ViPen-2. Описание протокола Bluetooth

v1.25 03.02.2023
AndreySchekalev@vibrocenter.ru – протокол

Файлы:
Types_Vipen2.h – структуры и константы ViPen-2
Цветом выделены отличия от протокола ViPen-1+

Изменения в файле

v1.25 03.02.2023	Добавил раздел Q&A – вопросы, которые возникают у программистов при реализации протокола

v1.24 18.04.2022	2. Бекон Advertising -> TVipen2AdvertisingData
	3. Данные TUserData -> TVipen2UserData
	
	Вместо поля Reserv теперь поле с версиями FW
	uint8_t Firmware;
	// Версии FW процессоров:
	// Старшие 4 бита = для SAME70, младшие 4 бита = для CC2640

1. Сервисы ViPen-2

Сервис
(Private service UUID):

413557AA213F42798530D38E41390000
413557AA-213F-4279-8530-D38E41390000

Характеристика для приёма TUserData (п.3)
(Private characteristic UUID):

Read, Notify, 15 байт

42EC1288B8A043DBAE0029F942ED0001
42EC1288-B8A0-43DB-AE00-29F942ED0001

Характеристика для управления прибором
(Private characteristic UUID):

Write 64 байта TVipen2MeasureSetup (п.4)

Read, Notify Статус 2 байта (п.5)

42EC1288B8A043DBAE0029F942ED0002
42EC1288-B8A0-43DB-AE00-29F942ED0002

Характеристика для передачи запроса сигнала (п.6) или текстового лога (п.10)
(Private characteristic UUID):

Write, 2 байта VIPEN2_GET_DATA или VIPEN2_GET_LOG

42EC1288B8A043DBAE0029F942ED0003
42EC1288-B8A0-43DB-AE00-29F942ED0003

Характеристика для приёма сигнала
(Private characteristic UUID):

Indicate, 236 байт
TVipen2_Waveform_Header или TVipen2_Waveform_Data (п.6)
или для приёма текстовой информации (п.10, текстовый массив 234 байта * 20 блоков = 4680 байт)

42EC1288B8A043DBAE0029F942ED0004
42EC1288-B8A0-43DB-AE00-29F942ED0004

2. Бекон Advertising

Обновились ли данные, можно смотреть по полю Timestamp.

Если Timestamp==0 – данных ещё нет.

31 байт, выравнивание на 1 байт

```
#pragma pack(1) // Структуры упакованные
typedef struct stVipen2AdvertisingData
{
	uint8_t Len1;
	uint8_t Type1;
	uint8_t Flag1;

	uint8_t Len9;
	uint8_t Type9;
	char Name9[5];

	uint8_t LenFF;
	uint8_t TypeFF;
	uint16_t ManID;
	uint8_t Addr; // ==0
	uint16_t DeviceNumber;     // Номер прибора
	uint32_t TimeStamp;  // Счётчик 1024 Гц для проверки, что появились новые данные
	int16_t Values[4];   // Velocity, Value, Excess, Temperature
	uint8_t Battery;     // Состояние батареи в процентах 0..100 %
	                     // Старший бит == 1 - прибор заряжается (зелёный светодиод)
	uint8_t Firmware;    // Версии FW процессоров:
	                     // Старшие 4 бита = для SAME70, младшие 4 бита = для CC2640

} TVipen2AdvertisingData;
#define szTVipen2AdvertisingData sizeof(TVipen2AdvertisingData)
static_assert(szTVipen2AdvertisingData == 31, "");
```

```
#define GAP_ADTYPE_FLAGS                    1
#define GAP_ADTYPE_LOCAL_NAME_COMPLETE      9
#define GAP_ADTYPE_MANUFACTURER_SPECIFIC    0xFF

#define GAP_ADTYPE_FLAGS_GENERAL    0x02
#define GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED    0x04

#define TI_COMPANY_ID (0x000D) // Company Identifier: Texas Instruments Inc. (13)


// Состояние батареи в процентах 0..100 %
// Старший бит == 1 - прибор заряжается (зелёный светодиод)
#define VIPEN2_CHARGE      (0x80)



// бекон по-умолчанию
// бекон
static TVipen2AdvertisingData Vipen2AdvertisingData =
{
	0x02, GAP_ADTYPE_FLAGS, GAP_ADTYPE_FLAGS_GENERAL |
GAP_ADTYPE_FLAGS_BREDR_NOT_SUPPORTED,
	0x06, GAP_ADTYPE_LOCAL_NAME_COMPLETE, 'V', 'i', 'P', '-', '2',
	0x14, GAP_ADTYPE_MANUFACTURER_SPECIFIC, TI_COMPANY_ID,
		0x00, 0x0001, 0x00000000, 0, 0, -200, 0, 0, 0
};
```

Velocity = Виброскорость, СКЗ 10..1000Гц, мм/с, *100, (7,1 мм/с = 0x02C6)

Value = Значение зависит от типа запрошенного сигнала (TVipen2MeasureSetup.MeasType) *10, (45 м/с² = 0x01C2):
Пик для Ускорения, м/с²
СКЗ для Скорости, мм/с
Размах для Перемещения, мкм

Excess = Эксцесс ускорения, о.е., *100, (0,1 = 0x000A, -2,0 = 0xFF38) – для диагностики подшипников

Temperature = Температура, град С, *100, (28,3° = 0x0B0E, -10,0° = 0xFC18)

В беконе в поле Battery хранится состояние батареи в процентах 0..100 %
Старший бит == 1 - прибор заряжается (зелёный светодиод)

Поле Firmware = Справочное поле для проверки версий FW процессоров:
Старшие 4 бита = для SAME70 = (текущая версия 43 & 0x0F) << 4
Младшие 4 бита = для CC2640 = текущая версия 6
Сейчас Firmware == 0xB6 (43 & 0x0F | 0x06)
Когда SAME70 выключен, Firmware == 0x06

3. Данные TUserData

Характеристика для приёма TUserData
Read, Notify, 15 байт

42EC1288B8A043DBAE0029F942ED0001
42EC1288-B8A0-43DB-AE00-29F942ED0001

Повторяют данные бекона Advertising

17 байт, выравнивание на 1 байт

```
typedef struct
{
	uint8_t Addr; // ==0
	uint16_t DeviceNumber;	// Номер прибора
	uint32_t TimeStamp;  // Счётчик 1024 Гц для проверки, что появились новые данные
	int16_t Values[4];   // Velocity, Value, Excess, Temperature
	uint8_t Battery;     // В беконе хранится состояние батареи в процентах 0..100 %
                                    // Старший бит == 1 - прибор заряжается (зелёный светодиод)
	uint8_t Firmware;    // Версии FW процессоров:
                             // Старшие 4 бита = для SAME70, младшие 4 бита = для CC2640

} TVipen2UserData;
#define szTVipen2UserData sizeof(TVipen2UserData)
static_assert(szTVipen2UserData == 17, "");


#pragma pack()
```

4. Управление прибором TVipen2MeasureSetup

Характеристика

42EC1288B8A043DBAE0029F942ED0002
42EC1288-B8A0-43DB-AE00-29F942ED0002

Write 64 байта TVipen2MeasureSetup

Позволяет читать состояние прибора и управлять Старт-Стоп-Выключением

Чтобы прибор не выключался через 10 минут, раз в 30 секунд будить его командой VIPEN2_BT_COMMAND_IDLE

Для команды Старт (VIPEN2_BT_COMMAND_START) заполнить структуру TVipen2MeasureSetup
Для остальных команд данные из структуры не испольуются.

```
// Команды, приходящие с Bluetooth
const uint32_t VIPEN2_BT_COMMAND_NONE = (0);    // Нет команды
const uint32_t VIPEN2_BT_COMMAND_START = (1);   // Старт чтения, в структуре - параметры
чтения
const uint32_t VIPEN2_BT_COMMAND_STOP = (2);    // Стоп чтения
const uint32_t VIPEN2_BT_COMMAND_IDLE = (3);    // Пока не отключаться
const uint32_t VIPEN2_BT_COMMAND_OFF = (4);     // Выключиться

typedef uint32_t TBluetoothCommand;
```

```
// Что измеряем ?
#define MEAS_TYPE_SPECTRUM              (0)    // Спектр: Стандартный канал: Ускорение
10-10000 Гц; Скорость 10-1000 Гц; Перемещение 10-200 Гц;
#define MEAS_TYPE_WAVEFORM              (1)    // Сигнал: Стандартный канал: Ускорение
10-10000 Гц; Скорость 10-1000 Гц; Перемещение 10-200 Гц;
#define MEAS_TYPE_SPECTRUM_SLOW         (2)    // Спектр: Тихоходный канал: 0,5-50 Гц;
Ускорение, Скорость, Перемещение
#define MEAS_TYPE_WAVEFORM_SLOW         (3)    // Сигнал: Тихоходный канал: 0,5-50 Гц;
Ускорение, Скорость, Перемещение
#define MEAS_TYPE_SPECTRUM_ENV          (4)    // Спектр огибающей Ускорения 0,5-10 кГц;
Только канал Ускорения
#define MEAS_TYPE_WAVEFORM_ENV          (5)    // Сигнал огибающей Ускорения 0,5-10 кГц;
Только канал Ускорения
#define MEAS_TYPE_COUNT                        (6)

#define MEAS_TYPE_SPECTRUM_MASK         (0)
#define MEAS_TYPE_WAVEFORM_MASK         (1)


#define MEAS_UNITS_ACCELERATION         (0)    // Ускорение
#define MEAS_UNITS_VELOCITY             (1) // Скорость
#define MEAS_UNITS_DISPLACEMENT         (2)    // Перемещение
#define MEAS_UNITS_COUNT                (3)
```

```
// Отсчётов в сигнале
#define SETUP_ALLX_256          (0)    // 256
#define SETUP_ALLX_1K           (1)    // 1024
#define SETUP_ALLX_2K           (2)    // 2048
#define SETUP_ALLX_8K           (3)    // 8192
```

```
#define SETUP_ALLX_COUNT    (4)

// Отсчётов в спектре
#define SETUP_ALLF_100            SETUP_ALLX_256       // 100 + 1
#define SETUP_ALLF_400            SETUP_ALLX_1K // 400 + 1
#define SETUP_ALLF_800            SETUP_ALLX_2K // 800 + 1
#define SETUP_ALLF_3200           SETUP_ALLX_8K // 3200 + 1
#define SETUP_ALLF_COUNT    SETUP_ALLX_COUNT

// Частота семплирования в сигнале
#define SETUP_DX_256_HZ           (0)    // 256 Гц
#define SETUP_DX_640_HZ           (1)    // 640 Гц
#define SETUP_DX_2560_HZ    (2)    // 2560 Гц
#define SETUP_DX_6400_HZ    (3)    // 6400 Гц
#define SETUP_DX_25600_HZ   (4)    // 25600 Гц
#define SETUP_DX_COUNT            (5)

// Верхняя частота в спектре
#define SETUP_FN_100_HZ           SETUP_DX_256_HZ            // 100 Гц
#define SETUP_FN_250_HZ           SETUP_DX_640_HZ            // 250 Гц
#define SETUP_FN_1000_HZ    SETUP_DX_2560_HZ     // 1000 Гц
#define SETUP_FN_2500_HZ    SETUP_DX_6400_HZ     // 2500 Гц
#define SETUP_FN_10000_HZ   SETUP_DX_25600_HZ    // 10000 Гц
#define SETUP_FN_COUNT            SETUP_DX_COUNT

const uint32_t SetupMeasAllX[SETUP_ALLX_COUNT] = { 256, 1024, 2048, 8192 };
const float SetupMeasdX[SETUP_DX_COUNT] = { 1.0f / 256.0f, 1.0f / 640.0f, 1.0f / 2560.0f,
1.0f / 6400.0f, 1.0f / 25600.0f };


// Усреднения для спектров
#define SETUP_AVG_NO        (0)    // Нет усреднений
#define SETUP_AVG_4_STOP    (1)    // Усреднить 4 спектра и остановить измерение
#define SETUP_AVG_10_STOP   (2)    // Усреднить 10 спектров и остановить измерение
#define SETUP_AVG_999             (3)    // Усреднять спектры всё время, пока не нажмут
Стоп
#define SETUP_AVG_COUNT           (4)
```

Эти поля только для производителя – выставлять в 0:

```
#define SETUP_EXTERNAL_SENSOR     (0)    // Сигнал подаётся с датчика
#define SETUP_INTERNAL_DAC        (1)    // Сигнал подаётся с внутреннего ЦАП -
используется для внутренних тестов

#define SETUP_READ_MODE                  (0)    // Режим измерения
#define SETUP_CALIBRATION_MODE    (1)    // Режим калибровки - только для производителя



#pragma pack(4) // Структуры выровнены на 4 байта


// Текущие настройки чтения
typedef struct
{
        TBluetoothCommand Command;

        uint32_t MeasType;   // MEAS_TYPE_xxx
        uint32_t MeasUnits;  // MEAS_UNITS_xxx

        uint32_t AllX;             // SETUP_ALLX_xxx или SETUP_ALLF_xxx
        uint32_t dX;         // SETUP_DX_xxx или SETUP_FN_xxx

        uint32_t Avg;        // SETUP_AVG_xxx
```

```
        uint32_t InternalDAC; // Переключение канала: 0 - Вход с датчиков, 1 - с DAC
        uint32_t CalibrationMode; // 0 - Работа, 1 - Калибровка

        uint32_t Reserv[8];

} TVipen2MeasureSetup;
#define szTVipen2MeasureSetup sizeof(TVipen2MeasureSetup)
static_assert(szTVipen2MeasureSetup == 64, "");
```

5. Статус прибора

Характеристика

42EC1288B8A043DBAE0029F942ED0002

42EC1288-B8A0-43DB-AE00-29F942ED0002

Read, Notify Статус 2 байта

```
// Биты состояния, Read, Notify:
#define VIPEN_STATE_STOPED (0<<0) // Прибор стоит
#define VIPEN_STATE_STARTED(1<<0) // Прибор в режиме измерения
#define VIPEN_STATE_NODATA (0<<1) // Данных нет (после инициализации)
#define VIPEN_STATE_DATA   (1<<1) // Есть данные
```

6. Получение сигналов/спектров

236 байт, , выравнивание на 4 байта
Порядок байт в int16_t – Low8, High8 (little-endian, интеловский)

Всего в сигнале до 8192 отсчётов по 2 байта = 1 заголовочный + до 71 блока (TVipen2_Waveform_Header .ViPen2_Data_Blocks) по 236 байт (по 117 отсчётов).
В спектре до 3201 линий по 2 байта
Лишние отсчёты (8192-ой и дальше) == 0 – их не учитывать.

Запрос измеренных данных:

Write команду в характеристику VIPEN2_GET_DATA

42EC1288B8A043DBAE0029F942ED0003
42EC1288-B8A0-43DB-AE00-29F942ED0003

```
// Команда для запроса сигнала/спектра
#define VIPEN2_GET_DATA           (0x0010) // запросить сигнал/спектр
```

Получение сигнала:

Читать характеристику по Indicate

42EC1288B8A043DBAE0029F942ED0004
42EC1288-B8A0-43DB-AE00-29F942ED0004

Прибор посылает до 72 блоков по 236 байт по Indicate.

Блок 0 – заголовок TVipen2_Waveform_Header

1..71 – данные TVipen2_Waveform_Data

TVipen2_Waveform_Header .ViPen2_Data_Blocks = количество блоков для приёма, включая блок TVipen2_Waveform_Header = 2..72

Смотреть по полю Hdr.ViPen2_Get_Wave_ID == Block.ViPen2_Get_Wave_ID, что сигнал не обновился во время передачи

Смотреть блоки по полю ViPen2_Get_Data_Block, что пришли все блоки

```
#pragma pack(1) // Структуры выровнены на 1 байт

#define VIPEN2_BLUETOOTH_PACKET_DATA_SIZE       (236)  // длина блока данных

#define VIPEN2_STAMPS_IN_BLOCK ((VIPEN2_BLUETOOTH_PACKET_DATA_SIZE-2)/2) // 117 отсчётов
в блоке
```

```
#define VIPEN2_MAX_DATA_BLOCKS  (VIPEN2_WAVEFORM_COUNT / VIPEN2_STAMPS_IN_BLOCK + 1) //
Длина сигнала = до 71 блока

typedef struct
{
	uint8_t ViPen2_Get_Data_Command;  // Команда
	uint8_t ViPen2_Get_Data_Block;		// Номер блока
	uint8_t ViPen2_Get_Wave_ID;		// Счётчик, позволяет проверить, что
качаем тот-же замер
                                                                    // Увеличивается на
1, при запросе заголовка
	uint8_t ViPen2_Data_Blocks;		// Количество блоков с данными, до 72: 0 =
Hdr; 1..71 = отсчёты

	uint32_t Timestamp;  //  Счётчик 1024 Гц, совпадает с User_Data. Timestamp
	float Coeff;  // Коэф перевода данных int16_t в float, 4 байта

	uint32_t DataType;   // MEAS_TYPE_xxx
	uint32_t DataUnits;  // MEAS_UNITS_xxx
	uint32_t DataLen; // Отсчётов
	float DataDX; // Шаг между отсчётами, сек или Гц

	int32_t SpectrumAvg;
	int32_t SpectrumAvgMax;

	int16_t Values[4];   // Velocity, Value, Excess, Temperature

	uint8_t Reading;     // 1 - идёт измерение; 0 - стоит
	uint8_t Align1[3];

	uint8_t Reserv2[VIPEN2_BLUETOOTH_PACKET_DATA_SIZE - 12 * 4];
} TVipen2_Waveform_Header;
#define szTVipen2_Waveform_Header sizeof(TVipen2_Waveform_Header)
static_assert(szTVipen2_Waveform_Header == VIPEN2_BLUETOOTH_PACKET_DATA_SIZE, "");



typedef struct
{
	uint8_t ViPen2_Get_Data_Block;    // Номер блока, 0..71
	uint8_t ViPen2_Get_Wave_ID;       // Счётчик, позволяет проверить, что качаем тот-
же замер
	int16_t Wave[VIPEN2_STAMPS_IN_BLOCK];    // отсчёты = 2 байта знаковое * 117
отсчётов в блоке
} TVipen2_Waveform_Data;
#define szTVipen2_Waveform_Data sizeof(TVipen2_Waveform_Data)
static_assert(szTVipen2_Waveform_Data == VIPEN2_BLUETOOTH_PACKET_DATA_SIZE, "");
```

7.Q&A

Правильно ли я понимаю, что для этого нужно использовать характеристику, отличную от той, что мы используем при обычном замере? То есть нужно использовать 42EC1288-B8A0-43DB-AE00-29F942ED0003? Или 42EC1288-B8A0-43DB-AE00-29F942ED0004? Я не совсем понял, в чем между ними разница?

Что мы получим Сигнал или Спектр - задаётся в параметрах измерения

TVipen2MeasureSetup.MeasType

В 42EC1288-B8A0-43DB-AE00-29F942ED0003 пишется команда VIPEN2_GET_DATA

После этого из 42EC1288-B8A0-43DB-AE00-29F942ED0004 читаются по Indicate блоки (Блок 0 – заголовок TVipen2_Waveform_Header; 1..71 – данные TVipen2_Waveform_Data)

Как указать интервал, в течение которого записывать данные? Или он по умолчанию задан?

Это задаётся в параметрах измерения:

TVipen2MeasureSetup.AllX TVipen2MeasureSetup.dX

Там в структуре есть поле DataType, что оно означает?

Одна из констант MEAS_TYPE_xxx = Сигнал или Спектр.

Обычно, совпадает с заданным в параметрах TVipen2MeasureSetup.MeasType

В этой же структуре есть поле SpectrumAvg и SpectrumAvgMax. Это ведь и есть спектры? То есть не нужно тогда применять алгоритм, который вы прислали ранее?

Это важно только для спектров:

SpectrumAvgMax вычисляется из поля TVipen2MeasureSetup.Avg = заданное количество усреднений спектров.

SpectrumAvg = реальное количество усреднений спектров в замере.

В этой же структуре есть поле Values. Правильно ли я понимаю, что там содержатся всем параметры, которые мы измеряем обычным замером? Какое тогда значение у них будет? Мы ведь измеряем в течение какого-то интервала.

TVipen2_Waveform_Header.Values[] совпадает с TVipen2AdvertisingData.Values[] и TVipen2UserData.Values[] в тот момент, когда запросили данные замера VIPEN2_GET_DATA (за время передачи, если прибор в режиме измерения TVipen2AdvertisingData.Values и TVipen2UserData.Values могут измениться)

Максимальное количество отсчётов ?

В Vipen2 максимальная длина в сигнале = 8192, в спектре = 3201

#define SETUP_ALLX_8K (3) // 8192

#define SETUP_ALLF_3200 SETUP_ALLX_8K // 3200 + 1

это ограничено внутренней памятью процессора = 384 кб

Как измерить данные ?

1. Заполнить структуру TVipen2MeasureSetup с командой Старт и параметрами

Послать её в 42EC1288-B8A0-43DB-AE00-29F942ED0002

const uint32_t VIPEN2_BT_COMMAND_START = (1); // Старт чтения, в структуре - параметры чтения

2. Читаете Статус из 42EC1288-B8A0-43DB-AE00-29F942ED0002

Там должен быть бит

#define VIPEN_STATE_STARTED (1<<0) // Прибор в режиме измерения

3. Прибор постоянно измеряет и обновляет данные внутри.

Если выбран режим Спектр с усреднением

#define SETUP_AVG_4_STOP (1) // Усреднить 4 спектра и остановить измерение

#define SETUP_AVG_10_STOP (2) // Усреднить 10 спектров и остановить измерение

то по окончанию прибор остановит измерения сам.

Иначе будет измерять, пока не получит команду Стоп.

4. Читаете Статус из 42EC1288-B8A0-43DB-AE00-29F942ED0002

Там должен появиться бит

#define VIPEN_STATE_DATA (1<<1) // Есть данные

5. Можно остановить измерение командой

const uint32_t VIPEN2_BT_COMMAND_STOP = (2); // Стоп чтения

А можно и не останавливать - чтение продолжится.

Но если нужен только один замер с точки, то лучше остановить

После остановки можно прочитать Статус из 42EC1288-B8A0-43DB-AE00-29F942ED0002

Там не должно быть бита

#define VIPEN_STATE_STARTED (1<<0) // Прибор в режиме измерения

6. Запрос измеренных данных:

Write команду в характеристику VIPEN2_GET_DATA

42EC1288-B8A0-43DB-AE00-29F942ED0003

7. Читать по Indicate (Read почему-то не работает) из

42EC1288-B8A0-43DB-AE00-29F942ED0004

Блок 0 (236 байт) – заголовок TVipen2_Waveform_Header

8. В заголовке есть поле

  uint32_t Timestamp; // Счётчик 1024 Гц, совпадает с User_Data. Timestamp

При каждом новом измерении оно обновляется и должно быть >0

По нему можно отслеживать, что измерение новое/обновилось.

9. Если новое, то читаете остальные блоки (ViPen2_Data_Blocks штук = HdrWaveVipen2.DataLen / VIPEN2_STAMPS_IN_BLOCK + 1 + 1;)
по Indicate

до 71 блока – данные TVipen2_Waveform_Data

А этот спектр как получить?

считать с прибора сигнал длиной Length

наложить окно Хемминга

засунуть в процедуру БПФ

взять Length / 2.56 + 1 линий комплексного спектра

посчитать амплитуду гармоник спектра

Status из 42EC1288-B8A0-43DB-AE00-29F942ED0002

Ждать статус нужно не 3, а

бит (1<<1) = 2

Статус - это битовая маска

Бит 0 - идёт чтение

Бит 1 - есть данные

Нужно ждать Бит 1.

Вернётся значение = 3, если прибор измеряет следующие данные или

значение = 2, если измерение остановлено.

то есть

if (status & (1<<1)) данные есть, можно посылать запрос на передачу замера.

Неверные данные в беконе

У Вас явно перепутан порядок байт в полях длинее 1 байта.

Мы используем little-endian (интеловский) порядок байт в int16_t и int32_t – сначала младший, потом старший байт.

Это соглашение используется во всех структурах прибора.

Если Вы используете Java, то там big-endian порядок. Вам нужно поменять местами байты.

uint32_t Counter - это счётчик, который отчситывает 1024 раза в секунду после включения прибора.

Там не должно быть огромного числа.

Velocity и Acceleration всегда положительные.

Excess = от -3.0 до 99.9. Может быть отрицательным.

Температура ~ от -50.0 до 250.0. Может быть отрицательным.

При направлении ручки в воздух измеряется температура стёклышка прибора.

Обычно она равна комнатной температуре вокруг.

показания всегда некорректные

Проверьте выравнивание в структуре бекона.
Всё должно быть выровнено на 1 байт (без выравнивания), длина структуры = 29 байт

Соединение по GATT разрывается через 80 секунд

В приборе, после соединения по GATT, если ничего не посылать по GATT 60 секунд,

то он насильно разрывает соединение GATT и начинает передавать Advertising Beacons.

После этого через 20 секунд Android генерирует статус GATT CONN TIMEOUT.

Тут я нашёл про это:

https://issuetracker.google.com/issues/37074926?pli=1

Это было сделано нами для защиты от зависания Android-программы и освобождения GATT.

Чтобы GATT не закрывался, нужно иногда (например, раз в 10 секунд) посылать команду

VIPEN2_BT_COMMAND_IDLE в 42EC1288-B8A0-43DB-AE00-29F942ED0002

Это также не даст прибору заснуть через 10 минут неактивности.

Что посылает прибор ?

Без соединения по GATT прибор посылает бекон TVipen2AdvertisingData.

Можно, например, стартовать измерение по GATT 42EC1288-B8A0-43DB-AE00-29F942ED0002, отсоединиться и принимать бекон. В нём будут обновляться показания замера (проверять, что они новые можно по полю TimeStamp).

При соединении по GATT беконы не посылаются.
Смотреть данные из характеристики 42EC1288-B8A0-43DB-AE00-29F942ED0001

При отсоединении по GATT беконы снова посылаются.

Что означает мигание светодиода ?

При приёме и посылке по Bluetooth мигает синий светодиод.

При перезагрузке прибора (в результате каой-то проблемы) быстро несколько раз мигает синий светодиод.

При измерении мигает красный светодиод.

Когда прибор на зарядном устройстве, загорается зелёный светодиод.