Начинаем работать с графическим контроллером FT800 FTDI

Сергей ДОЛГУШИН
dsa@efo.ru

В начале 2013 года компания FTDI анонсировала новую микросхему — графический контроллер FT800 EVE, который мы представили читателям в [1]. FT800 включает в себя три основных блока: графический контроллер, контроллер резистивного сенсорного экрана и аудиоконтроллер. Набор команд для управления всеми возможностями новой микросхемы достаточно велик, а порядок обмена между управляющим МК и FT800 имеет специфические особенности.
На наш взгляд, самым простым методом освоения графического контроллера будет использование готовых примеров производителя. Примеры, которые предлагаются для ознакомления с возможностями FT800, хорошо структурированы и содержат готовый набор API-функций для всех команд. В статье освещены основные моменты по переносу примеров FTDI на 8-разрядный МК компании Cypress семейства PSoC.

Введение

Новый графический контроллер EVE (Embedded Video Engine) FT800 за год своего существования уже завоевал две престижных награды как лучший продукт года — Elektra 2013 и BEEA 2013. Такое признание он заслужил благодаря своим уникальным особенностям. Набор графических функций, встроенных в микросхему FT800, позволяет использовать TFT-дисплей даже с 8-разрядными микроконтроллерами. Графические функции, помимо стандартных для вывода точек, линий, заливки экрана и т. п., позволяют на аппаратном уровне работать со сложными объектами — кнопками, слайдерами, текстом. Эти графические элементы могут отображаться на экране с различными эффектами, в плоском и объемном виде, с монохромной и градиентной заливкой. Помимо статичных объектов, FT800 может отображать на экране заставку в виде движущейся картинки или визуализировать режим ожидания анимацией. Эти функции FT800 отрабатывает самостоятельно, без участия управляющего микроконтроллера.

Кроме графических функций, микросхема включает в себя контроллер резистивного сенсорного экрана и аудиоконтроллер со встроенной библиотекой готовых звуковых эффектов. Совокупность этих возможностей, а также ее невысокая стоимость делают микросхему FT800 перспективной для новых разработок и менее требовательной к ресурсам управляющего контроллера, чем конкурентные решения. Отметим, что ни одно из существующих решений аналогичного функционального назначения не имеет сегодня встроенных контроллера сенсорного экрана и аудиоконтроллера.

Помимо самих микросхем, компания FTDI предлагает набор готовых графических модулей на базе TFT-дисплеев с размером по диагонали 3,5″ (320×240), 4,3″ и 5″ (480×272). Модули включают в себя все необходимые для работы компоненты, установленные аудиоусилитель и динамик. Модули серии VM800Bxxx поставляются с декоративной рамкой и монтажной панелью для установки в корпусе прибора (рис. 1).

Рис. 1. Графический модуль VM800Bxxx на базе FT800

Аппаратные средства

Вернемся к теме статьи и возможностям FT800. Микросхема поддерживает широкий набор графических функций, работу с сенсорным экраном, воспроизведение звука и ряд команд, управляющих режимами работы. Освоение «с нуля» всего функционального набора может оказаться трудоемкой задачей. Помочь в этом могут примеры производителя, которые демонстрируют работу всех функций микросхемы FT800. Эти примеры хорошо структурированы и могут использоваться в качестве библиотеки функций для применения в собственных проектах и/или понимания принципов управления графическим контроллером.

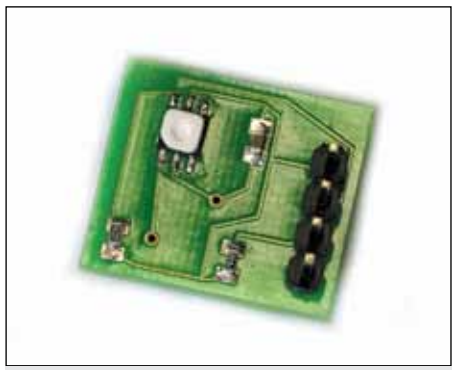

Рис. 2. Датчик Si7021, установленный на плату

Далее мы покажем, как адаптировать основной пример FTDI для использования с микроконтроллером Cypress PSoC CY8C29466 и компилятором ImageCraft. Повторим проект, показанный ранее в примере работы с графическим модулем uLCD-43PT компании 4D Systems [2]. Макет будет включать в себя следующие элементы: отладочную плату CY3210-PSOCEVAl1 с установленным микроконтроллером CY8C29466, модуль собственного изготовления с датчиком температуры и влажности Si7021 (рис. 2) и графический модуль FTDI VM800B43A-BK (рис. 3).

Основные параметры используемых компонентов следующие. Микросхема CY8C29466 — это простой 8-разрядный микроконтроллер на базе ядра Cypress M8C с производительностью до 4 MIPS, тактовой частотой ядра до 24 МГц, объемом флэш-памяти 32 кбайт и ОЗУ 2 кбайт. Используемая периферия: аппаратный интерфейс SPI для связи с графическим модулем на базе FT800 (тактовая частота 12 МГц, средняя скорость передачи 4 Мбит/с) и программная реализация интерфейса I^2C для управления датчиком Si7021.

Модуль с датчиком Si7021 — это плата, на которой установлены сам датчик, подтягивающие резисторы на линиях SDA и SCL и конденсатор в цепи питания. Si7021 является продолжением предыдущего удачного решения Silicon Labs — датчика температуры и влажности Si7005. Новый датчик обладает лучшими характеристиками, чем его предшественник. На текущий момент датчики температуры и влажности Silicon Labs являются самым дешевым решением на рынке при сопоставимых с конкурентами характеристиках [2, 3].

Графический модуль VM800B43A-BK включает в себя TFT-дисплей с разрешением 480×240 точек, его размер по диагонали — 4,3″. Дисплей имеет резистивный сенсорный экран. TFT-дисплей и плата управления установлены в пластиковом держателе, который предназначен для крепления модуля внутри корпуса прибора. С лицевой стороны устанавливается пластиковая декоративная рамка, которая также входит в состав этого модуля.

Плата управления модуля реализована на базе микросхемы FT800, линии интерфейса SPI могут работать с уровнями 5 и 3 В. (Сама микросхема FT800 работает только с уровнем 3 В). Питание модуля может осуществляться от 5-В или 3-В источника. На плате установлены аудиоусилитель и динамик для воспроизведения звуковых эффектов.

Программная реализация

Для реализации своей библиотеки команд для управления FT800 возьмем пример, описанный в [4]. Это базовый пример, в котором FTDI показывает работу всех поддерживаемых графическим контроллером команд. В том виде, в котором этот пример (и все остальные) дается производителем, он может быть запущен из-под ОС Windows или на платформе Arduino. Мы же покажем, что именно необходимо изменить для переноса его на МК CY8C29466. В результате внешний вид работающего приложения будет выглядеть следующим образом (рис. 4). Измеренные датчиком Si7021 значения температуры и влажности будут выводиться на экран в текстовом и графическом виде.

Основными файлами примера AN 245 являются:

- *SampleApp.c* — это пользовательское приложение. Оно будет полезно для понимания порядка инициализации FT800 при включении и последовательности команд для передачи в FT800 и последующего вывода на экран.
- *FT_Gpu_Hal.c* — это платформо-зависимая часть, которая реализует основные функции обмена по SPI и формирует протокол управления FT800. Соответственно, изменения в этом файле коснутся функций приема и передачи команд и данных между управляющим МК и FT800. Все остальное, как и в случае с CY8C29466 и компилятором Image Craft, остается без изменений.
- *FT_CoPro_Cmds.c* включает в себя API-функции всех команд для работы с графическим контроллером. Описание самих команд дано в руководстве программиста [5]. В зависимости от используемого компилятора могут понадобиться небольшие изменения, например, при работе с компилятором ImageCraft для МК PSoC пришлось заменить все переменные ***const ft_char8_t *string***, вызываемые в некоторых функциях, на ***ft_char8_t *string***.

Итак, файл *FT_Gpu_Hal.c* содержит в себе все команды для управления микросхемой FT800 по интерфейсу SPI. Первые три функции, описанные в этом файле, — это функции инициализации, открытия и закрытия интерфейса SPI. Здесь и далее по тексту весь листинг будем приводить с внесенными изменениями, оригинал можно посмотреть в [4].

Функция Ft_Gpu_Hal_Init(void)

В оригинале в этой функции осуществляется инициализация библиотеки MPSSE для микросхемы FT232H, которая используется для управления графическим модулем с ПК. В нашем случае эта функция преобразилась в установку линии CS в неактивное состояние. Если управление линией CS реализовано в библиотеке SPI, то эту функцию можно исключить:

Рис. 3. Компоненты модуля VM800B43A-BK

Рис. 4. Информация на дисплее, которая выводится в результате работы примера

```
ft_bool_t Ft_Gpu_Hal_Init(void)
{
   HAL_SPI_CSHigh(); // void HAL_SPI_CSHigh(void) { PRT0DR |= 0x10;};

   return TRUE;
}
```

*Функция Ft_Gpu_Hal_Open (Ft_Gpu_Hal_Context_t *host)*

В этой функции оставлены запуск блока SPI с требуемыми параметрами и инициализация переменных, являющихся указателями на текущую область памяти FT800 (дисплей-лист или FIFO), в которую будет записываться следующая графическая команда. Также устанавливается статус, что интерфейс готов к работе: FT_GPU_HAL_OPENED. Все указанные переменные хранятся в служебной структуре Ft_Gpu_Hal_Context_t (она определена в файле *FT_Gpu_Hal.h*), обращение к ним осуществляется через указатель host. В исходном варианте в этой функции также осуществляется настройка частоты интерфейса, при использовании PSoC первичный выбор рабочей частоты блока SPI делается на графическом уровне и автоматически добавляется в библиотеку. Переключения рабочей частоты блока SPI в нашем случае не требуется, так как рабочая частота выбранного МК ниже частоты FT800 в режимах инициализации и рабочем. Поэтому никаких дополнительных команд по управлению частотой блока SPI CY8C29466 в коде не требуется. Если частота SPI выбранного микроконтроллера может достигать 30 МГц, то на момент инициализации она не должна превышать 12 МГц, после инициализации ее можно поднять до максимума:

```
ft_bool_t Ft_Gpu_Hal_Open(Ft_Gpu_Hal_Context_t *host)
{
   SPIM_Start(SPIM_SPIM_MODE_0 | SPIM_SPIM_MSB_FIRST); // Включение SPI в PSoC и установка
его параметров — режим “Mode 0”, старший бит передается первым
   host->ft_cmd_fifo_wp = host->ft_dl_buff_wp = 0;
   host->status = FT_GPU_HAL_OPENED;
   return TRUE;
}
```

*Функция Ft_Gpu_Hal_Close(Ft_Gpu_Hal_Context_t *host)*

В этой функции происходит выключение блока SPI и установка соответствующего значения переменной status:

```
ft_void_t Ft_Gpu_Hal_Close(Ft_Gpu_Hal_Context_t *host)
{
   host->status = FT_GPU_HAL_CLOSED;
   SPIM_Stop();
}
```

*Функция Ft_Gpu_Hal_StartTransfer(Ft_Gpu_Hal_Context_t *host, FT_GPU_TRANSFERDIR_T rw, unsigned long addr)*

Функция ***Ft_Gpu_Hal_StartTransfer (Ft_Gpu_Hal_Context_t *host, FT_GPU_TRANSFERDIR_T rw, unsigned long addr)*** реализует команды чтения **Host Memory Read** и записи **Host Memory Write** [1, 5]. С помощью команды записи управляющий МК сообщает FT800, что далее будет передаваться набор команд на выполнение тех или иных графических операций, воспроизведение звука и т. п. Команда чтения позволяет управляющему МК получать информацию от FT800 (состояние регистров, результаты обработки сигналов сенсорного экрана и т. д.):

```
void Ft_Gpu_Hal_StartTransfer(Ft_Gpu_Hal_Context_t *host,FT_GPU_TRANSFERDIR_T rw,unsigned long addr)
{
   if (FT_GPU_READ == rw)
   {
      HAL_SPI_CSLow(); // установка CS в активное состояние
      while( ! (SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY ) ); // проверка готовности SPI
для передачи
      SPIM_SendTxData((addr & 0x00FF0000) >> 16); // отправка байта по SPI
      while( ! (SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY ) );
      SPIM_SendTxData(((addr & 0x0000FF00) >> 8);
      while( ! (SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY ) );
      SPIM_SendTxData(addr & 0x000000FF);
      while( ! (SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY ) );
      SPIM_SendTxData(0);
      host->status = FT_GPU_HAL_READING;
   }
else
   {
      HAL_SPI_CSLow();
      while( ! (SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY ) );
      SPIM_SendTxData(0x80 | (addr >> 16));
      while( ! (SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY ) );
      SPIM_SendTxData(addr >> 8);
      while( ! (SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY ) );
      SPIM_SendTxData(addr);

      host->status = FT_GPU_HAL_WRITING;
   }
}
```

*Функция Ft_Gpu_Hal_Transfer8(Ft_Gpu_Hal_Context_t *host, ft_uint8_t value)*

Функция чтения и записи байта ***Ft_Gpu_Hal_Transfer8(Ft_Gpu_Hal_Context_t *host, ft_uint8_t value)*** является базой для формирования всех управляющих команд (графических и аудио) и команд чтения регистров FT800 (например, получение информации о точке касания от контроллера сенсорного экрана или текущем указателе на адрес командного FIFO):

```
ft_uint8_t Ft_Gpu_Hal_Transfer8(Ft_Gpu_Hal_Context_t *host,ft_uint8_t value)
{
   BYTE MCU_Readbyte = 0;
while( ! (SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY ) ); // проверка готовности SPI для передачи
   SPIM_SendTxData(value); // отправка байта по SPI
   while( ! (SPIM_bReadStatus() & SPIM_SPIM_RX_BUFFER_FULL)); // проверка наличия данных в приемном
буфере SPI
   MCU_Readbyte = SPIM_bReadRxData(); / // чтение байта из буфера SPI
   return MCU_Readbyte;
}
```

*Функция Ft_Gpu_HostCommand(Ft_Gpu_Hal_Context_t *host, ft_uint8_t cmd)*

Последней функцией, в которой присутствует обмен по SPI, является ***Ft_Gpu_HostCommand(Ft_Gpu_Hal_Context_t *host, ft_uint8_t cmd)***. На основе этой функции формируются управляющие команды (**Host command**), предназначенные для выбора режима работы FT800 (активное состояние, режим готовности, спящий режим и выключение), выбора источника тактовой частоты, установки тактовой частоты и сброса контроллера FT800:

```
ft_void_t Ft_Gpu_HostCommand(Ft_Gpu_Hal_Context_t *host,ft_uint8_t cmd)
{
   HAL_SPI_CSLow();
   while( ! (SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY ) );
   SPIM_SendTxData(cmd);
   while( ! (SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY ) );
   SPIM_SendTxData(0);
   while( ! (SPIM_bReadStatus() & SPIM_SPIM_TX_BUFFER_EMPTY ) );
   SPIM_SendTxData(0);
   HAL_SPI_CSHigh();
}
```

*Функция Ft_Gpu_Hal_EndTransfer(Ft_Gpu_Hal_Context_t *host)*

Функция ***Ft_Gpu_Hal_EndTransfer(Ft_Gpu_Hal_Context_t *host)*** завершает обмен по интерфейсу SPI между управляющим контроллером и FT800 переводом линии CS в неактивное состояние:

```
ft_void_t Ft_Gpu_Hal_EndTransfer(Ft_Gpu_Hal_Context_t *host)
{
   HAL_SPI_CSHigh();
   host->status = FT_GPU_HAL_OPENED;
}
```

Остальные функции являются составными из вышеперечисленных или не требуют специальных пояснений, например формирование задержки или управление линией PD (Power Down).

После внесения этих изменений мы имеем готовую библиотеку функций FT800 для МК CY8C29466.

Закончим пример, внешний вид которого был показан на рис. 4.

Приведем листинг основного цикла программы, в котором осуществляются измерения и формирование команд для вывода на ди-

сплей. Часть, отвечающую за инициализацию FT800, можно посмотреть в соответствующих примерах и описаниях [1, 4, 5] и, соответственно, скопировать в свой проект. Все API-функции для работы с графикой описаны в *FT_CoPro_Cmds.c.* Их вид полностью соответствует тому, как они описаны в руководстве программиста [5]. Краткие комментарии по работе функций даны в листинге:

```
while(1)
  {
  Ft_Gpu_CoCmd_Dlstart(phost); // начало дисплей-листа (команд для отображения) сопроцессора
  Ft_App_WrCoCmd_Buffer(phost,CLEAR_COLOR_RGB(64,64,64)); //установка значений цвета, которые будут использоваться при очистке экрана
  Ft_App_WrCoCmd_Buffer(phost,CLEAR(1,1,1)); // очистка экрана
  Ft_App_WrCoCmd_Buffer(phost,COLOR_RGB(0xff,0xff,0xff)); // установка цвета графических элементов, следующих за этой командой, в данном примере — шрифта
  Ft_App_WrCoCmd_Buffer(phost,COLOR_A(255)); // установка прозрачности элементов, в данном примере — шрифта
  Ft_Gpu_CoCmd_Text(phost,(360), (0), 27, OPT_RIGHTX, label2); //вывод текстовой строки “Temperature=” в позицию x = 360, y = 0 с выравниванием по правой границе текста
  Ft_Gpu_CoCmd_Text(phost,(480), (0), 27, OPT_RIGHTX, stringT2); // вывод измеренного значения
  Ft_App_WrCoCmd_Buffer(phost,COLOR_A(255)); // прозрачность индикаторов
  if (gVal<=25)
  {Ft_Gpu_CoCmd_BgColor(phost, 0x00ff00);} // установка цвета фона индикатора
  else
  {Ft_Gpu_CoCmd_BgColor(phost, 0xff0000);} // установка цвета фона индикатора
  Ft_Gpu_CoCmd_Gauge(phost, 380,120,80,0,10,10,gVal,100); // вывод индикатора с центром в точке x = 360, y = 120, диаметром 80, внешний вид, количество основных делений — 10, количество вспомогательных делений — 10, измеренное значение, максимальное значение.

  Ft_Gpu_CoCmd_Text(phost,(120), (0), 27, OPT_RIGHTX, label1);
  Ft_Gpu_CoCmd_Text(phost,(240), (0), 27, OPT_RIGHTX, stringT);
  Ft_Gpu_CoCmd_BgColor(phost, 0x0000ff);
  Ft_Gpu_CoCmd_Gauge(phost, 120,120,80,0,10,10,gVal,100);

  Ft_App_WrCoCmd_Buffer(phost,DISPLAY()); // конец дисплей-листа
  Ft_Gpu_CoCmd_Swap(phost); // ожидание нового дисплей-листа после окончания вывода текущего
  Ft_App_Flush_Co_Buffer(phost);
  Ft_Gpu_Hal_WaitCmdfifo_empty(phost); // ожидание освобождения буфера FIFO

    // чтение значения влажности
    rxBuf = 0;
    rxBuf=I2Cm_fSendStart(0x40, I2Cm_WRITE);
    rxBuf=I2Cm_fWrite(0xF5);

    for (c=0; c<=15000; c++)
    {}

    rxBuf=0;
    while (rxBuf!=1)
    {
    rxBuf=I2Cm_fSendRepeatStart(0x40, I2Cm_READ);
    }

    rxBuf=I2Cm_bRead(I2Cm_ACKslave);
    humTempH = (rxBuf<<8);
    rxBuf=I2Cm_bRead(I2Cm_NAKslave);
    I2Cm_SendStop();
    humTempL = rxBuf;

    humTempH+humTempL;
    Humidity = (((float)humResult*125)/65536)-6; // формула приведения измеренных значений к значениям относительной влажности дана в [3]
    stringT = ftoa(Humidity, &iTemp);
    gVal=(ft_uint16_t)fround(Humidity);

//**************************************
// чтение значения температуры
    rxBuf=I2Cm_fSendStart(0x40, I2Cm_WRITE);
    rxBuf=I2Cm_fWrite(0xF3);

    for (c=0; c<=15000; c++)
    {}

    rxBuf=I2Cm_fSendRepeatStart(0x40, I2Cm_READ);

    rxBuf=I2Cm_bRead(I2Cm_ACKslave);
    humTempH = (rxBuf<<8);
    rxBuf=I2Cm_bRead(I2Cm_NAKslave);
    I2Cm_SendStop();
    humTempL = rxBuf;

    humResult = humTempH+humTempL;
    Temperature = (((float)humResult*175.72)/65536)-46.85; // формула приведения измеренных значений к градусам Цельсия дана в [3]

    stringT2 = ftoa(Temperature, &iTemp2);
    gVal = (ft_uint16_t)fround(Temperature);

  }
```

Таким образом, изменив ряд функций, отвечающих за обмен между МК и микросхемой FT800, мы получили готовый работающий набор API-функций от производителя. Такой подход существенно упрощает освоение графического контроллера и позволяет в короткие сроки реализовать свое графическое приложение.

Заключение

На этом примере мы хотели показать, как новый графический контроллер FTDI FT800 и графический модуль VM800B43A-BK на его базе работают с простым 8-разрядным контроллером. Причем производительность этого контроллера относительно небольшая и составляет всего 4 MIPS, а типовая скорость передачи по SPI — порядка 4 Мбит/с (с учетом всех внутренних задержек). При этом микроконтроллер успевает опрашивать датчик и отображать информацию на дисплее в графическом виде без каких-либо видимых глазу мерцаний экрана или других искажений, связанных с недостаточностью быстродействия для смены картинок. Задача микроконтроллера сводится к передаче списка команд, на основании которых осуществляется вывод изображения на дисплей. Управление дисплеем — это целиком и полностью работа микросхемы FT800.

В следующей статье мы расскажем про работу с пользовательскими шрифтами. Речь пойдет о конвертировании шрифтов в формат, поддерживаемый графическим контроллером FT800, и использовании этих шрифтов в приложениях.

Литература

1. Долгушин С. Графический контроллер EVE FT800 компании FTDI // Компоненты и технологии. 2013. № 11.
2. Долгушин С. Измеритель температуры и влажности на базе датчика Silicon Labs Si7005 и дисплейного модуля 4D Systems uLCD-43PT // Компоненты и технологии. 2014. № 2.
3. Datasheet. Si7021 I2C humidity and temperature sensor.
4. Application Note AN 245. FT800 Sample Application Introduction For VM800B and VM800C Development Kits and Windows PC.
5. FT800 Programmer Guide.