

Пример USB HID устройства на STM32F4-DISCOVERY

В данной статье речь пойдет об организации обмена информацией между ПК и платой STM32F4-DISCOVERY на основе интерфейса USB с использованием рабочего проекта, разработанного в среде CoIDE CooCox.

Чаще всего реализация проекта происходит в условиях ограниченных временных ресурсов. При этом уделить достаточно времени на решение сопутствующих задач, к которым можно отнести создание интерфейса взаимодействия с компьютером, не всегда представляется возможным. Протокол USB имеет достаточно сложное, громоздкое описание [1], и его реализация наверняка потребует определенных усилий и временных затрат.

Упростить задачу разработки можно применением более простого протокола HID, который изначально предназначался для устройств типа мыши, клавиатуры и джойстика. Однако из-за относительной простоты его реализации разработчики стали массово использовать этот протокол для поддержки различных устройств. К тому же HID-устройство не требует специального драйвера, и его может оказаться достаточно для решения многих поставленных задач низкоскоростного обмена с использованием современного решения на основе USB-интерфейса.

Значительно минимизировать затраты на разработку возможно при наличии хотя бы одного работающего примера, написанного на базе стандартной библиотеки среды программирования. В этом случае процесс адаптации примера под аналогичную пользовательскую задачу может быть выполнен с использованием минимально достаточного объема теоретического материала.

Общие сведения

Наиболее распространенным и простым в реализации типом USB-устройств является класс HID (Human Interface Devices) - устройства человеко-машинного интерфейса. В качестве примеров устройств HID наиболее часто приводятся устройства обеспечивающие взаимодействие между человеком и компьютером, такие как клавиатура, мышь, джойстик. Этот перечень сильно сужает представления о возможностях класса.

На самом деле спецификация класса HID [2,6] дает более широкий перечень устройств — это органы управления пультов и телефонов, различные датчики, медицинское оборудование и даже источники бесперебойного питания. А также устройства, которые не требуют человеческого взаимодействия, но предоставляющие информацию в аналогичном формате — например, считыватели штрих-кода, термометры, вольтметры, устройства, содержащие индикаторы и специализированные дисплеи. Все эти устройства могут быть отнесены к классу HID, несмотря на то что не имеют привычного интерфейса взаимодействия.

Драйвер для устройств HID интегрирован в операционную систему и не требует инсталляции при подключении устройства к хосту. Это преимущество способствует упрощению разработки и более широкому распространению этих устройств.

Основным неприятным ограничением HID-устройств является скорость. По спецификации максимальная частота опроса для приема данных ПК (хостом) - 1кГц(1мс), а максимальный размер передаваемого пакета - 64 байта. Путем несложных вычислений получаем, что максимальная скорость обмена может достигать 64000 байт/с (64Кбайт/с). Такая скорость значительно уступает максимальному значению стандарта USB 2.0 для *High-speed* 25—480 Мбит/с.

Организация USB устройства.

Интерфейс USB обеспечивает обмен данными между хостом и множеством периферийных устройств. Распределение пропускной способности шины производится ПК с помощью посылки маркеров. Устройства USB могут являться хабами, функциями или их комбинацией. Хаб обеспечивает дополнительные точки подключения устройств кшине. Функции представляют собой устройства, способные передавать или принимать

данные или управляющую информацию. Перед использованием функция должна быть сконфигурирована. Выполнение соответствующей функции происходит путем обращения хоста к конечной точке устройства. Физически конечные точки (End Point) — это буферы в устройстве USB, через которые происходит обмен данными по шине USB.

Логическую организацию USB-устройства удобно рассматривать в виде дерева рис. 1[3], в котором узлы ветвей (cfg, if, alt) обозначают режим работы. В конце ветви расположена конечная точка (ep) с уникальным адресом, выполняющая определённую функцию.

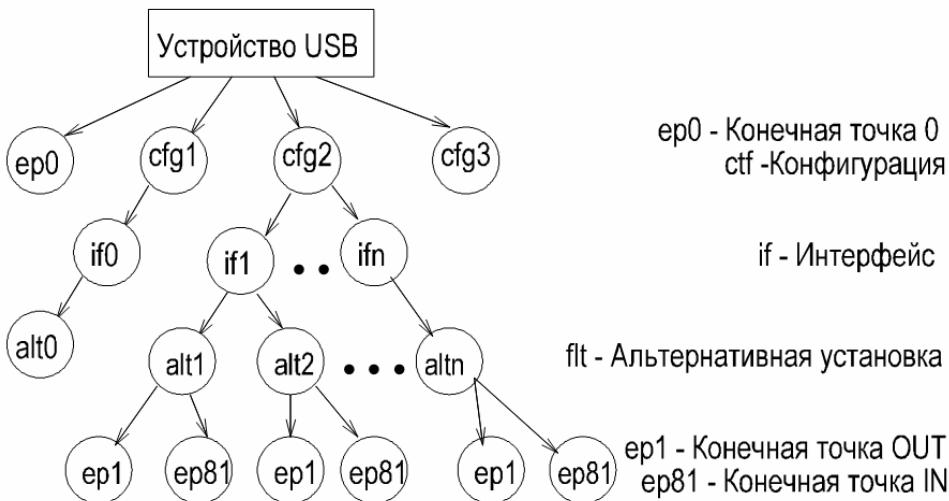


Рис.1 Логическая организация USB устройства.

Конфигурация (cfg) определяет глобальные настройки. Нумерация конфигураций начинается с 1, максимальное значение — 255. Номер 0 означает, что устройство не сконфигурировано. На следующем уровне иерархии находятся интерфейсы (if) — они позволяют разделить режимы по некоторым характерным особенностям в работе устройства. Конфигурация должна содержать хотя бы один интерфейс.

Альтернативные установки (alt) еще более конкретизируют режим работы и описывают частные случаи настройки — они определяют количество поддерживаемых в данном режиме конечных точек-функций и способы работы с ними.

Каждая конечная точка имеет свой уникальный 8-ми разрядный адрес. Конечные точки, входящие в состав альтернативных установок, являются односторонними. Направление передачи для данной конечной точки определяет старший бит ее адреса: 0 в старшем бите адреса имеют точки направления OUT, 1 — точки направления IN. Альтернативная установка может содержать до 15 точек направления OUT с адресами 1..0Fh и до 15 точек направления IN с адресами 81h..8Fh.

Конечная точка 0 (ep0) — называется контрольной, является двунаправленной и доступна в любом режиме работы устройства. Эта точка позволяет хосту опрашивать устройство для определения его типа, параметров и выполнять инициализацию устройства. Все остальные конечные точки находятся в неопределенном состоянии до тех пор, пока хост не выполнит процедуру конфигурирования устройства. В логической структуре возможны ветви, в которых отсутствуют какие-либо конечные точки — они используются для перевода устройства в режим низкого энергопотребления.

Кратко рассмотрим логическую организацию на примере. Допустим, имеется некое устройство управления с USB-интерфейсом для предоставления информации оператору на ПК. Дополнительно предусмотрена сервисная функция переконфигурирования устройства путем загрузки/изменения программы или данных. Очевидно, имеем две различные функциональные задачи (режимы): контроля и перепрограммирования. Для решения каждой задачи в устройстве должен быть определен свой интерфейс.

Интерфейс позволяет разделить режимы по некоторым характерным особенностям в работе устройства. Один интерфейс может быть определен для приема и отображения графических данных, а другой — для индикации числовых данных (параметров). В рассматриваемом примере, очевидно, графические и текстовые данные должны содержать буферы, отличающиеся по длине и частоте опроса. Для этого каждый интерфейс имеет один альтернативный вариант, который определяет количество доступных конечных точек и особенности работы с ними (объем буферов, частота опроса, направление передачи и др.) Набор одновременно поддерживаемых интерфейсов составляет конфигурацию устройства.

Если устройство способно обмениваться разнообразными по назначению пакетами (например, 1-й содержит массив данных каналов АЦП, 2-ой — значение массива переменных, 3-й — значение входных дискретных сигналов и т.д.), то каждый пакет может начинаться с байта идентификатора репорта — Report ID.

После того как хост определил факт подключения нового устройства к шине и его тип, ОС выполняет конфигурирование и подбор подходящего драйвера для данного устройства. Для этого спецификацией предусмотрены общие требования для всех устройств USB: это наличие дескриптора устройства (описание устройства) и механизма выполнения первичной конфигурации [4,5].

Для чтения дескриптора устройства и первичной инициализации используются нулевая конечная точка и стандартные запросы.

Дескриптор представляет собой массив констант для описания свойств устройства. В нем содержится информация: о количестве пакетов поддерживаемых устройством, их длине, назначении каждого байта (бита) в пакете. Таким образом, при подключении к компьютеру устройство с помощью дескриптора сообщает свои параметры, а ОС на основании этих данных и соответствующего драйвера общается с этим устройством.

Хост идентифицирует USB-устройство по его ID вендора и ID продукта (**Vendor ID - VID** и **Product ID - PID**).

Обмен между хостом и HID-устройством происходит с помощью специальной структуры — репорта (пакет фиксированного размера).

Спецификация шины определяет четыре различных типа передачи данных для конечных точек:

Control — используются хостом для конфигурирования устройства при подключении и управления. Протокол обеспечивает гарантированную доставку данных.

Bulk — применяются при необходимости обеспечения гарантированной доставки данных от хоста к функции или от функции к хосту, но время доставки не ограничено.

Isochronous — данный тип канала в основном используется для обмена потоковыми данными. Целостность и доставка данных не контролируются, зато скорость значительно выше, чем для Bulk каналов.

Interrupt — используются для реализации логических прерываний. Используются в том случае, когда требуется передавать одиночные пакеты данных небольшого размера с гарантией времени доставки.

Для обмена пакетом данных хост и устройство выполняют цикл низкоуровневых операций, представляющий последовательность «запрос–данные–подтверждение». Служебная информация с пакетом данных определяет адресата, целостность данных и готовность устройства к обмену. Эти функции реализованы в стандартных библиотеках среды разработки и при умелом использовании призваны существенно упростить процесс разработки программы.

Стандартные дескрипторы USB

В спецификации USB указана группа дескрипторов, которые должны выдаваться устройствами USB в ответ на стандартные запросы. Такие дескрипторы называются стандартными:

- DEVICE — общее описание устройства (всегда один);
- CONFIGURATION — описание поддерживаемых конфигураций. Дескрипторов данного типа может быть несколько;
- INTERFACE — описание интерфейса. Дескрипторов данного типа так же может быть несколько;
- ENDPOINT — описание конечной точки.

Дескрипторы устройства хост получает по запросу — с указанием типа. Таким способом можно запросить дескриптор устройства, квалификатор, дескриптор конфигурации и дескриптор строки.

Дескрипторы интерфейса и конечных точек по отдельности не адресуются, они содержатся в дескрипторе конфигурации.

В приведенных ниже **дескрипторах примера** перед мнемоническими названиями полей префикс b означает байт, w — двухбайтное слово, bm — битовую карту, i — целое число (однобайтный индекс), bcd — неупакованный BCD-формат, id — идентификатор (не число).

Все дескрипторы имеют общий формат. Первый байт указывает длину дескриптора в байтах, второй байт указывает тип дескриптора.

Дескриптор устройства (тип 1)

Дескриптор устройства описывает устройство в целом (версия USB, класс, производитель, модель, протокол, число возможных конфигураций). Для HS-устройств общее описание дополняется дескриптором-квалифиликатором, Device Qualifier Descriptor (типа 6), в котором указывается количество конфигураций при работе на другой скорости.

```
uint8_t USBD_DeviceDesc[USB_SIZ_DEVICE_DESC] =  
{  
    0x12,          /* bLength - Размер дескриптора в байтах = 18 */  
    USB_DEVICE_DESCRIPTOR_TYPE, /*bDescriptorType-Тип = 0x01*/  
    0x00,          /*bcdUSB -Версия USB =0200h означает USB 2.0*/  
    0x02,  
    0x00,          /*bDeviceClass Код класса (назначается USB Org) */  
    0x00,          /*bDeviceSubClass Код подкласса (назначается USB Org)*/  
    0x00,          /*bDeviceProtocol Код протокола (назначается USB Org)*/  
    0x40,          /*bMaxPacketSize40 Макс. размер пакета для EP0 = 64)*/  
    0x77,          /*idVendor (0x0477) VID (назначается USB Org)*/  
    0x04,  
    0x20,          /*idProduct = 0x5620 PID (назначается орг-ей - производителем)*/  
    0x56,  
    0x00,          /*bcdDevice rel. 2.00-номер версии устройства */  
    0x02,  
    1,             /* iManufacturer */  
    /*Индекс строки, описывающей производителя*/  
    2,             /* iProduct */  
    /*Индекс строки, описывающей продукт*/  
    3,             /* iSerialNumber */  
    /*Индекс строки, содержащей серийный номер*/  
    0x01          /*bNumConfigurations Число конфигураций ус-ва на данной скорости */  
};
```

Каждое USB устройство имеет один дескриптор устройства длиной 18 байт.

Поле *bMaxPacketSize0* определяет максимально возможный размер пакета для точки EP0.

Поле *bNumConfigurations* указывает количество конфигураций устройства (как минимум одна).

Если значение полей *iManufacturer*, *iProduct*, *iSerialNumber* не равно нулю, то содержит индекс строкового дескриптора с текстовой информацией о производителе, продукте и серийном номере устройства.

Значения для полей *bDeviceClass*, *bDeviceSubClass*, *bDeviceProtocol* предоставляет организация USB-IF.

Если поле *bDeviceClass* имеет значение 0, то каждый интерфейс устройства имеет собственное описание класса, и все имеющиеся интерфейсы работают независимо. В этом случае поле *bDeviceSubClass* должно быть равно 0.

Значение *bDeviceProtocol* равно 0 — описание протокола будет задано в описании интерфейса. Идентификатор изготовителя устройства, идентификатор продукта и номер версии устройства используются для идентификации устройства.

Дескрипторы конфигурации (тип 2)

Устройство USB может иметь несколько различных конфигураций. Дескриптор конфигурации указывает количество интерфейсов, способ питания, максимальное энергопотребление.

Дескрипторы интерфейса и конечных точек содержатся в теле дескриптора конфигурации. Для каждой из конфигураций устройство содержит свой дескриптор конфигурации.

Дескрипторы интерфейса (тип 4)

Дескриптор интерфейса содержит информацию об одном из интерфейсов, доступных при определенной конфигурации устройства.

Дескриптор конечной точки (тип 5)

Дескриптор конечной точки определяют номер, направление передачи, максимальный размер поля данных и интервал опроса.

```
***** Дескриптор конфигурации *****
static uint8_t USBD_HID_CfgDesc[USB_HID_CONFIG_DESC_SIZ] =
{
    0x09,           /* bLength: Размер дескриптора в байтах=9 */
    USB_CONFIGURATION_DESCRIPTOR_TYPE, /* bDescriptorType: Configuration 0x02 */
    USB_HID_CONFIG_DESC_SIZ, /*wTotalLength: Полная дл возвращаемых дан-х б-т=41*/
    0x00,
    0x01, /*bNumInterfaces: Количество интерфейсов =1 */
    0x01, /*bConfigurationValue: Величина, исп. как аргумент для выбора этой конф-и*/
    0x00, /*iConfiguration: Индекс строкового дескриптора этой конфигурацию=0 */
    0xE0, /*bmAttributes: bus powered and Support Remote Wake-up */
    0x32, /*MaxPower 100 mA: Максимальное энергопотребление в мА (*2) 50*2 */

***** Дескриптор интерфейса *****
/* 09 */
    0x09, /*bLength: Размер дескриптора в байтах=9 */
    USB_INTERFACE_DESCRIPTOR_TYPE, /*bDescriptorType=0x04 */
    0x00, /*bInterfaceNumber: Количество интерфейсов=0*/
    0x00, /*bAlternateSetting: Величина, исп-я для выбора альтернативной установки*/
```

```

0x02, /*bNumEndpoints Количество конечных точек, используемых в интерфейсе=2*/
0x03, /*bInterfaceClass: HID Код класса */
0x00, /*bInterfaceSubClass : 1=BOOT, 0=no boot*/
0x00, /*nInterfaceProtocol : 0=none, 1=keyboard, 2=mouse*/
0, /*iInterface: Индекс строкового дескриптора, описывающего этот интерфейс*/

```

- **bInterfaceNumber** - индекс дескриптора интерфейса.
- **bAlternativeSetting** может использоваться для задания альтернативных интерфейсов.
- **bNumEndpoints** показывает количество конечных точек, используемых в интерфейсе. Эта величина указывается без учета конечной точки 0.
- **bNumEndpoints** - указывает количество идущих далее дескрипторов конечных точек.
- **bInterfaceClass** используется для указания поддерживаемых классов – HID.
- **iInterface** строковое описание интерфейса.

```
***** HID-дескриптор *****
/* 18 */


```

```

0x09, /*bLength: длина HID-дескриптора=9*/
HID_DESCRIPTOR_TYPE, /*bDescriptorType: тип дескриптора - HID 0x21*/
0x00, /*bcdHID: HID номер версии HID 2.0*/
0x02,
0x00, /*bCountryCode: код страны (если нужен)=0*/
0x01, /*bNumDescriptors: Сколько дальше будет report дескрипторов=1*/
0x22, /*bDescriptorType Тип дескриптора - report=22*/
HID_REPORT_DESC_SIZE, /*wItemLength: длина report-дескриптора =84*/
0x00,

```

```
***** Дескрипторы конечной точки *****
/* 27 */


```

```

0x07, /*bLength: Размер дескриптора в байтах =7 */
USB_ENDPOINT_DESCRIPTOR_TYPE, /*bDescriptorType: Дес-р конечной точки (0x05)*/


```

```

HID_IN_EP, /*bEndpointAddress: Адрес конечной точки 0x81=1000 0001
биты 0..3 номер конечной точки =1
биты 4..6 зарезервированы, установлены в 0 (IN)
бит 7 направление 0 = Out, 1 = In */


```

```

0x03, /*bmAttributes: биты 0..1 тип передачи
00 = Control
01 = Isochronous
10 = Bulk
11 = Interrupt = Interrupt endpoint */


```

```

HID_IN_PACKET, /*wMaxPacketSize: Max. размер пакета этой конечной=63*/
0x00,

```

```

0x01, /*bInterval: Интервал опроса данных к. точки.(1 ms)*/


```

```
***** Дескрипторы конечной точки *****
/* 34 */


```

```

0x07, /*bLength: Размер дескриптора в байтах =7 */
USB_ENDPOINT_DESCRIPTOR_TYPE, /*bDescriptorType: Дес-р кт (0x05)*/
HID_OUT_EP, /*bEndpointAddress: Адрес конечной точки 0x01
номер конечной точки =1
бит 7 направление 0 = Out,*/


```

```

0x03, /*bmAttributes: Interrupt endpoint*/

```

```

HID_OUT_PACKET, /*wMaxPacketSize: Max. размер пакета этой конечной=63 */
0x00,

```

```

0x01, /*bInterval: Интервал опроса данных к. точки. (1 mc)*/


```

```

/* 41 */
};

- bEndpointAddress показывает, какую конечную точку описывает этот дескриптор.
- bmAttributes тип передачи - Control, Interrupt, Isochronous или Bulk.
- wMaxPacketSize - максимальный размер в байтах для этой конечной точки.
- bInterval - интервал опроса в мс.

```

Дескриптор репорта

Дескриптор содержит информацию об устройстве и структуре данных пакетов. Длина этого дескриптора зависит от числа и типа данных.

Элемент Usage Page определяет назначение устройства. Спецификацией [5,6] определены стандартные назначения устройств и устройства Vendor Defined Page. В зависимости от выбранного назначения интерпретируются поля данных, задаваемых элементом Usage.

Данный дескриптор репорта создан с помощью бесплатно распространяемой программы HID Descriptor Tool и содержит следующие элементы:

- USAGE PAGE — определяет назначение устройства.
- COLLECTION — объединяет следующие за ним элементы в группу, которую замыкает элемент END COLLECTION.
 - REPORT ID — идентификатор (префикс) сообщения.
 - USAGE — этот элемент задает начало конечной точки ее назначение и описание.
 - LOGICAL_MINIMUM и LOGICAL_MAXIMUM — задают интервал значений.
 - REPORT SIZE — число двоичных разрядов в элементе данных конечной точки.
 - REPORT COUNT — число элементов такой размерности.
 - OUTPUT/INPUT — элементы определяют направление конечной точки и завершают список.

В приведенном ниже дескрипторе репорта заданы два буфера 10 и 63 байта, для каждого из них определены по два REPORT_ID идентифицирующих назначение пакетов.

Буфер длиной 10 байт содержит REPORT_ID = 1,2 для передачи в устройство команд и данных для корректировки.

Буфер длиной 63 байта содержит REPORT_ID = 3,4 для приема с устройства пакетов графических и текстовых данных.

Число двоичных разрядов в элементе данных конечной точки — 8 (другие значения имеют смысл, например, при описании кнопок), диапазон значений — 0-255.

Передача пакета возможна, если его длина и REPORT_ID соответствуют заданным значениям в дескрипторе репорта. В примере входной пакет данных должен содержать 64 байта со значением префикса (первый байт) 3 или 4 (1+63).

Если в дескрипторе REPORT_ID не определен, то первый байт пакета используется для передачи данных и может принимать любое значение.

Интервал опроса указан в дескрипторах конечных точек bInterval =1мс, что соответствует максимальной частоте опроса.

```

/****************************************ReportDesc****/
static uint8_t HID_ReportDesc[HID_REPORT_DESC_SIZE] =
{
    0x06, 0x00, 0xff,      // USAGE_PAGE (Vendor Defined Page)
    0x09, 0x01,            // USAGE (Vendor Usage 1)
    0xa1, 0x01,            // COLLECTION (Application)
//7
    0x85, 0x01,            // REPORT_ID (1)
    0x09, 0x01,            // USAGE (Vendor Usage 1)
    0x15, 0x00,            // LOGICAL_MINIMUM (0)
    0x26, 0xff, 0x00,       // LOGICAL_MAXIMUM (255)
}

```

```

    0x75, 0x08,           // REPORT_SIZE (8)
    0x95, 10,             // REPORT_COUNT (10)
    0x85, 0x01,           // REPORT_ID (1)
    0x09, 0x01,           // USAGE (Vendor Usage 1)
    0x91, 0x02,           // OUTPUT (Data,Var,Abs)

//26
    0x85, 0x02,           // REPORT_ID (2)
    0x09, 0x02,           // USAGE (Vendor Usage 2)
    0x15, 0x00,           // LOGICAL_MINIMUM (0)
    0x26, 0xff, 0x00,     // LOGICAL_MAXIMUM (255)
    0x75, 0x08,           // REPORT_SIZE (8)
    0x95, 10,             // REPORT_COUNT (10)
    0x85, 0x02,           // REPORT_ID (2)
    0x09, 0x02,           // USAGE (Vendor Usage 2)
    0x91, 0x02,           // OUTPUT (Data,Var,Abs)

//45
    0x85, 0x03,           // REPORT_ID (3)
    0x09, 0x03,           // USAGE (Vendor Usage 3)
    0x15, 0x00,           // LOGICAL_MINIMUM (0)
    0x26, 0xff, 0x00,     // LOGICAL_MAXIMUM (255)
    0x75, 0x08,           // REPORT_SIZE (8)
    0x95, 63,              // REPORT_COUNT (63)
    0x85, 0x03,           // REPORT_ID (3)
    0x09, 0x03,           // USAGE (Vendor Usage 3)
    0x81, 0x02,           // INPUT (Data,Var,Abs)

//64
    0x85, 0x04,           // REPORT_ID (4)
    0x09, 0x04,           // USAGE (Vendor Usage 4)
    0x15, 0x00,           // LOGICAL_MINIMUM (0)
    0x26, 0xff, 0x00,     // LOGICAL_MAXIMUM (255)
    0x75, 0x08,           // REPORT_SIZE (8)
    0x95, 63,              // REPORT_COUNT (63)
    0x85, 0x04,           // REPORT_ID (4)
    0x09, 0x04,           // USAGE (Vendor Usage 4)
    0x81, 0x02,           // INPUT (Data,Var,Abs)
    0xc0                  // END_COLLECTION

// END_COLLECTION 84
};


```

Описание примера

Рассматриваемая программа предназначена для **примера** создания **USB HID** устройства на микроконтроллере STM32F4-DISCOVERY и демонстрации обмена данными с ПК. Реализована в среде CooCox CoIDE Version: 1.7.5. с использованием примеров.

Для визуализации обмена данными написана программа VisualDEEx Demo, на языке Delphi с использованием JEDI Visual Component Library (JVCL). Инсталляция программы на ПК и установка дополнительных драйверов не требуется. Программа предоставляется в виде одного исполняемого файла и обязательного ini-файла. Кроме того, папка с указанными файлами содержит сохраненные примеры обмена в текстовом виде. Эти файлы могут быть просмотрены программой VisualDEEx без подключения платы STM32F4.

VisualDEEx позволяет получить пакеты данных с устройства, вывести на индикацию, осуществить корректировку данных в устройстве, сохранить принятые данные в файл.

Пакеты данных подразделяются по способу отображения на графические и числовые/текстовые. ПО VisualDEEx предусматривает раздельный или смешанный прием пакетов.

В состав проекта usb_hid_64_11 входят следующие **основные** файлы:

usbd_desc.c, usbd_desc.h — содержат описание стандартного дескриптора устройства и всего, что с ним связано.

usbd_hid_core.c, usbd_hid_core.h — содержат дескрипторы конфигурации и репорта, функцию приема данных устройством **USBD_HID_DataOut**, передачи, инициализации.

stm32f4xx_conf.h — содержит перечень подключаемых периферийных модулей (при подключении необходимо раскомментировать соответствующие строки).

main.h — файл содержит множество определений типов данных, прототипов функций и текстовое описание массивов для обмена с хостом.

main.c — содержит:

- функцию настройки системы тактирования на частоту 168 МГц **SystemInit();**

- функции настройки портов ввода/вывода для работы со светодиодами и кнопкой:

- **STM32F4_Discovery_LEDInit(LED3)/ (LED4)/ (LED5)/ (LED6);**
- **STM32F4_Discovery_PBInit(BUTTON_USER,**

BUTTON_MODE_GPIO);

- функции настройки USB — **USBD_Init()** и таймера TIM6 на прерывание каждую 1 мс;

- функции передачи графических данных для режимов одной и шести переменных:

- **Tx1var;**
- **Tx6var;**

- функцию заполнения буфера параметров для последующей передачи в ПК **GetBufPar;**

- функцию заполнения буфера переменной DT **GetBufDT;**

- функцию заполнения 6 буферов bTX1[]-bTX6[] графическими переменными по заданным с ПК указателям **GetBuf6Var();**

- функцию настройки конфигурации для передачи (по команде с ПК — пакет ID=1) **RunCommandsRxID1;**

- функцию корректировки переменных (по команде с ПК - пакет ID=2) **RunChDataRxID2.**

Проект предоставлен полностью.

ini-файл

Основное назначение ini-файла — обеспечить гибкость конфигурирования программы VisuaDEEx. При изменении количества переменных или их имен достаточно внести изменения в соответствующие секции ini-файла для формирования новых визуальных компонент отображения параметров с заданными именами. Допустим, первоначально секция просмотра ini-файла имела вид:

```
[View]
1=Ud(B)
2=Id(A)
3=Uo(B)
4=Io(A)
5=Riz(kOm)
```

Где: [View] — имя секции (изменению не подлежат);
1... 5 — порядковые номера переменных;

Ud...Риз — имена переменных.

Для имитации изменения индицируемых параметров добавим еще одну переменную CouPacPar, которая инкрементируется при каждой отправке пакета параметров в ПК.

Под секцию параметров View (“Просмотр”) в массиве Par[64] (main.h) выделены байты с 31 по 50, т.е. 10 параметров по 2 байта. Байты 31-40 содержат значения Ud(B)..Риз(kOm) Помещаем 6-ю переменную CouPacPar в первые два пустых байта — 41 и 42.

0 1 2 3 ID M1 M2 A1.2 uint8_t Par [64]= { 4, 1, 0, 0, ...	63 S9,10 .0{
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 V1.1 V1.2 V2.1 V2.2 V3.1 V3.2 V4.1 V4.2 V5.1 V5.2 V6.1 V6.2 V7.1 V7.2 V8.1 V8.2 V9.1 V9.2 V10.1 V10.2	

И поместим значение переменной **CouPacPar** буфер Par .

```
// View (main.c)
    Par[31]=(p1>>8); Par[32]=p1; Par[33]=(p2>>8); Par[34]=p2; // Ud, Id
    Par[35]=(p3>>8); Par[36]=p3; Par[37]=(p4>>8); Par[38]=p4; //Uo, Io
    Par[39]=(p5>>8); Par[40]=p5; //Риз
Par[41]=(CouPacPar>>8);Par[42]=CouPacPar;
    Par[43]=0;Par[44]=0;Par[45]=0;Par[46]=0; //Резерв
    Par[47]=0;Par[48]=0;Par[49]=0;Par[50]=0; //Резерв
```

Добавим строку ”6 = Сч-к Пар ” в секцию [View] ini-файла и в результате в окне 12 рис.4 появится шестой элемент отображения переменной **CouPacPar** — ”Сч-к Пар”.

Некоторые замечания.

Компоненты индикации параметров и их имена создаются программой VisualDEEx динамически на основании содержания секций ini-файла. По своему усмотрению пользователь может изменить количество переменных в секции и их имена. Тип и кодирование данных должны соответствовать примеру.

Массив Par[64] должен быть заполнен в соответствии с шаблоном и учетом выбранных переменных в ini-файле.

Неиспользуемые байты секции могут содержать любую информацию или могут быть использованы в других секциях с соответствующими изменениями секции [Position].

Объекты секций Сигнализация и Авария кодируются 4 битами. Например, Par [64] =0x14 — соответствует: объект 15 “Вкл.”, параметр объекта 16 “Выше нормы”. Возможные варианты индикации приведены в main.h.

Переменные секций [Setting], [Coefficient] доступны для чтения и записи, остальные секции поддерживают только чтение.

Описание программы проекта.

Абстрактная модель программы предполагает аналого-цифровое преобразование трехфазных токов и напряжений, опрос состояния аппаратов и технологических параметров, а также передачу их значений в ПК посредством интерфейса USB. Для сокращения объема кода и облегчения его понимания в демонстрационную программу внесен ряд упрощений. В частности значения оцифрованных точек синусоид и значение параметров заданы готовыми массивами констант. Некоторые параметры, которые поддерживают корректировку в процессе демонстрации, заданы переменными.

В начале файла main.c размещены функции инициализации и настроек периферии МК, их перечень и краткое назначение изложены выше.

Основной текст программы содержится в бесконечном цикле **while** (1) и подпрограмме прерывания таймера TIM6 **TIM6_DAC_IRQHandler**. В цикле производится заполнение буфера параметров (**GetBufPar**), проверяется условие получение команды с ПК на изменение настроек обмена или корректировку параметров (**RunCommandsRxID1**, **RunChDataRxID2**). В процедуре **Button_state()** проверяется факт нажатия кнопки и производится установка соответствующего флага, который используется для разрешения передачи нескольких пакетов графических данных в режиме одной переменной. Так же в цикле проверяется изменение режима (графика/параметры) и осуществляется соответствующая светодиодная индикация. Указанные светодиоды могут быть включены с закладки “Передача” программы ПК записью в последний байт пакета значения 1 или 2 (режимы “Графика” и “Параметры” должны быть отключены).

Подпрограмма прерывания **TIM6_DAC_IRQHandler** вызывается каждую 1 мс. За счет организации пустых циклов обработки прерываний передача графических данных в ПК производится раз в 4 мс, а параметров — раз в 400 мс. Предварительно проверяется заданный режим (количество переменных в пакете — 1, 6; и тип пакета — графический/текстовый/смешанный). После чего заполняется InBuffer соответствующими данными и вызывается библиотечная функция инициализации передачи данных в ПК **USBD_HID_SendReport(&USB_OTG_dev, InBuffer, 64)** т.е. данные готовы и при получении запроса с хоста будут переданы. Объем InBuffer задан в явном виде — 64 и должен соответствовать значению, указанному в дескрипторе. Направление передачи в именах функций и буферов приводится относительно хоста.

```
336 while (1)
337 {
338     GetBufPar();                                // Заполнение буфера для передачи в ПК
339
340     if (OutBuffer[0]==1)                         // Получение команды режима с ПК
341     {
342         RunCommandsRxID1();
343     }
344
345     if (OutBuffer[0]==2)                         // Изменение переменных по команде с ПК
346     {
347         RunChDataRxID2();
348     }
349     Button_state();
350
351 // Индикация
352 if(ModeGr==0)    STM32F4_Discovery_LEDOff(LED6);
353 if(ModePar==0)   STM32F4_Discovery_LEDOff(LED3);
354
355 if((ModeGr==0)&(ModePar==0))
356     {
357     if (OutBuffer[10]==1) STM32F4_Discovery_LEDOn(LED3);
358     if (OutBuffer[10]==2) STM32F4_Discovery_LEDOn(LED6);
359     if (OutBuffer[10]>2){ STM32F4_Discovery_LEDOff(LED6); STM32F4_Discovery_LEDOff(LED3);}
360     }
361 }
362 }
```

Рис.2 Фрагмент программы. Бесконечный цикл.

Тип обмена задается в поле **bmAttributes** дескриптора конечной точки. В примере выбран тип передачи данных по прерыванию — INTERRUPT (функция **OTG_FS_IRQHandler** файла main.c), который используется при необходимости обмена через заданные временные интервалы. В случае отсутствие данных для передачи в

конечной точке устройства (пустые циклы функции **TIM6_DAC_IRQHandler**) запрос повторяется хостом по истечении заданного времени.

```
364 /*===== Вектор прерывания TIM6 1 mc =====*/
365 void TIM6_DAC_IRQHandler(void)
366 {
367
368     TIM6->SR &= ~TIM_SR UIF;
369     STM32F4_Discovery_LEDToggle(LED5);           // Сброс флага прерывания
370
371     Nullcycle++;
372     if (Nullcycle==2) NullcyclePar++;
373
374 if(ModeGr==1)                                     // Графический режим
375 {
376     if ((ModeDP==1)&(Nullcycle==4))               // Одна переменная
377     {
378         Tx1var();
379     }
380 // Передаем в ПК пакет содержащий 5 измерений 6 пер-х
381     if ((ModeDP==6)&(Nullcycle==4))               // 5 точек IP=3
382     {
383         STM32F4_Discovery_LEDToggle(LED6);
384         GetBufDT();                                // Заполнение буфера DT- 7-я гр-я переменная
385         GetBuf6Var();
386         Tx6var();
387     }
388 }
389 //-----
390 if(ModePar==1)
391 {
392     //Передача параметров в ПК между гр пакетами
393     if (NullcyclePar==100)
394     {
395         CouPacPar++;
396         STM32F4_Discovery_LEDToggle(LED3);
397         USBD_HID_SendReport (&USB_OTG_dev, Par, 64);
398     }
399 } // if(ModePar==1)
400
401 if (Nullcycle==5) Nullcycle=0;
402 if (NullcyclePar==100) NullcyclePar=0;
403 }
```

Рис. 3 Фрагмент программы. Вектор прерывания TIM6.

Работа с программой.

Программа VisualDEEx предназначена для демонстрации возможностей реализованного HID-устройства и служит для приема, отображения и корректировки данных. После подключения платы STM32F4-DISCOVERY к ПК и запуска программы, в окне 1 рис.4 появляется список подключенных HID-устройств к хосту.

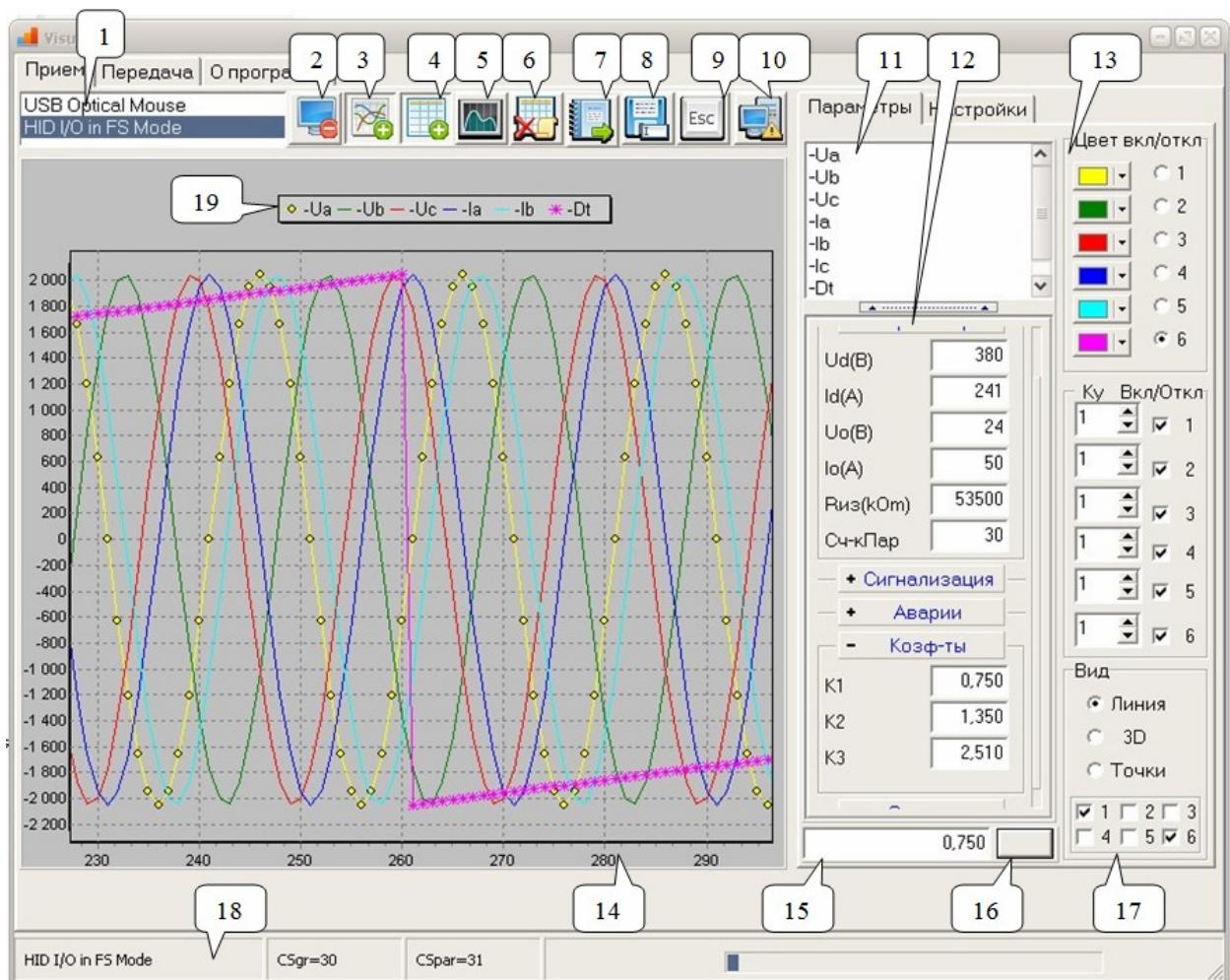


Рис.4 Главное окно программы VisualDEx.

После выбора устройства “HID I/O in FS Mode” становится доступной кнопка 2 активации обмена USB (знак “-” на копке означает, что кнопка не нажата, режим не активирован). Кнопки 3 и 4 предназначены для выбора вида принимаемых данных — графические, текстовые (параметры) или смешанный режим — обе кнопки нажаты.

В окне 11 отображается список имен графических переменных секции [VarSTM] ini-файла. Верхняя часть панели инструментов 13 совместно со списком переменных 11 предназначены для задания порядка отображаемых переменных 19, очередности и цвета вывода. По умолчанию заданы первые 6 переменных списка 11.

Значение параметров выводится в окне 12 в числовом и текстовом виде. Их количество и имена определены в ini-файле. Для корректировки параметров секций “Режим”, “Коэффициенты” и “Задание” необходимо выделить мышью изменяемый параметр, произвести ввод нового значения в окне 15 и сохранить значение в ОЗУ платы DISCOVERY кнопкой 16.

Средняя часть инструментов панели 13 предназначена для отключения отображения соответствующего графика и задания коэффициента усиления Ку. Перерисовка графиков панели 14 происходит в координатах 0...500 по завершению приема очередных 500 точек. Просмотр всего графика осуществляется кнопкой 5, сохранение — 8 (эти действия необходимо производить после останова обмена кнопкой 2 или 9).

Кнопкой 6 производится очистка всех буферов и переменных построения графиков. По нажатию кнопки 7 можно загрузить сохраненный файл для просмотра.

При работе с графиками поддерживаются стандартные приемы:

- увеличение фрагмента графика путем выделения мышью области вправо и вниз;
- восстановление масштаба — выделение вверх и влево;
- буксировка графика — правой кнопкой мыши.

При большом объеме графического файла увеличение фрагмента может быть достигнуто выполнением процедуры выделения области несколько раз. (Пример – рис.7)

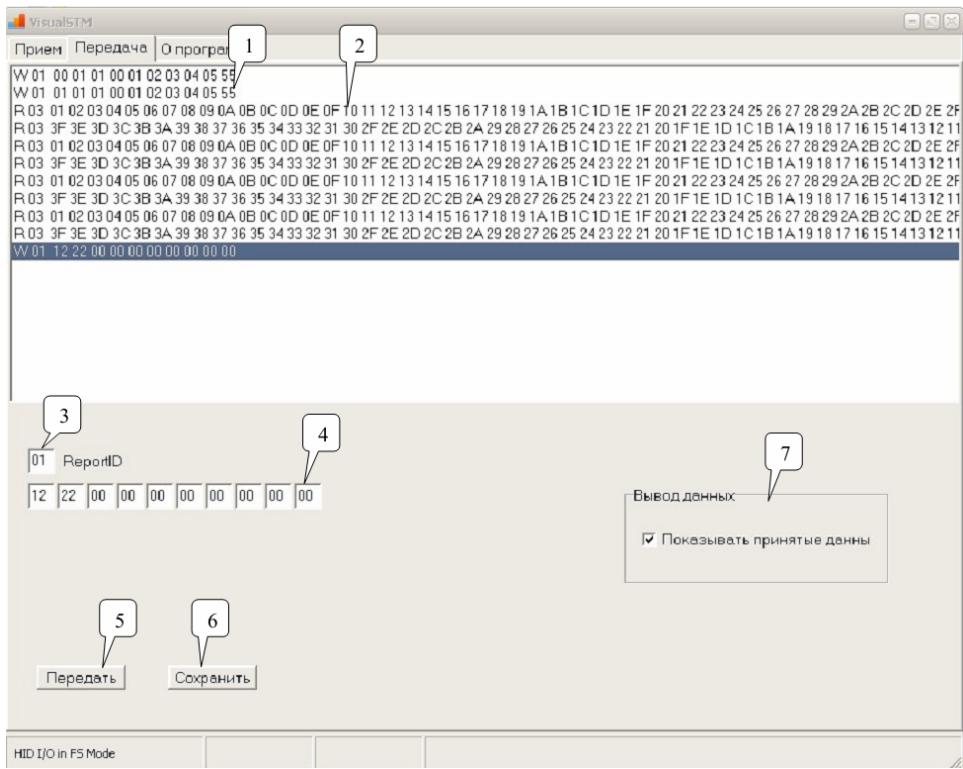


Рис. 5 Вид закладки “Передача”.

На рис. 5 приведен внешний вид закладки “Передача”. Страница содержит:

- 1 - окно отображения содержимого пакетов (W — передача, R — прием);
- 3 - поле ввода Report ID (значение 1 или 2);
- 4 - поле ручного ввода данных для передачи (последний байт используется для управления светодиодами платы);
- 5 - кнопка передать данные;
- 6 - кнопка сохранить данные окна 1 в текстовый файл;
- 7 - отключение индикации принятых данных — 2.

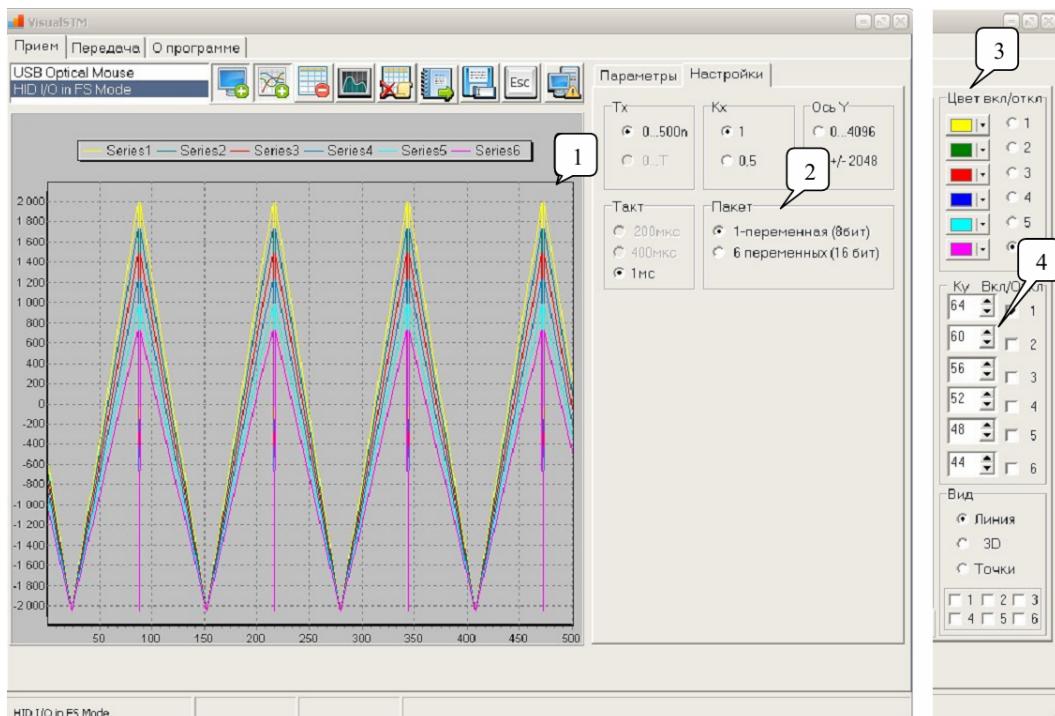


Рис.6 Иллюстрация графического режима одной переменной.

Рисунок 6 иллюстрирует прием данных в режиме одной переменной 2. Режим активируется нажатием/удержанием пользовательской кнопки платы. В этом режиме

хосту передается пакет, содержащий значение переменной-счетчика с модификацией 0-62-0. Значение переменной выводится шестью графиками 1 с различными коэффициентами усиления Ку-4.

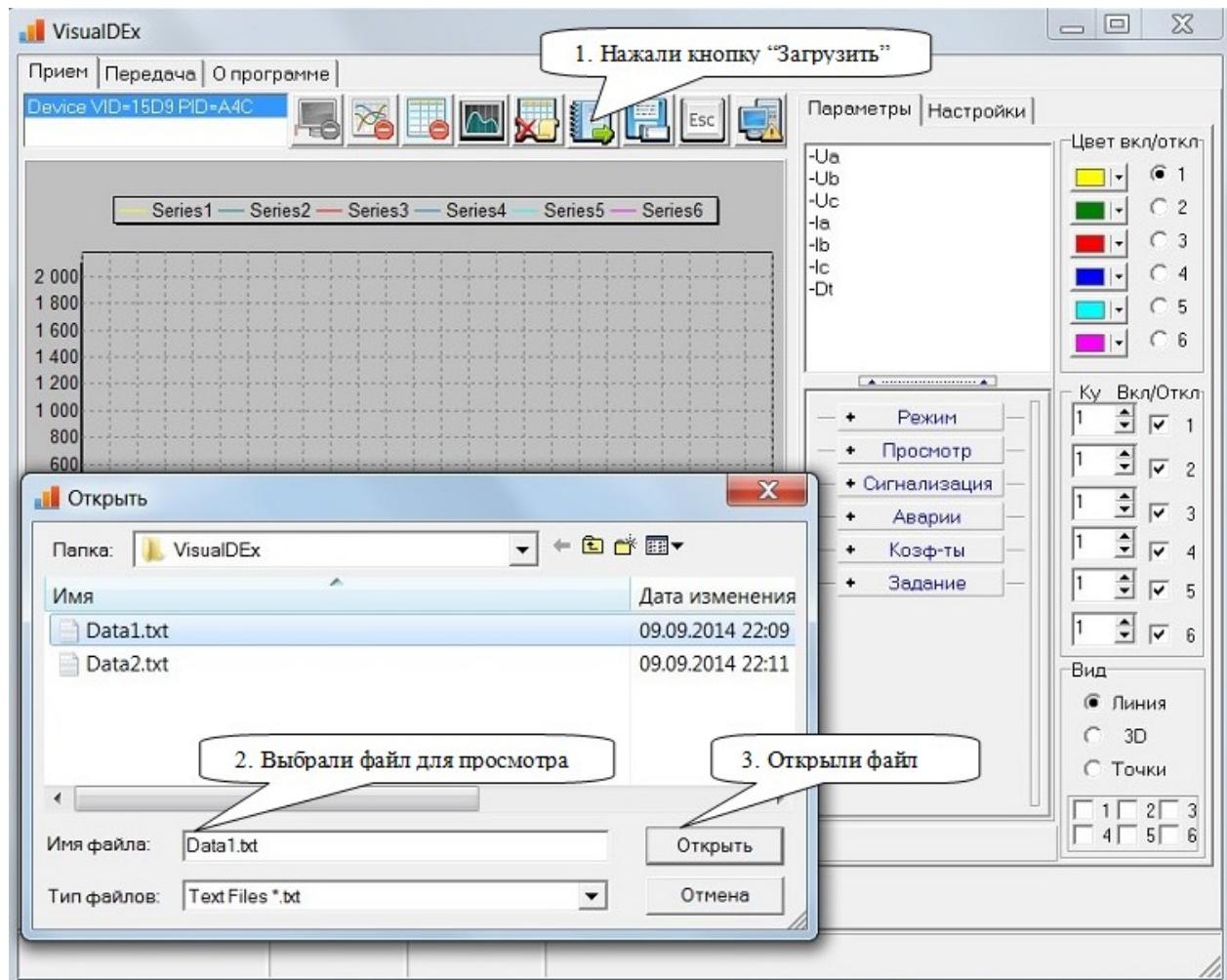


Рис. 7.1 Загрузка файла для просмотра (плата STM32F4-DISCOVERY отключена).

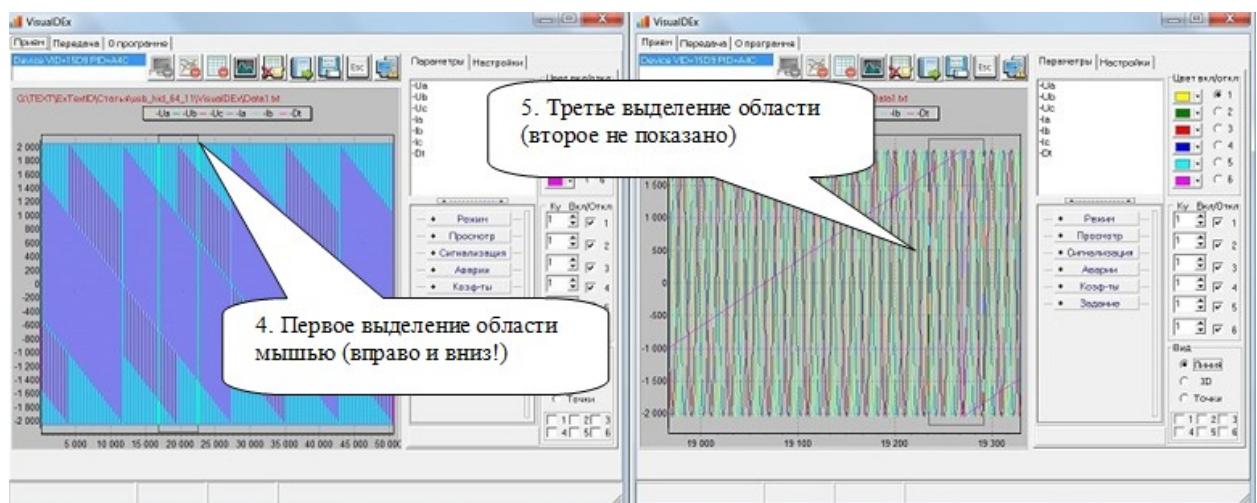


Рис. 7.2 Выделение области просмотра.

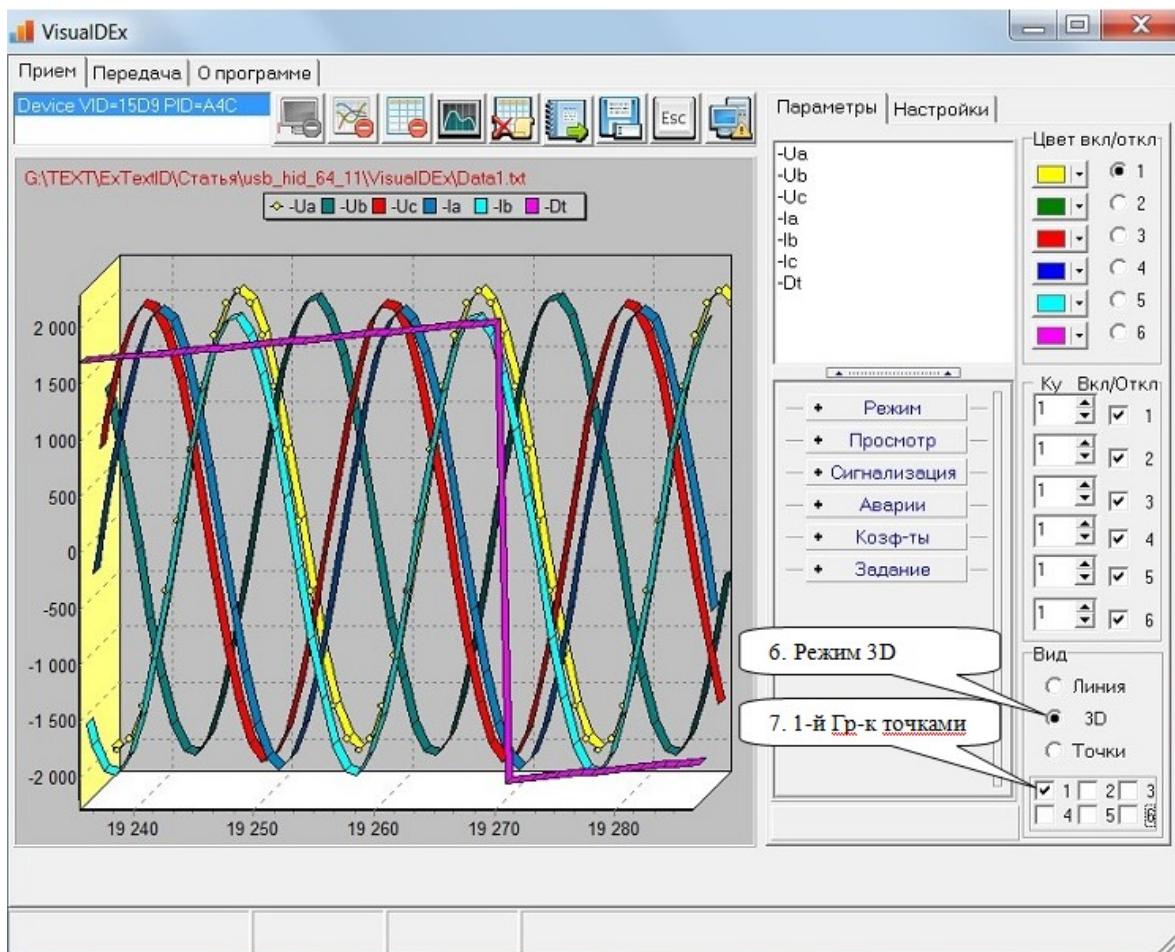


Рис. 7.3 Режим просмотра 3D.

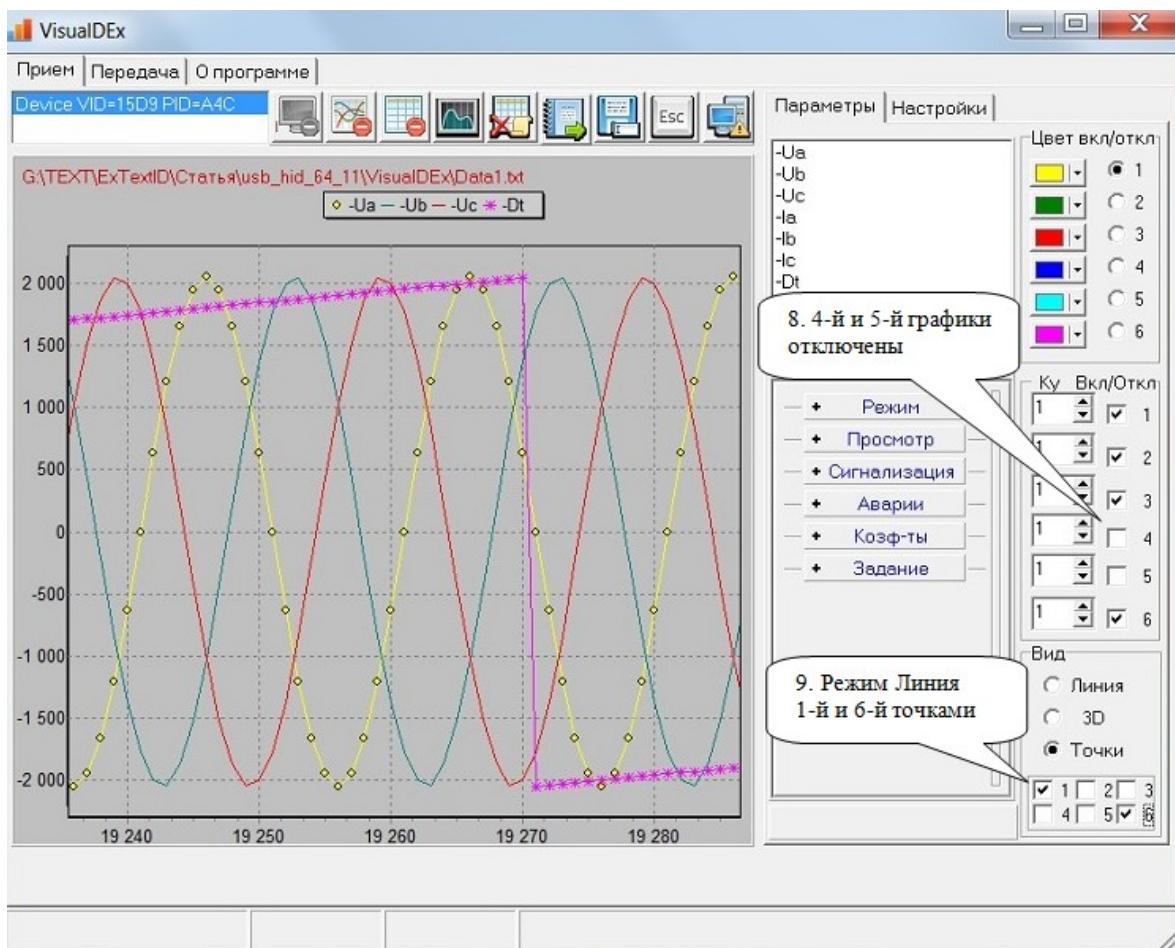


Рис. 7.4 Режим просмотра Линия/Точки.

Заключение

Практически вся работа с USB-интерфейсом в среде СооСох свелась к обработке данных буферов InBuffer и OutBuffer. Входные данные платы доступны после их приема и установки флага функцией **USBD_HID_DataOut** модуля usbd_hid_core.c, а выходные заносятся в буфер InBuffer по мере их готовности к передаче с учетом интервала опроса bInterval = 1мс, указанного в дескрипторе конечной точки (файл usbd_hid_core.c). Инициализация передачи данных в хост производится функцией **USBD_HID_SendReport**.

Для того, чтобы все это функционировало правильно, и нужны стандартные дескрипторы.

Если для решения прикладной задачи обмена через заданные временные интервалы достаточно по одной конечной точке на входные и выходные данные, то возможные варианты модификации дескрипторов сводятся к изменению периода опроса устройства bInterval и размере репортов — REPORT_COUNT. Что касается префикса REPORT_ID, то для устройств, назначение которых определяет разработчик этот параметр, как и значение USAGE[8] не несут смыслового значения, поскольку обработкой принятых данных хостом занимается не драйвер, а прикладная программа.

Изменения в дескрипторе репорта, например, для добавления (сокращения) количества REPORT_ID повлекут за собой изменения длины репорта и обязательно должны быть учтены в HID-дескрипторе. Изменения значений REPORT_COUNT должны быть учтены в размере буферов и процедурах приема/передачи данных.

Исходный текст проекта usb_hid_64_11, а также исполняемый файл программы визуализации с дополнительными файлами можно скачать в архиве.

Б.Горшков, 4bg@mail.ru

Литература

1. USB 2.0 Specification и др. [электронный ресурс] Режим доступа: http://www.usb.org/developers/docs/usb20_docs/usb_20_081114.zip
2. Описание класса устройств, взаимодействующих с человеком (HID). Спецификация— 6/27/01. Версия 1.11 www.microterm.ru/d/20158/d/hid_rus.pdf
3. Дмитрий Чекунов. Практикум программиста USB-устройств. СОВРЕМЕННАЯ ЭЛЕКТРОНИКА. – Цикл статей 2004-2005 г.
4. Агуров П. В. Интерфейс USB. Практика использования и программирования. – СПб: БХВ-Петербург, 2004. – 576 с.
5. USB in a NutShell - путеводитель по стандарту USB [электронный ресурс]. – Режим доступа: <http://microsin.net/programming/arm-working-with-usb/usb-in-a-nutshell-part1.html>
6. HID Usage Tables 10/28/2004 Version 1.12. www.usb.org
7. STM32 и USB-HID - это просто [электронный ресурс] Режим доступа: <http://ravenium.ru>
8. Tutorial about USB HID Report Descriptors. [электронный ресурс] Режим доступа: <http://eleccelerator.com/tutorial-about-usb-hid-report-descriptors/>
- 9 IR + USB HID = очередной пульт для компа [электронный ресурс] Режим доступа: <http://we.easylelectronics.ru/STM32/ir-usb-hid-ocherednoy-pult-dlya-kompa-chast-2.html>
10. USB [электронный ресурс] Режим доступа: <https://ru.wikipedia.org/wiki/USB>