Summary Data Protocol and Structure

Table of Contents

[**Summary Data Structure**](#_h225crk432vr) **3**

[**Event Types**](#_koqox7v2h6lk) **4**

[ResBit Awake Event (0x0000)](#_z5xbks9y4dnh) 4

[Data](#_eaaeohmuape) 4

[Trigger Pull Event (0x0001)](#_wuc6swawfyyg) 4

[Data](#_hcaivay2u8dq) 4

[Tilt Angle Event (0x0002)](#_ttmakehcuts9) 5

[Data](#_34tavraqqpjz) 5

[**Summary Transfer Protocol**](#_yfd6c8h5eq3w) **6**

[Workflow](#_8yvxz0tgms6t) 6

[Packet Structure](#_d34smn18x3h4) 7

[Response Structure](#_xawsoybsk3v2) 7

[Response Types](#_ikohmk1vjdig) 7

[Resend Packets (0x00)](#_uhnmjeudyz5l) 7

# 

# Summary Data Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Size (Bytes) | Data Type |
| Event Id | The identifier for the event | 2 | uint16 |
| Timestamp | The timestamp of the summary event. Unix Timestamp. | 4 | Uint32\_t (Unix Timestamp) |
| Size | The total size of the summary data | 1 | uint8\_t |
| Data | A block of that that can be used for whatever the summary event needs. | 20 | uint8\_t[20] |

# Event Types

The existing Resbit event types are documented below.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | ID | Total Data Size (Bytes) |
| ResBit Awake | Resbit Awakes | 0x0000 | 4 |
| Trigger Pulls | Number of trigger pulls | 0x0001 | 4 |
| Tilt Angle | Tilt angle | 0x0002 | 16 |

## ResBit Awake Event (0x0000)

This event is generated every time a resbit device wakes up. It contains how long the device was awake

### Data

Timestamp: The time when the device wakes up

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Size (Bytes) | Data Type |
| Time Awake | The amount of time in seconds the resbit device was awake | 4 | uint32\_t |

## Trigger Pull Event (0x0001)

This event is generated each time the ResBit device wakes up. It keeps track of the number of trigger pulls during each wakeup event. A trigger pull is defined as the HALL sensor exceeding the “Trigger Threshold Value” parameter in the Resbit device.cfg file.

### Data

Timestamp: The time the resbit device wakes up

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Size (Bytes) | Data Type |
| Number of Trigger Pulls | The number of times the trigger was pulled | 4 | uint32\_t |

## Tilt Angle Event (0x0002)

This event is generated every time a resbit device wakes up. It keeps track of the start, stop, min, and max tilt angle values during each wakeup event.

### Data

Timestamp: The time when the device wakes up

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Size (Bytes) | Data Type |
| Start Tilt Angle | The tilt angle at the beginning of the wake event. | 4 | float (single) |
| Stop Tilt Angle | The tilt angle at the end of the wake event. | 4 | float (single) |
| Min Tilt Angle | The minimum tilt angle measured during the wake event. | 4 | float (single) |
| Max Tilt Angle | The maximum tilt angle measured during the wake event. | 4 | float (single) |

# 

# Summary Transfer Protocol

Summary data will be transferred using the [ResBit Summary Service](https://docs.google.com/document/d/1sl5BtJPyj-gyvp7jXOQXlmbpdRGLp4TyecbgroYff6c/edit#heading=h.rq35b236x1n0). Data will be sent in chunks with syncing communication between each chunk. Each chunk will be broken down into individual tagged packets that will be streamed to the client device through BLE characteristic notification. No syncing will occur between packet sending.

## Workflow

1. Client connects to device
2. Client reads “ResBit Serial Number” characteristic for the ResBit serial number from the “[ResBit Summary Service](https://docs.google.com/document/d/1sl5BtJPyj-gyvp7jXOQXlmbpdRGLp4TyecbgroYff6c/edit#heading=h.rq35b236x1n0)”
3. Client subscribes to the *“Data”* characteristic notifications
4. Client subscribes to the *“Transfering”* characteristic notifications
5. (Optional) Client subscribes to the *“Transfer Error”* characteristic notifications
6. (Optional) Client subscribes to the *“Transfer Summary Data”* characteristic notifications
7. Client writes a 1 into the *“Transfer Summary Data”* characteristic to begin data transfer
8. Client Begins Transfering all stored summary data
   1. *“Transfering”* now has a value of 1
   2. Device will send over a number of [packets](#_d34smn18x3h4) through the Data characteristic.
      1. Client will be given these values through the characteristic notification. Data from notification will need to be quickly read and cached to be reconstructed later.
   3. *“Transfering”* now has a value of 0
   4. After sending all the packets, the device will wait 2 seconds for a response written to the *“Ack/Nack”* characteristic and the “Response” characteristic.
      1. Client will check all cached packets to make sure all packets were received.
         1. This can be done by checking the Packet Index field on the packets. If packets 0-(X-1) are all cached all packets have been received
         2. The total number of packets can be determined by reading the “Total Packets” field on any of the packets
      2. If all packets have been received Client writes a 1 into *“Ack/Nack”*
         1. Device sends next chunk
      3. If some packets are missing Client writes a “Resend Packets” response with the missing packets into *“Response”* then writes a 2 into *“Ack/Nack”*
         1. Format for *“Response”* is documented [here](#_ikohmk1vjdig)
         2. Device will resend missing packets
            1. *“Transfering”* will be 1 while sending packets and 0 when done
            2. This will repeat until all packets are received.
   5. Repeats until all stored summary data is sent
9. “Transfer Summary Data” value will become 0

## Packet Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Size (Bytes) | Data Type |
| Total Packets | The total number of packets in the chunk | 1 | uint8\_t |
| Packet Index | The index of the packet relative to the start of the chunk | 1 | uint8\_t |
| Data | The data within the packet | 181 | uint8\_t[18] |

Notes:

[1] Any ‘dummy’ bytes at the end of the packet (i.e. if there are 11 bytes of summary data in the packet, we will have 7 dummy bytes) will be set to 0.

## Response Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Size (Bytes) | Data Type |
| Response Type | The response type. Identifies what kind of response is being returned. | 1 | uint8\_t |
| Response Data | The data of the response. Don’t have to write the entire array. | 19 | uint8\_t[19] |

### Response Types

#### Resend Packets (0x00)

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Description | Size (Bytes) | Data Type |
| Number of Packets | The number of packets that need to be resent | 1 | uint8\_t |
| Packet Indices | The indices of the packet to resend. Up to 18 packets. Don’t have to write the entire array. | 18 | uint8\_t[18] |

Example:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Response Packet** | | | | | | | | | | | | | | | | | | | |
| Field | Response Type | Response Data | | | | | | | | | | | | | | | | | | |
| Byte Value | 0 | 3 | 0 | 4 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Byte Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |

Legend:

|  |  |
| --- | --- |
|  | Response type: Resend Packets |
|  | Number of packets: 3 |
|  | Packet indices: 0, 4, 9 |
|  | Ignored |