CELLSIM 4S Battery Pack Simulator

Features

- 4x Battery Cell Simulators in Series
- Settable Per-cell Output Voltage 2.5-4.5V ±4%*
- Output Current 0-200mA (600mA peak)
- Output Current Sense 0-200mA ±10%*
- Indicator LEDs for each cell
- 3.3V/5V Compatible UART command interface
- Powered by included 12V 1A power adapter

* = Tolerances are measured relative to full scale.

Applications

- Battery Management System (BMS) testing
- Programmable micro-scale linear power supply
- Pleasingly rectangular hand warmer

Principle of Operation

The CELLSIM 4S utilizes four SCBS-Pico battery simulator boards tied together with an SCBS-Array motherboard in order to simulate a four-cell battery pack. A UART interface on the front panel allows control of individual cell voltages and reading of cell currents from a master computer. The CELLSIM 4S is targeted at BMS testing applications; cells can read currents up to 200mA, and can provide transient currents up to 600mA in the settable voltage range without damage.

Indicator LED Behaviors

Each battery cell includes a status indicator LED on the front panel of the CELLSIM 4S. Each cell's LED turns on for 100ms as a self-test when the device is first powered on. After power-on, LEDs behave according to the meanings in the table below.

Blink Duration	Meaning
10ms	Packet Received
100ms	Register Read
500ms	Register Write

Device Form Factor

The CELLSIM 4S contains four SCBS-Pico devices installed in a vertical card form factor onto an SCBS-Array motherboard. Rails on the device enclosure provide mechanical stability for the cards when the front panel of the enclosure is installed. Status LEDs on each of the SCBS-Picos' microcontroller modules are connected to the front panel via heat-formed PMMA light pipes integrated into the front panel assembly.

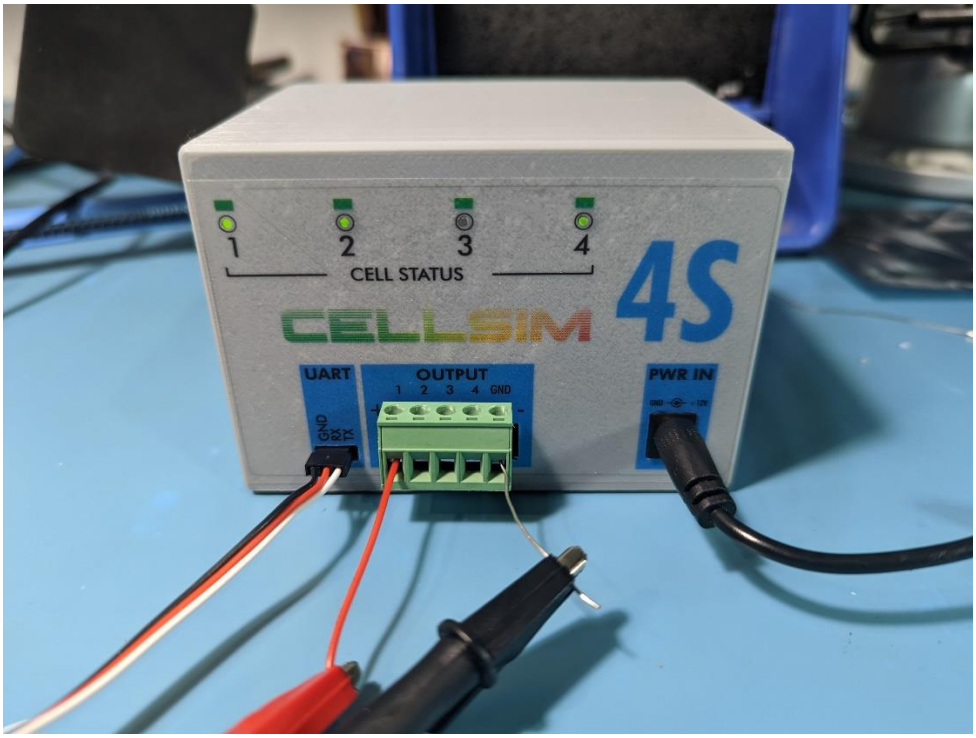
Left: CELLSIM 4S with front cover removed.

The SCBS-Array motherboard chains the four SCBS-Pico devices in series and supplies them power using four isolated DC-DC converters. UART communication is exposed to the master device through a connector on the front panel of the CELLSIM 4S, and is daisy-chained internally between the four SCBS-Pico boards using opto-isolators.

Device Electrical Connections

OUTPUT Connector

The terminals of the simulate battery cell are exposed on the front panel via the OUTPUT connector, with the leftmost terminal representing the positive terminal of the battery, and the rightmost terminal representing the negative terminal of the battery. Individual cell taps are exposed in between. Normal discharge of the battery (load across + and – terminals) can be simulated by connecting a load across the OUTPUT connector as shown in the figure below.



CELLSIM 4S connected to a load. Left: UART connection to master via USB-UART converter. Middle: OUTPUT connector with connections to DC load. Right: Power input barrel jack connector.

UART Connector

The UART connector is used to communicate with the battery cells. See the *UART Interface* section for connection details.

PWR IN Connector

The PWR IN connector accepts a center-positive 5.5x2.1mm barrel jack. The CELLSIM 4S is intended to work with the included 12V 1A power supply, but can work with any power supply that meets the criteria in the table below. The power input is reverse polarity protected up to 60V.

Parameter	Min	Typ	Max	Units
V_{in}	9	12	18	V
P_{in}	0.6		4	W

Communication Interface

The CELLSIM 4S device is controlled by a master device over UART in order to configure cell parameters, read cell currents, and set cell output voltages. Additional diagnostic data, such as cell firmware version, is also exposed over UART.

UART Interface

The CELLSIM 4S can be linked to a master computer using an off-the-shelf USB to UART converter. The UART interface supports multiple logic levels but only one baud rate and UART encoding scheme.

Parameter	**Value**
Baud Rate	9600 baud
Data Bits	8
Stop Bits	0 (N)
Parity Bit	1
Input Logic Level	3.3V, 5V
Output Logic Level	5V

Packet Checksum

All UART packets sent and received by CELLSIM 4S include an NMEA-style XOR checksum. An example code snippet for calculating the XOR checksum is included below, and is also available for use from the scbs_utils.py file in the Python SDK.

```
def calculate_checksum(contents_str):
    """
    @brief Calculates XOR checksum of a string.
    @param[in] contents_str String including header and contents (what would go between $ and
*).
    @retval Checksum (integer).
    """
    checksum = 0
    for i in range(len(contents_str)):
        checksum ^= ord(contents_str[i]) # XOR with character's ASCII value
```

Packet Format

CELLSIM 4S UART packets begin with a ‘$’ character, followed by a header of the form ‘BSXXX’ that represents the packet type, followed by one or more data fields, followed by a ‘*’ character and checksum. The end of a packet is indicated with a `\r\n` character sequence.

`$BSXXX,<VALUE>,..*XX\r\n`

Battery Simulator Command Packets

Packet Type	Packet Format
BSDIS	**Battery Simulator Discover** Discovers all battery simulator devices in the chain. Master transmits a BSDIS packet with PREV_CELL_ID=0, and each device increments PREV_CELL_ID and forwards the packet down the chain. The last device replies to the master with PREV_CELL_ID=n, where n is the number of devices in the chain. $BSDIS,<PREV_CELL_ID>*<CHECKSUM>
BSMRD	**Battery Simulator Multi Read** Reads a single register from all battery simulator devices simultaneously. Generates a single BSMRS packet reply upon success. $BSMRD,<REG_ADDR>*<CHECKSUM>
BSMWR	**Battery Simulator Multi Write** Writes a value to a single register on all battery simulator devices simultaneously. Generates a single BSMRS packet reply upon success. $BSMWR,<REG_ADDR>,<VALUE>*<CHECKSUM>
BSSRD	**Battery Simulator Single Read** Reads the value of a register on a single battery simulator, addressed via CELL_ID. Generates a single BSSRS packet reply upon success. $BSRRD,<CELL_ID>,<REG_ADDR>*<CHECKSUM>
BSSWR	**Battery Simulator Single Write** Writes a value to a register on a single battery simulator, addressed via CELL_ID. Generates a single BSSRS packet reply upon success. $BSRWR,<CELL_ID>,<REG_ADDR>,<VALUE>*<CHECKSUM>

Battery Simulator Response Packets

Packet Type	Packet Format
BSSRS	**Battery Simulator Single Response** Response from a single battery simulator (signed with CELL_ID). Includes a value that is a response to a BSSWR or BSSRD command, or an error created by another (invalid) command. Value can be “OK” (no error), “ERR:<CODE>” with an included error code, or “<VALUE>” where an actual value of a register is returned. $BSRSP,<CELL_ID>,<VALUE>*<CHECKSUM>

Getting Started Instructions

The following instructions describe the process of connecting CELLSIM 4S. CELLSIM 4S and its Python SDK are cross-platform compatible, and can be run on any device that supports Python (Windows, Linux, Mac).

Prerequisites: Python, Poetry (package manager).

Establishing UART Connection

1. Connect the USB to UART converter's UART output to the UART port on CELLSIM 4S. Connect the converter's USB port to the master computer.
2. Locate the COM port utilized by the USB to UART converter (e.g. COM7).
3. Open the scbs/scripts directory. In a Bash terminal (or Bash terminal emulator, if running on Windows), run `poetry shell`. This should open a virtual environment running the Python version specified in the pyproject.toml file. You may need to run `poetry install` in order to install necessary dependencies onto your system.
4. Run `python scbs_utils.py COM7`, where `COM7` is replaced with the COM port you located in step 2. You should be greeted with the SCBS Master Utility welcome text, as shown below. The SCBS Master Utility can be exited at any time by typing "EXIT", or with Ctrl+Z followed by ENTER.

```
$ python scbs_utils.py COM7
Welcome to the SCBS Master Utility!
Supported Commands:
    DIS - Cell Discover
        DIS <PREV_CELL_ID>
    MRD - Multi Read
        MRD <REG_ADDR>
    MWR - Multi Write
        MWR <REG_ADDR> <VALUE>
    SRD - Single Read
        SRD <CELL_ID> <REG_ADDR>
    SWR - Single Write
        SWR <CELL_ID> <REG_ADDR> <VALUE>
    SRS - Single Response
        SRS <CELL_ID> <VALUE>
Type EXIT to quit.
>>>
```

Initializing the Cells

Before other commands can be sent to CELLSIM 4S, the battery cells must first have their cell ID's initialized. This can be done by sending a BSDIS packet to the first cell in the chain. The response should indicate the total number of cells in the chain. The command and response should look like the following.

```
>>> DIS 0
        Sending: $BSDIS,0*53
        Response: b'$BSDIS,4*57\r\n'
```

After the DIS command is complete, cells can be addressed individually using their cell IDs. Note that the index of the first cell is 1, and the index of the last cell is 4.

Interacting with Cells

Cells can be interacted with via UART packet by writing to and reading from the register addresses below using MWR and SWR (to write) or MRD and SRD (to read).

Address	Register Name	Access Type	Description
0x1000	Output Voltage	READ/WRITE	Accepts a float between 2.5 and 4.5 and sets the output voltage of the cell to the corresponding value. Responds with OK if output voltage was set correctly. Responds with ERR:<CODE> if there was an error while writing to the register.
0x2000	Output Current	READ	Returns a float equal to the cell's measured output current, in mA. Note that the cell can only measure around 10-200mA; any values outside of this range will appear "railed" to the closest edge of the range.
0x3000	Firmware Version	READ	Responds with a string containing the firmware version of the battery cell. The firmware version is in the form <device_name>-<semantic_version>-<release_candidate> (e.g. "scbs_pico-0.1.0-rc1"). If the firmware version is stable (e.g. not a release candidate, the release_candidate field and preceding hyphen will be omitted.

Example of setting all cells to output 4.5V:

```
>>> MWR 1000 4.5
        Sending: $BSMWR,1000,4.5*77
        Response: b'$BSMWR,1000,4.5*77\r\n'
```

Example of setting cell 1 to output 3.3V:

```
>>> SWR 1 1000 3.3
        Sending: $BSSWR,1,1000,3.3*75
        Response: b'$BSSRS,1,OK*76\r\n'
```

Example of reading all cell currents:

```
>>> MRD 2000
        Sending: $BSMRD,2000*64
        Respons: b'$BSMRD,2000,107.13,110.69,108.25,105.76*64\r\n'
```

Example of reading the firmware version of cell 2:

```
>>> SRD 2 3000
        Sending: $BSSRD,2,3000*65
        Response: b'$BSSRS,2,scbs_pico-0.1.0*26\r\n'
```

Error Codes

UART commands that fail to execute successfully are replied to with error codes. As soon as a command packet is determined to be invalid, the command packet is dropped and an SRS (Single Response) packet is sent in its place containing the corresponding error code. Write-only operations with no need for forwarding, such as SWR (Single Write) are replied to with the “OK” (no error) error code in an SRS packet.

NOTE: All error codes are in hexadecimal (base 16).

Code	**Description**
OK	No error.
ERR:1	Address not recognized.
ERR:2	Maximum packet length exceeded (happens if too many cells are chained together and sent a BSMRD command).
ERR:3	Write to this address not supported.
ERR:F	Received an invalid packet.

Example of a failed read from a non-existent address 0x7000 (note that the error response is from the first cell in the chain; the MRD request is not forwarded to remaining cells as soon as it is found to be invalid):

```
>>> MRD 7000
        Sending: $BSMRD,7000*61
        Respons: b'$BSSRS,1,ERR:1*3C\r\n'
```

Example of a failed write to a read-only address:

```
>>> MWR 2000 45
        Sending: $BSMWR,2000,45*5a
        Response: b'$BSSRS,1,ERR:3*3E\r\n'
```

Revision History

Revision	Date	Author	Note
1.0.0	2022-12-08	J. McNelly	Initial release.