nrf_dongle UART protocol

This section documents commands available by UART.

Firmware uses 115200 baud rate.

All comands need to be terminated by `\r` or `\n` or both.

Version information

Branch: development

Version: 0.2.1

Commit hash: 993e3bd0d86e4e4ce4e5d7bb6856df8720df185f

General purpose commands

Info

Getting basic information about connected device.

Command

```
info
```

Response

```
Device information
firmware: nrf_dongle
firmware_version: 0.2.1-993e3bd
firmware_build: 20171027-142541
device_name: nrf_dongle
serial_number: bdb748c963b0d511d85058b9d8ef
mac_address: efd8b95850d8
device_state: 1
adv_state: 0
scan_state: 0
END
```

Reset

Reset device.

Command

```
reset
```

Response

No response, device resets.

Start DFU

Start DFU firmware update.

Command

```
dfu_mode_start
```

Response

No response, device resets into DFU bootloader and waits for further data. Use pc-nrfutil and existing scripts to perform DFU.

Restore defaults

Restore all memory to default state for given firmware.

Command

```
restore_defaults
```

Response

No response. Need to reset the device after applying.

Parameters

None

Device name

Set/get device name.

Command

```
device_name
```

Set

```
device_name=$name
```

Get

```
device_name
```

Response

Example response `device_name=6c6f72656d697073756d`

Parameters

- `$name` - Device name, should be passed as a hex string.

Real Time Clock

Set/get timestamp value. Start/stop RTC.

Command

```
timestamp
```

Set

```
timestamp=$clock_state $timestamp_value
```

Get

```
timestamp
```

Response

Example response `timestamp=01 597f24ce`

Parameters

- `$clock_state` - Real Time Clock state. Set `01` for switch on, `00` for switch off.
- `$timestamp_value` - Current timestamp value, should be passed as a hex value.

Scanning commands

Group of commands related to BLE scanning.

Scan start/stop

Command (start)

```
scan=01
```

Command (stop)

```
scan=00
```

Response

No response though after starting scan reports will start arriving in the following format:

```
@scan:de1b9ba05f2c,46489c12885b,-48,0,11,02010607ff4c0010020b00
@scan:de1b9ba05f2c,46489c12885b,-48,0,0,
@scan:de1b9ba05f2c,20fabb017779,-90,2,30,0201061aff4c000215c23104adb71cdd5910fcb27
7d51250190212b359c8
@scan:de1b9ba05f2c,c6ba66b8eb5b,-62,0,19,031900000201060b096e72665f646f6e676c65
@scan:de1b9ba05f2c,c6ba66b8eb5b,-61,0,19,031900000201060b096e72665f646f6e676c65
```

Advertising commands

Group of commands allowing to control BLE advertising data.

nrf_dongle has 8 banks for storing unique payloads and advertising settings. Each one is treated separately and can be enabled and disabled on demand. At least one advertising bank has to be enabled for advertising to be started.

Advertising data

Set/get advertising payload.

Command

```
adv_payload
```

Write

```
adv_payload:$bank_id=$data
```

Read

```
adv_payload:$bank_id
```

Parameters

- `$bank_id` - <00x00,0x07> range, specifies which advertising payload is being set. There are 8 advertising data payloads.
- `$data` should be passed as a hex string, so "123" will be "303132"

Advertising settings

Control other parameters of advertising.

Command

```
adv_settings
```

Write

Note: All parameters need to be specified as hex values. Each parameter is space separated

```
adv_settings:$bank_id=$enabled $connectable $adv_interval $tx_power $duration
```

Response

Example response

```
adv_settings:00=01 01 014d 07 00000141
```

Read

```
adv_settings:%bank_id
```

Response

Same as for write

Parameters

- `$bank_id` - <0x00,0x07> range, specifies which advertising payload is being set. There are 8 advertising data payloads.
- `$enabled` - `00` for disabled, `01` for enabled.
- `$connectable` - `00` for non connectable, `01` for connectable.
- `$adv_interval` - Two byte hex string with avertising interval. `<0064, 2710>` range (100-10000ms)
- `$tx_power` - Power of transmission. `<00, 07>` range.
- `$duration` - Duration of advertising before switching to next available bank. `<00000064, ffffffff>` range (100-∞ [ms]), set `FFFFFFFF` for disable switching.

Advertising enable/disable

Enable or disable advertising.

Command (enable)

```
adv=01
```

Command (disable)

```
adv=00
```

Response

Example response

```
adv:01
```

BLE commands

UART - BLE send

Command

```
ble_uart_tx:$connection_id=$payload
```

Short form is also available (debug/testing only)

```
#$connection_id$payload_length$payload
```

Example

Send "1234" string (31323334 as hex) to device with `$connection_id` =2:

```
ble_uart_tx:2=31323334
```

In short form (debug/testing only)

```
#020431323334
```

Response

If connection was successful then the following response will be returned by UART

```
OK
```

If an error ocurred:

```
Error: $err_code
```

Possible errors are:

- 8 - Device `$connection_id` is connected but notifications are not enabled
- 16 - Device `$connection_id` is not connected

Parameters

- `$connection_id` - BLE connection to use to send data.
- `$payload_length` - Length of payload to send (which will be defined as hex string). This parameter specifies length of the payload and not the hex string
- `$payload` - Hex string with data to send

BLE connections list

Print list of all aviable BLE connection slots.

Command

```
ble_connections
```

Response

```
BLE Connections
$connection_slot_num:$connection_id:$isConnected:$nus_notif_enabled:$mac_addr
END
```

Example response

```
BLE Connections
0:0:1:1:1cf69b668b64
1:0:0:0:000000000000
2:0:0:0:000000000000
3:0:0:0:000000000000
4:0:0:0:000000000000
5:0:0:0:000000000000
6:0:0:0:000000000000
7:0:0:0:000000000000
END
```

BLE disconenct

Disconnect single BLE connection using connection id.

Command

```
ble_disconnect:$connection_id
```

Response

```
@disconnected:$connection_id:$mac_addr
```

Example response

```
@disconnected:0:4a7e5a42755d
```

UART events

This section documents all possible UART events.

ub_uart_event_ready

Occurs when ub_dongle is turned on and ready to work.

Response

```
@ready
```

ub_uart_event_ble_scan_report

Occurs when BLE scan report is received and ready to display.

Response

```
@scan:$deviceMacHex,$peerMacHex,$rssi,$advType,$adv_data_size,$advData
```

Where: `$deviceMacHex` - Hexadecimal value of mac address of ub_dongle `$peerMacHex` - Hexadecimal value of mac address of scanned device `$rssi` - Received signal strength indicator `$advType` - Type of advertising `$adv_data_size` - Size of recived advertising data `$advData` - Recived advertising data

Example response

```
@scan:c6ba66b8eb5b,20fabb0173c6,-84,3,30,0201041aff4c000215c23104adb71cdd5910fcb27
7d512501900d2395cc8
```

ub_uart_event_ble_connected

Occurs when BLE connection is established.

Response

```
@connected:$connection_id:$peerMacHex
```

Where: `$connection_id` - Identification number of established connection `$peerMacHex` - Hexadecimal value of mac address of connected device

Example response

```
@connected:0:26f86f497e5f
```

ub_uart_event_ble_disconnected

Occurs when BLE connection is terminated.

Response

```
@disconnected:$connection_id:$peerMacHex
```

Where: `$connection_id` - Identification number of terminated connection `$peerMacHex` - Hexadecimal value of mac address of disconnected device

Example response

```
@disconnected:0:26f86f497e5f
```

ub_uart_event_ble_nus_receive

Occurs when BLE data is recived through NUS.

Response

```
@ble_uart_rx:$connection_id:$data_hex
```

Where: `$connection_id` - Identification number of device that is sending data throgh NUS `$data_hex` - Hexadecimal value of data recived through NUS

Example response

```
@ble_uart_rx:0:1234
```

ub_uart_event_ble_nus_notifications

Occurs when BLE NUS notifications are enabled or disabled.

Response

```
@ble_nus_notifications:$connection_id:$nus_notification_enabled:$peerMacHex
```

Where: `$connection_id` - Identification number of device that is sending data throgh NUS `$nus_notification_enabled` - `0` if notifications are disabled and `1` if notifications are enabled `$peerMacHex` - Hexadecimal value of mac address of connected device

Example response

```
@ble_nus_notifications:0:1:26f86f497e5f
```