Custom Advertising Data Types

Used in IoT Applications with Bluetooth® Low Energy Wireless Technology

UM70019/D

ON Semiconductor®

USER MANUAL

Overview

This document described custom advertising data formats used by demo and sample applications used by ON Semiconductor IoT platforms.

Custom advertising data formats are used to transmit board specific data frames that typically contain sensor or status information in non−connected broadcasting applications (e.g. sensor beacons).

DATA TYPES DEFINITION AND FORMATS

This part defines the custom data types used for Extended Inquiry Response (EIR), Advertising Data (AD), Scan Response Data (SRD), Additional Controller Advertising Data (ACAD), and OOB data blocks.

Each data type shall only be used in accordance with the requirements specified in Table 1.

O: Optional in this context (may appear more than once in a block)

C1: Optional in this context; shall not appear more than once in a block.

C2: Optional in this context; shall not appear more than once in a block and shall not appear in both the AD and SRD of the same extended advertising interval.

C3: Shall be used only with non−connectable advertising operation when used in Advertising Data context.

Table 1. PERMITTED USAGE OF DATA TYPES

Data Type	Context				
	EIR	AD	SRD	ACAD	OOB
Environmental Service Data (V3)	X	C1	C1	X	X
Environmental Service Data (V5)	X	C1	C1	X	X
Motion Service Data (V1)	X	C1, C3	C1	X	X
Tag Service Data (V0)	X	C1	C1	X	X

ENVIRONMENTAL SERVICE DATA (V3)

Description

This data type is used to transmit additional board−specific sensor data. This data type was designed to match the set of sensors available on RSL10−SOLARSENS−GEVK platform.

Format

The data format consists of 128 bit UUID Service Data frame followed by sensor data for temperature, relative humidity, atmospheric pressure and inclination angles.

Total length of the advertising data frame is 28 bytes, including the length byte.

All multi−byte fields of this advertising data type are defined as little−endian.

Table 2. FORMAT OF ENVIRONMENTAL SERVICE DATA (V3) FRAME

Octet	Field
0	Advertising Data Frame Length
1	Data Type
2 – 17	128 bit UUID
18 – 19	Temperature Value
20 – 21	Humidity Value
22 – 24	Atmospheric Pressure Value
25	Version
26	Tilt X Value
27	Tilt Y Value

Data Type

The sensor data are contained in standard *Service Data* frame with custom 128−bit UUID followed by 10 bytes of service data. The value of this field is set to *«Service Data − 128 bit UUID»* value defined by Bluetooth SIG.

128−bit UUID

Environmental Service uses custom 128 bit UUID 53ac89d1−ec35−5ebb−84e1−8dadb5d4db84

Temperature Value

Temperature is stored as 16 bit signed integer. Value unit is degree Celsius and value is scaled by factor of 100.

Value 2678 represents ambient temperature of 26.78°C.

Humidity Value

Relative humidity is stored as 16 bit unsigned integer. Humidity value is in percent and is scaled by factor of 100. Humidity value 5423 represents relative humidity value of 54.23%.

Pressure Value

Atmospheric pressure is stored as 24 bit unsigned integer. Pressure unit is Pascal and value is scaled by factor of 100. Value 10041400 represents atmospheric pressure of 100414 Pa.

Version Field

Value of this field shall always be set to 0.

Tilt Value

Tilt value is stored as 8 bit signed integer. Tilt values represent the inclination between X or Y axis of accelerometer and horizon. Tilt value is reported in degrees and is in range of ±90°.

The orientation of accelerometer axes for RSL10−SOLARSENS−GEVK is shown in Figure 1.
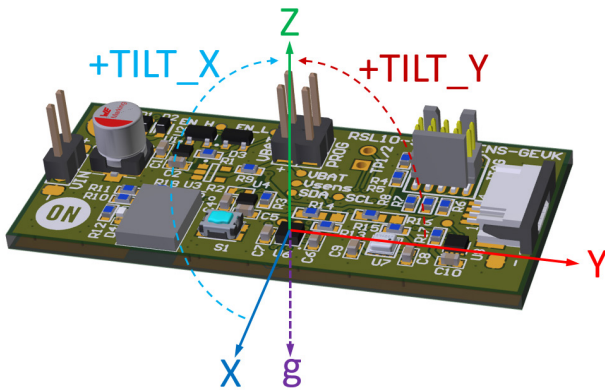

Figure 1. Accelerometer Axes Orientation and Tilt Direction for RSL10−SOLARSENS−GEVK

Example

The following advertisement data was captured by a scanner device:
02 01 04 1B 21 84 DB D4 B5 AD 8D E1 84 BB 5E 35 EC D1 89 AC 53 B2 08 9A 07 38 38 99 03 23 F1

The frame contains Flags frame followed by Environmental Service Data V3 frame offset 3 octets from the start of advertisement data. The following field values are extracted after processing of the packet data:

- Temperature = 22.26°C
- Relative Humidity = 19.46 %
- Atmospheric Pressure = 100414 Pa
- Version = 0
- Tilt X = 35°
- Tilt Y = −15°

ENVIRONMENTAL SERVICE DATA (V5)

Description

This data type is used to transmit additional board−specific sensor data. This data type was designed to match the set of sensors available on RSL10−SENSE−GEVK platform.

Format

The data format consists of 128 bit UUID Service Data frame followed by sensor data for temperature, relative humidity, atmospheric pressure and ambient light.

Total length of the advertising data frame is 28 bytes, including the length byte.

All multi−byte fields of this advertising data type are defined as little−endian.

Data Type

The sensor data are contained in standard *Service Data* frame with custom 128−bit UUID followed by 10 bytes of service data. The value of this field is set to *«Service Data − 128 bit UUID»* value defined by Bluetooth SIG.

Table 3. FORMAT OF ENVIRONMENTAL SERVICE DATA (V5) FRAME

Octet	Field
0	Advertising Data Frame Length
1	Data Type
2 – 17	128 bit UUID
18	Version
19 – 20	Temperature Value
21 – 22	Humidity Value
23 – 25	Atmospheric Pressure Value
26 – 27	Ambient Light Value

128−bit UUID

Environmental Service uses custom 128 bit UUID f0312309−9892−5ce9−9b8c−11610c0d388b

Version Field

Value of this field shall always be set to 0.

Temperature Value

Temperature is stored as 16 bit signed integer. Value unit is degree Celsius and value is scaled by factor of 100.

Value 2678 represents ambient temperature of 26.78°C.

If device does not support reporting of temperature value the field value shall be set to *0x8000*.

Humidity Value

Relative humidity is stored as 16 bit unsigned integer. Humidity value is in percent and is scaled by factor of 100. Humidity value 5423 represents relative humidity value of 54.23 %. If device does not support reporting of humidity value the field value shall be set to *0xFFFF*.

Pressure Value

Atmospheric pressure is stored as 24 bit unsigned integer. Pressure unit is Pascal and value is scaled by factor of 100. Value 10041400 represents atmospheric pressure of 100414 Pa. If device does not support reporting of pressure value the field value shall be set to *0xFFFFFF*.

Ambient Light Value

Value is stored as 16 bit unsigned integer. Light intensity is reported in Lux. If device does not support reporting of ambient light intensity the field value shall be set to *0xFFFF*.

Example

The following advertisement data was captured by a scanner device:
02 01 04 1B 21 8B 38 0D 0C 61 11 8C 9B E9 5C 92 98 09 23 31 F0 00 DD 09 EA 0B 14 9C 9A 20 03

The frame contains Flags frame followed by Environmental Service Data V5 frame offset 3 octets from the start of advertisement data. The following field values are extracted after processing of the packet data:

- Version = 0
- Temperature = 25.25°C
- Humidity = 30.50%
- Pressure = 101325 Pa
- Ambient Light = 800 Lux

See Appendix A for Python sample code for decoding of advertisement data frames that contain Environmental Service Data (V5) data frame.

MOTION SERVICE DATA (V1)

Description

This data type is used to transmit additional board−specific sensor data related to acceleration and orientation of the device. This data type was designed to match the set of sensors available on RSL10−SENSE−GEVK platform.

Format

The data format consists of 128 bit UUID Service Data frame followed by acceleration sensor configuration, acceleration data and orientation data. Total length of the advertising data frame is 31 bytes, including the length byte.

All multi−byte fields of this advertising data type are defined as little−endian.

Table 4. FORMAT OF MOTION SERVICE DATA (V1) FRAME

Octet	Field
0	Advertising Data Frame Length
1	Data Type
2 – 17	128 bit UUID
18	Version
19	Sample index
20	Sensor Settings
21 – 22	Acceleration X
23 – 24	Acceleration Y
25 – 26	Acceleration Z
27	Orientation X
28	Orientation Y
29	Orientation Z
30	Orientation W

Data Type

The sensor data are contained in standard *Service Data* frame with custom 128−bit UUID followed by 10 bytes of service data. The value of this field is set to *«Service Data − 128 bit UUID»* value defined by Bluetooth SIG.

128−bit UUID

Environmental Service uses custom 128 bit UUID 0523e12e−2659−5574−b7b3−dce9dc063620

Version Field

Value of this field shall always be set to 0.

Sample Index

Incrementing 8 bit counter that assigns number to each transmitted data sample. This value increases by 1 for each new data sample extracted from sensors. Consecutive packets with the same sample index number can be ignored as they contain the same sensor data.

Sensor Settings

This field contains additional information about acceleration data contained in the frame and specifies the sample rate, acceleration range and type of acceleration data.

Data type specifies for to interpret acceleration data in the frame. Supported data types are listed in Table 6.

Acceleration range determines the scaling factor that needs to be applied to acceleration data to convert it to g. Supported acceleration ranges are listed in Table 7**.**

Sample rate value determines the sample rate of acceleration and orientation data in Hz.

Table 5. SUPPORTED ACCELERATION RANGES

Bit	Sensor Setting
0	Data Type
1	
2	Acceleration Range
3	
4	Sample Rate
5	
6	
7	

Table 6. STRUCTURE OF SENSOR SETTINGS FIELD

Data Type	Description
0	Linear Acceleration
1	RFU
2	RFU
3	RFU

Table 7. SUPPORTED ACCELERATION DATA TYPES

Acceleration Range	Description
0	±2 g
1	±4 g
2	±8 g
3	RFU

Acceleration Data

Acceleration data is stored as three 16 bit signed integers. The X, Y and Z values represent acceleration applied to the device in the direction of given axis. Axis orientation used by RSL10–SENSE–GEVK is shown in Figure 2.

To process the raw acceleration data contained in the frame it needs to be scaled by reported acceleration sensor *range* and gravity constant *g* using Equation 1.

$$a = \frac{a_{raw}}{32768} \times range \times g$$

(eq. 1)

Orientation Data

Orientation data is stored as four 8 bit signed integers. The rotation is encoded as the four unit–less x, y, z, w components of a unit quaternion. Before using the raw orientation data contained in the frame it must be normalized to unit vector:

$$orientation = \frac{orientation_{raw}}{128}$$

(eq. 2)

The orientation of the device is represented by the rotation necessary to align the East–North–Up coordinates with the device's coordinates[1]. Orientation shares the same device axis orientation from Figure 2.
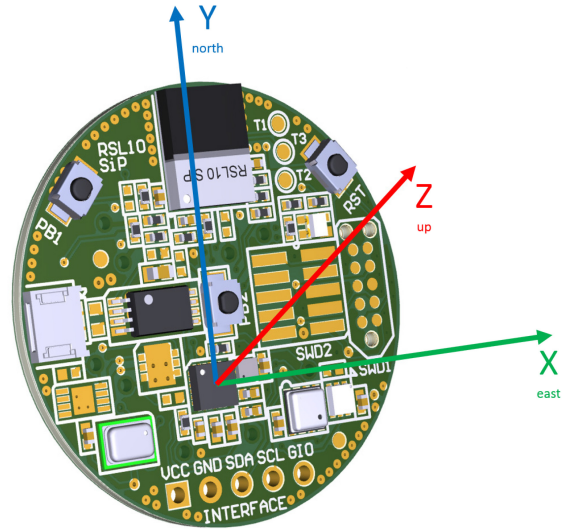
Figure 2. Axis Orientation Used by RSL10– SENSE–GEVK Evaluation Kit

Example

The following advertisement data was captured by a scanner device:
1E 21 20 36 06 DC E9 DC B3 B7 74 55 59 26 2E E1 23 05 00 58 64 4A 02 B0 FF E9 F2 6E 44 86 1E

The frame contains Motion Service Data V1 frame offset 0 octets from the start of advertisement data. The following field values are extracted after processing of the packet data:

- Version = 0
- Sample Index = 58
- Sensor Settings:
    - Sample Rate = 6 Hz
    - Acceleration Range = ±4 g
    - Data Type: Linear Acceleration
- Acceleration X = 0.70174072 m·s^{-2}
- Acceleration Y = −0.09580078 m·s^{-2}
- Acceleration Z = −4.01285522 m·s^{-2}
- Orientation X = 0.859375
- Orientation Y = 0.53125
- Orientation Z = −0.953125
- Orientation W = 0.234375

See Appendix B for Python sample code for decoding of advertisement data frames that contain Motion Service Data (V1) data frame.

---

1. Matches rotation vector definition used by Android Sensor API.

TAG SERVICE DATA (V0)

Description

This data type is used to transmit additional board–specific sensor and status data. This data type was designed to match the set of sensors available on SECO–RSL10–TAG–GEVB platform.

Format

The data format consists of 128 bit UUID Service Data frame followed by firmware version, device state and sensor data for temperature, atmospheric pressure and battery level.

Total length of the advertising data frame is 28 bytes, including the length byte.

All multi–byte fields of this advertising data type are defined as little–endian.

Table 8. FORMAT OF TAG SERVICE FRAME

Octet	Field
0	Advertising Data Frame Length
1	Data Type
2 – 17	128 bit UUID
18	Payload Version
19 – 20	Firmware Version
21	Device State
22 – 23	Ambient Temperature Value
24 – 26	Atmospheric Pressure Value
27	Battery Level

Data Type

The service data are contained in standard *Service Data* frame with custom 128–bit UUID followed by 10 bytes of service data. The value of this field is set to *«Service Data – 128 bit UUID»* value defined by Bluetooth SIG.

128–bit UUID

Tag Service uses custom 128 bit UUID:

`edc5e03b–21b7–5637–a616–fa11565e125f`

Payload Version Field

This field indicates the version of all payload fields that follow. Value of this field shall always be set to 0.

If other value is present then the Data Frame may use updated data format that might not match with the information provided below and should not be parsed.

Firmware Version Field

This 2 byte field provides information about the firmware revision of the device. The field consists of Major, Minor and Patch version numbers of the firmware image.

This versioning scheme matches with the default Firmware Over The Air (FOTA) implementation provided with FOTA enabled RSL10 evaluation kits. Reported version shall match with the version exposed by the device when switched into FOTA mode.

Table 9. FIRMWARE VERSION FIELD STRUCTURE

Octet	Bit	
	7 – 4	3 – 0
19	major [3:0]	minor [3:0]
20	patch [7:0]	

Device State Field

This field exposes additional information about the current state of the device including Device State, Motion event counter and Button event counter.

Table 10. DEVICE STATE FIELD STRUCTURE

Octet	Bit		
	7 – 6	5 – 3	2 – 0
21	device_state [1:0]	motion_ind [2:0]	button_ind [2:0]

Device State identifies current mode of operation of the device. This may indicate expected advertising interval and sensor configuration to the scanner.

Table 11. SUPPORTED DEVICE STATE VALUES

device_state [1:0]	Description
0x0	Default State – Idle state of the device.
0x1	Triggered State – Usually entered for a limited time after an external trigger occurs.
0x2 – 0x3	RFU – Reserved for future use.

Motion event counter is a free–running counter that is incremented every time after an acceleration event changes the state of device.

Button event counter is a free–running counter that is incremented every time after a button press event changes the state of device.

A scanner device can determine which event caused transition from Default to Triggered mode by comparing motion and button counters to previous captured packets.

Ambient Temperature Field

Temperature is stored as 16–bit signed integer. Used unit is degree Celsius and value is scaled by factor of 100. Value 2678 corresponds to temperature of 26.78 °C.

If device does not support reporting of temperature value the field value shall be set to *0x8000*.

Atmospheric Pressure Field

Atmospheric pressure is stored as 24 bit unsigned integer. Used unit is Pascal and value is scaled by factor of 100. Value 10041400 corresponds to pressure of 100414 Pa.

If device does not support reporting of pressure value the field value shall be set to *0xFFFFFF*.

Battery Level Field

Battery voltage is stored as 8 bit unsigned integer encoded to store values of typical 3 V battery powered use case.

$$V_{bat} = (value \times 9\,mV) + 1009\,mV$$

(eq. 3)

If device does not support reporting of battery voltage then the field value shall be set to *0x00*.

Example

The following advertisement data was captured by a scanner device:

02 01 04 1B 21 5F 12 5E 56 11 FA 16 A6 37 56 B7 21 3B E0 C5 ED 00 10 00 4A ED 09 12 92 93 E8

The frame contains Flags frame followed by Tag Service Data V0 frame offset 3 octets from the start of advertisement data. The following field values are extracted after processing of the packet data:

- Payload Version = 0
- Firmware Version
  - ♦ Major: 1
  - ♦ Minor: 0
  - ♦ Patch: 0
- Device Status
  - ♦ State: Triggered
  - ♦ Motion counter: 1
  - ♦ Button counter: 2
- Temperature = 25.41 °C
- Pressure = 96711.86 Pa
- Battery Voltage: 3.097 V

See Appendix C for Python sample code for decoding of advertisement data frames that contain Tag Service Data (V0) data frame.

Appendix A: Python Sample Code for Unpacking of Packets with Environmental Service Data (V5)

```python
#!/usr/bin/env python3

from binascii import unhexlify
from struct import unpack, unpack_from

# Captured advertising data containing Environmental Service Data (V5) data
# field as captured by scanner.
packet = \
    unhexlify('0201041B218B380D0C61118C9BE95C9298092331F000DD09EA0B149C9A2003')

# Binary pattern that identifies the ESD V5 advertising data field
# Contains AD field length and AD field type followed by ESD V5 128-bit UUID
env_v5_header = unhexlify('1B218B380D0C61118C9BE95C9298092331F0')

# Find start of ESD V5 advertising data in the packet
env_v5_offset = packet.find(env_v5_header)

if env_v5_offset >= 0:
    # Byte offset from start of AD field to first byte of data
    data_offset = env_v5_offset + 18
    # Number of ESD V5 data bytes
    data_size = 10
    data = packet[data_offset:(data_offset + data_size)]

    # Unpack and process the binary data
    out = {}
    # version: unsigned 8-bit integer
    out['version'] = unpack_from('B', data, 0)[0]
    # temperature: unsigned 16-bit integer, little endian, scaled by 100
    out['temperature'] = unpack_from('<h', data, 1)[0] / 100
    # humidity: unsigned 16-bit integer, little endian, scaled by 100
    out['humidity'] = unpack_from('<H', data, 3)[0] / 100
    # pressure: unsigned 24-bit integer, little endian, scaled by 100
    out['pressure'] = unpack('<I', data[5:8] + b'\x00')[0] / 100
    # ambient light: unsigned 16-bit integer, little endian
    out['ambient_light'] = unpack_from('<H', data, 8)[0]

    print(out)
else:
    print("Packet does not contain ESD V5 advertising data field!")
```

Appendix B: Python Sample Code for Unpacking of Packets with Motion Service Data (V1)

```python
#!/usr/bin/env python3

from binascii import unhexlify
from struct import unpack_from
import numpy as np

ACCEL_RANGE = [2, 4, 8, None]
DATA_TYPE = ['Linear Acceleraion', 'RFU', 'RFU', 'RFU']

# Captured advertising data containing Motion Service Data (V1) data
# field as captured by scanner.
packet = \
    unhexlify('1E21203606DCE9DCB3B7745559262EE123050064642200D6FFF1FFEF24983E')

# Binary pattern that identifies the Motion advertising data field
# Contains AD field length and AD field type followed by 128-bit UUID
mot_v5_header = unhexlify('1E21203606DCE9DCB3B7745559262EE12305')

# Find start of motion advertising data in the packet
mot_v5_offset = packet.find(mot_v5_header)

if mot_v5_offset >= 0:
    # Byte offset from start of AD field to first byte of data
    data_offset = mot_v5_offset + 18
    # Unpack binary data
    raw_data = unpack_from('<3B3h4b', packet, data_offset)

    # Process the raw data
    out = {}
    # version: unsigned 8-bit integer
    out['version'] = raw_data[0]
    # sample index: unsigned 8-bit integer
    out['sample_index'] = raw_data[1]

    # sensor settings: 8-bit bitfield
    sensor_settings = {}
    sensor_settings['sample_rate'] = ((raw_data[2] >> 4) & 0x0F)
    sensor_settings['accel_range'] = ACCEL_RANGE[((raw_data[2] >> 2) & 0x03)]
    sensor_settings['data_type'] = DATA_TYPE[(raw_data[2] & 0x03)]
    out['settings'] = sensor_settings

    # Acceleration Data
    accel = np.array(raw_data[3:6])
    # scale to correct range in ms2
    out['accel'] = accel / 32768 * out['settings']['accel_range'] * 9.81

    # Rotation data
    rot = np.array(raw_data[6:10])
    # scale to get unit quaternion
    out['rotation'] = rot / 128

    print(raw_data)
    print(out)
else:
    print("Packet does not contain Motion Service Data V1 frame!")
```

Appendix C: Python Sample Code for Unpacking of Packets with Tag Service Data (V0)

```python
#!/usr/bin/env python3

from binascii import unhexlify
from struct import unpack, unpack_from

DEV_STATE = ['Default', 'Triggered' , 'RFU1', 'RFU2']

# Captured advertising data containing Tag Service Data (V0) data
# field as captured by scanner.
packet = \
  unhexlify('0201041B215F125E5611FA16A63756B7213BE0C5ED001000607A0AB38E93E9')

# Binary pattern that identifies the Tag V0 advertising data field
# Contains AD field length and AD field type followed by the Tag Service
# 128-bit UUID
tag_v0_header = unhexlify('1B215F125E5611FA16A63756B7213BE0C5ED')

# Find start of Tag V0 advertising data in the packet
tag_v0_offset = packet.find(tag_v0_header)

if tag_v0_offset >= 0:
    # Byte offset from start of AD field to first byte of data
    data_offset = tag_v0_offset + 18
    # Unpack binary data
    raw_data = unpack_from('<4Bh3xB', packet, data_offset)

    # Unpack and process the binary data
    out = {}
    out['payload_version'] = raw_data[0]
    out['fw_version'] = {}
    out['fw_version']['major'] = (raw_data[1] >> 4)
    out['fw_version']['minor'] = (raw_data[1] & 0x0F)
    out['fw_version']['patch'] = raw_data[2]
    out['status'] = {}
    out['status']['dev_state'] = DEV_STATE[((raw_data[3] >> 6) & 0x03)]
    out['status']['motion_ind'] = (raw_data[3] >> 3) & 0x07
    out['status']['button_ind'] = raw_data[3] & 0x07
    out['temperature'] = raw_data[4] / 100
    out['batt_voltage'] = ((raw_data[5] * 9) + 1009) / 1000

    # pressure: unsigned 24-bit integer, little endian, scaled by 100
    pressure_data = unpack_from('<I', packet, data_offset + 6)[0]
    out['pressure'] = (pressure_data & 0x00FFFFFF) / 100

    print(out)
else:
    print("Packet does not contain Tag V0 advertising data structure!")
```

Bluetooth is registered trademark of Bluetooth SIG.

PUBLICATION ORDERING INFORMATION

◊