BEAM

BMeasure HTTP API Manual
Preliminary

Product	BMeasure-125i
API Manual Version	1.0
Hardware Version	1.4.0
Software version	1.9.0
Date	2022-03-29

1. Contents

Table of Contents

2. Introduction

The Beam BMeasure-125i unit is a flexible and powerful IoT system for data capture, data logging and control in the laboratory, industrial and remote sensing arenas. It is based around an 8 channel, fully differential, synchronous sampling, 24 bit ADC that can sample at speeds up to 128 ksps. Multiple units can be connected together to provide more synchronously sampled channels.

This document describes the network HTTP API library allowing programs to be written to control the operation of a BMeasure unit and acquire the data from it. This API provides a limited but simple to use access API for slow speed data. No special host libraries are used as simple HTTP/JSON ASCII protocol can be used. The API operates over the Ethernet or Wifi interfaces.

The BMeasure-lib API provides fuller access with high speed data acquisition. See the BMeasure-lib manual for more information on this. It is useful to read this document to see information on the underlying functionality provided by both API's.

3. BMeasure-125i Control API

The BMeasure-125 supports a few different API's. This section describes the HTTP/JSON-RPC API over the Ethernet or Wifi interfaces. With this API JSON formatted string requests are sent to the BMeasure's internal WEB server, via port 80 or 443 of its Network interface and JSON formatted string replies are sent back.

To use the API the URL http://<Host>/api or https://<Host>/api is used to send the JSON commands and fetch information and data. The following commands are provided and will operate with or without user/password authentication depending on the BMeasure's security setting. These commands match the BMeasure's native C++ API as described in the BMeasure-lib document and the Doxygen generated on-line API manuals.

Method	***Parameters***	***Reply***	***Notes***
getNodeInfo		NodeInfo, Error	Get the nodes info
getStatus		Status, Error	Get the nodes status
getInformation		Information, Error	Return detailed unit information
setMode	mode	Error	Set the unit to run (processing) or idle mode
getChannelConfig	channelNumber	ChannelConfig, Error	Return the given channel numbers configuration
getConfig		Config, Error	Return the current configuration
getMeasurementConfig		MeasureConfig, Error	Get the measurement configuration
setMeasurementConfig	MeasureConfig	Error	Set the measurement configuration
getDataProcessed	clear = True or False	Status, data	Return the RMS, Mean, PeakHigh and PeakLow values for all channels and their alarm states. Clear the averages and peaks for next measurement period if the clear parameter is True.
getDigital		Bits, Error	Return the state of the 8 digital IO bits
setDigital	Bits or Bitset	Error	Set the state of the 8 digital IO bits
setRelay	RelayNum, state	Error	Sets the numbered relay on or off.
getSwitch	switchNum	State, Error	Returns the given switches status
setAwgConfig	Save, awgConfig	Error	Set the AWG so the analogue output tracks the voltage measurement chanel.
getAlarms		Alarms, Error	Return alarm settings
setAlarms	Alarm level settings	Error	Set the alarm trigger levels

clearAlarms	Clear requested alarms	Error	Clear the selected alarms (0 - 5)

The JSON-RPC matches the JSON-RPC 2.0 API standard. See: https://www.jsonrpc.org/specification

The JSON requests are sent with the HTTP protocol. An example complete HTTP request, with JSON body is:

```
POST /api HTTP/1.1
Host: 192.168.201.46
Content-Type: application/json-rpc
Content-Length: 87

{"id": "b972f1d8-3ef4-429f-b5ca-88e42ef95cae", "method": "getStatus", "jsonrpc": "2.0"}
```

Note the use of the “POST” type to send the JSON request and the Content-Type header to specify the json-rpc content type.

Typically users will use an appropriate HTTP/JSON access library to communicate with the BMeasure-125i. As there are a large number of possible options here we provide examples of typical JSON raw string request and responses.

Some examples of specific API functions is given below. For more information seet the main BMeasure-lib API manual or the BMeasure-http-api-examples.

3.1. API setMode

The setMode method has a single parameter, the “mode” which can be set to “Idle” or “Run”. When in “Run” mode the BMeasure unit will be running measurements and responding to alarm conditions.

{"jsonrpc":"2.0", "id":0, "method":"setMode", "params":{"mode":"Run"}}

Response when ok:

{"jsonrpc":2.0,"id":0,"result":null}

Response when error:

{“jsonrpc”:”2.0”, “id”: 0, “error”:{“code”: -32601, “message”: “Unknown method”}}

3.2. API setAlarms

The setAlarms method has an array of alarm settings

```
{"jsonrpc":"2.0", "id":0, "method":"setAlarms",
        "params":{
                "alarms":[
                        {“channel”:0,”levelHigh”:22.0},
                        {“channel”:1,”levelHigh”:300.0},
                        ...
                ]
        }
}
```

Note that the “levelHigh” parameter should be a floating point number.

The response when ok:

{"jsonrpc":2.0,"id":0,"result":null}

Response when error:

{“jsonrpc”:”2.0”, “id”: 0, “error”:{“code”: -32601, “message”: “Unknown method”}}

3.3. API clearAlarms

The clearAlarms method has a single parameter, the “alarms” which is an array on integer values defining the channels whose alarms should be cleared.

{"jsonrpc":"2.0", "id":0, "method":"clearAlarms", "params":{"alarms":[0,1,2,3,4,5]}}

Response when ok:

{"jsonrpc":2.0,"id":0,"result":null}

Response when error:

{“jsonrpc”:”2.0”, “id”: 0, “error”:{“code”: -32601, “message”: “Unknown method”}}

3.4. API setAwgConfig

The setAwgConfig method has an array of AWG settings. This can be used to set the units AWG signal generator to provide a suitable signal. There are two main parameters that can be configured:

- “mode”: Set to “Sine” for sinusoidal waveforms. measurements.
- “amplitude”: Set for the peak amplitude of the signal.

The “save” parameter is a boolean to indicate if to save the configuration in the units non volatile memory configuration.

For example:

```
{"jsonrpc":"2.0", "id":0, "method":"setAwgConfig",
        "params":{
                "save":false,
                “awgConfig”:{
                        “mode”:”Sine”,
                        “amplitude”:1.0
                }
        }
}
```

Note that the “amplitude” and other such parameters should be floating point numbers.

Response when ok:

```
{"jsonrpc":2.0,"id":0,"result":null}
```

Response when error:

```
{“jsonrpc”:”2.0”, “id”: 0, “error”:{“code”: -32601, “message”: “Unknown method”}}
```

3.5. API getDataProcessed

The getData method has a single parameter, “clear”, which is a boolean. If “clear” is set to true the BMeasure will clear its ProcessedData state clearing the RMS, Mean, PeakHigh and PeakLow values for a new measurement. Otherwise these running averages and peak values are left alone.

As well as an array of values per channel, the response includes configuration and status information from the BMeasure unit.

```
{"jsonrpc":"2.0", "id":0, "method":"getDataProcessed", "params":{"clear": true}}
```

Response when ok:

```
{"jsonrpc":2.0,"id":0,"result":{
        "serialNumber":"12345",
        "name":"Test1",
        "location":"Module1",
        "status":0,
        "statusString":"Idle: Stopped",
        "time":"2020-01-28 13:36:28",
        “valid”:true,
        "data": [{
                'id': 'AIO0',
                'name': 'Device0',
                'alarm': False,
                'rms': 0.0003653912,
                'average': -0.0001852428,
                'peakHigh': 0.001338699,
                'peakLow': -0.001679246,
                'units': 'uA'
            },
            ...
        ]
        }
}
```

Response when error:

{“jsonrpc”:”2.0”, “id”: 0, “error”:{“code”: -32601, “message”: “Unknown method”}}

Note that the floating point values could have the string value “NAN” which indicates no measurement is available. In this case the “valid” parameter will be set to False.