Работаем с Vinculum II и W5100. USB от FTDI + Ethernet от Wiznet

Продолжаем цикл статей, посвященных хост-контроллеру Vinculum II компании FTDI.
В последнюю версию среды разработки VNC2 IDE Toolchain был включен драйвер для работы с Ethernet-мостом W5100 компании Wiznet. В настоящей статье предлагаем познакомиться с этой реализацией. Дополнительно рассмотрим процесс самостоятельного переноса фирменного драйвера W5100 от Wiznet на хост-контроллер Vinculum II. Решение такой задачи поможет познакомиться с работой интерфейса SPI контроллера VNC2.

Сергей ДОЛГУШИН
dsa@efo.ru

С момента начала серийного производства компанией FTDI хост-контроллера Vinculum II прошел год. Этот контроллер оказался удачным и востребованным решением. С его помощью можно добавить в разрабатываемое изделие возможность подключения различных USB-устройств: флэш-накопителей, принтеров, HID-устройств (клавиатуры, мыши и т. д.), модемов CDC-класса.

Кроме USB-интерфейса, в настоящее время для встраиваемых приложений также актуальна поддержка сетевых подключений. В связи с этим компания FTDI добавила поддержку хост-контроллером Ethernet-моста W5100 производства компании Wiznet. Микросхемы Ethernet-мостов Wiznet W3150A+, W5100, W5200 и W5300 представляют собой функционально законченные решения для встраиваемых приложений, где требуется наличие канала Ethernet. В частности, микросхема Ethernet-моста W5100 аппаратно реализует следующие протоколы транспортного, сетевого и канального уровней системы OSI (Open System Interconnection): TCP, UDP, IPv4, ICMP, ARP, IGMP и MAC. Также обеспечивается аппаратная поддержка протокола PPPoE (Point-to-point over Ethernet) с PAP/CHAP-протоколами аутентификации. Микросхема W5100 имеет встроенный модуль, который реализует физический уровень Ethernet [11].

К управляющему контроллеру Ethernet-мост может быть подключен по параллельному или SPI-интерфейсу. Для работы с VNC2 будем использовать SPI-интерфейс, так как драйвер FTDI поддерживает работу с W5100 только по этому каналу.

Для реализации нашего примера понадобятся следующие компоненты: отладочная плата VNCLO-MB1A (другое ее название Vinco) на базе 64-выводного хост-контроллера VNC2 (рис. 1), модуль внутрисхемной отладки VNC2 Debug Module и модуль FT811MJ на базе микросхемы

Рис. 1. Отладочная плата VNCLO-MB1A (Vinco)

W5100. Отладочная плата для контроллера Vinculum II выбрана из условий того, что портированные FTDI-драйверы W5100 «привязаны» к 64-выводному контроллеру VNC2. В принципе, можно использовать и другие отладочные модули, на которых установлен контроллер в таком же корпусе.

Модуль Ethernet-контроллера выбран так, чтобы познакомить разработчиков с продукцией российского производителя «Форт-Телеком», который серийно выпускает на базе микросхемы W5100 готовые модули FT810MJ, FT811MJ (рис. 2) и FT5100PD (рис. 3). FT810MJ и FT811MJ — это простые мезонинные модули, на базе которых реализована схема включения микросхемы W5100 со всей необходимой обвязкой. FT810MJ реализует только параллельный интерфейс для связи с управляющим контроллером, FT811MJ — параллельный и SPI. Модуль FT5100PD дополнительно имеет узел PoE (Power over Ethernet), который питает микросхему W5100 и может обеспечивать питание внешних устройств. Стандартный модуль FT5100PD-1212 обеспечивает выходное напряжение 12 В и ток до 1 А.

Рассмотрим подключение микросхемы W5100 к контроллеру VNC2 по интерфейсу SPI. Для этих целей будет использован модуль FT811MJ. Нам потребуются четыре линии для интерфейса SPI и одна линия для формирования аппаратного сброса. При необходимости может быть задействована дополнительная линия для передачи прерывания от W5100.

Компания Wiznet предоставляет для своих микросхем готовые драйверы. Для W5100 они были разработаны для микроконтроллеров Atmel, но могут быть легко портированы на платформы других производителей, в частности хост-контроллер USB FTDI. Для своего контроллера компания FTDI уже адаптировала драйвер Wiznet и включила его в последнюю версию VNC2 IDE Toolchain. Достоинством данной реализации является простота использования: нет необходимости в подробном изучении W5100, его регистров

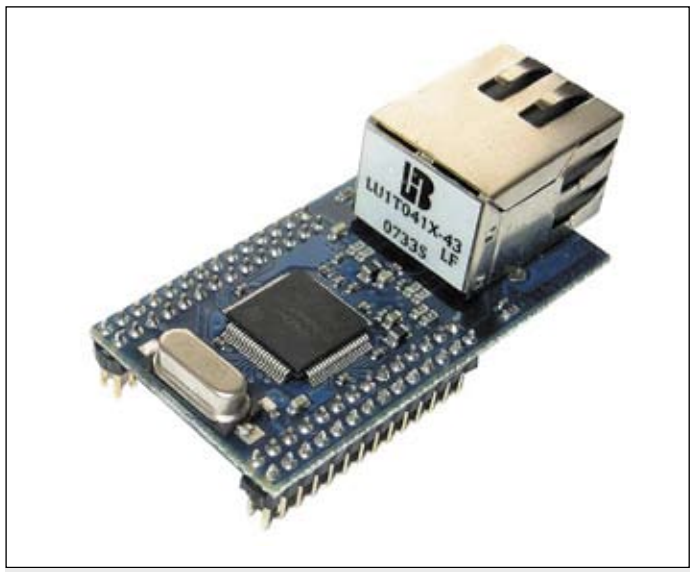

Рис. 2. Внешний вид модуля FT810MJ

Рис. 3. Внешний вид модуля FT5100PD

и порядка работы с ними. Но, к сожалению, на данный момент при использовании драйверов для W5100 от FTDI разработка будет жестко привязана к определенным выводам 64-выводной микросхемы VNC2. Это касается интерфейса SPI, по которому осуществляется управление и передача данных в микросхему Ethernet-моста, а также, например, интерфейса UART. Поэтому далее рассмотрим, как самостоятельно перенести исходный драйвер Wiznet на хост-контроллер FTDI. Такой подход более трудоемок, чем использование готового драйвера от FTDI. Но в данном случае приложение уже не будет привязано к 64-выводному корпусу и жестко заданным выводам периферийных интерфейсов. Сам же процесс переноса заключается, в основном, только в замене функций, отвечающих за работу с интерфейсами (в нашем случае — SPI).

В данной статье не будем останавливаться на вопросах, касающихся структуры приложения для VNC2, драйверов, их инициализации и работы с ними. Эти темы были затронуты ранее [1–3]. Остановимся только на деталях, касающихся работы с Ethernet-мостом W5100.

Описание примеров начнем с подключения модулей друг к другу. Подключение модуля FT811MJ к плате Vinco выполнено в соответствии с указанным в таблице.

Как было отмечено выше, компания FTDI портировала драйвер Wiznet для W5100 на свою платформу — микроконтроллер Vinculum II. В последней версии VNC2 IDE

Таблица. Подключение модуля FT811MJ к модулю VNCLO-MB1A

FT811	VNCLO-MB1A	
X2 – 2	J6 – 7	Reset
X2 – 3	J4 – 6	SCLK
X1 – 2	J4 – 5	MISO
X1 – 1	J4 – 4	MOSI
X2 – 4	J4 – 3	CS

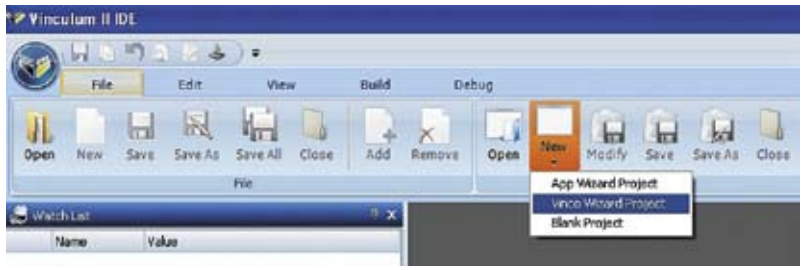

Рис. 4. Вызов мастера создания приложения на базе драйверов Vinco

Toolchain есть два примера, которые показывают основные моменты работы с данным драйвером, — это проекты *TCP* и *UDP*, которые, соответственно, демонстрируют работу с протоколами *TCP* и *UDP*. Для создания своего проекта можно воспользоваться этими примерами или мастером создания нового приложения (рис. 4), с помощью которого, собственно, и были созданы указанные выше *TCP* и *UDP*. Основными файлами в проекте являются *main.c* и *TCP.c*.

В первом файле описаны все необходимые переменные для вызова драйверов, инициализации и запуска ОС. Для корректной работы модуля FT811MJ с контроллером VNC2 потребуется внести дополнение в настройку интерфейса SPI, без которой модуль FT811MJ не будет работать. Это изменение касается функции ***void SPI_Master_Init()***, в которой задаются настройки интерфейса SPI контроллера VNC2. В конец данной функции добавляем аппаратное управление сигналом CS (chip select):

```
spim_iocb.ioctl_code = VOS_IOCTL_SPI_MASTER_AUTO_TOGGLE_
SS;
spim_iocb.set.param = SPI_MASTER_SS_AUTO_TOGGLE_ENABLE_
SS_0;
vos_dev_ioctl(hSPIm, &spim_iocb);
```

Другие изменения в данном файле могут потребоваться, если нужно добавить какие-либо дополнительные интерфейсы, например UART или USB [2, 3].

В файле *TCP.c* реализовано само тестовое приложение, состоящее из двух функций: ***void setup(void)*** и ***void loop(void)***. Рассмотрим их по порядку:

```
void setup(void)
{
  // Перед началом работы с W5100, Ethernet-мост
  // должен быть сброшен
  pinMode(30, OUTPUT);
  digitalWrite(30, LOW);
  delay(5);
  digitalWrite(30, HIGH);

  // Задаем MAC, IP, маску подсети и шлюз
  Ethernet.beginMacIpGwSn(mac_addr, ip_addr, gtw_addr, sub_mask);

  // Открываем порт 80 и ждем установления соединения
  Server.begin(80);

}
```

После выполнения данной функции все необходимые регистры W5100 проинициализированы и установлено соединение с удаленным устройством. По окончании данной функции можно передавать и принимать данные, как, например, это реализовано в следующем блоке:

```
void loop(void)
{

   // Проверка наличия данных во входном буфере
   status = Server.available(80, &clientInfo1);
   if (status == TRUE)
   {
      in_buffer[i] = Client.read(&clientInfo1); // чтение 1 байта
      i++;
      read = TRUE;
   }
   if ((status == FALSE) && (read))
   {
      // проверка условий:
      // принята последовательность “On”
if ((in_buffer[0] == ‘O’) && (in_buffer[1] == ‘n’))
      {
         Server.writeStr(“LED On!\n\r”, 80); // Уведомление удален-
ного клиента, что команда “On” приянта
      }
      // принята последовательность “Off”
      else if ((in_buffer[0] == ‘O’) && (in_buffer[1] == ‘f’) && (in_
buffer[2] == ‘f’))
      {
         digitalWrite(LED1, HIGH); // turn off the on-board LED
         Server.writeStr(“LED Off!\n\r”, 80); // Уведомление удален-
ного клиента, что команда “Off” приянта
      }
      read = FALSE;
      i = 0;
   }
}
```

Протестировать работу приложения можно с помощью утилиты telnet. Запускаем ее следующим образом: Меню **«Пуск» → «Выполнить» → cmd**. В командную строку вводим следующий текст: «telnet 192.168.22.242 50. 80». Это номер порта, который был указан при открытии сокета. После установления соединения, набрав «On», в ответ мы должны увидеть сообщение «Led On».

Данный пример наглядно иллюстрирует простоту реализации, которую предоставляют драйверы W5100 от FTDI. Достаточно общих знаний о микросхеме W5100, чтобы реализовать простое приложение, которое может выполнять функции удаленного управления линиями ввода-вывода микроконтроллера, простой преобразователь Ethernet-UART, работу с USB-накопителями или USB-принтерами.

Но, как уже говорилось выше, у данного решения есть существенный недостаток — невозможность переназначения выводов интерфейсов. Для обхода этих ограничений используем исходный драйвер Wiznet для моста W5100. При этом для реализации TCP- (клиент/сервер) или UDP-протоколов в проект потребуется добавить следующие файлы: *socket.c*, *socket.h*, *w5100.c*, *w5100.h* и *types.h*. Сами драйверы доступны на официальном сайте Wiznet [9] или на сайте официального дистрибьютора — компании «ЭФО» [10]. Архив с исходными кодами называется W5100 Driver Source ver 1.5 и размещен в разделе, посвященном микросхеме W5100.

В файле *socket.c* реализованы функции верхнего уровня, такие как «***Открыть сокет***», «***Закрыть сокет***», «***Установить соединение***» и др. Файл *w5100.c* содержит служебные функции и функции работы с регистрами моста W5100.

Основной задачей при переносе драйвера на выбранный целевой контроллер является за-

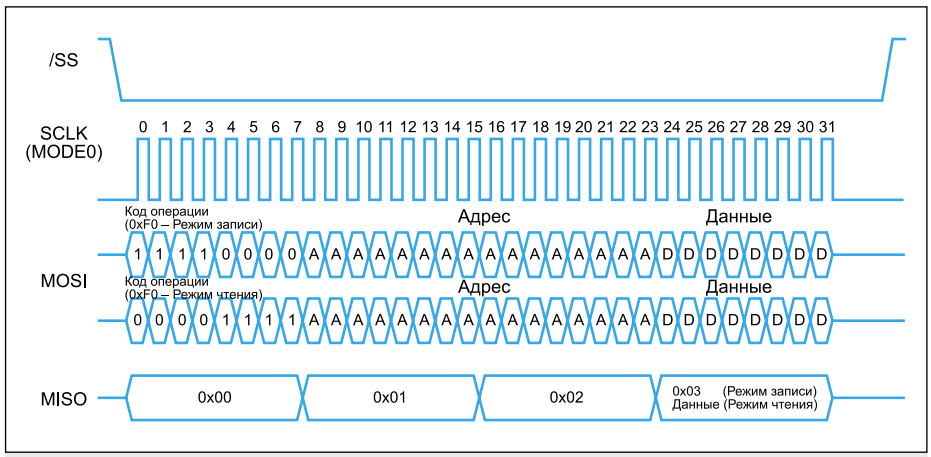

Рис. 5. Диаграмма обмена по SPI

мена служебных функций, отвечающих за связь микроконтроллера и W5100. Их всего четыре: ***IINCHIP_WRITE (uint16 addr,uint8 data)***, ***IINCHIP_READ (uint16 addr,uint8 data)***, ***wiz_write_buf (uint16 addr,uint8* buf,uint16 len)*** и ***wiz_read_buf (uint16addr, uint8* buf,uint16len)***. Эти функции определены в файле *w5100.c*. В листинге ниже приведен исходный код, реализующий запись и чтение по интерфейсу SPI для контроллера FTDI:

```
uint8 IINCHIP_WRITE(uint16 addr,uint8 data)
{
   char spi_buf[]={0xF0};
   char spi_buf1[2];
   char in_buf[4];

   vos_dev_write(hSPIm, spi_buf, 1, NULL);
   spi_buf1[0] = ((addr & 0xFF00) >> 8);
   spi_buf1[1] = (addr & 0x00FF);
   vos_dev_write(hSPIm, spi_buf1, 2, NULL);
   spi_buf1[0] = data;
   vos_dev_write(hSPIm, spi_buf1, 1, NULL);

   return 1;
}

uint8 IINCHIP_READ(uint16 addr)
{
   common_ioctl_cb_t spim_iocb;
   uint8 data;
   char spi_buf[]={0x0F};
   char spi_buf1[2];
   char *in_buf;
   char in_buf2[1];
   unsigned short dataAvail = 0;

   vos_dev_write(hSPIm, spi_buf, 1, NULL);
   spi_buf1[0] = ((addr & 0xFF00) >> 8);
   spi_buf1[1] = (addr & 0x00FF);
   vos_dev_write(hSPIm, spi_buf1, 2, NULL);
   vos_dev_write(hSPIm, 0x00, 1, NULL); // Формирование тактовых
импульсов для передачи байта данных от W5100 в управляющий
контроллер

spim_iocb.ioctl_code = VOS_IOCTL_COMMON_GET_RX_QUEUE_
STATUS;
vos_dev_ioctl(hSPIm, &spim_iocb);
dataAvail = spim_iocb.get.queue_stat;

   if (dataAvail > 1)
      {
         vos_dev_read(hSPIm, in_buf, (dataAvail-1), NULL);
      }

spim_iocb.ioctl_code = VOS_IOCTL_COMMON_GET_RX_QUEUE_
STATUS;
vos_dev_ioctl(hSPIm, &spim_iocb);
dataAvail = spim_iocb.get.queue_stat;
         vos_dev_read(hSPIm, in_buf2, dataAvail, NULL);
   data = in_buf2[0];

return data;
}
```

При обращении управляющего микроконтроллера к регистру W5100 в режиме записи/чтения стандартно передается четыре байта (рис. 5). Первый определяет операцию: запись — «F0», чтение — «0F». Далее следуют два байта адреса (в листинге передача двух байтов адреса осуществляется одним вызовом функции ***vos_dev_write***) и один байт данных. Каждый передаваемый управляющим контроллером байт сопровождается служебным байтом, передаваемым микросхемой W5100. Служебные байты содержат порядковый номер текущего байта, передаваемого микроконтроллером. Таким образом, при осуществлении операции записи в регистр W5100 Ethernet-мост передаст в микроконтроллер четыре байта, содержащие значения «0x00», «0x01», «0x02» и «0x03». В режиме чтения последний, четвертый байт, передаваемый от управляющего контроллера в W5100, является незначащим. Команда ***vos_dev_write (hSPIm, 0x00, 1, NULL)*** предназначена для того, чтобы блок SPI сформировал последовательность тактовых импульсов для передачи Ethernet-мостом W5100 байта данных.

Перед чтением данных из приемного буфера блока SPI VNC2 необходимо очистить его от служебных данных. Для этого введен дополнительный цикл чтения данных: ***vos_dev_read (hSPIm, in_buf, (dataAvail-1), NULL)***.

Других изменений вносить в драйвер Wiznet для переноса его на Vinculum II не требуется. Можно переходить к приложению:

```
// назначаем MAC-, IP-адрес, маску подсети
setGAR(gtw_addr);
   setSHAR(mac_addr);
   setSUBR(sub_mask);
   setSIPR(ip_addr);

   // устанавливаем размеры приемного и передающего буферов
   sysinit(0x00, 0x00);
   // открываем сокет с параметрами: 0 – номер сокета W5100,
   // режима работы (в примере протокол TCP),
   // 50 – номер порта, дополнительные параметры сокета
   // (см. 4, раздел 4.2 socket registers))
   socket(0, Sn_MR_TCP, 50, 0x00);

      do { // ожидаем запрос от клиента, если есть –
            // устанавливаем соединение
   statusWiz = listen(0);
   } while (statusWiz !=1);
```

```
while (1)
{
   // проверяем наличие данных в приемном буфере
   if ((len = getSn_RX_RSR(0)) > 0)
   {
   //считываем из буфера принятые данные
   statusWiz = recv(0, bufTCP, len);
   }
   if (len > 0)
{
// отсылаем принятые данные обратно
   statusWiz = send(0, bufTCP, len);
   }
   statusWiz = 0;

}
```

Данный пример демонстрирует работу моста W5100 в режиме TCP-сервера. Данные, принимаемые от клиента, передаются обратно, что может быть проверено любой терминальной программой на ПК, например telnet. После установления соединения можно протестировать работу примера.

Рассмотренная в статье связка хост-контроллера USB Vinculum II и Ethernet-моста W5100 может обеспечить разрабатываемое устройство широким набором периферийных интерфейсов: USB (хост/периферийное устройство), UART, SPI (мастер/ведомый) и Ethernet. Например, эти возможности можно использовать для сбора информации от различных приборов учета. Так, данные от счетчика электроэнергии можно получить по сети или подключив USB-накопитель.

Рассмотренные примеры работы с мостом W5100 могут быть полезны при переходе со снятой с производства популярной микросхемы W3100A-LF. Причина снятия ее с производства — ликвидация фабрикой-производителем Samsung технологического оборудования, соответствующего процессу 0,35 мкм. Микросхемы этой серии программно и аппаратно не совместимы с выпускаемыми сегодня остальными Ethernet-мостами Wiznet. Использование готового драйвера от FTDI может послужить хорошим подспорьем для быстрого перевода старых проектов на новую элементную базу.

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.
3. Долгушин С. А. Vinculum II — с чего начать? Работаем с интерфейсом UART и USB флэш-диском // Компоненты и технологии. 2011. № 7.
4. W5100. Datasheet. 2008. http://www.seeedstudio.com/depot/images/product/W5100_Datasheet_v1_1_6.pdf
5. Vinculo Development Module. Datasheet. Version 2.0. http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_Vinco.pdf
6. Техническое описание модуля FT811MJ. V. 1.1. ftp://ftp.efo.ru/pub/wiznet/FT811MJ_rus_v1.1.pdf
7. AN_170 Using the FTDI Vinco Libraries. http://www.ftdichip.com/Support/Documents/AppNotes/AN_170_Using_the_Vinco_Libraries.pdf
8. www.ftdichip.com
9. www.wiznet.co.kr
10. www.efo.ru
11. Кривченко И. В. Новая продукция WIZnet для приложений Embedded Internet // Компоненты и технологии. 2007. № 4.