Cambrionix
CHARGE • CONNECT • MANAGE

Cambrionix Universal Series
Terminal Command Reference for Cambrionix Universal Chargers

1 Introduction

The guide describes how to remotely control devices in the Universal Series (PPxx, Uxx and ThunderSync) via their control interface. Remote control permits the Universal Series to be integrated into a larger system that is controlled by a host computer.

On the U8 series, command is achieved via the smaller mini-USB socket. On the U16 and PPxx series, the host USB socket is used. On ThunderSync control is via a Thunderbolt port. Later products may use alternative transports to connect to the host. In all cases the system appears as a virtual serial port (also called a UART or VCP). On Microsoft Windows, the system will appear as a COM port. On macOS, a device file is created in the `/dev/` directory. This is of the form `/dev/tty.usbserial-S` where `S` is an alpha-numeric serial string unique to each device in the Universal Series.

Devices in the Universal Series are hereinafter collectively referred to as “U8-U16”.

Commands that are issued to the serial port are referred to as *terminal commands*.

The settings modified by commands in this document are volatile – that is, the settings are lost when the U8-U16 is rebooted or powered off.

This document is subject to change. Any parsing of data should be tolerant of new features being added.

2 Prerequisites

FTDI Drivers

U8-U16 devices incorporate a FT230X USB to UART converter IC from FTDI International. On Windows 7 or later, a driver may automatically be installed (if Windows is configured to download drivers from the internet automatically). If this is not the case, or if a Mac or Linux platform is used, the driver may be downloaded from www.ftdichip.com. The VCP drivers are required.

UART Settings

The default communications settings should be set to 115200 baud, 8 data bits, no parity, and 1 stop bit. This is sometimes referred to as 115200,8,N,1. No flow control is used. ANSI terminal emulation should be selected. Lines sent by the U8-U16 are terminated with <CR><LF>. Lines received by the U8-U16 should be terminated with at least <CR>. <LF> is ignored.

The U8-U16 will accept back-to-back bytes, however, the host computer should wait for a fresh command prompt from the U8-U16 before issuing a new command string.

The serial_speed command allows increasing the baud rate to 1000000. The U8-U16 will automatically drop back to 115200 on receipt of serial errors.

Boot text and command prompt

At boot, the U8-U16 will issue a string of ANSI escape sequences to reset an attached terminal emulator. This is followed by the title block, then a command prompt.

The command prompt is always `>>`, followed by <CR><LF>, except in boot mode where it is `boot>>` followed by <CR><LF>.

To reach a new boot prompt, send CTRL-C. This cancels any partially entered command string.

3 Command Summary

Command	Description
`mode <m> [p]`	Set mode <m> for port [p] or all ports
`mode c <p> [cp]`	Set charge mode for a port <p> with optional profile [cp] to enter immediately
`state [p]`	Show state for port [p] or all ports
`system`	Show hardware and firmware information
`health`	Show voltages, temperature, errors and boot flag
`cef`	Clear error flags
`sef [flags]...`	Set error flags
`crf`	Clear rebooted flag
`limits`	Show voltage and temperature limits
`list_profiles`	List all profiles on system
`en_profile <i> <1|0>`	Enable (`1`) or disable (`0`) profile <i>
`set_profiles <p> [l]`	Set profiles <`l`> associated with port <p>
`get_profiles <p>`	List profiles associated with port <p>
`sec [arm|disarm]`	Set or reveal security mode
`host [auto|manual]`	Show if USB host is present, and set mode change
`id`	Show id string
`bd`	Board description
`logc <s>`	Report mA for each port every <s> seconds
`loge [s]`	Report state for all ports every [s] seconds and events as they occur
`l`	Live view (periodically updated screen showing system state)
`remote [exit|kexit]`	Enter or exit mode where console is controlled by terminal
`ledb <p> <row> <ptn>`	Set individual LED on row <row>, port <p> to flash bit pattern <ptn>
`leds <row> [ptnstr]`	Set flash pattern of a string of LEDs on row <row> to string <ptnstr>
`keys`	Read key click event flags
`lcd <row> <col> <str>`	Write string <s> to LCD at row <row>, column <col>
`clcd`	Clear LCD
`beep [ms]`	Make console beep for [ms] milliseconds
`cls`	Clear terminal screen
`reboot [watchdog]`	Reboot (optionally by using watchdog timeout)
`serial_speed [test|fast|slow]`	Change serial interface speed
`set_delays`	Change internal delays

Notes

Throughout this text, compulsory parameters to terminal commands are shown in triangular brackets: < >. Optional parameters are shown in square brackets: [ ].

Text as it appears in the serial data stream is shown in the `Courier New` typeface.

Some products don't support all the commands e.g. PP15 doesn't have and LED or LCD display and so those commands aren't available.

4 id command

In order to easily identify the product that is providing the serial port being used for communication, there is the id command. The id command also provides some basic information about the firmware.

Syntax: `id`

Response:

A single line of text containing multiple name:value pairs that can be used to identify the product.

Name	Value
mfr	Manufacturer string eg cambrionix
mode	A string to describe which operating mode the firmware is in eg main
hw	A short string naming the hardware eg PP15S
hwid	A hexadecimal value used internally to identify the board eg 0x13
fw	A pseudo number representing the firmware revision eg 1.68
bl	A pseudo number representing the bootloader revision eg 0.15
sn	A serial number, currently not used, eg 000000
group	Used on some products to order firmware updates which is useful when updating boards that are daisy-chained together so that down-stream boards are updated and rebooted first.
fc	Firmware Code is used to denote which firmware the board accepts

Example

```
>> id
mfr:cambrionix,mode:main,hw:PP15S,hwid:0x13,fw:1.68,bl:0.15,sn:000000,group:-,fc:un
```

5 bd command

The bd command provides a description of the internal connections to the ports of the board. This includes all the charging ports as well as the virtual serial port and any expansion ports that may be present. This is to allow software to be able to navigate the USB connection tree without needing to have in-built specific knowledge of how this board was designed.

Syntax: `bd`

Response:

Some name value pairs indicating the presence or not of the optional features of the board. This is followed by a description of each USB hub in turn, listing what is attached to each port of that hub. Each port of a hub will be attached to a charging port, an expansion port, a downstream hub, a USB serial device or is unused.

The optional features are indicated by these entries:

Name	Value
Ports	The number of charging ports the product provides
Sync	A '1' indicates the product provides sync capability
Temp	A '1' indicates the product can measure temperature
EXTPSU	A '1' indicates the product is supplied with a voltage that is greater than 5.2V, generally 12V. This is usually provided by a laptop style PSU.

The attachment section can have the following entries, all indices are 1 based:

Name	Value	Description
Nodes	<n>	A number indicating the number of nodes this description set includes. A node will be either a USB hub or a USB controller.
Node <i> Type	<type>	<i> is an index indicating which node this is. <type> is an entry from the Node Type table below.
Node <i> Ports	<n>	A number indicating how many ports this node has. USB hubs generally have 4 or 7 ports.
Hub <i> Port <p>	Hub <j>	The USB hub <i> has a down-stream hub <j> connected to its port <p>
Hub <i> Port <p>	Control Port	The USB hub <i> has the USB serial port attached to port <p>
Hub <i> Port <p>	Expansion Port <e>	The USB hub <i> has an expansion port attached to port <p>
Hub <i> Port <p>	Port <c>	The USB hub <i> has the charging port <c> attached to port <p>
Hub <i> Port <p>	Optional Hub <j>	The USB hub <i> may have a down-stream hub <j> connected to its port <p> but this is optional so may not be fitted
Hub <i> Port <p>	Turbo Hub <j>	The USB hub <i> has a USB hub capable of operating in Turbo mode attached to port <p>
Hub <i> Port <p>	USB3 Hub <j>	The USB hub <i> has a USB 3.x hub attached to port <p>
Hub <i> Port <p>	Unused Port	The USB hub <i> has nothing attached to its port <p>

Node type can be one of the following:

Node Type	Description
Hub <j>	A USB 2.0 hub index <j>
Optional Hub <j>	A USB hub that may be fitted, index <j>
Root <r>	A USB controller with a root hub which also means the USB bus number will change
Turbo Hub <j>	A USB hub capable of operating in Turbo mode with index <j>
USB3 Hub <j>	A USB 3.x hub with index <j>

Example

```
>> bd
Ports: 15
Sync: 1
Temp: 1
EXTPSU: 1
Nodes : 5
Node 1 Type : Hub 1
Node 1 Ports : 4
Hub 1 Port 1 : Hub 3
Hub 1 Port 2 : Hub 5
Hub 1 Port 3 : Hub 4
Hub 1 Port 4 : Hub 2
Node 2 Type : Hub 2
Node 2 Ports : 4
Hub 2 Port 1 : Control Port
Hub 2 Port 2 : Port 1
Hub 2 Port 3 : Port 2
Hub 2 Port 4 : Port 3
Node 3 Type : Hub 3
Node 3 Ports : 4
Hub 3 Port 1 : Port 4
Hub 3 Port 2 : Port 5
Hub 3 Port 3 : Port 6
Hub 3 Port 4 : Port 7
Node 4 Type : Hub 4
Node 4 Ports : 4
Hub 4 Port 1 : Port 11
Hub 4 Port 2 : Port 10
Hub 4 Port 3 : Port 9
Hub 4 Port 4 : Port 8
Node 5 Type : Hub 5
Node 5 Ports : 4
Hub 5 Port 1 : Port 15
Hub 5 Port 2 : Port 14
Hub 5 Port 3 : Port 13
Hub 5 Port 4 : Port 12

>>
```

Parsing of the output should be flexible as new fields may be added in future.

6 `mode` command

Modes and states

Each port on the U8-U16 has several *modes* it can be placed in. When placed in a particular mode, the port will transition from one *state* to another according to whether a mobile device is attached, and whether that device is charging or not.

The available modes depend on the variant of the hardware. 'S' variants (U8S, U8S-EXT, U16S, PP15S, etc.) support a SYNC mode whereby attached mobile devices can communicate with a host computer via a built-in USB hub. 'C' variants do not support SYNC mode. All variants support charge, biassed and off modes.

The mode command

Each port can be placed into one of four modes by using the `mode` command.

Syntax: `mode` <m> [p] [cp]

<m> is one of the following mode characters:

Mode character <m>	Mode	Description
c	Charge	The port is readied for charging a device, and can detect if a device is attached or detached. If a device is attached, the charger profiles enabled for that port are tried one by one. Then the device is charged using the profile that yielded the highest current. During the above, the port is disconnected from the host USB bus.
s	Sync	The port is attached to the host USB bus via a on-board high-speed (480Mbit/s) USB hub. The device may draw charging current from VBUS depending on the device capabilities.
b	Biassed	The port is disconnected from the host USB bus. A weak current (approx 2mA) is applied to VBUS to detect if a device is attach or detached. No charging occurs.
o	Off	Power (VBUS) to the port is removed. No charging occurs. No device attach or detach detection is possible.

The port parameter, [p], is optional. It can be used to specify the port number. If left blank, all ports are affected by the command.

The charging profile parameter [cp] is optional but can only be used when putting a single port into charge mode. If specified then that port will directly enter charge mode using the chosen profile.

Response:

None (new command prompt appears)

Examples

To turn off all ports:

```
>> mode o
```

To put just port 2 in charge mode:

```
>> mode c 2
```

To put just port 4 in charge mode using profile 1:

```
>> mode c 4 1
```

Default port mode

The default mode after reset is charge mode. However, if a powered host computer is attached to the host USB socket at reset, the U8-U16 will transition to sync mode.

7 `state` command

After a port is placed into a particular mode (e.g. charge mode) it can transition into a number of states. The `state` command is used to list the state of each port. It also shows the current being delivered to the mobile device, any error flags, and the charge profile employed.

Syntax: `state` [p]

Where [p] is the port number.

Response:

Comma separated fields, one row per port.
Row format: p, current_ma, flags, profile_id, time_charging,time_charged,energy

Field	Description	
p	The port number pertaining to the row	
current_ma	Current being delivered to the mobile device, in mA (milliamperes)	
flags	List of case-sensitive flag characters, separated by spaces. O, S, B, I, P, C, F are mutually exclusive. A, D are mutually exclusive.	
	Flag	**Description**
	“O”	Port is in OFF mode
	“S”	Port is in SYNC mode
	“B”	Port is in BIASSED mode
	“I”	Port is in charge mode, and is IDLE
	“P”	Port is in charge mode, and is PROFILING
	“C”	Port is in charge mode, and is CHARGING
	“F”	Port is in charge mode, and is has FINISHED charging
	“A”	Device is ATTACHED to this port
	“D”	No device is attached to this port. Port is DETACHED
	“T”	Device has been stolen from port: THEFT
	“E”	ERRORs are present. See `health` command
	“R”	System has rebooted. See `crf` command
	“r”	Vbus is being reset during mode change
profile_id	The unique profile ID number. “0” if not charging or profiling	
time_charging	Time in seconds the port has been charging for	
time_charged	Time in seconds that the port has been charged for (x means not valid yet).	
energy	Energy the device has consumed in watthours (calculated every second)	

Note : Current measurement is typically has a resolution of 9.76mA or 11.18mA depending on the product.

Examples

Default state of U8C, with no attached devices (`D`) and all ports in charge mode, but idle (`I`):

```
>> state
 1, 0000, D I, 0, 0, x, 0.00
 2, 0000, D I, 0, 0, x, 0.00
 3, 0000, D I, 0, 0, x, 0.00
 4, 0000, D I, 0, 0, x, 0.00
 5, 0000, D I, 0, 0, x, 0.00
 6, 0000, D I, 0, 0, x, 0.00
 7, 0000, D I, 0, 0, x, 0.00
 8, 0000, D I, 0, 0, x, 0.00
```

An iPhone connected to port 5, which is charging at 1044mA using profile_id 1

```
>> state
 1, 0000, D I, 0, 0, x, 0.00
 2, 0000, D I, 0, 0, x, 0.00
 3, 0000, D I, 0, 0, x, 0.00
 4, 0000, D I, 0, 0, x, 0.00
 5, 1044, A C, 1, 5, x, 0.01
 6, 0000, D I, 0, 0, x, 0.00
 7, 0000, D I, 0, 0, x, 0.00
 8, 0000, D I, 0, 0, x, 0.00
```

Whilst the above iPhone was charging, another mobile device was attached to port 8. This is being profiled using profile_id 2 prior to charging:

```
>> state
 1, 0000, D I, 0, 0, x, 0.00
 2, 0000, D I, 0, 0, x, 0.00
 3, 0000, D I, 0, 0, x, 0.00
 4, 0000, D I, 0, 0, x, 0.00
 5, 0927, A C, 1, 10, x, 0.05
 6, 0000, D I, 0, 0, x, 0.00
 7, 0000, D I, 0, 0, x, 0.00
 8, 0048, A P, 2, 5, x, 0.01
```

Showing just the state of port 5:

```
>> state 5
 5, 1015, A C, 1, 10, x, 0.05
```

A global system error reported by the `E` flag:

```
>> state
 1, 0000, E D I, 0, 0, x, 0.00
 2, 0000, E D I, 0, 0, x, 0.00
 3, 0000, E D I, 0, 0, x, 0.00
 4, 0000, E D I, 0, 0, x, 0.00
 5, 0927, E A C, 1, 15, x, 0.00
 6, 0000, E D I, 0, 0, x, 0.00
 7, 0000, E D I, 0, 0, x, 0.00
 8, 0048, E A P, 2, 2, x, 0.01
```

Theft of device 5 reported by the `T` flag:

```
>> state
 1, 0000, D I, 0, 0, x, 0.00
 2, 0000, D I, 0, 0, x, 0.00
 3, 0000, D I, 0, 0, x, 0.00
 4, 0000, D I, 0, 0, x, 0.00
 5, 0000, T D I, 0, 0, x, 0.00
 6, 0000, D I, 0, 0, x, 0.00
 7, 0000, D I, 0, 0, x, 0.00
 8, 0000, D I, 0, 0, x, 0.00
```

8 Viewing system parameters

To view system-wide parameters, issue the `system` command.

Syntax:

```
system
```

Response:

First row: system title text, beginning with “Cambrionix”
Second and subsequent rows: parameter:value pairs, one pair per row

Parameter	Description	Possible values	
Hardware	Name of U8-U16 device. More names may be added in future.	“U8C”	8 Port Charger, 5V supply
		“U8C-EXT”	8 Port Charger, 12V (nominal) supply
		“U8S”	8 Port Charge+Sync, 5V supply
		“U8S-EXT”	8 Port Charge+Sync, 12V (nominal) supply
		“U16C”	16 Port Charge, 5V supply
		“U16S”	16 Port Charge+Sync, 5V Supply
		“Series8 Charger”	8 Port consumer product
		PP15C	15 Port Charger, 12V (nominal) supply
		PP15S	15 Port Charge+Sync, 12V (nominal) supply
Firmware	Firmware version string	Typically in “n.nn” format, where is decimal 0..9 Strings are permitted (e.g. “1.09-Alpha”) for some versions	
Compiled	Date and Time of the Firmware		
Group	Group letter read from PCB jumpers	1 character, 16 values: “-”, “A” .. “O” A dash (“-”) means no group jumper is fitted	
Panel ID	Panel ID number of front panel board	“None” if no panel was detected Otherwise “0” .. “15”	
LCD	Presence of LCD display	“Absent” or “Present” If product can support an LCD	

Example

```
>> system
cambrionix PP15S 15 Port USB Charge+Sync
Hardware: PP15S
Firmware: 1.68
Compiled: Feb 02 2017 12:02:05
Group: -
Panel ID: Absent
```

Notes

- The system title text may change across firmware releases.
- The “Group” parameter is updated every time the `system` command is run.
- The “Panel ID” is updated at power-up or reboot.
- The “LCD” parameter can only become “Present” at power-up or reboot. It can become “Absent” during run-time if the LCD is no longer detected, but is not guaranteed to become “Present” again if the LCD is re-attached whilst the U8-U16 is operating.

9 System health

The health command displays the supply rail voltages, PCB temperature, error flags and the rebooted flag.

Syntax:

```
health
```

Response:

parameter:value pairs, one pair per row.

Parameter	Description	Possible values	
5V Now	Present 5V supply voltage		
5V Min	Lowest 5V supply voltage seen		
5V Max	Highest 5V supply voltage seen		
5V Flags	List of 5V supply rail error flags, separated by spaces		No flags: voltage is acceptable
		“UV”	Under-voltage event occurred
		“OV”	Over-voltage event occurred
5V Rail 2 Now	Present 5V supply voltage		
5V Rail 2 Min	Lowest 5V supply voltage seen		
5V Rail 2 Max	Highest 5V supply voltage seen		
5V Rail 2 Flags	List of 5V supply rail error flags, separated by spaces		No flags: voltage is acceptable
		“UV”	Under-voltage event occurred
		“OV”	Over-voltage event occurred
12V Now	Present 12V supply voltage		
12V Min	Lowest 12V supply voltage seen		
12V Max	Highest 12V supply voltage seen		
12V Flags	List of 12V supply rail error flags, separated by spaces		No flags: voltage is acceptable
		“UV”	Under-voltage event occurred
		“OV”	Over-voltage event occurred
Temperature Now	Present PCB temperature, °C	“>100 C”	Temperature is above 100°C
		“<0.0 C”	Temperature is below 0°C
		“tt.t C”	Temperature, e.g. “32.2 C”
Temperature Max	Highest PCB temperature seen, °C	“>100 C”	Temperature is above 100°C
		“<0.0 C”	Temperature is below 0°C
		“tt.t C”	Temperature, e.g. “32.2 C”
Temperature Flags	Temperature error flags		No flags: temperature is acceptable
		“OT”	Over-temperature (over-heat) event occurred
Rebooted Flag	Used to detect if system has booted or rebooted	“R”	System has booted or rebooted
			Flag cleared using `crf` command

Note : 5V Rail 2 is optional, currently only present on a PP15

Note : 12V is optional, currently only present on a PP15, U8 Ext

Note : “12V” for some future products will become “Input” to better support different input voltages e.g. 18V

Note : voltage accuracy is typically 3% but can be up to 5%

Examples

A normal, healthy system. The rebooted flag has been cleared using the `crf` command:

```
>> health
System up for:    12368 seconds
5V Now:   5.21
5V Min:   5.21
5V Max:   5.24
5V Flags:
12V Now: 12.25
12V Min: 12.21
12V Max: 12.34
12V Flags:
Temperature Now: 36.0 C
Temperature Max: 38.1 C
Temperature Flags:
Rebooted flag:
```

A system with all error flags showing. In addition, the boot flag has not been cleared:

```
>> health
System up for:    12576 seconds
5V Now:   5.21
5V Min:   4.24
5V Max:   6.11
5V Flags: UV OV
12V Now: 12.25
12V Min: 9.21
12V Max: 14.34
12V Flags: UV OV
Temperature Now: 36.0 C
Temperature Max: 78.1 C
Temperature Flags: OT
Rebooted flag: R
```

10 Clearing the error flags

The OV, UV, and OT flags are latched 'on', and can be cleared using the `cef` command (“clear error flags”). If the error condition persists, the U8-U16 will set the flag again after it is cleared.

11 Setting the error flags

It can be useful to set the error flags to examine the system behaviour when an error occurs. To do this, use the `sef` command (“set error flags”)

Syntax:

`sef` <flags>

where <flags> is a list of one or more flags:

Flag	Function
5UV	5V rail under-voltage
5OV	5V rail over-voltage
12UV	12V rail under-voltage
12OV	12V rail over-voltage
OT	PCB over-temperature

Example

To set the 5UV and OT flags:

```
>> sef 5UV OT
```

Notes

1. Calling `sef` without parameters is valid, and sets no error flags.

2. Flags may be set using `sef` on a system that could not trigger those flags via hardware. For example, the 12UV and 12OV flags can be set by `sef` on non-EXT hardware variants, even though those hardware variants have no 12V rail.

12 Clearing the rebooted (R) flag

The boot flag can be cleared using the `crf` command ("clear rebooted flag"). This is useful to observe if the U8-U16 has rebooted in between commands. To do this, clear the `crf` flag at the start of a controlling script. Then poll it along with the other flags using the `health` or `state` commands. If the R flag is found to be set, then previous settings (such as profile selection, or port modes) will have been lost, and will need to be re-sent to the U8-U16.

13 Displaying system limits

To show the limits (thresholds) at which the under-voltage, over-voltage and over-temperature errors are triggered, issue the `limits` command.

Example

```
>> limits
5V Min:   4.85
5V Max:   5.60
12V Min: 10.00
12V Min: 14.00
Temperature: 70.0 C
```

Notes

- The limits are hard-coded into the firmware and cannot be changed by a terminal command.
- The supply rails are sampled every 1ms. They must be over or under voltage for 20ms before a flag is raised.
- The temperature is measured every 10ms. Batches of 32 samples are averaged to give the result.
- If the 5v Rail is sampled twice in a row below 4.5v the ports are shutoff

14 Profile Control

When a device is attached to a U8-U16, the firmware manipulates the USB data lines in various sequences (for example, applying different voltages, or resistor divider ratios). Each of these sequences is called a 'profile'. Many mobile devices will not charge properly unless they are presented with the correct profile. Typically, a device not presented with a profile it recognises will draw less than 500mA from the USB port. When presented with the correct profile, it will draw more than 500mA. This speeds up charging.

The firmware has a master list of charger profiles that are stored in the U8-U16 flash memory. When a mobile device is attached the U8-U16 tries each profile in turn, and measures the current that the mobile device draws. Once all the profiles have been tried, the U8-U16 selects the profile that drew the highest current.

In some cases it may not be desirable for the U8-U16 to scan all the profiles in this way. For example, if only devices manufactured by Acme Inc. are attached, we need only try the profile that Acme Inc. products recognise. This reduces the time delay between when a user attaches a device and sees evidence of the device charging properly.

The U8-U16 provides the means to limit the profiles tried, both on a 'global' level (across all ports) and on a port-by-port basis.

15 Global profiles

The firmware contains a 'global' list of profiles. These can be listed by using the `list_profiles` command:

```
>> list_profiles
  1, enabled
  2, enabled
  3, enabled
  4, enabled
```

Each profile listed has four parameters, separated by commas. They are in order: profile_id, enable_flag.

The profile_id is a unique number that always corresponds to one profile type. It is not varied across firmware releases. Although for current versions of firmware the profile_id increases contiguously, this will not necessarily always be the case. profile_id is a positive integer starting at 1. A profile_id of 0 is reserved for when the absence of a profile is to be indicated.

Each profile can be enabled or disabled (across all ports) by changing the enable flag.

16 Enabling and disabling profiles for all ports: `en_profile`

The `en_profile` command is used to enable and disable each profile. The effect applies to all ports.

Syntax: `en_profile` <i> <1|0>

where <i> is the profile id (obtained from list_profiles), and 1|0 is the enable flag (1 = enabled, 0 = disabled).

Example

To disable a profile for all ports use the command:

```
>> en_profile 2 0
```

This may be confirmed by using list_profiles:

```
>> list_profiles
  1, enabled
  2, disabled
  3, enabled
  4, enabled
```

Missing and unrecognised profiles are reported thus:

```
>> en_profile
*E409: Profile ID expected
>> en_profile 9
*E407: Unknown profile id: 9
```

The command has an immediate effect. If the command is issued whilst a port is profiling, then the command will only have an effect if that profile has not yet been reached in the profiling process.

Operation with no enabled profiles

If all profiles for a port are disabled, the port will transition into the biassed port state when a device is inserted. This permits device attach and detach detection to work, but no charging will occur. Security (theft detection) will still operate if all profiles are disabled, as will the attach (`A`) and detach (`D`) flags reported by the `state` command.

17 Individual port profiles: `get_profiles` and `set_profiles`

The profiling of each port operates in one of two ways. By default, all enabled profiles are tried on a port. They are tried in the order in which they are reported by `list_profiles`.

Alternatively, the user may assign an ordered list of profiles to each individual port. Between 1 and 4 profiles can exist in such a list.

To list the profiles assigned to a port, use the `get_profiles` command. For example, to list the profiles assigned to port 8:

```
>> get_profiles 8
  1, enabled
  2, enabled
  3, enabled
  4, enabled
```

The output is in the same format as `list_profiles`. However, only profiles assigned to that port are shown.

To assign profiles to an individual port, using the `set_profiles` command. This takes the port number, and a list of profile ids. For example, to restrict port 8 to just use profile 1:

```
>> set_profiles 8 1
```

which will result in:

```
>> get_profiles 8
  1, enabled
```

Or, to select profiles 2 and 3 for port 5:

```
>> set_profiles 5 2 3
```

which will result in:

```
>> get_profiles 5
  2, enabled
  3, enabled
```

To assign all system profiles to a port, issue set_profiles without a list of profiles. For example, to assign all profiles to port 3:

```
>> set_profiles 8
```

18 Boot mode

Boot mode is used to modify the firmware within the U8-U16. It is not documented here. However, if you find the system in boot mode (where the command prompt is `boot>>`), you can return to normal operation by issuing the `reboot` command or by power-cycling the system. In boot mode, the PWM (fan) output is increased to maximum duty cycle, as the temperature servo loop is not operating.

19 Device security

The U8-U16 can log if a device was unexpectedly removed from a port. The `sec` command can be used to put all ports into an 'armed' security state. If a device is removed in the armed state, then an alarm will sound on the console, and the `T` flag is shown text to the port in the listing from the `state` command. The `T` flag is also shown in the live view.

Syntax:

`sec` [arm|disarm]

Response (to no parameters):

`armed|disarmed`

Response (to arm|disarm parameter):

none

Examples

To arm the system:

```
>> sec arm
```

To disarm:

```
>> sec disarm
```

Reveal armed state:

```
>> sec
disarmed
```

To clear all theft bits and silence a sounding alarm, disarm then re-arm the system.

Notes

If more advanced security operation is required via the terminal, consider polling the `A` (attached) and `D` (detached) flags provided by the `state` command. This might be useful where a script controlling the U8-U16 is running on the host computer, and needs to report device removal via an email, for example.

If theft detection is needed, but no device charging or syncing is desired, set the ports to biassed mode.

20 Host detection

The U8-U16 looks at the HOST USB socket for a 5V VBUS supply from an attached host computer. If the VBUS line is present at boot, or goes from low to high (i.e. the host computer is turned on, or a powered host computer is attached to the U8-U16), then the U8-U16 will change into sync mode. Similarly, if the VBUS line goes from high to low (i.e. a powered host computer was detached from the U8-U16, or the host computer was switched off), then the U8-U16 will enter charge mode.

The `host` command can be used to reveal if a host computer is attached or not. It can also be used to prevent the U8-U16 from automatically changing modes.

Syntax: `host` [mode]

Mode string [mode]	Description
`auto`	The mode of all ports changes automatically as described above
`manual`	Only the front-panel switches, EXT inputs, and terminal commands can be used to change modes. The host USB socket has no effect.

Response (if no parameter is supplied):

Present: <`yes|no`>

Mode change: <`auto|manual`>

Response (if parameter is supplied)

None (new command prompt appears)

Notes

- The presence of the host computer is still reported if the mode is set to manual.
- On U8C, U8C-EXT, U16C and PP15C the host command is present, but is useless.
- Only the U8 can report the host to be not present as it is the only board that has a separate control and host connection.

Examples

To set host mode to manual:

```
>> host manual
```

To see if a host is present, and the reveal the mode:

```
>> host
Present: no
Mode change: auto
```

And with a host attached:

```
>> host
Present: yes
Mode change: auto
```

21 Logging port current

The logc command is used to display the current for all ports at a preset time interval.

Syntax:

`logc` [seconds] (seconds is in range 1..32767)

Response:

CSV (comma separated values), which can be copied into a spreadsheet or grapher program.

Example

Here is a device being attached to port 5, left for 6 seconds, and then removed:

```
>> logc 1
Logging current with period (mins:secs): 00:01
Press q or CTRL-C to stop

000000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000
000001, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000
000002, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000
000003, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000
000004, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000
000005, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000
000006, 0000, 0000, 0000, 0000, 0956, 0000, 0000, 0000
000007, 0000, 0000, 0000, 0000, 1005, 0000, 0000, 0000
000008, 0000, 0000, 0000, 0000, 1005, 0000, 0000, 0000
000009, 0000, 0000, 0000, 0000, 1024, 0000, 0000, 0000
000010, 0000, 0000, 0000, 0000, 1005, 0000, 0000, 0000
000011, 0000, 0000, 0000, 0000, 1034, 0000, 0000, 0000
000012, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000
```

Notes

- The parameter is specified in seconds, but is confirmed as minutes:seconds for convenience:

```
>> logc 600
Logging current with period (mins:secs): 10:00
```

- Current logging works in both charge and sync modes.
- Some terminal emulators (e.g. PuTTY) can save the incoming serial data stream directly to a file. This is useful for subsequent plotting of the CSV output using a spreadsheet or graphing application.
- The current measurement ADC has a 9.76mA or 11.18mA step-size depending on the product. The range is 0 to 2488.76mA or 2850.9mA depending on product. The output is rounded to 1mA prior to display.

22 Logging events

The loge command is used to report port status change events and periodically report the state of all ports.

Syntax:

`loge` [seconds] (seconds is in range 0..32767)

Response:

CSV (comma separated values), which can be copied into a spreadsheet or graphing program.

If a seconds value of '0' is specified then the periodic reporting is disabled and only port status change events will be reported.

If no seconds parameter is supplied a default value of 60s will be used.

A timestamp is output before each event or periodic report.

Example

Here is a device being attached to port 4, left for 6 seconds, and then removed:

```
>> loge 3
Logging events
Press Ctrl-C to stop

System up for     7161
 1, 0000, R D I, 0, 0, x, 0.00
 2, 0000, R D I, 0, 0, x, 0.00
 3, 0000, R D I, 0, 0, x, 0.00
 4, 0000, R D I, 0, 0, x, 0.00
 5, 0000, R D I, 0, 0, x, 0.00
 6, 0000, R D I, 0, 0, x, 0.00
 7, 0000, R D I, 0, 0, x, 0.00
 8, 0000, R D I, 0, 0, x, 0.00
System up for     7164
 1, 0000, R D I, 0, 0, x, 0.00
 2, 0000, R D I, 0, 0, x, 0.00
 3, 0000, R D I, 0, 0, x, 0.00
 4, 0009, R A P, 1, 0, x, 0.00
 5, 0000, R D I, 0, 0, x, 0.00
 6, 0000, R D I, 0, 0, x, 0.00
 7, 0000, R D I, 0, 0, x, 0.00
 8, 0000, R D I, 0, 0, x, 0.00
Event at     7164
 4, 0009, R A P, 1, 0, x, 0.00
System up for     7167
 1, 0000, R D I, 0, 0, x, 0.00
 2, 0000, R D I, 0, 0, x, 0.00
 3, 0000, R D I, 0, 0, x, 0.00
 4, 1971, R A C, 1, 1, x, 0.00
 5, 0000, R D I, 0, 0, x, 0.00
 6, 0000, R D I, 0, 0, x, 0.00
 7, 0000, R D I, 0, 0, x, 0.00
 8, 0000, R D I, 0, 0, x, 0.00
Event at     7167
 4, 1971, R A C, 1, 1, x, 0.00
System up for     7170
 1, 0000, R D I, 0, 0, x, 0.00
 2, 0000, R D I, 0, 0, x, 0.00
 3, 0000, R D I, 0, 0, x, 0.00
 4, 0000, R D I, 0, 0, x, 0.00
 5, 0000, R D I, 0, 0, x, 0.00
 6, 0000, R D I, 0, 0, x, 0.00
 7, 0000, R D I, 0, 0, x, 0.00
 8, 0000, R D I, 0, 0, x, 0.00
Event at     7170
 4, 0000, R D I, 0, 0, x, 0.00
```

Notes

- Terminal commands are accepted while in this mode but commands are not echoed and the command prompt is not issued.

23 Beep

Makes the console piezo beep at 2kHz for specified number of milliseconds. The beep is performed as a background task - so the system can process other commands whilst the beep runs.

Syntax:

`beep` [ms] (range 0..32767)

Response:

None (new command prompt appears).

Notes

- The time [ms] has a resolution of 10ms
- A longer, running beep will not be interrupted by a shorter of zero-length beep.
- Keyboard beeps will not be audible whilst a `beep` command is sounding.
- The staccato beep from a sounding alarm is overridden by the continuous tone from a `beep` command. However, when the continuous `beep` completes, the system will return to the staccato beep.
- The beep pitch and front-panel key click sound can not be altered.
- Sending CTRL-G (ASCII “BEL”) from the terminal will cause a short beep to be generated.
- For a beep to be heard, a front panel with a piezo sounder needs to be attached to the main board, or a sounder circuit needs to be attached to the PZO/PIEZO outputs on an expansion header.

24 Clear Screen

Sends ANSI escape sequences to clear and reset the terminal screen.

Syntax:

`cls`

Response:

None (new command prompt appears).

25 Reboot

Reboots the firmware.

Syntax:

`reboot` [`watchdog`]

Response:

None (new command prompt appears).

If the watchdog parameter is included (i.e. “`reboot watchdog`”) then the system will lock into an infinite, unresponsive loop whilst the watchdog timer expires. The expiration takes several seconds, after which the system will reboot.

If the reboot command is issued without a parameter, the reboot command is executed immediately.

After a reboot, the USB serial port connection to the host on U16 series devices is reset. This will result in a terminal emulator (e.g. PuTTY or ZTerm) shutting the connection. U8 series devices keep the serial connection to the host open during a reset.

At boot, a string of ANSI escape sequences is sent to the host to reset the terminal emulator. It is suggested that the host software discards everything up to and including the “`>>`” command prompt.

Although all the firmware is reset when a reboot command is issued, the USB port controllers are not *fully* reset, although they are returned to their default mode. To perform a full reset, power-cycle the U8-U16.

The `reboot` command is also present in boot mode. There it takes no parameter.

Rebooting sets the “R” (rebooted) flag, which is reported by the `health` and `state` commands.

26 Serial speed

Changes the current serial speed.

Syntax:

```
serial_speed  [test|fast|slow]
```

OK|Error (new command prompt appears).

To increase the serial UART speed to 1Mbaud use the following sequence:

`>> serial_speed  test`	
`OK`	OK is returned if the product supports the increase in speed
`>> serial_speed fast`	Serial speed is now increased. The next “>>” will be at 1Mbaud
`>> serial_speed fast`	The command must be repeated at 1Mbaud to confirm

It is suggested that the host flush the serial buffer after the first “`serial_speed fast`“ before the speed is changed to 1Mbaud. If the next “>>” is missed by the host this isn’t important. Just continue with the sequence.

If any error is detected in the above sequence the speed increase won’t occur or will be reset to 115200baud.

If during operation at 1Mbaud any serial errors are detected the speed is automatically dropped to 115200baud without warning. The host code must be aware of this and take suitable action. It is suggested if the link regularly fails not to try increase the speed again.

Before exiting the host should return the speed back to 115200baud with the following command

```
>> serial_speed  slow
```

Failure to do so when a subsequent connection is made at the default 115200baud will result in the first characters being lost until the U8-U16 detects the incorrect baud rate as serial errors and drops back to 115200baud.

27 Set delays

Changes internal delays

Syntax:

```
set_delays  <port_reset_delay_ms> <attach_blanking_ms> <deattach_count>
<deattach_sync_count> <attach_count>
```

(new command prompt appears).

Default values are 400 2000 30 14 14

The use of this command is not recommended and may prevent correct charging.

28 Live view

Live view provides a self-refreshing view of the port states and flags. Ports can be commanded using single key presses. To enter live view, use the '`l`' command.

Live view is designed for interactive use by humans, not computers. It makes extensive use of ANSI escape sequences to control the cursor position. Do not try to script the control of the live view - its commands and layout may change greatly between firmware versions.

The terminal size (rows, columns) must be large enough or the display will be corrupted. The U8-U16 attempts to set the number of rows and columns of the terminal when entering live view mode.

Example

```
cambrionix PP15S 15 Port USB Charge+Sync (live view)
Port  Flags   mA   State            Profile      Start    End    Energy

> 01  A       68   Sync
> 02           0   Charge (idle)
> 03           0   Charge (idle)
> 04           0   Charge (idle)
> 05           0   Charge (idle)
> 06           0   Charge (idle)
> 07           0   Charge (idle)
> 08  A       29   Charged          Profile 1     4493     365  14.02Wh
> 09           0   Charge (idle)
> 10           0   Charge (idle)
> 11           0   Charge (idle)
> 12           0   Charge (idle)
> 13           0   Charge (idle)
> 14           0   Charge (idle)
> 15           0   Charge (idle)

Host present: Yes
5V Rail 5.17V / 5.20V      12V Rail: 12.26V    Temperature: 46.7C
Total Current:    107mA     Total Power :   0W
Seconds since power on: 4877

Flags:     A:Attached, E:System Error, e:Port Error
Commands:  o)ff c)harge  s)ync  q)uit live view
           Type 2-digit port number (e.g. 01). / toggles all ports
           Selection: --
```

29 Console remote control

The LEDs, switches and LCD on the console can be controlled via terminal commands. This allows the firmware control of the console to be disabled and for the console to be controlled by the user's software instead. This might be useful where a single board computer or suchlike is connected to the U8-U16.

Entering remote control mode

To disable the console control from the firmware, and allow it to be controlled via the terminal, issue the `remote` command without parameters:

```
>> remote
```

The LEDs will be turned off when entering remote control mode. The LCD will be unaffected, and previous text will remain. Use `clcd` to clear the LCD (described later).

Leaving remote control mode

To leave remote control mode, and allow the console to be controlled by the firmware once more, issue the `remote exit` command:

```
>> remote exit
```

The LEDs will be reset and the LCD cleared when leaving remote control mode.

Leaving remote control mode when a key is pressed

Sometimes it can be useful to exit remote control mode when a console key is pressed. For example, you may want to display text on the LCD, but return back to the normal charging status screen when a key is pressed. The `kexit` parameter to `remote` tells the U8-U16 to enter remote control mode, but exit automatically when a console key is pressed:

```
>> remote kexit
```

Notes

1. In `remote kexit` mode, the `keys` command will not return key events.
2. It *is* permitted to move from `remote` mode into `remote kexit` mode, and vice-versa.
3. The key click sound still operates regardless of the remote mode.
4. Charging, syncing and security still operate in remote mode. However, their status will not be reported to the console, and the user will need to poll the status flags (using the `state` and `health` commands) to determine the system state.
5. If the `keys`, `lcd`, or `clcd` commands are issued when not in `remote` or `remote kexit` mode, then an error message will be shown, and the command will not be executed.
6. If the `ledb` or `leds` commands are issued when not in `remote` or `remote kexit` mode, then lower case parameters are ignored.

Writing to the LEDs

There are two methods to write to the LEDs in remote control mode: `ledb` and `leds`. First however, the operation of the LEDs will be described.

There are three LEDs per port. Each LED has a separate flash pattern assigned to it. The flash pattern is an 8 bit byte. Each bit is repeatedly scanned in sequence from MSB to LSB (i.e. left to right). A '1' bit turns the LED on, and a '0' turns it off. For example, a bit pattern of decimal 128 (binary 10000000b) would pulse the LED briefly. A bit pattern of decimal 127 (binary 01111111b) would see the LED on for most of the time, only turning off briefly.

`ledb` command

The `ledb` command can be used to assign a flash bit pattern to an individual LED.

Syntax:

`ledb` <port> <row> <ptn> [H | R]

<port> is the port number, starting at 1
<row> is the LED row number, starting at 1. Typically these are arranged as follows:

Row	LED Function
1	Charged
2	Charging
3	Sync mode

<ptn> can be specified as decimal (range 0..255), hexadecimal (range 00h to ffh) or binary (range 00000000b to 11111111b). Hexadecimal number must end with 'h'. Binary numbers must end with 'b'. More significant digits can be omitted for all radices. For example, '0b' is the same as '00000000b'. Hexadecimal numbers are not case-sensitive.

[H | R] optional parameters ‘H’ takes over control of the LED without a remote command. ‘R’ releases control of the LED back to normal operation.

Example

To flash the charging LED on port 8 at 50/50 duty cycle, use:

```
>> ledb 8 2 11110000b
```

To turn on the port 1 charged LED continuously (i.e. no flashing):

```
>> ledb 1 1 ffh
```

To turn off the port 1 sync LED:

```
>> ledb 1 3 0
```

`leds` command

The `leds` command can be used to assign a string of flash patterns to one row of LEDs. This is much faster for controlling an entire row of LEDs. Just three calls of the `leds` command can set all the LEDs on the system.

Syntax:

```
leds <row> [ptnstr]
```

<row> is address as for `ledb` above.

`[ptnstr]` is a string of characters, one per port, starting at port 1. Each character represents a different flash pattern to be assigned to the port. A string of 8 characters will assign flash patterns to 8 ports. The valid pattern characters are:

Pattern Character	LED function	Flash pattern
`0 (zero)`	Off	`00000000`
`1 (one)`	On continuously (not flashing)	`11111111`
`f`	Flash fast	`10101010`
`m`	Flash medium speed	`11001100`
`s`	Flash slowly	`11110000`
`p`	Single pulse	`10000000`
`d`	Double pulse	`10100000`
`O (upper case O)`	Off (no remote command needed)	`00000000`
`C`	On (no remote command needed)	`11111111`
`F`	Flash fast (no remote command needed)	`10101010`
`M`	Flash medium speed (no remote command needed)	`11001100`
`S`	Flash slowly (no remote command needed)	`11110000`
`P`	Single pulse (no remote command needed)	`10000000`
`D`	Double pulse (no remote command needed)	`10100000`
`R`	Release “no remote command needed “ LEDs back to normal use	
`x`	Unchanged	unchanged

The uppercase commands are useful if the default charging and sync behaviour is required but the host requires one or more LEDs to behave differently e.g. attract user attention to a particular port.

Example

To set up the following flash pattern on the row containing the 'charged' LEDs:

Port	LED Function
1	Unchanged
2	On
3	Flash fast
4	Single pulse
5	Off
6	On continuously
7	On continuously
8	Unchanged

Issue the command:

```
>> leds 1 x1fp011
```

Note that the first LED (port 1) needed to be skipped using the `x` character. Port 8 was not altered as the pattern string only contained 7 characters.

Notes

1. For both `ledb` and `leds` commands, the position in the flash pattern is synchronised across all LEDs. If no LEDs are lit, then the position in the flash pattern is reset and kept at the MSB. Therefore the first LED to be set to a flash pattern starts at the first bit (the MSB) in the flash pattern. Subsequent LEDs may start part-way through their flash patterns, depending on where the `ledb` or `leds` command was issued.
2. No harm or erroneous behaviour will result if the LEDs are written to, but do not physically exist.
3. The LED state is not re-established when `remote` mode is exited and then re-entered.

30 Reading the console key state

The U8-U16 has three console keys. These are debounced in software. When a key is pressed, a key 'click' flag is set. This flag remains set until it is read. To read the key click flags, use the `keys` command. The result is a comma-separated list, with one flag per key:

```
>> keys
1, 0, 0
>> keys
0, 0, 0
```

Keys A, B and C are listed respectively. A '1' means the key has been pressed since `keys` was last called. The flags are zeroed after `keys` is run:

Notes

`keys` only works in `remote` mode. It does not work in `remote kexit` mode.

31 Writing to the LCD

If an LCD is attached, it can be written to by the `lcd` command.

Syntax:

`lcd` <row> <col> <string>

<row> is 0 for the first row, and 1 for the second row.
<col> is the column number, starting at 0.
<string> gets displayed on the LCD. It may contain spaces before, within and after.

Example

To write “Hello, world” on the far left of the second row:

```
>> lcd 1 0 Hello, world!
```

Displaying Icons

As well as ASCII characters, the LCD can display several custom icons. These are accessed by sending the escape sequence <ESC> c, where c is the character '1' .. '8':

c	Icon
1	Empty battery
2	Continuously animated battery
3	Cambrionix filled 'o' glyph
4	Full battery
5	Padlock
6	Egg timer
7	Custom numeral 1 (aligned to right of bitmap)
8	Custom numeral 1 (aligned to middle of bitmap)

32 Clearing the LCD

The lcd may be cleared by using the `clcd` command.

33 Error reporting

Commands that succeed will output their required results, ending with a new command prompt. Failed commands will output an error code of the form “*Ennn: Error explanation”, followed by a new command prompt. “nnn” is always a three digit decimal number.

Example

Specifying a non-existent port to the `mode` command:

```
>> mode c 17
*E410: Port number must be 1..8
```

34 Fatal errors and the command prompt

When the system encounters a fatal error, the error is reported to the terminal immediately in the following format:

```
*FATAL ERROR Ennn: Error text
```

Ennn is a three-digit decimal error reference number. “Error text” describes the error.

The console LCD will display the error number. The console LEDs will show the error code in binary on the 'charged' LEDs.

After a fatal error has been reported the terminal will only be receptive to CTRL-C (ASCII decimal 3) and ENTER (ASCII decimal 13). If either of these are received, then the system will enter bootloader mode. If CTRL-C or ENTER are not received within the watchdog timeout period (approximately 9 seconds) then the system will reboot.

Important

If a fatal error occurs whilst a controlling script is sending a CTRL-C or ENTER character to the U8-U16, then bootloader mode will be entered. It is vital that the controlling script recognises bootloader mode, and knows how to exit it.

Bootloader mode is indicated by the prompt `boot>>` (sent on a new line)
The normal command prompt is `>>` (sent on a new line).

In bootloader mode, non-bootloader commands will be responded to with:

```
*E900: Invalid bootloader command
```

To exit bootloader mode, use the `reboot` command, and wait for the normal command prompt to return.

For testing purposes, bootloader mode can be entered by using the `boot` command.

35 Terminology

Term	Explanation
U8 devices	Any device in the U8 sub-series. E.g. U8C, U8C-EXT, U8S, U8S-EXT
U16 devices	Any device in the U16 sub-series. E.g. U16C, U16S
U8-U16	Any device in the Universal Series. E.g. U8C, U8C-EXT, U8S, U8S-EXT, U16C, U16S.
UART	Universal Asynchronous Receiver Transmitter. The hardware that drives a serial port connection.
VCP	Virtual COM port
/dev/	Devices directory on Linux and macOS
IC	Integrated Circuit
PWM	Pulse width modulation. The duty cycle is the percent of time the PWM is in the high (active) state
<CR>	Carriage return character (ASCII decimal 13)
<LF>	Line feed character (ASCII decimal 10)
Sync mode	Synchronisation mode (U8-U16 provides USB connection to host computer)
Console	The front-panel attached to the U8-U16. The Console provides the keys, piezo sounder and LCD
Terminal	Program that allows characters to be sent and received over a UART connection
Port	USB socket on the front of U8-U16 that is used to connect mobile devices.
MSB	Most significant bit
LSB	Least significant bit
<ESC>	Escape character (ASCII decimal 27)

36 Revision History

Revision	Date	Author	Comments
1.0	02-JUL-2012	AEM	Initial draft release, documented for firmware version 1.03
1.1	03-JUL-2012	AEM	Added comment on10ms resolution for beep timing
1.2	04-JUL-2012	AEM	Grammatical error corrected
1.3	15-JUL-2012	AEM	Added `sef`, `remote`, `ledb`, `leds`, `keys` and `lcd` commands. Corrected 400uA bias current to 2mA. Valid for firmware version 1.06
1.4	26-JUL-2012	ST	Removed references to the C3 product.
1.5	17-JUN-2013	DP	Updates to profiles and state command
1.6	14-NOV-2014	AJG	Added loge command and setting profile on mode c
1.7	02-FEB	DP	Added new commands (1.68 onwards) bd, serial_speed and set_delays. Documented new LED features. Added information on accuracy of voltage and current measurement. Minor edits
1.8	22-MAR-2018	AJG	Add new commands and correct phone number, merge two versions
1.9	21-FEB-2019	AJG	Add new fc field to id string and rebranded