the library is blocked, call as7050_vsc_initialize at first.

Return values

ERR_POINTER	Detected NULL pointer for parameter p_dac_ref.
ERR_NO_DATA	No DAC reference value data is available.
ERR_EVENT	Error occurred during DAC reference calculation.

4.4 AS7050 Application Manager Definitions

4.5 AS7050 Chip Library Definitions

Description of the used data types.

Data Structures

- struct as7050_version
- union as7050_config_aoc_t
- union as7050_config_led_t
- union as7050_config_pd_t
- union as7050_config_ppg_t
- union as7050_config_sinc_t
- union as7050_config_seq_t
- union as7050_config_ref_t
- union as7050_config_gpio_t
- union as7050_config_ctrl_t
- union as7050_config_standby_t
- union as7050_config_fifo_t
- union as7050_config_ecg_t
- union as7050_config_afe_t
- union as7050_config_amp_t
- union as7050_config_tia_t
- union as7050_config_iir_t
- struct as7050_meas_config_t

Macros

- #define AS7050_FIFO_DATA_BUFFER_SIZE 2048
- #define AS7050_MAX_GROUP_SIZE 128
- #define AS7050_CHANNEL_FLAG_GSR_OFFSET 12

Typedefs

- typedef enum as7050_reg_group_ids as7050_reg_group_ids_t
- typedef uint8_t dac_ref_control_t
- typedef uint8_t dac_ref_status_t
- typedef struct as7050_version as7050_version_t
- typedef void(∗ as7050_callback_t) (err_code_t error, uint8_t ∗p_data, uint16_t data_num, agc_status_t ∗p←_agc_status, uint8_t agc_status_num, void ∗p_cb_param)

  Callback function, which transfers the measurement results to the application.

Enumerations

- enum as7050_reg_group_ids {
  AS7050_REG_GROUP_ID_CTRL = 0,
  AS7050_REG_GROUP_ID_GPIO = 1,
  AS7050_REG_GROUP_ID_STANDBY = 2,
  AS7050_REG_GROUP_ID_IIR = 3,
  AS7050_REG_GROUP_ID_REF = 4,
  AS7050_REG_GROUP_ID_AOC,
  AS7050_REG_GROUP_ID_PPG = 6,
  AS7050_REG_GROUP_ID_ECG = 7,
  AS7050_REG_GROUP_ID_AMP = 8,
  AS7050_REG_GROUP_ID_TIA,
  AS7050_REG_GROUP_ID_AFE,
  AS7050_REG_GROUP_ID_SINC = 11,
  AS7050_REG_GROUP_ID_LED = 12,
  AS7050_REG_GROUP_ID_PD = 13,
  AS7050_REG_GROUP_ID_FIFO = 14,
  AS7050_REG_GROUP_ID_SEQ = 15,
  AS7050_REG_GROUP_ID_NUM = 16 }
- enum as7050_channel_flags_t {
  AS7050_CHANNEL_FLAG_NONE = 0x0000,
  AS7050_CHANNEL_FLAG_PPG_1 = 0x0001,
  AS7050_CHANNEL_FLAG_PPG_2 = 0x0002,
  AS7050_CHANNEL_FLAG_PPG_3 = 0x0004,
  AS7050_CHANNEL_FLAG_PPG_4 = 0x0008,
  AS7050_CHANNEL_FLAG_PPG_5 = 0x0010,
  AS7050_CHANNEL_FLAG_PPG_6 = 0x0020,
  AS7050_CHANNEL_FLAG_PPG_7 = 0x0040,
  AS7050_CHANNEL_FLAG_PPG_8 = 0x0080,
  AS7050_CHANNEL_FLAG_ECG = 0x0100,
  AS7050_CHANNEL_FLAG_STATUS = 0x0200,
  AS7050_CHANNEL_FLAG_GSR = 0xF000 }
- enum FIFO_DATA_MARKERS {
  FIFO_DATA_MARKER_PPG_2_8 = 0x00,
  FIFO_DATA_MARKER_ECG = 0x01,
  FIFO_DATA_MARKER_PPG_1 = 0x02,
  FIFO_DATA_MARKER_STATUS = 0x03 }
- enum dac_ref_control {
  DAC_REF_CONTROL_CALC = 0,
  DAC_REF_CONTROL_ABORT }
- enum dac_ref_status {
  DAC_REF_STATUS_RUNNING = 0,
  DAC_REF_STATUS_ABORTED,
  DAC_REF_STATUS_FINISHED }
- enum as7050_reg_addresses {
  AS7050_REGADDR_GPIO1_CFG = 0x10,
  AS7050_REGADDR_GPIO2_CFG = 0x11,
  AS7050_REGADDR_INT_CFG = 0x12,
  AS7050_REGADDR_IO_CFGA = 0x13,
  AS7050_REGADDR_IO_CFGB = 0x14,
  AS7050_REGADDR_GPIO1_CFGB = 0x15,
  AS7050_REGADDR_GPIO2_CFGB = 0x16,
  AS7050_REGADDR_INT_CFGB = 0x17,

```
AS7050_REGADDR_PD_OFFSET_CFG = 0x18,
AS7050_REGADDR_PPG_MOD_CFGA = 0x19,
AS7050_REGADDR_PPG_MOD_CFGB = 0x1a,
AS7050_REGADDR_PPG_MOD_CFGC = 0x1b,
AS7050_REGADDR_PPG_MOD_CFGD = 0x1c,
AS7050_REGADDR_PPG_MOD_CFGE = 0x1d,
AS7050_REGADDR_TIA_CFGA = 0x1e,
AS7050_REGADDR_TIA_CFGB = 0x1f,
AS7050_REGADDR_ECG_AMP_CFGA = 0x20,
AS7050_REGADDR_ECG_AMP_CFGB = 0x21,
AS7050_REGADDR_ECG_AMP_CFGC = 0x22,
AS7050_REGADDR_PDSEL_CFG = 0x23,
AS7050_REGADDR_ECG_SOURCE = 0x24,
AS7050_REGADDR_ECG_MOD_CFGA = 0x25,
AS7050_REGADDR_TIA_CFGC = 0x26,
AS7050_REGADDR_LOWVDS_WAIT = 0x27,
AS7050_REGADDR_LED1_ICTRL = 0x28,
AS7050_REGADDR_LED2_ICTRL = 0x29,
AS7050_REGADDR_LED3_ICTRL = 0x2a,
AS7050_REGADDR_LED4_ICTRL = 0x2b,
AS7050_REGADDR_LED5_ICTRL = 0x2c,
AS7050_REGADDR_LED6_ICTRL = 0x2d,
AS7050_REGADDR_LED7_ICTRL = 0x2e,
AS7050_REGADDR_LED8_ICTRL = 0x2f,
AS7050_REGADDR_REF_CFGA = 0x30,
AS7050_REGADDR_REF_CFGB = 0x31,
AS7050_REGADDR_AFE_DAC0L = 0x32,
AS7050_REGADDR_AFE_DAC1L = 0x33,
AS7050_REGADDR_AFE_DACH = 0x34,
AS7050_REGADDR_AFE_CFGA = 0x35,
AS7050_REGADDR_AFE_CFGB = 0x36,
AS7050_REGADDR_CONTROL = 0x37,
AS7050_REGADDR_CGB_CFG = 0x38,
AS7050_REGADDR_ECG_AMP_CFGE = 0x39,
AS7050_REGADDR_AFE_GSR = 0x3a,
AS7050_REGADDR_SEQ_SAMPLE = 0x40,
AS7050_REGADDR_SEQ_PPGA = 0x41,
AS7050_REGADDR_SEQ_PPGB = 0x42,
AS7050_REGADDR_PD_PPG1 = 0x43,
AS7050_REGADDR_PD_PPG2 = 0x44,
AS7050_REGADDR_PD_PPG3 = 0x45,
AS7050_REGADDR_PD_PPG4 = 0x46,
AS7050_REGADDR_PD_PPG5 = 0x47,
AS7050_REGADDR_PD_PPG6 = 0x48,
AS7050_REGADDR_PD_PPG7 = 0x49,
AS7050_REGADDR_PD_PPG8 = 0x4a,
AS7050_REGADDR_PD_TIA = 0x4b,
AS7050_REGADDR_LED_INIT = 0x4c,
AS7050_REGADDR_LED_PPG1 = 0x4d,
AS7050_REGADDR_LED_PPG2 = 0x4e,
AS7050_REGADDR_LED_PPG3 = 0x4f,
AS7050_REGADDR_LED_PPG4 = 0x50,
AS7050_REGADDR_LED_PPG5 = 0x51,
AS7050_REGADDR_LED_PPG6 = 0x52,
AS7050_REGADDR_LED_PPG7 = 0x53,
AS7050_REGADDR_LED_PPG8 = 0x54,
AS7050_REGADDR_LED_TIA = 0x55,
AS7050_REGADDR_LED_MODE = 0x56,
AS7050_REGADDR_SEQ_COUNT = 0x57,
AS7050_REGADDR_SEQ_MODE = 0x59,
AS7050_REGADDR_SEQ_START = 0x5a,
AS7050_REGADDR_SINC_PPG_CFGA = 0x5b,
AS7050_REGADDR_SINC_PPG_CFGB = 0x5c,
AS7050_REGADDR_SINC_PPG_CFGC = 0x5d,
AS7050_REGADDR_SINC_ECG_CFGA = 0x5e,
AS7050_REGADDR_SINC_ECG_CFGB = 0x5f,
AS7050_REGADDR_SINC_ECG_CFGC = 0x60,
AS7050_REGADDR_IIR_CFG = 0x61,
AS7050_REGADDR_IIR_COEFF_ADDR = 0x62,
AS7050_REGADDR_IIR_COEFF_DATA = 0x63,
AS7050_REGADDR_OVS_CFG = 0x64,
AS7050_REGADDR_AOC_IOS_PPG1 = 0x65,
AS7050_REGADDR_AOC_IOS_PPG2 = 0x66,
AS7050_REGADDR_AOC_IOS_PPG3 = 0x67,
AS7050_REGADDR_AOC_IOS_PPG4 = 0x68,
AS7050_REGADDR_AOC_IOS_PPG5 = 0x69,
AS7050_REGADDR_AOC_IOS_PPG6 = 0x6a,
AS7050_REGADDR_AOC_IOS_PPG7 = 0x6b,
AS7050_REGADDR_AOC_IOS_PPG8 = 0x6c,
AS7050_REGADDR_AOC_PPG_THH = 0x6d,
AS7050_REGADDR_AOC_PPG_THL = 0x6e,
AS7050_REGADDR_AOC_PPG_CFG = 0x6f,
AS7050_REGADDR_AOC_IOS_ECG = 0x70,
AS7050_REGADDR_AOC_ECG_THH = 0x71,
AS7050_REGADDR_AOC_ECG_THL = 0x72,
AS7050_REGADDR_AOC_ECG_CFG = 0x73,
AS7050_REGADDR_AOC_IOS_LEDOFF = 0x74,
AS7050_REGADDR_FIFO_CTRL = 0x75,
AS7050_REGADDR_FIFO_THRESHOLD = 0x76,
AS7050_REGADDR_FIFO_LEVEL0 = 0x77,
AS7050_REGADDR_FIFO_LEVEL1 = 0x78,
AS7050_REGADDR_STATUS = 0x80,
AS7050_REGADDR_STATUS_CGBA = 0x81,
AS7050_REGADDR_STATUS_CGBB = 0x82,
AS7050_REGADDR_STATUS_MOD = 0x83,
AS7050_REGADDR_STATUS_LED = 0x84,
AS7050_REGADDR_IRQ_ENABLE = 0x88,
AS7050_REGADDR_GPIO_IO = 0x8f,
AS7050_REGADDR_REVISION = 0x90,
AS7050_REGADDR_STANDBY_CFGA = 0xb0,
AS7050_REGADDR_STANDBY_CFGB = 0xb1,
AS7050_REGADDR_PPG_BYTEL = 0xf4,
AS7050_REGADDR_PPG_BYTEM = 0xf5,
AS7050_REGADDR_PPG_BYTEH = 0xf6,
AS7050_REGADDR_ECG_BYTEL = 0xf8,
AS7050_REGADDR_ECG_BYTEM = 0xf9,
AS7050_REGADDR_ECG_BYTEH = 0xfa,
AS7050_REGADDR_FIFOL = 0xfc,
```

AS7050_REGADDR_FIFOM = 0xfd,
AS7050_REGADDR_FIFOH = 0xfe,
AS7050_REGADDR_BYTE0 = 0xff }

- enum as7050_channel_t {
AS7050_CHANNEL_DISABLED = 0,
AS7050_CHANNEL_PPG_1 = 1,
AS7050_CHANNEL_PPG_2 = 2,
AS7050_CHANNEL_PPG_3 = 3,
AS7050_CHANNEL_PPG_4 = 4,
AS7050_CHANNEL_PPG_5 = 5,
AS7050_CHANNEL_PPG_6 = 6,
AS7050_CHANNEL_PPG_7 = 7,
AS7050_CHANNEL_PPG_8 = 8,
AS7050_CHANNEL_ECG = 9 }

4.5.1 Detailed Description

Description of the used data types.

These are the type definitions used by AS7050 chip library.

4.5.2 Macro Definition Documentation

4.5.2.1 AS7050_FIFO_DATA_BUFFER_SIZE `#define AS7050_FIFO_DATA_BUFFER_SIZE 2048`

Maximum buffer size for saving FIFO data

4.5.2.2 AS7050_MAX_GROUP_SIZE `#define AS7050_MAX_GROUP_SIZE 128`

maximum space for payload of register groups

4.5.2.3 AS7050_CHANNEL_FLAG_GSR_OFFSET `#define AS7050_CHANNEL_FLAG_GSR_OFFSET 12`

Bit position of GSR data inside as7050_channel_flags_t

4.5.3 Typedef Documentation

4.5.3.1 as7050_reg_group_ids_t `typedef enum as7050_reg_group_ids as7050_reg_group_ids_t`

Definition of the register group IDs

4.5.3.2 dac_ref_control_t `typedef uint8_t dac_ref_control_t`

Type for dac_ref_control.

4.5.3.3 dac_ref_status_t `typedef uint8_t dac_ref_status_t`

Type for dac_ref_status.

4.5.3.4 as7050_version_t `typedef struct as7050_version as7050_version_t`

Version information of the library

4.5.3.5 as7050_callback_t `typedef void(* as7050_callback_t) (err_code_t error, uint8_t *p_data, uint16_t data_num, agc_status_t *p_agc_status, uint8_t agc_status_num, void *p_cb_param)`

Callback function, which transfers the measurement results to the application.

This callback type will be registered via the function ::as7050_initialize. During the measurement, this function transfers the cyclic results.

Parameters

`in`	*error*	Default ERR_SUCCESS, otherwise an error is occurred during measurement and the measurement. stops. See error_codes
`in`	*p_data*	Pointer to the measurement data, the content depends on configuration.
`in`	*data_num*	Number of 16bit data
`in`	*p_agc_status*	Pointer to AGC status structure
`in`	*agc_status_num*	Number of AGC status elements
`in`	*p_cb_param*	Application parameter which was defined during call of ::as7050_initialize.

4.5.4 Enumeration Type Documentation

4.5.4.1 as7050_reg_group_ids `enum as7050_reg_group_ids`

Definition of the register group IDs

Enumerator

AS7050_REG_GROUP_ID_CTRL	Group-ID of all registers of group 'Control'. See as7050_config_ctrl_t
AS7050_REG_GROUP_ID_GPIO	Group-ID of all registers of group 'GPIO'. See as7050_config_gpio_t
AS7050_REG_GROUP_ID_STANDBY	Group-ID of all registers of group 'Standby'. See as7050_config_standby_t

Enumerator

AS7050_REG_GROUP_ID_IIR	Group-ID of all registers of group 'IIR filter'. See as7050_config_iir_t
AS7050_REG_GROUP_ID_REF	Group-ID of all registers of group 'Reference'. See as7050_config_ref_t
AS7050_REG_GROUP_ID_AOC	Group-ID of all registers of group 'Automatic Offset Control'. See as7050_config_aoc_t
AS7050_REG_GROUP_ID_PPG	Group-ID of all registers of group 'PPG'. See as7050_config_ppg_t
AS7050_REG_GROUP_ID_ECG	Group-ID of all registers of group 'ECG'. See as7050_config_ecg_t
AS7050_REG_GROUP_ID_AMP	Group-ID of all registers of group 'amplifier'. See as7050_config_amp_t
AS7050_REG_GROUP_ID_TIA	Group-ID of all registers of group 'Transimpedance amplifier'. See as7050_config_tia_t
AS7050_REG_GROUP_ID_AFE	Group-ID of all registers of group 'Analog front end'. See as7050_config_afe_t
AS7050_REG_GROUP_ID_SINC	Group-ID of all registers of group 'Sinc filter'. See as7050_config_sinc_t
AS7050_REG_GROUP_ID_LED	Group-ID of all registers of group 'LED'. See as7050_config_led_t
AS7050_REG_GROUP_ID_PD	Group-ID of all registers of group 'Photodiodes'. See as7050_config_pd_t
AS7050_REG_GROUP_ID_FIFO	Group-ID of all registers of group 'FIFO'. See as7050_config_fifo_t
AS7050_REG_GROUP_ID_SEQ	Group-ID of all registers of group 'Sequencer'. See as7050_config_seq_t
AS7050_REG_GROUP_ID_NUM	Maximum number of supported register groups

4.5.4.2 as7050_channel_flags_t enum as7050_channel_flags_t

Channel flags

Enumerator

AS7050_CHANNEL_FLAG_NONE	No channel is selected
AS7050_CHANNEL_FLAG_PPG_1	Flag for PPG 1 channel
AS7050_CHANNEL_FLAG_PPG_2	Flag for PPG 2 channel
AS7050_CHANNEL_FLAG_PPG_3	Flag for PPG 3 channel
AS7050_CHANNEL_FLAG_PPG_4	Flag for PPG 4 channel
AS7050_CHANNEL_FLAG_PPG_5	Flag for PPG 5 channel
AS7050_CHANNEL_FLAG_PPG_6	Flag for PPG 6 channel
AS7050_CHANNEL_FLAG_PPG_7	Flag for PPG 7 channel
AS7050_CHANNEL_FLAG_PPG_8	Flag for PPG 8 channel
AS7050_CHANNEL_FLAG_ECG	Flag for ECG channel
AS7050_CHANNEL_FLAG_STATUS	Flag for status channel
AS7050_CHANNEL_FLAG_GSR	When performing GSR measurement, DAC 0 and DAC 1 are measured on the ECG channel. The chip alternates between DAC 0 and DAC 1 automatically. The bits masked by this bitmask contain the number of samples that are taken per DAC before the chip alternates to the other DAC.

4.5.4.3 FIFO_DATA_MARKERS enum FIFO_DATA_MARKERS

First two bits of the measurement data, which describes the type of the sample

Enumerator

FIFO_DATA_MARKER_PPG_2↩ _8	PPG sub samples 2 - 8
FIFO_DATA_MARKER_ECG	ECG sample
FIFO_DATA_MARKER_PPG_1	First PPG sample
FIFO_DATA_MARKER_STATUS	Status marker for AOC information

4.5.4.4 dac_ref_control enum dac_ref_control

Control states for DAC reference calculation

Enumerator

DAC_REF_CONTROL_CALC	Start or continue the DAC reference calculation
DAC_REF_CONTROL_ABORT	Abort the DAC reference calculation

4.5.4.5 dac_ref_status enum dac_ref_status

Status for DAC reference calculation

Enumerator

DAC_REF_STATUS_RUNNING	DAC reference calculation is running
DAC_REF_STATUS_ABORTED	DAC reference calculation was aborted
DAC_REF_STATUS_FINISHED	DAC reference calculation is finished

4.5.4.6 as7050_reg_addresses enum as7050_reg_addresses

Register definition of AS7050

Enumerator

AS7050_REGADDR_GPIO1_CFG	Register GPIO1_CFG
AS7050_REGADDR_GPIO2_CFG	Register GPIO2_CFG
AS7050_REGADDR_INT_CFG	Register INT_CFG
AS7050_REGADDR_IO_CFGA	Register IO_CFGA
AS7050_REGADDR_IO_CFGB	Register IO_CFGB
AS7050_REGADDR_GPIO1_CFGB	Register GPIO1_CFGB
AS7050_REGADDR_GPIO2_CFGB	Register GPIO2_CFGB
AS7050_REGADDR_INT_CFGB	Register INT_CFGB
AS7050_REGADDR_PD_OFFSET_CFG	Register PD_OFFSET_CFG
AS7050_REGADDR_PPG_MOD_CFGA	Register PPG_MOD_CFGA
AS7050_REGADDR_PPG_MOD_CFGB	Register PPG_MOD_CFGB
AS7050_REGADDR_PPG_MOD_CFGC	Register PPG_MOD_CFGC
AS7050_REGADDR_PPG_MOD_CFGD	Register PPG_MOD_CFGD
AS7050_REGADDR_PPG_MOD_CFGE	Register PPG_MOD_CFGE
AS7050_REGADDR_TIA_CFGA	Register TIA_CFGA
AS7050_REGADDR_TIA_CFGB	Register TIA_CFGB
AS7050_REGADDR_ECG_AMP_CFGA	Register ECG_AMP_CFGA
AS7050_REGADDR_ECG_AMP_CFGB	Register ECG_AMP_CFGB
AS7050_REGADDR_ECG_AMP_CFGC	Register ECG_AMP_CFGC
AS7050_REGADDR_PDSEL_CFG	Register PDSEL_CFG
AS7050_REGADDR_ECG_SOURCE	Register ECG_SOURCE
AS7050_REGADDR_ECG_MOD_CFGA	Register ECG_MOD_CFGA
AS7050_REGADDR_TIA_CFGC	Register TIA_CFGC
AS7050_REGADDR_LOWVDS_WAIT	Register LOWVDS_WAIT
AS7050_REGADDR_LED1_ICTRL	Register LED1_ICTRL
AS7050_REGADDR_LED2_ICTRL	Register LED2_ICTRL
AS7050_REGADDR_LED3_ICTRL	Register LED3_ICTRL
AS7050_REGADDR_LED4_ICTRL	Register LED4_ICTRL
AS7050_REGADDR_LED5_ICTRL	Register LED5_ICTRL
AS7050_REGADDR_LED6_ICTRL	Register LED6_ICTRL
AS7050_REGADDR_LED7_ICTRL	Register LED7_ICTRL
AS7050_REGADDR_LED8_ICTRL	Register LED8_ICTRL
AS7050_REGADDR_REF_CFGA	Register REF_CFGA
AS7050_REGADDR_REF_CFGB	Register REF_CFGB
AS7050_REGADDR_AFE_DAC0L	Register AFE_DAC0L
AS7050_REGADDR_AFE_DAC1L	Register AFE_DAC1L
AS7050_REGADDR_AFE_DACH	Register AFE_DACH
AS7050_REGADDR_AFE_CFGA	Register AFE_CFGA
AS7050_REGADDR_AFE_CFGB	Register AFE_CFGB
AS7050_REGADDR_CONTROL	Register CONTROL
AS7050_REGADDR_CGB_CFG	Register CGB_CFG
AS7050_REGADDR_ECG_AMP_CFGE	Register ECG_AMP_CFGE
AS7050_REGADDR_AFE_GSR	Register AFE_GSR

Enumerator

AS7050_REGADDR_SEQ_SAMPLE	Register SEQ_SAMPLE
AS7050_REGADDR_SEQ_PPGA	Register SEQ_PPGA
AS7050_REGADDR_SEQ_PPGB	Register SEQ_PPGB
AS7050_REGADDR_PD_PPG1	Register PD_PPG1
AS7050_REGADDR_PD_PPG2	Register PD_PPG2
AS7050_REGADDR_PD_PPG3	Register PD_PPG3
AS7050_REGADDR_PD_PPG4	Register PD_PPG4
AS7050_REGADDR_PD_PPG5	Register PD_PPG5
AS7050_REGADDR_PD_PPG6	Register PD_PPG6
AS7050_REGADDR_PD_PPG7	Register PD_PPG7
AS7050_REGADDR_PD_PPG8	Register PD_PPG8
AS7050_REGADDR_PD_TIA	Register PD_TIA
AS7050_REGADDR_LED_INIT	Register LED_INIT
AS7050_REGADDR_LED_PPG1	Register LED_PPG1
AS7050_REGADDR_LED_PPG2	Register LED_PPG2
AS7050_REGADDR_LED_PPG3	Register LED_PPG3
AS7050_REGADDR_LED_PPG4	Register LED_PPG4
AS7050_REGADDR_LED_PPG5	Register LED_PPG5
AS7050_REGADDR_LED_PPG6	Register LED_PPG6
AS7050_REGADDR_LED_PPG7	Register LED_PPG7
AS7050_REGADDR_LED_PPG8	Register LED_PPG8
AS7050_REGADDR_LED_TIA	Register LED_TIA
AS7050_REGADDR_LED_MODE	Register LED_MODE
AS7050_REGADDR_SEQ_COUNT	Register SEQ_COUNT
AS7050_REGADDR_SEQ_MODE	Register SEQ_MODE
AS7050_REGADDR_SEQ_START	Register SEQ_START
AS7050_REGADDR_SINC_PPG_CFGA	Register SINC_PPG_CFGA
AS7050_REGADDR_SINC_PPG_CFGB	Register SINC_PPG_CFGB
AS7050_REGADDR_SINC_PPG_CFGC	Register SINC_PPG_CFGC
AS7050_REGADDR_SINC_ECG_CFGA	Register SINC_ECG_CFGA
AS7050_REGADDR_SINC_ECG_CFGB	Register SINC_ECG_CFGB
AS7050_REGADDR_SINC_ECG_CFGC	Register SINC_ECG_CFGC
AS7050_REGADDR_IIR_CFG	Register IIR_CFG
AS7050_REGADDR_IIR_COEFF_ADDR	Register IIR_COEFF_ADDR
AS7050_REGADDR_IIR_COEFF_DATA	Register IIR_COEFF_DATA
AS7050_REGADDR_OVS_CFG	Register OVS_CFG
AS7050_REGADDR_AOC_IOS_PPG1	Register AOC_IOS_PPG1
AS7050_REGADDR_AOC_IOS_PPG2	Register AOC_IOS_PPG2
AS7050_REGADDR_AOC_IOS_PPG3	Register AOC_IOS_PPG3
AS7050_REGADDR_AOC_IOS_PPG4	Register AOC_IOS_PPG4
AS7050_REGADDR_AOC_IOS_PPG5	Register AOC_IOS_PPG5
AS7050_REGADDR_AOC_IOS_PPG6	Register AOC_IOS_PPG6
AS7050_REGADDR_AOC_IOS_PPG7	Register AOC_IOS_PPG7

Enumerator

AS7050_REGADDR_AOC_IOS_PPG8	Register AOC_IOS_PPG8
AS7050_REGADDR_AOC_PPG_THH	Register AOC_PPG_THH
AS7050_REGADDR_AOC_PPG_THL	Register AOC_PPG_THL
AS7050_REGADDR_AOC_PPG_CFG	Register AOC_PPG_CFG
AS7050_REGADDR_AOC_IOS_ECG	Register AOC_IOS_ECG
AS7050_REGADDR_AOC_ECG_THH	Register AOC_ECG_THH
AS7050_REGADDR_AOC_ECG_THL	Register AOC_ECG_THL
AS7050_REGADDR_AOC_ECG_CFG	Register AOC_ECG_CFG
AS7050_REGADDR_AOC_IOS_LEDOFF	Register AOC_IOS_LEDOFF
AS7050_REGADDR_FIFO_CTRL	Register FIFO_CTRL
AS7050_REGADDR_FIFO_THRESHOLD	Register FIFO_THRESHOLD
AS7050_REGADDR_FIFO_LEVEL0	Register FIFO_LEVEL0
AS7050_REGADDR_FIFO_LEVEL1	Register FIFO_LEVEL1
AS7050_REGADDR_STATUS	Register STATUS
AS7050_REGADDR_STATUS_CGBA	Register STATUS_CGBA
AS7050_REGADDR_STATUS_CGBB	Register STATUS_CGBB
AS7050_REGADDR_STATUS_MOD	Register STATUS_MOD
AS7050_REGADDR_STATUS_LED	Register STATUS_LED
AS7050_REGADDR_IRQ_ENABLE	Register IRQ_ENABLE
AS7050_REGADDR_GPIO_IO	Register GPIO_IO
AS7050_REGADDR_REVISION	Register REVISION
AS7050_REGADDR_STANDBY_CFGA	Register STANDBY_CFGA
AS7050_REGADDR_STANDBY_CFGB	Register STANDBY_CFGB
AS7050_REGADDR_PPG_BYTEL	Register PPG_BYTEL
AS7050_REGADDR_PPG_BYTEM	Register PPG_BYTEM
AS7050_REGADDR_PPG_BYTEH	Register PPG_BYTEH
AS7050_REGADDR_ECG_BYTEL	Register ECG_BYTEL
AS7050_REGADDR_ECG_BYTEM	Register ECG_BYTEM
AS7050_REGADDR_ECG_BYTEH	Register ECG_BYTEH
AS7050_REGADDR_FIFOL	Register FIFOL
AS7050_REGADDR_FIFOM	Register FIFOM
AS7050_REGADDR_FIFOH	Register FIFOH
AS7050_REGADDR_BYTE0	Register BYTE0

4.5.4.7 as7050_channel_t `enum as7050_channel_t`

AGC channel selection

4.6 Error Codes

Typedefs

- typedef enum error_codes err_code_t

Enumerations

- enum error_codes {
  ERR_SUCCESS = 0,
  ERR_PERMISSION = 1,
  ERR_MESSAGE = 2,
  ERR_MESSAGE_SIZE = 3,
  ERR_POINTER = 4,
  ERR_ACCESS = 5,
  ERR_ARGUMENT = 6,
  ERR_SIZE = 7,
  ERR_NOT_SUPPORTED = 8,
  ERR_TIMEOUT = 9,
  ERR_CHECKSUM = 10,
  ERR_OVERFLOW = 11,
  ERR_EVENT = 12,
  ERR_INTERRUPT = 13,
  ERR_TIMER_ACCESS = 14,
  ERR_LED_ACCESS = 15,
  ERR_TEMP_SENSOR_ACCESS = 16,
  ERR_DATA_TRANSFER = 17,
  ERR_FIFO = 18,
  ERR_OVER_TEMP = 19,
  ERR_IDENTIFICATION = 20,
  ERR_COM_INTERFACE = 21,
  ERR_SYNCHRONISATION = 22,
  ERR_PROTOCOL = 23,
  ERR_MEMORY = 24,
  ERR_THREAD = 25,
  ERR_SPI = 26,
  ERR_DAC_ACCESS = 27,
  ERR_I2C = 28,
  ERR_NO_DATA = 29,
  ERR_SYSTEM_CONFIG = 30,
  ERR_USB_ACCESS = 31,
  ERR_ADC_ACCESS = 32,
  ERR_SENSOR_CONFIG = 33,
  ERR_SATURATION = 34,
  ERR_MUTEX = 35,
  ERR_ACCELEROMETER = 36,
  ERR_CONFIG = 37,
  ERR_BLE = 38,
  ERR_FILE = 39,
  ERR_DATA = 40,
  ERR_BUSY = 41 }

4.6.1 Detailed Description

Generic error codes used by ams libraries.

4.6.2 Typedef Documentation

4.6.2.1 err_code_t `typedef enum error_codes err_code_t`

This definition will be used for function return values.

4.6.3 Enumeration Type Documentation

4.6.3.1 error_codes `enum error_codes`

Error Codes.

Enumerator

ERR_SUCCESS	Operation was successful.
ERR_PERMISSION	Operation is not permitted.
ERR_MESSAGE	Message is invalid. (Unsupported message type, incorrect CRC, ...)
ERR_MESSAGE_SIZE	Message has the wrong size.
ERR_POINTER	Pointer is invalid. (NULL pointer, pointer wrong memory region, ...)
ERR_ACCESS	Access is denied.
ERR_ARGUMENT	Argument is invalid.
ERR_SIZE	An argument has the wrong size.
ERR_NOT_SUPPORTED	Function is not supported or not implemented.
ERR_TIMEOUT	Operation timed out.
ERR_CHECKSUM	Checksum comparision failed.
ERR_OVERFLOW	Data overflow occurred.
ERR_EVENT	Getting or setting an event failed. (Event queue is full or empty, unexpected event received, ...)
ERR_INTERRUPT	Getting or setting an interrupt failed. (Interrupt resource is not available, ...)
ERR_TIMER_ACCESS	Accessing the timer peripheral failed.
ERR_LED_ACCESS	Accessing the LED peripheral failed.
ERR_TEMP_SENSOR_ACCESS	Accessing the temperature sensor failed.
ERR_DATA_TRANSFER	Communication error occurred.
ERR_FIFO	FIFO operation failed.
ERR_OVER_TEMP	Overtemperature detected.

Enumerator

ERR_IDENTIFICATION	Sensor identification failed.
ERR_COM_INTERFACE	Generic communication interface error. (Communication interface is not available, error while opening or closing a communication interface, ...)
ERR_SYNCHRONISATION	Synchronization error occurred.
ERR_PROTOCOL	Generic protocol error occurred.
ERR_MEMORY	Memory allocation error occurred.
ERR_THREAD	Thread handling operation failed.
ERR_SPI	Accessing the SPI peripheral failed.
ERR_DAC_ACCESS	Accessing the DAC peripheral failed.
ERR_I2C	Accessing the I2C peripheral failed.
ERR_NO_DATA	No data available.
ERR_SYSTEM_CONFIG	System configuration failed. (System resource is not available, system resource generates an error, ...)
ERR_USB_ACCESS	Accessing the USB peripheral failed.
ERR_ADC_ACCESS	Accessing the ADC peripheral failed.
ERR_SENSOR_CONFIG	Sensor configuration failed.
ERR_SATURATION	Saturation detected.
ERR_MUTEX	Mutex handling operation filed.
ERR_ACCELEROMETER	Accessing the accelerometer failed.
ERR_CONFIG	Software component is unusable due to incomplete or incorrect configuration.
ERR_BLE	BLE stack handling operation failed.
ERR_FILE	File handling operation failed.
ERR_DATA	Internal data inconsistency detected.
ERR_BUSY	Module is busy.

5 Data Structure Documentation

5.1 as7050_config_afe_t::afe_regs Struct Reference

Data Fields

- uint8_t afe_dac0l
- uint8_t afe_dac1l
- uint8_t afe_dach
- uint8_t afe_cfga
- uint8_t afe_cfgb
- uint8_t afe_gsr

5.1.1 Detailed Description

Register content for AFE configuration

5.1.2 Field Documentation

5.1.2.1 afe_dac0l `uint8_t as7050_config_afe_t::afe_regs::afe_dac0l`

Content of register AS7050_REGADDR_AFE_DAC0L

5.1.2.2 afe_dac1l `uint8_t as7050_config_afe_t::afe_regs::afe_dac1l`

Content of register AS7050_REGADDR_AFE_DAC1L

5.1.2.3 afe_dach `uint8_t as7050_config_afe_t::afe_regs::afe_dach`

Content of register AS7050_REGADDR_AFE_DACH

5.1.2.4 afe_cfga `uint8_t as7050_config_afe_t::afe_regs::afe_cfga`

Content of register AS7050_REGADDR_AFE_CFGA

5.1.2.5 afe_cfgb `uint8_t as7050_config_afe_t::afe_regs::afe_cfgb`

Content of register AS7050_REGADDR_AFE_CFGB

5.1.2.6 afe_gsr `uint8_t as7050_config_afe_t::afe_regs::afe_gsr`

Content of register AS7050_REGADDR_AFE_GSR

5.2 as7050_config_amp_t::amp_regs Struct Reference

Data Fields

- uint8_t ecg_amp_cfga
- uint8_t ecg_amp_cfgb
- uint8_t ecg_amp_cfgc
- uint8_t ecg_amp_cfge

5.2.1 Detailed Description

Register content for amplifer configuration

5.2.2 Field Documentation

5.2.2.1 ecg_amp_cfga `uint8_t as7050_config_amp_t::amp_regs::ecg_amp_cfga`

Content of register AS7050_REGADDR_ECG_AMP_CFGA

5.2.2.2 ecg_amp_cfgb `uint8_t as7050_config_amp_t::amp_regs::ecg_amp_cfgb`

Content of register AS7050_REGADDR_ECG_AMP_CFGB

5.2.2.3 ecg_amp_cfgc `uint8_t as7050_config_amp_t::amp_regs::ecg_amp_cfgc`

Content of register AS7050_REGADDR_ECG_AMP_CFGC

5.2.2.4 ecg_amp_cfge `uint8_t as7050_config_amp_t::amp_regs::ecg_amp_cfge`

Content of register AS7050_REGADDR_ECG_AMP_CFGE

5.3 as7050_config_aoc_t::aoc_regs Struct Reference

Data Fields

- uint8_t aoc_ios_ppg1
- uint8_t aoc_ios_ppg2
- uint8_t aoc_ios_ppg3
- uint8_t aoc_ios_ppg4
- uint8_t aoc_ios_ppg5
- uint8_t aoc_ios_ppg6
- uint8_t aoc_ios_ppg7
- uint8_t aoc_ios_ppg8
- uint8_t aoc_ppg_thh
- uint8_t aoc_ppg_thl
- uint8_t aoc_ppg_cfg
- uint8_t aoc_ios_ecg
- uint8_t aoc_ecg_thh
- uint8_t aoc_ecg_thl
- uint8_t aoc_ecg_cfg
- uint8_t aoc_ios_ledoff

5.3.1 Detailed Description

Register content for AOC configuration

5.3.2 Field Documentation

5.3.2.1 aoc_ios_ppg1 `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ios_ppg1`

Content of register AS7050_REGADDR_AOC_IOS_PPG1

5.3.2.2 aoc_ios_ppg2 `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ios_ppg2`

Content of register AS7050_REGADDR_AOC_IOS_PPG2

5.3.2.3 aoc_ios_ppg3 `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ios_ppg3`

Content of register AS7050_REGADDR_AOC_IOS_PPG3

5.3.2.4 aoc_ios_ppg4 `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ios_ppg4`

Content of register AS7050_REGADDR_AOC_IOS_PPG4

5.3.2.5 aoc_ios_ppg5 `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ios_ppg5`

Content of register AS7050_REGADDR_AOC_IOS_PPG5

5.3.2.6 aoc_ios_ppg6 `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ios_ppg6`

Content of register AS7050_REGADDR_AOC_IOS_PPG6

5.3.2.7 aoc_ios_ppg7 `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ios_ppg7`

Content of register AS7050_REGADDR_AOC_IOS_PPG7

5.3.2.8 aoc_ios_ppg8 `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ios_ppg8`

Content of register AS7050_REGADDR_AOC_IOS_PPG8

5.3.2.9 aoc_ppg_thh `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ppg_thh`

Content of register AS7050_REGADDR_AOC_PPG_THH

5.3.2.10 aoc_ppg_thl `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ppg_thl`

Content of register AS7050_REGADDR_AOC_PPG_THL

5.3.2.11 aoc_ppg_cfg `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ppg_cfg`

Content of register AS7050_REGADDR_AOC_PPG_CFG

5.3.2.12 aoc_ios_ecg `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ios_ecg`

Content of register AS7050_REGADDR_AOC_IOS_ECG

5.3.2.13 aoc_ecg_thh `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ecg_thh`

Content of register AS7050_REGADDR_AOC_ECG_THH

5.3.2.14 aoc_ecg_thl `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ecg_thl`

Content of register AS7050_REGADDR_AOC_ECG_THL

5.3.2.15 aoc_ecg_cfg `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ecg_cfg`

Content of register AS7050_REGADDR_AOC_ECG_CFG

5.3.2.16 aoc_ios_ledoff `uint8_t as7050_config_aoc_t::aoc_regs::aoc_ios_ledoff`

Content of register AS7050_REGADDR_AOC_IOS_LEDOFF

5.4 as7050_config_afe_t Union Reference

Data Structures

- struct afe_regs

Data Fields

- struct as7050_config_afe_t::afe_regs reg_vals
- uint8_t reg_buffer [sizeof(struct afe_regs)]

5.4.1 Detailed Description

Register group for configuration of the Analog Front End (AFE)

5.4.2 Field Documentation

5.4.2.1 reg_vals `struct as7050_config_afe_t::afe_regs as7050_config_afe_t::reg_vals`

Register content for AFE configuration

5.4.2.2 reg_buffer `uint8_t as7050_config_afe_t::reg_buffer[sizeof(struct afe_regs)]`

Register content for AFE configuration

5.5 as7050_config_amp_t Union Reference

Data Structures

- struct amp_regs

Data Fields

- struct as7050_config_amp_t::amp_regs reg_vals
- uint8_t reg_buffer [sizeof(struct amp_regs)]

5.5.1 Detailed Description

Register group for configuration of the amplifer

5.5.2 Field Documentation

5.5.2.1 reg_vals `struct as7050_config_amp_t::amp_regs as7050_config_amp_t::reg_vals`

Register content for amplifer configuration

5.5.2.2 reg_buffer `uint8_t as7050_config_amp_t::reg_buffer[sizeof(struct amp_regs)]`

Register content for amplifer configuration

5.6 as7050_config_aoc_t Union Reference

Data Structures

- struct aoc_regs

Data Fields

- struct as7050_config_aoc_t::aoc_regs reg_vals
- uint8_t reg_buffer [sizeof(struct aoc_regs)]

5.6.1 Detailed Description

Register group for configuration of Automatic Offset Control (AOC)

5.6.2 Field Documentation

5.6.2.1 reg_vals `struct as7050_config_aoc_t::aoc_regs as7050_config_aoc_t::reg_vals`

Register content for AOC configuration

5.6.2.2 reg_buffer `uint8_t as7050_config_aoc_t::reg_buffer[sizeof(struct aoc_regs)]`

Register content for AOC configuration

5.7 as7050_config_ctrl_t Union Reference

Data Structures

- struct ctrl_regs

Data Fields

- struct as7050_config_ctrl_t::ctrl_regs reg_vals
- uint8_t reg_buffer [sizeof(struct ctrl_regs)]

5.7.1 Detailed Description

Register group for configuration of the CONTROL register

5.7.2 Field Documentation

5.7.2.1 reg_vals `struct as7050_config_ctrl_t::ctrl_regs as7050_config_ctrl_t::reg_vals`

Register content for CONTROL register

5.7.2.2 reg_buffer `uint8_t as7050_config_ctrl_t::reg_buffer[sizeof(struct ctrl_regs)]`

Register content for CONTROL register

5.8 as7050_config_ecg_t Union Reference

Data Structures

- struct ecg_regs

Data Fields

- struct as7050_config_ecg_t::ecg_regs reg_vals
- uint8_t reg_buffer [sizeof(struct ecg_regs)]

5.8.1 Detailed Description

Register group for configuration of the ECG channel

5.8.2 Field Documentation

5.8.2.1 reg_vals `struct as7050_config_ecg_t::ecg_regs as7050_config_ecg_t::reg_vals`

Register content for ECG configuration

5.8.2.2 reg_buffer `uint8_t as7050_config_ecg_t::reg_buffer[sizeof(struct ecg_regs)]`

Register content for ECG configuration

5.9 as7050_config_fifo_t Union Reference

Data Structures

- struct fifo_regs

Data Fields

- struct as7050_config_fifo_t::fifo_regs reg_vals
- uint8_t reg_buffer [sizeof(struct fifo_regs)]

5.9.1 Detailed Description

Register group for configuration of the FIFO handling

5.9.2 Field Documentation

5.9.2.1 reg_vals `struct as7050_config_fifo_t::fifo_regs as7050_config_fifo_t::reg_vals`

Register content for FIFO configuration

5.9.2.2 reg_buffer `uint8_t as7050_config_fifo_t::reg_buffer[sizeof(struct fifo_regs)]`

Register content for FIFO configuration

5.10 as7050_config_gpio_t Union Reference

Data Structures

- struct gpio_regs

Data Fields

- struct as7050_config_gpio_t::gpio_regs reg_vals
- uint8_t reg_buffer [sizeof(struct gpio_regs)]

5.10.1 Detailed Description

Register group for configuration of the GPIOs

5.10.2 Field Documentation

5.10.2.1 reg_vals `struct as7050_config_gpio_t::gpio_regs as7050_config_gpio_t::reg_vals`

Register content for GPIO configuration

5.10.2.2 reg_buffer `uint8_t as7050_config_gpio_t::reg_buffer[sizeof(struct gpio_regs)]`

Register content for GPIO configuration

5.11 as7050_config_iir_t Union Reference

Data Structures

- struct iir_regs

Data Fields

- struct as7050_config_iir_t::iir_regs reg_vals
- uint8_t reg_buffer [sizeof(struct iir_regs)]

5.11.1 Detailed Description

Register group for configuration of the Infinite Impulse Response filter (IIR-filter)

5.11.2 Field Documentation

5.11.2.1 reg_vals `struct as7050_config_iir_t::iir_regs as7050_config_iir_t::reg_vals`

Register content for IIR configuration

5.11.2.2 reg_buffer `uint8_t as7050_config_iir_t::reg_buffer[sizeof(struct iir_regs)]`

Register content for IIR configuration

5.12 as7050_config_led_t Union Reference

Data Structures

- struct led_regs

Data Fields

- struct as7050_config_led_t::led_regs reg_vals
- uint8_t reg_buffer [sizeof(struct led_regs)]

5.12.1 Detailed Description

Register group for LED configuration

5.12.2 Field Documentation

5.12.2.1 reg_vals `struct as7050_config_led_t::led_regs as7050_config_led_t::reg_vals`

Register content for LED configuration

5.12.2.2 reg_buffer `uint8_t as7050_config_led_t::reg_buffer[sizeof(struct led_regs)]`

Register content for LED configuration

5.13 as7050_config_pd_t Union Reference

Data Structures

- struct pd_regs

Data Fields

- struct as7050_config_pd_t::pd_regs reg_vals
- uint8_t reg_buffer [sizeof(struct pd_regs)]

5.13.1 Detailed Description

Register group for configuration of the photodiodes

5.13.2 Field Documentation

5.13.2.1 reg_vals `struct as7050_config_pd_t::pd_regs as7050_config_pd_t::reg_vals`

Register content for photodiodes configuration

5.13.2.2 reg_buffer `uint8_t as7050_config_pd_t::reg_buffer[sizeof(struct pd_regs)]`

Register content for photodiodes configuration

5.14 as7050_config_ppg_t Union Reference

Data Structures

- struct ppg_regs

Data Fields

- struct as7050_config_ppg_t::ppg_regs reg_vals
- uint8_t reg_buffer [sizeof(struct ppg_regs)]

5.14.1 Detailed Description

Register group for configuration of the PPG channels

5.14.2 Field Documentation

5.14.2.1 reg_vals `struct as7050_config_ppg_t::ppg_regs as7050_config_ppg_t::reg_vals`

Register content for PPG configuration

5.14.2.2 reg_buffer `uint8_t as7050_config_ppg_t::reg_buffer[sizeof(struct ppg_regs)]`

Register content for PPG configuration

5.15 as7050_config_ref_t Union Reference

Data Structures

- struct ref_regs

Data Fields

- struct as7050_config_ref_t::ref_regs reg_vals
- uint8_t reg_buffer [sizeof(struct ref_regs)]

5.15.1 Detailed Description

Register group for configuration of the reference registers

5.15.2 Field Documentation

5.15.2.1 reg_vals `struct as7050_config_ref_t::ref_regs as7050_config_ref_t::reg_vals`

Register content for REF-register configuration

5.15.2.2 reg_buffer `uint8_t as7050_config_ref_t::reg_buffer[sizeof(struct ref_regs)]`

Register content for REF-register configuration

5.16 as7050_config_seq_t Union Reference

Data Structures

- struct seq_regs

Data Fields

- struct as7050_config_seq_t::seq_regs reg_vals
- uint8_t reg_buffer [sizeof(struct seq_regs)]

5.16.1 Detailed Description

Register group for configuration of the sequencer

5.16.2 Field Documentation

5.16.2.1 reg_vals `struct as7050_config_seq_t::seq_regs as7050_config_seq_t::reg_vals`

Register content for sequencer configuration

5.16.2.2 reg_buffer `uint8_t as7050_config_seq_t::reg_buffer[sizeof(struct seq_regs)]`

Register content for sequencer configuration

5.17 as7050_config_sinc_t Union Reference

Data Structures

- struct sinc_regs

Data Fields

- struct as7050_config_sinc_t::sinc_regs reg_vals
- uint8_t reg_buffer [sizeof(struct sinc_regs)]

5.17.1 Detailed Description

Register group for configuration of the SINC filter

5.17.2 Field Documentation

5.17.2.1 reg_vals `struct as7050_config_sinc_t::sinc_regs as7050_config_sinc_t::reg_vals`

Register content for SINC filter configuration

5.17.2.2 reg_buffer `uint8_t as7050_config_sinc_t::reg_buffer[sizeof(struct sinc_regs)]`

Register content for SINC filter configuration

5.18 as7050_config_standby_t Union Reference

Data Structures

- struct standby_regs

Data Fields

- struct as7050_config_standby_t::standby_regs reg_vals
- uint8_t reg_buffer [sizeof(struct standby_regs)]

5.18.1 Detailed Description

Register group for configuration for STANDBY mode

5.18.2 Field Documentation

5.18.2.1 reg_vals `struct as7050_config_standby_t::standby_regs as7050_config_standby_t::reg_vals`

Register content for STANDBY configuration

5.18.2.2 reg_buffer `uint8_t as7050_config_standby_t::reg_buffer[sizeof(struct standby_regs)]`

Register content for STANDBY configuration

5.19 as7050_config_tia_t Union Reference

Data Structures

- struct tia_regs

Data Fields

- struct as7050_config_tia_t::tia_regs reg_vals
- uint8_t reg_buffer [sizeof(struct tia_regs)]

5.19.1 Detailed Description

Register group for configuration of the Transimpedance Amplifer (TIA)

5.19.2 Field Documentation

5.19.2.1 reg_vals `struct as7050_config_tia_t::tia_regs as7050_config_tia_t::reg_vals`

Register content for TIA configuration

5.19.2.2 reg_buffer `uint8_t as7050_config_tia_t::reg_buffer[sizeof(struct tia_regs)]`

Register content for TIA configuration

5.20 as7050_meas_config_t Struct Reference

Data Fields

- uint32_t ppg_sample_period_us
- uint32_t ecg_sample_period_us
- uint32_t max_adc_count
- uint16_t fifo_map
- uint16_t fifo_threshold
- uint8_t sample_size
- uint8_t max_adc_bit_width
- uint16_t reserved

5.20.1 Detailed Description

Measurement settings, which can be readout after register configuration

5.20.2 Field Documentation

5.20.2.1 ppg_sample_period_us `uint32_t as7050_meas_config_t::ppg_sample_period_us`

Sample period of PPG signals in microseconds

5.20.2.2 ecg_sample_period_us `uint32_t as7050_meas_config_t::ecg_sample_period_us`

Sample period of ECG signals in microseconds

5.20.2.3 max_adc_count `uint32_t as7050_meas_config_t::max_adc_count`

Maximum possible ADC count for the current configuration

5.20.2.4 fifo_map `uint16_t as7050_meas_config_t::fifo_map`

Definition which channels are mapped inside FIFO. (More than one flag can be set) See as7050_channel_flags_t

5.20.2.5 fifo_threshold `uint16_t as7050_meas_config_t::fifo_threshold`

FIFO threshold, when data shall be read

5.20.2.6 sample_size `uint8_t as7050_meas_config_t::sample_size`

3 or 4 bytes for every sample

5.20.2.7 max_adc_bit_width `uint8_t as7050_meas_config_t::max_adc_bit_width`

Maximum bit width of the ADC count for the current configuration

5.20.2.8 reserved `uint16_t as7050_meas_config_t::reserved`

only for alignment, not used. Always set to 0

5.21 as7050_version Struct Reference

Data Fields

- uint8_t major
- uint8_t minor
- uint8_t patch

5.21.1 Detailed Description

Version information of the library

5.21.2 Field Documentation

5.21.2.1 major `uint8_t as7050_version::major`

Major version position

5.21.2.2 minor `uint8_t as7050_version::minor`

Minor version position

5.21.2.3 patch `uint8_t as7050_version::patch`

Patch version position

5.22 as7050_config_ctrl_t::ctrl_regs Struct Reference

Data Fields

- uint8_t control

5.22.1 Detailed Description

Register content for CONTROL register

5.22.2 Field Documentation

5.22.2.1 control `uint8_t as7050_config_ctrl_t::ctrl_regs::control`

Content of register AS7050_REGADDR_CONTROL

5.23 as7050_config_ecg_t::ecg_regs Struct Reference

Data Fields

- uint8_t ecg_source
- uint8_t ecg_mod_cfga

5.23.1 Detailed Description

Register content for ECG configuration

5.23.2 Field Documentation

5.23.2.1 ecg_source `uint8_t as7050_config_ecg_t::ecg_regs::ecg_source`

Content of register AS7050_REGADDR_ECG_SOURCE

5.23.2.2 ecg_mod_cfga `uint8_t as7050_config_ecg_t::ecg_regs::ecg_mod_cfga`

Content of register AS7050_REGADDR_ECG_MOD_CFGA

5.24 as7050_config_fifo_t::fifo_regs Struct Reference

Data Fields

- uint8_t fifo_ctrl
- uint8_t fifo_threshold

5.24.1 Detailed Description

Register content for FIFO configuration

5.24.2 Field Documentation

5.24.2.1 fifo_ctrl `uint8_t as7050_config_fifo_t::fifo_regs::fifo_ctrl`

Content of register AS7050_REGADDR_FIFO_CTRL

5.24.2.2 fifo_threshold `uint8_t as7050_config_fifo_t::fifo_regs::fifo_threshold`

Content of register AS7050_REGADDR_FIFO_THRESHOLD

5.25 as7050_config_gpio_t::gpio_regs Struct Reference

Data Fields

- uint8_t gpio1_cfg
- uint8_t gpio2_cfg
- uint8_t gpio1_cfgb
- uint8_t gpio2_cfgb
- uint8_t gpio_io

5.25.1 Detailed Description

Register content for GPIO configuration

5.25.2 Field Documentation

5.25.2.1 gpio1_cfg `uint8_t as7050_config_gpio_t::gpio_regs::gpio1_cfg`

Content of register AS7050_REGADDR_GPIO1_CFG

5.25.2.2 gpio2_cfg `uint8_t as7050_config_gpio_t::gpio_regs::gpio2_cfg`

Content of register AS7050_REGADDR_GPIO2_CFG

5.25.2.3 gpio1_cfgb `uint8_t as7050_config_gpio_t::gpio_regs::gpio1_cfgb`

Content of register AS7050_REGADDR_GPIO1_CFGB

5.25.2.4 gpio2_cfgb `uint8_t as7050_config_gpio_t::gpio_regs::gpio2_cfgb`

Content of register AS7050_REGADDR_GPIO2_CFGB

5.25.2.5 gpio_io `uint8_t as7050_config_gpio_t::gpio_regs::gpio_io`

Content of register AS7050_REGADDR_GPIO_IO

5.26 as7050_config_iir_t::iir_regs Struct Reference

Data Fields

- uint8_t iir_cfg
- int16_t iir_coeff_data_sos [12][5]

5.26.1 Detailed Description

Register content for IIR configuration

5.26.2 Field Documentation

5.26.2.1 iir_cfg `uint8_t as7050_config_iir_t::iir_regs::iir_cfg`

Content of register AS7050_REGADDR_IIR_CFG

5.26.2.2 iir_coeff_data_sos `int16_t as7050_config_iir_t::iir_regs::iir_coeff_data_sos[12][5]`

RAM area where IIR-coefficients will be saved

5.27 as7050_config_led_t::led_regs Struct Reference

Data Fields

- uint8_t lowvds_wait
- uint8_t led1_ictrl
- uint8_t led2_ictrl
- uint8_t led3_ictrl
- uint8_t led4_ictrl
- uint8_t led5_ictrl
- uint8_t led6_ictrl
- uint8_t led7_ictrl
- uint8_t led8_ictrl
- uint8_t led_init
- uint8_t led_ppg1
- uint8_t led_ppg2
- uint8_t led_ppg3
- uint8_t led_ppg4
- uint8_t led_ppg5
- uint8_t led_ppg6
- uint8_t led_ppg7
- uint8_t led_ppg8
- uint8_t led_tia
- uint8_t led_mode

5.27.1 Detailed Description

Register content for LED configuration

5.27.2 Field Documentation

5.27.2.1 lowvds_wait `uint8_t as7050_config_led_t::led_regs::lowvds_wait`

Content of register AS7050_REGADDR_LOWVDS_WAIT

5.27.2.2 led1_ictrl `uint8_t as7050_config_led_t::led_regs::led1_ictrl`

Content of register AS7050_REGADDR_LED1_ICTRL

5.27.2.3 led2_ictrl `uint8_t as7050_config_led_t::led_regs::led2_ictrl`

Content of register AS7050_REGADDR_LED2_ICTRL

5.27.2.4 led3_ictrl `uint8_t as7050_config_led_t::led_regs::led3_ictrl`

Content of register AS7050_REGADDR_LED3_ICTRL

5.27.2.5 led4_ictrl `uint8_t as7050_config_led_t::led_regs::led4_ictrl`

Content of register AS7050_REGADDR_LED4_ICTRL

5.27.2.6 led5_ictrl `uint8_t as7050_config_led_t::led_regs::led5_ictrl`

Content of register AS7050_REGADDR_LED5_ICTRL

5.27.2.7 led6_ictrl `uint8_t as7050_config_led_t::led_regs::led6_ictrl`

Content of register AS7050_REGADDR_LED6_ICTRL

5.27.2.8 led7_ictrl `uint8_t as7050_config_led_t::led_regs::led7_ictrl`

Content of register AS7050_REGADDR_LED7_ICTRL

5.27.2.9 led8_ictrl `uint8_t as7050_config_led_t::led_regs::led8_ictrl`

Content of register AS7050_REGADDR_LED8_ICTRL

5.27.2.10 led_init `uint8_t as7050_config_led_t::led_regs::led_init`

Content of register AS7050_REGADDR_LED_INIT

5.27.2.11 led_ppg1 `uint8_t as7050_config_led_t::led_regs::led_ppg1`

Content of register AS7050_REGADDR_LED_PPG1

5.27.2.12 led_ppg2 `uint8_t as7050_config_led_t::led_regs::led_ppg2`

Content of register AS7050_REGADDR_LED_PPG2

5.27.2.13 led_ppg3 `uint8_t as7050_config_led_t::led_regs::led_ppg3`

Content of register AS7050_REGADDR_LED_PPG3

5.27.2.14 led_ppg4 `uint8_t as7050_config_led_t::led_regs::led_ppg4`

Content of register AS7050_REGADDR_LED_PPG4

5.27.2.15 led_ppg5 `uint8_t as7050_config_led_t::led_regs::led_ppg5`

Content of register AS7050_REGADDR_LED_PPG5

5.27.2.16 led_ppg6 `uint8_t as7050_config_led_t::led_regs::led_ppg6`

Content of register AS7050_REGADDR_LED_PPG6

5.27.2.17 led_ppg7 `uint8_t as7050_config_led_t::led_regs::led_ppg7`

Content of register AS7050_REGADDR_LED_PPG7

5.27.2.18 led_ppg8 `uint8_t as7050_config_led_t::led_regs::led_ppg8`

Content of register AS7050_REGADDR_LED_PPG8

5.27.2.19 led_tia `uint8_t as7050_config_led_t::led_regs::led_tia`

Content of register AS7050_REGADDR_LED_TIA

5.27.2.20 led_mode `uint8_t as7050_config_led_t::led_regs::led_mode`

Content of register AS7050_REGADDR_LED_MODE

5.28 obj_access_table_entry_t Struct Reference

Data Fields

- obj_id_t obj_id
- err_code_t(∗ obj_fct_pnt_read)(uint16_t subindex, uint8_t read_len, uint8_t ∗answer_payload, uint8_t ∗answer_len)
- err_code_t(∗ obj_fct_pnt_write)(uint16_t subindex, uint8_t ∗input_data, uint8_t input_len)

5.28.1 Detailed Description

Table entry for the BLE object table

5.28.2 Field Documentation

5.28.2.1 obj_id `obj_id_t obj_access_table_entry_t::obj_id`

Object ID ; see obj_id_t

5.28.2.2 obj_fct_pnt_read `err_code_t(∗ obj_access_table_entry_t::obj_fct_pnt_read) (uint16_t subindex, uint8_t read_len, uint8_t ∗answer_payload, uint8_t ∗answer_len)`

function pointer to the function processing this object access request

5.28.2.3 obj_fct_pnt_write `err_code_t(∗ obj_access_table_entry_t::obj_fct_pnt_write) (uint16_t subindex, uint8_t ∗input_data, uint8_t input_len)`

function pointer to the function processing this object access request

5.29 as7050_config_pd_t::pd_regs Struct Reference

Data Fields

- uint8_t pdsel_cfg
- uint8_t pd_ppg1
- uint8_t pd_ppg2
- uint8_t pd_ppg3
- uint8_t pd_ppg4
- uint8_t pd_ppg5
- uint8_t pd_ppg6
- uint8_t pd_ppg7
- uint8_t pd_ppg8
- uint8_t pd_tia

5.29.1 Detailed Description

Register content for photodiodes configuration

5.29.2 Field Documentation

5.29.2.1 pdsel_cfg `uint8_t as7050_config_pd_t::pd_regs::pdsel_cfg`

Content of register AS7050_REGADDR_PDSEL_CFG

5.29.2.2 pd_ppg1 `uint8_t as7050_config_pd_t::pd_regs::pd_ppg1`

Content of register AS7050_REGADDR_PD_PPG1

5.29.2.3 pd_ppg2 `uint8_t as7050_config_pd_t::pd_regs::pd_ppg2`

Content of register AS7050_REGADDR_PD_PPG2

5.29.2.4 pd_ppg3 `uint8_t as7050_config_pd_t::pd_regs::pd_ppg3`

Content of register AS7050_REGADDR_PD_PPG3

5.29.2.5 pd_ppg4 `uint8_t as7050_config_pd_t::pd_regs::pd_ppg4`

Content of register AS7050_REGADDR_PD_PPG4

5.29.2.6 pd_ppg5 `uint8_t as7050_config_pd_t::pd_regs::pd_ppg5`

Content of register AS7050_REGADDR_PD_PPG5

5.29.2.7 pd_ppg6 `uint8_t as7050_config_pd_t::pd_regs::pd_ppg6`

Content of register AS7050_REGADDR_PD_PPG6

5.29.2.8 pd_ppg7 `uint8_t as7050_config_pd_t::pd_regs::pd_ppg7`

Content of register AS7050_REGADDR_PD_PPG7

5.29.2.9 pd_ppg8 `uint8_t as7050_config_pd_t::pd_regs::pd_ppg8`

Content of register AS7050_REGADDR_PD_PPG8

5.29.2.10 pd_tia `uint8_t as7050_config_pd_t::pd_regs::pd_tia`

Content of register AS7050_REGADDR_PD_TIA

5.30 as7050_config_ppg_t::ppg_regs Struct Reference

Data Fields

- uint8_t ppg_mod_cfga
- uint8_t ppg_mod_cfgb
- uint8_t ppg_mod_cfgc
- uint8_t ppg_mod_cfgd
- uint8_t ppg_mod_cfge

5.30.1 Detailed Description

Register content for PPG configuration

5.30.2 Field Documentation

5.30.2.1 ppg_mod_cfga `uint8_t as7050_config_ppg_t::ppg_regs::ppg_mod_cfga`

Content of register AS7050_REGADDR_PPG_MOD_CFGA

5.30.2.2 ppg_mod_cfgb `uint8_t as7050_config_ppg_t::ppg_regs::ppg_mod_cfgb`

Content of register AS7050_REGADDR_PPG_MOD_CFGB

5.30.2.3 ppg_mod_cfgc `uint8_t as7050_config_ppg_t::ppg_regs::ppg_mod_cfgc`

Content of register AS7050_REGADDR_PPG_MOD_CFGC

5.30.2.4 ppg_mod_cfgd `uint8_t as7050_config_ppg_t::ppg_regs::ppg_mod_cfgd`

Content of register AS7050_REGADDR_PPG_MOD_CFGD

5.30.2.5 ppg_mod_cfge `uint8_t as7050_config_ppg_t::ppg_regs::ppg_mod_cfge`

Content of register AS7050_REGADDR_PPG_MOD_CFGE

5.31 as7050_config_ref_t::ref_regs Struct Reference

Data Fields

- uint8_t ref_cfga
- uint8_t ref_cfgb

5.31.1 Detailed Description

Register content for REF-register configuration

5.31.2 Field Documentation

5.31.2.1 ref_cfga `uint8_t as7050_config_ref_t::ref_regs::ref_cfga`

Content of register AS7050_REGADDR_REF_CFGA

5.31.2.2 ref_cfgb `uint8_t as7050_config_ref_t::ref_regs::ref_cfgb`

Content of register AS7050_REGADDR_REF_CFGB

5.32 as7050_config_seq_t::seq_regs Struct Reference

Data Fields

- uint8_t cgb_cfg
- uint8_t seq_sample
- uint8_t seq_ppga
- uint8_t seq_ppgb
- uint8_t seq_mode

5.32.1 Detailed Description

Register content for sequencer configuration

5.32.2 Field Documentation

5.32.2.1 cgb_cfg `uint8_t as7050_config_seq_t::seq_regs::cgb_cfg`

Content of register AS7050_REGADDR_CGB_CFG

5.32.2.2 seq_sample `uint8_t as7050_config_seq_t::seq_regs::seq_sample`

Content of register AS7050_REGADDR_SEQ_SAMPLE

5.32.2.3 seq_ppga `uint8_t as7050_config_seq_t::seq_regs::seq_ppga`

Content of register AS7050_REGADDR_SEQ_PPGA

5.32.2.4 seq_ppgb `uint8_t as7050_config_seq_t::seq_regs::seq_ppgb`

Content of register AS7050_REGADDR_SEQ_PPGB

5.32.2.5 seq_mode `uint8_t as7050_config_seq_t::seq_regs::seq_mode`

Content of register AS7050_REGADDR_SEQ_MODE

5.33 as7050_config_sinc_t::sinc_regs Struct Reference

Data Fields

- uint8_t sinc_ppg_cfga
- uint8_t sinc_ppg_cfgb
- uint8_t sinc_ppg_cfgc
- uint8_t sinc_ecg_cfga
- uint8_t sinc_ecg_cfgb
- uint8_t sinc_ecg_cfgc
- uint8_t ovs_cfg

5.33.1 Detailed Description

Register content for SINC filter configuration

5.33.2 Field Documentation

5.33.2.1 sinc_ppg_cfga `uint8_t as7050_config_sinc_t::sinc_regs::sinc_ppg_cfga`

Content of register AS7050_REGADDR_SINC_PPG_CFGA

5.33.2.2 sinc_ppg_cfgb `uint8_t as7050_config_sinc_t::sinc_regs::sinc_ppg_cfgb`

Content of register AS7050_REGADDR_SINC_PPG_CFGB

5.33.2.3 sinc_ppg_cfgc `uint8_t as7050_config_sinc_t::sinc_regs::sinc_ppg_cfgc`

Content of register AS7050_REGADDR_SINC_PPG_CFGC

5.33.2.4 sinc_ecg_cfga `uint8_t as7050_config_sinc_t::sinc_regs::sinc_ecg_cfga`

Content of register AS7050_REGADDR_SINC_ECG_CFGA

5.33.2.5 sinc_ecg_cfgb `uint8_t as7050_config_sinc_t::sinc_regs::sinc_ecg_cfgb`

Content of register AS7050_REGADDR_SINC_ECG_CFGB

5.33.2.6 sinc_ecg_cfgc `uint8_t as7050_config_sinc_t::sinc_regs::sinc_ecg_cfgc`

Content of register AS7050_REGADDR_SINC_ECG_CFGC

5.33.2.7 ovs_cfg `uint8_t as7050_config_sinc_t::sinc_regs::ovs_cfg`

Content of register AS7050_REGADDR_OVS_CFG

5.34 as7050_config_standby_t::standby_regs Struct Reference

Data Fields

- uint8_t standby_cfga
- uint8_t standby_cfgb

5.34.1 Detailed Description

Register content for STANDBY configuration

5.34.2 Field Documentation

5.34.2.1 standby_cfga `uint8_t as7050_config_standby_t::standby_regs::standby_cfga`

Content of register AS7050_REGADDR_STANDBY_CFGA

5.34.2.2 standby_cfgb `uint8_t as7050_config_standby_t::standby_regs::standby_cfgb`

Content of register AS7050_REGADDR_STANDBY_CFGB

5.35 as7050_config_tia_t::tia_regs Struct Reference

Data Fields

- uint8_t pd_offset_cfg
- uint8_t tia_cfga
- uint8_t tia_cfgb
- uint8_t tia_cfgc

5.35.1 Detailed Description

Register content for TIA configuration

5.35.2 Field Documentation

5.35.2.1 pd_offset_cfg `uint8_t as7050_config_tia_t::tia_regs::pd_offset_cfg`

Content of register AS7050_REGADDR_PD_OFFSET_CFG

5.35.2.2 tia_cfga `uint8_t as7050_config_tia_t::tia_regs::tia_cfga`

Content of register AS7050_REGADDR_TIA_CFGA

5.35.2.3 tia_cfgb `uint8_t as7050_config_tia_t::tia_regs::tia_cfgb`

Content of register AS7050_REGADDR_TIA_CFGB

5.35.2.4 tia_cfgc `uint8_t as7050_config_tia_t::tia_regs::tia_cfgc`

Content of register AS7050_REGADDR_TIA_CFGC

Index