Продолжение. Начало в № 8'2014

Графический контроллер EVE FT800 FTDI и микроконтроллер SAMD21 Atmel. Работаем с графическими изображениями

Сергей ДОЛГУШИН
dsa@efo.ru

В данной статье мы продолжим рассказ о возможностях графического контроллера FT800 FTDI и микроконтроллера SAMD21 Atmel [1–5]. Рассмотрим работу с USB-хостом SAMD21 на примере чтения JPEG-изображений с USB флэш-диска и вывода их на экран TFT-дисплея с помощью контроллера FT800.

В предыдущей статье [4] мы подробно рассказали о базовых принципах работы FT800 с графическими изображениями. Работа с изображениями в формате JPEG практически ничем не отличается от изложенного в первой части. Графический контроллер FT800 аппаратно поддерживает декодирование JPEG-изображений, а в задачу управляющего микроконтроллера (МК) входят только запись изображения в графическую память FT800 и передача команды на отображение. Данный формат изображений очень удобен, особенно если предполагается использование внешних накопителей, например карт памяти или USB-дисков. Компания FTDI предлагает ряд демонстрационных приложений, показывающих работу с картами памяти SD. Эти примеры созданы для нового модуля VM800P, управление которым осуществляется с помощью МК Atmega 328P (рис. 1). В свою очередь, мы рассмотрим пример, где носителем информации будет служить USB-диск.

Еще пару лет назад выбор простых МК со встроенным USB-хостом был ограничен. Самыми доступными были решения от FTDI и Microchip. Эти производители одними из первых предложили комплексное решение для встраиваемых приложений, предусматривающее аппаратную реализацию USB-хоста и набор готовых драйверов наиболее востребованных классов USB-устройств. Сегодня выбор подобных решений стал больше. И одно из них — новый МК SAMD21 компании Atmel [5], который по своим функциональным возможностям и цене уверенно опережает Vinculum II и может быть рекомендован в качестве альтернативы FTDI. Для стандартных классов USB-устройств MSC, HID и CDC предпочтительнее МК SAMD21, нежели микросхема FTDI. Именно этот МК и будет фигурировать в нашем примере.

Рис. 1. Графический модуль VM800P

Основная задача демонстрации — чтение изображения с USB-диска и вывод его на экран TFT-дисплея. В примере использованы отладочная плата XPlained Pro ATSAMD21-XPRO и графический модуль VM800B43A. Подключение графического модуля к отладочной плате было описано в [4].

Начнем с краткого знакомства с USB-интерфейсом МК SAMD21. Он может быть сконфигурирован для работы в качестве периферийного устройства или хоста. Этот блок полностью удовлетворяет спецификации USB 2.1 и поддерживает передачу данных в режимах low-speed и full-speed. Для работы блока в периферийном режиме никаких дополнительных внешних компонентов не требуется, для работы в режиме хоста необходим внешний кварцевый резонатор.

Для поддержки USB на программном уровне в ASF (Atmel Software Framework) включен драйвер USB-хоста, состоящий из трех блоков (рис. 2):

- UHD (USB Host Driver) — драйвер низкого уровня, который реализует физический протокол;
- UHI (USB Host Interface) обеспечивает работу с классами USB-устройств. В UHI реализована поддержка трех стандартных классов USB-устройств: MSC, CDC и HID. Кроме стандартных классов, в Atmel Software Framework входит шаблон для создания собственных драйверов (Vendor Specific). Класс устройства, который будет поддерживать USB-хост в приложении, выбирается при добавлении драйвера в проект. Также возможна одновременная поддержка нескольких классов;

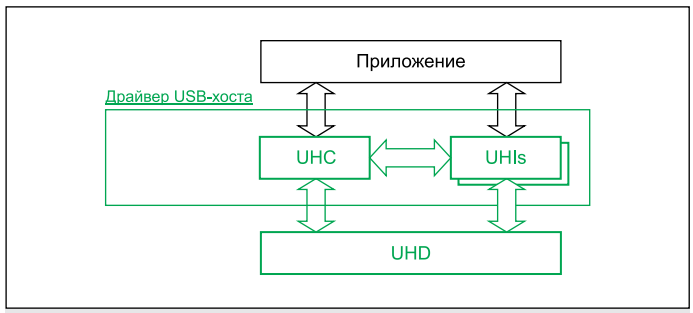

Рис. 2. Структура драйвера USB ASF

- UHC (USB Host Controller) представляет базовый интерфейс для взаимодействия пользовательского приложения и USB-устройства. В него включены стандартные для всех USB-устройств команды.

В качестве файловой системы Atmel предлагает использовать свободно распространяемую библиотеку FatFS, которая также включена в ASF. Полное описание и примеры работы с данной файловой системой находятся по ссылке [6]. Из важных достоинств FatFS отметим совместимость с файловыми системами Windows и поддержку длинных имен файлов. Это важные факторы, определяющие удобство работы для конечного пользователя. То есть любые файлы, созданные на ПК с ОС Windows и записанные на внешний носитель, могут быть прочитаны устройством. А пользователю нет нужды помнить об ограничениях на длину имени.

Самый простой путь для первого знакомства с возможностями USB-интерфейса микросхемы SAMD21 — использование готовых примеров, включенных в ASF (рис. 3). В нашем случае это USB Host MSC FatFS, демонстрирующий работу с USB-диском, создание нового файла и запись в него текстовой строки.

Пользовательская часть программы очень проста:

```
for (lun = LUN_ID_USB; (lun < LUN_ID_USB +uhi_msc_mem_get_lun()) &&(lun < MAX_DRIVE); lun++)
{
// Check if LUN has been already tested
      if (TEST_OK == lun_states[lun] || TEST_ERROR == lun_states[lun])
      {
      continue;
      }
      // Монтируем диск
      memset(&fs, 0, sizeof(FATFS));
      res = f_mount(lun, &fs);
      if (FR_INVALID_DRIVE == res)
      {
      // Диск отсутствует
      lun_states[lun] = TEST_NO_PRESENT;
      continue;
      }
      // Создаем новый файл с именем test_file_name, если файл существует, он заменяется новым
      test_file_name[0] = lun + '0';
      res = f_open(&file_object, (char const *)test_file_name, FA_CREATE_ALWAYS | FA_WRITE);
      if (res == FR_NOT_READY)
      {
      // диск был извлечен или не выполнена инициализация диска
lun_states[lun] = TEST_NO_PRESENT;
      f_close(&file_object);
      continue;
      }
      if (res != FR_OK)
      {
      // ошибка записи
      lun_states[lun] = TEST_ERROR;
      f_close(&file_object);
      continue;
      }
      // Запись текстовой строки MSG_TEST в файл
      f_puts(MSG_TEST, &file_object);
      lun_states[lun] = TEST_OK;
      f_close(&file_object);
}
```

Операции по детектированию подключения USB-устройства и его инициализации производятся драйвером автоматически. В общем случае программисту нет необходимости вмешиваться в данный процесс. Информацию об этих процессах можно получать посредством функций UHC-драйвера. В примере USB Host MSC FatFS

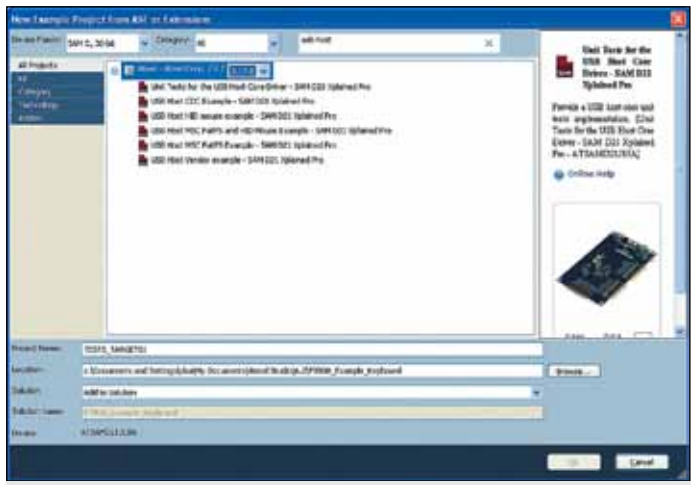

Рис. 3. Примеры работы с драйверами USB-классов

реализован пользовательский интерфейс **ui.c**, который предназначен для информирования пользователя о процессах, происходящих на USB-шине. В частности, результат процесса определения наличия USB-устройства реализован следующим образом:

```
void ui_usb_connection_event(uhc_device_t *dev, bool b_present)
{
   UNUSED(dev);
   if (b_present) {

      Text_Screen("USB Connected"); // вывод текстовой строки на экран дисплея при подключении
USB-устройства
      delay_ms(500);
      Clear_Screen(); // очистка экрана

   }
   if (!b_present) {
      Text_Screen("USB Disconnected"); // вывод текстовой строки на экран дисплея при отключении
USB-устройства
      delay_ms(500);
      Clear_Screen(); // очистка экрана
      ui_enum_status = UHC_ENUM_DISCONNECT;
   }
}
```

Данная функция вызывается после срабатывания прерывания при изменении состояния линий DP или DM, что говорит драйверу UHD о подключении/отключении USB-устройства. Мы добавили в эту функцию вывод информационного сообщения на экран TFT-дисплея и очистку экрана, предварительно включив в проект необходимые для работы с FT800 драйверы и библиотеки [4]:

```
ft_void_t Text_Screen (const ft_char8_t* label)
{

      Ft_Gpu_CoCmd_Dlstart(phost);
      Ft_App_WrCoCmd_Buffer(phost,CLEAR_COLOR_RGB(64,64,64));
      Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
      Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
      Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,FT_DispHeight/2,25,OPT_CENTERX|OPT_CENTERY,label);
      Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
      Ft_Gpu_CoCmd_Swap(phost);
      Ft_App_Flush_Co_Buffer(phost);
      Ft_Gpu_Hal_WaitCmdfifo_empty(phost);

}

ft_void_t Clear_Screen (void)
{

      Ft_Gpu_CoCmd_Dlstart(phost);
      Ft_App_WrCoCmd_Buffer(phost,CLEAR_COLOR_RGB(64,64,64));
      Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
      //Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
      //Ft_Gpu_CoCmd_Text(phost,FT_DispWidth/2,FT_DispHeight/2,25,OPT_CENTERX|OPT_CENTERY,label);
      Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
      Ft_Gpu_CoCmd_Swap(phost);
      Ft_App_Flush_Co_Buffer(phost);
      Ft_Gpu_Hal_WaitCmdfifo_empty(phost);

}
```

Пример USB Host MSC FatFS знакомит нас с процессом создания файла и записи в него данных. Кроме того, в нем уже сделаны все необходимые настройки системы тактирования МК и реализованы полезные функции, информирующие пользователя о статусе USB-интерфейса.

Теперь вернемся к нашей задаче отображения на экране JPEG-изображений, хранящихся на USB-диске. В общем случае сначала нужно получить список файлов, хранящихся на диске. Чтение содержимого диска или его раздела осуществляется командой **f_readdir**(). Эта функция должна вызываться после подключения внешнего носителя функцией **f_mount**():

```
// чтение директории
res=f_opendir(&mydir, path);
if (res == FR_OK)
{

Ft_Gpu_CoCmd_Dlstart(phost);
   Ft_App_WrCoCmd_Buffer(phost,CLEAR_COLOR_RGB(64,64,64));
   Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1));
   Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(255,255,255));
   str_pos=0;
   for (;;)
      {
      res = f_readdir(&mydir, &names);  // чтение содержимого директории
      if (res != FR_OK || names.fname[0] == 0) break; // окончание по достижении конца или ошибке
      if (names.fname[0] == '.') continue;
      #if _USE_LFN
         fn = *names.lfname ? names.lfname : names.fname;
       #else
         fn = names.fname;
      #endif
   Ft_Gpu_CoCmd_Text(phost,100,10+str_pos,21,OPT_CENTER,str);
      str_pos = str_pos+15;
      }
   Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
   Ft_Gpu_CoCmd_Swap(phost);
   Ft_App_Flush_Co_Buffer(phost);
   Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
   delay_ms(500);
   }
```

В результате выполнения этого кода на экран дисплея будет выведен список файлов и папок, хранящихся в корневой директории диска. Чтобы вывести изображения с диска на экран, трансформируем данный код следующим образом:

```
// вывод изображений на экран
res=f_opendir(&mydir, path);
if (res == FR_OK)
{
   str_pos=0;
   for (;;)
   {
      res = f_readdir(&mydir, &names);
      if (res != FR_OK || names.fname[0] == 0) break;
      if (names.fname[0] == '.') continue;
      #if _USE_LFN
      fn = *names.lfname ? names.lfname : names.fname;
      #else
      fn = names.fname;
      #endif
      if (names.fname[9] == 'j') // упрощенный, для примера, вариант фильтрации файлов с расширением jpg
      {

         Ft_Gpu_Hal_WrCmd32(phost, CMD_LOADIMAGE); // команда загрузки JPEG
         Ft_Gpu_Hal_WrCmd32(phost, 0); // адрес в графической памяти, куда записывается изображение
         Ft_Gpu_Hal_WrCmd32(phost, 0);
            // формат загружаемого изображения: 0 – RGB565, 1 – L8 (черно-белое)

         res = f_open(&file_object, fn, FA_OPEN_EXISTING | FA_READ);
         FileLen = names.fsize;
         while(FileLen > 0)
            {
               ft_uint16_t blocklen = FileLen>4096?4096:FileLen;
               res = f_read(&file_object, bufferf, blocklen, &br); // чтение из файла
               FileLen -= blocklen;
               Ft_Gpu_Hal_WrCmdBufFromFlash(phost,bufferf, blocklen);
                  // запись в графическое ОЗУ содержимого файла
               delay_ms(10);
            }

         Ft_App_WrCoCmd_Buffer(phost, CMD_DLSTART);
         Ft_App_WrCoCmd_Buffer(phost, CLEAR_COLOR_RGB(0,0,0));
         Ft_App_WrCoCmd_Buffer(phost, CLEAR(1,1,1));
         Ft_App_WrCoCmd_Buffer(phost, COLOR_RGB(255,255,255));
         Ft_App_WrCoCmd_Buffer(phost, BEGIN(BITMAPS));
            // начало работы с графическими примитивами
         Ft_App_WrCoCmd_Buffer(phost, BITMAP_SOURCE(0));
            // адрес в графической памяти, где находится изображение
         Ft_App_WrCoCmd_Buffer(phost, BITMAP_LAYOUT(RGB565,ImgW*2,ImgH));
            // указываются параметры изображения
         Ft_App_WrCoCmd_Buffer(phost, BITMAP_SIZE(BILINEAR,BORDER,BORDER,ImgW,ImgH));
         Ft_App_WrCoCmd_Buffer(phost, VERTEX2F(xoffset*16,yoffset*16));
         Ft_App_WrCoCmd_Buffer(phost, END());

         Ft_App_WrCoCmd_Buffer(phost,DISPLAY());
         Ft_Gpu_CoCmd_Swap(phost);
         Ft_App_Flush_Co_Buffer(phost);
         Ft_Gpu_Hal_WaitCmdfifo_empty(phost);
         delay_ms(500);
         }
      }
   }
```

В процессе чтения раздела диска МК, найдя файл с расширением *.jpg*, выведет его на экран. В итоге все JPEG-изображения будут поочередно выведены с диска на дисплей.

При желании данный пример можно объединить с примером FTDI [9], показывающим реализацию приложения для просмотра изображений. На наш взгляд, один из интересных моментов в этом примере — анимация смены одной картинки другой.

Итак, мы показали базовые методы работы с графическим контроллером FT800, реализующие вывод растровых изображений на экран TFT-дисплея. Использование сжатых изображений, в том числе в популярном формате JPEG, положительно сказывается на снижении нагрузки на управляющий МК, что связано с уменьшением объема передаваемых данных в FT800. Большой объем встроенной графической памяти FT800 также этому способствует, позволяя загрузить за один раз несколько изображений. В дальнейшем, в ходе выполнения приложения, вызов требуемой картинки осуществляется одной командой. На текущий момент ни один из распространенных графических контроллеров не может самостоятельно работать со сжатыми изображениями. Эту функцию, если требуется поддержка, например формата JPEG, должен выполнять управляющий МК.

Также отметим удачную реализацию USB-интерфейса в новом МК Atmel SAMD21. Поддержка наиболее востребованных классов USB-устройств и интуитивно понятные примеры делают задачу реализации USB-хоста максимально простой. В сравнении с ранее популярным микроконтроллером Vinculum II FTDI процесс освоения SAMD21 гораздо проще. На текущий момент единственными преимуществами Vinculum II остаются два порта USB и поддержка USB-устройств на базе микросхем FTDI. Для тех, кто ориентируется на Vinculum II в качестве USB-хоста с поддержкой MSC- или CDC-классов, мы рекомендуем обратить внимание на SAMD21.

Литература

1. Долгушин С. Графический контроллер EVE FT800 компании FTDI // Компоненты и технологии. 2013. № 11.
2. Долгушин С. Начинаем работать с графическим контроллером FT800 FTDI // Компоненты и технологии. 2014. № 5.
3. Долгушин С. Графический контроллер EVE FT800 FTDI. Работа с пользовательскими шрифтами, кнопками и сенсорным экраном // Компоненты и технологии. 2014. № 6.
4. Долгушин С. Графический контроллер EVE FT800 FTDI и микроконтроллер SAMD21 Atmel. Работаем с графическими изображениями // Компоненты и технологии. 2014. № 8.
5. Сазанов Д. SAMD — новая линейка микроконтроллеров с ядром ARM Cortex-M0+ компании Atmel // Компоненты и технологии. 2014. № 5.
6. http://elm-chan.org/fsw/ff/00index_e.html
7. AT06475: SAMD21 USB.
8. Atmel AVR4950: ASF — USB Host Stack.
9. http://www.ftdichip.com/Support/SoftwareExamples/EVE/FT_App_Imageviewer.zip