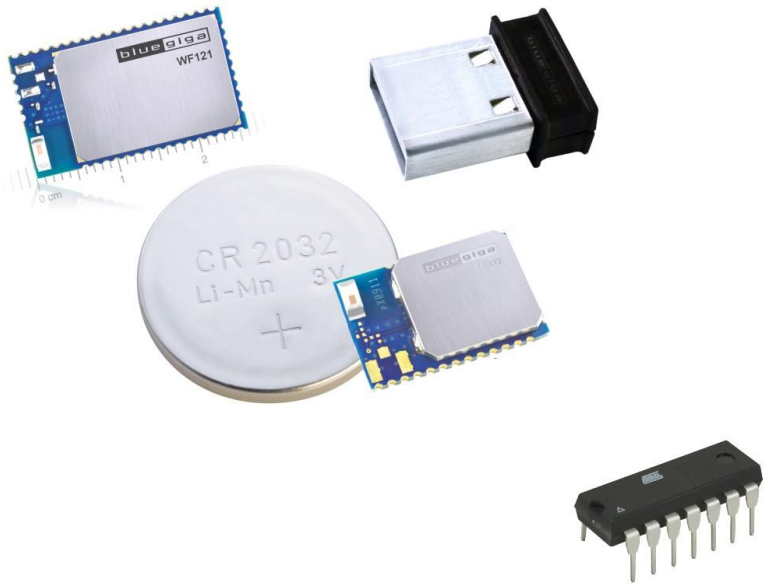
BGAPI and BGLib Overview

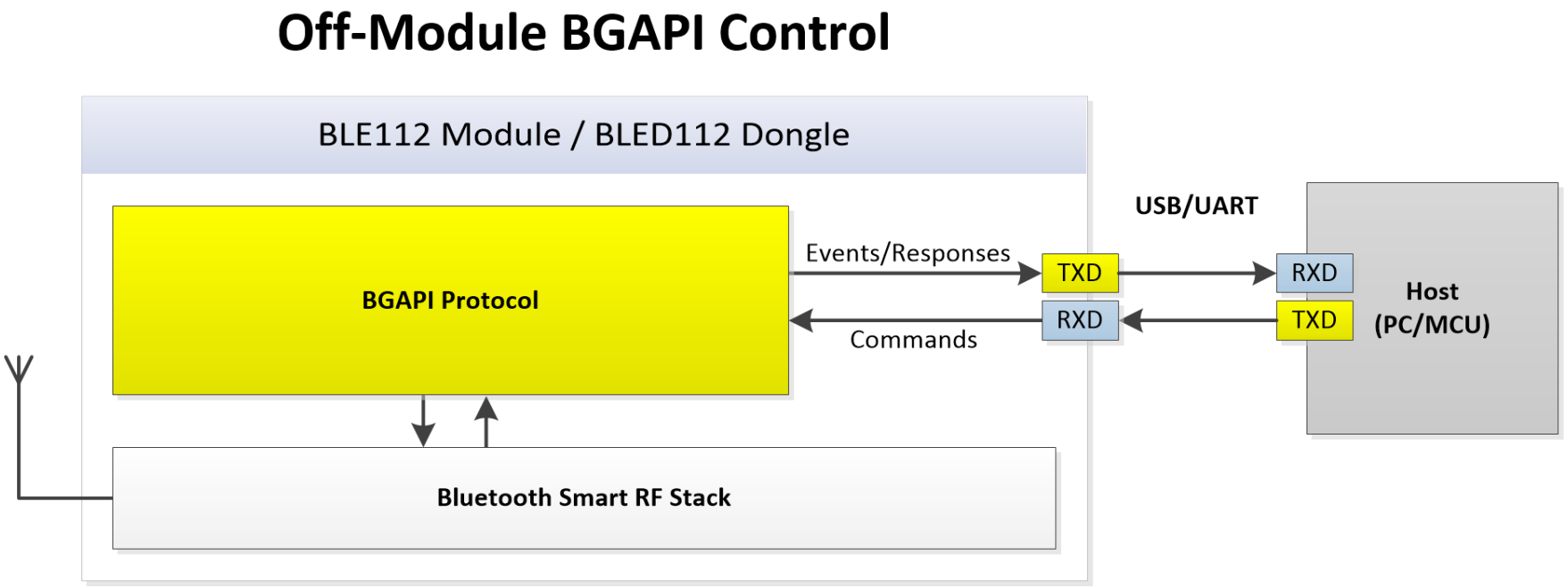
Table of Contents



What is BGAPI?

- Binary protocol used to communicate with BLE(D)11x and WF121
- A set of packets built with a standard, documented structure
- Underlying functionality powering interpreted BGScript language
- Advantages:
  - Increased speed compared to BGScript
  - No interpreter overhead
  - More flexibility (functions, compiler directives, more RAM)
- Disadvantages:
  - Increased design complexity (BGAPI control from host)
  - Additional external components (MCU)
  - Additional consideration for wake-up pin accommodating optional sleep modes

BGAPI Stack Structure



BGAPI Packet Structure

- Every packet has a 4-byte header
- Packets *may* have between 0-60 payload data bytes after header

Octet	Octet bits	Length	Description	Notes
Octet 0	7	1 bit	**Message Type (MT)**	0: Command/Response 1: Event
...	6:3	4 bits	**Technology Type (TT)**	0000: Bluetooth 4.0 single mode 0001: Wi-Fi
...	2:0	3 bits	**Length High (LH)**	Payload length (high bits)
Octet 1	7:0	8 bits	**Length Low (LL)**	Payload length (low bits)
Octet 2	7:0	8 bits	**Class ID (CID)**	Command class ID
Octet 3	7:0	8 bits	**Command ID (CMD)**	Command ID
Octet 4-n	-	0 - 2048 Bytes*	**Payload (PL)**	Up to 2048 bytes of payload

**Maximum actual payload length in protocol usage is 60 bytes, due to memory limitations on internal module CPUs. The 11-bit <length> field theoretical maximum is 2048, but full packet length limit is 64 bytes, leaving 4 for header and 60 for payload data.*

BGAPI Packet Sample

- The **system_boot** event is sent when the BLE device is powered or reset
- BGAPI **system_boot** event packet in hexadecimal notation:

80 0C 00 00 01 00 01 00 01 00 64 00 02 00 01 01

Octet	Octet bits	Length	Description	Packet Data	Meaning
Octet 0	7	1 bit	**Message Type (MT)**	0x80 bit 7 = **1**	Event
...	6:3	4 bits	**Technology Type (TT)**	0xx80 bits 6:3 = **0000**	Bluetooth 4.0 single mode
...	2:0	3 bits	**Length High (LH)**	0x80 bits 2:0 = 000 (LH) 0x0C = 12 (LL) → **0 + 12 = 12 (length)**	Payload length = 12 bytes
Octet 1	7:0	8 bits	**Length Low (LL)**		
Octet 2	7:0	8 bits	**Class ID (CID)**	**00**	Command class ID is **system**
Octet 3	7:0	8 bits	**Command ID (CMD)**	**00**	Command ID is **boot**
Octet 4-15	All	12 bytes	**Payload (PL)**	**01 00 01 00 01 00** **3F 00 02 00 01 01**	Data payload of **system_boot** event *(see API Reference Manual)*

BGAPI Packet Types

- Three types of packets in BGAPI:
  - **Command** (MT bit = 0, flow is host → module)
  - **Response** (MT bit = 0, flow is module → host)
  - **Event** (MT bit = 1, flow is module → host)

- **Command** packets control the module, sent from connected host

- **Response** packets are immediate replies after sending commands
  - *Every* command generates a response, except **system_reset**

- **Event** packets come when events occur on the module, e.g:
  - New remote device connected
  - Attribute value written
  - Software timer tick

BGAPI Wake-up Pin and Sleep Mode for BLE11x Modules

- BLE11x optionally uses sleep modes to minimize power consumption
- If allowed, the module will NOT receive UART data from an external microcontroller or other UART-connected host while asleep
- Use the <*wakeup_pin*> tag in hardware.xml to configure a pin which you can assert to wake up the module while sending data
- Events and responses coming *from* the module will automatically wake up from sleep if necessary

BGAPI Additional Notes

- All multi-byte integer-based data payload fields (int16, uint16, int32, uint32, bd_addr) use **little-endian byte order**. Examples:
  - UUID: 0x2902 = **`02 29`**
  - MAC: 0x00078045F38A = **`8A F3 45 80 07 00`**

- The **uint8array** data type is made up of a **length** byte followed by that number of data bytes. Examples:
  - "ABCD" = **`04 41 42 43 44`**
  - "Hello world" = **`0B 48 65 6C 6C 6F 20 77 6F 72 6C 64`**

- BGAPI requires that a complete, valid packet be completed within one second from the start of the packet. If the packet is invalid or is incomplete after one second, a BGAPI protocol error event will occur and the parser will return to an "idle" state, waiting for a new packet

BGAPI Additional Notes

- BGAPI is a synchronous, full-duplex protocol, and the parser will accept commands sent in a group, e.g. the following “system_hello” and “system_info” commands sent as a single transmission:

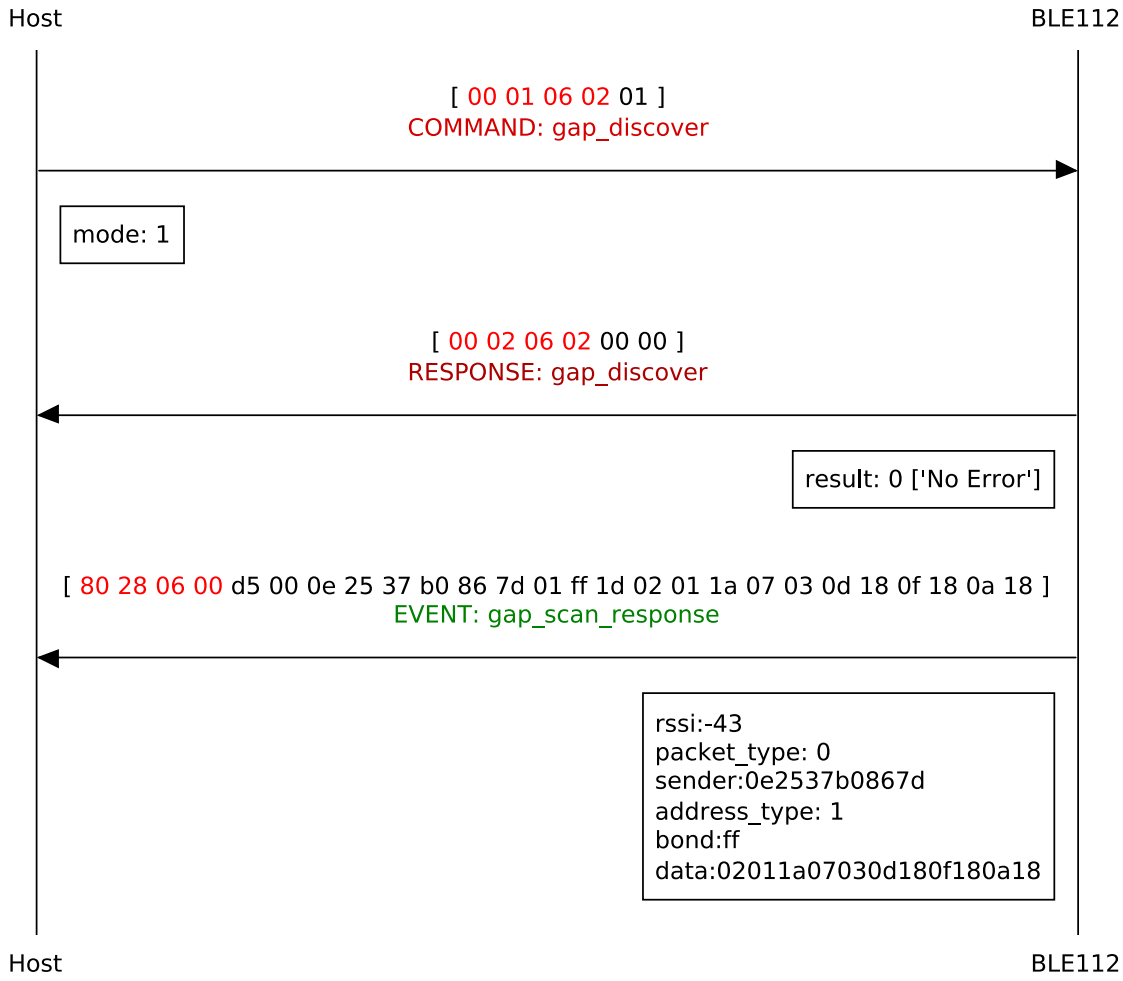
  `00 00 00 01 00 00 00 08`

- Responses will be generated in order and sent back for each command

- Event packets may be generate at any time, including in between command and response packets under certain circumstances; the BGAPI parser on your host device should be able to process packets accordingly

- A single BGAPI packet will *always* be sent in its entirety, and never split in the middle by another packet

BGAPI Packet Flow



The host sends a "gap_discover" command to the module with the 5-byte command packet shown here. The single payload data byte is the "mode" parameter used by gap_discover.

The BLE112 will immediately come back with a response packet. The response should have matching CID and CMD bytes (0x06 and 0x02 in this case), and usually also includes a 16-bit "result" code. A result of 0 always indicates success.

The "gap_scan_response" event comes whenever the BLE112 receives an advertisement packet from a remote device. Once you start scanning, these events may arrive at any time until you stop scanning or connect to a device.

BGAPI “Packet Mode” for BLE11x + UART

- If flow control is not used, some data may be lost when sending to the BLE module over UART, due to 1-byte internal hardware buffer
- Setting “mode” to “packet” on UART configuration enables DMA buffering to prevent overwriting data without flow control, e.g.:

```
<usart channel="1" alternate="1" baud="115200" endpoint="api"
flow="false" mode="packet" />
```

- Requires a preceding “length” byte before any command packet, equal to the number of bytes in the packet not including length byte

  *(this byte is **not** present on responses/events coming from the module)*

  - 4-byte “system_hello” command in default mode:
    `00 00 00 01`
  - 4-byte “system_hello” command in packet mode with “length” byte:
    `04 00 00 00 01`

BGAPI Module Connectivity Comparison Chart

	BLE112	BLE113	BLED112	WF121
				
Interface	*BGAPI supported?*	*BGAPI supported?*	*BGAPI supported?*	*BGAPI supported?*
UART	Yes	Yes	*n/a*	Yes
USB	Yes	*n/a*	Yes	Yes
SPI	No	No	*n/a*	Yes
I^2C	No	No	*n/a*	No

- BGAPI host control is possible over different interfaces
- BGAPI communications should be limited to **one** interface for a given project to ensure predictable behavior

What is BGLib?

What is BGLib?

- Code library which implements BGAPI protocol in ANSI C
- Generated from XML during internal SDK build process
- Functions have names and parameters which follow BGAPI naming convention, for example:
  - ble_cmd_system_hello
  - ble_rsp_system_hello
  - ble_evt_system_boot
- Known ports of BGLib:
  - ANSI C (Bluegiga original)
  - Arduino C++
  - AVR C
  - C# .NET
  - Java

BGLib Project Deployment Concepts

- All BGLib projects require two separate parts:
  - Module firmware project
  - Host application project

- BGLib project deployment process
  1. Create and flash module firmware project
  2. Obtain or write BGLib code for desired target platform
  3. Implement "**send_api_packet**" and "**read_api_packet**" (or similar) data I/O functions as required
  4. Copy and implement event and response functions from stub file*
  5. Comment out implemented stubs in stub source file*

  **Some BGLib libraries on some platforms use functions pointers or events instead of stub functions*

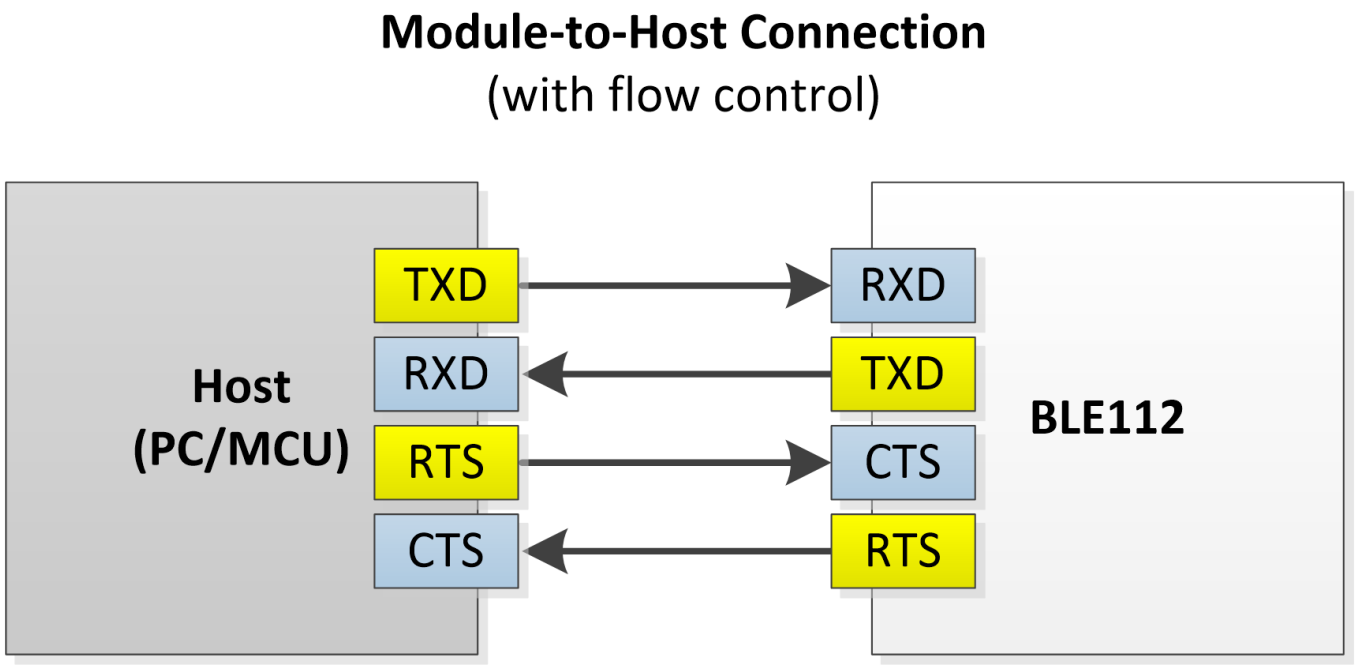
BLE(D)11x setup for BGAPI/BGLib

BLE(D)11x setup for BGAPI/BGLib

- Hardware connections
  - UART control (usually BLE11x + MCU), use flow control if possible and packet mode if needed
  - USB control (usually BLED112 + PC or embedded Linux)

- Project configuration
  - project.xml
  - hardware.xml
    - Connect USB or UART to “api” endpoint
  - config.xml
  - gatt.xml
    - GATT database setup done here since it cannot be changed by BGAPI
  - cdc.xml
    - Required for BLED112, not used for BLE112 unless using USB

BLE11x UART hardware connections for BGAPI/BGLib



- If a wake-up pin is configured, this should be connected as well. It is an input for the BLE11x, driven by the host.

BLE11x Wake-up Pin Operation

- The correct procedure for using the wake-up pin to send BGAPI packets over UART is as follows:

  1. Assert the wake-up pin from an external host
  2. Process the "**hardware_io_port_status**" BGAPI event packet which is generated and sent out the module's TX pin
  3. Send the desired BGAPI command packet to the module
  4. Wait until you receive **at least the first byte of the BGAPI response packet** before de-asserting the wake-up pin

- **Important:**

  – Step 2 above is critical because some sent data may be ignored if you do not process the port status event before starting to send data.

  – Step 4 above is critical because if you de-assert the wake-up pin too soon (e.g. immediately after the last byte is placed in the TX buffer of the attached UART host), then the last byte or two may not be properly clocked into the module before it goes to sleep again, resulting in lost or corrupt data.

WF121 setup for BGAPI/BGLib

WF121 setup for BGAPI/BGLib

- Hardware connections
  - UART control (usually WF121 + MCU), use flow control if possible
  - SPI control (usually WF121 + MCU)
  - USB control (usually WF121 + PC or embedded Linux system)

- Project configuration
  - project.xml
    - Set “api” to “true” on USB, UART, or SPI interface
  - cdc.xml
    - Required only if using USB
  - script.bgs
    - WF121 supports BGScript + BGAPI simultaneously, allowing e.g. basic wifi radio control on-module and specific application logic off-module

Thank you!