AS7058 Firmware RPC Protocol Description

Firmware Version: 3.3.0
2023-07-12

ams-OSRAM AG
Tobelbader Strasse 30
8141 Premstaetten, Austria
T +43 3136 500-0
ams-osram.com

Table of Contents

List of Tables

Glossary

Term	Description
AC	Alternating current
ADC	Analog-to-digital converter
AGC	Auto Gain Control
BioZ	Bioimpedance
BLE	Bluetooth Low Energy
CCCD	Client Characteristic Configuration Descriptor
CDC ACM	Communications Device Class Abstract Control Model
DAC	Digital-to-analog converter
DC	Direct current
ECG	Electrocardiogram
EDA	Electrodermal activity
FIFO	First in, first out
GAP	Generic Access Profile
GATT	Generic Attribute Profile
HRM	Heart rate monitoring
I/Q	In-Phase/Quadrature
I2C	Inter-Integrated Circuit
IIR	Infinite impulse response
LED	Light-emitting diode
MTU	Maximum Transmission Unit
OTP	One-Time Programmable
PD	Photodiode
PIO	Digital input/output pin
PPG	Photoplethysmogram
PRV	Pulse rate variability
PWM	Pulse width modulation
RPC	Remote Procedure Call
RRM	Respiration Rate Monitoring

Term	Description
SPI	Serial Peripheral Interface
SpO2	Peripheral oxygen saturation
USB	Universal Serial Bus
VCSEL	Vertical-cavity surface-emitting laser

AS7058 Firmware

The AS7058 firmware is used on AS7058 evaluation hardware, such as the AS7058 EVK. It is to perform measurements using the AS7058 AFE and the AS7058 software components. The firmware contains the Chip Library, the Application Manager, and Bio Applications with Vital Signs Algorithms.

As a CSS Core Firmware-based firmware, it implements the RPC Protocol and exposes it via Bluetooth Low Energy and USB. Bluetooth Low Energy is mainly used to access the firmware via iOS and Android mobile apps. USB is the main interface used by the AS7058 desktop GUI application.

For identification purposes, all firmware based on the CSS Core Firmware uses several IDs. The ID values used by the AS7058 firmware are described in the table below.

Table 1: IDs used by AS7058 Firmware

Name	Description	Value
USB Product ID	16-bit product ID as required by USB specifications. This value is unique within ams OSRAM.	0x100A
Product Sub-ID	16-bit product-specific sub-ID that is used for further firmware product identification. The AS7058 firmware always uses the same value.	0x0000

CSS Core Firmware

The CSS Core Firmware is a collection of reusable software components used in many product-specific ams OSRAM firmware. It supports Nordic Semiconductor nRF52840/nRF52832 (Nordic platform) and STMicroelectronics STM32F413 (Unicom platform) microcontrollers.

Among other components, the CSS Core Firmware includes:

- microcontroller peripheral drivers with platform-agnostic interfaces,
- support for custom PCBs via board support packages,
- a bootloader for Nordic Semiconductor nRF52840/nRF52832 microcontrollers,
- an extendable super loop implementation that already handles initialization of the included software components and provides user feedback via status LEDs, and
- an extendable communication interface called RPC Protocol that is exposed via USB and Bluetooth Low Energy.

It is also possible to use the CSS Core Firmware as a standalone firmware that includes all its software components but no additional product-specific components.

RPC Protocol

The RPC Protocol is a serial protocol for communication between a Host Device such as a computer or mobile phone and an embedded device running CSS Core Firmware components (CSS Core Firmware Device). By default, the CSS Core Firmware implements system control commands and commands which expose its microcontroller peripheral drivers. Product-specific firmware can implement additional commands.

The RPC Protocol can be used via USB or Bluetooth Low Energy. While the RPC Protocol is only available when the CSS Core Firmware Device is not in bootloader mode, the additional device identification methods described in this document are also available in bootloader mode.

RPC Message

An RPC Message consists of five fields. Depending on the communication layer, the data inside some of the fields may be encoded in different fields, and additional fields may be added by the communication layer-specific protocol.

Table 2: RPC Message Fields

Name	Description
Command ID	Identifies the command that is being sent.
Target ID	Usually addresses a specific instance of a software component. The exact meaning of this field is command-specific.
Error Code	Contains an error code (see Table 3).
Payload Length	Size of the payload.
Payload	Command-specific payload.

Table 3: Error Codes

Value	Description
0	Operation was successful.
1	Operation is not permitted.
2	Message is invalid. (Unsupported message type, incorrect CRC, …)
3	Message has the wrong size.
4	Pointer is invalid. (NULL pointer, pointer wrong memory region, …)
5	Access is denied.
6	Argument is invalid.

Table 3 (Continued): Error Codes

Value	Description
7	An argument has the wrong size.
8	Function is not supported or not implemented.
9	Operation timed out.
10	Checksum comparision failed.
11	Data overflow occurred.
12	Getting or setting an event failed. (Event queue is full or empty, unexpected event received, ...)
13	Getting or setting an interrupt failed. (Interrupt resource is not available, ...)
14	Accessing the timer peripheral failed.
15	Accessing the LED peripheral failed.
16	Accessing the temperature sensor failed.
17	Communication error occurred.
18	FIFO operation failed.
19	Overtemperature detected.
20	Sensor identification failed.
21	Generic communication interface error. (Communication interface is not available, error while opening or closing a communication interface, ...)
22	Synchronization error occurred.
23	Generic protocol error occurred.
24	Memory allocation error occurred.
25	Thread handling operation failed.
26	Accessing the SPI peripheral failed.
27	Accessing the DAC peripheral failed.
28	Accessing the I2C peripheral failed.
29	No data available.
30	System configuration failed. (System resource is not available, system resource generates an error, ...)
31	Accessing the USB peripheral failed.
32	Accessing the ADC peripheral failed.
33	Sensor configuration failed.
34	Saturation detected.
35	Mutex handling operation filed.

Table 3 (Continued): Error Codes

Value	Description
36	Accessing the accelerometer failed.
37	Software component is unusable due to incomplete or incorrect configuration.
38	BLE stack handling operation failed.
39	File handling operation failed.
40	Internal data inconsistency detected.
41	Module is busy.

Communication Flows

There are two flows of communication defined.

Synchronous Communication

The Host Device sends an RPC Message to the CSS Core Firmware Device. (Request) The CSS Core Firmware Device processes the received message and sends an RPC Message to the Host Device. (Response)

In the Request message, the value of the Error Code field must always be 0 (see Table 3). In the Response message, the value of the Error Code field is 0 (see Table 3) if the received message was processed successfully. A non-success error code was used when processing failed. The Response message always contains the Command ID and Target ID values of the Request message.

During regular operation, no subsequent Request must be sent until a Response is received. In case the Host Device suspects that a message has been lost during transmission, the Host Device may retry communication by sending a new Request.

Asynchronous Communication

The CSS Core Firmware Device sends a Response without receiving a corresponding Request.

Asynchronous messages are also assigned a Command ID that is used in the corresponding field. The value of the Error Code field is 0 (see Table 3) if no error occurred. A non-success error code is used when an error occurred.

When a synchronous Request message is received, an asynchronous Response message may be sent before sending the synchronous Response message.

USB Interface

To expose the RPC Protocol via USB, the CSS Core Firmware implements a USB Communications Device Class (CDC) Abstract Control Model (ACM) device. This is commonly known as a “COM port” or “serial device”. Modern operating systems support communication with USB CDC ACM devices without the installation of product-specific drivers.

Identification

The USB device descriptor of a CSS Core Firmware-based device contains USB Vendor ID 0x1325 (ams-OSRAM AG) and a product-specific USB Product ID. The generic, non-product-specific version of the CSS Core Firmware uses USB Product ID 0x1000. When the device is in bootloader mode, it always uses USB Product ID 0x1003. Furthermore, the serial number of the device can be obtained via the device descriptor.

Additionally, nRF52840-based devices provide a custom USB string descriptor at index 0x80 in both bootloader mode and non-bootloader mode. USB string descriptors are always encoded using UTF-16LE. This string contains the USB Product ID and an additional 16-bit Product Sub-ID value. The lower 15 bits of the Product Sub-ID have a product-specific meaning. The most significant bit (Bit 15) of the Product Sub-ID is set when the device is in bootloader mode and is unset otherwise. When in bootloader mode, the values of the USB Product ID, and the lower 15 bits of the Product Sub-ID, are identical to the values used in the non-bootloader mode. (Note: When the device is in bootloader mode, because there is no firmware installed, the USB Product ID and the Product Sub-ID are both set to 0xFFFF.)

Table 4: Format of USB String Descriptor 0x80

Character Index	Description
1:0	String format, should always be “01”.
5:2	USB Product ID as a string in hexadecimal representation without leading “0x”.
9:6	Product Sub-ID as a string in hexadecimal representation without leading “0x”.

Protocol

When communicating with the device, the RTS (Request To Send) control signal needs to be sent.

An RPC Message sent over USB always contains all the fields in the original specified order. It also contains a synchronization byte at the start of the message and a CRC-16-CCITT checksum at the end of the message. The synchronization byte always contains the value 0x55. The checksum is calculated over all the bytes of the message, except the checksum bytes. No padding bytes are inserted between the fields.

Table 5: Data Types of RPC Message Fields (USB)

Name	Data Type
Synchronization Byte	Unsigned 8-bit integer.
Command ID	Unsigned 8-bit integer.
Target ID	Unsigned 8-bit integer.
Error Code	Unsigned 8-bit integer.
Payload Length	Unsigned 32-bit little-endian integer.
Payload	Array of unsigned 8-bit integers.
Checksum	Unsigned 16-bit little-endian integer.

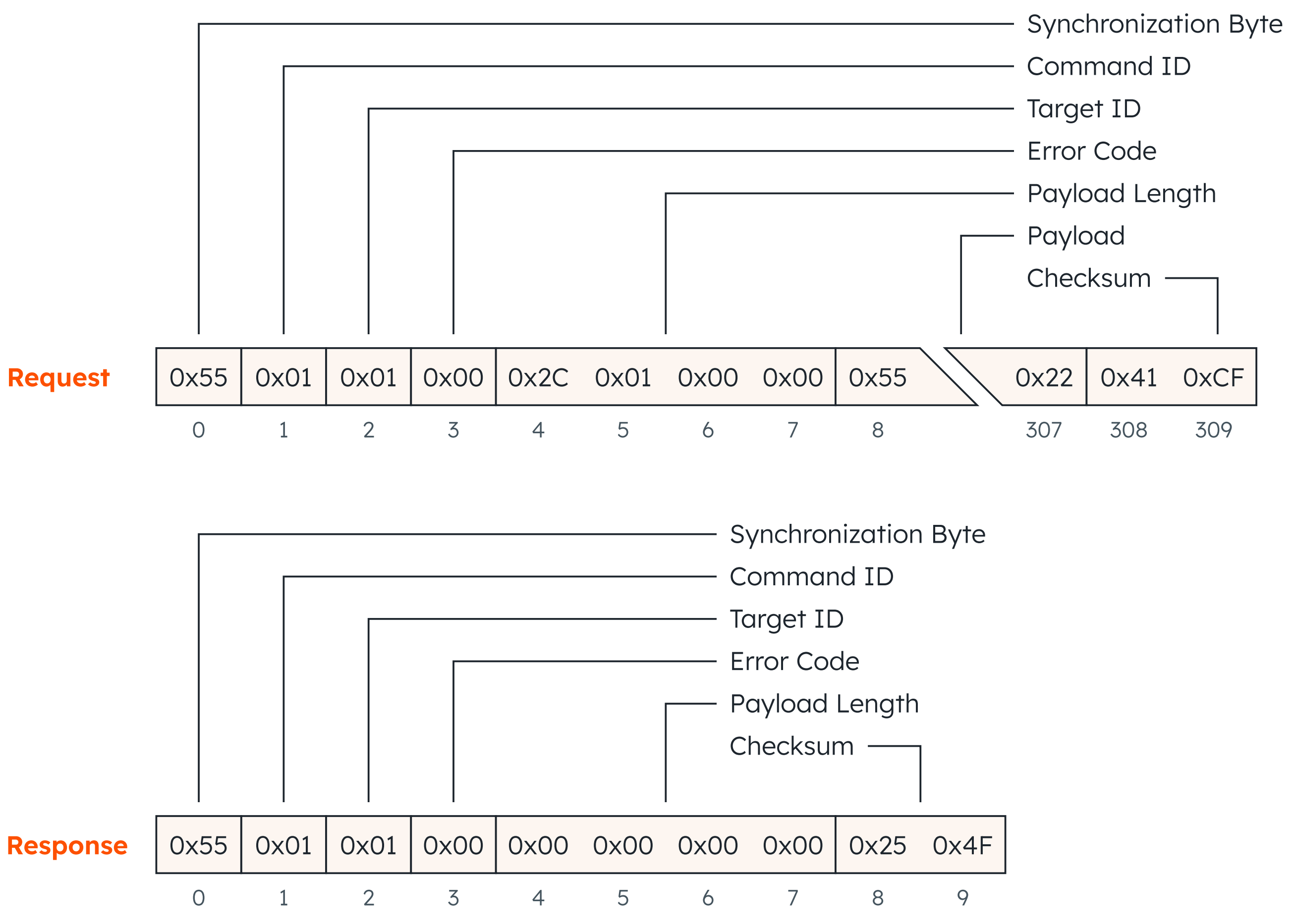

Figure 1: Synchronous communication flow via USB with Command ID 0x01, Target ID 0x01, 300 bytes of Request payload, and 0 bytes of Response payload

Bluetooth Low Energy Interface

To expose the RPC Protocol via Bluetooth Low Energy, the CSS Core Firmware implements the GAP Peripheral role and a GATT Server. The Host Device must implement the GAP Central role and a GATT Client. It must support the LE Data Packet Length Extension and a LL data payload length of 162 bytes.

GAP

For identification of CSS Core Firmware-based devices, the BLE advertisement data can be inspected. The advertisement data of CSS Core Firmware-based devices contains Manufacturer-Specific Data in both bootloader mode and non-bootloader mode. It contains the USB Product ID and an additional 16-bit Product Sub-ID value. The lower 15 bits of the Product Sub-ID have a product-specific meaning. The most significant bit (Bit 15) of the Product Sub-ID is set when the device is in bootloader mode and is unset otherwise. When in bootloader mode, the values of the USB Product ID and the lower 15 bits of the Product Sub-ID are identical to the values used in the non-bootloader mode. (Note: When the device is in bootloader mode, because there is no firmware installed, the USB Product ID and the Product Sub-ID are both set to 0xFFFF.)

Table 6: BLE Advertisement Manufacturer-Specific Data

Byte Index	Description
1:0	Bluetooth SIG Company Identifier assigned to ams-OSRAM AG (0x0B21) as an unsigned little-endian integer.
2	Format ID, should always be 0x01.
4:3	USB Product ID as an unsigned little-endian integer.
6:5	Product Sub-ID as an unsigned little-endian integer.
7	Bits 7:0 of the serial number.
8	Bits 15:8 of the serial number.
9	Bits 23:16 of the serial number.
10	Bits 31:24 of the serial number.
11	Bits 39:32 of the serial number.
12	Bits 47:40 of the serial number.

While the serial number is encoded as binary data in the BLE advertisement data, other methods to access the serial number (Device Information Service, RPC command, USB device descriptor) typically provide it as a UTF-8 string. To convert the binary data to a string, the bytes 7 to 12 of the BLE advertisement data must be interpreted as an unsigned 48-bit little-endian integer value that is output as a zero-padded, uppercase hexadecimal string. No prefix (such as “0x”) or suffix must be added to the string. In Python, assuming that the BLE advertisement data is stored in a `bytes` object in a variable named `data`, the string can be generated using `format(int.from_bytes(data[7:13], byteorder="little"), "012X")`.

The CSS Core Firmware performs undirected advertising indefinitely whilst there is no USB connection. The advertising interval is set to 30 milliseconds for the first 30 seconds of advertising. Afterward, the advertising interval is changed to 250 milliseconds.

The CSS Core Firmware does not support pairing or bonding. For all connections, GAP Security Mode 1 Level 1 (No Security) is used.

When a BLE connection is established, the CSS Core Firmware requests the following the connection parameters, which must be supported by the Host Device:

- Minimum acceptable connection interval: 15 milliseconds
- Maximum acceptable connection interval: 30 milliseconds
- Peripheral latency (also known as slave latency): 0 connection intervals
- Supervisory timeout: 4 seconds

The Host Device must support an ATT MTU size of 158 bytes.

Coexistence with USB

When a Host Device connects via USB (i.e. asserts the RTS control signal), the active BLE connection is terminated, and advertising is suspended for the duration of the USB connection.

RPC GATT Service

The GATT Server provides the proprietary RPC Service with the proprietary RPC Characteristic.

Table 7: RPC Service and Characteristic

Name	UUID	Characteristic Properties
RPC Service	`fe8a042a-c4e3-11ea-87d0-0242ac130003`	n/a
RPC Characteristic	`fe8a0438-c4e3-11ea-87d0-0242ac130003`	Write Without Response, Notify

The maximum data size that can be written to the Characteristic Value is limited to 244 bytes. Since the size of an RPC message may exceed this limit, a fragmentation layer is introduced.

The Host Device fragments a Request message into one or more RPC BLE Fragments and writes the fragments sequentially without response to the Characteristic Value of the RPC Characteristic. The CSS Core Firmware Device fragments a Response message into one or more RPC BLE Fragments and provides them sequentially using unacknowledged notifications on the Characteristic Value of the RPC Characteristic. **The Host Device must enable notifications via the Client Characteristic Configuration Descriptor (CCCD) of the RPC Characteristic to receive Response messages.**

RPC BLE Fragmentation

The first byte (Byte 0) of each RPC BLE Fragment contains a Fragment Header. This header controls the structure of the RPC BLE Fragment.

Table 8: Bit Fields of the Fragment Header

Name	Bits	Description
Command Header Present	7	A Command Header is present in the fragment when this bit is set. The presence of a Command Header marks the start of an RPC Message.
Target ID Present	6	The Target ID field is present in the Command Header when this bit is set. When this bit is not set, the default value for the Target ID field is used. This field must be ignored when the Command Header Present bit is not set.
Error Code Present	5	The Error Code field is present in the Command Header when this bit is set. When this bit is not set, the default value for the Error Code field is used. This field must be ignored when the Command Header Present bit is not set.
Extended Payload Length	4	The size of the Payload Length field in the Command Header is 4 bytes when this bit is set. When this bit is not set, the size of the Payload Length field is 1 byte. This field must be ignored when the Command Header Present bit is not set.
Reserved	3	Reserved for future use. Always set it to zero.
Incrementing Pattern	2:0	A 3-bit counter used for fragment loss detection. The counter value is reset to 0x00 for the first fragment of an RPC Message. For each subsequent fragment, the counter value is incremented by one. When the counter reaches the value 0x07, it overflows to the value 0x00 when it is next incremented.

The presence of the Command Header marks the beginning of an RPC Message. If the Command Header is present, it always follows immediately after the Fragment Header.

Table 9: Fields of the Command Header

Name	Data Type	Default Value
Command ID	Unsigned 8-bit integer.	n/a
Target ID	Unsigned 8-bit integer, only present if the Target ID Present bit is set in the Fragment Header.	0x00
Error Code	Unsigned 8-bit integer, only present if the Error Code Present bit is set in the Fragment Header.	0 (see Table 3)

Table 9 (Continued): Fields of the Command Header

Name	Data Type	Default Value
Payload Size	Unsigned 8-bit integer or unsigned 32-bit little-endian integer, depending on the value of the Extended Payload Length bit in the Fragment Header.	n/a

The remaining size of the RPC BLE Fragment is used to store the Payload data of the RPC Message. Payload data, starting with the first byte of the Payload, is appended to the RPC BLE Fragment until the maximum fragment size has been reached or all Payload data has been appended.

Once the maximum fragment size has been reached and there is pending Payload data, a new RPC BLE Fragment is created. The new RPC BLE Fragment does not contain a Command Header, and Payload data is appended immediately after the Fragment Header until the maximum fragment size has been reached or all Payload data has been appended. The first Payload data byte to append is the byte following the last byte that has been appended to the previous RPC BLE Fragment. This process is repeated until all Payload data has been appended.

When receiving a Response message, the Incrementing Pattern of each RPC BLE Fragment must be inspected by the Host Device to ensure that no RPC BLE Fragment has been lost. If a fragment loss is detected, all the fragments must be discarded. There is no mechanism to request the re-transmission of a Response message. Note that the Incrementing Pattern cannot be used to detect the loss of the last fragment of a message. The loss of the last fragment of a message can either be detected when a subsequent message is received or by implementing a timeout mechanism.

When the CSS Core Firmware detects a loss of fragments or receives malformed fragments, the received fragments are silently discarded and no Response message is sent. The CSS Core Firmware detects the loss of the last fragment of a message only when a subsequent message is received as it does not implement a timeout mechanism.

It is guaranteed that the CSS Core Firmware does not start transmission of a new Response message while the fragments of a previous Response message are still transmitted.

The maximum size of a RPC BLE Fragment (including Fragment Header and Command Header) is 155 bytes.

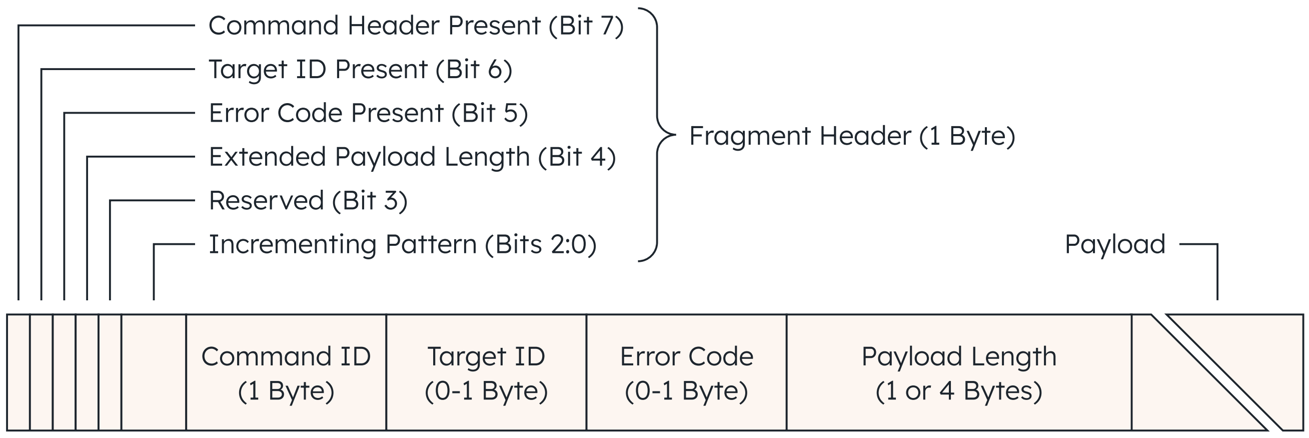
Command Header Present (Bit 7)
Target ID Present (Bit 6)
Error Code Present (Bit 5)
Extended Payload Length (Bit 4)
Reserved (Bit 3)
Incrementing Pattern (Bits 2:0)
Fragment Header (1 Byte)
Payload
Command ID (1 Byte)
Target ID (0-1 Byte)
Error Code (0-1 Byte)
Payload Length (1 or 4 Bytes)

Figure 2: Structure of an RPC BLE Fragment

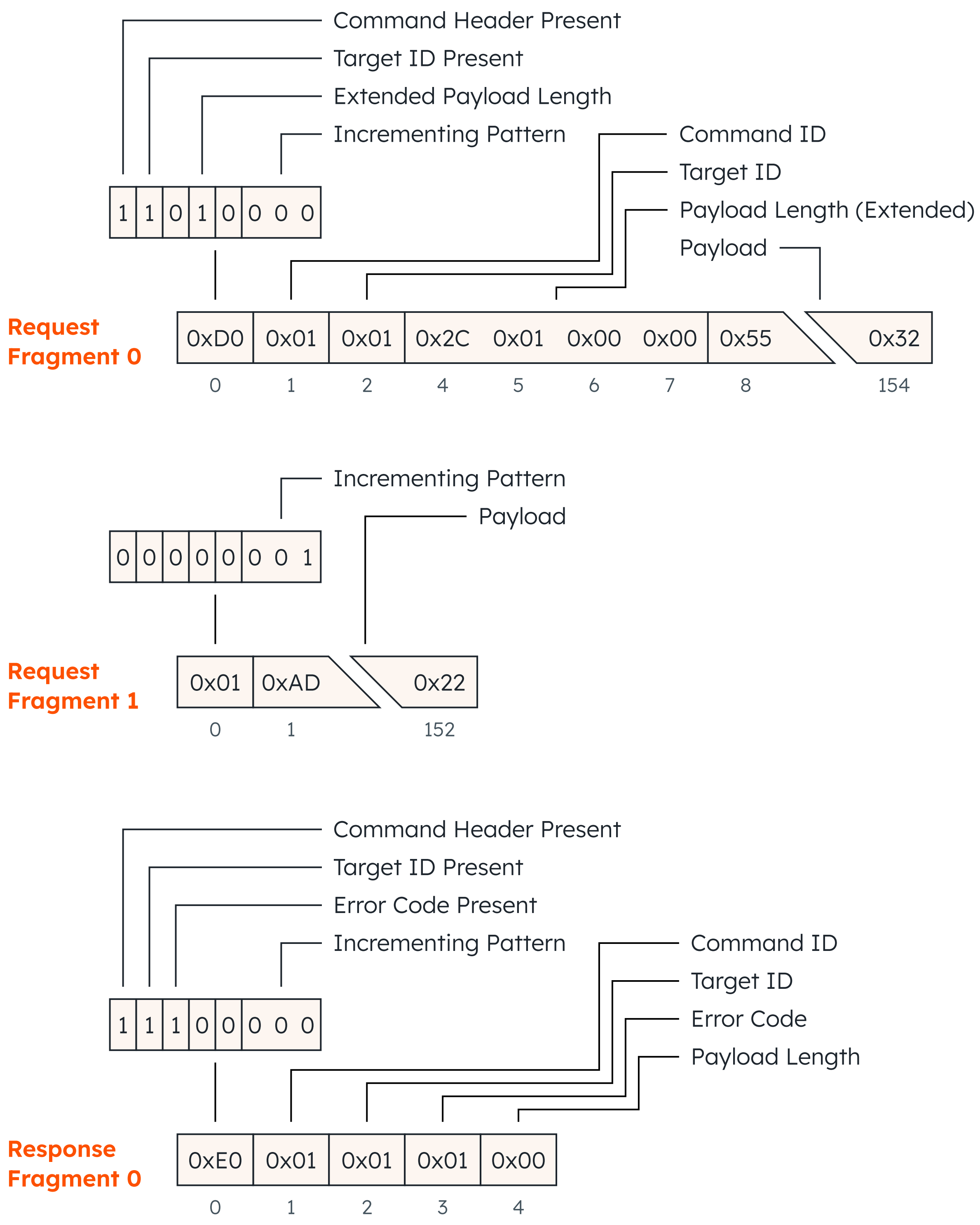

Figure 3: Synchronous communication flow via BLE with Command ID 0x01, Target ID 0x01, 300 bytes of Request payload, Error Code 0x01 returned in Response, and 0 bytes of Response payload

Other GATT Services

In addition to the RPC Service, the CSS Core Firmware in non-bootloader mode also implements the Device Information Service and the Battery Service as specified by Bluetooth SIG.

Data Types and Common Definitions

Throughout this document, the following data types and definitions are used. Wherever structures are described using a bulleted list, the listed fields are meant to be placed contiguously in the listed order with no implicit padding bytes inserted between fields.

Table 10: Data Types

Type	Description
string	UTF-8-encoded string, not NULL-terminated.
uint8	Unsigned 8-bit integer.
uint16	Unsigned 16-bit little endian integer.
uint24	Unsigned 24-bit little endian integer.
uint32	Unsigned 32-bit little endian integer.
int16	Signed 16-bit little endian integer (two's complement).
int32	Signed 32-bit little endian integer (two's complement).
float	Single-precision little endian IEEE 754 floating point number.
Optional *x*	Field of type *x* which may or may not be present.
Array of *x*	Contiguous sequence of fields of type *x* with no padding bytes inserted between items.

Table 10 (Continued): Data Types

Type	Description
AGC Configuration	Structure with the following fields: • **mode** (uint8, 1 Byte): Selects the AGC algorithm mode (see Table 11). • **led_control_mode** (uint8, 1 Byte): Selects the LED amplitude control mode (see Table 12). • **channel** (uint8, 1 Byte): Selects the PPG channel that is controlled (see Table 13). • **led_current_min** (uint8, 1 Byte): The lower bound of the LED current range as a register value. • **led_current_max** (uint8, 1 Byte): The upper bound of the LED current range as a register value. • **rel_amplitude_min_x100** (uint8, 1 Byte): The lower bound of the targeted PPG signal amplitude, relative to the size of the target PGG signal range. The unit of this field is percent. The minimum valid value is 0, and the maximum valid value is 100. • **rel_amplitude_max_x100** (uint8, 1 Byte): The upper bound of the targeted PPG signal amplitude, relative to the size of the target PGG signal range. The unit of this field is percent. This value must not be less than rel_amplitude_min_x100. The maximum valid value is 100. • **rel_amplitude_motion_x100** (uint8, 1 Byte): The minimum PPG signal amplitude at which the PPG signal is considered a motion artifact. The threshold is relative to the size of the target PGG signal range. The unit of this field is percent. This value must not be less than rel_amplitude_max_x100. The maximum valid value is 100. • **num_led_steps** (uint8, 1 Byte): The number of steps the LED current range is partitioned in. When the AGC algorithm determines that the LED current needs to be increased or decreased and the bounds of the LED current range are not yet reached, the LED current is adjusted by one step. If led_control_mode is set to 1 (see Table 12), this value must not be zero and must not be greater than the size of the LED current range. • **reserved** (Array of uint8, 3 Bytes): Padding bytes. • **threshold_min** (int32, 1 Byte): The lower bound of the target PPG signal range. The unit of this field is ADC counts. • **threshold_max** (int32, 1 Byte): The upper bound of the target PPG signal range. The unit of this field is ADC counts. This value must be greater than threshold_min. **Note: When using AGC mode 1 (see Table 11), all fields other than mode and channel shall be set to zero.**
Accelerometer Sample	Structure with the following fields: • **x** (int16, 2 Bytes): X-axis accelerometer value. • **y** (int16, 2 Byte): Y-axis accelerometer value. • **z** (int16, 2 Bytes): Z-axis accelerometer value.

Table 10 (Continued): Data Types

Type	Description
AGC Status	Structure with the following fields: • **pd_offset_change** (uint8, 1 Byte): Change information for the photodiode offset current (see Table 14). • **pd_offset_current** (uint8, 1 Byte): Current value of the photodiode offset current as a register value. • **led_current_change** (uint8, 1 Byte): Change information for the LED current (see Table 14). • **led_current_current** (uint8, 1 Byte): Current value of the LED current as a register value.
Status Event	Structure with the following fields: • **status_seq** (uint8, 1 Byte): Content of register STATUS_SEQ if the corresponding interrupt occurred, 0 otherwise. • **status_led** (uint8, 1 Byte): Content of register STATUS_LED if the corresponding interrupt occurred, 0 otherwise. • **status_asata** (uint8, 1 Byte): Content of register STATUS_ASATA if the corresponding interrupt occurred, 0 otherwise. • **status_asatb** (uint8, 1 Byte): Content of register STATUS_ASATB if the corresponding interrupt occurred, 0 otherwise. • **status_vcsel** (uint8, 1 Byte): Content of register STATUS_VCSEL if the corresponding interrupt occurred, 0 otherwise. • **status_vcsel_vss** (uint8, 1 Byte): Content of register STATUS_VCSEL_VSS if the corresponding interrupt occurred, 0 otherwise. • **status_vcsel_vdd** (uint8, 1 Byte): Content of register STATUS_VCSEL_VDD if the corresponding interrupt occurred, 0 otherwise. • **status_leadoff** (uint8, 1 Byte): Content of register STATUS_LEADOFF if the corresponding interrupt occurred, 0 otherwise. • **status_iir** (uint8, 1 Byte): 1 if IIR interrupt occurred, 0 otherwise.
Sample with PD Offset	• **adc_value** (uint24, 3 Bytes): ADC counts. • **pd_offset** (uint8, 1 Byte): PD offset current register value.

Table 11: Operating Modes of the AGC Algorithm

Value	Description
0	Default mode.
1	AGC algorithm is disabled but returns at least one status package for the selected channel.

Table 12: Amplitude Control Modes of the AGC Algorithm

Value	Description
0	No amplitude control enabled.
1	Internal amplitude control enabled.
2	Amplitude is controlled by an external algorithm.

Table 13: Sub-Sample ID

Value	Description
0	Sub-sample is disabled.
1	Sub-sample ID of sub-sample 1 of PPG modulator 1.
2	Sub-sample ID of sub-sample 2 of PPG modulator 1.
3	Sub-sample ID of sub-sample 3 of PPG modulator 1.
4	Sub-sample ID of sub-sample 4 of PPG modulator 1.
5	Sub-sample ID of sub-sample 5 of PPG modulator 1.
6	Sub-sample ID of sub-sample 6 of PPG modulator 1.
7	Sub-sample ID of sub-sample 7 of PPG modulator 1.
8	Sub-sample ID of sub-sample 8 of PPG modulator 1.
9	Sub-sample ID of sub-sample 1 of PPG modulator 2.
10	Sub-sample ID of sub-sample 2 of PPG modulator 2.
11	Sub-sample ID of sub-sample 3 of PPG modulator 2.
12	Sub-sample ID of sub-sample 4 of PPG modulator 2.
13	Sub-sample ID of sub-sample 5 of PPG modulator 2.
14	Sub-sample ID of sub-sample 6 of PPG modulator 2.
15	Sub-sample ID of sub-sample 7 of PPG modulator 2.
16	Sub-sample ID of sub-sample 8 of PPG modulator 2.
17	Sub-sample ID of sub-sample 1 of sequence 1 of the ECG modulator.
18	Sub-sample ID of sub-sample 2 of sequence 1 of the ECG modulator.
19	Sub-sample ID of sub-sample 1 of sequence 2 of the ECG modulator.

Table 14: AGC Current Change Information

Value	Description
0	Current not changed.

Table 14 (Continued): AGC Current Change Information

Value	Description
1	Current increased.
2	Current decreased.
3	Current needs to be decreased, but is at the lower limit.
4	Current needs to be increased, but is at the upper limit.
5	Current is not controlled as amplitude control is disabled.
6	Current is undefined. For example the channel has no connected LED.

Register Groups

The AS7058 Chip Library groups functionally related registers of the AS7058 AFE into register groups. See Table 15 for a list of available register groups.

Table 15: Definition of the Register Group IDs

Value	Description
0	ID of the `power` register group. See Table 16 for the contents of the group.
1	ID of the `control` register group. See Table 17 for the contents of the group.
2	ID of the `led` register group. See Table 18 for the contents of the group.
3	ID of the pd register group. See Table 19 for the contents of the group.
4	ID of the `ios` register group. See Table 20 for the contents of the group.
5	ID of the `ppg` register group. See Table 21 for the contents of the group.
6	ID of the `ecg` register group. See Table 22 for the contents of the group.
7	ID of the `sinc` register group. See Table 23 for the contents of the group.
8	ID of the `iir` register group. See Table 24 for the contents of the group.
9	ID of the `seq` register group. See Table 25 for the contents of the group.
10	ID of the `post` register group. See Table 26 for the contents of the group.
11	ID of the `fifo` register group. See Table 27 for the contents of the group.

Table 16: Contents of the Power Register Group

Field	Type	Size	Description
pwr_on	uint8	1	Content of register PWR_ON.
pwr_iso	uint8	1	Content of register PWR_ISO.

Table 16 (Continued): Contents of the Power Register Group

Field	Type	Size	Description
clk_cfg	uint8	1	Content of register CLK_CFG.
ref_cfg1	uint8	1	Content of register REF_CFG1.
ref_cfg2	uint8	1	Content of register REF_CFG2.
ref_cfg3	uint8	1	Content of register REF_CFG3.
standby_on1	uint8	1	Content of register STANDBY_ON1.
standby_on2	uint8	1	Content of register STANDBY_ON2.
standby_en1	uint8	1	Content of register STANDBY_EN1.
standby_en2	uint8	1	Content of register STANDBY_EN2.
standby_en3	uint8	1	Content of register STANDBY_EN3.
standby_en4	uint8	1	Content of register STANDBY_EN4.
standby_en5	uint8	1	Content of register STANDBY_EN5.
standby_en6	uint8	1	Content of register STANDBY_EN6.
standby_en7	uint8	1	Content of register STANDBY_EN7.
standby_en8	uint8	1	Content of register STANDBY_EN8.
standby_en9	uint8	1	Content of register STANDBY_EN9.
standby_en10	uint8	1	Content of register STANDBY_EN10.
standby_en11	uint8	1	Content of register STANDBY_EN11.
standby_en12	uint8	1	Content of register STANDBY_EN12.
standby_en13	uint8	1	Content of register STANDBY_EN13.
standby_en14	uint8	1	Content of register STANDBY_EN14.

Table 17: Contents of the Control Register Group

Field	Type	Size	Description
i2c_mode	uint8	1	Content of register I2C_MODE.
int_cfg	uint8	1	Content of register INT_CFG.
if_cfg	uint8	1	Content of register IF_CFG.
gpio_cfg1	uint8	1	Content of register GPIO_CFG1.
gpio_cfg2	uint8	1	Content of register GPIO_CFG2.
io_cfg	uint8	1	Content of register IO_CFG.

Table 18: Contents of the LED Register Group

Field	Type	Size	Description
vcsel_password	uint8	1	Content of register VCSEL_PASSWORD.
vcsel_cfg	uint8	1	Content of register VCSEL_CFG.
vcsel_mode	uint8	1	Content of register VCSEL_MODE.
led_cfg	uint8	1	Content of register LED_CFG.
led_drv1	uint8	1	Content of register LED_DRV1.
led_drv2	uint8	1	Content of register LED_DRV2.
led1_ictrl	uint8	1	Content of register LED1_ICTRL.
led2_ictrl	uint8	1	Content of register LED2_ICTRL.
led3_ictrl	uint8	1	Content of register LED3_ICTRL.
led4_ictrl	uint8	1	Content of register LED4_ICTRL.
led5_ictrl	uint8	1	Content of register LED5_ICTRL.
led6_ictrl	uint8	1	Content of register LED6_ICTRL.
led7_ictrl	uint8	1	Content of register LED7_ICTRL.
led8_ictrl	uint8	1	Content of register LED8_ICTRL.
led_irng1	uint8	1	Content of register LED_IRNG1.
led_irng2	uint8	1	Content of register LED_IRNG2.
led_sub1	uint8	1	Content of register LED_SUB1.
led_sub2	uint8	1	Content of register LED_SUB2.
led_sub3	uint8	1	Content of register LED_SUB3.
led_sub4	uint8	1	Content of register LED_SUB4.
led_sub5	uint8	1	Content of register LED_SUB5.
led_sub6	uint8	1	Content of register LED_SUB6.
led_sub7	uint8	1	Content of register LED_SUB7.
led_sub8	uint8	1	Content of register LED_SUB8.
lowvds_wait	uint8	1	Content of register LOWVDS_WAIT.

Table 19: Contents of the Photodiodes Register Group

Field	Type	Size	Description
pdsel_cfg	uint8	1	Content of register PDSEL_CFG.
ppg1_pdsel1	uint8	1	Content of register PPG1_PDSEL1.

Table 19 (Continued): Contents of the Photodiodes Register Group

Field	Type	Size	Description
ppg1_pdsel2	uint8	1	Content of register PPG1_PDSEL2.
ppg1_pdsel3	uint8	1	Content of register PPG1_PDSEL3.
ppg1_pdsel4	uint8	1	Content of register PPG1_PDSEL4.
ppg1_pdsel5	uint8	1	Content of register PPG1_PDSEL5.
ppg1_pdsel6	uint8	1	Content of register PPG1_PDSEL6.
ppg1_pdsel7	uint8	1	Content of register PPG1_PDSEL7.
ppg1_pdsel8	uint8	1	Content of register PPG1_PDSEL8.
ppg2_pdsel1	uint8	1	Content of register PPG2_PDSEL1.
ppg2_pdsel2	uint8	1	Content of register PPG2_PDSEL2.
ppg2_pdsel3	uint8	1	Content of register PPG2_PDSEL3.
ppg2_pdsel4	uint8	1	Content of register PPG2_PDSEL4.
ppg2_pdsel5	uint8	1	Content of register PPG2_PDSEL5.
ppg2_pdsel6	uint8	1	Content of register PPG2_PDSEL6.
ppg2_pdsel7	uint8	1	Content of register PPG2_PDSEL7.
ppg2_pdsel8	uint8	1	Content of register PPG2_PDSEL8.
ppg2_afesel1	uint8	1	Content of register PPG2_AFESEL1.
ppg2_afesel2	uint8	1	Content of register PPG2_AFESEL2.
ppg2_afesel3	uint8	1	Content of register PPG2_AFESEL3.
ppg2_afesel4	uint8	1	Content of register PPG2_AFESEL4.
ppg2_afeen	uint8	1	Content of register PPG2_AFEEN.

Table 20: Contents of the PD Offset Control Register Group

Field	Type	Size	Description
ios_ppg1_sub1	uint8	1	Content of register IOS_PPG1_SUB1.
ios_ppg1_sub2	uint8	1	Content of register IOS_PPG1_SUB2.
ios_ppg1_sub3	uint8	1	Content of register IOS_PPG1_SUB3.
ios_ppg1_sub4	uint8	1	Content of register IOS_PPG1_SUB4.
ios_ppg1_sub5	uint8	1	Content of register IOS_PPG1_SUB5.
ios_ppg1_sub6	uint8	1	Content of register IOS_PPG1_SUB6.
ios_ppg1_sub7	uint8	1	Content of register IOS_PPG1_SUB7.

Table 20 (Continued): Contents of the PD Offset Control Register Group

Field	Type	Size	Description
ios_ppg1_sub8	uint8	1	Content of register IOS_PPG1_SUB8.
ios_ppg2_sub1	uint8	1	Content of register IOS_PPG2_SUB1.
ios_ppg2_sub2	uint8	1	Content of register IOS_PPG2_SUB2.
ios_ppg2_sub3	uint8	1	Content of register IOS_PPG2_SUB3.
ios_ppg2_sub4	uint8	1	Content of register IOS_PPG2_SUB4.
ios_ppg2_sub5	uint8	1	Content of register IOS_PPG2_SUB5.
ios_ppg2_sub6	uint8	1	Content of register IOS_PPG2_SUB6.
ios_ppg2_sub7	uint8	1	Content of register IOS_PPG2_SUB7.
ios_ppg2_sub8	uint8	1	Content of register IOS_PPG2_SUB8.
ios_ledoff	uint8	1	Content of register IOS_LEDOFF.
ios_cfg	uint8	1	Content of register IOS_CFG.
aoc_sar_thres	uint8	1	Content of register AOC_SAR_THRES.
aoc_sar_range	uint8	1	Content of register AOC_SAR_RANGE.
aoc_sar_ppg1	uint8	1	Content of register AOC_SAR_PPG1.
aoc_sar_ppg2	uint8	1	Content of register AOC_SAR_PPG2.

Table 21: Contents of the PPG Register Group

Field	Type	Size	Description
ppgmod_cfg1	uint8	1	Content of register PPGMOD_CFG1.
ppgmod_cfg2	uint8	1	Content of register PPGMOD_CFG2.
ppgmod_cfg3	uint8	1	Content of register PPGMOD_CFG3.
ppgmod1_cfg1	uint8	1	Content of register PPGMOD1_CFG1.
ppgmod1_cfg2	uint8	1	Content of register PPGMOD1_CFG2.
ppgmod1_cfg3	uint8	1	Content of register PPGMOD1_CFG3.
ppgmod2_cfg1	uint8	1	Content of register PPGMOD2_CFG1.
ppgmod2_cfg2	uint8	1	Content of register PPGMOD2_CFG2.
ppgmod2_cfg3	uint8	1	Content of register PPGMOD2_CFG3.

Table 22: Contents of the ECG Register Group

Field	Type	Size	Description
bioz_cfg	uint8	1	Content of register BIOZ_CFG.
bioz_excit	uint8	1	Content of register BIOZ_EXCIT.
bioz_mixer	uint8	1	Content of register BIOZ_MIXER.
bioz_select	uint8	1	Content of register BIOZ_SELECT.
bioz_gain	uint8	1	Content of register BIOZ_GAIN.
ecgmod_cfg1	uint8	1	Content of register ECGMOD_CFG1.
ecgmod_cfg2	uint8	1	Content of register ECGMOD_CFG2.
ecgimux_cfg1	uint8	1	Content of register ECGIMUX_CFG1.
ecgimux_cfg2	uint8	1	Content of register ECGIMUX_CFG2.
ecgimux_cfg3	uint8	1	Content of register ECGIMUX_CFG3.
ecgamp_cfg1	uint8	1	Content of register ECGAMP_CFG1.
ecgamp_cfg2	uint8	1	Content of register ECGAMP_CFG2.
ecgamp_cfg3	uint8	1	Content of register ECGAMP_CFG3.
ecgamp_cfg4	uint8	1	Content of register ECGAMP_CFG4.
ecgamp_cfg5	uint8	1	Content of register ECGAMP_CFG5.
ecgamp_cfg6	uint8	1	Content of register ECGAMP_CFG6.
ecgamp_cfg7	uint8	1	Content of register ECGAMP_CFG7.
ecg_bioz	uint8	1	Content of register ECG_BIOZ.
leadoff_cfg	uint8	1	Content of register LEADOFF_CFG
leadoff_thresl	uint8	1	Content of register LEADOFF_THRESL.
leadoff_thresh	uint8	1	Content of register LEADOFF_THRESH.

Table 23: Contents of the Sinc Filter Register Group

Field	Type	Size	Description
ppg_sinc_cfga	uint8	1	Content of register PPG_SINC_CFGA.
ppg_sinc_cfgb	uint8	1	Content of register PPG_SINC_CFGB.
ppg_sinc_cfgc	uint8	1	Content of register PPG_SINC_CFGC.
ppg_sinc_cfgd	uint8	1	Content of register PPG_SINC_CFGD.
ecg1_sinc_cfga	uint8	1	Content of register ECG1_SINC_CFGA.
ecg1_sinc_cfgb	uint8	1	Content of register ECG1_SINC_CFGB.

Table 23 (Continued): Contents of the Sinc Filter Register Group

Field	Type	Size	Description
ecg1_sinc_cfgc	uint8	1	Content of register ECG1_SINC_CFGC.
ecg2_sinc_cfga	uint8	1	Content of register ECG2_SINC_CFGA.
ecg2_sinc_cfgb	uint8	1	Content of register ECG2_SINC_CFGB.
ecg2_sinc_cfgc	uint8	1	Content of register ECG2_SINC_CFGC.
ecg_sinc_cfg	uint8	1	Content of register ECG_SINC_CFG.

Table 24: Contents of the IIR Filter Register Group

Field	Type	Size	Description
iir_cfg	uint8	1	Content of register IIR_CFG.
reserved	uint8	1	Unused padding byte.
iir_coeff_data_sos	Multi-dimensional array of int16 with dimensions 12-by-5	120	Coefficients of the cascaded IIR second-order sections. Up to 12 sections are supported. Each section has five coefficients.

Table 25: Contents of the Sequencer Register Group

Field	Type	Size	Description
irq_enable	uint8	1	Content of register IRQ_ENABLE.
ppg_sub_wait	uint8	1	Content of register PPG_SUB_WAIT.
ppg_sar_wait	uint8	1	Content of register PPG_SAR_WAIT.
ppg_led_init	uint8	1	Content of register PPG_LED_INIT.
ppg_freql	uint8	1	Content of register PPG_FREQL.
ppg_freqh	uint8	1	Content of register PPG_FREQH.
ppg1_sub_en	uint8	1	Content of register PPG1_SUB_EN.
ppg2_sub_en	uint8	1	Content of register PPG2_SUB_EN.
ppg_mode_1	uint8	1	Content of register PPG_MODE1.
ppg_mode_2	uint8	1	Content of register PPG_MODE2.
ppg_mode_3	uint8	1	Content of register PPG_MODE3.
ppg_mode_4	uint8	1	Content of register PPG_MODE4.
ppg_mode_5	uint8	1	Content of register PPG_MODE5.
ppg_mode_6	uint8	1	Content of register PPG_MODE6.

Table 25 (Continued): Contents of the Sequencer Register Group

Field	Type	Size	Description
ppg_mode_7	uint8	1	Content of register PPG_MODE7.
ppg_mode_8	uint8	1	Content of register PPG_MODE8.
ppg_cfg	uint8	1	Content of register PPG_CFG.
ecg_freql	uint8	1	Content of register ECG_FREQL.
ecg_freqh	uint8	1	Content of register ECG_FREQH.
ecg1_freqdivl	uint8	1	Content of register ECG1_FREQDIVL.
ecg1_freqdivh	uint8	1	Content of register ECG1_FREQDIVH.
ecg2_freqdivl	uint8	1	Content of register ECG2_FREQDIVL.
ecg2_freqdivh	uint8	1	Content of register ECG2_FREQDIVH.
ecg_subs	uint8	1	Content of register ECG_SUBS.
leadoff_initl	uint8	1	Content of register LEADOFF_INITL.
leadoff_inith	uint8	1	Content of register LEADOFF_INITH.
ecg_initl	uint8	1	Content of register ECG_INITL.
ecg_inith	uint8	1	Content of register ECG_INITH.
sample_num	uint8	1	Content of register SAMPLE_NUM.

Table 26: Contents of the Post-Processing Register Group

Field	Type	Size	Description
pp_cfg	uint8	1	Content of register PP_CFG.
ppg1_pp1	uint8	1	Content of register PPG1_PP1.
ppg1_pp2	uint8	1	Content of register PPG1_PP2.
ppg2_pp1	uint8	1	Content of register PPG2_PP1.
ppg2_pp2	uint8	1	Content of register PPG2_PP2.

Table 27: Contents of the FIFO Register Group

Field	Type	Size	Description
fifo_threshold	uint8	1	Content of register FIFO_THRESHOLD.
fifo_ctrl	uint8	1	Content of register FIFO_CTRL.

Measurement Modes

The AS7058 Chip Library supports a number of measurement modes for different use cases. See Table 28 for a list of available measurement modes and their identifiers.

Table 28: Measurement Modes

Value	Description
0	Default measurement mode for regular FIFO data acquisition.
1	Special measurement mode for BioZ.
2	Special measurement mode to obtain impedance reference values for EDA.
3	Special measurement mode to perform PD offset calibration.

Regular

The regular measurement mode is suited for continuous standard PPG and ECG data acquisition. The data stream received from the AS7058 AFE is internally provided to the AS7058 Application Manager. The AS7058 Application Manager output is provided via command CMD_ID_VSC_AM_APP_OUTPUT.

Before starting a measurement, a suitable configuration must be written to the registers of the AS7058 AFE. The regular measurement mode does not modify register values during measurement.

BioZ

In the BioZ special measurement mode, the AS7058 Chip Library performs I/Q measurements of different impedances while the AS7058 AFE operates in a special BioZ measurement mode. The BioZ measurement mode is used for both system calibration and subject measurements. Note that the measurements are performed continuously, i.e. output is generated continuously until the measurement is explicitly stopped.

Before starting a measurement, a suitable configuration must be written to the registers of the AS7058 AFE. During measurement, the BioZ measurement mode temporarily modifies some register values.

Configuration

The BioZ measurement mode is configured via command CMD_ID_VSC_CL_CONFIG_SPECIAL_MEASUREMENT and Target ID value 1 (see Table 28). The payload of the command has the following fields with a total size of 4 bytes:

- **num_averages** (uint16, 2 Bytes): Number of samples to average for each impedance measurement.

- **num_dropped_first_samples** (uint16, 2 Bytes): Number of samples to drop per impedance measurement before averaging. Because of settling time, BioZ results can be optimized by dropping the first samples.

Output

The output of the BioZ measurement mode is internally provided to the AS7058 Application Manager, which passes the output to the BioZ application, provided that the application is enabled. When using the BioZ measurement mode for subject measurements, only the BioZ application output provided via the AS7058 Application Manager is usually of interest.

When performing a BioZ measurement for system calibration purposes, the output of the BioZ measurement mode is intended to be passed to the BioZ application as part of its configuration. The output is provided via command CMD_ID_VSC_CL_SPECIAL_MEASUREMENT_RESULT using Target ID value 1 (see Table 28). It has the following payload fields with a total size of 64 bytes:

- **ref_resistor** (uint32, 4 Bytes): Resistance of the reference resistor in ohm. This value is calculated from OTP data.
- **ref_temperature_adc** (uint32, 4 Bytes): ADC counts of the temperature when the resistance of the reference resistor was measured. This value is calculated from OTP data.
- **temperature_adc** (uint32, 4 Bytes): ADC counts of the temperature during the BioZ measurements. It is set to 0xFFFFFFFF if temperature measurement is disabled.
- **slope** (int32, 4 Bytes): Slope of the temperature scaled by 1000000.
- **short_impedance_in_phase** (uint32, 4 Bytes): In-phase component of the short circuit impedance measurement.
- **short_impedance_quadrature** (uint32, 4 Bytes): Quadrature component of the short circuit impedance measurement.
- **resistor_impedance_in_phase** (uint32, 4 Bytes): In-phase component of the internal 2 kiloohms reference resistor impedance measurement.
- **resistor_impedance_quadrature** (uint32, 4 Bytes): Quadrature component of the internal 2 kiloohms reference resistor impedance measurement.
- **body_impedance_in_phase** (uint32, 4 Bytes): In-phase component of the body impedance measurement.
- **body_impedance_quadrature** (uint32, 4 Bytes): Quadrature component of the body impedance measurement.
- **wrist_impedance_in_phase** (uint32, 4 Bytes): In-phase component of the wrist impedance measurement.
- **wrist_impedance_quadrature** (uint32, 4 Bytes): Quadrature component of the wrist impedance measurement.
- **finger_impedance_in_phase** (uint32, 4 Bytes): In-phase component of the finger impedance measurement.
- **finger_impedance_quadrature** (uint32, 4 Bytes): Quadrature component of the finger impedance measurement.
- **total_impedance_in_phase** (uint32, 4 Bytes): In-phase component of the total impedance measurement.
- **total_impedance_quadrature** (uint32, 4 Bytes): Quadrature component of the total impedance measurement.

EDA Scaling

In the EDA Scaling special measurement mode, the AS7058 Chip Library performs calibration measurements for EDA measurements. Note that the measurements are performed continuously, i.e. output is generated continuously until the measurement is explicitly stopped.

Before starting a measurement, a suitable configuration must be written to the registers of the AS7058 AFE. During measurement, the EDA measurement mode temporarily modifies some register values.

Configuration

The EDA Scaling measurement mode is configured via command CMD_ID_VSC_CL_CONFIG_SPECIAL_MEASUREMENT and Target ID value 2 (see Table 28). The payload of the command has the following field with a size of 2 bytes:

- **num_averages** (uint16, 2 Bytes): Number of samples to average for each EDA+ or EDA- measurement.

Output

The output of the EDA Scaling measurement mode is intended to be passed to the EDA application as part of its configuration.

It is provided via command CMD_ID_VSC_CL_SPECIAL_MEASUREMENT_RESULT using Target ID value 2 (see Table 28) and has the following payload fields with a total size of 32 bytes:

- **ref_resistor** (uint32, 4 Bytes): Resistance of the reference resistor in ohm. This value is calculated from OTP data.
- **ref_temperature_adc** (uint32, 4 Bytes): ADC counts of the temperature when the resistance of the reference resistor was measured. This value is calculated from OTP data.
- **temperature_adc** (uint32, 4 Bytes): ADC counts of the temperature during the BioZ measurements. It is set to 0xFFFFFFFF if temperature measurement is disabled.
- **slope** (int32, 4 Bytes): Slope of the temperature scaled by 1000000.
- **short_p** (uint32, 4 Bytes): Measured EDA+ while electrical circuit has a short.
- **short_n** (uint32, 4 Bytes): Measured EDA- while electrical circuit has a short.
- **resistor_p** (uint32, 4 Bytes): Measured EDA+ while electrical circuit is connected with the reference resistor.
- **resistor_n** (uint32, 4 Bytes): Measured EDA- while electrical circuit is connected with the reference resistor.

PD Offset Calibration

In the PD Offset Calibration special measurement mode, the AS7058 Chip Library performs calibration measurements for use with the PD offset reconstruction preprocessing feature of the AS7058 Application Manager. Note that while the other measurement modes generate output continuously, only a single PD Offset Calibration output is generated after starting a measurement. However, it is still required to stop the measurement explicitly.

Before starting a measurement, a suitable configuration must be written to the registers of the AS7058 AFE. During measurement, the PD Offset Calibration measurement mode temporarily modifies some register values.

Configuration

The PD Offset Calibration measurement mode is configured via command CMD_ID_VSC_CL_CONFIG_SPECIAL_MEASUREMENT and Target ID value 3 (see Table 28). The payload of the command has the following fields with a total size of 6 bytes:

- **sub_sample_id** (uint8, 1 Byte): Sub-sample to use for calibration, see Table 13. Depending on the selected sub-sample, either PPG modulator 1 or PPG modulator 2 is calibrated. ECG sub-samples must not be selected.
- **reserved** (1 Byte): Padding byte, reserved for future use. Must be set to zero.
- **num_averages** (uint16, 2 Bytes): Number of samples to acquire and average per calibration point measurement.
- **ppg_sample_period_multiplier** (uint16, 2 Bytes): Base sample period multiplier to use during calibration. The base sample period is 31.25 microseconds. The resulting sample period is obtained by incrementing the multiplier by one and by multiplying the base sample period by the incremented multiplier. For example, a multiplier value of 9 results in sample period of (9 + 1) × 31.25 microseconds = 312.5 microseconds.

Output

The output of the PD Offset Calibration measurement mode is intended to be used for configuring the PD offset compensation preprocessing feature of the AS7058 Application Manager, see command CMD_ID_VSC_AM_CONFIGURE_PREPROCESSING.

It is provided via command CMD_ID_VSC_CL_SPECIAL_MEASUREMENT_RESULT using Target ID value 3 (see Table 28) and has the following payload field with a size of 60 bytes:

- **compensation_table** (Array of uint32, 60 Bytes): Contains the measured ADC count offsets for the selected PPG modulator different for different PD offset current values. PD offset calibration takes the lower 4 bits of the PD offset current of the selected PPG1 or PPG2 sub-sample from the chip configuration and modifies the upper 4 bits during calibration. The first item in the array contains the ADC count offset when the upper 4 bits are set to value 1. The last item contains the offset when the upper 4 bits are set to value 15.

Bio Applications

The firmware automatically provides the outputs of the regular and BioZ measurement modes to the AS7058 Application Manager. The AS7058 Application Manager processes the acquired data using the enabled applications and provides the post-processing result via the RPC protocol. See Table 29 for a list of available applications and their identifiers.

Table 29: Identifiers of Vital Signs Applications

Value	Description
0	Identifies the Raw Data application.
1	Identifies the HRM application.
2	Identifies the SpO2 application.
3	Identifies the Signal Range Detection application.
4	Identifies the BioZ application.
5	Identifies the Electrodermal Activity application.
6	Identifies the Streaming application.
7	Identifies the Respiration Rate application.

Raw Data

The Raw Data application provides raw measurement data acquired by the AS7058 AFE and raw AGC status information. It can also provide raw accelerometer measurement data. This application is only functional when the regular measurement mode is used.

Signal Routing

The Raw Data application always outputs the data of all the enabled sub-samples. Therefore, it does not support signal routing.

Preprocessing

The Raw Data application does not support signal sample preprocessing.

Configuration

The Raw Data application is configured via the command CMD_ID_VSC_AM_APP_CONFIG using Target ID value 0 (see Table 29). The size of the command payload is 1 byte.

- **include_acc** (uint8, 1 Byte): Accelerometer data is included in the output when this field is set to 1. The output always contains zero accelerometer samples when this field is set to 0.

Output

The output of the Raw Data application is provided via the command CMD_ID_VSC_AM_APP_OUTPUT using Target ID value 0 (see Table 29). The size of the command payload is variable, but never exceeds 149 bytes.

- **packet_counter** (uint8, 1 Byte): Contains the number of outputs that have been sent during the current measurement session. When a new measurement session starts, the counter is reset so that this field is set to zero in the first output. The counter wraps-around to zero after reaching the maximum value (255).
- **fifo_samples_num** (uint8, 1 Byte): Contains the number of FIFO samples in the output data.
- **acc_samples_num** (uint8, 1 Byte): Contains the number of accelerometer samples in the output data.
- **flags_and_agc_statuses_num** (1 Byte):
    - **reserved** (Bits 7:6): Reserved for future use. The bits are always unset in the current implementation.
    - **ext_event_occurrence_num_present** (Bit 5): Indicates the presence of the external event occurrence count in the output data. It is present when this bit is set.
    - **status_events_present** (Bit 4): Indicates the presence of status event information in the output data. It is present when this bit is set.
    - **agc_statuses_num** (Bit 3:0, least-significant bits): Contains the number of AGC statuses in the output data as an unsigned integer. AGC statuses are only present in the output data if at least one AGC algorithm instance is enabled and if the status of at least one AGC algorithm instance is different from the corresponding previously provided AGC status. If AGC statuses are provided, the number of provided AGC statues is equivalent to the number of enabled AGC algorithm instances, i.e. the current AGC statuses of all enabled AGC algorithm instances are provided. If AGC statuses are not provided, this number is set to zero.
- **fifo_samples** (Array of uint24, variable-length): This field contains FIFO samples as received from the AS7058 FIFO data. The number of contained FIFO samples is provided by the *fifo_samples_num* field.
- **acc_samples** (Array of Accelerometer Sample, variable-length): This field contains accelerometer samples as received from the accelerometer. The number of contained accelerometer samples is provided by the *acc_samples_num* field.
- **agc_statuses** (Array of AGC Status, variable-length): This field contains AGC status information as received from the AS7058 Chip Library. The number of statuses is provided by the *agc_statuses_num* field. The provided AGC statuses are valid for all FIFO samples contained in this Raw Data application output. They are also valid for all subsequent Raw Data application outputs that do not contain AGC status information (i.e. where the value of the *agc_statuses_num* field is zero) until a Raw Data application output is received that does contain AGC status information.
- **status_events** (Optional Status Event, 0 or 9 Bytes): This field contains status events of the AS7058 AFE. The presence of this field is determined by the *status_events_present* field.

- **ext_event_occurrence_num** (Optional uint8, 0 or 1 Byte): This field contains the number of external events that have occurred while the data present in this output data instance has been acquired. If the field is not present, no external event have occurred. The presence of this field is determined by the *ext_event_occurrence_num_present* field.

Heart Rate Monitoring

The AS7058 firmware contains the ams OSRAM heart rate monitoring algorithm. Its input data are PPG samples from a single sub-sample of the AS7058 AFE and accelerometer data. The output of the algorithm contains a heart rate value and pulse rate variability (PRV) data. This application is only functional when the regular measurement mode is used.

Signal Routing

When configuring the signal routing for the HRM application via command CMD_ID_VSC_AM_SET_SIGNAL_ROUTING and Target ID value 1 (see Table 29), build the array containing the sub-sample IDs using the indices provided in Table 30.

Table 30: Sensor Signals Provided to the HRM Bio App

Value	Description
0	PPG signal.

Preprocessing

Signal sample preprocessing can be enabled for the HRM application via the command CMD_ID_VSC_AM_CONFIGURE_PREPROCESSING using Target ID value 1 (see Table 29). When PD offset compensation is enabled, it is applied to the signal if PD offset currents are available for its samples. The signal sample preprocessing support of the HRM application is experimental because of bit width limitations of the underlying algorithm.

Configuration

The HRM application is configured via the command CMD_ID_VSC_AM_APP_CONFIG using Target ID value 1 (see Table 29). The size of the command payload is 1 byte.

- **enable_prv** (uint8, 1 Byte): PRV data is included in the output when set to 1. It is not included in the output when set to 0. Note that the *prv_ms* field is always present in the output, independent of this setting.

Output

The output of the HRM application is provided via the command CMD_ID_VSC_AM_APP_OUTPUT using Target ID value 1 (see Table 29). The size of the command payload is 16 bytes.

- **heart_rate** (uint16, 2 Bytes): Contains the heart rate. The unit is 0.1 bpm.
- **quality** (uint8, 1 Byte): Contains information about the quality of the heart rate signal. A value of zero means the best quality.
- **motion_frequency** (uint8, 1 Byte): Contains the detected motion frequency. The unit is bpm. A value of zero means that no motion has been detected.
- **prv_ms** (Array of uint16, 10 Bytes): Contains PRV data. The unit is milliseconds. This field has a fixed size, but the number of valid items is dynamic.
- **prv_ms_num** (uint8, 1 Byte): Contains the number of valid *prv_ms* items.
- **reserved** (uint8, 1 Byte): Padding byte.

SpO2 Monitoring

The AS7058 firmware contains the ams OSRAM SpO2 algorithm. Its input data are red light PPG samples, infrared light PPG samples, and ambient light samples. Each type of input data is provided by a different sub-sample of the AS7058 AFE. The output of the algorithm contains an SpO2 value, a heart rate value, a Perfusion Index value, and an average R-value. This application is only functional when the regular measurement mode is used.

Signal Routing

When configuring the signal routing for the SpO2 application via command CMD_ID_VSC_AM_SET_SIGNAL_ROUTING and Target ID value 2 (see Table 29), build the array containing the sub-sample IDs using the indices provided in Table 31.

Table 31: Sensor Signals Provided to the SpO2 Bio App

Value	Description
0	Red PPG signal.
1	Infrared PPG signal.
2	Ambient light signal.

Preprocessing

Signal sample preprocessing can be enabled for the SpO2 application via the command CMD_ID_VSC_AM_CONFIGURE_PREPROCESSING using Target ID value 2 (see Table 29). When PD offset compensation is enabled, it is applied to all signal for whose samples PD offset currents are available. The signal sample preprocessing support of the SpO2 application is experimental because of bit width limitations of the underlying algorithm.

Configuration

The SpO2 application is configured via the command CMD_ID_VSC_AM_APP_CONFIG using Target ID value 2 (see Table 29). The size of the command payload is 10 bytes.

- **a** (uint16, 2 Bytes): Quadratic correction coefficient a.

- **b** (uint16, 2 Bytes): Quadratic correction coefficient b.
- **c** (uint16, 2 Bytes): Quadratic correction coefficient c.
- **dc_comp_red** (uint16, 2 Bytes): DC compensation for the red signal.
- **dc_comp_ir** (uint16, 2 Bytes): DC compensation for the infrared signal.

Output

The output of the SpO2 application is provided via the command CMD_ID_VSC_AM_APP_OUTPUT using Target ID value 2 (see Table 29). The size of the command payload is 18 bytes.

- **status** (uint8, 1 Byte): The value of the field is zero when the structure contains valid SpO2 data. It is one when no result is present in the data.
- **quality** (uint8, 1 Byte): Contains the quality of the signal. The unit is 1%.
- **spo2** (uint16, 2 Bytes): Contains the SpO2 measurement. The unit is 0.01%.
- **heart_rate** (uint16, 2 Bytes): Contains the heart rate. The unit is 0.1 bpm.
- **pi** (uint16, 2 Bytes): Contains the Perfusion Index measurement. The unit is 0.01%.
- **average_r** (uint16, 2 Bytes): Contains the average R-value. The unit is 10000.
- **reserved** (8 Bytes): Reserved bytes for backwards compatibility. They are always set to zero.

Respiration Rate Monitoring

The AS7058 firmware contains the ams OSRAM respiration rate monitoring algorithm. Its input data are PPG samples from a single sub-sample of the AS7058 AFE. The output of the algorithm contains a respiratory rate value and a corresponding confidence values. This application is only functional when the regular measurement mode is used.

Signal Routing

When configuring the signal routing for the RRM application via command CMD_ID_VSC_AM_SET_SIGNAL_ROUTING and Target ID value 7 (see Table 29), build the array containing the sub-sample IDs using the indices provided in Table 32.

Table 32: Sensor Signals Provided to the Respiration Rate Bio App

Value	Description
0	PPG signal that is used for respiration rate monitoring.

Preprocessing

Signal sample preprocessing can be enabled for the RRM application via the command CMD_ID_VSC_AM_CONFIGURE_PREPROCESSING using Target ID value 7 (see Table 29). When PD offset compensation is enabled, it is applied to the signal if PD offset currents are available for its samples.

Configuration

The RRM application does not support configuration. It is not required to send CMD_ID_VSC_AM_APP_CONFIG commands with Target ID value 7 (see Table 29). If such a command is sent despite not being necessary, the size of the command payload must be zero bytes.

Output

The output of the RRM application is provided via the command CMD_ID_VSC_AM_APP_OUTPUT using Target ID value 7 (see Table 29). The size of the command payload is 4 bytes.

- **respiratory_rate** (uint16, 2 Bytes): Respiration rate in breaths per minute multiplied by 100.
- **confidence** (uint8, 1 Byte): Confidence value associated with the respiratory rate ranging between 0 and 100. A value of 100 represents maximum certainty.
- **reserved** (uint8, 1 Byte): Padding byte.

Note that the first output is available approximately 20 seconds after the start of the measurement. After the first output is available, successive outputs are available approximately every second.

Signal Range Detection

The Signal Range Detection application observes the current value of a signal and splits the ADC range into three signal status regions with configurable boundaries. It outputs the signal status region into which the signal currently falls. This application is only functional when the regular measurement mode is used.

Signal Routing

When configuring the signal routing for the Signal Range Detection application via command CMD_ID_VSC_AM_SET_SIGNAL_ROUTING and Target ID value 3 (see Table 29), build the array containing the sub-sample IDs using the indices provided in Table 33.

Table 33: Sensor Signals Provided to the Signal Range Detection Bio App

Value	Description
0	Signal to monitor.

Preprocessing

Signal sample preprocessing can be enabled for the Signal Range Detection application via the command CMD_ID_VSC_AM_CONFIGURE_PREPROCESSING using Target ID value 3 (see Table 29). When PD offset compensation is enabled, it is applied to the signal if PD offset currents are available for its samples.

Configuration

The Signal Range Detection application is configured via the command CMD_ID_VSC_AM_APP_CONFIG using Target ID value 3 (see Table 29). The size of the command payload is 12 bytes.

- **lower_threshold** (int32, 4 Bytes): Lower threshold of the center signal status region in raw counts.
- **upper_threshold** (int32, 4 Bytes): Upper threshold of the center signal status region in raw counts. The upper threshold must not be lower than the lower threshold.
- **change_detection_samples_num** (uint8, 1 Byte): Minimum number of consecutive samples that need to fall into the new signal status region before the signal status change may trigger. For example, if this value is set to three, the signal status changes once four consecutive samples that fall into the same new signal status region have been received.
- **reserved** (3 Bytes): Reserved for future use. Always set them to zero when configuring the application.

Output

The output of the Signal Range Detection application is provided via the command CMD_ID_VSC_AM_APP_OUTPUT using Target ID value 3 (see Table 29). The size of the command payload is 1 byte.

- **flags** (1 Byte):
  - **reserved** (Bits 7:5): Reserved for future use. The bits are always unset in the current implementation.
  - **output_reason** (Bit 4): Indicates whether this output is generated because the signal status region changed or whether this is a periodic status update. The signal status region changed when this bit is set.
  - **reserved** (Bits 3:2): Reserved for future use. The bits are always unset in the current implementation.
  - **signal_status** (Bit 1:0, least-significant bits): Contains the current signal status region of the signal. The lower region is represented using value 0b00, the center region is represented using value 0b01, and the upper region is represented using value 0b10.

BioZ

The BioZ application corrects BioZ samples acquired using the special BioZ Chip Library measurement mode. To use the application, a system calibration using known reference impedances needs to be performed. The calibration measurement results are passed to to the BioZ application via its configuration. This application is only functional when the BioZ measurement mode is used.

Signal Routing

The BioZ application receives input data from a special Chip Library measurement mode that automatically selects the appropriate sub-samples. Therefore, it does not support signal routing.

Preprocessing

The BioZ application does not support signal sample preprocessing.

Configuration

The BioZ application is configured via the command CMD_ID_VSC_AM_APP_CONFIG using Target ID value 4 (see Table 29). The size of the command payload is 92 bytes.

- **scaling_results** (64 Bytes): Results of the calibration measurement using the special measurement mode 1 (see Table 28). This data is obtained via command CMD_ID_VSC_CL_SPECIAL_MEASUREMENT_RESULT after performing the special measurement using commands CMD_ID_VSC_CL_CONFIG_SPECIAL_MEASUREMENT and CMD_ID_VSC_START_MEASUREMENT.
- **ref_known_imp_body_magnitude** (uint32, 4 Bytes): Magnitude of the known reference body impedance used for calibration scaled by 1000. For example, a magnitude of 15.2 is represented using value 15200.
- **ref_known_imp_body_phase** (int32, 4 Bytes): Phase of the known reference body impedance used for calibration in degrees scaled by 1000. For example, 15.2 degrees is represented using value 15200.
- **ref_known_imp_wrist_magnitude** (uint32, 4 Bytes): Magnitude of the known reference wrist impedance used for calibration scaled by 1000.
- **ref_known_imp_wrist_phase** (int32, 4 Bytes): Phase of the known reference wrist impedance used for calibration in degrees scaled by 1000.
- **ref_known_imp_finger_magnitude** (uint32, 4 Bytes): Magnitude of the known reference finger impedance used for calibration scaled by 1000.
- **ref_known_imp_finger_phase** (int32, 4 Bytes): Phase of the known reference finger impedance used for calibration in degrees scaled by 1000.
- **input_drop_num** (uint8, 1 Byte): Number of measurement data inputs to ignore after one measurement data input has been received. The rate of generated application outputs can be controlled using this setting. For example, if this field is set to 2, every third measurement data input is processed.
- **reserved** (3 Bytes): Reserved for future use. Always set them to zero when configuring the application.

Output

The output of the BioZ application is provided via the command CMD_ID_VSC_AM_APP_OUTPUT using Target ID value 4 (see Table 29). The size of the command payload is 24 bytes.

- **imp_body_magnitude** (uint32, 4 Bytes): Magnitude of the body impedance scaled by 1000.
- **imp_body_phase** (int32, 4 Bytes): Phase of the body impedance in degrees scaled by 1000.
- **imp_wrist_magnitude** (uint32, 4 Bytes): Magnitude of the wrist impedance scaled by 1000.
- **imp_wrist_phase** (int32, 4 Bytes): Phase of the wrist impedance in degrees scaled by 1000.
- **imp_finger_magnitude** (uint32, 4 Bytes): Magnitude of the finger impedance scaled by 1000.
- **imp_finger_phase** (int32, 4 Bytes): Phase of the finger impedance in degrees scaled by 1000.

EDA

The EDA application calculates body resistance values from acquired data. To use the EDA application, it is required to perform a scaling measurement using the corresponding special Chip Library measurement mode. The scaling measurement results are passed to to the EDA application via its configuration. While a special Chip Library measurement mode needs to be used for performing the scaling measurement, the application is only functional when the regular measurement mode is used.

Signal Routing

When configuring the signal routing for the EDA application via command CMD_ID_VSC_AM_SET_SIGNAL_ROUTING using Target ID value 5 (see Table 29), build the array containing the sub-sample IDs using the indices provided in Table 34.

Table 34: Sensor Signals Provided to the AS7058 Electrodermal Activity Application

Value	Description
0	Signal where EDA+ and EDA- samples are provided alternatingly.
1	Temperature signal.

Preprocessing

The EDA application does not support signal sample preprocessing.

Configuration

The EDA application is configured via the command CMD_ID_VSC_AM_APP_CONFIG using Target ID value 5 (see Table 29). The size of the command payload is 44 bytes.

- **scaling_results** (32 Bytes): Results of the special measurement mode 2 (see Table 28). This data is obtained via command CMD_ID_VSC_CL_SPECIAL_MEASUREMENT_RESULT after performing the special measurement using commands

CMD_ID_VSC_CL_CONFIG_SPECIAL_MEASUREMENT and CMD_ID_VSC_START_MEASUREMENT.

- **eda_avg_num** (uint32, 4 Bytes): Number of positive and negative EDA samples to average for a single EDA calculation. eda_avg_num positive EDA samples and eda_avg_num negative EDA samples are averaged. Must be at least 1.
- **eda_drop_num** (uint32, 4 Bytes): Number of subsequent samples of positive and negative EDA samples to drop after an average has been calculated. eda_avg_num positive EDA samples and eda_avg_num negative EDA samples are dropped. This allows for sample rate reduction.
- **temperature_delta_threshold** (uint32, 4 Bytes): Temperature threshold in ADC counts for triggering a recalibration warning. Impedance scaling should be re-performed when the corresponding bit is set in the application output. The warning is triggered when the current temperature is less than (scaling_results.temperature_adc - temperature_delta_threshold) or when the current temperature is greater than (scaling_results.temperature_adc + temperature_delta_threshold).

Output

The output of the EDA application is provided via the command CMD_ID_VSC_AM_APP_OUTPUT using Target ID value 5 (see Table 29). The size of the command payload is 16 bytes.

- **flags** (4 Bytes):
  - **reserved** (Bits 31:1): Reserved for future use. The bits are always unset in the current implementation.
  - **recalibration_warning** (Bit 0, least-significant bit): Indicates whether impedance scaling should be re-performed due to temperature change. Re-perform impedance scaling when this bit is set.
- **resistance** (int32, 4 Bytes): Average of the positive and negative EDA resistance in ohm.
- **resistance_positive** (int32, 4 Bytes): Positive EDA resistance in ohm.
- **resistance_negative** (int32, 4 Bytes): Negative EDA resistance in ohm.

Streaming

The Streaming application outputs different categories of data over a single application. An instance of information output via the Streaming application is referred to as an item. Each item has an item ID associated that indicates the category of data. In the AS7058 firmware, the item IDs listed in Table 35 are supported.

Table 35: Identifiers of Streaming App Items

Value	Description
0	FIFO data stream.
1	AGC statuses.
2	Accelerometer data.
3	Sensor events.

Table 35 (Continued): Identifiers of Streaming App Items

Value	Description
4	External events.
5	Samples of PPG modulator 1, sub-sample 1.
6	Samples of PPG modulator 1, sub-sample 2.
7	Samples of PPG modulator 1, sub-sample 3.
8	Samples of PPG modulator 1, sub-sample 4.
9	Samples of PPG modulator 1, sub-sample 5.
10	Samples of PPG modulator 1, sub-sample 6.
11	Samples of PPG modulator 1, sub-sample 7.
12	Samples of PPG modulator 1, sub-sample 8.
13	Samples of PPG modulator 2, sub-sample 1.
14	Samples of PPG modulator 2, sub-sample 2.
15	Samples of PPG modulator 2, sub-sample 3.
16	Samples of PPG modulator 2, sub-sample 4.
17	Samples of PPG modulator 2, sub-sample 5.
18	Samples of PPG modulator 2, sub-sample 6.
19	Samples of PPG modulator 2, sub-sample 7.
20	Samples of PPG modulator 2, sub-sample 8.
21	Samples of ECG modulator, sequence 1, sub-sample 1.
22	Samples of ECG modulator, sequence 1, sub-sample 2.
23	Samples of ECG modulator, sequence 2, sub-sample 1.

Signal Routing

The Streaming application does not support signal routing. All available data is provided in the output, provided that the corresponding item IDs have not been suppressed via the application configuration.

Preprocessing

Signal sample preprocessing can be enabled for the Streaming application via the command CMD_ID_VSC_AM_CONFIGURE_PREPROCESSING using Target ID value 6 (see Table 29). When PD offset compensation is enabled, it is applied to all signal for whose samples PD offset currents are available. When a sample is preprocessed, only the value after preprocessing is provided via the corresponding item ID, for example 5 (see Table 35). Signal sample preprocessing has no effect on data provided via other item IDs.

Configuration

The Streaming application is configured via the command CMD_ID_VSC_AM_APP_CONFIG using Target ID value 6 (see Table 29). The size of the command payload is 32 byte.

- **item_filter** (uint64, 8 Bytes): Used to suppress items with specific IDs in the output packages. Bit index 0 corresponds to item ID 0, bit index 1 to item ID 1. If a bit is not set, items with the corresponding ID are suppressed. Default value is 0xFFFFFFFFFFFFFFFF, i.e. no items are suppressed by default.
- **max_output_size** (uint32, 4 Bytes): Maximum size of a single generated output. Default value is 150 bytes.
- **reserved** (4 Bytes): Unused padding byte. Must to be set to zero.

Output

The first byte of each output generated by the Streaming application contains an Output Counter field of type uint8. The counter contains the number of outputs that have been sent during the current measurement session. When a new measurement session starts, the counter is reset so that it is set to zero in the first output. The counter wraps-around to zero after reaching the maximum value (255).

Items follow immediately after the the Output Counter field. A single output generated by the Streaming application can hold multiple items. Each item has 2-byte header, followed by the payload data. Inside a single Streaming application output, the items are ordered in the order they were generated. No padding bytes are inserted between items. The Streaming application may split an item into multiple items, for example if the complete item does not fit into a single output. The Fragmentation bit in the item header indicates whether a split has been performed. When an item has been split, the items it has been split into may be contained in different Streaming application outputs. The size of a Streaming application output never exceeds the configured maximum output size.

Table 36: Streaming Item Header Fields

Name	Bits	Description
Item ID	15:10	Contains the item ID, see Table 35.
Fragmentation	9	If this bit is set, this item has been split and is continued in the next item.
Payload Size	8:0	Contains the size of the payload in bytes.

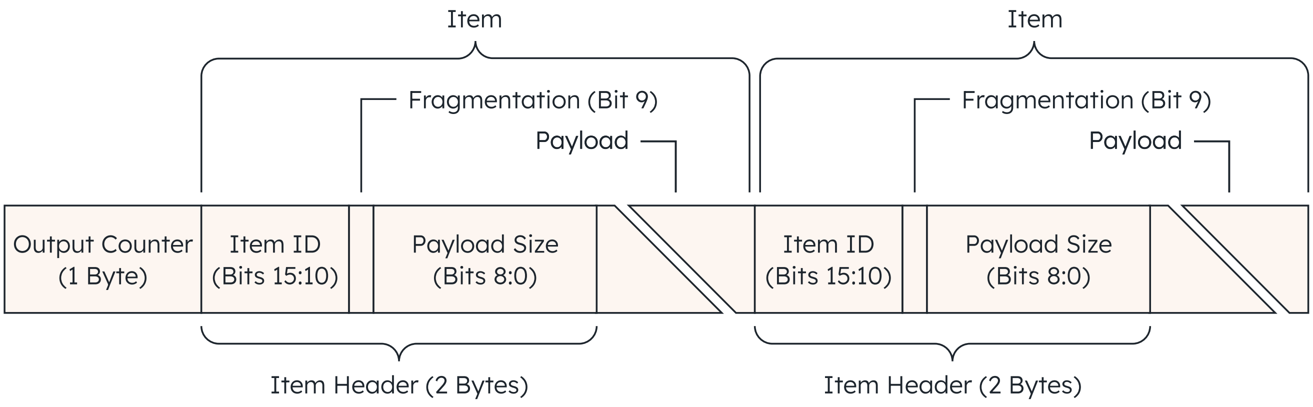

Figure 4: Structure of a Streaming Application Output with Two Items

Table 37: Streaming Item Payload Formats

Item ID	Type	Description
0 (see Table 35)	Array of uint8	FIFO data stream as received from the AS7058 AFE.
1 (see Table 35)	Array of AGC Status	Status information from the AGC algorithm. One status is generated per enabled AGC instances.
2 (see Table 35)	Array of Accelerometer Sample	Acquired accelerometer data.
3 (see Table 35)	Status Event	Status events of the AS7058 AFE.
4 (see Table 35)	uint8	Number of external events that have occurred since this item was last output.
5 (see Table 35) to 23 (see Table 35)	Array of Sample with PD Offset	Extracted samples from the corresponding sub-sample in the order they were acquired. If signal sample preprocessing is enabled, the preprocessed samples are output instead of the original samples.

Supported Commands

The AS7058 firmware implements commands to perform measurements using the AS7058 AFE. They make the interface of the AS7058 Chip Library and AS7058 Application Manager available via USB and Bluetooth Low Energy. Additionally, the CSS Core Firmware implements commands to readout firmware and device information, and to control the peripherals of the microcontroller.

Vital Signs Commands

The following commands are supported exclusively by the AS7058 firmware.

CMD_ID_VSC_INITIALIZE
Command ID: 0x64

This command initializes the AS7058 Chip Library, the AS7058 Application Manager, and the accelerometer driver. It must be sent before any other command accessing one of these components is used.

Target ID

Ignored, set it to zero.

Input Payload (Variable Length)

- **interface_description** (string, variable-length): Interface selection string that is passed to the OSAL of the AS7058 Chip Library. This field is currently ignored, set it to an empty string (zero bytes).

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_SHUTDOWN
Command ID: 0x65

This command de-initializes the AS7058 Chip Library, the AS7058 Application Manager, and the accelerometer driver. If there is an active measurement session, sending this command stops the measurement session. The software components must be re-initialized using the CMD_ID_VSC_INITIALIZE command, which must be sent before any other command accessing one of the components is used.

Target ID

Ignored, set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_CL_SET_REG_GROUP
Command ID: 0x66

This command writes a group of registers by calling the corresponding AS7058 Chip Library function. It can only be sent while there is no active measurement session.

Target ID

Ignored, set it to zero.

Input Payload (Variable Length)

- **reg_group_id** (uint8, 1 Byte): ID of the register group to write (see Table 15).
- **reg_group_data** (variable-length): Register group content to write.

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_CL_GET_REG_GROUP
Command ID: 0x67

This command reads a group of registers by calling the corresponding AS7058 Chip Library function. It can only be sent while there is no active measurement session.

Target ID

Ignored, set it to zero.

Input Payload (1 Byte)

- **reg_group_id** (uint8, 1 Byte): ID of the register group to read (see Table 15).

Output Payload (Variable Length)

- **reg_group_data** (variable-length): Content of the register group.

CMD_ID_VSC_CL_SET_AGC_CONFIG
Command ID: 0x68

This command sets the AGC configuration by calling the corresponding AS7058 Chip Library function. It can only be sent while there is no active measurement session.

Target ID

Ignored, set it to zero.

Input Payload (Variable Length)

- **agc_config** (Array of AGC Configuration, variable-length): Each configuration item configures one instance of the AGC algorithm. Up to four AGC instances are supported. If less than four configuration items are provided, the remaining AGC algorithm instances are disabled.

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_CL_GET_AGC_CONFIG
Command ID: 0x69

This command gets the current AGC configuration by calling the corresponding AS7058 Chip Library function.

Target ID

Ignored, set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (Variable Length)

- **agc_config** (Array of AGC Configuration, variable-length): Configurations of the enabled AGC algorithm instances. The number of items in this array is equal to the number of enabled AGC algorithm instances.

CMD_ID_VSC_CL_WRITE_REGISTER
Command ID: 0x6A

This command writes a single register by calling the corresponding AS7058 Chip Library function.

Target ID

Ignored, set it to zero.

Input Payload (2 Bytes)

- **reg_address** (uint8, 1 Byte): Register address.
- **reg_value** (uint8, 1 Byte): Register value to write.

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_CL_READ_REGISTER
Command ID: 0x6B

This command reads a single register by calling the corresponding AS7058 Chip Library function.

Target ID

Ignored, set it to zero.

Input Payload (1 Byte)

- **reg_address** (uint8, 1 Byte): Register address.

Output Payload (1 Byte)

- **reg_value** (uint8, 1 Byte): Current value of the register.

CMD_ID_VSC_CL_GET_MEAS_CONFIG
Command ID: 0x6C

This command returns the measurement configuration as provided by the AS7058 Chip Library.

Target ID

Ignored, set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (28 Bytes)

- **ppg_sample_period_us** (uint32, 4 Bytes): Sample period of PPG sub-samples in microseconds.
- **ecg_seq1_sample_period_us** (uint32, 4 Bytes): Sample period of ECG sequencer 1 sub-samples in microseconds.
- **ecg_seq2_sample_period_us** (uint32, 4 Bytes): Sample period of ECG sequencer 2 sub-samples in microseconds.
- **fifo_map** (uint32, 4 Bytes): Bitfield indicating which sub-samples are enabled. The bits corresponding to the enabled sub-samples are set. Sub-samples whose bits are not set are disabled. See Table 38. More than one bit can be set.
- **agc_channels** (Array of uint8, 4 Bytes): IDs of the sub-samples that are controlled by the individual AGC algorithm instances (see Table 13). The value 0 (see Table 13) indicates that the AGC algorithm instance is disabled.
- **sar_map** (uint32, 4 Bytes): Bitfield indicating which sub-samples have SAR enabled. When SAR is enabled for a sub-sample, the corresponding bit is set. SAR is not enabled for a sub-sample when the corresponding bit is not set. See Table 38. More than one bit can be set.
- **sar_transfer_mode** (uint8, 1 Byte): Indicates how the SAR status is provided by the AS7058 AFE (see Table 39).
- **reserved** (Array of uint8, 3 Bytes): Padding bytes, reserved for future use.

Table 38: Sub-Sample Flags

Value	Description
0x0	No sub-sample is selected.
0x1	Flag for sub-sample 1 of PPG modulator 1.
0x2	Flag for sub-sample 2 of PPG modulator 1.
0x4	Flag for sub-sample 3 of PPG modulator 1.
0x8	Flag for sub-sample 4 of PPG modulator 1.
0x10	Flag for sub-sample 5 of PPG modulator 1.
0x20	Flag for sub-sample 6 of PPG modulator 1.
0x40	Flag for sub-sample 7 of PPG modulator 1.
0x80	Flag for sub-sample 8 of PPG modulator 1.
0x100	Flag for sub-sample 1 of PPG modulator 2.
0x200	Flag for sub-sample 2 of PPG modulator 2.
0x400	Flag for sub-sample 3 of PPG modulator 2.
0x800	Flag for sub-sample 4 of PPG modulator 2.
0x1000	Flag for sub-sample 5 of PPG modulator 2.
0x2000	Flag for sub-sample 6 of PPG modulator 2.
0x4000	Flag for sub-sample 7 of PPG modulator 2.
0x8000	Flag for sub-sample 8 of PPG modulator 2.
0x10000	Flag for sub-sample 1 of sequence 1 of the ECG modulator.
0x20000	Flag for sub-sample 2 of sequence 1 of the ECG modulator.
0x40000	Flag for sub-sample 1 of sequence 2 of the ECG modulator.

Table 39: Represents How the AS7058 Afe Provides the Sar Status

Value	Description
0	PD offset is provided in a separate FIFO item.
1	ADC data and upper four bits of the PD offset are contained in a single FIFO item. The AS7058 AFE truncates the ADC data to make room for the upper four bits of the PD offset. The lower four bits of the PD offset (which are not controlled by SAR) must be known from configuration or must be read from the corresponding register.

CMD_ID_VSC_GET_VERSION
Command ID: 0x6D

This command returns the version of a software component.

Target ID

Selects the software component (see Table 40).

Table 40: Identifiers of Vital Signs Software Components

Value	Description
0	ID of Chip Library.
1	ID of Application Manager.

Input Payload (0 Bytes)

- *None*

Output Payload (Variable Length)

- **version** (string, variable-length): Version string of the software component (not NULL-terminated).

CMD_ID_VSC_START_MEASUREMENT
Command ID: 0x6E

This command starts a measurement session. This command puts the AS7058 Application Manager into measurement state, starts accelerometer measurements, and starts measurements using the AS7058 AFE by calling the corresponding AS7058 Chip Library function. During the measurement session, measurement data is provided asynchronously. If bio applications are enabled in the AS7058 Application Manager, output is provided via the CMD_ID_VSC_AM_APP_OUTPUT command. If a special measurement mode (any measurement mode other than mode 0 (see Table 28)) is used, the results of the special measurements are provided via CMD_ID_VSC_CL_SPECIAL_MEASUREMENT_RESULT, independent of the applications enabled in the AS7058 Application Manager.

Target ID

Ignored, set it to zero.

Input Payload (0 - 1 Byte)

- **mode** (Optional uint8, 0 - 1 Byte): Measurement mode (see Table 28). If this field is not present, measurement mode 0 (see Table 28) is assumed.

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_STOP_MEASUREMENT
Command ID: 0x6F

This command stops an active measurement session. It stops AS7058 measurements by calling the corresponding AS7058 Chip Library function, stops accelerometer measurements, and causes the AS7058 Application Manager to leave the measurement state.

Target ID

Ignored, set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_AM_SET_SIGNAL_ROUTING
Command ID: 0x70

This command configures which AS7058 sub-samples are used as bio application signals for a given application by calling the corresponding AS7058 Application Manager function. It can only be sent while there is no active measurement session.

Target ID

ID of the app to configure (see Table 29).

Input Payload (Variable Length)

- **sub_sample_ids** (Array of uint8, variable-length): IDs of the sub-samples that will be used as signals for the given app. Please refer to Bio Applications for the list of supported signals per app and the indicies that should be used when building this array.

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_AM_ENABLE_APPS
Command ID: 0x71

This command enables and disables bio applications by calling the corresponding AS7058 Application Manager function. It can only be sent while there is no active measurement session.

Target ID

Ignored, set it to zero.

Input Payload (4 Bytes)

- **enabled_apps** (uint32, 4 Bytes): Bitfield that sets which apps are enabled and disabled. The bits corresponding to the enabled apps are set. Apps whose bits are not set are disabled. See Table 41. More than one bit can be set.

Table 41: Application Flags Bitmasks

Value	Description
0x1	Bitmask where the bit representing the Raw Data Streaming application is set.
0x2	Bitmask where the bit representing the HRM application is set.
0x4	Bitmask where the bit representing the SpO2 application is set.
0x8	Bitmask where the bit representing the Signal Range Detection application is set.
0x10	Bitmask where the bit representing the BioZ application is set.
0x20	Bitmask where the bit representing the EDA application is set.
0x40	Bitmask where the bit representing the Streaming application is set.
0x80	Bitmask where the bit representing the Respiration Rate application is set.

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_AM_APP_CONFIG
Command ID: 0x72

This command configures a given bio application by calling the corresponding AS7058 Application Manager function. It can only be sent while there is no active measurement session.

Target ID

ID of the app to configure (see Table 29).

Input Payload (Variable Length)

- **configuration** (variable-length): Application-specific configuration (see Bio Applications).

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_AM_APP_OUTPUT
Command ID: 0x73

This command is used for asynchronous messages sent by the firmware containing bio application output during an active measurement session.

Target ID

ID of the app that provides output (see Table 29).

Input Payload (0 Bytes)

- *None*

Output Payload (Variable Length)

- **data** (variable-length): Application-specific data (see Bio Applications).

CMD_ID_VSC_MEAS_ERROR
Command ID: 0x74

This command is used for asynchronous messages sent by the firmware when an error occurred during an active measurement session. The occurrence of an error stops the measurement session. The error code is contained in the corresponding field of the RPC message.

Target ID

Ignored, set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_AM_EXT_EVENT
Command ID: 0x75

This command increments the external event counter of the AS7058 Application Manager by calling the corresponding function of the AS7058 Application Manager.

Target ID

Ignored, set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_ACC_SET_SAMPLE_PERIOD
Command ID: 0x76

This command sets the sample period of the accelerometer. The accelerometer supports the following sample periods: 1 second (1 Hertz), 100 milliseconds (10 Hertz), 40 milliseconds (25 Hertz), 20 milliseconds (50 Hertz), 10 milliseconds (100 Hertz), and 5 milliseconds (200 Hertz).

Target ID

Ignored, set it to zero.

Input Payload (4 Bytes)

- **sample_period** (uint32, 4 Bytes): Sample period of the accelerometer in microseconds.

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_ACC_GET_SAMPLE_PERIOD
Command ID: 0x77

This command gets the sample period of the accelerometer. See CMD_ID_VSC_ACC_SET_SAMPLE_PERIOD for the list of supported sample periods.

Target ID

Ignored, set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (4 Bytes)

- **sample_period** (uint32, 4 Bytes): Sample period of the accelerometer in microseconds.

CMD_ID_VSC_CL_CONFIG_SPECIAL_MEASUREMENT
Command ID: 0x78

This command configures special measurement modes of the AS7058 Chip Library.

Target ID

ID of the special measurement mode. Supports mode 1 (see Table 28), mode 2 (see Table 28), and mode 3 (see Table 28).

Input Payload (Variable Length)

- **configuration** (variable-length): Mode-specific configuration (see Measurement Modes).

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_CL_SPECIAL_MEASUREMENT_RESULT
Command ID: 0x79

This command is used for asynchronous messages sent by the firmware containing the results of a special measurement.

Target ID

ID of the special measurement mode. This is either mode 1 (see Table 28), mode 2 (see Table 28), or mode 3 (see Table 28).

Input Payload (0 Bytes)

- *None*

Output Payload (Variable Length)

- **output** (variable-length): Mode-specific output (see Measurement Modes).

CMD_ID_VSC_AM_ENABLE_PREPROCESSING
Command ID: 0x7A

This command enables and disables signal sample pre-processing features for a bio application by calling the corresponding AS7058 Application Manager function. It can only be sent while there is no active measurement session.

Target ID

ID of the app to configure (see Table 29).

Input Payload (4 Bytes)

- **flags** (uint32, 4 Bytes): Flags of enabled preprocessing features, see Table 42. A preprocessing feature is enabled when the corresponding bit is set and disabled when the bit is not set.

Table 42: Signal Sample Pre-Processing Features Bitmasks

Value	Description
0x1	Bitmask where the bit representing the PD offset compensation preprocessing feature is set.

Output Payload (0 Bytes)

- *None*

CMD_ID_VSC_AM_CONFIGURE_PREPROCESSING Command ID: 0x7B

This command configures a signal sample pre-processing feature by calling the corresponding AS7058 Application Manager function. It can only be sent while there is no active measurement session.

Target ID

ID of the signal sample preprocessing feature to configure (see Table 43).

Table 43: Identifiers of the Signal Sample Pre-Processing Features

Value	Description
0	Represents PD offset compensation.

Input Payload (120 Bytes)

- **compensation_table_ppg1** (Array of uint32, 60 Bytes): Compensation value table for PPG modulator 1 as acquired via measurement mode 3 (see Table 28).
- **compensation_table_ppg2** (Array of uint32, 60 Bytes): Compensation value table for PPG modulator 2 as acquired via measurement mode 3 (see Table 28).

Output Payload (0 Bytes)

- *None*

General Firmware Commands

The following commands are supported by any CSS Core Firmware-based firmware, including the AS7058 firmware.

CMD_BASE_ID_APPL_NAME Command ID: 0x00

This command returns a human-readable string identifying the firmware.

Target ID

Reserved for future use. Always set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (Variable Length)

- **application_name** (string, variable-length): Name of the firmware (not NULL-terminated).

CMD_BASE_ID_VERSION
Command ID: 0x01

This command returns the firmware version.

Target ID

Reserved for future use. Always set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (Variable Length)

- **version_string** (string, variable-length): Version string of the application, for example "1.0.0" (not NULL-terminated).

CMD_BASE_ID_RESET
Command ID: 0x02

This command triggers the firmware to restart. Note that no response is sent when the command is processed successfully.

Target ID

Reserved for future use. Always set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (0 Bytes)

- *None*

CMD_BASE_ID_I2C_CONFIG
Command ID: 0x03

This command initializes the I2C interface. It is intended for internal use. Using this command is potentially unsafe.

Target ID

Selects the I2C interface (see Table 44).

Table 44: I2C Interfaces of the AS7058 Evaluation Hardware

Value	Description
0	Primary interface.
	I2C bus used by AS7058.
	I2C bus used by LIS2DH12 (accelerometer).

Input Payload (8 Bytes)

- **enable** (uint8, 1 Byte): 1 - interface will be enabled.
- **reserved** (Array of uint8, 3 Bytes): Reserved for future usage, set to zero.
- **frequency** (uint32, 4 Bytes): I2C frequency in Hertz.

Output Payload (0 Bytes)

- *None*

CMD_BASE_ID_I2C_XFER
Command ID: 0x04

This command transfers an I2C datagram. It is intended for internal use. Using this command is potentially unsafe. Deprecated since Core Firmware version 3.0.0. Use CMD_BASE_ID_I2C_XFER_16BIT instead.

Target ID

Selects the I2C interface (see Table 44).

Input Payload (2 - 257 Bytes)

- **address** (uint8, 1 Byte): I2C slave device address.
- **recv_size** (uint8, 1 Byte): Number of bytes that will be read after sending data.
- **send_data** (Array of uint8, 0 - 255 Bytes): Bytes which will be sent before data is read.

Output Payload (recv_size Bytes)

- **recv_data** (Array of uint8, recv_size Bytes): Content of the received data.

CMD_BASE_ID_SPI_CONFIG
Command ID: 0x05

This command initializes the SPI interface. It is intended for internal use. Using this command is potentially unsafe.

Target ID

Selects the SPI interface (see Table 45).

Table 45: SPI Interfaces of the AS7058 Evaluation Hardware

Value	Description
0	Primary interface.
1	Secondary interface.

Input Payload (8 Bytes)

- **enable** (uint8, 1 Byte): 1 - interface will be enabled.
- **mode** (uint8, 1 Byte): See Table 46.
- **first_bit** (uint8, 1 Byte): See Table 47.
- **reserved** (uint8, 1 Byte): Reserved for future usage, set to zero.
- **frequency** (uint32, 4 Bytes): SPI bus frequency in Hertz.

Table 46: SPI Modes

Value	Description
0	Phase: first edge, polarity: low.
1	Phase: first edge, polarity: high.
2	Phase: second edge, polarity: low.
3	Phase: second edge, polarity: high.

Table 47: SPI First Bit Declarations

Value	Description
0	Phase: first edge, Polarity: low.
1	Phase: first edge, Polarity: high.

Output Payload (0 Bytes)

- *None*

CMD_BASE_ID_SPI_XFER
Command ID: 0x06

This command transfers an SPI datagram. It is intended for internal use. Using this command is potentially unsafe.

Target ID

Selects the SPI interface (see Table 45).

Input Payload (1 - 65535 Bytes)

- **send_data** (Array of uint8, 1 - 65535 Bytes): Bytes which will be sent before data is read.

Output Payload (Size of send_data)

- **recv_data** (Array of uint8, Size of send_data): Content of received data.

CMD_BASE_ID_PIO_CONFIG Command ID: 0x07

This command initializes the PIO (Pin Input/Output) interface. It is intended for internal use. Using this command is potentially unsafe.

Target ID

Selects the pin (see Table 48).

Table 48: Pins of the AS7058 Evaluation Hardware

Value	Description
0	P0.11 (MODE Button).
1	P0.13 (Red LED).
2	P0.14 (Green LED).
3	P0.15 (Blue LED).
4	P0.26 (SDA).
5	P0.27 (SCL).
6	P1.01 (D2).
7	P1.02 (D3).
8	P1.08 (D4).
9	P1.10 (D5).
10	P1.11 (D6).
	AS7058 interrupt signal (active-high).
11	P1.12 (D7).
12	P1.03 (D8).
	LTC2950 (push button controller) kill signal (active-low).

Table 48 (Continued): Pins of the AS7058 Evaluation Hardware

Value	Description
13	P0.03 (A0).
	Hardware ID bit 0 (active-high).
14	P0.04 (A1).
	Hardware ID bit 1 (active-high).
15	P0.28 (A2).
	LTC2950 (push button controller) interrupt signal (active-low).
16	P0.29 (A3).
17	P0.30 (A4).
	LIS2DH12 (accelerometer) interrupt signal (active-high).
18	P0.31 (A5).
19	P1.15 (SCK).
20	P1.13 (MO).
21	P1.14 (MI).
22	P0.08 (RX).
	Hardware revision bit 0 (active-high).
23	P0.06 (TX).
	Hardware revision bit 1 (active-high).
24	P0.18 (RST).
25	P1.09 (CHG).
26	P0.12 (VUSB_DET).

Input Payload (3 Bytes)

- **enable** (uint8, 1 Byte): 1 - pin will be enabled.
- **mode** (uint8, 1 Byte): See Table 49.
- **pull** (uint8, 1 Byte): See Table 50.

Table 49: Pin Modes

Value	Description
0	Input pin, no interrupt enabled.
1	Input pin, interrupt on rising edge enabled.
2	Input pin, interrupt on falling edge enabled.

Table 49 (Continued): Pin Modes

Value	Description
3	Input pin, interrupt on both edges enabled.
4	Push-pull output pin.
5	Open drain ouput pin.

Table 50: Pull Resistor Configurations

Value	Description
0	Using no pull resistors.
1	Using pull-up resistor.
2	Using pull-down resistor.

Output Payload (0 Bytes)

- *None*

CMD_BASE_ID_PIO_XFER
Command ID: 0x08

This command sets or gets the state of a pin. It is intended for internal use. Using this command is potentially unsafe.

Target ID

Selects the pin (see Table 48).

Input Payload (0 - 1 Byte)

- **state** (Optional uint8, 0 - 1 Byte): New pin state (see Table 51).

Table 51: Pin States

Value	Description
0	Reset state (0).
1	Set state (1).
2	Toggle pin.

Output Payload (1 Byte)

- **state** (uint8, 1 Byte): Current pin state (see Table 51).

CMD_BASE_ID_PIO_STATE
Command ID: 0x09

This command is only used for asynchronous messages sent by the firmware when a pin interrupt occurs.

Target ID

Indicates which pin generated the event (see Table 48).

Input Payload (0 Bytes)

- *None*

Output Payload (1 Byte)

- **state** (uint8, 1 Byte): Current pin state. See Table 51.

CMD_BASE_ID_SYS_START_BL
Command ID: 0x0A

This command triggers the firmware to start the bootloader. Note that no response is sent when the command is processed successfully.

Target ID

Reserved for future use. Always set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (0 Bytes)

- *None*

CMD_BASE_ID_PWM_CONFIG
Command ID: 0x0B

This command initializes a PWM channel. It is intended for internal use. Using this command is potentially unsafe.

Target ID

Selects the PWM channel (see Table 52).

Table 52: PWM Channels of the AS7058 Evaluation Hardware

Value	Description
0	PWM channel 0.

Input Payload (8 Bytes)

- **enable** (uint8, 1 Byte): 1 - interface will be enabled.
- **reserved** (uint8, 1 Byte): Reserved for future usage, set to zero.
- **duty_cycle** (uint16, 2 Bytes): Duty cycle. 0 represents 0%, 1000 represents 100%.
- **frequency** (uint32, 4 Bytes): PWM frequency in Hertz.

Output Payload (0 Bytes)

- *None*

CMD_BASE_ID_TEST_REQ
Command ID: 0x0C

This command requests a stream of test messages. It is intended for internal use. Using this command is potentially unsafe.

Target ID

Reserved for future use. Always set it to zero.

Input Payload (8 Bytes)

- **count** (uint16, 2 Bytes): Number of test messages to send.
- **size** (uint16, 2 Bytes): Size of test message payload.
- **delay_us** (uint32, 4 Bytes): Minimum time between the transmission of two test messages in microseconds.

Output Payload (0 Bytes)

- *None*

CMD_BASE_ID_TEST_RSP
Command ID: 0x0D

This command is only used for asynchronous messages sent by the firmware when test message streaming is enabled via CMD_BASE_ID_TEST_REQ.

Target ID

Reserved for future use. It is currently always set to zero.

Output Payload (0 - 65535 Bytes)

- **counter** (Optional uint16, 0 - 2 Bytes): Number of remaining test messages, not including the current message. Field is only present if requested payload size is larger than the size of this field.
- **pattern** (Array of uint8, 0 - 65535 Bytes): All payload bytes not used by other fields contain their index in the payload (modulo 255), i.e. payload bytes at indices 2 and 258 both have the value 2.

CMD_BASE_ID_I2C_XFER_16BIT
Command ID: 0x0E

This command transfers an I2C datagram. It is intended for internal use. Using this command is potentially unsafe.

Target ID

Selects the I2C interface (see Table 44).

Input Payload (4 - 65539 Bytes)

- **address** (uint8, 1 Byte): I2C slave device address.
- **reserved** (uint8, 1 Byte): Reserved for future usage, set to zero.
- **recv_size** (uint16, 2 Byte): Number of bytes that will be read after sending data.
- **send_data** (Array of uint8, 0 - 65535 Bytes): Bytes which will be sent before data is read.

Output Payload (recv_size Bytes)

- **recv_data** (Array of uint8, recv_size Bytes): Content of received data.

CMD_BASE_ID_HW_REV
Command ID: 0x0F

This command returns the hardware revision.

Target ID

Reserved for future use. Always set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (Variable Length)

- **hw_rev** (string, variable-length): Hardware revision of the target (not NULL-terminated).

CMD_BASE_ID_HW_PLATFORM
Command ID: 0x10

This command returns the hardware platform used by this device.

Target ID

Reserved for future use. Always set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (2 Bytes)

- **type** (uint8, 1 Byte): Type of the hardware platform (see Table 53).
- **variant** (uint8, 1 Byte): A variant of the hardware platform. This field is platform-type-specific.

Table 53: Hardware Platform Types

Value	Description
0	Unknown hardware platform type.
1	Unicom (STM32) hardware platform type.
2	Nordic (nRF5) hardware platform type.

CMD_BASE_ID_ADC_CONFIG
Command ID: 0x11

This command initializes a specific ADC source. It is intended for internal use. Using this command is potentially unsafe.

Target ID

Selects the ADC source (see Table 54).

Table 54: ADC Sources of the AS7058 Evaluation Hardware

Value	Description
0	Battery monitoring input.
1	Board pin A0.
2	Board pin A1.
3	Board pin A2.

Table 54 (Continued): ADC Sources of the AS7058 Evaluation Hardware

Value	Description
4	Board pin A3.
5	Board pin A4.
6	Board pin A5.
7	Internal MCU power supply monitor.

Input Payload (16 Bytes)

- **enable** (uint8, 1 Byte): 1 - The source will be enabled, 0 - The source will be disabled.
- **reference** (uint8, 1 Byte): The analog reference to be used for the selected source (see Table 55).
- **reserved** (Array of uint8, 2 Bytes): Not used. Set to zero (padding bytes).
- **gain** (uint32, 4 Bytes): Analog input gain for the selected source (see Table 56).
- **acquisition_time** (uint32, 4 Bytes): Acquisition time for the selected source (see Table 57).
- **acquisition_period** (uint32, 4 Bytes): The interval for the periodic sampling of the source. In units of 1 ms. A value of 0 means a single conversion. Currently, the periodic sampling is not supported. The only valid value is 0. Any other value will result in an error.

Table 55: Nordic Platform Voltage Reference Settings

Value	Description
1	Vdd/4 reference.
2	Internally regulated 0.6V reference.

Table 56: Nordic Platform Input Gains

Value	Description
167	Gain of 1/6.
200	Gain of 1/5.
250	Gain of 1/4.
333	Gain of 1/3.
500	Gain of 1/2.
1000	Gain of 1.
2000	Gain of 2.
4000	Gain of 4.

Table 57: Nordic Platform Acquisition Times

Value	Description
3000	Acquisition time of 3000 ns.
5000	Acquisition time of 5000 ns.
10000	Acquisition time of 10000 ns.
15000	Acquisition time of 15000 ns.
20000	Acquisition time of 20000 ns.
40000	Acquisition time of 40000 ns.

Output Payload (0 Bytes)

- *None*

CMD_BASE_ID_ADC_CONVERT
Command ID: 0x12

This command triggers the conversion on the selected source and returns the conversion value. It is intended for internal use. Using this command is potentially unsafe.

Target ID

Selects the ADC source (see Table 54).

Input Payload (0 Bytes)

- *None*

Output Payload (4 Bytes)

- **value** (int32, 4 Bytes): The ADC conversion value for the selected source.

CMD_BASE_ID_SERIAL_NUMBER
Command ID: 0x13

This command returns the serial number of the hardware.

Target ID

Reserved for future use. Always set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (Variable Length)

- **serial number** (string, variable-length): Serial number (not NULL-terminated).

CMD_BASE_ID_MODEL_NUMBER
Command ID: 0x14

This command returns the model number of the device.

Target ID

Reserved for future use. Always set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (Variable Length)

- **model number** (string, variable-length): Model number (not NULL-terminated).

CMD_BASE_ID_CORE_FW_VERSION
Command ID: 0x15

This command returns the version of the Core Firmware this firmware is based on.

Target ID

Reserved for future use. Always set it to zero.

Input Payload (0 Bytes)

- *None*

Output Payload (Variable Length)

- **version_string** (string, variable-length): Version string of the underlying Core Firmware, for example "1.0.0" (not NULL-terminated).

Revision Information

Date	Comment
2023-03-28	Added description of the Respiration Rate bio application. Updated information when signal sample preprocessing is performed.
2023-02-17	Improved the description of AGC-related fields in the Raw Data application output.
2022-12-07	Restructured the document. Added description of the PD Offset Calibration measurement mode, the Streaming application, the signal sample preprocessing feature of the Application Manager, and AGC mode 1. Updated the description of command CMD_ID_VSC_CL_GET_MEAS_CONFIG. Corrected the description of command CMD_ID_VSC_START_MEASUREMENT and the BioZ measurement mode output. Renamed the Raw Data Streaming application to Raw Data and updated the maximum size of a Raw Data application output. Updated the output description of the SpO2 app.
2022-09-23	Updated the maximum size of a Raw Data Streaming application output. Updated BioZ application configuration and output. Updated commands related to special measurement modes (EDA Scaling, BioZ).
2022-07-28	Updated the maximum size of a Raw Data Streaming application output.
2022-07-22	Added descriptions of the bio applications Signal Range Detection, BioZ, and EDA. Updated commands related to special measurement modes (impedance scaling).
2022-05-20	Initial version.

This document is automatically generated when a new firmware is released. The table above is only updated when the contents of this document have been changed. Section, table, figure, and page numbers may change between document revisions. Corrections of typographical errors are not explicitly mentioned in the table above.