# Batlab Python Script Quick Start Guide

# Introduction

The Lexcelon Batlab Python library has been designed for hobbyists and more advanced users who would like to incorporate the Batlab hardware into their own cell testing workflow and/or environment. It is not an expectation for users to have to create their own intricate scripts to accomplish tasks with the Batlab unit; when the full GUI is ready for release, the low level communication with Batlab memory registers will be accomplished 'behind-the-scenes.' In the meantime, the Lexcelon software team has created a Python utility script that not only acts as an example of what can be accomplished with the Batlab Python library, but also as a means for our Kickstarter backers to communicate with the Batlab unit prior to the release of the GUI.

It is important to note that the command-line environment is not as polished as the GUI planned for release in the new future. Its promotion now as an alternative stems from our understanding that Kickstarter backers who have waited patiently are anxious to start using their units. We believe this Quick Start Guide can make that happen!

# Hardware Limits

It is important for users to be knowledgeable of Batlab hardware limits prior to usage. The Batlab has been designed to be resilient to common failure modes that have been anticipated as having a chance of occurrence during typical device usage. However, it is important for users to acquaint themselves with functional limitations to ensure that the Batlab unit is being operated within its rated tolerances. Though the device has a great range of functionality, there are still limitations that must be considered:

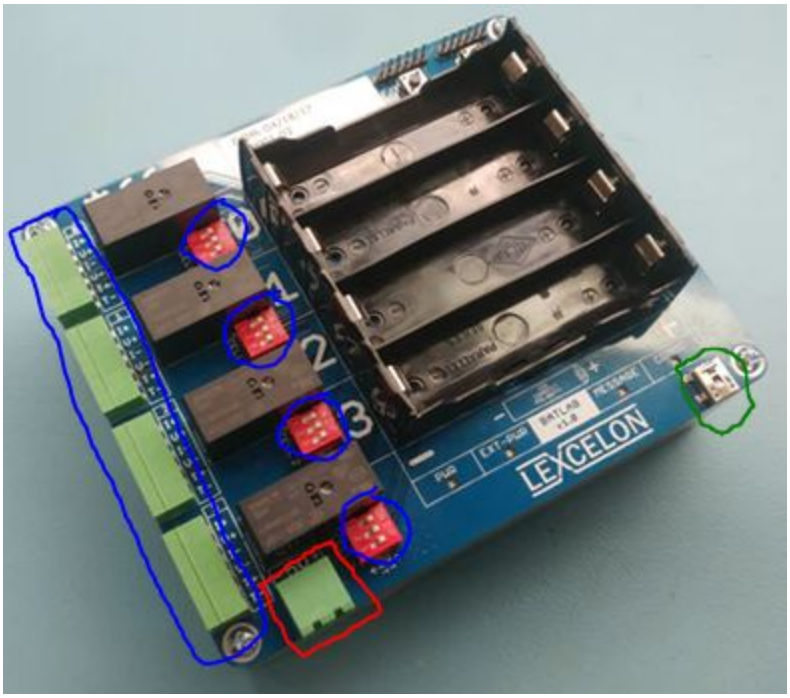
- The maximum current capable of being measured is 4.096A.
- The maximum current synthesis is 4A.
- Thermal dissipation limits impose a 4A per channel max threshold
    - NOTE: The maximum current dissipation the device can handle may differ from the technical specification limits of the cells being tested; users should review their cell data sheets to ensure that constructed tests do not breach the rated cell threshold.
- The maximum voltage capable of being measured is 4.5V.
- Charging operations of the device are designed for 4.75 - 5.20V from the external power supply.
- Batlab communication with the PC is designed for USB 4.95 - 5.05V (powered USB hub is recommended for usage cases with more than one Batlab).

# Basic Device Overview

The Lexcelon Batlab v1.0 is designed to collect voltage, current, temperature, and impedance measurements of up to 4 Lithium-ion cells simultaneously. The device is controlled by register transactions that are initiated by test commands passed to the Batlab through a USB connection from a host PC. The charging and discharging of cells can be controlled at constant rates or by using sinusoidal charge or discharge waveforms. Measurements are continuously taken by the device, and if a safety limit is reached, all current flow is stopped.

## Device Connections

The external Batlab connections are shown in the markup below:



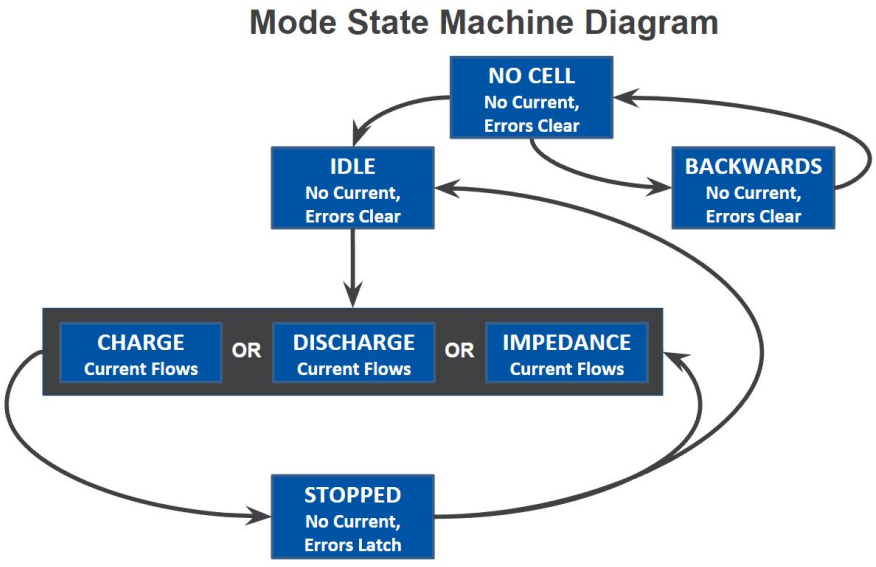

**BLUE:** These are external headers that may be used to connect the Batlab to a non-18650 cell slot tray. The red dip-switches must all be flipped to the off (downward) position to use the external headers instead of the 18650 form factor sized tray that is soldered to the Batlab.
**RED:** This port is for the external 5V power supply. Note that all charge and discharge functionalities require the 5V power supply to be connected.
**GREEN:** This port is for the communications protocol connection to the USB computer. The port is of Mini-B USB factor.

# Cell Channel LED Blink Patterns

A state machine internal to the Batlab firmware documents the MODE of each cell channel. An LED next to each cell channel of the cell tray is used to communicate information to the user about the instantaneous MODE of the cell by blinking in a different pattern for each MODE. A summary of the MODE LED blink patterns is provided in the table below:

| **Mode** | **LED Blink Pattern** |
|---|---|
| NO CELL | Off |
| IDLE | Blip |
| BACKWARDS | Fast Blink |
| CHARGE | Ramp Up |
| DISCHARGE | Ramp Down |
| IMPEDANCE | Sinusoid |
| STOPPED | Solid On |

A diagram of typical mode transitions is shown below. More information about the MODE and transitions can be obtained from reading the Programmer User's Manual (PUM), which is available on the Resources tab of our website: [https://www.lexcelon.com/resources/](https://www.lexcelon.com/resources/)



# Python Utility Script Prerequisites

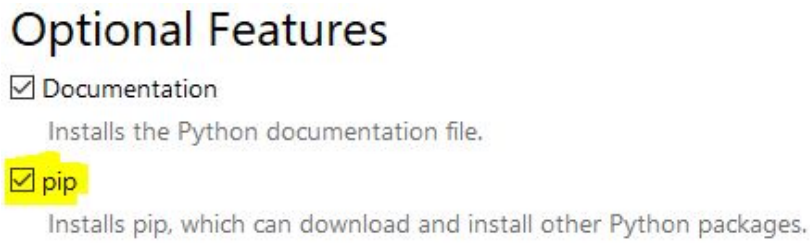
There are a couple prerequisites for using the Batlab utility script. We will address them here.

## Downloading Python 3

The main prerequisite for using the script is installing the Python programming language to your computer. The Batlab Python library is, at the moment, only compatible with Python 3. This means that you will need to ensure that the correct version of Python is installed before attempting to use the Batlab script. Python is free to download from the Python website:

https://www.python.org/downloads/


When stepping through the installation process for Python 3, be sure to have the 'pip' Optional Feature checked. This will simplify installing the Batlab package in the next prerequisite.




## Downloading and Installing the Batlab Python Package

The Batlab library and accompanying utility script are compiled into a Python package that can be acquired and installed using a simple 'pip install' command in a Command Prompt window. Once Python 3 is installed, open a Command Prompt window and execute the following command:

py -3 -m pip install batlab

```
C:\Users\Laeddis>py -3 -m pip install batlab
Requirement already satisfied: batlab in d:\software\lexcelon\python comm\batlab-software-python-master
Requirement already satisfied: pyserial in c:\users\laeddis\appdata\local\programs\python\python36-32\lib\site-packages (from batlab)
```

The ‘pip’ command will install the batlab package and another package called ‘pyserial’ that is leveraged within the utility script to manage the serial communication ports. Once both of these are installed, you will be able to execute the utility script from any directory on your machine by typing the following command in the Command Window:

batlabutil

NOTE: When you call the script using the command above, you will then enter into the command-line environment to communicate with your Batlab unit. Any measurement files that will be produced in the session will be saved in the directory in which you executed the script command. Additionally, the utility commands dependent on file inputs will search for the files within the directory in which you executed the script command. For this reason, we recommend creating a Batlab specific folder in which you may store your measurement and ‘Test Settings’ files. The ‘Test Settings’ file will be discussed in the next section.

It should also be noted that the Batlab Python package can be updated to the latest revision using the following command:

py -3 -m pip install batlab -U

## Downloading Default ‘Test Settings’ Template

When executing tests in the utility environment, one of the inputs is a ‘Test Settings’ file that stores user-provided test constraints. Test Settings are in the JSON file format and follow a particular structure that will be discussed in the Python Utility Script Usage Example section of this document. In the future, the Test Settings will be generated as part of user interaction with the software GUI. At this point, the Test Settings must be constructed manually. To assist this process, a Test Settings template is available from our Resources page:


Download the template and place it into a folder of your choosing. We recommend creating a Test Settings folder from which you will always execute the Batlab utility script (this ensures that the measurement output files are stored in a reasonable location on your system and that the Test Settings will be within the same directory that you are operating the command-line interface within).

NOTE: It is extremely important reference the table of minimum and maximum values that is provided in the Script Usage Example section of this document before changing any of the template values. The utility script does not force the parameters to be within the specified ranges, so it is up to the user to ensure that operation is within the limits.

## Download the Programmer User's Manual

One recommendation prior to proceeding to actual script usage is to have a copy of the Programmer User's Manual (PUM) on hand. The manual is available from our resources page:

https://www.lexcelon.com/resources/

This manual documents not only the communications protocol used by the Batlab, but also tabulates a summary of all registers available for read and/or write. See section 4.2.2.4 of the manual for information on the registers.

# Python Utility Script Execution

To execute the 'batlabutil.py' utility script, you simply run the 'batlabutil' command from a Command Window. However, we recommend that you execute the utility script from the location on your system that houses any Test Settings files that you may want to load during your usage session.

```
D:\Software\Lexcelon\Python Comm\batlab-software-python-master\batlab>batlabutil
```

The first output of the utility script is a list of commands available to you, their syntax, and a description of their function. If you need to reference this list later on deep into a session and do not want to scroll up, simply type 'help'.

```
Batlab Utility Script
General Commands
 list                                - list ports and SNs of connected batlabs
 active (port)                       - set currently active batlab. Or read value
 quit                                - Exit the program
 help                                - show this help
 constants                           - show the register and namespace names
Active Batlab Test Commands
 cycletest [cell] [cellname]         - Start cycle test using loaded settings
 dischargetest [cell] [cellname] (timeout)  - Start disch tst w/ loaded setngs
 load settings [settings filename] - Load in test settings from .json file
 view settings                       - display current test settings
Active Batlab Lower-Level Commands
 write [namespace] [addr] [data]    - General Purpose Command for Writing Regs
 read [namespace] [addr]            - General purpose Command for Reading Regs
    example: "read UNIT FIRMWARE_VER"
 info                                - return unit namespace information
 measure (cell)                      - return voltage,current,temp,charge
 setpoints (cell)                     - return setpoint information for cell
 impedance [cell]                    - take Z measurment
 charge [cell] (setpoint)            - begin charge on cell, optionally set I
 sinewave [cell] (setpoint)          - begin sine on cell, optionally set f in Hz
 discharge [cell] (setpoint)         - begin discharge on cell, optionally set I
 stop (cell)                         - set all cells to stop mode, or only 1 cell
 reset (cell)                        - set all cells to IDLE mode, or only 1 cell
 firmware load [firmware bin file] - load firmware update from local bin file
 firmware update                     - check for firmware update. Load if needed
 firmware check                      - check for firmware update.
No Batlab Currently Set As Active
```

The bracketed arguments next to the function call are arguments that are required for the command to work. The arguments that are in parenthesis are optional. The descriptions note what the optional inputs do. A list of constants that can be used in place of hexadecimal values for the [namespace] and [address] inputs is provided by typing ‘constants’.

```
>>>constants
NAMESPACE LIST: CELL0, CELL1, CELL2, CELL3, UNIT, BOOTLOADER, COMMS
CELL REGISTERS: MODE, ERROR, STATUS, CURRENT_SETPOINT, REPORT_INTERVAL
CELL REGISTERS: TEMPERATURE, CURRENT, VOLTAGE, CHARGEL, CHARGEH
CELL REGISTERS: VOLTAGE_LIMIT_CHG, VOLTAGE_LIMIT_DCHG, CURRENT_LIMIT_CHG
CELL REGISTERS: CURRENT_LIMIT_DCHG, TEMP_LIMIT_CHG, TEMP_LIMIT_DCHG
CELL REGISTERS: CURRENT_PP, VOLTAGE_PP
UNIT REGISTERS: SERIAL_NUM, DEVICE_ID, FIRMEWARE_VER, VCC, SINE_FREQ
UNIT REGISTERS: SETTINGS, SINE_OFFSET, SINE_MAGDIV, LED_MESSAGE, LOCK
COMM REGISTERS: LED0, LED1, LED2, LED3, PSU, PSU_VOLTAGE
```

For more information on the namespace and address architecture, you can reference the Programmer User's Manual.

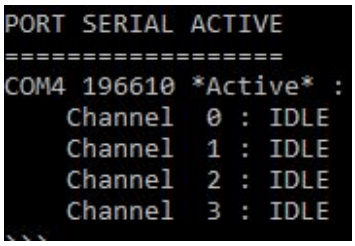
NOTE: While the utility script technically grants the ability to write to specific registers, we do not recommend or intend for users to do so unless they are working on the development of their own advanced scripts. If you are a user that plans to do this, please read the PUM in its entirety to develop a full understanding of the communications architecture.

# Python Utility Script Usage Example

This section documents how to set-up a simple cycletest using the Batlab utility script.

## Step 1: Connect Batlab

Before using the script to perform meaningful actions, one must first connect a Batlab unit to the host computer. When this is complete, you may type 'list' to confirm that the Batlab is connected. The script output will indicate which COM ID the unit has been assigned alongside its Serial Number and the current MODE of each cell channel.




## Step 2: Load Test Settings

All Test Settings are to be stored in JSON file format. In the future, you will be able to generate test setting files through the GUI. In the short-term, this process must be done manually. However, a 'default.JSON' file exists in the 'batlab' folder of the folder and may be leveraged as a starting point to create your own Test Settings files. An example of a Test Setting file layout is shown below. I have modified default.json to produce my own settings file BatlabDemo.json.

```
{
    "acceptableImpedanceThreshold": 0.2,
    "batlabToolkitGUIVersion": "0.0.1",
    "cellNames": [
        "Demo-Cell_0001",
        "Demo-Cell_0002",
        "Demo-Cell_0003",
        "Demo-Cell_0004"
    ],
    "cellPlaylistName": "BatlabDemoPlaylist",
    "chargeCurrentSafetyCutoff": 4,
    "chargeRate": 1.5,
    "chargeTemperatureCutoff": 45,
    "dischargeCurrentSafetyCutoff": 4,
    "dischargeRate": 2,
    "dischargeTemperatureCutoff": 45,
    "highVoltageCutoff": 4.2,
    "impedanceReportingPeriod": 300,
    "lowVoltageCutoff": 3.5,
    "numMeasurementCycles": 1,
    "numWarmupCycles": 0,
    "reportingPeriod": 1,
    "restPeriod": 30,
    "sineWaveFrequency": 39.0625,
    "sineWaveMagnitude": 0,
    "storageDischarge": true,
    "storageDischargeVoltage": 3.7
}
```


The “cellNames” variable is an array of the cell names and should have unique identifiers for each cell you are planning to test. In the example above, you can note that there are four separate elements of the “cellNames” array, which means that I plan to test four cells with these settings. The names provided as elements will be used in the output measurement file as the test is running.

The “cellPlaylistName” string value “BatlabDemoPlaylist” will be used in the measurement file name. The “batlabToolkitGUIVersion” will be output to the measurement file as a comment indicating what version of the GUI was used to generate the Test Settings file. You do not need to be concerned about changing the “batlabToolKitGUIVersion” parameter. The rest of the parameters correspond to specific test values and/or thresholds. In the future, the GUI will constrain these values. For now, you must be mindful of the expected minimum and maximum values assigned to them. The table below summarizes the parameters. **Any values outside the minimum and maximum range could result in hazardous operational conditions.**

| BATLAB TEST SETTING FILE PARAMETERS | | | | |
|---|---|---|---|---|
| Parameter | Description | Units | Min Value | Max Value |
| “acceptanceImpedanceThreshold” | To be used in the future as a means of determining which cells do not meet user’s standards. | Ohms | 0.02 | 200 |
| “batlabToolkitGUIVersion” | GUI Version used to generate the test measurements. | N/A | N/A | N/A |
| “cellNames” | Array of cell names. | N/A | N/A | N/A |
| “cellPlaylistNames” | Name of the ‘test playlist’ that will be used in the output measurement file name to denote the testing session. Meaningful name suggestion could be the name of project. | N/A | N/A | N/A |
| “chargeCurrentSafetyCutoff” | Cell channel exceedance of this current threshold during ‘charge’ will stop the test in that channel. | Amps | 0.25 | 4.096 |
| “chargeRate” | Current setpoint target while ‘charging.’ | Amps | 0.25 | 4.096 |
| “chargeTemperatureCutoff” | Cell channel exceedance of this temperature threshold during ‘charge’ will stop the test in that channel. | Deg C | -inf | inf |
| “dischargeCurrentSafetyCutoff” | Cell channel exceedance of this current threshold during ‘discharge’ will stop the test in that channel. | Amps | 0.25 | 4.096 |
| “dischargeRate” | Current setpoint target while ‘discharging.’ | Amps | 0.25 | 4.096 |
| “dischargeTemperatureCutoff” | Cell channel exceedance of this temperature threshold during ‘discharge’ will stop the test in that channel. | Deg C | 25 | 80 |
| “highVoltageCutoff” | Target voltage while charging. Signifies end of the ‘charge’ phase of cycle. | Volts | 3 | 4.4 |
| “impedanceReportingPeriod” | The time between periodic impedance measurements that occur while tests are running. | Sec | 10 | 3600 |

| | | | | |
|---|---|---|---|---|
| “lowVoltageCutoff” | Target voltage while discharging. Signifies end of the ‘discharge’ phase of cycle. | Volts | 2 | 3.6 |
| “numMeasurementCycles” | The number of cycles to be performed during the test with recorded measurement. | Cycles | 0 | 10000 |
| “numWarmpupCycles” | The number of non-measurement cycles to be performed before the actual measurement cycles begin. | Cycles | 0 | 100 |
| “reportingPeriod” | The time between measurement samples for each cell channel. | Sec | 0.5 | 3600 |
| “restPeriod” | The time between completion of cycle phase and beginning the next phase. | Sec | 0 | 3600 |
| “sineWaveFrequency” | Discrete frequency selection of the sine wave for impedance measurement. Only change value to multiple of 39.025. | Hz | 39.0625 | 1054.6875 |
| “sineWaveMagnitude” | Discrete selection of peak-to-peak sine wave magnitude. Only change value to 0 for 2App, 1 for 1App, 2 for 0.5App, or 3 for 0.25App. | App | 0 | 4 |
| “storageDischarge” | Boolean to signify whether a ‘storage discharge’ phase should occur after the completion of test. This permits users to discharge voltage to known state once testing is completed. | N/A | N/A | N/A |
| “storageDischargeVoltage” | The storage voltage value to which cells should be discharged during the ‘storage discharge’ phase (if applicable). | Volts | 2 | 4.4 |

When setting up a test, the first step is to load the safety parameters that will be used for the test. It is important to note that the Test Setting .json file must be in the same directory as the batlabutil script. Assuming that the BatlabDemo.json exists in the same directory as batlabutil, the settings load would be accomplished using the following command:

```
>>>load settings BatlabDemo.json
```

NOTE: The user MUST use the ‘load settings [settings filename]’ command for loading test parameter settings. This ensures that cells being tested as part of a test ‘playlist’ all use the same test settings. In most applications, cells MUST be tested with the same parameters for their results to be comparable in a meaningful way.

After this command, the utility will output the setting values so that you can visually confirm them to be correct for your intended test.

```
>>>load settings BatlabDemo.json
Currently Loaded Test Settings - batlab-log_BatlabDemoPlaylist.csv
============================================================
cellPlaylistName             : BatlabDemoPlaylist
chargeCurrentSafetyCutoff    : 4
chargeRate                   : 1.5
chargeTemperatureCutoff      : 45
dischargeCurrentSafetyCutoff : 4
dischargeRate                : 2
dischargeTemperatureCutoff   : 45
highVoltageCutoff            : 4.2
impedanceReportingPeriod     : 300
lowVoltageCutoff             : 3.5
numMeasurementCycles         : 1
numWarmupCycles              : 0
reportingPeriod              : 1
restPeriod                   : 30
sineWaveFrequency            : 39.0625
sineWaveMagnitude            : 0
storageDischarge             : True
storageDischargeVoltage      : 3.7
Results File will be written to testResults/ batlab-log_BatlabDemoPlaylist.csv
```

When you load a Test Setting file, it sets the global settings object to have the settings specified in the file. When you start a test on any cell, the cell channel object makes a copy of whatever the global settings object is at the time, and then uses those settings in the test. This means that a loaded Test Settings file will override the setpoints of each cell when you use an Active Test Command.

Loading a Test Setting file will also initiate the creation of a .csv file that leverages the “cellPlaylistName” parameter in its title. From the image above you will note that the name of the file is ‘batlab-log_BatlabDemoPlaylist.csv’. If we were to open this .csv file, we would first see a comment that is a mirror copy of the Test Settings file that we just loaded into the Batlab.

```
"{
    ""acceptableImpedanceThreshold"": 0.2,
    ""batlabToolkitGUIVersion"": ""0.0.1"",
    ""cellNames"": [
        ""Demo-Cell_0001"",
        ""Demo-Cell_0002"",
        ""Demo-Cell_0003"",
        ""Demo-Cell_0004""
    ],
    ""cellPlaylistName"": ""My Playlist"",
    ""chargeCurrentSafetyCutoff"": 4,
    ""chargeRate"": 1.5,
    ""chargeTemperatureCutoff"": 45,
    ""dischargeCurrentSafetyCutoff"": 4,
    ""dischargeRate"": 2,
    ""dischargeTemperatureCutoff"": 45,
    ""highVoltageCutoff"": 4.2,
    ""impedanceReportingPeriod"": 300,
    ""lowVoltageCutoff"": 3.5,
    ""numMeasurementCycles"": 1,
    ""numWarmupCycles"": 0,
    ""reportingPeriod"": 1,
    ""restPeriod"": 30,
    ""sineWaveFrequency"": 39.0625,
    ""sineWaveMagnitude"": 0,
    ""storageDischarge"": true,
    ""storageDischargeVoltage"": 3.7
}",,,,,,,,,,,,,,,,,
```


## Step 3: Initiate A Test

Assuming that our Batlab has 5V external power connected (the EXT-PWR LED shows solid green), we can now initiate a test. When you initiate a test, it is very important to use the same cell names in the argument that were defined in the Test Settings file. This is due to the fact that the measurement file uses the name provided in the argument in each row of recorded data for that cell channel. If you do not use a name that aligns with your Test Setting file names, you will potentially forget which cell you were testing. We recommended labeling cells ahead of time so that there will be no confusion. For example, if I have a cell that I have physically labeled Demo-Cell_0001, populated into the CELL0 slot, and plan to test with the loaded BatlabDemo.json Test Settings, I would now simply type:

```
>>>cycletest CELL0 Demo-Cell_0001
```

The Batlab fan will immediately kick on to cool the unit as the test is being conducted. The fan also runs for 1 minute following the completion of any test to return the Batlab to its base temperature. While the test is running, the .csv file shown in the previous image will be appended with measurements taken for CELL0 channel of the Batlab.

```
Cell Name,Batlab SN,Channel,Timestamp (s),Voltage (V),Current (A),Temperature (C),Impedance (Ohm),Energy (J),Charge (Coulombs),Test Stat
Demo-Cell_0001,196610,0,2017-10-29 20:25:13.172416,3.9434,1.6168,22.4133,,8.6993,2.2065,PRECHARGE,,,,,,
Demo-Cell_0001,196610,0,2017-10-29 20:25:14.213540,3.9451,1.6062,22.4133,,16.4956,4.1831,PRECHARGE,,,,,,
Demo-Cell_0001,196610,0,2017-10-29 20:25:15.252590,3.9459,1.5931,22.4065,,24.2343,6.1446,PRECHARGE,,,,,,
Demo-Cell_0001,196610,0,2017-10-29 20:25:16.297574,3.9459,1.5846,22.4202,,31.9170,8.0918,PRECHARGE,,,,,,
Demo-Cell_0001,196610,0,2017-10-29 20:25:17.339740,3.9468,1.5782,22.4202,,35.7455,9.0615,PRECHARGE,,,,,,
Demo-Cell_0001,196610,0,2017-10-29 20:25:18.380447,3.9471,1.5651,22.4065,,43.3489,10.9880,PRECHARGE,,,,,,
Demo-Cell_0001,196610,0,2017-10-29 20:25:19.420806,3.9468,1.5480,22.4065,,50.8769,12.8955,PRECHARGE,,,,,,
Demo-Cell_0001,196610,0,2017-10-29 20:25:20.460082,3.9468,1.5414,22.3997,,54.6156,13.8425,PRECHARGE,,,,,,
Demo-Cell_0001,196610,0,2017-10-29 20:25:21.501469,3.9468,1.5288,22.4065,,62.0448,15.7250,PRECHARGE,,,,,,
Demo-Cell_0001,196610,0,2017-10-29 20:25:22.543486,3.9467,1.5137,22.3997,,69.4019,17.5891,PRECHARGE,,,,,,
```

A header at the top of the file lists the column values and their respective units. If a test were running simultaneously on Demo-Cell_0002, then the rows might alternate or stagger between Demo-Cell_0001 and Demo-Cell_0002 data. If you want to stop a test midway through the test, you would simply type:

```
>>>stop CELL0
```

The CELL0 MODE would then transition to MODE_STOPPED and the test would complete:

```
>>>measure CELL0
CELL0: 3.8672 V 0.0000 A 72.1348 degF 0.0000 Coulombs MODE_STOPPED ERR_NONE
```

```
>>>Test Completed: Batlab 196610 , Channel 0
```

You may then reset the MODE to IDLE by typing:

```
>>>reset 0
>>>measure 0
CELL0: 3.8528 V 0.0000 A 71.9621 degF 1.1207 Coulombs MODE_IDLE ERR_NONE
```