NeoSpectra Micro BLE Interface

Legal Information

Copyright

Patents

The technology discussed in this document may be protected by one or more patent grants.

Granted

The technology discussed in this document is protected by one or more of the following patent grants:
U.S. Patent No. x,xxx,xxx, y,yyy,yyy. International Patent No. xx,xxx,xxx, and so on. Other relevant patent grants may also exist.

Pending

The technology discussed in this document is protected by one or more of the following patent applications:
U.S. Application No. x,xxx,xxx, y,yyy,yyy, and so on. International Application No. x,xxx,xxx, y,yyy,yyy and so on. Other relevant patent applications may also exist.

References

1. Si-Ware Systems. Document Control Procedure. SWS-12010001 a1. September 26, 2012.
2. NeoSpectra™ SWS62231 – Development Kits' Developer Manual. SWS internal document, accessed: 7 March 2018.

Table of Contents

1 Introduction

This document describes the BLE interface of NeoSpectra Micro DVK Bluetooth communication service on the Raspberry PI zero w board.

The BLE communication service is based on Nordic UART Service (NUS).
NUS Application is a firmware example that emulates a serial port over BLE.

The UUID of the Nordic UART Service is **"6E400001-B5A3-F393-E0A9-E50E24DCCA9E"**.

This service exposes two characteristics; one for transmitting and another for receiving.

- Tx Characteristic with UUID **"6E400003-B5A3-F393-E0A9-E50E24DCCA9E"**.
- Rx Characteristic with UUID **"6E400002-B5A3-F393-E0A9-E50E24DCCA9E"**.

When the peer enables notification for the Tx Characteristic, the DVK can send data to the peer as notifications. The DVK will transmit all data over UART as notifications.

Peer can start sending data to the DVK by writing to the Rx Characteristic of the service.

The maximum amount of user data in a packet is 20 bytes. If number of bytes needed is less than 20 bytes, then the rest of user data bytes are neglected.

2 Packets Structure

2.1 Command Packets Format

The following table contains description of each filed of the command packet.

Packet	Size (bytes)	Data Type	Description
Operation ID	1	Int	Specify the operation number requested
scanTime	3	Int	Duration of the scan in milliseconds with a minimum of 10 ms and a maximum of 2^{24} ms
commonWavNum	1	Int	Specify the number of points used for the wave number: 0: Disable common wave number 1: 65 points. 2: 129 points. 3: 257 points. 4: 513 points. 5: 1024 points. 6: 2048 points. 7: 4096 points.
opticalGain	1	Int	0: use the optical gain settings saved on the DVK. 1: use the calculated optical gain settings. 2: use external optical gain settings.
apodizationSel	1	Int	Select one of the Apodization windows: 0: Boxcar

			1: Gaussian 2: Happ-Genzel 3: Lorenz
zeroPadding	1	Int	Specify the number of points used in the FFT: 1: 8k points. 2: 16k points. 3: 32k points.
Mode	1	Int	Select the required run mode: 0: Single mode 1: Reserved (Not supported)

2.2 Response Packets Format

The response which is received as a reply to different command packets has the following format:

Packet 1	Status(1 byte)	Data Length(2 bytes)	Zeros(rest of bytes)
N Packets	Payload packets (20 bytes maximum user data for each). The number of these packets depends on the data length sent in packet 1.		

According to the statue byte in packet 1, the payload packets can be received or not based on the following scheme:

- Status = 0 -> No Errors, payload packet can be acquired.
- Status = otherwise -> Error, no payload packet can be acquired.

3 Commands and Responses

3.1 Commands

3.1.1 runPSD

- Description: Requests to perform a scan and returns a Power Spectral Density (PSD).

Operation ID	3
scanTime	Required
commonWavNum	Required
opticalGain	Required
apodizationSel	Required
zeroPadding	Required
Mode	Required

3.1.2 runBackground

- Description: Request to perform a background reading.

Operation ID	4
scanTime	Required
commonWavNum	Required
opticalGain	Required
apodizationSel	Required
zeroPadding	Required

Mode	Required

3.1.3 runAbsorbance

- Description: Request to perform a scan and returns the absorbance.
- Prerequisite operation: “runBackground”.

Operation ID	5
scanTime	Required
commonWavNum	Required
opticalGain	Required
apodizationSel	Required
zeroPadding	Required
Mode	Required

3.1.4 runGainAdj

- Description: Calculate the required gain for a certain sample.

Operation ID	6
scanTime	Not Required
commonWavNum	Not Required
opticalGain	Not Required
apodizationSel	Not Required
zeroPadding	Not Required
Mode	Not Required

3.1.5 burnGain

- Description: Burns the calculated gain adjustment on the DVK.

Operation ID	7
scanTime	Not Required
commonWavNum	Not Required
opticalGain	Not Required
apodizationSel	Not Required
zeroPadding	Not Required
Mode	Not Required

3.1.6 burnSelf

- Description: Burns the self-correction parameters on the DVK.

Operation ID	8
scanTime	Not Required
commonWavNum	Not Required
opticalGain	Not Required
apodizationSel	Not Required
zeroPadding	Not Required
Mode	Not Required

3.1.7 burnWLN

- Description: Burns the wavelength correction parameters on the DVK.

Operation ID	9
scanTime	Not Required
commonWavNum	Not Required
opticalGain	Not Required
apodizationSel	Not Required
zeroPadding	Not Required
Mode	Not Required

3.1.8 runSelfCorr

- Description: Calculates the self-correction parameters.

Operation ID	10
scanTime	Required
commonWavNum	Required
opticalGain	Required
apodizationSel	Required
zeroPadding	Required
Mode	Not Required

3.1.9 runWavelengthCorrBG

- Description: Takes a background reading for the wavelength correction.

Operation ID	11
scanTime	Required
commonWavNum	Required
opticalGain	Required
apodizationSel	Required
zeroPadding	Required
Mode	Not Required

3.1.10 runWavelengthCorr

- Description: performs the wavelength correction.
- Prerequisite Operation: “runWavelengthCorrBG”.

Operation ID	12
scanTime	Required
commonWavNum	Required
opticalGain	Required
apodizationSel	Required
zeroPadding	Required
Mode	Not Required

3.1.11 restoreDefaults

- Description: restores the default gain and correction parameters.

Operation ID	13
scanTime	Required
commonWavNum	Required

opticalGain	Required
apodizationSel	Required
zeroPadding	Required
Mode	Required

3.1.12 setOpticalSettings

- Description: Select the optical gain settings to be used during the scan.
- This operation has a special command packet format.

Operation ID	27
optical gain value	Two bytes.

3.1.13 setCalibrationWells_1

- Description: Set the first three values of the calibration wells.
- This operation has a special command packet format.
- Note: The calibration wells must be quantized by fraction length of 20.

Operation ID	90
first calibration wells value	Four bytes
second calibration wells value	Four bytes
third calibration wells value	Four bytes

3.1.14 setCalibrationWells_2

- Description: Set the last two values of the calibrationwells.
- This operation has a special command packet format.
- Note: The calibration wells must be quantized by fraction length of 20.

Operation ID	91
fourth calibration wells value	Four bytes
fifth calibration wells value	Four bytes

3.2 Responses

3.2.1 runPSD and runAbsorbance

Number of payload packets is determined according to following rules:

1. commonWavNum in command packet is disabled (Taking value 0):
    - Data received contains y-axis (PSD/Absorbance) double values followed by x-axis (Wave Number) double values.
    - Data length received in packet 1 = number of double values of PSD/Absorbance
    - **Payload packets** = ceil[(data length in packet 1) * 8 * 2 / 20]
    - 8 -> length of double data.
    - 2 -> duplicate data length for both y and x values.
    - 20 -> maximum number of user data bytes per packet.
    - After receiving all bytes for double data from the payload packets, y-axis and x-axis values can be interpreted by constructing double values from the received bytes.

2. commonWavNum in command packet is enabled (Taking any value other than 0):
    - Data received contains y-axis (PSD/Absorbance) double values followed by x-initial and x-step Init64 values.
    - Data length received in packet 1 = number of values of PSD/Absorbance.
    - **Payload packets** = ceil[(data length in packet 1 + 2) * 8 / 20]
    - 2 -> two extra Init64 values for x-initial and x-step.
    - 8 -> length of double data.
    - 20 -> maximum number of user data bytes per packet.
    - After receiving all bytes from the payload packets, y-axis values can be interpreted by constructing double values from the received bytes.
    - X-initial and x-step can be interpreted by constructing Int64 vales from the received bytes.
    - X-axis values can be interpreted as follows:
        i. X(i+1) = X(i) + x-step
        ii. X-initial is value for X(1).
    - After interpreting x-axis values, they have to be converted from Int64 to double as follows:
        i. X(i) = (X(i) >> 3) * 1000
        ii. X(i) = X(i) / (1 << 3)

3.2.2 runGainAdj

- Data length received in packet 1 = 2 bytes which represents the gain value.
- One payload packet is received with gain value represented in the first two bytes.

3.2.3 Rest of Operations

- Data length received in packet 1 = 1
- One payload packet is received with 20 bytes of random data.