Batlab v1.0 Programmer User's Manual: Rev A

1. Document Purpose **3**

2. Device Overview **3**

3. Firmware Overview **3**

3.1 Control Processor Firmware Overview 4

3.2 Communications Firmware Overview 5

4. Batlab Communication Protocol **5**

4.1 Batlab Packet Structure 6

Table 1: Batlab Command Packet Structure 6

Table 2: Batlab Response Packet Structure 7

Table 3: Batlab Stream Packet Structure 8

4.2 Batlab Registers 9

4.2.1 Register Scoping 9

Table 5: Register Scope Definitions 9

4.2.2 Registers Specification 9

4.2.2.1 0x00: Cell Mode Register 9

Table 6: Mode Register Structure 10

4.2.2.1.1 Mode State Machine 10

4.2.2.1.2 Mode LED Blink Patterns 11

Table 7: LED Blink Patterns 11

4.2.2.2 0x02: Cell Status Register 12

Table 7: Status Register Structure 12

4.2.2.3 0x06: Unit Settings Register 13

Table 8: Settings Register Structure 13

4.2.2.4 All Registers 13

Table 9: Cell Namespace Register Summary 14

Table 10: Unit Namespace Register Summary 16

Table 11: COMMS Namespace Register Summary 18

4.3 ADC Conversion to Physical Units 19

4.3.1 Voltage Conversion 19

4.3.2 Current Conversion 19

4.3.3 Temperature Conversion 19

4.3.4 VCC Conversion 21

4.3.5 Charge Conversion 21

5. Batlab Bootloader 21

5.1 Entering Bootloader Mode 21
5.2 Bootloader Communication Protocol 22
5.3 Bootloader Packet Structure 22
Table 12: Bootloader Command Packet Structure 22
Table 13: Bootloader Command Packet Structure 23
5.4 Bootloader Registers 24
Table 14: Bootloader Command Packet Structure 24
5.5 Application Software Bootloader Access 25
5.6 Loading New Bootloader Images 25

6. Software Development 26

6.1 Firmware Development 26
6.2 Batlab Software GUI Development 26
6.3 Python Development Scripts 26
6.4 Code Access 26

1. Document Purpose

The purpose of this document is to equip Batlab users who are interested in writing their own firmware with a comprehensive overview of Batlab code structure and protocol. The content contained herein is intended to act as a foundation for the development of safe and influential user-content that can be made available to other hobbyists. As hobbyists ourselves, we understand how important technical documentation can be to making new or improved versions of open source software. If this document does not include the technical details you are searching for, please let us know at lexceloncorp@gmail.com. Your feedback will be of benefit to the entire Batlab user base and can be incorporated into future revisions of this document.

2. 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 user-constructed 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.

3. Firmware Overview

The Batlab firmware can be understood as two distinct firmware sets. The first set is for the Communications Processor. The second set is for the Control Processor. The communications firmware acts as an intermediary between the host PC and the control processor. It is also responsible for controlling the Batlab LEDs and detecting whether an external power supply unit is present. The control processor utilizes the messages processed by the communications firmware to update a cell state machine internal to the Batlab and to take measurements accordingly.

3.1 Control Processor Firmware Overview

The control processor firmware consists of a main control loop and a parallel interrupt signal loop. A flow diagram of the firmware operation and control loops is depicted in Figure 3.1-1.

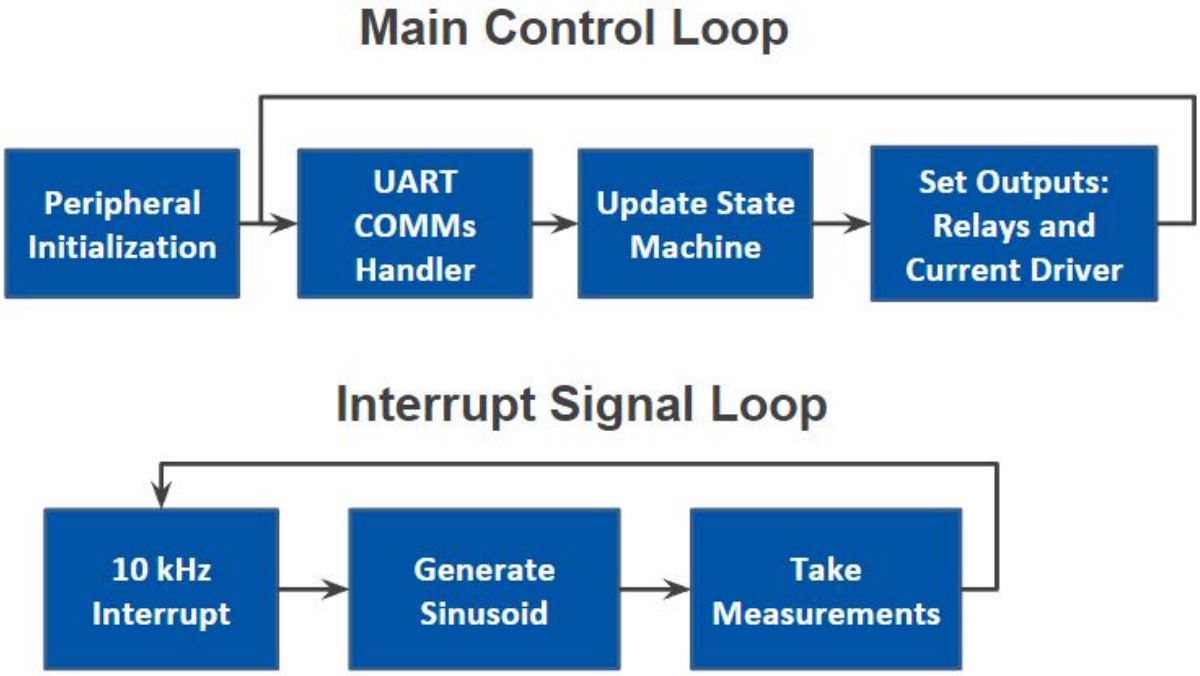

Figure 3.1-1 Firmware Operation State-Flow Diagram

When the control processor is powered up, it first initializes various subsystems such as General-purpose input/output (GPIO) port configuration and timer modules. After initialization, the firmware enters the main control loop that can achieve speeds of several Hertz. The first component of the loop performs a check of UART messages to determine whether any messages have been received or need to be sent. If any transmissions are required, the transfers are initiated at this time and carried out in the background. Then, if write-to-register messages were received, the registers are updated appropriately.

After processing of the UART messages has completed, information about each cell is read, and the state of each cell is re-computed based on the information reading and the current state of the cell (i.e. the operational MODE register). Once the cell states have been updated, the final function of the main control loop is to update all of the outputs. The outputs include the relays and the current driver for each cell. The loop has then reached its end and begins the procedure again.

An interrupt signal loop runs in parallel to the main control loop. A 10 kHz interrupt signal drives the generation of a sinusoid based on cell measurements in the event that the current driver needs to be fed from a generated sine wave.

3.2 Communications Firmware Overview

The communications processor firmware consists of a control loop as depicted in Figure 3.2-1. After peripheral initialization, the communications processor first handles the transmission and receival of USB messages. After this, the process handles the transmission and receival of UART messages. Next, the board LEDs are updated based on the setting read from the control register for LEDs. Finally, the USB data buffer is flushed.

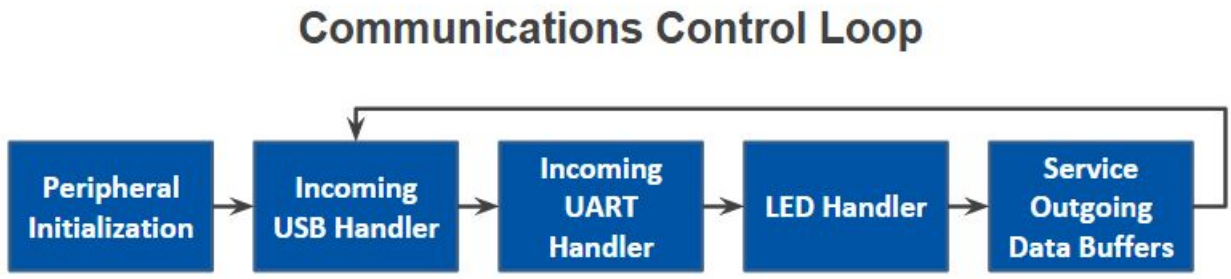

Figure 3.2-1 UART COMMs Handler State-Flow Diagram

In addition to these responsibilities, the communications firmware will occasionally read the voltage of the external PSU port to determine whether an external PSU is connected and has a voltage within the operational tolerance range. A register is updated by the communications firmware to document the external PSU status. The main processor will then query that register to set any fault status for the external PSU.

4. Batlab Communication Protocol

The Batlab uses a communication scheme of register transactions. The registers are used during test operations to store measurement data, cell states, internal Batlab unit calibrations, and more. Reads and writes are initiated to the registers by way of a packet communications protocol. The communication is divided among three types of packets: Command Packets, Response Packets, and Stream Packets. Command and Response Packets are identical in structure. Response packets are an ‘echo’ of a received Command Packet. The directional flow of packet transmission is depicted in Figure 4-1. The specific structure of each packet is defined in the subsequent sections of this document.

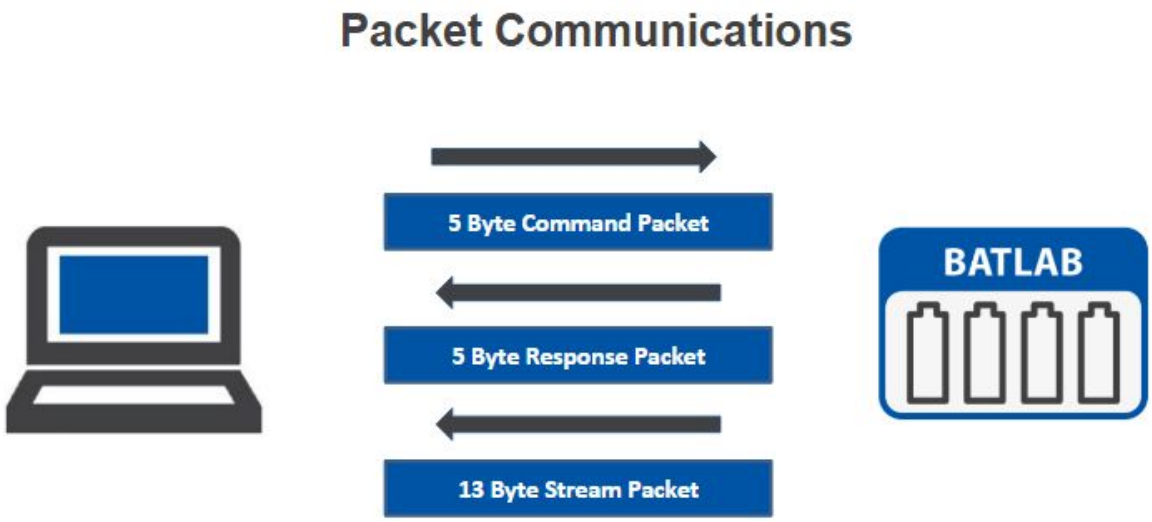

Figure 4-1. Packet Communications Diagram

4.1 Batlab Packet Structure

All packets sent to the Batlab from the PC are 5 Bytes long and are given the nomenclature ‘Commands.’ The byte values and purposes of a Batlab Command Packet are summarized in Table 1.

Table 1: Batlab Command Packet Structure

Byte	Potential Value(s)		Purpose
Byte 1 Packet Header	0xAA		Signifies the start of a command or response packet in the data stream.
Byte 2 8-bit Namespace Address	0x00		Signifies communication with cell 0.
	0x01		Signifies communication with cell 1.
	0x02		Signifies communication with cell 2.
	0x03		Signifies communication with cell 3.
	0x04		Signifies global communication with Batlab unit.
	0x05		Signifies communication with the Bootloader.
	0xFF		Special value reserved for communication with the COMMs processor.
Byte 3 8-bit Register Address	MSB	Bit 7	Top bit of address used for signifying either read (0) or write (1).
	Register Address	Bit 6 to Bit 0	Bits housing the specific register address (total of 128 possible registers)

Table 1: Batlab Command Packet Structure (Cont.)

Byte	Potential Value(s)	Purpose
Byte 4 and 5 Data Payload	Two Byte Data Payload	If the command is a 'write', Byte 4 and Byte 5 store the register 'write' data payloads in Little Endian format (LOW Byte then HIGH Byte).
	0x0000	This value signifies that the command is a 'read' command and contains no 'write' data.

For every packet that the Batlab receives from the PC, the Batlab will respond with a 5 byte packet. The packets are given the nomenclature 'Responses'. The byte values and purposes are summarized in Table 2.

Table 2: Batlab Response Packet Structure

Byte	Potential Value(s)	Purpose
Byte 1 Packet Header	0xAA	Signifies the start of a command or response packet in the data stream.
Byte 2 Command Packet Cell Namespace Address	Copy of Command Packet Cell Namespace Address	Byte 2 of the Response Packet houses a copy of the Namespace Address from the Command Packet that the Response Packet is responding to.
Byte 3 Command Packet 8-bit Register Address	Copy of Command Packet 8-bit Register Address	Byte 3 of the Response Packet houses a copy of the 8-bit Register Address from the Command Packet that the Response Packet is responding to.
Byte 4 and Byte 5 Data Payload	Two Byte Data Payload	If the response is to a 'read' command, Byte 4 and Byte 5 store the register 'read' data payloads in Little Endian format (LOW Byte then HIGH Byte).
	0x0000	This value signifies that the response is to a 'write' command and that the write was successful.
	0x0101	This value signifies that the response is to 'write' command and that the write was unsuccessful.

Additionally, the Batlab can be configured to send out ‘Steam Packets’ at regular intervals for every cell in the system during a test. The Stream Packets are 13 Bytes long and following the packet structure summarized in Table 3.

Table 3: Batlab Stream Packet Structure

Byte	Potential Value(s)	Purpose
Byte 1 Packet Header	0xAF	Signifies the start of a stream packet.
Byte 2 Cell Address	0x00	Signifies communication with cell 0.
	0x01	Signifies communication with cell 1.
	0x02	Signifies communication with cell 2.
	0x03	Signifies communication with cell 3.
Byte 3 Message Type	0x00	Signifies that the packet is a regular Stream Packet.
Byte 4 and Byte 5 REG_MODE Data	Two Byte Mode Data	Byte 4 and Byte 5 together comprise the Mode register data in Little Endian format. The mode describes the current state of the cell relative to the test state machine for the cell. Examples of this are whether the cell test is charging or discharging.
Byte 6 and Byte 7 REG_STATUS Data	Two Byte Status Data	Byte 6 and Byte 7 together comprise the Status register data in Little Endian format. The status describes the instantaneous standing of a cell relative to various threshold limits and fault indications. This data is a bit field of specific indicator flags.
Byte 8 and Byte 9 REG_TEMPERATURE Data	Two Byte Temperature Data	Byte 8 and Byte 9 together comprise the Temperature register data in Little Endian format.
Byte 10 and Byte 11 REG_CURRENT Data	Two Byte Current Data	Byte 10 and Byte 11 together comprise the Current register data in Little Endian format.
Byte 12 and Byte 13 REG_VOLTAGE Data	Two Byte Voltage Data	Byte 11 and Byte 12 together comprise the Voltage register data in Little Endian format.
NOTE: All register values in a stream packet are in direct alignment with the information available from a direct query to the register (no additional processing or structuring occurs at the time of Stream Packet compilation)		

4.2 Batlab Registers

Batlab Registers are internal to each Batlab connected to the host PC and are used to store cell, unit, and communications information.

4.2.1 Register Scoping

Each Batlab register has a scope. It is essential to understand the meaning of register scope before the purpose of each register can be understood in full. The definitions of each scope are summarized in Table 5.

Table 5: Register Scope Definitions

Register Scope Type	Namespace	Scope Definition
CELL	0x00 to 0x03	The register is addressable to a particular battery cell within a Batlab Unit.
UNIT	0x04	The register is addressable to the Batlab Unit.
COMMS	0xFF	The register is addressable to specific LEDs on the the Batlab Unit circuit board or the boolean ON/OFF state and voltage value of the external power supply unit powering the Batlab. The LEDs use blink patterns to communicate information about the cell modes to the user. The update of LEDs is determined as part of the communications loop internal to each Batlab Unit.
NOTE: While in Bootloader Mode (see Section 5), a write to a Unit or Cell scoped register will yield a non-response since the Batlab registers defined in this table are only applicable while operating in the ‘Application Software’.		

4.2.2 Registers Specification

All registers are 2 bytes wide and have 7 bit addresses. Thus, there are 128 possible registers. Registers 0x00-0x3F are reserved for Cell scoped registers. The following is an overview of the registers and their functions. Special attention is given to the Mode, Status, and Settings registers, as they serve an important role in every test.

4.2.2.1 0x00: Cell Mode Register

This register holds the ‘mode’ of the cell and controls the behavior of the Batlab. These modes are meant to be directly controlled by the user to start and stop tests on the cells. The output of the LEDS, state of the relays, and state of the current driver are dictated by these modes. A state machine internal to the Batlab controls the conditions that trigger the transition to new modes. The Batlab modes and their respective descriptions are summarized in Table 6.

Table 6: Mode Register Structure

Register Name	Scope	Type (R/W)	Address	Units	Register Description
MODE	Cell	R/W	0x00	Code	Sets the test mode for the cell.

Mode Values	Mode Descriptions
0x0000 MODE_NO_CELL	No battery is detected in the cell slot.
0x0001 MODE_BACKWARDS	A cell has been entered backwards into the cell slot.
0x0002 MODE_IDLE	A cell has been detected in the cell slot, but no test has started.
0x0003 MODE_CHARGE	A charge test has been initiated by the user. Current is permitted to flow. Stream Packets are sent at an interval defined by “Reporting Frequency” with mode, status, temperature, current, and voltage information. If any limit is exceeded while in charge mode, the state will immediately transition to ‘Stopped’.
0x0004 MODE_DISCHARGE	A discharge test has been initiated by the user. Current is permitted to flow, and Stream Packets with mode, status, temperature, current, and voltage information are sent at an interval defined by “Reporting Frequency”. If any limit is exceeded while in discharge mode, the state will immediately transition to ‘Stopped’.
0x0005 MODE_IMPEDANCE	An impedance test has been initiated by the user. Current is permitted to flow, and Stream Packets with mode, status, temperature, current, and voltage information are sent at an interval defined by “Reporting Frequency”. If any limit is exceeded while in impedance mode, the state will immediately transition to ‘Stopped’.
0x0006 MODE_STOPPED	A safety limit has been exceeded during a test and all current flow through the failed cell is inhibited.

4.2.2.1.1 Mode State Machine

All cells start in the NO_CELL mode. When a cell is connected, the cell mode transitions to IDLE. If a backwards cell is detected, the mode transitions to BACKWARDS. While in the NO_CELL, BACKWARDS, or IDLE modes, current is prevented from flowing through the cell and the ERROR register will always read 0x00.

The user initiates the start of a test by changing the mode to CHARGE, DISCHARGE, or IMPEDANCE. Once a test is running, the Batlab will individually monitor each connected cell for a variety of safety conditions such as overvoltage and overtemperature. If any of the condition thresholds are exceeded on a particular cell, the cell’s mode immediately transitions to ‘STOPPED’ and current ceases to flow. At the transition to ‘STOPPED’ mode, the safety

conditions in the STATUS register are latched into the ERROR register. The ERROR register is not cleared when the fault condition is cleared so that the user may query the ERROR register to diagnose the cause for the transition to the 'STOPPED' state. After the user reads the ERROR register, the user should set the cell mode back to IDLE to clear the ERROR register. A diagram of typical mode transitions is depicted in Figure 4.2.2.1.1-1.

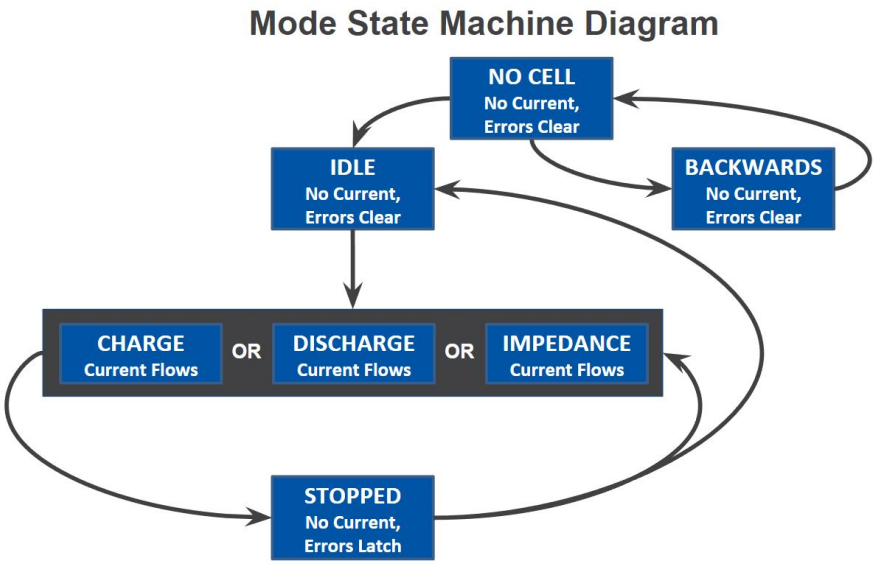

Figure 4.2.2.1.1-1 Typical Mode Transitions

4.2.2.1.2 Mode LED Blink Patterns

The Cell LEDs are illuminated in different patterns that each correspond to the various cell modes. The blink patterns are summarized in Table 7.

Table 7: LED Blink Patterns

Mode	LED Blink Pattern	LED Register Code
NO CELL	Off	0x0000
IDLE	Blip	0x0001
BACKWARDS	Fast Blink	0x0003
CHARGE	Ramp Up	0x0006
DISCHARGE	Ramp Down	0x0007
IMPEDANCE	Sinusoid	0x0008
STOPPED	Solid On	0x0004
n/a	Slow Blink	0x0002

4.2.2.2 0x02: Cell Status Register

The Status register is Cell scoped and provides information about the instantaneous status of a cell relative to system configurations and parameter limits. The information in the Status register is stored as a bit field. The Status register is summarized in Table 7.

Table 7: Status Register Structure

Register Name	Scope	Type (R/W)	Address	Units	Register Description
STATUS	Cell	R	0x02	Bit field	Provides instantaneous Status information about a given cell.

Status Values	Status Descriptions
0x0001 STAT_VOLTAGE_LIMIT_CHG	This flag is set when the cell reaches the high voltage limit during a charge test. Latched into the error register if it causes the mode to transition to ‘STOPPED’.
0x0002 STAT_VOLTAGE_LIMIT_DCHG	This flag is set when the cell reaches the low voltage limit during a discharge test. Latched into the error register if it causes the mode to transition to ‘STOPPED’.
0x0004 STAT_CURRENT_LIMIT_CHG	This flag is set when the cell reaches the high current limit during a charge test. Latched into the error register if it causes the mode to transition to ‘STOPPED’.
0x0008 STAT_CURRENT_LIMIT_DCHG	This flag is set when the cell reaches the high current limit during a discharge test. Latched into the error register if it causes the mode to transition to ‘STOPPED’.
0x0010 STAT_TEMP_LIMIT_CHG	This flag is set when the cell reaches the high temperature limit during a charge test. Latched into the error register if it causes the mode to transition to ‘STOPPED’.
0x0020 STAT_TEMP_LIMIT_DCHG	This flag is set when the cell reaches the high temperature limit during a discharge test. Latched into the error register if it causes the mode to transition to ‘STOPPED’.
0x0040 STAT_BACKWARDS	This flag is set when a backwards battery is detected in the cell slot. Cleared when the battery polarity is corrected or the cell is removed.
0x0080 STAT_NO_CELL	This flag indicates that no battery is connected to the cell slot.
0x0100 STAT_NO_PSU	This flag indicates either that no external power supply is connected to the Batlab or that the voltage of the external power supply is not set to the required Batlab rated voltage.
0x0200 STAT_NOT_INITIALIZED	This flag indicates whether peripheral initialization procedures are still ongoing for the cell.
0x0400 STAT_NOT_CALIBRATED	This flag indicates whether the cell is utilizing the measurement calibration offsets that are determined during assembly by the manufacturer to correct linear deviations in parameter measurements.

4.2.2.3 0x06: Unit Settings Register

The Settings register is Unit scoped and allows for configuration of system control configurations and advanced debugging. The Settings register entries are stored as Code. The Settings register is summarized in Table 8.

Table 8: Settings Register Structure

Register Name	Scope	Type (R/W)	Address	Units	Register Description
SETTINGS	Unit	R/W	0x06	Code	Provides information about system configurations and parameter limitations.

Settings Values	Settings Descriptions
0x0001 SET_TRIM_OUTPUT	First of two possible current control strategies. This setting enables a closed-loop control strategy to tune the current setpoint so that the actual current output closely matches the current setpoint. This is necessary because various factors like power supply voltage and temperature will affect the current output. In firmware version 1, this setting is OFF by default, but users are strongly recommended to turn it on.
0x0002 SET_VCC_COMPENSATION	Second of two possible current control strategies. This setting enables an open-loop strategy to tune the current setpoint based on VCC measurements (see SET_TRIM_OUTPUT). This open-loop control is not as effective as SET_TRIM_OUTPUT, so users are encouraged to use the SET_TRIM_OUTPUT control strategy instead.
0x4000 SET_SAFETY_DISABLE	Setting used for debugging. This disables all of the Batlab safety features. Users SHOULD NOT USE THIS setting unless familiar with what they are doing.
0x8000 SET_DEBUG	Setting used for debugging. This changes the output of some of the LEDs on the PCB with debug information. Users SHOULD NOT USE THIS setting unless familiar with what they are doing.

4.2.2.4 All Registers

A plethora of other registers exist for both internal use and external use during tests. A summary of all registers used within the Batlab for the Cell, Unit, and COMMS Namespaces is provided in Tables 9, 10, and 11, respectively. The tables include the default values for each register loaded on startup.

Table 9: Cell Namespace Register Summary

Cell Namespace - Namespace 0x00-0x03						
Register Name	**Scope**	**Type**	**Address**	**Default Value**	**Units**	**Description**
MODE	Cell	R/W	0x00	No Cell	Mode Codes	Current State machine for the cell. This register is used to start and stop tests.
ERROR	Cell	R	0x01	0x0000	Bit Flags	Describes which error caused 'STOPPED' mode to be entered.
STATUS	Cell	R	0x02	0x0000	Bit Flags	Describes real-time stat of the cell concerning safety limits.
CURRENT_SETPOINT	Cell	R/W	0x03	256 (2A)	X out of 640 where 128=1A (on 5V ref; 5V = 5A)	The charge and discharge current setpoint.
REPORT_INTERVAL	Cell	R/W	0x04	0 (None)	0.1 seconds	Stream Packets are sent at this interval (set to 0 for no stream packets).
TEMPERATURE	Cell	R	0x05	n/a	16bit signed ADC count @ 5V	Raw ADC count of thermistor voltage divider.
CURRENT	Cell	R	0x06	n/a	16bit signed ADC count @ 4.096V = 4.096A	Cell current measurement.
VOLTAGE	Cell	R	0x07	n/a	16bit signed ADC count @ 4.5V	Cell voltage measurement.
CHARGEL	Cell	R/W0	0x08	0x0000	Coulombs = (6 * (register / 2**15)) * 4.096 / 9.765625	Low 2 bytes of Charge (always positive, so you must clear it if you change current direction). Note that this register rolls over around 91 amp-hours.
CHARGEH	Cell	R/W0	0x09	0x0000	Coulombs = (6 * (register / 2**15)) * 4.096 / 9.765625	High 2 bytes of Charge.
VOLTAGE_LIMIT_CHG	Cell	R/W	0x0A	30584 (4.2V)	16bit signed ADC count @ 4.5V	High Voltage Limit
VOLTAGE_LIMIT_DCHG	Cell	R/W	0x0B	20389 (2.8V)	16bit signed ADC count @ 4.5V	Low Voltage Limit
CURRENT_LIMIT_CHG	Cell	R/W	0x0C	32000 (4A)	16bit signed ADC count @ 4.096V = 4.096A	High current limit in charge condition

Table 9: Cell Namespace Register Summary (Cont.)

Register Name	Scope	Type	Address	Default Value	Units	Description
CURRENT_LIMIT_DCHG	Cell	R/W	0x0D	32000 (4A)	16bit signed ADC count @ 4.096V = 4.096A	High current limit in discharge condition
TEMP_LIMIT_CHG	Cell	R/W	0x0E	25092 (45°C)*	16bit signed ADC count @ 4.5V	High temperature limit in charge condition.
TEMP_LIMIT_DCHG	Cell	R/W	0x0F	20825 (65°C)*	16bit signed ADC count @ 4.5V	High temperature limit in discharge condition.
DUTY	Cell	R	0x10	0x0000	X out of 640 where 128=1A (on 5V ref; 5V = 5A)	Debug register for reading the duty cycle value.
COMPENSATION	Cell	R	0x11	0x0000	X out of 640 where 128=1A (on 5V ref; 5V = 5A)	Debug register for reading current compensation.
CURRENT_PP	Cell	R	0x12	n/a	16bit signed ADC count @ 4.096V = 4.096A	Current peak-to-peak magnitude.
VOLTAGE_PP	Cell	R	0x13	n/a	16bit signed ADC count @ 4.5V	Voltage peak-to-peak magnitude.
CURRENT_CALIB_OFF	Cell	R/W	0x14	0x0000	16bit signed ADC count @ 4.096 = 4.096A	Current measurement calibration offset (as determined by manufacturer)
CURRENT_CALIB_SCA	Cell	R/W	0x15	0x4000	Unitless; 16384 is nominal	Current measurement calibration scale factor. This is a divisor for the current measurement.
TEMP_CALIB_R	Cell	R/W	0x16	0x05DC	Ohms, 1.5kΩ nominal (based on resistor value)	The calibrated resistance of the standard resistor in the thermistor network.
TEMP_CALIB_B	Cell	R/W	0x17	0x0D34	Kelvin, 3380 nominal	The calibrated “B” value obtained by measuring difference between 50C and 25C.
CURRENT_CALIB_PP	Cell	R/W	0x18	0x4000	Unitless; 16384 is nominal	AC current calibration scale factor. This value is a scale divisor for the AC current measurement.
VOLTAGE_CALIB_PP	Cell	R/W	0x19	0x4000	Unitless; 16384 is nominal	Voltage measurement calibration scale factor. This is a divisor for the voltage measurement.
CURR_CALIB_PP_OFF	Cell	R/W	0x1A	0x0000	16bit signed ADC count @ 4.096V = 4.096App	Peak-to-peak AC current measurement offset.
VOLT_CALIB_PP_OFF	Cell	R/W	0x1B	0x0000	16bit signed ADC count @ 4.5Vpp	Peak-to-peak AC voltage measurement offset.

Table 10: Unit Namespace Register Summary

Unit Namespace - Namespace 0x04						
Register Name	**Scope**	**Type**	**Address**	**Default Value**	**Units**	**Description**
SERIAL_NUM	Unit	R/W*	0x00	n/a	Number	Unique Serial number for the Batlab unit. NOTE: This register is only writable once.
DEVICE_ID	Unit	R/W*	0x01	n/a	Number	Unique Device ID for the Batlab unit. NOTE: This register is only writable once.
FIRMWARE_VER	Unit	R	0x02	n/a	Number	Unique firmware version identifier.
VCC	Unit	R	0x03	n/a	16bit signed ADC count of 4.096V measurement with 5V reference	System USB Bus Voltage
SINE_FREQ	Unit	R/W	0x04	1 (39.0625 Hz)	Units of (10000 / 256) Hz	Frequency of the sine wave for impedance measurement.
SYSTEM_TIMER	Unit	R	0x05	n/a	0.1 Hz	Counter for system timing
SETTINGS	Unit	R/W	0x06	0x0000	Boolean	Turn on and off settings. Bit 0 is enable_trim_output. Bit 1 is enable current compensation based on VCC.
SINE_OFFSET	Unit	R/W	0x07	16 (0.125A)	X out of 640 when 128=1A (on 5V ref; 5V = 5A)	Minimum for sine wave generation. NOTE: It is not recommended for users to change this register from its default value of 16 (0.125A). Doing so will result in current measurement discrepancies, since the calibration values are based on having a value of 16 in this register.
SINE_MAGDIV	Unit	R/W	0x08	2 (0.5App)	Unitless; divides 2 amp peak-to-peak magnitude by 2^value	Sets magnitude for impedance test. 4 amp peak-to-peak maximum. NOTE: The value of this register is meant to be one of the following four options: - 0 (2 App) - 1 (1 App) - 2 (0.5 App) - 3 (0.25 App) Values greater than three are not recommended, because they result in peak-to-peak current waveforms that are difficult to read and may result in loss of accuracy.

Table 10: Unit Namespace Register Summary (Cont.)

Register Name	Scope	Type	Address	Default Value	Units	Description
LED_MESSAGE	Unit	R/W	0x09	OFF	LED Code	Communicates to user which Batlab unit cells should be placed into. 0 = OFF 2 = FLASH_SLOW 3 = FLASH_FAST 4 = ON NOTE: The Message LED remains solid on while in Bootloader Mode. Additionally, the Message LED will 'fast' blink' if no serial number is loaded into the device (This should never occur in a delivered unit unless the user was developing for the Batlab and over-wrote the EEPROM).
BOOTLOAD	Unit	W	0x0A	0x0000	n/a	Transition to bootloader mode is initiated upon any write to this register.
VOLT_CH_CALIB_OFF	Unit	R/W	0x0B	0x0000	16bit signed ADC count @ 4.5V	Calibration offset for voltage measurement while in charge state
VOLT_CH_CALIB_SCA	Unit	R/W	0x0C	0x4000	Unitless; 16384 is nominal	Calibration scale factor for voltage measurement while in charge state. This is a divisor for Charge Voltage measurement.
VOLT_DC_CALIB_OFF	Unit	R/W	0x0D	0x0000	16bit signed ADC count @ 4.5V	Calibration offset for voltage measurement while in discharge state
VOLT_DC_CALIB_SCA	Unit	R/W	0x0E	0x4000	Unitless; 16384 is nominal	Calibration scale factor for voltage measurement while in discharge state. This is a divisor for Discharge Voltage measurement.
LOCK	Unit	R/W	0x0F	0x0000	Boolean	1 = Locked 0 = Unlocked When this register is in the locked state, interrupt driven measurements are frozen. This function may be used as a 'critical section' in order to read values that span multiple registers.

Table 11: COMMS Namespace Register Summary

COMMS Namespace - Namespace 0xFF						
Register Name	**Scope**	**Type**	**Address**	**Default Value**	**Units**	**Description**
LED0	COMMS	R/W	0x00	0 (OFF)	Code	LED for Cell Slot 0. Follows blink pattern as determined by the Mode of Cell 0.
LED1	COMMS	R/W	0x01	0 (OFF)	Code	LED for Cell Slot 1. Follows blink pattern as determined by the Mode of Cell 1.
LED2	COMMS	R/W	0x02	0 (OFF)	Code	LED for Cell Slot 2. Follows blink pattern as determined by the Mode of Cell 2.
LED3	COMMS	R/W	0x03	0 (OFF)	Code	LED for Cell Slot 3. Follows blink pattern as determined by the Mode of Cell 3.
EXTERNAL_PSU	COMMS	R	0x04	0x0000 (NO_PSU)	Boolean	Boolean indication of whether External Power supply voltage is within rated needs of the Batlab unit.
EXTERNAL_PSU_VOLTAGE	COMMS	R	0x05	n/a	10 Bit ADC Voltage @ ½ External PSU Voltage	Voltage of External Power supply.
EXTERNAL_PSU_CUTOFF_LOW	COMMS	R/W	0x06	511 (4.3V)	10 Bit ADC Voltage @ ½ External PSU Voltage	Detection voltage threshold for External Power Supply
EXTERNAL_PSU_CUTOFF_HIGH	COMMS	R/W	0x07	612 (5.15V)	10 Bit ADC Voltage @ ½ External PSU Voltage	Upper detection voltage threshold for External Power Supply
EXTERNAL_PSU_CUTOFF_HYST	COMMS	R/W	0x08	5 (50mV)	10 Bit ADC Voltage @ ½ External PSU Voltage	Hysteresis voltage level for External Power Supply detection

4.3 ADC Conversion to Physical Units

Each measurement from the Batlab is read from its respective Batlab register. However, care must be taken to convert the raw ADC values of the registers into a measurement with human-readable units. This section defines the conversion equations for both main test measurements and other registers storing data in ADC format.

4.3.1 Voltage Conversion

The voltage register is reported as a signed 16-bit 2's complement number, where the ADC measures voltage on a 4.5 volt reference.

$$V = \frac{V_{ADC} \, x \, 4.5}{2^{15}-1}$$

V = Voltage in Volts
V_{ADC} = Raw voltage register reading interpreted as a signed 16-bit number.

4.3.2 Current Conversion

The current register is reported as a signed 16-bit 2's complement number, where the ADC measures voltage on a 4.096 volt reference, and where 1V = 1A.

$$I = \frac{I_{ADC} \, x \, 4.096}{2^{15}-1}$$

I = Current in Amps
I_{ADC} = Raw current register reading interpreted as a signed 16-bit number.

4.3.3 Temperature Conversion

The temperature reading is slightly more complicated to convert than the voltage and current measurements. The value reported in the temperature register is the raw ADC measurement of a thermistor divider voltage. The temperature conversion follows from an exponential relationship with the measured resistance of the thermistor, so it is first necessary to convert the register value into a resistance.

NOTE: One of the calibration constants is the true resistance of the other resistor in the divider. This value can be read upon first connecting to the Batlab and then be used for all subsequent temperature measurements. From a design perspective, the existence of calibration constants led to the need for the user to bear the responsibility of programming the temperature safety limits using the Batlab software. The floating point math necessary to convert the temperature

threshold provided by the user is much better suited for a PC than the 8-bit PIC controller on the Batlab.

$$R_{therm} = \frac{R_{div}}{\frac{2^{15}-1}{T_{ADC}} - 1}$$

R_{div} = REG_TEMP_CALIB_R = 1500 Ohms (Nominally)
T_{ADC} = Raw temperature register reading interpreted as a signed 16-bit number.

The temperature may then be calculated using both the R_{therm} term and another calibration constant, B, which is unitless and appears in the temperature equation and in the thermistor datasheet.

$$T_{inv} = \frac{1}{T_o} + \frac{ln(\frac{R_{therm}}{R_o})}{B}$$

$$T = \frac{1}{T_{inv}} - 273.15$$

B = REG_TEMP_CALIB_B = 3380 (Nominally)
T_o = 25 °C + 273.15 = Ambient Temperature in Kelvin
Ro = 10000 Ohms = Data Sheet Specced Thermistor Resistance at T_o
T = Temperature in degrees Celsius
F = (T x 1.8) + 32 = Temperature in degrees Fahrenheit

4.3.4 VCC Conversion

The VCC register reports the voltage a measurement of the 4.096V reference voltage with VCC set as the ADC reference. The register is reported as a signed 16-bit 2's complement number, where the ADC measures voltage on a 4.096 volt reference.

$$VCC = \frac{4.096\ x\ (2^{15}-1)}{VCC_{ADC}}$$

VCC = VCC in Volts
VCC_{ADC} = Raw VCC register reading interpreted as a signed 16-bit number.

4.3.5 Charge Conversion

Charge is reported in a 32 bit register (or rather two, 16 bit registers which must be concatenated).

$$C = (6\ x\ \frac{C_{REG}}{2^{15}})\ x\ (\frac{4.096}{9.765625})$$

C_{REG} = 32 bit register value
C = Charge in Coulombs (may be converted to amp-hours by dividing by 3600)

5. Batlab Bootloader

The Bootloader is a special piece of code used for loading the operating system when the Batlab is powered on. The Batlab Bootloader takes up program memory 0x0000-0x03FF. Whenever the Programmable Interface Controller (PIC) boots up, it begins at address 0x0000, which is the start of the bootloader code.

5.1 Entering Bootloader Mode

Upon startup, the bootloader pauses a few milliseconds before using a set of criteria to determine whether or not it should enter either the ‘Bootloading Mode,’ or ‘Application Software.’ The Batlab Bootloader shall enter the ‘Bootloading Mode,’ when **any** of the following conditions are met:

- An EEPROM flag for entering ‘Bootloading Mode,’ is set
- The Batlab ‘Reset’ Button is actively held at the time of the few millisecond expiration
- The checksum on the currently written image cannot be successfully verified

In all other cases, the program counter is set at 0x0400 (the start of ‘Application Software,’ code). The main application software starts at address 0x0400 and ends at 0x3FFF. Thus, the main application image is exactly 15360 bytes.

5.2 Bootloader Communication Protocol

After entry into Bootloader mode, the EEPROM flag for entering the mode is cleared so that the Batlab will boot back into the main application software in the event that a loss of power occurs before any part of the program is erased. Following this clearance, the Bootloader waits for a command to be received over UART (commands are sent from the host PC over USB but are

routed to the PIC over UART). While in Bootloader Mode, the ‘Message LED,’ is solid (no blinks).

5.3 Bootloader Packet Structure

Commands sent to and Responses sent from the Bootloader follow the packet structure identical to those of the Batlab Communication Protocol outlined in Section 4. The byte values and purposes of a Bootloader Command Packet are summarized in Table 12.

Table 12: Bootloader Command Packet Structure

Byte	Potential Value(s)		Purpose
Byte 1 Packet Header	0xAA		Signifies the start of a command or response packet in the data stream.
Byte 2 8-bit Namespace Address	0x05		Signifies communication with the Bootloader.
Byte 3 8-bit Register Address	MSB	Bit 7	Top bit of address used for signifying either read (0) or write (1).
	Register Address	Bit 6 to Bit 0	Bits housing the specific register address (total of 128 possible registers)
Byte 4 and 5 Data Payload	Two Byte Data Payload		If the command is a ‘write’, Byte 4 and Byte 5 store the register ‘write’ data payloads in Little Endian format (LOW Byte then HIGH Byte).
	0x0000		This value signifies that the command is a ‘read’ command and contains no ‘write’ data.

For every packet that the Batlab Bootloader receives, the Bootloader will respond with a 5 byte packet. The packets are given the nomenclature ‘Responses’. The byte values and purposes are summarized in Table 13.

Table 13: Bootloader Command Packet Structure

Byte	Potential Value(s)	Purpose
Byte 1 Packet Header	0xAA	Signifies the start of a command or response packet in the data stream.
Byte 2 Command Packet Cell Namespace Address	Copy of Command Packet Cell Namespace Address	Byte 2 of the Response Packet houses a copy of the Namespace Address from the Command Packet that the Response Packet is responding to.
Byte 3 Command Packet 8-bit Register Address	Copy of Command Packet 8-bit Register Address	Byte 3 of the Response Packet houses a copy of the 8-bit Register Address from the Command Packet that the Response Packet is responding to.
Byte 4 and Byte 5 Data Payload	Two Byte Data Payload	If the response is to a ‘write’ command, Byte 4 and Byte 5 store the register ‘write’ data payloads in

		Little Endian format (LOW Byte then HIGH Byte).
	0x0000	This value signifies that the response is to ‘read’ a register and that the read was successful.
	0x0101	This value signifies that the response is to ‘read’ a register and that the read was unsuccessful.

5.4 Bootloader Registers

There are only three Batlab Bootloader registers. A summary of the Bootloader registers is provided in Table 14.

Table 14: Bootloader Command Packet Structure

Bootloader Namespace - Namespace 0x05						
Register Name	**Scope**	**Type**	**Address**	**Default Value**	**Units**	**Description**
REG_BOOTLOAD	BOOTLOADER	W	0x00	n/a	Code	Any write to this register will first check to see if the image is valid by doing a checksum. The checksum is the single byte sum of 0x0400-0x3ffe. That value must be equal to the checksum computed at compile time, which is placed at address 0x3fff.

REG_ADDR	BOOTLOADER	R/W	0x01	0x0400	Code	A write to this register sets the pointer location for writes to program memory. Valid values for this register are 0x0400 thru 0x3FFF.
REG_DATA	BOOTLOADER	R/W	0x02	0x0000	Code	A write to this register writes the low byte of the command data into the location in program memory pointed to by REG_ADDR. NOTE: Upon your first write to REG_DATA, the entire application will be erased! This occurs because FLASH memory must be erased before you can write to it. The design handles this by erasing the whole image on the first data write. No more erasing will be required for subsequent writes.

5.5 Application Software Bootloader Access

The Bootloader Mode may be accessed from the main application software by writing 0x0000 to the BOOTLOAD register in the unit space. Following the Batlab Communication Protocol defined in Chapter 4, the command would be structured as follows:

Byte 1: 0xAA
Byte 2: 0x04
Byte 3: 0x8A
Byte 4: 0x00
Byte 5: 0x00

Alternatively, access to Bootloader Mode can be achieved by holding down the ‘Reset’ button for at least 0.5 seconds.

5.6 Loading New Bootloader Images

To load a new image using the Bootloader, users may follow the following procedure:

1) Enter Bootloader Mode by sending USB command “0xAA, 0x04, 0x8A, 0x00, 0x00” (this will exit the main application)

2) Obtain the HEX file of the latest firmware image. The HEX file will be in standard INTEL HEX format
3) Convert the INTEL HEX image file into an array of 15360 bytes
4) Write 0x0400 to REG_ADDR using USB command “0xAA, 0x05, 0x81, 0x00, 0x04”
5) Write the first byte in the array to REG_DATA using the USB command “0xAA, 0x05, 0x82, DATA, 0x00”
6) Write 0x0401 to REG_ADDR using USB command “0xAA, 0x05, 0x81, 0x01, 0x04”
7) Write the second byte in the array to REG_DATA using the USB command “0xAA, 0x05, 0x82, DATA, 0x00”
8) Repeat until byte 15360 is written, which should correspond to address 0x3fff
NOTE: Image may then be verified by reading back all DATA bytes one by one.
10) Load Application by sending BOOTLOAD command “0xAA, 0x05, 0x80, 0x00, 0x00”

Alternatively, this update process may be initiated automatically using the Batlab Software GUI functions.

6. Software Development

Both the Batlab firmware and Batlab Software GUI are licensed as Open Source software. The Lexcelon team encourages both the redistribution and improvement of Batlab software so that users may fine tune its functions to meet their own needs and the needs of the hobbyist community at large. To aid in this endeavor, this section provides information on the Batlab development environments.

6.1 Firmware Development

The Batlab firmware was developed in MPLABX IDE and uses the free version of the XC8 compiler.

6.2 Batlab Software GUI Development

The Batlab software GUI was designed using C++ and Qt Creator 4.1.0.

6.3 Python Development Scripts

A Python library has been developed for hobbyists and more advanced users who would like to incorporate the Batlab hardware in their own cell testing workflow or environment. An example interface script that leverages various aspects of the Python library has been developed to permit users to interact with the Batlab in a command line environment. Documentation for this library is available alongside its storage location on the Lexcelon Github.

6.4 Code Access

All of the code repositories for Batlab software, firmware, and helper scripts will be stored on the Lexcelon public GitHub page: https://github.com/Lexcelon