# RecoilGun BLE Protocol

|  |  |  |  |
| --- | --- | --- | --- |
| ***Version 2.4*** | ***Matteo Scordino*** | ***16 Aug 2017*** | ***Initial Version*** |

Table of Contents

[RecoilGun BLE Protocol 1](#_gjdgxs)

[Introduction 2](#_30j0zll)

[Definitions 2](#_1fob9te)

[Protocol 3](#_3znysh7)

[BLE Advertising 3](#_2et92p0)

[BLE Services 3](#_3dy6vkm)

[Service “Generic Access” 3](#_1t3h5sf)

[Service “Device Information” 3](#_4d34og8)

[Service RecoilGun 4](#_2s8eyo1)

[Characteristic ID 4](#_17dp8vu)

[Characteristic “Telemetry” 5](#_3rdcrjn)

[Characteristic “Control” 7](#_35nkun2)

[Characteristic “Config” 8](#_1ksv4uv)

# Introduction

The recoil gun firmware system is designed to interface with applications using the BLE communication system. The BLE system sends and receives data in the form of “Services” each of which is split into “characteristics”. A characteristic is essentially a 20 byte chunk of data that can be either readable, writable or notified (notified means that data is spontaneously transferred when the value changes). A Service is a collection of characteristics that has been grouped together for encapsulation.

Our system assumes that data and commands are bit packed into these 20 byte chunks which are then transferred by BLE to and from the hardware. This document explains each “characteristic” and how the bits of these 20 byte chunks are used.

### Definitions

The document explains the function of bits of data in the BLE characteristic data chunks. Each table assumes that the chunks are ordered in the same way as the table. E.g. A 2 byte chunk could be broken down as follows:

|  |  |  |
| --- | --- | --- |
| **Name** | **DataType** | **Description** |
| 1st bits of chunk | U4 |  |
| 2nd bits of chunk | U4 |  |
| 3rd bits of chunk | U8 |  |

* U4 = Unsigned 4-bit value; firstly the low nibble then the high nibble of a byte.
* U8 = Unsigned 8-bit byte.
* U16 = Unsigned 16-bit little-endian word.
* S16 = Signed 2s complement 16-bit little-endian word.
* U32 = Unsigned 32-bit little-endian word.
* GUID = 128-bit value that is a globally unique identifier. Shown in hexadecimal.

# Protocol

## BLE Advertising

The RecoilGun uses an advertising period of 187.5ms.

The advertising data contains:

* Device is Connectable
* Service ID 0x9D10
* Device Local Name

The Device Local Name is an ASCII string concatenating the “SRG1\_” prefix with the 16 characters representing the hex UUID of the device (the same UUID is also available in the ID characteristic of the Recoil service). E.g. SRG1\_BF7EB8569758B65F

When running the bootloader, the Device Local Name is an ASCII string in the form “SRBX\_YYYYYY” with:

- X being 1 for Rifles and 2 for Pistols

- YYYYYY being the first 6 characters of the hex UUID of the device (the same UUID is also available in the ID characteristic of the Recoil service). E.g. SRB2\_BF7EB8

## BLE Services

The Gun will act as a BLE peripheral, implementing the services 1800, 1801 and RecoilGun.

## Service “Generic Access”

GUID: { 1800 }

This is the standard GATT service that needs to be implemented.

* Characteristic “Device Name”. GUID: { 2A00 } Attributes: Read-Write

The device name is an ASCII string concatenating the “SRG1\_” prefix with the 16 characters representing the hex UUID of the device (the same UUID is also available in the ID characteristic of the Recoil service). E.g. SRG1\_BF7EB8569758B65F

## Service “Device Information”

GUID: { 180A }

This is a standard GATT service. Apple devices running test programs such as “LightBlue” prefer to have this implemented, and the implementation is supplied with most BLE demos.

* Characteristic “Manufacturer name String”. GUID: { 2A29 } Attributes: Read

## Service RecoilGun

GUID: { E6F59D10-8230-4a5c-B22F-C062B1D329E3 }

This service contains all the content from the gun. The gun needs only implement this service and the standard service. It contains at least 4 Bluetooth low-energy Characteristics.

### Characteristic ID

GUID: { E6F59D11-8230-4a5c-B22F-C062B1D329E3 }. Attributes: Read-Notify

The app may read this characteristic.

This contains information about the gun identity.

|  |  |  |
| --- | --- | --- |
| **Name** | **DataType** | **Description** |
| Version | U16 | Version number of the firmware. This is the SVN revision of the build.  A value of 0 indicates the firmware was built with uncommitted changes (i.e. it cannot be rebuilt from a clean checkout) |
| UUID | U8\*8 | Unique, fixed identifier for this gun. |
| GunModel | U8 | Type of gun the hardware is assembled in.  1 = rifle  2 = pistol |
| Padding | U8\*3 | NA |
| ConfigCRC | U32 | CRC32 of the current configuration. |
| BL Version | U16 | Version number of the bootloader firmware. This is the SVN revision of the build.  A value of 0 indicates the firmware was built with uncommitted changes (i.e. it cannot be rebuilt from a clean checkout) |

Notes:

This characteristic allows the app to detect the type of gun that it has connected. This could be used for example to show an in image of the model of gun that has been connected.

Please notice that UUID is unrelated to the GunID that is used in Telemetry and Command. The UUID is a hardware-generated, globally unique ID of the gun (you can think of it as a network MAC address.) The GunID instead is a logical ID that must be supplied by the app and that is used to identify shots and bind them to the shooter.

### Characteristic “Telemetry”

GUID: { E6F59D12-8230-4a5c-B22F-C062B1D329E3 }. Attributes: Notify

The app is expected to subscribe to this characteristic.

This contains telemetry and controller values from the gun.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Data**  **Type** | **Description** | |
| Pkt Cnt | U4 | The counter of this packet. | |
| Cmd Cnt | U4 | Latest command counter received from the app. | |
| GunID | U8 | The identifier for this gun. 0x01…0x10 is valid, 0 is invalid. | |
| Buttons | U8 | Digital buttons from controller; one bit per button.  0x01 = trigger  0x02 = reload  0x04 = walkie talkie  0x08 = reset  0x10 = power  0x20 = recoil cnt | |
| Pressed | U4\*6 | Number of times that buttons have been pressed (mod 16):  Trigger, reload, walkie talkie, reset, power, recoil cnt | |
| Voltage | S16 | Battery voltage (in mV)  This is formally a signed number, but of course a negative value is not expected. | |
| IrEvents | Struct {}  \*2 | **2 MSBs of each struct (bytes 8,9 and 11,12 in the packet):**  U16 = infrared payload, defined as:  Bit 15-10: shot counter  Bit9-6: Weapon type of   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Bits** | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | **Type** | Shooter ID | | | | | | Weapon ID | | | | Shot counter | | | | | | | **Grenade** | G | G | G | G | G | G | 1 | 1 | J | J | J | J | S | S | S | S | | **Gun** | A | A | A | A | A | A | W | W | W | W | R | R | R | C | C | C |  * A = id of the shooter as set by the application (1..16)  (default grenade == 50) * W = Weapon ID the shot comes from (0-11 for the gun, 12-15 for the grenade) * C = counter of shots fired (counts up from 0 and wraps) * R = rounds in a plasma shot, according to the following formula: rounds = (RRR+1) \* 4.  e.g.: RRR=0 ->rounds=4   RRR=7->rounds=32 * G = grenade id (6 bit hash of 16 bit serial number) * J = grenade random (4 bit random counter) for distinguishing grenade ids. * S = grenade state (mostly acts as a countdown timer)   **1 LSB of each struct (bytes 10 and 13 in the packet):**  U4 = counter of this event  U4 = which sensor this comes from (0=invalid, 1,2,4,8=IR sensor as a bit map) | |
| WeaponAmmo | U8 | Amount of ammo remaining for firing independently. | |
| GunFlags | U8 | Bit map:  0x01 = Reload mode (clip out, can’t fire)  0x02 = Clip-on IR sensor disconnected  Other values are reserved for future use |
| Selected Weapon Type | U8 | Currently selected Weapon Type |
| *Reserved* | U8\*3 | Reserved for future expansion | |

**Notes**:

When the gun firmware system detects that it has been hit by another player's IR beam, that hit can be detected by the app via the “Telemetry” characteristic (IrEvent). Each IrEvent contains the following information:

**Sensor ID**- there are 4 sensors which can receive the hit, this sensor information could potentially be used to identify the direction that the hit came from.

**Shot Counter**- each bullet fired from any player has a shot count associated with it. This shot count is required so that we can be sure that we only take damage from a bullet once. To further explain this, a shot could be detected by more than one of the IR sensors. using the shot id the firmware can detect if 2 sensors pick up the same shot and then condense them into a single hit event. Additionally, on the apps side, users can check the shot ID differs in each hit event so that they can be sure the app is not applying damage more than once for each bullet (this could for example be used to prevent 2 or more players from being hit by the same bullet.)

**Weapon type**- the “weapon type” which is currently set on the firing player's gun firmware

**Gun ID**- the unique id of the shooter, this allows the app to identify who the shooter was.

### Characteristic “Control”

GUID: { E6F59D13-8230-4a5c-B22F-C062B1D329E3 }. Attributes: Read-Write

The app is expected to write to this characteristic on occasion.

This characteristic lets the app control or guide the gun during gameplay.

|  |  |  |
| --- | --- | --- |
| **Name** | **DataType** | **Description** |
| PktCounter | U4 | The counter of this packet. The packet will be ignored if this is not changed since the previous transmission. |
| CmdCounter | U4 | Increments each time we want the following packet to be acted upon. Allows retransmission without duplicate actions. |
| IR\_ack | U8 | Sequence of infrared events that have been received. |
| Action | U16 | Control actions to exert (bit mask)  0x0000 = Do nothing.  0x0001 = Shoot gun, uses LED1,2 or both per selected weapon type  0x0002 = Set Reloading mode: cannot fire, “clip is out”  0x0004 = Unset Reloading mode and set ammo. Has priority over “Set Reloading” action if contemporaneous  0x0008 = Recoil, according to selected weapon type (currently fixed to 500ms, motor behaviour is not defined)  0x0010 = Muzzle flash, selected weapon type  0x0020 = Turn power off in one second  0x0040 = Output stats on UART (debug only)  0x0080 = Sync (forget outstanding actions and IR events)  0x0100 = Reboot to bootloader mode |
| GunID | U8 | Gun ID for shooting |
| WeaponType | U8 | Weapon id to use for firing (0-11 is valid, 12-15 is reserve for the granade) |
| WeaponAmmo | U8 | Amount of ammo remaining for firing independently. Only applied if the Unset Reloading mode action is set |
| *Reserved* | U8\*13 | Reserved for future expansion. Sending these bytes is optional. |

**Notes:**

Since the connection between trigger presses and activation of the IR emitters is hardcoded to the firmware, the base concept of reloading must also be handled by the firmware. To achieve this the firmware has a concept of ammo count. Each time the app wishes to reload the weapon it should set ammo count to the size of the clip. The firmware will then automatically reduce this ammo count each time it activates the IR emitter and will not respond to trigger presses once the value reaches zero.

**Please notice that when shooting via BLE action, the ammo count is not checked (the shoot takes place regardless of ammo count) and it's not decremented. If the app fires the gun via this action, it is also responsible of updating ammo/implementing ammo related logic.**

### Characteristic “Config”

GUID: { E6F59D14-8230-4a5c-B22F-C062B1D329E3 }. Attributes: Read-Write

The app is expected to write to this characteristic whenever the config table CRC does not match the config table CRC the app has.

This characteristic lets the app alter the configuration table (see below) using a TLV (Tag, Length, Value) format.

|  |  |  |
| --- | --- | --- |
| **Name** | **DataType** | **Description** |
| Tag | U16 | The config parameter to write to |
| Length | U8 | The length of the following value, in bytes |
| Value | U8\*Length | Value to be written |

#### Configuration table

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Length**  **(bytes)** | **Description** |
| 0-11 | WD0 to WD11 | 9 | Weapon Definition (ID 0 defines Weapon 0, etc)   |  |  |  | | --- | --- | --- | | **Field** | **Data**  **Type** | **Notes** | | TriggerMode | U8 | Default 0x01  0x00 = plasma mode  0x01-0xFD = burst of up to N shots  0xFE = full auto  0xFF = RESERVED | | RateOfFire | U8 | Rate of fire for burst or plasma mode, in units of 50ms.  In plasma this is how often 4 new rounds get added to the charge | | PowerIR1 | U8 | Power for IR LED1 (long range)  **Note**: internally, this gets scaled down to 0-18 (with 18=100%), rounding down. That means that a value of 25 will actually generate a 5.5% duty cycle (25\*18/255 with truncation) | | PowerIR2 | U8 | Power for IR LED2 (short range)  **Note**: internally, this gets scaled down to 0-18 (with 18=100%), rounding down. That means that a value of 25 will actually generate a 5.5% duty cycle (25\*18/255 with truncation) | | PowerLED1 | U8 | Power for the green “muzzle” LED | | PowerLED2 | U8 | Power for the white “power” LED  **Only used in debug builds, LED2 is meant for Power indication only** | | PowerMotor | U8 | Time the recoil motor will run for, in units of 5ms. Default is 18 (90ms. i.e. one recoil per shot)  This is internally corrected for depending on battery voltage (as the motor is directly connected to unregulated VBAT) | | FlashLED1 | U4 | LED1 Flash mode for gunshot.  0x0 = none,  0x1 = square wave (once per shot)  0x2 = glow (Follows trigger and ammo, then starts a square wave on release)  0x03 = On (Follows trigger) | | FlashLED2 | U4 | LED2 Flash mode for gunshot.  Same values as above. **Only used in debug builds, LED2 is meant for Power indication only** | | FlashParam1 | U4 | for square wave: num of flashes  for glow: number of flashes for the square wave that happens on trigger release  unused otherwise | | FlashParam2 | U4 | for square wave: length of flash in 100ms steps  for glow: length of the first glow period in 500ms steps (gets faster as trigger is kept pressed) and length of final square wave flash in 100ms  unused otherwise | |
| 16 | ShotConfig | 2 | Config of shot behaviour.   |  |  |  | | --- | --- | --- | | **Field** | **Data**  **Type** | **Notes** | | Auto Feedback | U4 | Bitmask, default 0x3  0x1 = auto recoil ON  0x02 = auto flash on shooting ON (which leds will flash depend on weapon)  0x04 = auto flash on being shot ON  (the power led flashes). **Only implemented in debug builds, LED2 is meant for Power indication only)** | | Unused | U4 | **-** | | TriggerMode Override | U8 | Overrides the weapon-defined trigger mode. Default 0xFF  0x00 = plasma mode  0x01-0xFD = burst of up to N shots  0xFE = full auto  0xFF = Use weapon-defined firing mode. | |
| 17 | IRConfig | 3 | Config of IR TX/RX behaviour. These parameters have the potential of breaking the functionality of IR emitters and sensors, so deviations from the default settings should be thoroughly tested.   |  |  |  | | --- | --- | --- | | **Field** | **Data**  **Type** | **Notes** | | TX Repeats | U4 | Number of times each “bullet” packet is repeated over IR. Default is 2.  Be aware that increasing this can disrupt shooting functionality: if the total transmission time for all repeats is longer that the rate of fire period, bullets will overlap and cancel each other out. | | TX Flags | U4 | Bitmask, default 0x1  0x1 = Randomise TX times  Together with TXRepeats>1, this makes it less likely that 2 players shooting contemporaneously at the same target will cancel each other out. | | RX Enable | U8 | Mask to enable or disable each individual IR sensor. Default 0xF  0x01 = enable sensor 0 (Gun)  0x02 = enable sensor 1 (Gun)  0x03 = enable sensor 2 (Gun)  0x04 = enable sensor 3 (Clip-on) | | Clip-On RX Check Interval | U8 | Period between tests of the presence the Clip-on sensor, in 500ms units.  Beware that:   1. Due to behaviour of the IR sensor, absence is only declared after 3 consecutive tests fail. 2. Presence is detected immediately 3. Each test disables the clip-on sensor for a few milliseconds. Setting this too low increases the probability that one shot is not received | |
| 18 | TBD |  |  |
| 19 | TBD |  |  |
| 20 | TBD |  |  |
| 21 | TBD |  |  |

**Notes:**

The Recoil gun system firmware is optimised to reduce lag between trigger presses and the guns output (firing of the IR beam, recoil feedback system and muzzle flash effect). To remove any lag that could be introduced by allowing the app to control the gun's output, we have directly hooked gun output to trigger presses in the firmware. This means that the exact details of how the feedback system and muzzle flash operate are essentially hardcoded into the firmware. To allow the app some control over how these gun outputs function we have included the concept of a “trigger mode”. The system will provide up to 16 configurable “weapon modes” each of which can use a different trigger mode. Trigger modes define a behaviour for how to activate various systems of the gun when the trigger is pressed. For example, in the “plasma gun” trigger mode, pressing the trigger will glow the muzzle leds while the trigger is held and then activate the IR emitters when the trigger is released. In contrast when using the “Burst3” trigger mode, pressing the trigger will flash the muzzle LEDs and then activate the IR emitters 3 times while the trigger is held (releasing the trigger will do nothing). The “Weapon mode” concept allows for a trigger mode to be tuned by augmenting the behaviour of the trigger mode (e.g. increase the brightness of the muzzle LEDS). Up to 16 weapon modes can be created and then activated while in game.

**The Weapon mode also sets the weapon type that is sent in the IR beam when firing.**