Celium Devices

BE33 – AT Command Document

Overview:

BE-XX series of BLE modules are controlled by a simple, intutive serial ascii command interface. There are three different types of data sets that are exchanged between the module and host controller; **Commands**, **Responses** and **Events.**

1. **Commands:** These are used to command the ble module to perform a particular function. All the commands follow the structure described.

CMD+<command>=<parameter 1>,<parameter 2><CR><LF>

CMD – Op code to identify the data stream as a command
<command> - Name of command *@refer command list (table x)*
<parameters> - Parameters expected by the command *@refer command list (table x)*
<CR><LF> - Every command ends with "\r\n" characters.

There are 2 types of commands; SET commands and GET commands.

(a) **SET commands**: These commands are used to set a parameters for a particular function. eg. "*CMD+NAME=test123\r\n*". This command sets the name of the device to '*test123*'.
(b) GET commands: These commands are used to get default/stored parameters from the module. eg. "*CMD?NAME\r\n*" This command fetches the name of the module.

All the BE-XX modules responds to all the commands with status codes.

2. **Response:** The BE-XX modules sends out data packets in response to commands sent. All the reponses follow the structure described.

RSP=<status>,<paramter 1>,<parameter 2><CR><LF>

RSP – Op code to identify the data stream as a Response
<status> - Execution status of a particular command that was sent by the host controller. *@refer satus code list (table x)*
<parameters> - Any parameters requested by the host controller *@refer command list (table x)*
<CR><LF> - Every response ends with "\r\n" characters.

3**. Events:** BE-XX module generates events to notify the host controller of special conditions like "peer connection" , "disconnection", "data reception" etc. All the events follow the structure described

EVT+<event>=<parameter 1>,<parameter 2><CR><LF>

EVT – Op Code to identify the data stream as an Event
<event> - Name of the event @refer event list (table x)
<parameters> - Any parameters that would be needed by the host controller @refer event lsit (table x)
<CR><LF> - Every event ends with "\r\n" characters.

Default settings:

NAME – BE33-C/AT
BAUD – 115200
HWFC – Disable
PASSKEY – 123456
TXPWR – 0 dBm
LONG RANGE – Disabled

Pin Configuration:

Pin Number	Pin Name	Pin Designation	Functionality
10	GPIO05	P0.30/AIN6	Wake Up
17	GPIO09	P0.04/AIN2	UART RXD
18	GPIO10	P0.05/AIN3	UART TXD
19	GPIO11	P1.09	UART RTS
20	GPIO12	P0.11	UART CTS
23	GPIO13	P0.15	Peer Connection Indication
24	GPIO14	P0.17	Busy
29	GPIO15	P0.18	Reset

GATT Server Configuration:

The module uses custom serial profile mentioned below to facilitate the data transfer between the module and connected peer.

Service UUID: 6E 40 **00 01** B5 A3 F3 93 E0 A9 E5 0E 24 DC CA 9F
TX Char UUID: 6E 40 **00 02** B5 A3 F3 93 E0 A9 E5 0E 24 DC CA 9F
RX Char UUID: 6E 40 **00 03** B5 A3 F3 93 E0 A9 E5 0E 24 DC CA 9F

Command list

Set Commands

Sl No	Command	Description
1	CMD+NAME=<name><CR><LF>	Sets the name of the device. Eg. **CMD+NAME=test123\r\n**. Sets the name of the device to '*test123*' **Note:** For this to take effect, soft reset the module using the command "CMD+RESET=0\r\n" The device name length should not exceed 20 bytes. **Response:** **RSP=<status>\r\n** @refer status codes
2	CMD+BAUD=<baud><CR><LF>	Sets the baud rate of the module. Eg. **CMD+BAUD=0\r\n.** Sets the baud rate of the device. <baud> can take the following values. 0 – 9600 1 – 19200 2 – 56000 3 – 115200 4 – 230400 5 – 460800 6 – 921600 7 – 1000000 **Note:** For this to take effect, soft reset the module using the command "CMD+RESET=0\r\n" **Response:** **RSP=<status>\r\n** @refer status codes
3	CMD+HWFC=<hwfc><CR><LF>	Sets the hardware flow control for the UART interface. Eg. **CMD+HWFC=1\r\n** . Enables the HWFC <hwfc> can take the following values 0 – Disables HWFC 1 – Enable HWFC **Note:** For this to take effect, soft reset the module

		using the command “CMD+RESET=0\r\n” **<u>Response:</u>** **RSP=<status>\r\n** @refer status codes
4	CMD+ADV=<adv><CR><LF>	Starts / Stops bluetooth advertising. Eg. **CMD+ADV=1\r\n** starts bluetooth advertising. <adv> can take the following values 0 – Stops advertising 1 – Starts advertising **<u>Response:</u>** **RSP=<status>\r\n** @refer status codes
5	CMD+TXPWR=<txpwr><CR><LF>	Sets the bluetooth transmision power. Eg. **CMD+TXPWR=-4\r\n** Sets the bluetooth tx pwr to -4 dBm **<u>Note:</u>** Acceptable <txpwr> values are -40, -20, -16, -12, -8, -4, 0, 3, 4, 8 **<u>Response:</u>** **RSP=<status>\r\n** @refer status codes
6	CMD+RESET=<type><CR><LF>	Resets the module Eg. **CMD+RESET=0\r\n** Soft resets the BLE module. <type> can take the following values 0 – Soft Reset 1 – Factory Reset **<u>Note:</u>** Wait fot “*EVT+READY\r\n*” event from the module before sending any other commands. **<u>Response:</u>** If succesful this command does not return any response. If not succesful the module returns the following response. **RSP=<status>\r\n** @refer status codes

7	CMD+PIN=<passkey><CR><LF>	Sets the bluetooth passkey required for pairing. Eg. **CMD+PIN=000000\r\n** Sets passkey to *"000000"* **Note:** > The Passkey should be 0 - 9 in ASCII. The passkey should be 6 digits. > No peer should be connected while setting the passkey. > Default Passkey is **'123456'** **Response:** **RSP=<status>\r\n** @refer status codes
8	CMD+DISCON=<conn_handle><CR><LF>	Disconnects a connected peer. Eg. **CMD+DISCON=<conn_handle>\r\n** **Response:** **RSP=<status>\r\n** @refer status codes
9	CMD+DATA=<conn_handle>,<data><CR><LF>	Sends data to a connected peer. Eg. **CMD+DATA=<conn_handle>,hi\r\n** Sends "hi" to the connected peer. **Note:** The peer should be connected and should have notifications enabled. The length of data should not exceed 192 bytes. **Response:** **RSP=<status>\r\n** @refer status codes
10	CMD+DELPEERS=<parameter><CR><LF>	Delete bonded peers Eg. **CMD+DELPEERS=0\r\n** **Note:** <parameter> is a reserved parameter and should always be set to 0. **Response:** **RSP=<status>\r\n** @refer status codes
11	CMD+SYSOFF=<parameter><CR><LF>	Puts the module in deep sleep Eg. **CMD+SYSOFF=0\r\n** **Note:**

		<parameter> is a reserved parameter and should always be set to 0. The module wakes up when the WAKE_UP pin is pulled low. **Response:** If succesful this command does not return any response. If not succesful the module returns the following response. **RSP=<status>\r\n** @refer status codes
12	CMD+SCAN=<start/stop><CR><LF>	Command to start/stop scanning Eg. **CMD+SCAN=1\r\n** **Note:** <start/stop> - 0: stop scanning, 1:start scanning **Response:** **RSP=<status>\r\n** @refer status codes Upon successful execution, EVT+ADVRPT event is generated when advertisement packets are recieved.
13	CMD+SCANPARAM=<active_scan>,<scan_window>,<scan_interval>,<scan_timeout><CR><LF>	Command to set scan parameters Eg. **CMD+SCANPARAM=0,50,100,10000\r\n** active_scan – False scan_window – 50ms scan_interval – 100ms scan_timeout – 10000ms (10Secs) **Note:** Default Scan Parameters active_scan – True, scan window – 500ms scan interval – 500ms scan_timeout – 0 (No timeout, scans till you stop) **Response:** **RSP=<status>\r\n** @refer status codes
14	CMD+CON=<addr_type>,<add><CR><LF>	Command to connect to peripheral devices using their mac address.

		Eg. **CMD+CON=1,0154abde1209\r\n** <addr_type> - 1, random address. The user can get the peer address type from EVT+ADVRPT. <addr> - peer address, 01:54:ab:de:12:09 **Response:** **RSP=<status>\r\n** @refer status codes
15	CMD+LREN=<en/dis><CR><LF>	Command to enable/disable long range mode. Eg. **CMD+LREN=1\r\n** 1 – Enable, 0 – Disable **Note:** Reset the module after enabling long range **Response:** **RSP=<status>\r\n** @refer status codes

Get Commands:

Sl No	Command	Description
1	CMD?NAME<CR><LF>	Gets the device name Eg. **CMD?NAME\r\n** If successful, returns a response with success status and device name as a parameter. **RSP=0,BE33-C/AT\r\n** Else, the module returns only the status code. **RSP=<status>\r\n** *@refer status codes*
2	CMD?ADDR<CR><LF>	Gets the mac address of the bluetooth module Eg. **CMD?ADDR\r\n** If successful, returns a response with success status and mac address as a parameter. **RSP=0,5B252135FECD\r\n**
3	CMD?BAUD<CR><LF>	Gets UART baud rate Eg. **CMD?BAUD\r\n** If successful, returns a response with success status and baud rate as a parameter. **RSP=0,<baud>\r\n** <baud> can take the following values. 0 – 9600 1 – 19200 2 – 56000 3 – 115200 4 – 230400 5 – 460800 6 – 921600 7 – 1000000 Else, the module returns only the status code. **RSP=<status>\r\n** *@refer status codes*
4	CMD?HWFC<CR><LF>	Gets HWFC Eg. **CMD?HWFC\r\n** If successful, returns a repose with success status and HWFC as a parameter. **RSP=0,<hwfc>\r\n** <hwfc> can take the following values 0 – HWFC is diabled

		1 – HWFC is enabled Else, the module returns only the status code. **RSP=<status>\r\n** *@refer status codes*
5	CMD?BLEHWVER<CR><LF>	Gets the hardware version of the module Eg. **CMD?BLEHWVER\r\n** If successful, returns a repose with success status and BLE hardware version as a parameter. **RSP=0,<blehwver>\r\n** Else, the module returns only the status code. **RSP=<status>\r\n** *@refer status codes*
6	CMD?BLEFWVER<CR><LF>	Gets the firmware version of the module Eg. **CMD?BLEFWVER\r\n** If successful, returns a repose with success status and BLE hardware version as a parameter. **RSP=0,<blefwver>\r\n** Else, the module returns only the status code. **RSP=<status>\r\n** *@refer status codes*
7	CMD?STATUS<CR><LF>	Gets the operational status of the module Eg. **CMD?STATUS\r\n** If successful, returns a respose with success status and ‘operatinal status’as a parameter. **RSP=0,0\r\n** – If the module is IDLE **RSP=0,1\r\n** – If the module is advertising **RSP=0,2\r\n** – If the module is connected Else, the module returns only the status code. **RSP=<status>\r\n** *@refer status codes*
8	CMD?TXPWR<CR><LF>	Gets the TXPWR of the module Eg. **CMD?TXPWR\r\n** If successful, return a response with success status code and txpwr as a parameter. **RSP=0,<txpwr><CR><LF>** Else, the module returns only the status code. **RSP=<status>\r\n** *@refer status codes*
9	CMD?PIN<CR><LF>	Gets bluetooth pairing pin

		Eg. **CMD?PIN\r\n** If successful, returns a response with success status code and pin as a parameter. **RSP=0,123456\r\n** Else, the module returns only the status code. **RSP=<status>\r\n *@refer status codes***
10	CMD?SCANPARAM<CR><LF>	Command to get scan parameters Eg. **CMD?SCANPARAM\r\n** If successful, returns a response with success status code and scan parameters. **RSP=0,<active_scan>,<scan_window>,<scan_interval>,<scan_timeout><CR><LF>** Else, the module return only the status code **RSP=<status>\r\n *@refer status codes***
11	CMD?LREN<CR><LF>	Command to get long range status Eg. **CMD?LREN\r\n** If successful, returns a response with success status code and long range mode. **RSP=0,1\r\n** Else, the module return only the status code **RSP=<status>\r\n *@refer status codes***

Events:

Sl No	Events	Description
1	EVT+READY<CR><LF>	This event indicates that the module has completed the initial configuration and is now avaible to recieve commands. This event typically occurs on power cycle, soft reset, factory reset and on waking up from deep sleep.
2	EVT+CON=<conn_handle>,<role>,<addr_type>,<peer address><CR><LF>	This event occurs when the module establishes a connection with a peer. <conn_handle> - This parameter identifies the connected peer. <role> - This parameter gives the role of the connected peer. 0 – Invalid 1 – Peripheral 2 – Central <addr_type> - This parameter gives the address type of the connected peer. 0 – Public 1 – Random Static 2 – Random Private Resolvable 3 – Random Private Non-Resolvable 4 – Anonymous 9 – Invalid <peer_addr> - Peer mac address **Note:** <peer_address> is a 6 byte hexadeciamal value not an ascii value
3	EVT+DISCON=<conn_handle>,<role>,<addr_type>,<peer address><CR><LF>	This event occurs when the module disconnects from a peer. <conn_handle> - This parameter identifies the connected peer. <role> - This parameter gives the role of the connected peer. 0 – Invalid 1 – Peripheral 2 – Central <addr_type> - This parameter gives the address type of the connected peer. 0 – Public 1 – Random Static 2 – Random Private Resolvable 3 – Random Private Non-Resolvable

		4 – Anonymous 9 – Invalid <peer_addr> - Peer mac address **Note:** <peer_address> is a 6 byte hexadeciamal value not an ascii value
4	EVT+TIMEOUT<CR><LF>	This event occurs when internal bluetooth processess have timed out.
5	EVT+NOTIFY=<parameter><CR><LF>	This event occurs when the peer enables or disables notifications. **EVT+NOTIFY=0\r\n** – Notification disabled **EVT+NOTIFY=1\r\n** – Notification enabled
6	EVT+DATA=<conn_handle>,<data><CR><LF>	This event occurs when the module receives data from the peer. <conn_handle> - Connection handle of the peer from which the data is received.
7	EVT+BONDED=<parameter><CR><LF>	This event occurs when the module bonds with a peer. EVT+BONDED=0\r\n – Bonding unsuccessful EVT+BONDED=1\r\n – Bonding successful.
8	EVT+MEMFULL<CR><LF>	This event is generated when the internal memory is not sufficient to store additional peers. use **CMD+DELPEERS=0\r\n** to delete all the bonded peers to make room for new ones.
9	EVT+ADVEND<CR><LF>	This event is generated when the advertising terminates.
10	EVT+ADVRPT=<peer_addr_type>,<peer_addr>,<name>,<rssi><CR><LF>	This event is generated when the advertising packets are found when the user starts scanning. **<pee_addr_type>** 0 – Public 1 – Random Static 2 – Private Resolvable 3 – Private Non Resolvable 127 – Anonymous **<peer_address>** - peer mac address **<name>** - peer name, if the name is part of the advertising packet **<rssi>** - signal strength

Status Codes:

Status codes are part of ‘response’ data set. The status codes are in ASCII format.

Sl No	Status Code	Description
1	0	SUCCESS
2	1	INVALID_COMMAND
3	2	INVALID_PARAMETER
4	3	INVALID_DATASIZE
5	4	INVALID_STATE
6	5	BUSY
7	6	INTERNAL_ERROR
8	7	INVALID_ATHORIZATION
9	8	TIMEOUT
10	9	SYSOFF_FAILED
11	10	NO_MEMORY
12	11	INVALID_CONN_HANDLE
13	12	EXCEEDED_MAX_CONN_CNT
14	13	INVALID_OPCODE
15	14	INVALID_OPERAND
12	99	UNKOWN_ERROR