

quectel_bc66_drv

Generated by Doxygen 1.9.1

| | |
|---|-----------|
| 1 Data Structure Index | 1 |
| 1 Data Structure Index | 1 |
| 1.1 Data Structures | 1 |
| 2 File Index | 1 |
| 2.1 File List | 1 |
| 3 Data Structure Documentation | 2 |
| 3.1 bc66_at_cmd_t Struct Reference | 2 |
| 3.1.1 Detailed Description | 2 |
| 3.1.2 Field Documentation | 2 |
| 3.2 bc66_obj_t Struct Reference | 3 |
| 3.2.1 Field Documentation | 3 |
| 4 File Documentation | 5 |
| 4.1 /Users/jcbecerra/dev/fw/iot/quectel_bc66_driver/src/bc66_drv.c File Reference | 5 |
| 4.1.1 Detailed Description | 7 |
| 4.1.2 Macro Definition Documentation | 7 |
| 4.1.3 Enumeration Type Documentation | 8 |
| 4.1.4 Function Documentation | 8 |
| 4.1.5 Variable Documentation | 13 |
| 4.2 /Users/jcbecerra/dev/fw/iot/quectel_bc66_driver/src/bc66_drv.h File Reference | 13 |
| 4.2.1 Detailed Description | 14 |
| 4.2.2 Enumeration Type Documentation | 15 |
| 4.2.3 Function Documentation | 16 |
| Index | 23 |

1 Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

| | |
|-------------------------------|----------|
| bc66_at_cmd_t | |
| BC66 Command struct | 2 |
| bc66_obj_t | 3 |

2 File Index

2.1 File List

Here is a list of all files with brief descriptions:

[/Users/jcbecerra/dev/fw/iot/quectel_bc66_driver/src/bc66_drv.c](#)
MIT License

5

[/Users/jcbecerra/dev/fw/iot/quectel_bc66_driver/src/bc66_drv.h](#)
MIT License

13

3 Data Structure Documentation

3.1 bc66_at_cmd_t Struct Reference

BC66 Command struct.

Data Fields

- const char * [cmd](#)
at command sentence
- [cmd_flg_t](#) [cmd_flags](#)
flags for command implementation (see
- char * [cmd_rsp](#)
expected command response
- [uint32_t](#) [rsp_timeout](#)
response timeout [ms]

3.1.1 Detailed Description

BC66 Command struct.

3.1.2 Field Documentation

3.1.2.1 [cmd](#) const char* cmd

at command sentence

3.1.2.2 [cmd_flags](#) [cmd_flg_t](#) cmd_flags

flags for command implementation (see
[flags](#) [enum](#))

3.1.2.3 cmd_rsp char* cmd_rsp

expected command response

3.1.2.4 rsp_timeout uint32_t rsp_timeout

response timeout [ms]

The documentation for this struct was generated from the following file:

- /Users/jcbecerra/dev/fw/iot/quectel_bc66_driver/src/bc66_drv.c

3.2 bc66_obj_t Struct Reference

```
#include <bc66_drv.h>
```

Data Fields

- void(* [func_init_ptr](#))()
uart initialize function pointer
- void(* [func_delay](#))(uint32_t t)
delay function pointer
- int(* [func_w_bytes_ptr](#))(uint8_t *txc, uint16_t len)
write bytes function pointer
- int(* [func_r_bytes_ptr](#))(uint8_t *rxc, uint16_t size)
read one-byte function pointer
- struct {
 void(* [MDM_PSM_EINT_N](#))(size_t pin_value)
 delay function pointer
 void(* [MDM_PWRKEY_N](#))(size_t pin_value)
 modem power key function pointer
 void(* [MDM_RESET_N](#))(size_t pin_value)
 modem reset function pointer
 void(* [MDM_RI](#))()
 modem ring interrupt function pointer
} [control_lines](#)

3.2.1 Field Documentation**3.2.1.1** struct { ... } control_lines

3.2.1.2 func_delay void(* func_delay) (uint32_t t)

delay function pointer

3.2.1.3 func_init_ptr void(* func_init_ptr) ()

uart initialize function pointer

3.2.1.4 func_r_bytes_ptr int(* func_r_bytes_ptr) (uint8_t *rxc, uint16_t size)

read one-byte function pointer

3.2.1.5 func_w_bytes_ptr int(* func_w_bytes_ptr) (uint8_t *txc, uint16_t len)

write bytes function pointer

3.2.1.6 MDM_PSM_EINT_N void(* MDM_PSM_EINT_N) (size_t pin_value)

delay function pointer

3.2.1.7 MDM_PWRKEY_N void(* MDM_PWRKEY_N) (size_t pin_value)

modem power key function pointer

3.2.1.8 MDM_RESET_N void(* MDM_RESET_N) (size_t pin_value)

modem reset function pointer

3.2.1.9 MDM_RI void(* MDM_RI) ()

modem ring interrupt function pointer

The documentation for this struct was generated from the following file:

- [/Users/jcbecerra/dev/fw/iot/quectel_bc66_driver/src/bc66_drv.h](#)

4 File Documentation

4.1 /Users/jcbecerra/dev/fw/iot/quectel_bc66_driver/src/bc66_drv.c File Reference

MIT License.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include "bc66_drv.h"
```

Data Structures

- struct [bc66_at_cmd_t](#)
BC66 Command struct.

Macros

- #define [CMD_END_LINE](#) "\r\n"
End of line command chars.
- #define [RSP_OK](#) "\r\nOK\r\n"
Ok response.
- #define [RSP_ERROR](#) "\r\nERROR\r\n"
Error response.
- #define [RSP_END_OF_LINE](#) "\r\n"
End of line response chars.
- #define [RSP_TIMEOUT](#) "BC66_TIMEOUT\r\n"
Answer when a timeout is occurred.
- #define [RSP_NO_CMD_IMPLEMENTED](#) "BC66_NO_CMD\r\n"
The command is not implemented.
- #define [MAX_RSP_SIZE](#) 64
Max AT response size.

Enumerations

- enum [cmd_flg_t](#) { [TEST](#) = 0x1 , [READ](#) = 0x2 , [WRITE](#) = 0x4 , [EXE](#) = 0x8 }
Command possibilities indicator flags.

Functions

- void `bc66_init (bc66_obj_t *bc66_obj)`
Function to initialize bc66 object.
- void `bc66_deinit (bc66_obj_t *bc66_obj)`
Function to initialize bc66 object.
- `bc66_ret_t bc66_send_at_command (bc66_cmd_type_t cmd_type, const bc66_cmd_list_t cmd_lst, const char *exp_rsp, const char *arg_fmt,...)`
Function to send at command sentence to bc66 module through an external function communication.
- char * `bc66_get_at_response (char *rsp)`
Function to get any response stored in the RX buffer.
- void `bc66_reset (void)`
Reset the module when PIN is low.
- void `bc66_power_on ()`
Pull down PWRKEY to turn on the module.
- void `bc66_power_off ()`
Pull up PWRKEY to turn off the module.
- char * `bc66_get_last_response (void)`
Function to get last modem response.
- bool `bc66_send_cmd_AT (void)`
Send AT command to sync baud rate.
- `bc66_ret_t bc66_set_echo_mode (bool echo)`
Set Command Echo Mode.
- `bc66_ret_t bc66_set_power_saving_mode (int mode)`
Power Saving Mode Setting (PSM).
- `bc66_ret_t bc66_set_psd_conn (pdp_type_t pdp_type, const char *apn, const char *user, const char *pass)`
Set Default PSD Connection.
- `bc66_ret_t bc66_set_mqtt_parameters (uint16_t keepalive, bool dataformat, bool session, bool version)`
Used to configure optional parameters of MQTT.
- `bc66_ret_t bc66_open_net_mqtt_client (const char *server_ip, uint16_t server_port)`
Open a Network for MQTT Client.
- `bc66_ret_t bc66_connect_mqtt_client (const char *client_id, const char *user, const char *pass)`
Connect a Client to MQTT Server.
- `bc66_ret_t bc66_disconn_mqtt_client (void)`
Disconnect a Client from MQTT Server.
- `bc66_ret_t bc66_publish_msg_mqtt (const char *topic, const char *msg, int qos)`
Publish Messages.

Variables

- const `bc66_at_cmd_t bc66_cmds_list []`
Define AT commands list: order must be equal to commands definition enum bc66_cmd_list_t.

4.1.1 Detailed Description

MIT License.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright

Juan Cruz Becerra

BC66 NB-IoT modem driver. (<https://www.quectel.com/product/bc66.htm>)

AT Command Syntax The AT or at prefix must be set at the beginning of each command line. Entering <CR> will terminate a command line. Commands are usually followed by a response that includes <CR><LF><response><CR><LF>. Throughout this document, only the responses are presented, <CR><LF> are omitted intentionally.

Types of AT Commands and Responses

- Test Command AT+<x>=?
- Read Command AT+<x>?
- Write Command AT+<x>=<n>
- Execution Command AT+<x>

Date

03/15/2021

Author

Eng. Juan Cruz Becerra

Version

1.0.0

4.1.2 Macro Definition Documentation

4.1.2.1 CMD_END_LINE `#define CMD_END_LINE "\r\n"`

End of line command chars.

4.1.2.2 MAX_RSP_SIZE `#define MAX_RSP_SIZE 64`

Max AT response size.

4.1.2.3 RSP_END_OF_LINE `#define RSP_END_OF_LINE "\r\n"`
End of line response chars.

4.1.2.4 RSP_ERROR `#define RSP_ERROR "\r\nERROR\r\n"`
Error response.

4.1.2.5 RSP_NO_CMD_IMPEMENTED `#define RSP_NO_CMD_IMPEMENTED "BC66_NO_CMD\r\n"`
The command is not implemented.

4.1.2.6 RSP_OK `#define RSP_OK "\r\nOK\r\n"`
Ok response.

4.1.2.7 RSP_TIMEOUT `#define RSP_TIMEOUT "BC66_TIMEOUT\r\n"`
Answer when a timeout is occurred.

4.1.3 Enumeration Type Documentation

4.1.3.1 cmd_flg_t `enum cmd_flg_t`
Command possibilities indicator flags.

Enumerator

| | |
|-------|----------------------------------|
| TEST | Command has test possibility. |
| READ | Command has read possibility. |
| WRITE | Command has write possibility. |
| EXE | Command has execute possibility. |

4.1.4 Function Documentation

4.1.4.1 bc66_connect_mqtt_client() `bc66_ret_t bc66_connect_mqtt_client (`
`const char * client_id,`
`const char * user,`
`const char * pass)`

Connect a Client to MQTT Server.

Parameters

| | |
|------------------|--|
| <i>client_id</i> | : The client identifier. The max length is 128 bytes. |
| <i>user</i> | : User name of the client. It can be used for authentication. The max length is 256 bytes. |
| <i>pass</i> | : Password corresponding to the user name of the client. It can be used for authentication. The max length is 256 bytes. |

Returns

See `bc66_ret_t` return codes.

4.1.4.2 bc66_deinit() `void bc66_deinit (`
`bc66_obj_t * bc66_obj)`

Function to initialize bc66 object.

Parameters

| | |
|-----------------|--|
| <i>bc66_obj</i> | |
|-----------------|--|

4.1.4.3 bc66_disconn_mqtt_client() `bc66_ret_t bc66_disconn_mqtt_client (`
`void)`

Disconnect a Client from MQTT Server.

Used when a client requests a disconnection from MQTT server. A DISCONNECT message is sent from the client to the server to indicate that it is about to close its TCP/IP connection.

Returns

See `bc66_ret_t` return codes.

4.1.4.4 bc66_get_at_response() `char* bc66_get_at_response (`
`char * rsp)`

Function to get any response stored in the RX buffer.

Parameters

| | |
|------------|-------------------|
| <i>rsp</i> | : response to get |
|------------|-------------------|

Returns

Response if found, NULL otherwise

4.1.4.5 bc66_get_last_response() `char* bc66_get_last_response (`
`void)`

Function to get last modem response.

If send a new AT command, the buffer which contain the last response will be erased.

Returns

Pointer to RX buffer with last response.

4.1.4.6 bc66_init() `void bc66_init (`
`bc66_obj_t * bc66_obj)`

Function to initialize bc66 object.

Parameters

| | |
|-----------------|--|
| <i>bc66_obj</i> | |
|-----------------|--|

4.1.4.7 bc66_open_net_mqtt_client() `bc66_ret_t bc66_open_net_mqtt_client (`
`const char * server_ip,`

```
uint16_t server_port )
```

Open a Network for MQTT Client.

Parameters

| | |
|--------------------|----------------------------|
| <i>server_ip</i> | : server ip (string) |
| <i>server_port</i> | : server port (0 to 65535) |

Returns

See `bc66_ret_t` return codes.

4.1.4.8 `bc66_power_off()` `void bc66_power_off ()`

Pull up PWRKEY to turn off the module.

4.1.4.9 `bc66_power_on()` `void bc66_power_on ()`

Pull down PWRKEY to turn on the module.

4.1.4.10 `bc66_publish_msg_mqtt()` `bc66_ret_t bc66_publish_msg_mqtt (` `const char * topic,` `const char * msg,` `int qos)`

Publish Messages.

Used to publish messages by a client to a server for distribution to interested subscribers.

Parameters

| | |
|--------------|--|
| <i>topic</i> | : Topic that the client wants to subscribe to or unsubscribe from. The maximum length is 255 bytes. |
| <i>msg</i> | : The message that needs to be published. The maximum length is 700 bytes. If in data mode (after > is responded), the maximum length is 1024 bytes |
| <i>qos</i> | : Integer type. The QoS level at which the client wants to publish the messages. <ul style="list-style-type: none">• 0 At most once• 1 At least once• 2 Exactly once |

Returns

See `bc66_ret_t` return codes.

4.1.4.11 `bc66_reset()` `void bc66_reset (` `void)`

Reset the module when PIN is low.

4.1.4.12 `bc66_send_at_command()` `bc66_ret_t bc66_send_at_command (` `bc66_cmd_type_t cmd_type,` `const bc66_cmd_list_t cmd_lst,` `const char * exp_rsp,`

```
const char * arg_fmt,
... )
```

Function to send at command sentence to bc66 module through an external function communication.

Parameters

| | |
|-----------------|---|
| <i>cmd_type</i> | : BC66_CMD_TEST, BC66_CMD_READ, BC66_CMD_WRITE or BC66_CMD_EXE type. |
| <i>cmd_lst</i> | : command to send (see command list). |
| <i>rsp</i> | : pointer to expected response text. |
| <i>arg_fmt</i> | : arguments format (like printf function) and must be send all arguments too. |

Returns

See `bc66_ret_t` return codes.

4.1.4.13 `bc66_send_cmd_AT()` `bool bc66_send_cmd_AT (`
`void)`

Send AT command to sync baud rate.

Returns

See `bc66_ret_t` return codes.

4.1.4.14 `bc66_set_echo_mode()` `bc66_ret_t bc66_set_echo_mode (`
`bool echo)`

Set Command Echo Mode.

This Execution Command determines whether or not the UE echoes characters received from external MCU during command state.

The command takes effect immediately. Remain valid after deep-sleep wakeup. The configuration will be saved to NVRAM (should execute AT&W after this command is issued).

Parameters

| | |
|-------------|--|
| <i>echo</i> | <ul style="list-style-type: none"> • false: Echo mode OFF • true: Echo mode ON |
|-------------|--|

Returns

See `bc66_ret_t` return codes.

4.1.4.15 `bc66_set_mqtt_parameters()` `bc66_ret_t bc66_set_mqtt_parameters (`
`uint16_t keepalive,`
`bool dataformat,`
`bool session,`
`bool version)`

Used to configure optional parameters of MQTT.

Parameters

| | |
|-------------------|---|
| <i>keepalive</i> | : Configure the keep-alive time. The range is 0-3600. The default value is 120. Unit: second. It defines the maximum time interval between messages received from a client. If the server does not receive a message from the client within 1.5 times of the keep-alive time period, it disconnects the client as if the client has sent a DISCONNECT message. 0 The client is not disconnected |
| <i>dataformat</i> | : The format of sent and received data. <ul style="list-style-type: none"> • 0 Text format • 1 Hex format |
| <i>session</i> | : The session type. <ul style="list-style-type: none"> • 0 The server must store the subscriptions of the client after it is disconnected. • 1 The server must discard any previously maintained information about the client and treat the connection as "clean". |
| <i>version</i> | : The version of MQTT protocol. <ul style="list-style-type: none"> • 0 MQTT v3.1 • 1 MQTT v3.1.1 |

Returns

See `bc66_ret_t` return codes.

4.1.4.16 `bc66_set_power_saving_mode()` `bc66_ret_t bc66_set_power_saving_mode (`
`int mode)`

Power Saving Mode Setting (PSM).

Power Saving Mode Setting.

Parameters

| | |
|-------------|--|
| <i>mode</i> | Integer type. Disable or enable the use of PSM in the UE <ul style="list-style-type: none"> • 0 Disable the use of PSM • 1 Enable the use of PSM • 2 Disable the use of PSM and discard all parameters for PSM or, if available, reset to the default values. |
|-------------|--|

Returns

See `bc66_ret_t` return codes.

4.1.4.17 `bc66_set_psd_conn()` `bc66_ret_t bc66_set_psd_conn (`
`pdp_type_t pdp_type,`
`const char * apn,`
`const char * user,`
`const char * pass)`

Set Default PSD Connection.

This command sets the PSD connection settings for PDN connection on power-up. When attaching to the NB-IoT network on power-on, a PDN connection setup must be performed. In order to allow this to happen, PDN connection settings must be stored in NVRAM, thus making it to be used by the modem during the attach procedure.

Parameters

| | |
|-----------------|--|
| <i>pdp_type</i> | : Specify the type of packet data protocol. |
| <i>apn</i> | : A logical name that is used to select the GGSN or the external packet data network. The maximum configurable APN length is 99 bytes. |
| <i>user</i> | : The user name for accessing to the IP network. (Optional) |
| <i>pass</i> | : The password for accessing to the IP network. (Optional) |

Returns

See `bc66_ret_t` return codes.

4.1.5 Variable Documentation

4.1.5.1 `bc66_cmds_list` `const bc66_at_cmd_t bc66_cmds_list[]`

Define AT commands list: order must be equal to commands definition enum `bc66_cmd_list_t`.

4.2 /Users/jcbecerra/dev/fw/iot/quectel_bc66_driver/src/bc66_drv.h File Reference

MIT License.

```
#include <stdint.h>
#include <stdbool.h>
#include <stddef.h>
```

Data Structures

- struct `bc66_obj_t`

Enumerations

- enum `bc66_cmd_type_t` { `BC66_CMD_TEST` , `BC66_CMD_READ` , `BC66_CMD_WRITE` , `BC66_CMD_EXE` }
- AT command possibility. Erch command can test and/or read and/or write and/or execute. Use with `bc66_send_at_command(...)` function.*
- enum `bc66_cmd_list_t` {
`bc66_cmd_list_AT` , `bc66_cmd_list_ATI` , `bc66_cmd_list_ATE` , `bc66_cmd_list_CEREG` ,
`bc66_cmd_list_CESQ` , `bc66_cmd_list_CGATT` , `bc66_cmd_list_CGPADDR` , `bc66_cmd_list_QCGDEFCONT`
, `bc66_cmd_list_CIMI` , `bc66_cmd_list_CPIN` , `bc66_cmd_list_CPSMS` , `bc66_cmd_list_QNBIOTEVENT` ,
`bc66_cmd_list_QMTCFG` , `bc66_cmd_list_QMTOPE` , `bc66_cmd_list_QMTCLOSE` , `bc66_cmd_list_QMTCONN`
, `bc66_cmd_list_QMTDISC` , `bc66_cmd_list_QMTSUB` , `bc66_cmd_list_QMTUNS` , `bc66_cmd_list_QMTPUB`
, `bc66_cmd_list_size` }
- This is the commands implemented list.*
- enum `bc66_ret_t` {
`bc66_ret_success` , `bc66_ret_timeout` , `bc66_ret_error` , `bc66_ret_out_of_range` ,
`bc66_ret_not_init` , `bc66_ret_no_cmd_implemented` }
- bc66 library api return*
- enum `pdp_type_t` { `pdp_type_ip` , `pdp_type_ipv6` , `pdp_type_ipv4v6` , `pdp_type_non_ip` }
- Enumeration to specify the type of packet data protocol.*

Functions

- void `bc66_init (bc66_obj_t *bc66_obj)`
Function to initialize bc66 object.
- char * `bc66_get_at_response (char *rsp)`
Function to get any response stored in the RX buffer.
- `bc66_ret_t bc66_send_at_command (bc66_cmd_type_t cmd_type, const bc66_cmd_list_t cmd_lst, const char *exp_rsp, const char *arg_fmt,...)`
Function to send at command sentence to bc66 module through an external function communication.
- void `bc66_reset (void)`
Reset the module when PIN is low.
- void `bc66_power_on ()`
Pull down PWRKEY to turn on the module.
- void `bc66_power_off ()`
Pull up PWRKEY to turn off the module.
- char * `bc66_get_last_response (void)`
Function to get last modem response.
- bool `bc66_send_cmd_AT (void)`
Send AT command to sync baud rate.
- `bc66_ret_t bc66_set_echo_mode (bool echo)`
Set Command Echo Mode.
- `bc66_ret_t bc66_set_power_saving_mode (int mode)`
Power Saving Mode Setting.
- `bc66_ret_t bc66_set_psd_conn (pdp_type_t pdp_type, const char *apn, const char *user, const char *pass)`
Set Default PSD Connection.
- `bc66_ret_t bc66_set_mqtt_parameters (uint16_t keepalive, bool dataformat, bool session, bool version)`
Used to configure optional parameters of MQTT.
- `bc66_ret_t bc66_open_net_mqtt_client (const char *server_ip, uint16_t server_port)`
Open a Network for MQTT Client.
- `bc66_ret_t bc66_connect_mqtt_client (const char *client_id, const char *user, const char *pass)`
Connect a Client to MQTT Server.
- `bc66_ret_t bc66_disconn_mqtt_client (void)`
Disconnect a Client from MQTT Server.
- `bc66_ret_t bc66_publish_msg_mqtt (const char *topic, const char *msg, int qos)`
Publish Messages.

4.2.1 Detailed Description

MIT License.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Copyright

Juan Cruz Becerra

BC66 NB-IoT modem driver. (<https://www.quectel.com/product/bc66.htm>)

AT Command Syntax The AT or at prefix must be set at the beginning of each command line. Entering <CR> will terminate a command line. Commands are usually followed by a response that includes <CR><LF>. Throughout this document, only the responses are presented, <CR><LF> are omitted intentionally.

Types of AT Commands and Responses

- Test Command AT+<x>=?
- Read Command AT+<x>?
- Write Command AT+<x>=<n>
- Execution Command AT+<x>

Date

03/15/2021

Author

Eng. Juan Cruz Becerra

Version

1.0.0

4.2.2 Enumeration Type Documentation**4.2.2.1 bc66_cmd_list_t** enum `bc66_cmd_list_t`

This is the commands implemented list.

Enumerator

| | |
|--|---|
| <code>bc66_cmd_list_AT</code> | AT command. Use to sync baud rate. |
| <code>bc66_cmd_list_ATI</code> | Display Product Identification Information. |
| <code>bc66_cmd_list_ATE</code> | Set Command Echo Mode. |
| <code>bc66_cmd_list_CEREG</code> | EPS Network Registration Status. |
| <code>bc66_cmd_list_CESQ</code> | Extended Signal Quality. |
| <code>bc66_cmd_list_CGATT</code> | PS Attachment or Detachment. |
| <code>bc66_cmd_list_CGPADDR</code> | Show PDP Addresses. |
| <code>bc66_cmd_list_QCGDEFCONT</code> | Set Default PSD Connection Settings. |
| <code>bc66_cmd_list_CIMI</code> | Request International Mobile Subscriber Identity. |
| <code>bc66_cmd_list_CPIN</code> | Enter PIN. |
| <code>bc66_cmd_list_CPSMS</code> | Power Saving Mode Setting. |
| <code>bc66_cmd_list_QNBIOTEVENT</code> | Enable/Disable NB-IoT Related Event Report. |
| <code>bc66_cmd_list_QMTCFG</code> | Configure Optional Parameters of MQTT. |
| <code>bc66_cmd_list_QMTOOPEN</code> | Open a Network for MQTT Client. |
| <code>bc66_cmd_list_QMTCLOSE</code> | Close a Network for MQTT Client. |
| <code>bc66_cmd_list_QMTCONN</code> | Connect a Client to MQTT Server. |
| <code>bc66_cmd_list_QMTDISC</code> | Disconnect a Client from MQTT Server. |
| <code>bc66_cmd_list_QMTSUB</code> | Subscribe to Topics. |
| <code>bc66_cmd_list_QMTUNS</code> | Unsubscribe from Topics. |
| <code>bc66_cmd_list_QMTPUB</code> | Publish Messages. |
| <code>bc66_cmd_list_size</code> | Is not a command. Only to know commands quantity. |

4.2.2.2 `bc66_cmd_type_t` enum `bc66_cmd_type_t`

AT command possibility. Erch command can test and/or read and/or write and/or execute. Use with `bc66_send_at_command(...)` function.

Enumerator

| | |
|-----------------------------|------------------------|
| <code>BC66_CMD_TEST</code> | Send AT TEST command. |
| <code>BC66_CMD_READ</code> | Send AT READ command. |
| <code>BC66_CMD_WRITE</code> | Send AT WRITE command. |
| <code>BC66_CMD_EXE</code> | Send AT TEST command. |

4.2.2.3 `bc66_ret_t` enum `bc66_ret_t`

bc66 library api return

Enumerator

| | |
|--|---|
| <code>bc66_ret_success</code> | Modem data process successful. |
| <code>bc66_ret_timeout</code> | Response timeout. |
| <code>bc66_ret_error</code> | Modem response with error message. |
| <code>bc66_ret_out_of_range</code> | At least some argument is out of range. |
| <code>bc66_ret_not_init</code> | |
| <code>bc66_ret_no_cmd_implemented</code> | <code>RSP_NO_CMD_IMPEMENTED</code> . |

4.2.2.4 `pdp_type_t` enum `pdp_type_t`

Enumeration to specify the type of packet data protocol.

Enumerator

| | |
|------------------------------|--|
| <code>pdp_type_ip</code> | Internet Protocol (IETF STD 5). |
| <code>pdp_type_ipv6</code> | Internet Protocol version 6 (IETF RFC 2460). |
| <code>pdp_type_ipv4v6</code> | Dual IP stack (see 3GPP TS 24.301). |
| <code>pdp_type_non_ip</code> | Transfer of Non-IP data to external packet network (see 3GPP TS 24.301). |

4.2.3 Function Documentation

4.2.3.1 `bc66_connect_mqtt_client()` `bc66_ret_t` `bc66_connect_mqtt_client` (``` const char * client_id, const char * user, const char * pass) ```

Connect a Client to MQTT Server.

Parameters

| | |
|-------------------------------|---|
| <code><i>client_id</i></code> | : The client identifier. The max length is 128 bytes. |
|-------------------------------|---|

Parameters

| | |
|-------------|--|
| <i>user</i> | : User name of the client. It can be used for authentication. The max length is 256 bytes. |
| <i>pass</i> | : Password corresponding to the user name of the client. It can be used for authentication. The max length is 256 bytes. |

Returns

See `bc66_ret_t` return codes.

4.2.3.2 bc66_disconn_mqtt_client() `bc66_ret_t bc66_disconn_mqtt_client (`
`void)`

Disconnect a Client from MQTT Server.

Used when a client requests a disconnection from MQTT server. A DISCONNECT message is sent from the client to the server to indicate that it is about to close its TCP/IP connection.

Returns

See `bc66_ret_t` return codes.

4.2.3.3 bc66_get_at_response() `char* bc66_get_at_response (`
`char * rsp)`

Function to get any response stored in the RX buffer.

Parameters

| | |
|------------|-------------------|
| <i>rsp</i> | : response to get |
|------------|-------------------|

Returns

Response if found, NULL otherwise

4.2.3.4 bc66_get_last_response() `char* bc66_get_last_response (`
`void)`

Function to get last modem response.

If send a new AT command, the buffer which contain the last response will be erased.

Returns

Pointer to RX buffer with last response.

4.2.3.5 bc66_init() `void bc66_init (`
`bc66_obj_t * bc66_obj)`

Function to initialize bc66 object.

Parameters

| | |
|-----------------|--|
| <i>bc66_obj</i> | |
|-----------------|--|

4.2.3.6 bc66_open_net_mqtt_client() `bc66_ret_t bc66_open_net_mqtt_client (`
 `const char * server_ip,`
 `uint16_t server_port)`

Open a Network for MQTT Client.

Parameters

| | |
|--------------------|----------------------------|
| <i>server_ip</i> | : server ip (string) |
| <i>server_port</i> | : server port (0 to 65535) |

Returns

See `bc66_ret_t` return codes.

4.2.3.7 bc66_power_off() `void bc66_power_off ()`

Pull up PWRKEY to turn off the module.

4.2.3.8 bc66_power_on() `void bc66_power_on ()`

Pull down PWRKEY to turn on the module.

4.2.3.9 bc66_publish_msg_mqtt() `bc66_ret_t bc66_publish_msg_mqtt (`
 `const char * topic,`
 `const char * msg,`
 `int qos)`

Publish Messages.

Used to publish messages by a client to a server for distribution to interested subscribers.

Parameters

| | |
|--------------|--|
| <i>topic</i> | : Topic that the client wants to subscribe to or unsubscribe from. The maximum length is 255 bytes. |
| <i>msg</i> | : The message that needs to be published. The maximum length is 700 bytes. If in data mode (after > is responded), the maximum length is 1024 bytes |
| <i>qos</i> | : Integer type. The QoS level at which the client wants to publish the messages. <ul style="list-style-type: none">• 0 At most once• 1 At least once• 2 Exactly once |

Returns

See `bc66_ret_t` return codes.

4.2.3.10 bc66_reset() `void bc66_reset (`
 `void)`

Reset the module when PIN is low.

4.2.3.11 bc66_send_at_command() `bc66_ret_t bc66_send_at_command (`
 `bc66_cmd_type_t cmd_type,`

```

const bc66_cmd_list_t cmd_lst,
const char * exp_rsp,
const char * arg_fmt,
... )

```

Function to send at command sentence to bc66 module through an external function communication.

Parameters

| | |
|-----------------|---|
| <i>cmd_type</i> | : BC66_CMD_TEST, BC66_CMD_READ, BC66_CMD_WRITE or BC66_CMD_EXE type. |
| <i>cmd_lst</i> | : command to send (see command list). |
| <i>rsp</i> | : pointer to expected response text. |
| <i>arg_fmt</i> | : arguments format (like printf function) and must be send all arguments too. |

Returns

See bc66_ret_t return codes.

4.2.3.12 bc66_send_cmd_AT() `bool bc66_send_cmd_AT (`
`void)`

Send AT command to sync baud rate.

Returns

See bc66_ret_t return codes.

4.2.3.13 bc66_set_echo_mode() `bc66_ret_t bc66_set_echo_mode (`
`bool echo)`

Set Command Echo Mode.

This Execution Command determines whether or not the UE echoes characters received from external MCU during command state.

The command takes effect immediately. Remain valid after deep-sleep wakeup. The configuration will be saved to NVRAM (should execute AT&W after this command is issued).

Parameters

| | |
|-------------|--|
| <i>echo</i> | <ul style="list-style-type: none"> • false: Echo mode OFF • true: Echo mode ON |
|-------------|--|

Returns

See bc66_ret_t return codes.

4.2.3.14 bc66_set_mqtt_parameters() `bc66_ret_t bc66_set_mqtt_parameters (`
`uint16_t keepalive,`
`bool dataformat,`
`bool session,`
`bool version)`

Used to configure optional parameters of MQTT.

Parameters

| | |
|-------------------|---|
| <i>keepalive</i> | : Configure the keep-alive time. The range is 0-3600. The default value is 120. Unit: second. It defines the maximum time interval between messages received from a client. If the server does not receive a message from the client within 1.5 times of the keep-alive time period, it disconnects the client as if the client has sent a DISCONNECT message. 0 The client is not disconnected |
| <i>dataformat</i> | : The format of sent and received data. <ul style="list-style-type: none"> • 0 Text format • 1 Hex format |
| <i>session</i> | : The session type. <ul style="list-style-type: none"> • 0 The server must store the subscriptions of the client after it is disconnected. • 1 The server must discard any previously maintained information about the client and treat the connection as "clean". |
| <i>version</i> | : The version of MQTT protocol. <ul style="list-style-type: none"> • 0 MQTT v3.1 • 1 MQTT v3.1.1 |

Returns

See `bc66_ret_t` return codes.

4.2.3.15 `bc66_set_power_saving_mode()` `bc66_ret_t bc66_set_power_saving_mode (`
`int mode)`

Power Saving Mode Setting.

Parameters

| | |
|-------------|--|
| <i>mode</i> | Integer type. Disable or enable the use of PSM in the UE <ul style="list-style-type: none"> • 0 Disable the use of PSM • 1 Enable the use of PSM • 2 Disable the use of PSM and discard all parameters for PSM or, if available, reset to the default values. |
|-------------|--|

Returns

See `bc66_ret_t` return codes.

Power Saving Mode Setting.

Parameters

| | |
|-------------|--|
| <i>mode</i> | Integer type. Disable or enable the use of PSM in the UE <ul style="list-style-type: none"> • 0 Disable the use of PSM • 1 Enable the use of PSM • 2 Disable the use of PSM and discard all parameters for PSM or, if available, reset to the default values. |
|-------------|--|

Returns

See `bc66_ret_t` return codes.

4.2.3.16 `bc66_set_psd_conn()` `bc66_ret_t bc66_set_psd_conn (`
`pdp_type_t pdp_type,`
`const char * apn,`
`const char * user,`
`const char * pass)`

Set Default PSD Connection.

This command sets the PSD connection settings for PDN connection on power-up. When attaching to the NB-IoT network on power-on, a PDN connection setup must be performed. In order to allow this to happen, PDN connection settings must be stored in NVRAM, thus making it to be used by the modem during the attach procedure.

Parameters

| | |
|-----------------|--|
| <i>pdp_type</i> | : Specify the type of packet data protocol. |
| <i>apn</i> | : A logical name that is used to select the GGSN or the external packet data network. The maximum configurable APN length is 99 bytes. |
| <i>user</i> | : The user name for accessing to the IP network. (Optional) |
| <i>pass</i> | : The password for accessing to the IP network. (Optional) |

Returns

See `bc66_ret_t` return codes.

Index

[/Users/jcbecerra/dev/fw/iot/quectel_bc66_driver/src/bc66_drv.c](#), [bc66_drv.h](#), [15](#)
[5](#)
[BC66_CMD_READ](#)
[/Users/jcbecerra/dev/fw/iot/quectel_bc66_driver/src/bc66_drv.h](#), [bc66_drv.h](#), [16](#)
[13](#)
[BC66_CMD_TEST](#)
[bc66_drv.h](#), [16](#)
[bc66_cmd_type_t](#)
[bc66_drv.h](#), [16](#)
[BC66_CMD_WRITE](#)
[bc66_drv.h](#), [16](#)
[bc66_cmds_list](#)
[bc66_drv.c](#), [13](#)
[bc66_connect_mqtt_client](#)
[bc66_drv.c](#), [8](#)
[bc66_drv.h](#), [16](#)
[bc66_deinit](#)
[bc66_drv.c](#), [8](#)
[bc66_disconn_mqtt_client](#)
[bc66_drv.c](#), [9](#)
[bc66_drv.h](#), [17](#)
[bc66_drv.c](#)
[bc66_cmds_list](#), [13](#)
[bc66_connect_mqtt_client](#), [8](#)
[bc66_deinit](#), [8](#)
[bc66_disconn_mqtt_client](#), [9](#)
[bc66_get_at_response](#), [9](#)
[bc66_get_last_response](#), [9](#)
[bc66_init](#), [9](#)
[bc66_open_net_mqtt_client](#), [9](#)
[bc66_power_off](#), [10](#)
[bc66_power_on](#), [10](#)
[bc66_publish_msg_mqtt](#), [10](#)
[bc66_reset](#), [10](#)
[bc66_send_at_command](#), [10](#)
[bc66_send_cmd_AT](#), [11](#)
[bc66_set_echo_mode](#), [11](#)
[bc66_set_mqtt_parameters](#), [11](#)
[bc66_set_power_saving_mode](#), [12](#)
[bc66_set_psd_conn](#), [12](#)
[CMD_END_LINE](#), [7](#)
[cmd_flg_t](#), [8](#)
[EXE](#), [8](#)
[MAX_RSP_SIZE](#), [7](#)
[READ](#), [8](#)
[RSP_END_OF_LINE](#), [7](#)
[RSP_ERROR](#), [8](#)
[RSP_NO_CMD_IMPEMENTED](#), [8](#)
[RSP_OK](#), [8](#)
[RSP_TIMEOUT](#), [8](#)
[TEST](#), [8](#)
[WRITE](#), [8](#)
[bc66_drv.h](#)
[BC66_CMD_EXE](#), [16](#)
[bc66_cmd_list_AT](#), [15](#)
[bc66_cmd_list_ATE](#), [15](#)
[bc66_cmd_list_ATI](#), [15](#)
[bc66_at_cmd_t](#), [2](#)
[cmd](#), [2](#)
[cmd_flags](#), [2](#)
[cmd_rsp](#), [2](#)
[rsp_timeout](#), [3](#)
[BC66_CMD_EXE](#)
[bc66_drv.h](#), [16](#)
[bc66_cmd_list_AT](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_ATE](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_ATI](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_CEREG](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_CESQ](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_CGATT](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_CGPADDR](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_CIMI](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_CPIN](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_CPSMS](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_QCGDEFCONT](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_QMTCFG](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_QMTCLOSE](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_QMTCONN](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_QMTDISC](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_QMTOPEN](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_QMTPUB](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_QMTSUB](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_QMTUNS](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_QNBIOTEVENT](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_size](#)
[bc66_drv.h](#), [15](#)
[bc66_cmd_list_t](#)

- bc66_cmd_list_CEREG, 15
- bc66_cmd_list_CESQ, 15
- bc66_cmd_list_CGATT, 15
- bc66_cmd_list_CGPADDR, 15
- bc66_cmd_list_CIMI, 15
- bc66_cmd_list_CPIN, 15
- bc66_cmd_list_CPSMS, 15
- bc66_cmd_list_QCGDEFCONT, 15
- bc66_cmd_list_QMTCFG, 15
- bc66_cmd_list_QMTCLOSE, 15
- bc66_cmd_list_QMTCONN, 15
- bc66_cmd_list_QMTDISC, 15
- bc66_cmd_list_QMTOPE, 15
- bc66_cmd_list_QMTPUB, 15
- bc66_cmd_list_QMTSUB, 15
- bc66_cmd_list_QMTUNS, 15
- bc66_cmd_list_QNBIOTEVENT, 15
- bc66_cmd_list_size, 15
- bc66_cmd_list_t, 15
- BC66_CMD_READ, 16
- BC66_CMD_TEST, 16
- bc66_cmd_type_t, 16
- BC66_CMD_WRITE, 16
- bc66_connect_mqtt_client, 16
- bc66_disconn_mqtt_client, 17
- bc66_get_at_response, 17
- bc66_get_last_response, 17
- bc66_init, 17
- bc66_open_net_mqtt_client, 17
- bc66_power_off, 18
- bc66_power_on, 18
- bc66_publish_msg_mqtt, 18
- bc66_reset, 18
- bc66_ret_error, 16
- bc66_ret_no_cmd_implemented, 16
- bc66_ret_not_init, 16
- bc66_ret_out_of_range, 16
- bc66_ret_success, 16
- bc66_ret_t, 16
- bc66_ret_timeout, 16
- bc66_send_at_command, 18
- bc66_send_cmd_AT, 19
- bc66_set_echo_mode, 19
- bc66_set_mqtt_parameters, 19
- bc66_set_power_saving_mode, 20
- bc66_set_psd_conn, 21
- pdp_type_ip, 16
- pdp_type_ipv4v6, 16
- pdp_type_ipv6, 16
- pdp_type_non_ip, 16
- pdp_type_t, 16
- bc66_get_at_response
 - bc66_drv.c, 9
 - bc66_drv.h, 17
- bc66_get_last_response
 - bc66_drv.c, 9
 - bc66_drv.h, 17
- bc66_init
 - bc66_drv.c, 9
 - bc66_drv.h, 17
- bc66_obj_t, 3
 - control_lines, 3
 - func_delay, 3
 - func_init_ptr, 4
 - func_r_bytes_ptr, 4
 - func_w_bytes_ptr, 4
 - MDM_PSM_EINT_N, 4
 - MDM_PWRKEY_N, 4
 - MDM_RESET_N, 4
 - MDM_RI, 4
- bc66_open_net_mqtt_client
 - bc66_drv.c, 9
 - bc66_drv.h, 17
- bc66_power_off
 - bc66_drv.c, 10
 - bc66_drv.h, 18
- bc66_power_on
 - bc66_drv.c, 10
 - bc66_drv.h, 18
- bc66_publish_msg_mqtt
 - bc66_drv.c, 10
 - bc66_drv.h, 18
- bc66_reset
 - bc66_drv.c, 10
 - bc66_drv.h, 18
- bc66_ret_error
 - bc66_drv.h, 16
- bc66_ret_no_cmd_implemented
 - bc66_drv.h, 16
- bc66_ret_not_init
 - bc66_drv.h, 16
- bc66_ret_out_of_range
 - bc66_drv.h, 16
- bc66_ret_success
 - bc66_drv.h, 16
- bc66_ret_t
 - bc66_drv.h, 16
- bc66_ret_timeout
 - bc66_drv.h, 16
- bc66_send_at_command
 - bc66_drv.c, 10
 - bc66_drv.h, 18
- bc66_send_cmd_AT
 - bc66_drv.c, 11
 - bc66_drv.h, 19
- bc66_set_echo_mode
 - bc66_drv.c, 11
 - bc66_drv.h, 19
- bc66_set_mqtt_parameters
 - bc66_drv.c, 11
 - bc66_drv.h, 19
- bc66_set_power_saving_mode
 - bc66_drv.c, 12
 - bc66_drv.h, 20
- bc66_set_psd_conn
 - bc66_drv.c, 12

- bc66_drv.h, [21](#)
- cmd
 - bc66_at_cmd_t, [2](#)
- CMD_END_LINE
 - bc66_drv.c, [7](#)
- cmd_flags
 - bc66_at_cmd_t, [2](#)
- cmd_flg_t
 - bc66_drv.c, [8](#)
- cmd_rsp
 - bc66_at_cmd_t, [2](#)
- control_lines
 - bc66_obj_t, [3](#)
- EXE
 - bc66_drv.c, [8](#)
- func_delay
 - bc66_obj_t, [3](#)
- func_init_ptr
 - bc66_obj_t, [4](#)
- func_r_bytes_ptr
 - bc66_obj_t, [4](#)
- func_w_bytes_ptr
 - bc66_obj_t, [4](#)
- MAX_RSP_SIZE
 - bc66_drv.c, [7](#)
- MDM_PSM_EINT_N
 - bc66_obj_t, [4](#)
- MDM_PWRKEY_N
 - bc66_obj_t, [4](#)
- MDM_RESET_N
 - bc66_obj_t, [4](#)
- MDM_RI
 - bc66_obj_t, [4](#)
- pdp_type_ip
 - bc66_drv.h, [16](#)
- pdp_type_ipv4v6
 - bc66_drv.h, [16](#)
- pdp_type_ipv6
 - bc66_drv.h, [16](#)
- pdp_type_non_ip
 - bc66_drv.h, [16](#)
- pdp_type_t
 - bc66_drv.h, [16](#)
- READ
 - bc66_drv.c, [8](#)
- RSP_END_OF_LINE
 - bc66_drv.c, [7](#)
- RSP_ERROR
 - bc66_drv.c, [8](#)
- RSP_NO_CMD_IMPEMENTED
 - bc66_drv.c, [8](#)
- RSP_OK
 - bc66_drv.c, [8](#)
- RSP_TIMEOUT
 - bc66_drv.c, [8](#)
- rsp_timeout
 - bc66_at_cmd_t, [3](#)
- TEST
 - bc66_drv.c, [8](#)
- WRITE
 - bc66_drv.c, [8](#)