ResBit I2C Coms

Interface Control Document(ICD)

[**Requirements**](#_dm2j90ntmj08) **4**

[**General Procedure**](#_2lnrrpnu0ivq) **5**

[**Message Structure**](#_6rpzsvri2p64) **5**

[Message](#_ts78at6k9oti) 5

[Message Response](#_qh653vdtsmqw) 5

[Message Response Codes](#_8h2sbiudza7) 6

[**Packet Structure**](#_l3p8xmcrn7rb) **6**

[Packet Header](#_dobni36tcqd9) 6

[Packet Body](#_h1j8fxtgis4f) 7

[Packet Response (from receiver)](#_cp9hqol2c9gn) 7

[Packet Response Codes](#_fe3smfs5ayj4) 7

[**Messages**](#_aizw9c9dxvi0) **8**

[ResBit MCU to BlueBits MCU](#_e18k08cl3nxs) 8

[ResBit Peripheral messages (0x0001 - 0x00FF)](#_fe771wi55nv2) 8

[Get Status Message - 0x0000](#_o31c2vwa468g) 8

[Set Power State - 0x0001](#_h99blrchrdwi) 8

[Message Data](#_39cc99fvehqj) 8

[Response Data](#_hy8fv0koe5sy) 8

[Power State Id’s](#_m6px5xhl8r81) 8

[Who Am I - 0x0002](#_mej4n5lqs5cc) 9

[Message Data](#_hdjd55jctgqj) 9

[Response Data](#_c1ad3pgfuj6y) 9

[Device Id’s](#_v7xay81vsmfr) 9

[BlueBitsMessages (0x0100 - 0x01FF)](#_ay6yn4ku1sgr) 9

[Summary Data - 0x0100](#_u8u9ltq5txyp) 9

[Message Data](#_3ufkan2opvs) 9

[Response Data](#_2iqdma5mcil7) 10

[Configuration Message - 0x0101](#_23yje7w979am) 10

[Message Data](#_6ss5one3zvyz) 10

[Response Data](#_o3usyeinb87l) 10

[Config Variables](#_k0ltktb9qarm) 10

[Set ResBit Info - 0x0102](#_vlgiiqj0h9ex) 10

[Message Data](#_ysgzj2fro5vt) 10

[Response Data](#_revegsw6dw5z) 11

[BlueBits MCU to ResBit MCU](#_2l43fz2tzovg) 11

# Requirements

1. Must support simultaneous Bi-direction communication
2. Must support some form of transmission error detection i.e. CRC or checksum
3. Must support message sizes of at least 512 B
4. Must be able to handle multiple message types with arbitrary data formats and lengths
5. Must be able to send multiple event types and multiple events in a single message
6. Must support being able to send configuration data over to the BlueBits device
7. Must support waking up the ResBit MCU
8. Must support waking up the BlueBits MCU

# General Procedure

The ResBit MCU and BlueBits MCU will be connected via an I2C interface (400 kHz), with the ResBit MCU as the master and the BlueBits MCU as a slave. They will also share a GPIO line configured as an input on the ResBit MCU and as an output on the BlueBits MCU. This GPIO line will allow the BlueBits MCU to indicate to the ResBit MCU that a message is pending, so that the ResBit MCU will know to initiate an I2C read from the BlueBits MCU. This GPIO line will be active-high.

I2C ack/nack behavior will function per the I2C protocol, and will serve to regulate the I2C bus itself. However, these ack/nacks will not indicate whether a packet was parsed successfully and/or understood on either end.

# Message Structure

Messages will begin with a message ID followed by the message data. The top bit of the message ID is reserved, and is used to indicate whether the message is a response. A ‘1’ in this bit position indicates that the message is a response to a previous message matching the message ID.

Message data length is not specified in the message itself, as data lengths are handled by the underlying packet layer. Message data length is capped at 510 bytes.

Inaction on the message sender’s side (after the start of message transmission and before message conclusion) for over *n* milliseconds shall result in a message timeout on the receiving side.

## Message

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **# bytes** |
| Message ID | 15-bit message identifier with response flag set to 0 | 2 |
| Message data | The data for the message | 0-510 |

## Message Response

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **# bytes** |
| Message ID | 15-bit message identifier with response flag set to 1 | 2 |
| Message response code | Code indicating message reception success or failure | 1 |
| Message response data | The data for the message response | 0-509 |

### Message Response Codes

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Value** |
| Success | Message received and parsed successfully | 0x00 |
| Id error | Message Id not recognized | 0x01 |
| Data error | Error occurred when parsing message contents | 0x02 |

# Packet Structure

Messages will be split into variable-length packets which are each preceded by a fixed-length packet header. This will allow the ResBit MCU, acting as the I2C master, to initiate a read of a packet header of a known length, followed by a read of a length specified in that packet header.

Every packet shall be followed by a packet response from the receiving side. Following a packet response indicating the previous packet was not received successfully, packet transfers shall be retried up to three times, after which the message sender shall allow the message to timeout.

## Packet Header

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **# bytes** |
| CRC | CRC for validating the packet using CRC-16-IBM | 2 |
| Number of Packets | The total number of packets in the message | 1 |
| Packet Number | The packet number out of the total packets (zero-based numbering) | 1 |
| Data Length | The length of the data (bytes) in the packet | 1 |

## Packet Body

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **# bytes** |
| Data | The data for the packet - needs to match the data length specified in the packet header | 0 - 250 |

## 

## Packet Response (from receiver)

If a message sender receives a packet response with an invalid CRC, the message shall be allowed to timeout.

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **# bytes** |
| CRC | CRC for validating the response using CRC-16-IBM | 2 |
| Packet response code | Code indicating packet reception success or failure | 1 |

### Packet Response Codes

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Value** |
| Success | Packet successfully received | 0x00 |
| CRC Error | CRC in packet header did not match CRC generated by receiver | 0x01 |
| Invalid Number of Packets | Invalid number of packets specified in packet header | 0x02 |
| Invalid Packet Number | Invalid packet number specified in packet header | 0x03 |
| Invalid Data Length | Invalid data length specified in packet header | 0x04 |
| Packet Order Error | Packet received out of order (per packet number in packet header) | 0x05 |
| Data Length Mismatch | Data length specified in packet header did not match data length in packet body | 0x06 |
| Unknown Error | Generic error case | 0x7F |

# 

# Messages

## ResBit MCU to BlueBits MCU

### ResBit Peripheral messages (0x0001 - 0x00FF)

#### Get Status Message - 0x0000

TBC

#### Set Power State - 0x0001

Sets the power state of the peripheral device.

##### Message Data

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **# bytes** |
| Power State Id | Power state identifier | 2 |

##### Response Data

None

##### Power State Id’s

|  |  |
| --- | --- |
| **Power State** | **Id** |
| Awake | 0x0000 |
| Sleep | 0x0001 |
| Protected | 0x0002 |

#### Who Am I - 0x0002

Returns a string containing device id, firmware version, and hardware version.

##### Message Data

None

##### Response Data

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **# bytes** |
| Device Id | Identifier associated with the device type | 2 |
| Hardware Version | Hardware version, split into 8-bit fields (i.e. 0x00 : 0x01 : 0x00 : 0x12) | 4 |
| Software Version | Software version, split into 8-bit fields (i.e. 0x00 : 0x01 : 0x02 : 0x12) | 4 |
| Firmware Version | Firmware version, split into 8-bit fields (i.e. 0x00 : 0x01 : 0x02 : 0x12) | 4 |

##### Device Id’s

|  |  |
| --- | --- |
| **Device** | **Id** |
| BlueBits | 0x0001 |

### BlueBitsMessages (0x0100 - 0x01FF)

#### Summary Data - 0x0100

Sends summary data to BlueBits device.

##### Message Data

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **# bytes** |
| Count | Number of summary events in the message | 2 |
| Summary Data | Summary events data, see [BlueBits Summary Data Transfer Protocol](https://drive.google.com/a/saec-kv.com/open?id=1wnK5hIlPSNUe6rMZpi219GBpDbDX4Hb6k_We5R6ezD8) for summary data structure | 0-508 |

##### Response Data

None

#### Configuration Message - 0x0101

Sends configuration data to BlueBits device.

##### Message Data

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **# bytes** |
| Config Id | Identifier associated with the config variable | 1 |
| Config Data | Data to be written to config variable | 1-4 |

##### Response Data

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **# bytes** |
| Config Response Code | Code indicating whether config variable has been updated or not. | 1 |

##### Config Variables

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Id** | **Data Description** | **# bytes** |
| Advert On Time | 0x00 | Seconds, represented as an unsigned integer | 2 |
| Advert Off Time Min | 0x01 | Seconds, represented as an unsigned integer | 2 |
| Advert Off Time Max | 0x02 | Seconds, represented as an unsigned integer | 2 |
| Always advertise | 0x03 | Boolean | 1 |
| Advert interval | 0x04 | Milliseconds, represented as an unsigned integer | 2 |

##### Config Response Codes

|  |  |  |
| --- | --- | --- |
| **Type** | **Description** | **Value** |
| Success | Config value updated successfully | 0x00 |
| Unknown Id | Config Id not recognized | 0x01 |
| Data length | Config data length does not match expected value | 0x02 |
| Data value | Config data value not within expected range | 0x03 |

# 

#### Set ResBit Info - 0x0102

Sends ResBit info to BlueBits device.

##### Message Data

|  |  |  |
| --- | --- | --- |
| **Field** | **Description** | **# bytes** |
| Hardware UUID | 96-bit hardware UUID | 12 |
| Hardware Version | Hardware version, split into 8-bit fields (i.e. 0x00 : 0x01 : 0x00 : 0x12) | 4 |
| Firmware Version | Firmware version, split into 8-bit fields (i.e. 0x00 : 0x01 : 0x02 : 0x12) | 4 |

##### Response Data

None

## BlueBits MCU to ResBit MCU

TBC