DIY Flow Bench

Serial API Specification

Revision 1: April 15, 2020 – Jason Estes

Table of Contents

[Serial Communication 3](#_Toc37869635)

[Bluetooth (SPP) 3](#_Toc37869636)

[Protocol Description 4](#_Toc37869637)

[API Reference 5](#_Toc37869638)

[V - Get Version 5](#_Toc37869639)

[C - Get Capabilities 5](#_Toc37869640)

[L - Perform Leak Test Calibration 5](#_Toc37869641)

[l - Perform Leak Test 7](#_Toc37869642)

[O - Flow Offset Calibration 7](#_Toc37869643)

[F - Get Measured Flow 8](#_Toc37869644)

# Serial Communication

## Bluetooth (SPP)

Serial communication between the DIY Flow Bench and a computer, tablet or cellphone is handled using a Serial to Bluetooth module (HC-06) connected to the Arduino TX and RX pins.

*(Insert diagram here to show how to wire the HC-06 to the Arduino)*

# Protocol Description

The DIY Flow Bench uses a simple command/response protocol. All commands begin with a single unsigned byte which is sent to the controller identifying what command is being requested. This byte, known as the Identifier, is always an ASCII character between 0x21 and 0x7E. Some commands consist of only the Identifier while others contain a data stream that is sent to the controller following the Identifier.

Every command received by the controller will initiate a response. Like the request, some responses will contain only a single unsigned byte while other will have a data stream that follows that unsigned byte. The first, and sometimes the only, unsigned byte of the response will contain the Identifier. However, the most significant bit will indicate success or failure of the command. If the most significant bit of the response is set indicating failure, no data stream will follow regardless of the command that was sent.

0x00 indicates success

0x80 indicates failure

When a date stream is present in either the command or the response, the last unsigned byte of the command or response will always be a checksum. That check sum is calculated by looping through each byte of the data stream, including the identifier, summing each byte with the previous result and mod the new result by 255. The final result is then appended as the last byte in the data stream. The following pseudo code demonstrates this algorithm.

In the following example, Data is an array of bytes with the following values:  
[ 0x56, 0x78, 0x9A, 0xBC ]

Byte Checksum = 0;

Foreach (Byte In Data)

Checksum = (Checksum + Byte) MOD 0xFF

Return Checksum

The resulting Checksum in this case would be 0x26.

# API Reference

## V - Get Version

The Get Version command is used to get the sketch version running on the connected Arduino.

Request

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘V’ – ASCII 0x56 | Unsigned Byte |

Response

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘V’ – ASCII 0x56 | Unsigned Byte |
| 2 | Major | Unsigned Byte |
| 3 | Minor | Unsigned Byte |
| 4 | Revision | Unsigned Byte |
| 5 | Checksum | Unsigned Byte |

## C - Get Capabilities

The Get Capabilities command is used to get the functionality that is supported by both the firmware and hardware installed and configured on the flow bench.

Request

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘C’ – ASCII 0x43 | Unsigned Byte |

Response

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘V’ – ASCII 0x43 | Unsigned Byte |
| 2 | Environment Byte - 1 | Unsigned Byte |
| 3 | Environment Byte - 2 | Unsigned Byte |
| 4 | Automation Byte - 1 | Unsigned Byte |
| 5 | Automation Byte - 2 | Unsigned Byte |
| 6 | Checksum | Unsigned Byte |

The table below shows the bits contained in byte #2 of the Get Capabilities response.

**Note:** For each bit in the mask, 0 indicates disabled and 1 indicates enabled.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **(MSB) 0** | **1** | **2** | **3** | **4** | **5** | **6** | **7 (LSB)** |
| Ambient Temperature | Barometric  Pressure | Velocity Probe | Reserved | Reserved | Reserved | Reserved | Reserved |

## 

The table below shows the bits contained in byte #3 of the Get Capabilities response.

**Note:** For each bit in the mask, 0 indicates disabled and 1 indicates enabled.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **(MSB) 0** | **1** | **2** | **3** | **4** | **5** | **6** | **7 (LSB)** |
| Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |

The table below shows the bits contained in byte #4 of the Get Capabilities response.

**Note:** For each bit in the mask, 0 indicates disabled and 1 indicates enabled.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **(MSB) 0** | **1** | **2** | **3** | **4** | **5** | **6** | **7 (LSB)** |
| Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |

The table below shows the bits contained in byte #5 of the Get Capabilities response.

**Note:** For each bit in the mask, 0 indicates disabled and 1 indicates enabled.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **(MSB) 0** | **1** | **2** | **3** | **4** | **5** | **6** | **7 (LSB)** |
| Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |

## L - Perform Leak Test Calibration

The Perform Leak Test Calibration command is used measure and store the value seen by the MAF sensor when the test port is sealed off completely and the vacuum system is running at full speed. This value will be stored by the system and can be used later to verify no leaks have developed anywhere in the system by running a [Leak Test](#_Perform_Leak_Test_1).

Request

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘L’ – ASCII 0x4C | Unsigned Byte |

Response

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘L’ – ASCII 0x4C | Unsigned Byte |
| 2 | Current Flow – High Nibble and Fractional Part | Unsigned Byte |
| 3 | Current Flow – Low Byte | Unsigned Byte |
| 4 | Checksum | Unsigned Byte |

## l - Perform Leak Test

The Perform Leak Test command should be used each day to verify no leaks have developed in the flow bench. This is done by sealing off the test port completely with the vacuum system running at full speed and comparing the MFA sensor reading to the value that was saved during the last [Leak Test Calibration](#_Perform_Leak_Test).

Request

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘l’ – ASCII 0x6C | Unsigned Byte |

Response

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘l’ – ASCII 0x6C | Unsigned Byte |
| 2 | Test Status | Unsigned Byte |
| 3 | Current Flow – High Nibble and Fractional Part | Unsigned Byte |
| 4 | Current Flow – Low Byte | Unsigned Byte |
| 5 | Checksum | Unsigned Byte |

The table below shows the bits contained in byte number 2 of the Perform Leak Test Response.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **(MSB) 0** | **1** | **2** | **3** | **4** | **5** | **6** | **7 (LSB)** |
| Pass (0) Fail (1) | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |

The table below shows the bits contained in byte number 3 of the Perform Leak Test Response.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **(MSB) 0** | **1** | **2** | **3** | **4** | **5** | **6** | **7 (LSB)** |
| Fractional Part (MSB) | Fractional Part | Fractional Part | Fractional Part (LSB) | Hole Part High Nibble (MSB) | Hole Part High Nibble | Hole Part High Nibble | Hole Part High Nibble (LSB) |

## O - Flow Offset Calibration

The Flow Offset Calibration command is used to calculate the difference between what the MAF sensor “thinks” its flowing vs what the bench is actually flowing. This offset is used when calculating the final CFM displayed by the flow bench.

Request

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘O’ – ASCII 0x4F | Unsigned Byte |

Response

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘O’ – ASCII 0x4F | Unsigned Byte |
| 2 | Offset – Decimal Part | Unsigned Byte |
| 3 | Offset – Fractional Part | Unsigned Byte |
| 4 | Checksum | Unsigned Byte |

## F - Get Measured Flow

The Get Measured Flow command is used to get the calculated flow, in CFM from the controller.

Request

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘F’ – ASCII 0x46 | Unsigned Byte |

Response

|  |  |  |
| --- | --- | --- |
| **Byte** | **Description** | **Data Type** |
| 1 | Character ‘F’ – ASCII 0x46 | Unsigned Byte |
| 2 | Current Flow – High Nibble and Fractional Part | Unsigned Byte |
| 3 | Current Flow – Low Byte | Unsigned Byte |
| 4 | Checksum | Unsigned Byte |

The table below shows the bits contained in byte number 2 of the Get State Response.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **(MSB) 0** | **1** | **2** | **3** | **4** | **5** | **6** | **7 (LSB)** |
| Fractional Part (MSB) | Fractional Part | Fractional Part | Fractional Part (LSB) | Hole Part High Nibble (MSB) | Hole Part High Nibble | Hole Part High Nibble | Hole Part High Nibble (LSB) |