# BBC micro:bit Bluetooth Profile

# Supplementary Information

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Changes** |
| 1.0 | 20th May 2015 | Martin Woolley, Bluetooth SIG | Initial version |
| 1.1 | 21st May 2015 | Martin Woolley, Bluetooth SIG | Added issue: review WRITE vs WRITE WITHOUT RESPONSE choices  Added System LED to LED Service  Added section on new Notification Service |
| 1.2 | 22nd May 2015 | Martin Woolley, Bluetooth SIG | Modifications to profile design following review with BBC micro:bit team:   * removed Button n Command characteristics from the Button Service: superceded by new Event Service * Introduced Event Service as generalised container for client and micro:bit originated events * Added Long Press button state to enumeration associated with Button State characteristics * Removed Magnetometer and Accelerometer configuration characteristics   Made further improvements   * Now have two instances of same Button State characteristic instead of 2 distinct Button 1 State and Button 2 State characteristics * Revised use of Write vs Write Without Response * Changed names of Analogue and Digital Port characteristic to not refernece a pin/port number. Intention is that there will be multiple instances of the same characteristic type for multiple pins. Whether we have this service or not is still open to question. |
| 1.3 | 29th May 2015 | Martin Woolley, Bluetooth SIG | Modifications following review with Arm. This version was verbally signed off in the meeting.   * IO Port Service is now called the IO Pin Service since it is largely concerned with individual pins on the board. Detailed requirements established and service design will now consist of:   + 18 characteristics, each representing 1 pin   + A characteristic which allows each pin to be configured for use as either an analogue or digital pin   + A charactertistic representing the collection of all pins to be addressed as a parallel port * Event Service to have a new characteristic called Client Event Requirements. This allows the client device to indicate the types of event it wants to be notified about. * LED Service will allow strings of text to be sent to it by the client for displaying one character at a time on the LED display. Scrolling behaviour can be controlled and configured via suitable characteristics. * Sequence diagrams updated where appropriate. * It is assumed that the device will use the Nordic Semiconductor S110 soft device as opposed to the S130 previously assumed. Note that this is not materially important to the profie design. |
| 1.4 | 20th August 2015 | Martin Woolley  Bluetooth SIG | Event Service characteristic value data types and formats changed to align with micro:bit run time data types and to fix an issue with the Client Requirements and Microbit Requirements characteristics and their ability to map to the larger than expected range of event ID values. |
| 1.5 | 18th September 2015 | Martin Woolley  Bluetooth SIG | **Optimisation During Profile Implementation**  Button State 2 characteristic given new, distinct UUID of E95D**DA91**-251D-470A-A062-FA1922DFA9A8  Removed the System LED State characteristic from the LED Service since it cannot be controlled from the BLE MCU.  Removed the Scrolling State characteristic from the LED Service due to complexity and memory constraints.  Changed LED Matrix State use of “Write Without Response” to “Write” so that no further writes can be made until there’s been an ACK back from the previous one.  Removed Write property from MicroBit Requirements characteristic.  Removed “Write Requests vs Write No Response Commands” section from this document since there are no longer assumptions being made in this area.  Removed PnP ID, IEEE 11073-20601 Regulatory Certification Data List, System ID and Software Revision String characteristics from the Device Information Service (all are optional).  Removed optional characteristics Peripheral Privacy Flag, Reconnection Address and Peripheral Preferred Connection Parameters from Generic Access Service.  Corrected description of the DFU service to reflect the fact that it is a variation of the Nordic DFU service.  Allocated UUIDs to the DFU service and its two characteristics from the standard micro:bit UUID base value.  Specified correct properties (operations) for the DFU service characteristics and added Client Characteristic Configuration Descriptor to DFU Flash Code since it supports notifications.  Removed characteristics representing all pins except for the edge connector pins, pin 0, pin 1 and pin 2. |
| 1.6 | 17th October 2015 | Martin Woolley  Bluetooth SIG | **Optimisation During Profile Implementation**  Removed Battery Service from profile. micro:bit has no way of determining the level of a connected battery.  Added simple Temperature Service after discovery there are temperature sensors in the micro:bit.  Accelerometer and Magnetometer period characteristics now have uint16 fields instead of uint8 which required scaling up by multipling by 10.  Accelerometer Data and Magnetometer Data characteristics now use signed 16 bit integer fields for each of their X, Y and Z parts.  New characteristic Magnetometer Bearing added to the Magnetometer Service. Provides current bearing in degrees.  Replaced individual IO Pin characteristics and the IO Parallel Port characteristic with the Pin Data characteristic and Pin IO Configuration Charactertistic. The characteristic previously named IO Pin Configuration was renamed Pin AD Configuration. This change allows all pins to be addressed individually or as a specified collection of pins, reduces the amount of memory required on the micro:bit and makes efficient use of the Bluetooth Low Energy link.  DFU Service renamed “DFU Control Service” to better distinguish from the Nordic Semiconductor DFU Service which is not part of this profile but which is used for over the air updates of micro:bit firmware.  Added Generic Attribute Service which was always assumed (it’s mandatory) but not explicitly shown due to an issue in the earlier beta version of Bluetooth Developer Studio.  Changed LED Matrix State use of “Write Without Response” to “Write” so that no further writes can be made until there’s been an ACK back from the previous one.  Changed “Micro:Bit” to “micro:bit” throughout.  Added further information on services and characteristics where this is not available in the associated HTML profile reports. |

Table of Contents

[BBC micro:bit Bluetooth Profile 1](#_Toc430695161)

[Supplementary Information 1](#_Toc430695162)

[Revision History 2](#_Toc430695163)

[Introduction 6](#_Toc430695164)

[Profile Design 6](#_Toc430695165)

[Status of the Design 6](#_Toc430695166)

[Assumptions 6](#_Toc430695167)

[micro:bit Hardware Specification 6](#_Toc430695168)

[General Design Assumptions 7](#_Toc430695169)

[GATT Services 8](#_Toc430695170)

[About the Device Information Service 9](#_Toc430695171)

[About the Accelerometer Service 9](#_Toc430695172)

[About the Magnetometer Service 9](#_Toc430695173)

[About the Temperature Service 9](#_Toc430695174)

[About the Button Service 9](#_Toc430695175)

[About the LED Service 9](#_Toc430695176)

[About the IO Pin Service 10](#_Toc430695177)

[About the Event Service 10](#_Toc430695178)

[About the DFU Control Service 11](#_Toc430695179)

[Open Issues 11](#_Toc430695180)

[Appendix A – Example Sequence Diagrams 12](#_Toc430695181)

## Introduction

The BBC micro:bit will ship with a default Bluetooth Low Energy Profile flashed to it. This profile, in the terminology in use by the BBC is to be used with the “tethered device”.

The profile consists of various “services” and “characteristics” designed to give easy access to the micro:bit’s hardware so that initial exploration of the device’s capabilities may take place using a corresponding, standard smart phone application.

Given the nature of micro:bit and the tools which will be available to developers, it will be possible for the profile to be partly or completely changed and replaced with a profile of the developer’s own design. The latter case is out of scope for this document which focuses on the standard, default profilefor the tethered device scenario only.

## Profile Design

The profile was designed using Bluetooth Developer Studio and is presented in the form of two associated PDF documents depicting the profile at different levels of detail, which should be consulted alongside this supplement.

## Status of the Design

This version of the profile design was signed off on 28th May 2015. It was anticipated however that minor changes would be required and that these will be identified and applied iteratively during the implementation of the profile. For example data types may need optimising. Any further changes will mostly require the profile designer, Martin Woolley of the Bluetooth SIG to work with Joe Finney of Lancaster University.

**The design should now be considered to be under change control.**

V1.5 10th September: The profile is now being implemented and some optimisation is taking place largely to reduce the memory footprint.

## Assumptions

This initial profile design is based on discussions with other members of the team and various assumptions. As far as possible, assumptions have been documented here for reference purposes.

### micro:bit Hardware Specification

**Bluetooth Low Energy**

Nordic Semiconductor nRF51822 with S110 soft device capable of both central and peripheral mode

Bluetooth 4.1 compliant

**Sensors**

Accelerometer

Magnetometer

**User Interface**

2 buttons

25 LED matrix (red)

1 x System LED (yellow)

**I/O**

18 pins which may be used as either analogue or digital pins according to explicit configuration. The device firmware will automatically configure the input/output mode of a pin according to I/O operations addressed to it.

**Power**

1 x cell battery

### General Design Assumptions

micro:bit will act as a GAP peripheral and advertise so that GAP central devices such as a smart phone can discover and connect to it.

Standard Bluetooth SIG “adopted” services will be used where appropriate in conjunction with custom services designed specifically for micro:bit. As such the micro:bit will be shipped with a custom Bluetooth profile. At the time of writing, the Generic Access Service and Device Information Service have been identified as useful adopted services and included in the profile.

The micro:bit “tethered” profile is based around the capabilities and features of the micro:bit device itself. It is not tightly coupled to any particular application of the device such as video control or telephony. It is however able to indicate actions it wishes a connected client device to perform or signal events that have occurred and which the client is expected to act upon in some way.

All services are “primary services” and so may be discovered and enumerated by a client wishing to determine the capabilities of the device.

Ease of use has been considered to be more important than having absolute configurability of all aspects of the hardware (e.g. sensors) in the default profile.

## GATT Services

|  |  |  |
| --- | --- | --- |
| **Service** | **Type** | **Description** |
| Generic Access Service | Adopted | Provides generic information about the device. Mandatory in GATT profiles. |
| Generic Attribute Service | Adopted | Can inform clients of changes in the attribute table such as reassigned handle values.  Mandatory in GATT profiles. |
| Device Information Service | Adopted | Provides more comprehensive details about the device and its manufacturer |
| Accelerometer Service | Custom | Provides access to the accelerometer sensor state and configuration of the frequency with which readings are reported. |
| Magnetometer Service | Custom | Provides access to the magnetometer sensor state and configuration of the frequency with which readings are reported. Provides access to a “current bearing” value in degrees. |
| Temperature Service | Custom | Provides access to a simplified, integer temperature measurement in celsius, derived from several sensors in the micro:bit. |
| Button Service | Custom | Allow button state changes to be notified to the client |
| LED Service | Custom | Allows access to both the LED “display” grid and the system status LED |
| IO Pin Service | Custom | Allows access to and configuration of IO pins on the edge connector. |
| Event Service | Custom | Allows the micro:bit to inform the connected client of the types of event it wants to be informed about.  Allows the client to inform the micro:bit of relevant events.  Allows micro:bit to inform the client of events originating on the micro:bit.  Event data includes both a type and a reason or origin. |

The following sections elaborate on the description of a service and/or its characteristics where this seemed worthwhile.

### About the Device Information Service

This is an adopted service which defines 9 characteristics, all of which are optional members of of the service. For micro:bit we chose to include 5 of those characteristics only.

See <https://developer.bluetooth.org/gatt/services/Pages/ServiceViewer.aspx?u=org.bluetooth.service.device_information.xml>

### About the Accelerometer Service

#### Characteristics

**Accelerometer Data**: Contains accelerometer measurements for X, Y and Z axes as 3 unsigned 16 bit values in that order and in little endian format. Data can be read on demand or notified periodically. Values are in the range +/-1000 and in milli-newtons.

**Accelerometer Period:** Determines the frequency with which accelerometer data is reported in milliseconds.

### About the Magnetometer Service

#### Characteristics

**Magnetometer Data**: Contains magnetometer measurements for X, Y and Z axes as 3 unsigned 16 bit values in that order and in little endian format. Data can be read on demand or notified periodically.

**Magnetometer Period**: Determines the frequency with which magnetometer data is reported in milliseconds.

**Magnetometer Bearing**: Compass bearing in degrees from North.

### About the Temperature Service

#### Characteristics

**Temperature**: Signed integer 8 bit value in celsius.

### About the Button Service

The Button Service exposes the two buttons on the micro:bit and allows their state to be read on demand by a connected client or the client to subscribe to notifications of state change. 3 button states are defined and represented by a simple numeric enumeration: 0 = not pressed, 1 = pressed, 2 = long press.

### About the LED Service

The service provides the client with direct control of each individual LED in the display grid. The client may also work at a higher level of abstraction and send strings of text to be displayed one character at a time on the LED display, with configurable scrolling transitions from one character to the next.

A single characteristic containing a 32 bit mask (7 bits are unused) represents all 25 LEDs with a 0 bit indicating LED OFF and a 1 indicating LED ON. The characteristic may be written or read in a single GATT operation allowing efficient manipulation of all LEDs in the grid.

Other characteristics allow a text string to be written to it by the client for display and the scrolling speed may be set.

#### Characteristics

**LED Matrix State**: Allows the state of any|all LEDs in the 5x5 grid to be set to on or off with a single GATT operation. Consists of a 32 bit field with bits 0 - 24 representing the off (0) or on (1) state of the corresponding LED.

**LED Text**: A UTF-8 string to be shown on the LED display.

**Scrolling Speed**: Specifies a millisecond delay to wait for in between showing each character on the display.

### About the IO Pin Service

#### Characteristics

**Pin Data**: Contains data relating to zero or more pins. Structured as a variable length array of up to 19 Pin Number / Value pairs. Pin Number and Value are each uint8 fields. Note however that the micro:bit has a 10 bit ADC and so values are compressed to 8 bits with a loss of resolution.

WRITE: Clients may write values to one or more pins in a single GATT write operation. A pin to which a value is to be written must have been configured for output using the Pin IO Configuration characteristic. Any attempt to write to a pin which is configured for input will be ignored.

NOTIFY: Notifications will deliver Pin Number / Value pairs for those pins defined as input pins by the Pin IO Configuration characteristic and whose value when read differs from the last read of the pin.

READ: A client reading this characteristic will receive Pin Number / Value pairs for **all** those pins defined as input pins by the Pin IO Configuration characteristic.

The associated Pin AD Configuration characteristic allows the client to indicate how each pin is to be used, as either an analogue or a digital pin.

**Pin IO Configuration**: A bit mask which allows each pin to be configured for input or output use. Bit n corresponds to pin n where 0 <= n < 19. A value of 0 means configured for output and 1 means configured for input.

**Pin AD Configuration**: A bit mask which allows each pin to be configured for analogue or digital use. Bit n corresponds to pin n where 0 <= n < 19. A value of 0 means digital and 1 means analogue.

### About the Event Service

The Event Service allows events or commands to be notified to the micro:bit by a connected client and it allows micro:bit to notify the connected client of events or commands originating from with the micro:bit. The micro:bit can inform the client of the types of event it is interested in being informed about (e.g. an incoming call) and the client can inform the micro:bit of types of event it wants to be notified about. The term “event” will be used here for both event and command types of data.

Events may have an associated value.

Note that specific event ID values including any special values such as those which may represent wild cards are not defined here. The micro:bit run time documentation should be consulted for this information.

Multiple events of different types may be notified to the client or micro:bit at the same time.

Event data is encoded as an array of structs each encoding an event of a given type together with an associated value. Event Type and Event Value are both defined as uint16 and therefore the length of this array will always be a multiple of 4.

|  |
| --- |
| struct event {  uint16 event\_type;  uint16 event\_value;  }; |

The Event Service has four characteristics in total:

**micro:bit Requirements** is a variable length list of event data structures which indicates the types of client event, potentially with a specific value which the micro:bit wishes to be informed of when they occur. The client should read this characteristic when it first connects to the micro:bit. It may also subscribe to notifications to that it can be informed if the value of this characteristic is changed by the micro:bit firmware.

**Client Requirements** is a variable length list of event data structures which indicates the types of micro:bit event, potentially with a specific value which the client wishes to be informed of when they occur. The client should write to this characteristic when it first connects to the micro:bit.

**micro:bit Event** contains one or more event structures which should be notified to the client. It supports notifications and as such the client should subscribe to notifications from this characteristic.

**Client Event** is a writable characteristic which the client may write one or more event structures to, to inform the micro:bit of events which have occurred on the client. These should be of types indicated in the micro:bit Requirements characteristic bit mask.

### About the DFU Control Service

Allows clients to initiate the micro:bit pairing and over the air firmware update procedures. Firmware updates are actually handled by the Nordic Semiconductor DFU service which is not part of this profile, after the micro:bit enters an alternate bootloader.

#### Characteristics

**DFU Control**: Writing 0x01 initiates rebooting the micro:bit into the Nordic Semiconductor bootloader if the DFU Flash Code characteristic has been written to with the correct secret key. Writing 0x02 to this charactertistic means "request flash code".

**DFU Flash Code**: Allows a client to indicate that it "knows" the flash code produced when pairing. Notifications are used to deliver the flash code to a client during the initial pairing process.

## Open Issues

In no particular order:

1. Device will require listing on the Bluetooth SIG site. BBC are aware of the need and process.

2. The full list of event IDs needs to be defined and documented elsewhere. These are just tables of numeric identifiers with an associated meaning.

# Appendix A – Example Sequence Diagrams

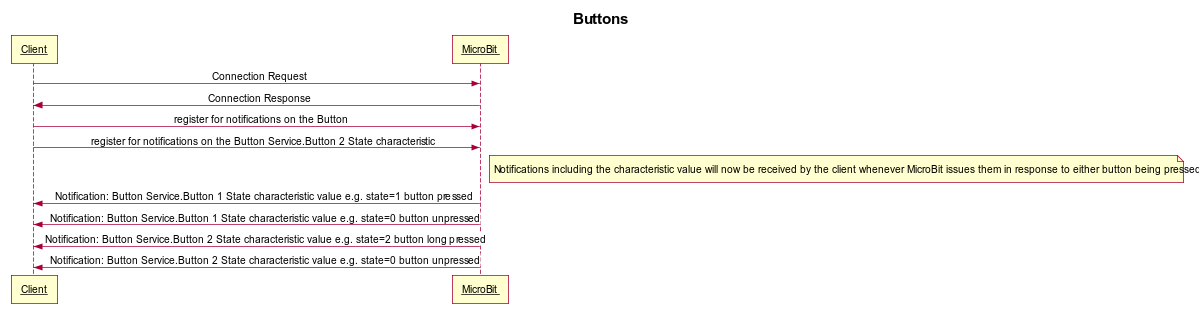


Figure 1 - Button Notifications

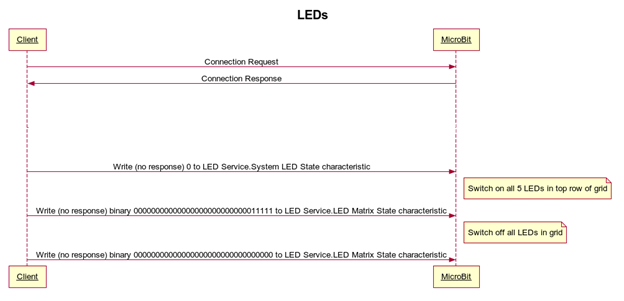


Figure 2 - LED control

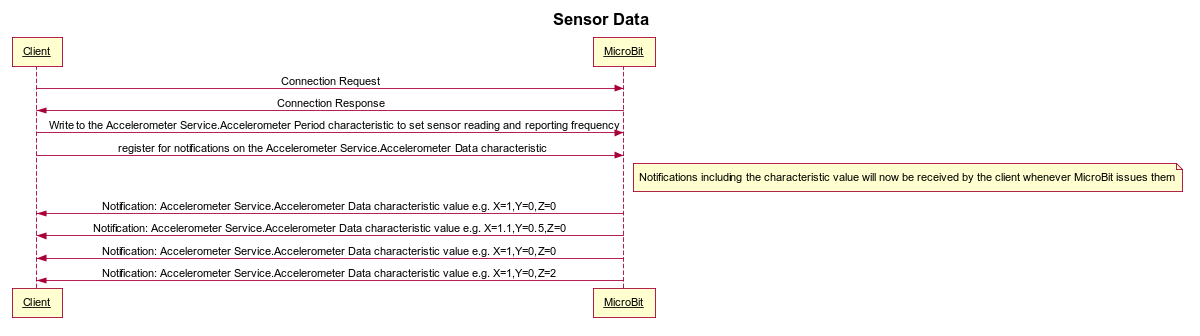


Figure 3 - Accelerometer config and data notifications

NB: Figure 3 shows accelerometer data as an example. The same pattern applies to the Magnetometer Service.

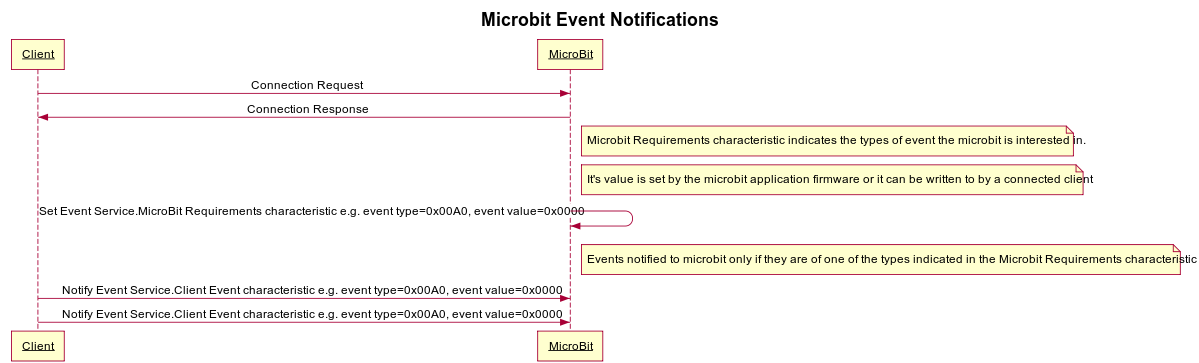


Figure 4 - Client determining micro:bit notification requirements and then sending relevant events as they arise

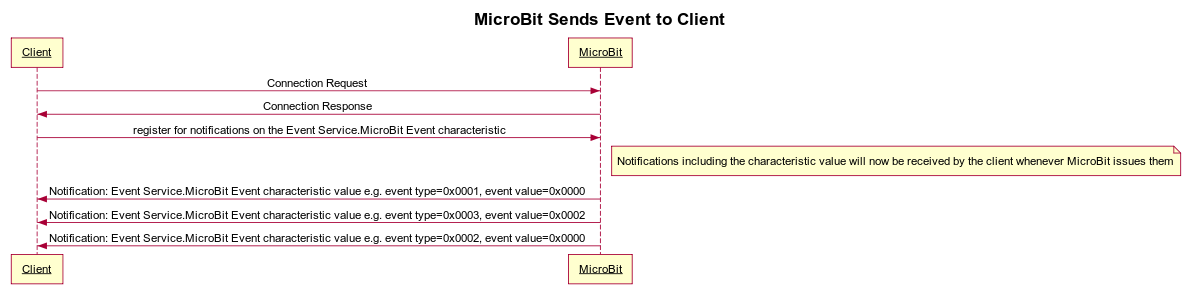


Figure 5 - micro:bit sending events / commands to the client

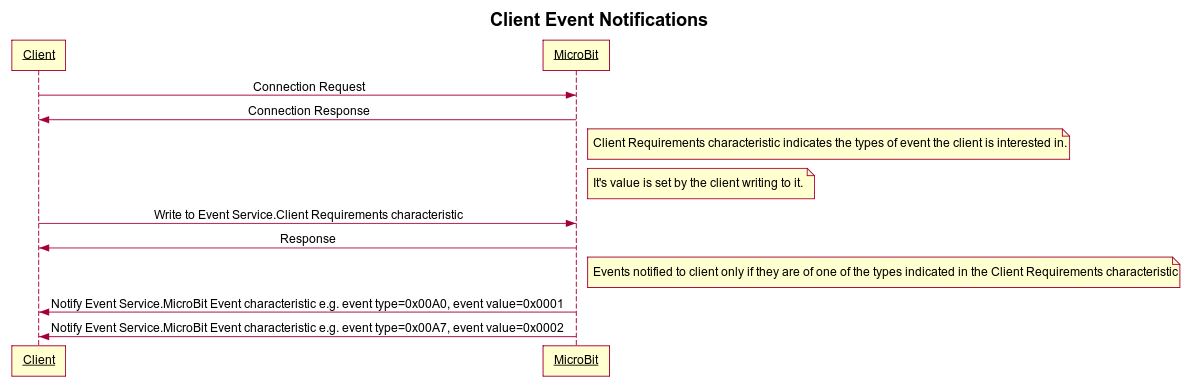


Figure 6 - Client Event Requirements and Notifications