# 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design following review with BBC Micro:Bit team:   * removed Button n Command characteristics from the Button Service: superceded by new Event Service * Introduced Event Service as generalised container for client and Micro:Bit originated events * Added Long Press button state to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 | 10th September 2015 | Martin Woolley  Bluetooth SIG | 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 the it is a variation of the Nordic DFU service designed to accommodate the microbit pairing process. |

Table of Contents

[BBC Micro:Bit Bluetooth Profile 1](#_Toc429653484)

[Supplementary Information 1](#_Toc429653485)

[Revision History 2](#_Toc429653486)

[Introduction 5](#_Toc429653487)

[Profile Design 5](#_Toc429653488)

[Status of the Design 5](#_Toc429653489)

[Assumptions 5](#_Toc429653490)

[Micro:Bit Hardware Specification 5](#_Toc429653491)

[General Design Assumptions 6](#_Toc429653492)

[GATT Services 7](#_Toc429653493)

[About the Button Service 7](#_Toc429653494)

[About the LED Service 7](#_Toc429653495)

[About the IO Pin Service 8](#_Toc429653496)

[About the Event Service 8](#_Toc429653497)

[About the Nordic DFU Service 9](#_Toc429653498)

[Open Issues 10](#_Toc429653499)

[Appendix A – Example Sequence Diagrams 11](#_Toc429653500)

## 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, Battery 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 |
| Device Information Service | Adopted | Provides more comprehensive details about the device and its manufacturer |
| Battery Service | Adopted | Provides battery level |
| 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. |
| 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 ports |
| 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 where this seemed worthwhile.

### 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.

The service has 2 instances of the same Button State characteristic, 1 instance for each of the two physical buttons.

### 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.

Three other characteristics allow a text string to be written to it by the client for display, scrolling speed to be set and scrolling behaviour to be switched on or off.

The board also has a single “system LED” which is yellow. A distinct characteristic representing the state of this LED is included in the LED service.

### About the IO Pin Service

18 of the physical pins on the Micro:Bit are exposed by this service for use by the client. One characteristic per pin exists in the service and these characteristic represent values read from or written to the corresponding pin. A configuration characteristic allows all pins to be configured for either analogue or digital use via a bit mask. A further characteristic allows multiple pins to be collectively addressed in a single operation, thus allowing parallel or parallel-like operations to be supported. The characteristic value consists of a variable length array of structs each of which has an 8 bit pin number and an associated 8 bit value.

### 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. It is assumed that where no value is associated with an event, the value part of the event struct will contain 0x0000. The micro:bit runtime documentation should clarify this. Event data is a fixed length struct so a magic value meaning “no value” must be defined. 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 Service

A service derived from the Nordic DFU service is included in the profile.

## Open Issues

In no particular order:

1. UUIDs need finalising. Apart from SIG adopted items, a temporary UUID has been allocated so far. They may be perfectly acceptable.

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

3. 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

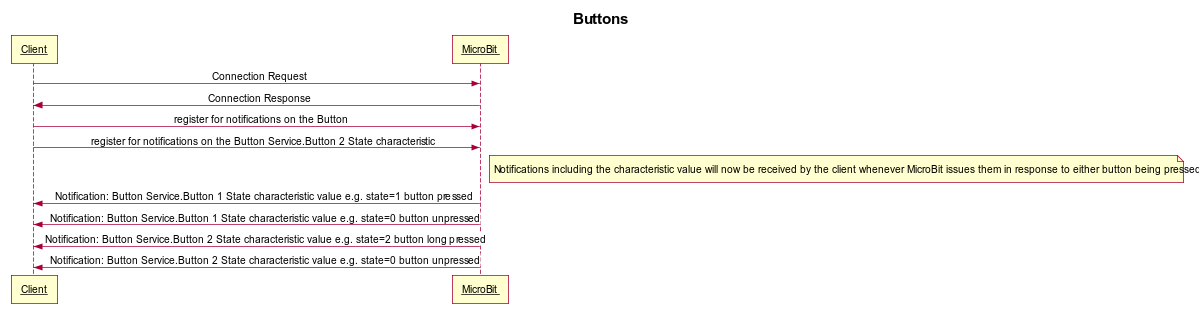


Figure 1 - Button Notifications

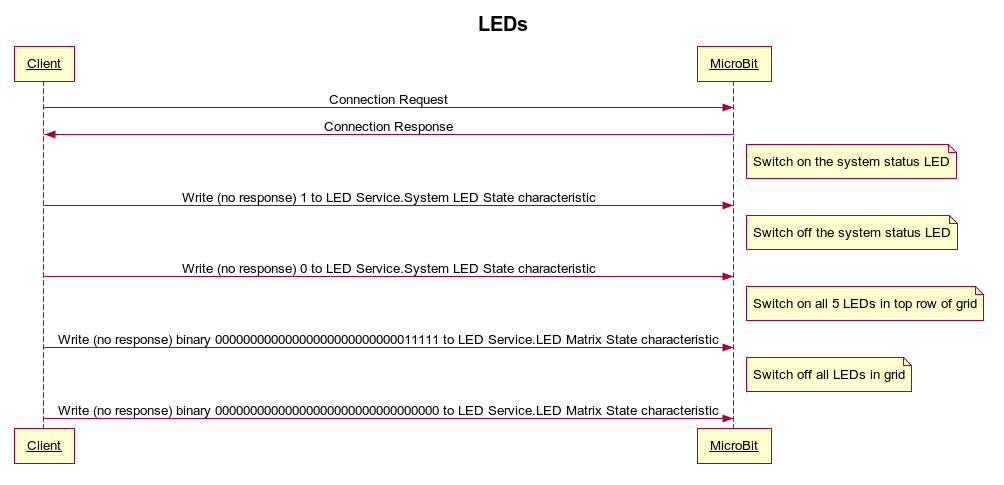


Figure 2 - LED control

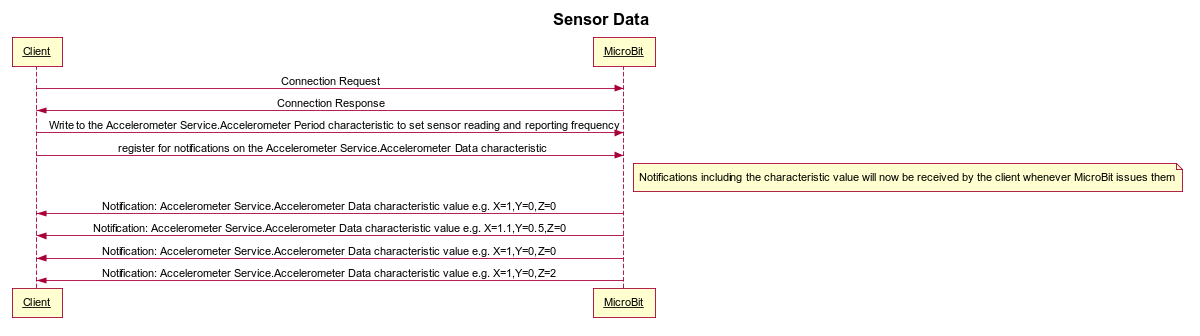


Figure 3 - Accelerometer config and data notifications

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

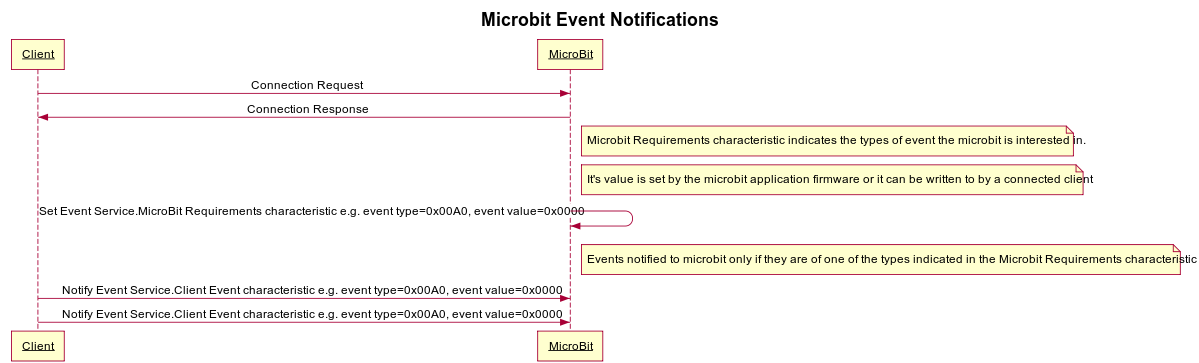


Figure 4 - Client determining Micro:Bit notification requirements and then sending relevant events as they arise

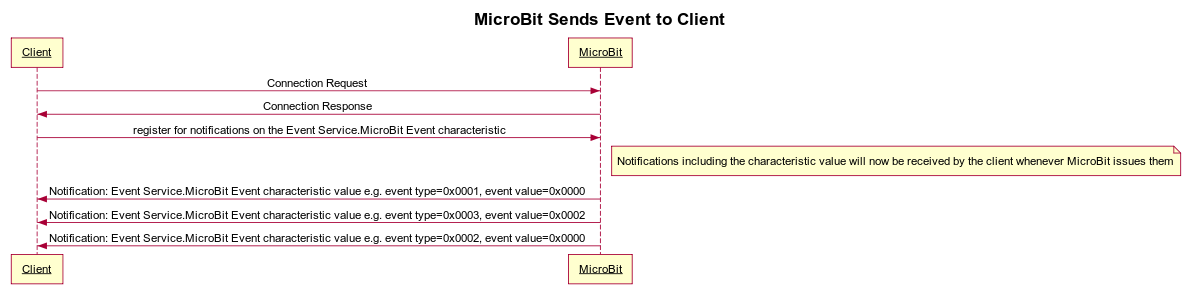


Figure 5 - Micro:Bit sending events / commands to the client

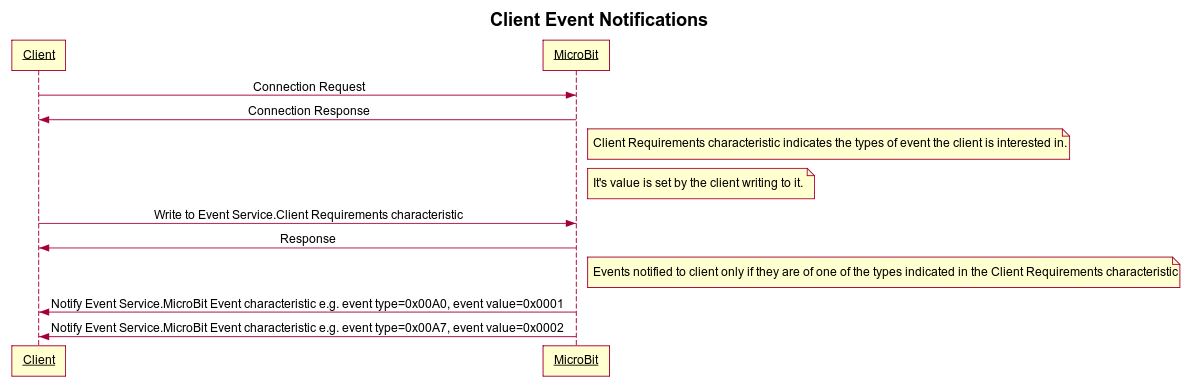


Figure 6 - Client Event Requirements and Notifications