Vinculum II — с чего начать?
Работаем с интерфейсом UART и USB флэш-диском

Сергей ДОЛГУШИН
dsa@efo.ru

Эта статья посвящена работе с хост-контроллером USB Vinculum II и является продолжением статьи [2]. В ней мы рассмотрим базовые принципы работы с периферийным интерфейсом — UART и с USB-хостом на примере записи данных на USB флэш-диск.

Одним из основных применений хост-контроллеров USB является работа с USB флэш-дисками. Не ошибемся, если скажем, что 90% приложений, реализуемых на базе микроконтроллеров Vinculum, работают с USB-накопителями. Во вводной статье [2] были описаны структура приложения для VNC2, общие моменты по инициализации драйверов и конфигурация выводов, здесь же мы продолжим рассказ о Vinculum II и опишем работу с USB-диском и интерфейсом UART.

За основу возьмем проект ***HelloWorld***, поставляемый в составе дистрибутива Vinculum II IDE. Пример ***HelloWorld*** демонстрирует основные принципы работы с USB флэш-диском. В нем показаны инициализация необходимых для работы драйверов и библиотек, а также базовые команды для работы с файлами. В примере реализована циклическая запись фиксированной строки в файл на USB флэш-диске. Мы рассмотрим сам пример ***HelloWorld***, а затем дополним его код возможностью записи в файл данных, полученных по интерфейсу UART.

Для реализации нашей задачи подойдет любой из модулей серии V2DIP (рис. 1) или плата V2Eval. Для связи модуля и ПК по интерфейсу UART удобно использовать переходной кабель TTL-232R-3V3 (рис. 2), который также применяют для программирования микроконтроллера по интерфейсу UART. На рис. 3 приведено схематичное изображение того, как подключается кабель к отладочному модулю V2DIP2-48.

Драйверы, поставляемые FTDI, позволяют работать хост-контроллеру с накопителями, которые соответствуют классу устройств BOMS (Bulk Only Mass Storage). Vinculum II может работать с USB-накопителями на низком (драйвер BOMS) и высоком (драйвер FAT) уровнях. Для стандартных приложений работа на высоком уровне всегда предпочтительней, так как обеспечивается совместимость с ПК. В настоящее время драйвер FAT поддерживает работу с файловыми структурами FAT 32 и FAT 16. USB-накопители должны быть отформатированы так, чтобы размер сектора был равен 512 байтам. Имена файлов и директорий следует вводить большими буквами, они должны состоять из 11 символов. Использование служебных символов и знаков препинания не допускается, кроме пробела. В исходном примере ***HelloWorld*** применяется дополнительная библиотека ввода/вывода ***stdio***, позволяющая работать с привычным форматом имен файлов. Ее использование определяется предпочтениями программиста.

Для работы контроллера VNC2 с флэш-диском необходимы три драйвера. Их иерархическая структура приведена на рис. 4. Все драйверы должны быть инициализированы до запуска ОС, начиная с аппаратного драйвера USB-хоста и заканчивая драйвером FAT. Функция ***usbhost_init()*** обеспечивает инициализацию драйвера USB-хоста контроллера, установку требуемых параметров работы и назначение USB-портов:

```
usbhost_init(-1, VOS_DEV_USB_HOST, &usb_ctx);
```

Параметры, передаваемые в функцию, задают следующие установки:

- Первый подключает или отключает первый USB-порт контроллера. Значение "-1" говорит, что данный порт не используется.
- Второй осуществляет аналогичную функцию по отношению ко второму USB-порту. Порт активируется при передаче ему номера устройства ***VOS_DEV_USB_HOST***. Напомним, что устройством для ОС является любой драйвер периферийного узла, каждое устройство имеет свой номер, под которым регистрируется в ОС.
- Третий параметр передает в функцию структуру, содержащую информацию о том, сколько интерфейсов должен поддерживать данный USB-порт, количество конечных точек, ожидаемое количество изохронных потоков и количество потоков, включающее в себя остальные типы: control, bulk и interrupt. Если в проекте используются оба порта USB, то указываются общие количества, которые поровну делят-

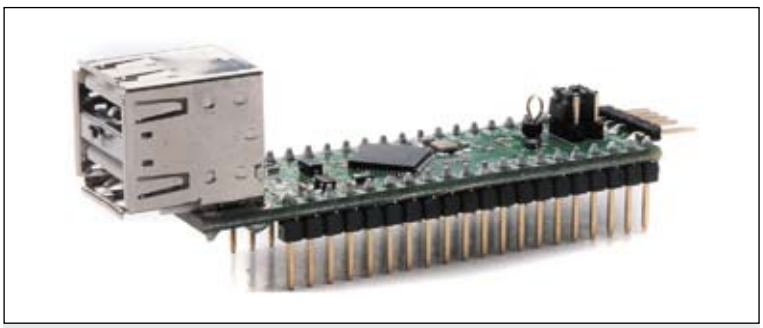

Рис. 1. Модуль V2DIP2-48

Рис. 2. Кабель TTL-232R-3V3

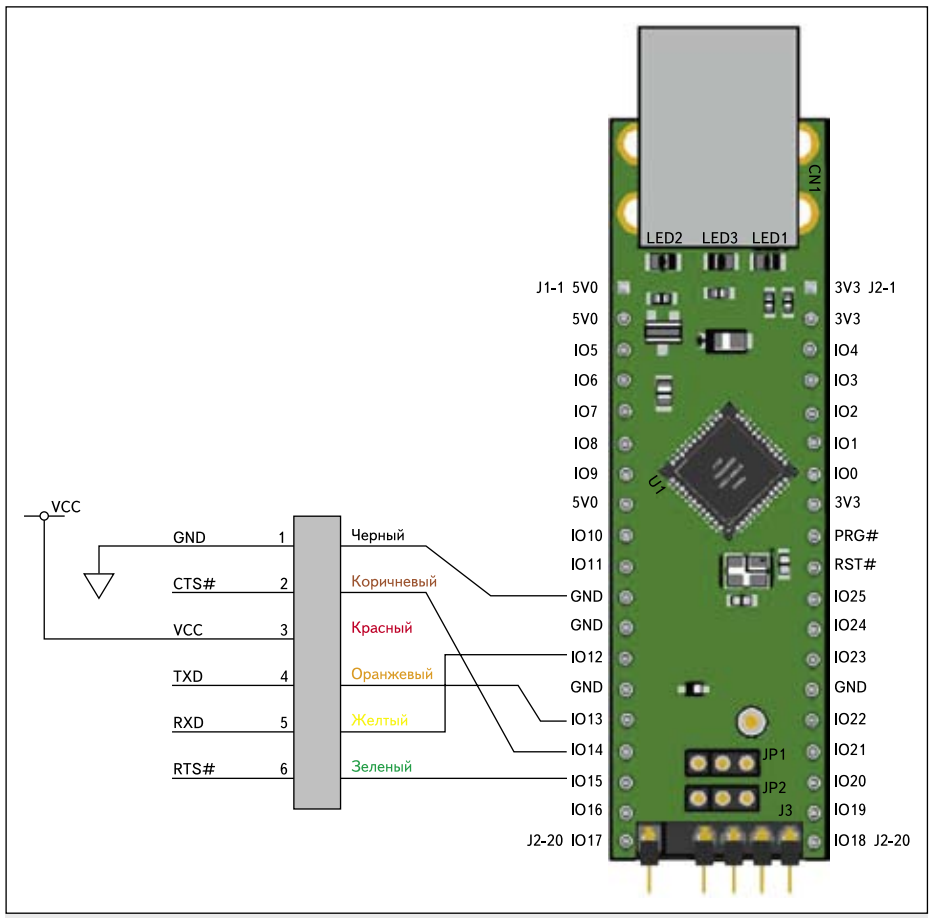

Рис. 3. Подключение кабеля TTL-232R-3V3 к модулю V2DIP2-48

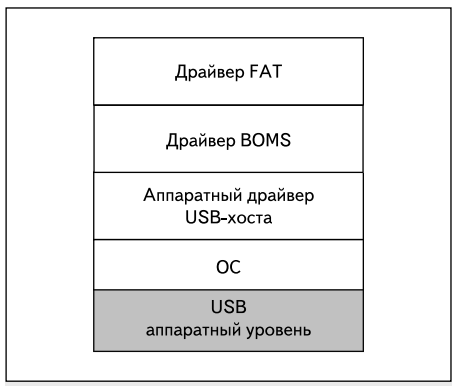

Рис. 4. Иерархическая структура драйверов для работы с USB-накопителями

ся между портами. Стандартное устройство Bulk Only Mass Storage обязательно должно иметь 1 интерфейс, 2 конечные точки Bulk In и Bulk Out. Иногда (это зависит от дополнительных функциональных возможностей, реализованных производителем в конкретном устройстве) оно может иметь дополнительные интерфейсы и, соответственно, конечные точки. Эти дополнительные функции могут потребовать использования другого, отличного от класса BOMS, драйвера.

После инициализации USB-хоста инициализируем драйверы BOMS и FAT. Для этого достаточно передать в соответствующие функции номера устройств.

```
boms_init(VOS_DEV_BOMS);
fatdrv_init(VOS_DEV_FAT);
```

На этом инициализация драйверов, необходимых для работы с USB-накопителем, закончена. Далее описываем поток (thread) (аналогично [2]) и запускаем планировщик ОС.

Пользовательский поток примера ***HelloWorld*** выполняет следующие задачи: определение статуса USB-устройства; определение его класса; подключение драйверов BOMS и FAT; работа с файлом (открытие файла, запись в него строки и закрытие файла).

Рассмотрим эти задачи. Обнаружение подключенного устройства и определение его статуса производится следующим образом:

```
hc_iocb.ioctl_code = VOS_IOCTL_USBHOST_GET_CONNECT_STATE;
hc_iocb.get = &connectstate;
vos_dev_ioctl(hUsb, &hc_iocb);
```

По IOCTL-вызову (третья строчка листинга) запрашиваем драйвер USB-хоста о статусе устройства командой ***VOS_IOCTL_USBHOST_GET_CONNECT_STATE***. Статус устройства возвращается драйвером в переменную connectstate и может иметь следующие состояния:

- ***PORT_STATE_DISCONNECTED*** — устройство не подключено.
- ***PORT_STATE_CONNECTED*** — устройство подключено, но не завершен (не выполнен) процесс его энумерации.
- ***PORT_STATE_ENUMERATED*** — процесс энумерации устройства успешно завершен.

Статус ***PORT_STATE_ENUMERATED*** говорит о том, что USB-устройство доступно для работы. На этом этапе необходимо определить, какое устройство было подключено. То есть нужно определить его класс:

```
hc_iocb.ioctl_code = VOS_IOCTL_USBHOST_DEVICE_FIND_
HANDLE_BY_CLASS;
hc_iocb.handle.dif = NULL;
hc_iocb.set = &hc_iocb_class;
hc_iocb.get = &ifDev;
hc_iocb_class.dev_class = USB_CLASS_MASS_STORAGE;
hc_iocb_class.dev_subclass = USB_SUBCLASS_MASS_STORAGE_SCSI;
hc_iocb_class.dev_protocol = USB_PROTOCOL_MASS_STORAGE_
BOMS;
if (vos_dev_ioctl(hUsb, &hc_iocb) != USBHOST_OK)
{
    // обработка ошибки
}
```

Запрос ***vos_dev_ioctl(hUsb, &hc_iocb)*** по вызову ***VOS_IOCTL_USBHOST_DEVICE_FIND_HANDLE_BY_CLASS*** должен вернуть в программу идентификатор устройства, если оно соответствует классу накопителей (Mass Storage), подклассу SCSI и использует протокол BOMS. В противном случае возвращается значение NULL.

Аналогично происходит поиск устройства другого класса и получение его идентификаторов. Стандартные параметры классов определены в файле ***USB.h***. Обнаружив устройство, соответствующее нашим требованиям, то есть USB флэш-диск, выполняем открытие драйвера BOMS и подключение его к обнаруженному устройству:

```
hBoms = vos_dev_open(VOS_DEV_BOMS);
boms_iocb.ioctl_code = MSI_IOCTL_BOMS_ATTACH;
boms_iocb.set = &boms_att;
boms_iocb.get = NULL;
boms_att.hc_handle = hUsb;
boms_att.ifDev = ifDev;
status = vos_dev_ioctl(hBoms, &boms_iocb);
```

Для подключения драйвера BOMS к устройству выполняем передачу идентификаторов открытого USB-порта ***hUSB*** и USB флэш-диска ***ifDev***. Подключение производится IOCTL-вызовом ***MSI_IOCTL_BOMS_ATTACH***. После успешного завершения подключения BOMS-драйвера переходим к инициализации и подключению драйвера FAT:

```
fat_ioctl.ioctl_code = FAT_IOCTL_FS_ATTACH;
fat_ioctl.set = &fat_att;
fat_att.msi_handle = hBoms;
fat_att.partition = 0;
status = vos_dev_ioctl(hFAT, &fat_ioctl);
```

На этом завершается процесс подключения всех необходимых для работы с диском драйверов, и можно приступать к работе с файла-

ми. Для этого предоставляется стандартный набор команд, таких как открытие и создание файла/директории, запись в файл, чтение из файла и другие команды. С полным перечнем команд можно познакомиться в файле справки среды разработки. Для работы с файлами можно использовать API-функции драйвера FAT или набор команд, описанный в библиотеке ***stdio***.

В исходном примере производитель предлагает использовать команды библиотеки ***stdio***:

```
fsAttach(hFAT);
file = fopen("TEST.TXT", "a+");
    if (file == NULL)
        {
            // обработка ошибки
        }
    if (fwrite(tx_buf, strlen(tx_buf), sizeof(char), file) == -1)
        {
            // обработка ошибки
        }
    if (fclose(file) == -1)
        {
            // обработка ошибки
        }
```

Функция ***fsAttach()*** подключает библиотеку ***stdio*** к драйверу FAT. Функции ввода/вывода этой библиотеки можно использовать с различными периферийными устройствами контроллера. Подключение к нужному драйверу обеспечивается передачей требуемого идентификатора в ***fsAttach()***.

Перед тем как начинать работать с файлом, его необходимо открыть или создать. Это осуществляется командой ***fopen()***, которая создает файл, если он новый, или открывает уже существующий. В эту функцию передаются имя файла и режим доступа к файлу. Режим определяет, какие операции могут быть осуществлены с открытым файлом и в какое место файла устанавливается указатель позиции. Режимы доступа являются стандартными, например, "a+" обозначает, что разрешен доступ к файлу для чтения и записи, указатель позиции в файле установлен в конец файла. В результате успешного выполнения функция ***fopen()*** возвращает указатель на файл.

Запись в файл осуществляется функцией ***fwrite()***. В функцию передаем по порядку:

- Указатель на массив данных, которые будем записывать в файл, в примере это строка вида ***char *tx_buf = "Hello World! \n"***.
- Размер массива данных в байтах.
- Размер элемента в массиве.
- Указатель на файл, полученный при открытии файла.

После выполнения операции записи в файл его необходимо закрыть для вступления изменений в силу. Закрытие выполняется функцией ***fclose()***. Далее закрываются драйверы FAT и BOMS, и приложение начинает новый цикл работы.

Получив представление о том, как происходит работа с файлами на диске, приступим ко второй части — добавлению в проект возможности доступа к файлам на флэш-диске, используя интерфейс UART (рис. 5). Включение в проект нового периферийного блока требует внесения следующих изменений: добавления драйвера интерфейса UART (***uart.a***, ***uart.h*** и ***ioct.h***) и необходимых для его работы переменных; передачи ОС нового значения количества используемых устройств; конфигурации линии ввода/вывода (в исходном проекте ***HelloWorld*** этот этап не требовался).

Рис. 5. Иерархическая структура драйверов для работы с интерфейсом UART

Инициализируем драйвер интерфейса:

```
uartContext.buffer_size = VOS_BUFFER_SIZE_128_BYTES;
uart_init(VOS_DEV_UART,&uartContext);
```

Функция регистрирует драйвер в диспетчере устройств в соответствии с номером ***VOS_DEV_UART***, который был присвоен устройству. Также функция резервирует место в памяти для работы драйвера и под буфер обмена. В этом случае буфер обмена задан равным 128 байтам. Инициализация драйвера, как обычно, должна быть выполнена до запуска планировщика.

После инициализации драйвера зададим необходимые параметры и настройки интерфейса UART. Конфигурация устройства и дальнейшая работа с ним начинается с открытия его драйвера функцией ***vos_dev_open()***:

```
UART = vos_dev_open(VOS_DEV_UART);
```

После этого в любом порядке устанавливаются необходимые параметры. В данном случае настройка начинается с выбора скорости обмена. Скорость может быть установлена в пределах от 300 бод до 3 Мбод, в нашем проекте установим ее равной 9600:

```
uart_iocb.ioctl_code = VOS_IOCTL_UART_SET_BAUD_RATE;
uart_iocb.set.uart_baud_rate = UART_BAUD_9600;
status = vos_dev_ioctl(hUART, &uart_iocb);
```

Выбираем режим контроля потока (для выбранной скорости передачи контроль потока можно не использовать). При больших скоростях обмена использование контроля передачи обязательно, иначе возможна потеря данных. Стандартно для UART поддерживаются режимы аппаратного контроля с использованием сигналов RTS/CTS (DTR/DSR) или программный X-on/X-off:

```
uart_iocb.ioctl_code = VOS_IOCTL_UART_SET_FLOW_CONTROL;
uart_iocb.set.param = UART_FLOW_NONE;
status = vos_dev_ioctl(hUART, &uart_iocb);
```

Выбираем количество бит (7 или 8) в посылке:

```
uart_iocb.ioctl_code = VOS_IOCTL_UART_SET_DATA_BITS;
uart_iocb.set.param = UART_DATA_BITS_8;
status = vos_dev_ioctl(hUART, &uart_iocb);
```

Определяем количество стоповых бит (1 или 2):

```
uart_iocb.ioctl_code = VOS_IOCTL_UART_SET_STOP_BITS;
uart_iocb.set.param = UART_STOP_BITS_1;
status = vos_dev_ioctl(hUART, &uart_iocb);
```

Выбираем контроль четности:

```
uart_iocb.ioctl_code = VOS_IOCTL_UART_SET_PARITY;
uart_iocb.set.param = UART_PARITY_NONE;
status = vos_dev_ioctl(hUART, &uart_iocb);
```

При использовании скоростей обмена более 115 кбод необходимо будет включить поддержку DMA. На меньших скоростях ее можно не использовать:

```
// iocb.ioctl_code = VOS_IOCTL_COMMON_ENABLE_DMA;
// Status = vos_dev_ioctl(hUART, &iocb);
```

На заключительном этапе конфигурации UART активируем прерывание:

```
vos_enable_interrupts(VOS_UART_INT_IEN);
```

Интерфейс UART готов к работе, можно приступать к модификации рабочей функции ***firmware()***. В модифицированном примере воспользуемся API-функциями драйвера FAT. Код исходного примера ***HelloWorld***, начинающийся с функции подключения библиотеки ***stdio*** — ***fsAttach()*** и завершающийся функцией закрытия файла ***fclose()***, заменим следующим:

```
fatctx = fat_open(hBoms, 0, NULL);

do {
        uart_iocb.ioctl_code = VOS_IOCTL_COMMON_GET_RX_
QUEUE_STATUS;
            vos_dev_ioctl(hUART, &uart_iocb);
            num_written = uart_iocb.get.queue_stat; // сколько байт
принято
            if (num_written==20) // переходим к записи в файл, если
количество принятых данных равно 20 байтам
{
        status = fat_fileOpen(fatctx, &fd, "UARTDT TXT", FILE_
MODE_APPEND_PLUS);
        status = fat_fileWrite(&fd, num_written, NULL, hUART, NULL);
        status = fat_fileClose(&fd);
        break;
}
    }while(1);
```

API-интерфейс FAT-драйвера подключается функцией ***fat_open(hBoms, 0, NULL)***. FAT-драйвер может работать с устройствами BOMS, SD-картами и другими устройствами,

которые соответствуют параметрам, определенным в файле ***MSI.h***. В функцию передаем два параметра:

- ***hBoms*** — идентификатор на открытое устройство класса Mass Storage.
- 0 — номер раздела (***partition***) диска, 0 определяет 1 раздел.

Третий параметр в примере является статусом, возвращаемым этой функцией. NULL сообщает функции, что не требуется возвращать статус состояния. Кроме статуса состояния, функция возвращает указатель на структуру fatContext. Эта структура является идентификатором API-интерфейса драйвера и используется для обращения к подключенному устройству Mass Storage.

Затем в цикле организованы опрос блока UART, запись данных в файл и выход из цикла после записи указанного количества байт. Проверка наличия данных в приемном буфере UART осуществляется с помощью IOCTL-запроса ***VOS_IOCTL_COMMON_GET_RX_QUEUE_STATUS***. В ответ на него драйвер возвращает количество данных, принятых и записанных в буфер на текущий момент. Аналогично производится проверка наличия данных для интерфейсов SPI и FIFO.

При наличии требуемого количества байтов данных в приемном буфере приложение переходит к работе с файлом: открытие файла, запись в него и закрытие файла. Открываем или создаем файл функцией ***fat_fileOpen()***, в которую передаются следующие параметры:

- Указатель на API-интерфейс.
- Указатель на область памяти, где хранится информация о файле.
- Имя файла. При использовании этой функции действуют следующие ограничения: используются только заглавные буквы; точка не используется, заменяется пробелом; длина имени, включая пробел и расширение, составляет 11 символов.
- Режим доступа к файлу. ***FILE_MODE_APPEND_PLUS*** указывает, что файл открыт для чтения и записи, позиция указателя установлена на конец файла, то есть новые данные будут добавляться к уже существующим.

После выполнения функция возвращает код успешного завершения или ошибки. С помощью функции ***fat_fileWrite()*** выполняется запись в файл. В функцию должны быть переданы следующие переменные:

- Указатель на область памяти, где хранится информация о файле.
- Количество записываемых байтов.
- Массив данных, которые будут записываться. NULL, если осуществляется прямая запись из периферийного узла контроллера.
- Указатель на периферийный блок, из которого принимаются данные. NULL, если используется промежуточный массив. В примере используется этот вариант передачи данных.
- Указатель, по которому можно получить количество записанных функцией данных. NULL, если не используется.

По окончании записи необходимо закрыть файл для вступления в силу изменений.

Приложение будет ожидать получения данных по интерфейсу UART; при поступлении 20 байт они будут записаны в файл ***UARTDT.TXT***. Этот процесс будет повторяться циклически.

Чтение из файла и передача его содержимого реализуется по аналогичному алгоритму. Чтобы прочитать всю информацию из файла, необходимо внести в листинг следующие изменения:

- Изменить режим доступа к файлу в функции ***fat_fileOpen()*** на ***FILE_MODE_READ***, указатель позиции будет выставлен в начало файла.
- С помощью функции ***fat_dirEntrySize()*** получить размер файла, если хотим считать все данные из файла.
- Заменить функцию записи на функцию чтения — ***fat_fileRead()***. По аналогии с функцией записи передать: указатель на файл; количество байтов, которое требуется прочитать (для чтения всего файла используется значение, полученное функцией ***fat_dirEntrySize()***); массив данных, если будет использоваться промежуточный буфер; указатель на блок UART, если данные будут передаваться по этому интерфейсу; количество считанных в процессе выполнения функции байтов.

Заключение

В завершение отметим, что компания FTDI продолжает развитие программного обеспечения для хост-контроллера Vinculum II. Например, последнее обновление среды разработки включает в себя обновленный FAT-драйвер с новыми командами для копирования файлов и директорий с одного USB-диска на другой. В будущем можно ожидать добавления одновременной поддержки драйвером BOMS SD-карты и USB-дисков. ■

Литература

1. Долгушин С. А. Vinculum II — новый хост-контроллер USB от FTDI // Компоненты и технологии. 2010. № 9.
2. Долгушин С. А. Vinculum II — с чего начать? Работа с портами ввода/вывода // Компоненты и технологии. 2011. № 5.