Реализация профиля клавиатуры USB HID на плате STM32 Mini

Андрей Шаронов, Валерий Володин, Равиль Бикметов (г. Пермь)

Интерфейс USB становится всё более популярным для подключения микроконтроллерных устройств к компьютеру. Чаще всего используется профиль виртуального последовательного порта CDC, но профиль USB HID обладает своими преимуществами. В статье описывается реализация устройства HID и HID-клавиатуры на основе платы STM32 Mini.

На сегодняшний день интерфейс USB широко представлен на персональных компьютерах и зачастую является практически единственным интерфейсом для периферийных устройств. Наиболее распространённым средством подключения микроконтроллера (МК) к порту USB долгое время были ИС типа FT232, эмулирующие виртуальный последовательный порт через выводы интерфейса UART микроконтроллера. С появлением МК, оснащённых контроллером USB, стал использоваться профиль CDC, который также создаёт виртуальный последовательный порт.

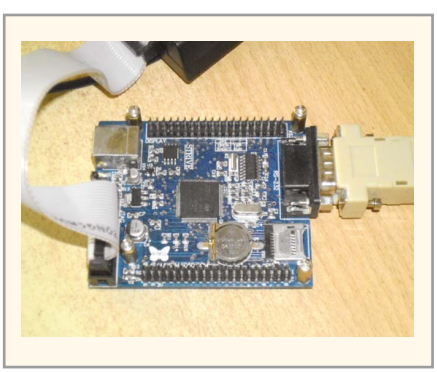

Рис. 1. Внешний вид платы STM32 Mini

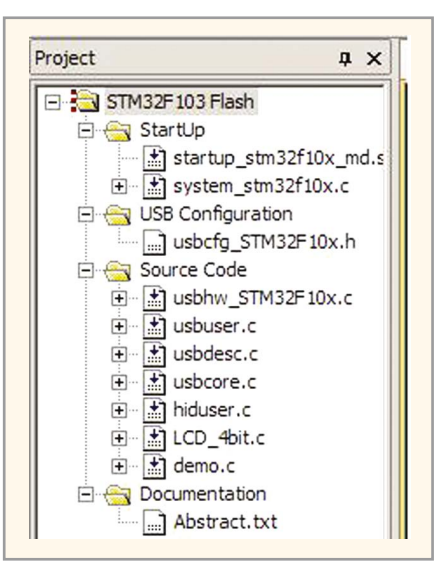

Рис. 2. Дерево проекта устройства USB HID

Наравне с профилем виртуального последовательного порта существует профиль HID, формально предназначенный для устройств взаимодействия с человеком. Фактически, его можно применить для подключения любого прибора, которому достаточно скорости обмена 64 кбит/с (именно с такой скоростью передаются данные в профиле HID). При такой низкой скорости передачи данных профиль HID имеет ряд преимуществ:

- устройство USB HID не требует установки драйверов под ОС Windows;
- в управляющую программу не требуется вводить окно выбора виртуального последовательного порта, к которому подключено устройство;
- профиль USB HID используют стандартные устройства ввода, такие как клавиатура, мышь и джойстик, и разработчик может заставить компьютер «видеть» в своём приборе одно из этих устройств.

Когда была поставлена задача разработки устройства, эмулирующего USB-клавиатуру, выбор профиля HID был очевиден. В качестве аппаратной платформы использовалась плата STM32 Mini компании OurSTM на основе микроконтроллера STM32F103VET. Внешний вид платы показан на рисунке 1.

Перенос стандартного примера USB HID от Keil на плату STM32 Mini

Прежде чем приступать к проекту эмулятора USB-клавиатуры для платы STM32 Mini, было решено адаптировать для этой платы проект устройства HID, предлагаемый в комплекте поставки программной среды Keil uVision. Для переноса был выбран проект для платы Keil MCBSTM32, находящийся в папке C:\каталог установки Keil\ARM\Boards\Keil\MCBSTM32\USBHID. Схема платы STM32 Mini поставляется в комплекте с самой платой; схема MCBSTM32 доступна в Интернете, например, в [1]. При сравнении плат были найдены следующие различия:

- в плате MCBSTM32 используется МК типа STM32F103RB; в плате STM32 Mini используется, как уже говорилось выше, STM32F103VET;
- транзистор, управляющий подтяжкой линии USB D+ к плюсу питания (чтобы сообщить Windows о подключении нового устройства), подключён к разным выводам портов микроконтроллера у разных плат, – к PC13 на STM32 Mini и к PD2 на MCBSTM32;
- в составе STM32 Mini имеется только одна кнопка и один светодиод, подключённые к портам PB15 и PB5, соответственно;
- в составе STM32 Mini поставляется графический ЖКИ, более сложный в управлении, чем символьный дисплей платы MCBSTM32, поэтому было решено отказаться от использования дисплея в проекте.

С учётом этих особенностей перенос проекта на новую плату осуществлялся поэтапно:

1. Замена микроконтроллера в свойствах проекта;
2. Изменения в процедурах инициализации USB в соответствии с подключением транзистора;
3. Удаление вызова функций, обслуживающих ЖКИ;
4. Изменение процедуры инициализации портов управления светодиодами и обработки кнопок, а также управляющих функций Call-back.

Прежде чем приступить к изменению проекта, ознакомимся с его структурой. В таблице 1 приведено описание функций каждого файла из состава проекта устройства USB HID от Keil. Дерево проекта показано на рисунке 2.

Заменим МК в настройках проекта. Для этого переходим в настройки проекта, например, Project → Options for Target... и открываем вкладку *Device*, где заменяем МК STM32F103RB на

STM32F103VE. Выбираем OK и закрываем диалоговое окно.

Переходим к файлу *usbhw_STM32F10X.c*. Нас интересуют функции *USB_Init ( )* и *USB_Connect( )*, отвечающие, соответственно, за начальную настройку USB-интерфейса и сообщение компьютеру об обнаружении нового устройства. В данных функциях требуется изменить команды, отвечающие за настройку порта и управление транзистором, отвечающим за подтяжку линии D+ к плюсу питания. Поскольку на плате STM32 Mini транзистор подключен к выводу PC13 вместо PD2, функции следует изменить в соответствии с листингом 1.

Теперь перейдём к основному циклу программы, т.е. к файлу *demo.c*. Здесь необходимо убрать функции, отвечающие за инициализацию ЖКИ и вывод на него информации. Фактически, необходимо закомментировать подключение заголовочного файла *LCD.h* и удалить следующие строки:

```
lcd_init();
lcd_clear();
lcd_print (" MCBSTM32 HID ");
set_cursor (0, 1);
lcd_print (" www.keil.com ");
```

После чего следует изменить строки инициализации портов ввода-вывода, к которым подключены светодиоды и кнопки. В нашем случае необходимо включить тактирование порта B и настроить PB5 на вывод, а PB15 – на ввод, обеспечив подтяжку к плюсу питания. После всех изменений главный цикл программы будет выглядеть следующим образом (см. листинг 2).

Теперь осталось изменить функции Call-back, отвечающие за считывание кнопок и управление светодиодами. В нашем случае это *GetInReport( )* и *SetOutReport( ),* соответственно. В функции считывания ставим опрос кнопки, подключённой к порту PB15. Как и в оригинальном примере, защита от дребезга контактов не предусмотрена. Функция *GetInReport( )* будет выглядеть следующим образом:

```
void GetInReport (void)
{
   InReport = 0x00;
   if ((GPIOB->IDR & 0x8000) == 0)
   InReport |= 0x01;
}
```

Листинг 1

```
void USB_Init (void)
{
 RCC->APB1ENR |= (1 << 23);  /* enable clock for USB */

 USB_IntrEna ();             /* Enable USB Interrupts */

 /* Control USB connecting via SW */
 RCC->APB2ENR |= (1 << 4);   /* enable clock for GPIOC */
 GPIOC->BSRR = 0x2000;       /* set PC13 */
 GPIOC->CRH &= ~0x00F00000; /* clear port PC13 */
 GPIOC->CRH |= 0x00700000;   /* PC13 General purpose output
 open-drain, max speed 50 MHz */
}

void USB_Connect (BOOL con) {

 if (con) {
 GPIOC->BRR = 0x2000; /* reset PC13 */
 CNTR = CNTR_FRES; /* Force USB Reset */
 CNTR = 0;
 ISTR = 0;            /* Clear Interrupt Status */
 CNTR = CNTR_RESETM | CNTR_SUSPM | CNTR_WKUPM;
/* USB Interrupt Mask */
 } else {
 CNTR = CNTR_FRES | CNTR_PDWN; /* Switch Off USB Device */
 GPIOC->BSRR = 0x2000; /* set PC13 */
 }
}
```

Листинг 2

```
int main (void)
{
 RCC->APB2ENR |= (1UL << 3);
 GPIOB->CRL   &= ~0xFFFFFFFF;
 GPIOB->CRL   |= 0x33333333;
 GPIOB->CRL   &= ~0xFFFFFFFF;
 GPIOB->CRL   |= 0x83333333;
 GPIOB->ODR   |=       0x8000;

 USB_Init(); /* USB Initialization */
 USB_Connect(__TRUE); /* USB Connect */

 while (1); /* Loop forever */
}
```

Таблица 1. Структура проекта устройства USB HID от Keil

Файл	Описание
startup_stm32f10x_md.s	Файл начальной инициализации МК
system_stm32f10x.c	Файл с функциями настройки МК, в частности, системы тактирования
usbcfg_STM32F10X.h	Файл настройки параметров интерфейса и профилей USB
usbhw_STM32F10X.c	Функции низкого уровня для организации подключения и обмена по шине USB
usbuser.c	Файл, содержащий функции «конечных точек» USB
usbdesc.c	Дескриптор USB-устройства
usbcore.c	Модуль ядра USB
hiduser.c	Файл, содержащий функции, необходимые для работы профиля HID
LCD_4bit.c	Функции для работы с дисплеем платы
demo.c	Файл, содержащий главный цикл программы и функции Call-back

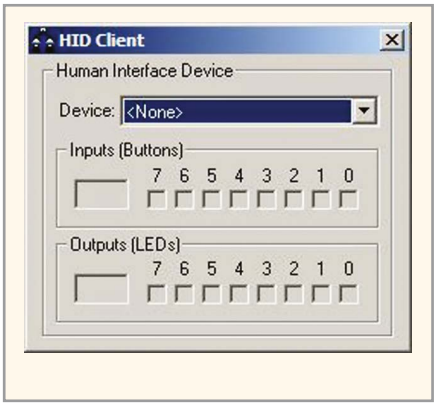

Рис. 3. Окно программы для тестирования устройства HID

Листинг 3

```
#if USB_CONFIGURE_EVENT
void USB_Configure_Event (void)
{
       U8 t=0;

 if (USB_Configuration) { /* Check if USB is configured */
       for (t=0; t<8; t++)
              InReport[t]=0;
       USB_WriteEP(0x81, InReport, sizeof(InReport));
 }
}
#endif
```

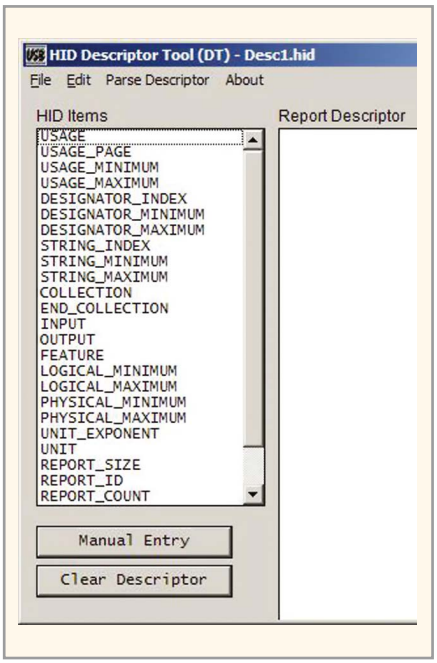

Рис. 4. Окно программы HID Descriptor Tool

В функции *SetOutReport* реализовано зажигание светодиода на порту PB5 младшим битом кадра *OutReport*:

```
void SetOutReport (void)
{
    if (OutReport & 0x01)
       GPIOB->ODR   |= 0x20;
    else
       GPIOB->ODR   &= ~0x20;
}
```

После компиляции программы и загрузки её в плату через отладчик необходимо извлечь кабель USB и вставить его снова. Система Windows должна указать на появление нового устройства и установить необходимые драйверы, после чего устройство появится в диспетчере задач. Проверить работоспособность можно с помощью тестовой утилиты, предлагаемой в пакете от Keil. Она находится в папке C:\каталог установки Keil\ARM\Utilies\HID_Client\Release, рабочее окно показано на рисунке 3.

До начала работы с платой её необходимо отметить (выбрать) в выпадающем списке. После чего можно управлять светодиодом, устанавливая или снимая галочку с младшего разряда, и регистрировать нажатие кнопки.

Эмуляция клавиатуры USB HID

Гораздо более интересным применением профиля HID, по мнению автора, является реализация USB-клавиатуры. Компания Keil не предлагает пример такого проекта, хотя эмуляция мыши присутствует для отладочных плат и достаточно легко адаптируется для других плат. Однако нас интересовала именно клавиатура. Достаточно подробно этот вопрос был рассмотрен в обсуждении [2].

Прежде всего, необходимо заставить компьютер обнаружить плату в качестве клавиатуры. Для этого следует изменить дескриптор устройства – файл *usbdesc.c*, а именно, исправить содержимое структуры *HID_ReportDescriptor[ ]*, а также устройства VID и PID в структуре *HID_DeviceDescriptor[ ]*.

Для заполнения первой структуры воспользуемся программой HID Descriptor Tool [3], окно которой показано на рисунке 4. В составе программы есть образцы дескрипторов нескольких USB-устройств. Файл *Keyboard.hid* содержит дескриптор клавиатуры. Этот файл необходимо открыть с помощью программы, и содержимое, которое появится в поле *Report Descriptor*, следует поместить в структуру *HID_ReportDescriptor[ ]*.

В качестве устройства VID и PID были использованы идентификаторы примера USB-клавиатуры от Texas Instruments – VID 0x2047 и PID 0x0401. После приведённых выше изменений проект можно скомпилировать и загрузить в плату. Выключение и включение разъёма USB заставит Windows обнаружить новое устройство – клавиатуру. В этом можно убедиться, открыв диспетчер устройств, среди которых будет дополнительная USB-клавиатура.

Далее необходимо внести изменения в основной текст программы. Дескриптор предполагает, что сообщение, передаваемое компьютеру, – *InReport* – является массивом 8 байт, имеющим структуру, показанную в таблице 2. Исходя из этого, в файле *demo.c* исправим объявление переменной *InReport* на объявление восьмибайтового массива того же типа. Также необходимо исправить объявление этой переменной и в файле *demo.h*. В таблице 3 приведена структура нулевого байта InReport – поля информации о нажатых управляющих клавишах.

После этого откроем файл *usbuser.c*. В функцию обработки запроса конфигурации *USB_Configure_Event( )* необходимо добавить обнуление массива *InReport*. В результате изменений функция выглядит следующим образом – листинг 3.

Также необходимо внести изменения в функцию обработки первой конечной точки, так как в предыдущем варианте программы передавался только один байт. После изменения данная функция будет выглядеть следующим образом:

```
void USB_EndPoint1 (U32 event)
{
 switch (event) {
 case USB_EVT_IN:
 GetInReport();
 USB_WriteEP(0x81, InReport,
 sizeof(InReport));
 break;
 }
}
```

Теперь перейдем к файлу *hiduser.c*. Здесь необходимо изменить функцию *HID_GetReport( )* – сделать так, чтобы в случае прихода запроса на чтение в буфер *EP0Buf* копировалось содержимое всего массива *InReport* (см. листинг 4).

Теперь осталось изменить функцию обработки кнопок в файле *demo.c*. В нашем случае нажатие на кнопку платы будет соответствовать нажатию клавиши с цифрой 1 на клавиатуре. Таблицу скан-кодов клавиатуры USB HID можно найти в [4].

```
void GetInReport (void)
{
   InReport[2] = 0x00;
   if ((GPIOB->IDR & 0x8000) == 0)
   InReport[2]=30;
}
```

Скомпилируем проект и снова загрузим его в плату. После переподключения платы к USB она сможет эмулировать нажатие клавиши «1» и печатать эту цифру в любом текстовом редакторе.

Переменная *OutReport* в данном примере отвечает за состояние светодиодов клавиатуры. Она не нуждается в каких-либо изменениях. Структура битового поля данной переменной приведена в таблице 4, в соответствии с которой были внесены изменения в функцию Call-back управления светодиодами:

```
void SetOutReport (void)
{
if (OutReport & 0x02)
        GPIOB->ODR  |=  0x20;
    else
        GPIOB->ODR  &=  ~0x20;
 }
```

После таких изменений зажигание светодиода платы происходит по нажатию клавиши Caps Lock. Проверить работоспособность функции достаточно просто, поскольку ОС Windows рассматривает все подключённые к компьютеру клавиатуры как единое целое. Поэтому нажатие клавиши Caps Lock на одной клавиатуре приведёт к зажиганию соответствующего светодиода на всех подключённых клавиатурах.

Листинг 4

```
BOOL HID_GetReport (void)
{
 unsigned char t=0;

 /* ReportID = SetupPacket.wValue.WB.L; */
 switch (SetupPacket.wValue.WB.H) {
 case HID_REPORT_INPUT:
 GetInReport();
 for (t=0; t<sizeof(InReport); t++)
       EP0Buf[t] = InReport[t];
 break;
 case HID_REPORT_OUTPUT:
 return (__FALSE); /* Not Supported */
 case HID_REPORT_FEATURE:
 /* EP0Buf[] = ...; */
 /* break; */
 return (__FALSE); /* Not Supported */
 }
 return (__TRUE);
}
```

Таблица 2. Структура сообщения, передаваемого компьютеру от клавиатуры

Байт	Назначение
0	Поле, содержащее информацию о нажатии управляющих клавиш клавиатуры
1	Зарезервировано
2	Скан-код первой нажатой клавиши
3	Скан-код второй нажатой клавиши
4	Скан-код третьей нажатой клавиши
5	Скан-код четвёртой нажатой клавиши
6	Скан-код пятой нажатой клавиши
7	Скан-код шестой нажатой клавиши

Таблица 3. Структура поля, содержащего информацию о нажатых управляющих клавишах

Бит	Назначение
0	Левая клавиша Ctrl
1	Левая клавиша Shift
2	Левая клавиша Alt
3	Левая клавиша Windows
4	Правая клавиша Ctrl
5	Правая клавиша Shift
6	Правая клавиша Alt
7	Правая клавиша Windows

Таблица 4. Структура битового поля переменной OutReport

Бит	Назначение
0	Состояние светодиода NumLock, 0 – выключен, 1 – включён
1	Состояние светодиода CapsLock, 0 – выключен, 1 – включён
2	Состояние светодиода ScrollLock, 0 – выключен, 1- включён
3..7	Зарезервировано

Литература

1. MCBSTM32 V1.1. Schematics. www.keil.com/mcbstm32/mcbstm32-schematics.pdf.
2. LPC2368 HID KeyBoard. www.keil.com/forum/13867/.
3. HID Descriptor Tool. www.usb.org/developers/ hidpage/dt2_4.zip.
4. USB HID Keyboard Scan Codes. www.mindrunway.ru/IgorPlHex/USBKeyScan.pdf.