# ABC-MEMS WiFi Interface

# Open Extensions

September 25 2017

# 1 Introduction

This document describes ABC-MEMS extensions to the DDCI protocol. It is designed as a simple master-slave protocol to perform simple operations on the ABC-MEMS instrument, separate from the core DDCI protocol.

## 1.1 Connection Management

The protocol description below does not include anything about connection management. It only defines the protocol once the connection has been established. Connection management aspects are not modified from the normal instrument behavior. That includes the following:

- Basic communications are based on TCP/IP, and use port No 50 000.
- The instrument is client and connects to a server, which must be listening for incoming connection requests. After the socket is established, the server will take the role of master (or host), and send commands to the instrument (the slave). The instrument will respond to host commands.
- All connection parameters stay the same:
    - Router/Access-Point parameters and security definitions
    - Server IP Address or Domain Name
    - Connection enable and schedule

Basically the instrument will attempt to connect at regular intervals, as defined by its WiFi setup. Once connected to a router, and allocated an IP address through DHCP, it will contact the server at the defined IP address or domain name, and try to open a socket on port 50 000. All of that is automated and occurs without any control input from the host or device. Once the socket is open, the instrument becomes a slave and waits for incoming transactions as defined below, to which it will respond.

## 1.2 Timeouts

The host must maintain communications with the device, otherwise the device will timeout and close the socket. The timeout deadline is 1 minute. That means that if the instrument does not receive at least one transaction per minute it will timeout and close the socket.

## 1.3 Retries

The instrument will try to connect (or reconnect if the connection has failed) at the specified interval. In order to restore a failed connection quickly, simply specify a short connection schedule (1 minute for instance).

# 2 Protocol

## 2.1 Endianness:

The protocol is little-endian.

## 2.2 Transactions

The protocol is based on transactions sent by the host and received by the instrument.

For each transaction:

- The host sends a command-block to the instrument. This 12-byte command-block defines the operation and various parameters (see below).

- If the transaction is a write, the host sends all the bytes to write to the instrument (if any), after the command block. The instrument then sends an acknowledge back to the host (1 byte at value 0x32).
- If the transaction is a read, the instrument sends back all the requested bytes to the host, according to the formats described below. In that case, the instrument does not return the Ack. The returned bytes indicate implicitly that the instrument accepted and responded to the command.

The transaction command-block is constructed as follows:

| Field | Size (bytes) | Usage |
|---|---|---|
| *TaskCode* | 4 | Defines the type of transaction (see below for list of Task Codes). It is transmitted LSB first. |
| *Address* | 4 | This is a byte address. It has different meanings depending on the transaction. It is transmitted LSB-first |
| *Length* | 4 | This is a byte length It has different meanings depending on the transaction. It is transmitted LSB-first |

**Table 1**

| Field | Size (bytes) | Value |
|---|---|---|
| *Ack* | 1 | 0x032 |

**Table 2**

### 2.3 Task-Codes

The following list of Task-Codes is defined at this point.

*Note: The task codes must be transmitted LSB-first. For instance for the Misc_Read Task-Code that is 052, then 0x6D, then 0x63, then 0x51.*

| Operation | Task-Code | Parameters |
|---|---|---|
| *Misc_Read* | 0x51636D52 | Reads data from the instrument. The specific entity is defined by the *Address* field (see below) |
| *Reset* | 0x51636D53 | Resets the instrument immediately and without further notice. Note that the reset will interrupt the TCP link. |
| *WiFi_Stop* | 0x51636D54 | This command stops WiFi processing and powers down the WiFi interface. It should be used once the WiFi connection is no longer needed.<br><br>If that command is not used, the WiFi module will stop on its own if no communications have been received from the host within 1 minute. It is preferable to stop the WiFi module |

| | | |
|---|---|---|
| | | because the connection can then be reinstated more quickly. Otherwise, even if the connection schedule is very short, the instrument will first wait for the 1 minute timeout before trying to reconnect. |
| *Record_Flash_Read* | 0x51636D55 | Reads 128 bytes from the record Flash. The read address is specified by the *Address* field.<br><br>The instrument returns the 128 bytes. |
| *Record_Flash_Erase* | 0x51636D56 | Erase one 64kB sector in the record Flash. The erase address is specified by the *Address* field.<br><br>The instrument returns an *Ack* after the erase has completed.<br><br>Note: *This operation will only work when the instrument is not recording.*<br><br>*Note: The erase operation also re-initializes the write pointer. So it is highly recommended to always erase all the Flash at once (erase all the sectors one by one), or none at all.* |
| *Misc_Write* | 0x51636D57 | Writes data to the instrument. The specific entity is defined by the *Address* field, and the data to write is defined by the *Length* field (see below).<br><br>The instrument returns an *Ack* after the write has completed. |

**Table 3**

### 2.4 Misc_Read Task-Code

The following describes the types of variables that can be read by the host, and how to interpret the instrument's response.

| Variable | Address | Size (Bytes) | Parameters |
|---|---|---|---|
| *IIF* | 0 | 128 | The IIF (*Instrument Identification Field*) is a group of bytes that identifies the instrument. The instrument responds 128 bytes to this Read transaction:<br><br>• *Model Name* – ASCII String – The first 4 bytes (U32) indicate length (N). Then the following N bytes are the ASCII characters. The length bytes are sent LSB-first.<br><br>• *FW Rev* – ASCII String – The first 4 bytes (U32) indicate length (N). Then the following N bytes are the ASCII characters. The length bytes are sent LSB-first<br><br>• *Serial Number* – ASCII String – The first 4 bytes (U32) indicate length (N). Then the following N bytes are the ASCII characters. The length bytes |

| | | | |
|---|---|---|---|
| | | | are sent LSB-first<br>• *Date of Birth* – U64 – This is a 64-bit unsigned value that represents a Universal Time Code (UTC). The UTC is the number of seconds elapsed since Jan 1st 1904 at 0h00. Values 0x0000000000000000 and 0xFFFFFFFFFFFFFFFF represent invalid values.<br>• *Unused bytes follow up to the 128th byte read*. |
| ICF | 1 | 128 | The ICF (Instrument Calibration Field) is a group of bytes that contains the instrument calibration and other parameters. The instrument responds 128 bytes to this Read transaction:<br>• *Date of Calibration* – U64 – This is a 64-bit unsigned value that represents a Universal Time Code (UTC). The UTC is the number of seconds elapsed since Jan 1st 1904 at 0h00. Values 0x0000000000000000 and 0xFFFFFFFFFFFFFFFF represent invalid values.<br>• *User ID* – ASCII String – The first 4 bytes (U32) indicate length (N). Then the following N bytes are the ASCII characters. The length bytes are sent LSB-first<br>• *Unused bytes follow up to the 128th byte read.* |
| IP-Address | 2 | 4 | The instrument responds 4 bytes that represent the IP address allocated to the instrument by the DHCP server. The bytes are sent LSB first. |
| *Temperature-Float ($T_{\circ C}$)* | 6 | 4 | $T_{\circ C}$ is a group of 4 bytes (Sgl format), representing the temperature of the instrument. |
| *BattVoltage-Float ($V_{Batt}$)* | 7 | 4 | $V_{Batt}$ is a group of 4 bytes (Sgl format), representing the battery voltage. |
| *Record_On_Off* | 8 | 1 | Record_On_Off is a single byte that indicates if the instrument is recording or not:<br>0 -> AutoRec Engaged – Not Recording<br>1 -> Not Recording<br>2 -> Standard Recording<br>3 -> Autorec Engaged - Recording |
| *UTC* | 9 | 8 | The Universal Time Code (UTC) is a 64-bit value that represents the number of seconds elapsed since Jan 1st 1904 at midnight (Greenwich Mean Time) |

| | | | |
|---|---|---|---|
| | | | The bytes are sent LSB-first |
| *RSSI* | 10 | 1 | RSSI is a single byte that indicates the RSSI of the ongoing WiFi connection. The RSSI is indicated in dBm, as a signed integer. The value is normally negative, with a value close to zero indicating a strong signal. |
| | | | |

**Table 4**

### 2.5 Misc_Write Task-Code

The following describes the types of variables that can be written by the host.

| Variable | Address | Size (Bytes) | Parameters |
|---|---|---|---|
| *Record_Start_Stop* | 8 | 4 | Start or Stop the recording. The contents of the *Length* field determines if the recording is started or stopped:<br><br>1 -> Start Recording<br><br>2 -> Start AutoRec<br><br>0 -> Stop |
| *UTC_Correction* | 9 | 4 | The contents of the *Length* field represent a correction to apply to the UTC. That correction is expressed in seconds.<br><br>The bytes are sent LSB-first.<br><br>*Note: While the instrument is recording, the UTC is used to time the recording. Modifying the UTC while a recording is in progress is not recommended.* |

**Table 5**