Начинаем работать с Ethernet-контроллерами W7100A компании WIZnet

Сергей ДОЛГУШИН
dsa@efo.ru

Статья посвящена Ethernet-контроллерам W7100A и W7100A-64QFN компании WIZnet. Эти контроллеры, на наш взгляд, незаслуженно обойдены вниманием российских разработчиков, хотя существует множество приложений, где они с успехом могут быть востребованы.

Основная продукция южнокорейской компании WIZnet, куда входят интегральные микросхемы сетевых контроллеров или аппаратных Ethernet-мостов W5300, W5200, W5100 и W3150A+, хорошо знакома отечественным разработчикам. Ethernet-мосты представляют собой функционально законченные решения для встраиваемых приложений, где требуется наличие канала Ethernet. Эти микросхемы аппаратно реализуют протоколы транспортного, сетевого и канального уровней системы OSI (Open System Interconnection): TCP, UDP, IPv4, ICMP, ARP, IGMP и MAC, а также обеспечивают аппаратную поддержку протокола PPPoE (Point-to-point over Ethernet) с PAP/CHAP протоколами аутентификации. Микросхемы W5300, W5200 и W5100 имеют встроенный в кристалл узел, который реализует физический уровень Ethernet [1].

Все перечисленные микросхемы работают под управлением внешнего микроконтроллера по параллельному или последовательному (SPI) интерфейсам. А что если в разрабатываемом устройстве есть только интерфейс UART или ресурсов управляющего микроконтроллера (ограниченные производительность, память и т. п.) недостаточно для работы с аппаратным мостом? В этом случае поможет Ethernet-контроллер W7100A.

Микросхема W7100A объединяет на одном кристалле аппаратный блок Ethernet-моста со встроенными уровнями MAC и PHY, а также MSC-51 совместимое процессорное ядро (рис. 1). Ethernet-блок этого контроллера аналогичен по характеристикам микросхеме W5300. Он поддерживает те же протоколы и обеспечивает до восьми соединений на аппаратном уровне.

Ethernet-контроллеры W7100A производятся в двух корпусах — LQFP-100 и QFN-64. Микросхемы, выполненные в 64-выводном корпусе, имеют несколько отличий:

- Режим работы (auto-negotiation) физического уровня задается только программным путем (для 100-выводного корпуса — программным и аппаратным).
- Доступно 19 линий ввода/вывода общего назначения (для 100-выводного доступно 32 линии).
- Отсутствует возможность подключения внешней памяти.
- Отсутствуют сигналы управления таймерами.
- Доступны только два сигнала, индицирующие состояние линии Ethernet: связь установлена (Link) и скорость (низкий уровень — 10 Мбайт, высокий — 100 Мбайт).
- Доступна одна линия внешнего прерывания (для 100-выводного — четыре линии).

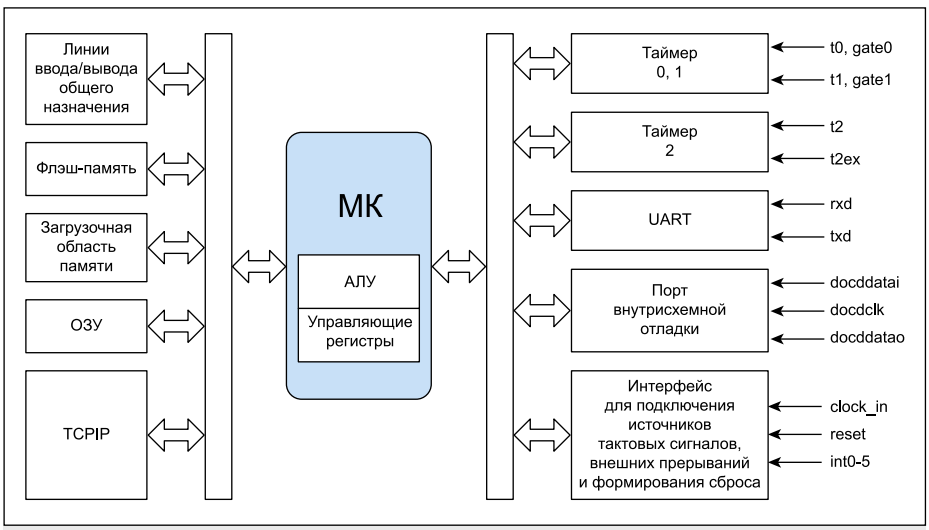

Рис. 1. Блок-схема Ethernet-контроллера W7100A

Для программирования и отладки в Ethernet-контроллерах W7100A предусмотрен специализированный отладочный узел (Debug). Программирование и отладка контроллера в данном случае осуществляется с помощью внутрисхемного отладчика iMCU7100 Debugger (рис. 2) по трехпроводному интерфейсу. Также предусмотрена возможность программирования микросхем W7100A по интерфейсу UART. Этот режим активируется переводом вывода W7100A BOOTEN в высокое состояние.

Для тестирования, разработки и отладки приложений на базе контроллеров W7100A компания WIZnet предлагает набор iMCU7100 EVB (рис. 3). В состав этого набора входят сама отладочная плата на базе контроллера W7100A, кабели Ethernet и COM, адаптер питания и символьный дисплей (16 символов × 2 строки).

Также на базе этой микросхемы выпускаются готовые модули Wiz220IO и Wiz107SR. Они могут быть использованы как готовые или отладочные устройства.

Модуль Wiz220IO (рис. 4) и отладочный набор на его базе Wiz220IO-EVB демонстрируют возможности микросхемы W7100A по реализации веб-сервера. При использовании фирменного ПО (firmware), которое изначально «прошито» в Ethernet-контроллер W7100A, могут быть реализованы следующие функции: управление устройствами по восьми дискретным цифровым выходам и восьми входам; работа с двумя источниками аналогового сигнала (например, датчика-

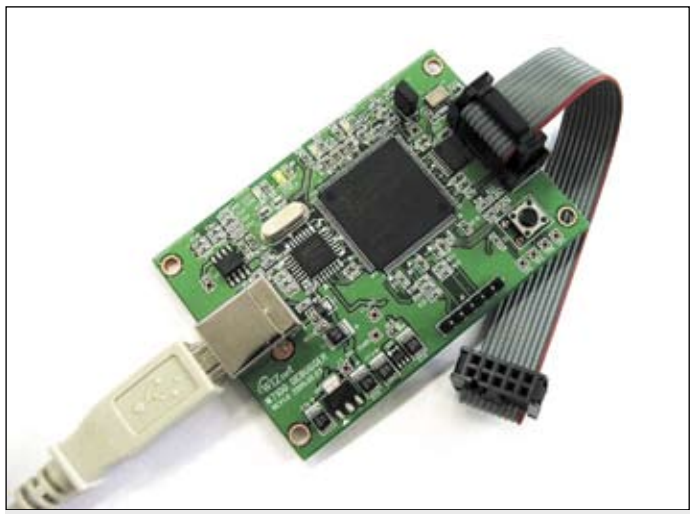

Рис. 2. Внутрисхемный отладчик iMCU7100 Debugger

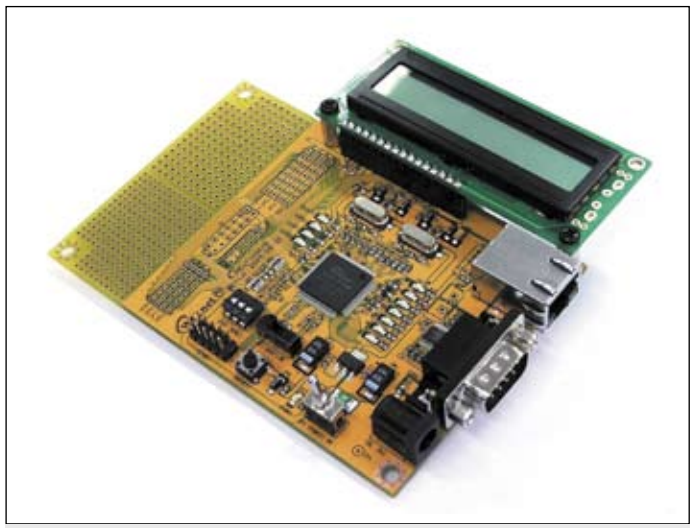

Рис. 3. Отладочный набор iMCU7100 EVB

ми давления) в диапазоне от 0 до 5 В; цифро-аналоговое преобразование по двум линиям в диапазоне от 0 до 4 В; прием и передача данных по интерфейсу UART.

Рис. 4. Модуль Wiz220IO

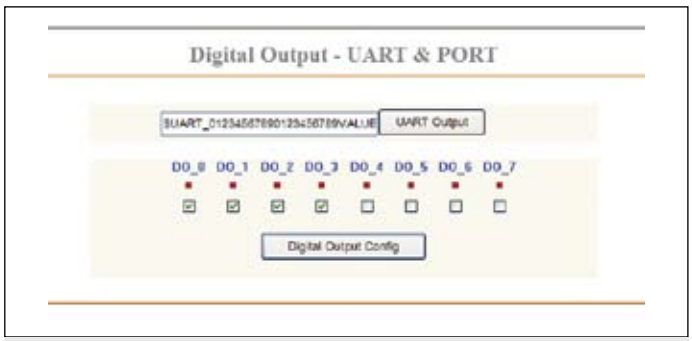

Рис. 5. Пример веб-страницы

Управление модулем осуществляется через веб-интерфейс, что избавляет разработчика от необходимости написания ПО верхнего уровня и обеспечивает совместимость с различными ОС. На рис. 5 приведена одна из веб-страниц, с помощью которой осуществляется управление выходными линиями модуля и передача текстового сообщения по интерфейсу UART. Элементы управления реализованы стандартными средствами языка HTML. Модуль Wiz220IO представляет собой готовое решение, на базе которого можно реализовать удаленную систему управления и контроля. Дизайн веб-интерфейса модуля можно изменять и загружать в Ethernet-контроллер W7100A, не затрагивая при этом его основной программы. Подробнее про работу с модулем Wiz220IO в режиме веб-сервера и реализации HTTP-протокола мы расскажем в следующей статье.

Модуль Wiz107SR реализует готовый конвертер интерфейсов Ethernet-RS-232 (рис. 6). С базовым ПО модуль обеспечивает передачу данных по каналу RS-232 со скоростью до 230 кбод. Управление приемом/передачей — аппаратное (RTS/CTS) или программное. По цене и своим функциональным характеристикам этот модуль идеально подходит для встраиваемых приложений. Небольшие габариты модуля (48×30×18 мм) и удобные крепежные отверстия позволяют установить его в любом удобном месте внутри корпуса прибора. В принципе этот модуль допускает возможность использования на его базе ПО собственной разработки, если фирменное ПО не решает требуемую задачу полностью.

Модуль Wiz108SR представляет собой конвертер интерфейсов Ethernet-RS-422/485. По основным характеристикам он соответствует Wiz107SR, за исключением того, что на плате установлены микросхемы драйверов физического уровня RS-422/485.

Программная поддержка микросхемы W7100A включает в себя драйвер Ethernet в исходных кодах, примеры приложений, демон-

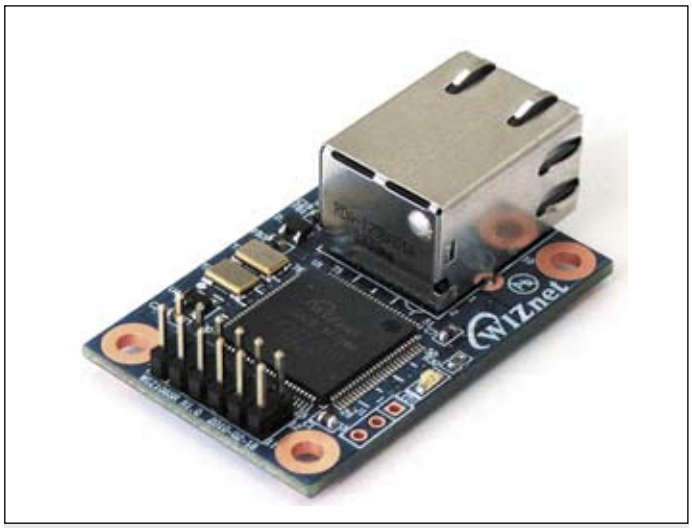

Рис. 6. Модуль Wiz107SR

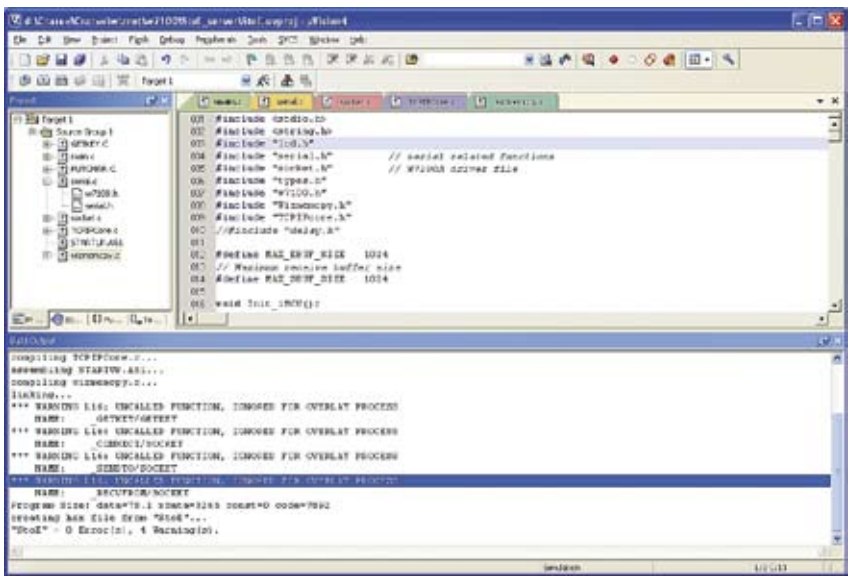

Рис. 7. Проект SToE, открытый в Keil uVision 4

стрирующие различные режимы работы, и специализированные утилиты для отладки (W7100 Debugger) и программирования (WizISP) Ethernet-моста. Все это доступно на официальном сайте WIZnet [4].

Рассмотрим работу с Ethernet-контроллером W7100A на примере реализации преобразователя Ethernet — UART. Такие преобразователи широко востребованы, а рассматриваемый Ethernet-контроллер хорошо подходит по своим возможностям и цене. В качестве отладочного набора будем использовать комплект Wiz220IO-EVB, в который входит модуль Wiz220IO и базовая плата Wiz220 Base Board. На базовой плате реализована схема питания модуля Wiz220IO и интерфейс RS-232 (выведен на стандартный разъем DB9–Male). Остальные элементы базовой платы для данного примера несущественны, о них можно получить информацию, обратившись к руководству пользователя Wiz220IO-EVB.

За основу приложения возьмем пример SToE Server, который можно скачать, пройдя по следующей ссылке [6]. Для работы с приложением понадобится компилятор Keil uVision версии 3 или 4 (рис. 7). Данный проект включает в себя следующие файлы:

- *socket.c*;
- *socket.h*;
- *TCPIIPCore.c*;
- *TCPIIPCore.h*;
- *types.h*;
- *iinchip_conf.h*;
- *W7100.h*.

Эти файлы составляют драйвер для работы с микроконтроллером и Ethernet-блоком. Файлы *serial.c* и *serial.h* — выбор и настройка режима работы UART; *putchar.c* и *putchar.h* — вывод байта в UART; *main.c* — собственно реализует само приложение.

Описание примера начнем с интерфейса UART. Блок последовательного интерфейса контроллера W7100A работает в полнодуплексном режиме со скоростью передачи данных от 2400 до 230400 бит/с. UART контроллера может работать в четырех режимах, для стандартных приложений наиболее подходящим является режим 1 (Mode 1). В этом режиме передается 1 старт-бит, 8 бит данных и 1 стоп-бит. Тактирование блока UART, то есть установка его скорости передачи, осуществляется с помощью таймера 1 или 2. Эти параметры задаются в примере в файле *serial.c*:

```
void InitSerial(void)
{
  ET1 = 0;          // Отключение прерывания таймера
  TMOD = 0x20;      // Выбор режима работы, для тактирования
                    // UART таймер переводится в режим 2 (Mode 2).
                    // В этом режиме таймер играет роль
                    // 8-разрядного счетчика с автозагрузкой.
                    // Скорость передачи задается установкой
                    // регистров PCON и TH1, см. следующие
                    // строки листинга
  PCON |= 0x80;     // Соответствие скоростей передачи и значения
                    // бита SMOD см. в таблице 6.5 [3]

  TH1 = 0xFE;       // Соответствие скоростей передачи и значения
                    // регистра TH1 см. в таблице 6.5 [3], в данном
                    // случае скорость равна 230 кбит/с
  TR1 = 1;          // Запускаем таймер 1 установкой бита TR1
                    // конфигурационного регистра таймеров 0 и 1
                    // TCON (см. раздел 5 [3])
  SCON = 0x50;      // Задаем режим работы Mode 1 блока UART
                    // (биты SM0=0, SM1=1, SM2=2), разрешаем
                    // работу (REN=1)
  RI = 0;           // Сброс флага прерывания, индицирующего
                    // окончание приема данных
  TI = 0;           // Сброс флага прерывания, индицирующего
                    // окончание передачи данных
  ES = 1;           // Разрешаем прерывания блока UART
}
```

Прием данных осуществляется в процедуре обработки прерывания UART — ***void Int() interrupt 4***, описанной в файле *main.c*. Процедура читает входящие данные из буфера UART и помещает их во внутреннюю память. При достижении заданного порога MAX_SBUF_SIZE данные передаются в блок Ethernet.

Передача данных осуществляется функцией ***char putchar (char c)***, описанной в одноименном файле:

```
// Записываем данные в буфер UART
    SBUF = c;
    // Ждем завершения передачи
    while(!TI);
    TI = 0;
    return c;
```

В результате описанных действий интерфейс UART подготовлен к дальнейшей работе.

Переходим к интерфейсу Ethernet и протоколу TCP/IP. Подготовка к работе начинается с инициализации блока Ethernet, которая заключается в присвоении сетевых параметров (MAC-адрес, IP-адрес, адрес шлюза и маска подсети) и задании размеров приемных и выходных буферов для каждого из сокетов. Под приемный и передающий буферы зарезервировано 16 кбайт для каждого. Эти 16 кбайт могут быть распределены между 8 сокетами в зависимости от задач. При использовании одного сокета все 16 кбайт приемного и 16 кбайт передающего буфера могут быть назначены этому сокету. Инициализация выполняется функцией ***void Init_Network(void)*** в файле *main.c*. Процедура проста и не имеет существенных отличий от описанной в [2] для моста W5100. Как и для моста W5100, после инициализации Ethernet-контроллер W7100A должен отвечать на команду ***ping***.

Процессы установления соединения, ожидание входящего подключения, приема/передачи данных и отключения реализованы в функции ***void StoE(SOCKET s, uint16 port, uint8 xdata * e_buf, uint16 mode)*** в файле *main.c*:

```
switch (getSn_SR(s))      // Проверка текущего статуса сокета s
  {
  case SOCK_ESTABLISHED: // Соединение с клиентом
                         // установлено


    if((e_len = getSn_RX_RSR(s)) > 0) // Проверка наличия
                                      // данных в приемном
                                      // буфере Ethernet
    {
      if(e_len > MAX_EBUF_SIZE) e_len = MAX_EBUF_SIZE;
      s_len = recv(s, e_buf, e_len);   // Читаем принятые данные
      for(i=0; i < s_len; i++)
      {
        putchar(*(e_buf + i));        // Передаем их в UART
      }
    }
    SerialToEthernet(s);         // Если есть принятые данные со
                                 // стороны UART, передаем их
                                 // сетевому клиенту
    break;

  case SOCK_CLOSE_WAIT:  // Получен запрос от клиента
                         // на завершение сессии
    printf("CLOSE_WAIT : %d\r\n",(int) s);
    disconnect(s);               // Осуществляем отключение

    break;

  case SOCK_CLOSED:          // Сокет закрыт
    printf("CLOSED. CH : %d\r\n", (int)s);
    close(s);
    socket(s,Sn_MR_TCP,port,mode);  // Открываем сокет s
                                    // в режиме TCP
    break;

  case SOCK_INIT:    // Сокет открыт
    listen(s);       // Ожидаем запрос на подключение
                     // от удаленного клиента
                     // printf("StoE Started. CH : %d \r\n", (int)s);
    break;

  }
```

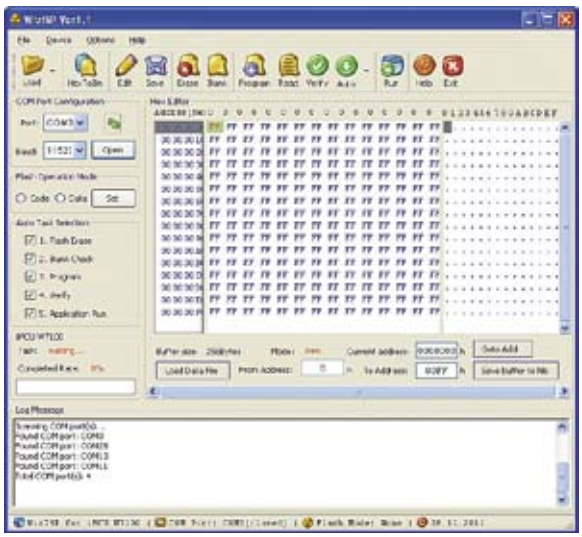

Рис. 8. Утилита WizISP для программирования контроллера W7100A

Функция начинает работу с проверки статусного регистра Sn_SR сокета, указанного при вызове этой функции. В зависимости от его текущего состояния (сокет открыт, соединение установлено, ожидание закрытия соединения или сокет закрыт; подробнее о данном регистре в [3, стр. 92]) будут выполнены действия, описанные в листинге выше. Все функции по работе с сокетами, а также чтение и передача информации реализованы производителем. Использование протоколов TCP или UDP позволяет быстро реализовать сетевой канал обмена, как это показано в данном примере реализации преобразователя UART — Ethernet.

Остается убедиться в работоспособности приложения. Загрузка программы в контроллер W7100A осуществляется по интерфейсу UART с помощью утилиты WizISP (рис. 8). Этот режим активируется подачей «лог. 1» на выход BOOTEN перед запуском Ethernet-контроллера. В качестве переходных кабелей для программирования можно использовать преобразователи USB-Serial компании FTDI. Если используется интерфейс RS-232, подойдут кабели US232R-100 или UC232R-10; если используется прямой выход UART контроллера W7100A — TTL232R-3V3.

На примере программирования модуля Wiz220IO процесс будет выглядеть следующим образом. Модуль будет программироваться и тестироваться в составе набора Wiz220IO — EVB, то есть установленным на плату Wiz220IO Base Board. На этой плате имеются все необходимые элементы для питания модуля Wiz220IO (входное напряжение питания платы Base Board — 5 В) и драйвер интерфейса RS-232. Для подключения отладочного набора к ПК используем преобразователь US232R-100. Дополнительно понадобится переходник с разъемами DB9–Female, так как на преобразователе US232R-100 и базовой плате Wiz220IO установлены разъемы DB9–Male.

Для программирования модуля Wiz220IO будет удобно установить джампер на контактные площадки, помеченные как J3 [Boot]. Контактных площадок две: одна связана с выводом BOOTEN контроллера W7100A, а вторая — с шиной питания 3,3 В. Устанавливаем этот джампер и включаем питание: модуль перейдет в режим ожидания загрузки. На ПК запускаем утилиту WizISP. Начнем процедуру программирования с конвертации hex-файла, полученного в результате компиляции проекта в *bin*-формат. Эта процедура запускается из панели инструментов кнопкой **HexToBin**. Затем установим соединение между утилитой и модулем WizISP. За это ответственны настройки COM Port Configuration. Выбираем номер порта, который присвоен ОС кабелю US232R-100, скорость обмена оставляем по умолчанию равной 115 кбит/с и нажимаем кнопку **Open**. Если все действия были выполнены правильно, в информационном окне **Log Message** утилиты WizISP появится сообщение вида:

```
COM9 is opened @115200.
The flash ROM of W7100 is in CODE mode.
```

Режим загрузки **Flash Operation Mode** устанавливаем в положение **Code**, то есть загрузка будет осуществляться в область памяти программ. Далее производим следующие операции по порядку выполнения: стирание (***Erase***), проверка выполнения стирания (***Blank check***), запись (***Program***), чтение памяти (***Read***) и проверка записи (***Verify***). Эти команды доступны из основного меню **Device** и панели инструментов. Статус проверок стирания и записи отобразится в информационном окне в следующем виде:

```
Blank check successfully.
Flash ROM was verified successfully.
```

Теперь наша микросхема запрограммирована, и нам остается протестировать работу приложения. Самый простой и доступный способ для тестирования — использование стандартной утилиты Windows HyperTerminal. Внешний вид и настройки приведены на рис. 9.

Модуль подключается к ПК по сети и по USB с использованием виртуального COM-порта. Настроив оба канала, активируем соединение и подаем питание на модуль Wiz220IO. В окне COM-порта должно появиться информационное сообщение, аналогичное изображенному

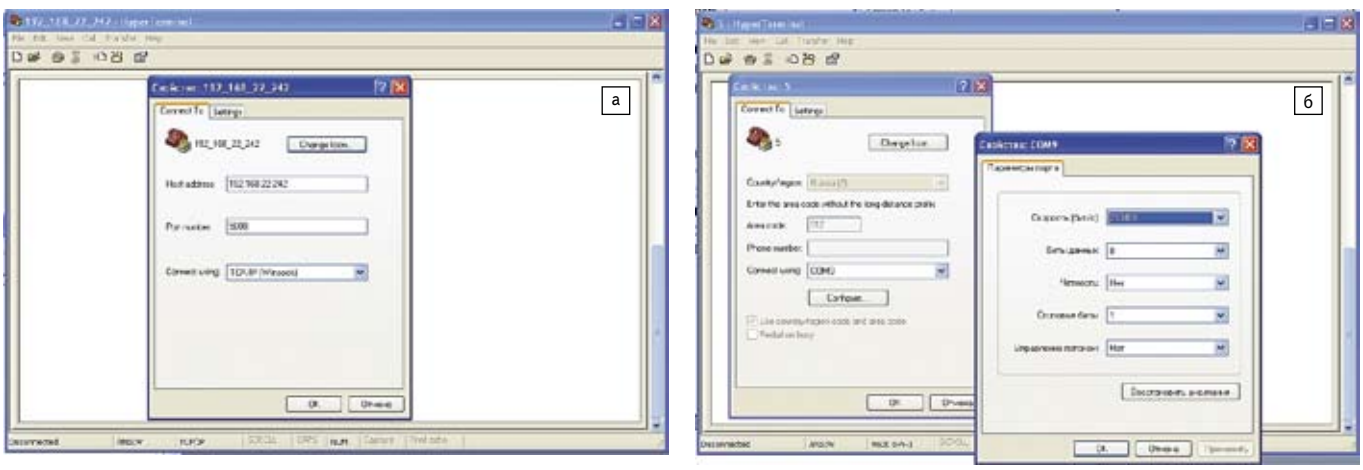

Рис. 9. Приложение HyperTerminal: а) настройки Ethernet-соединения; б) настройки COM-соединения

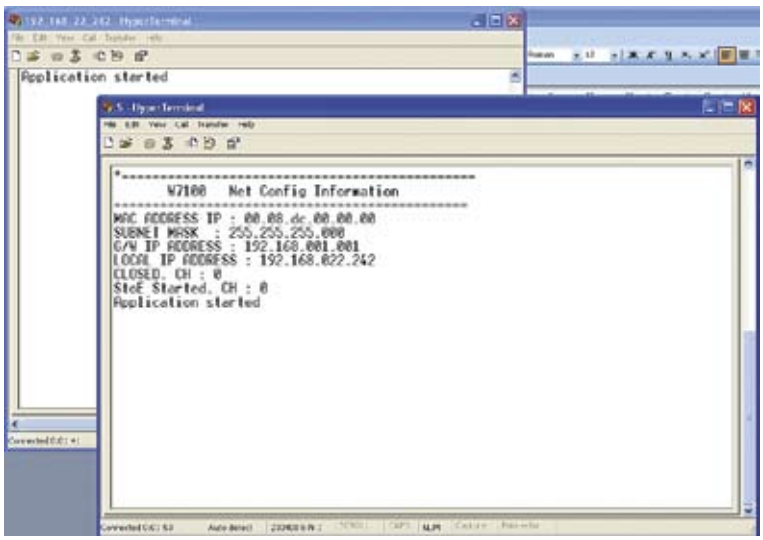

Рис. 10. Тестирование работы приложения

на рис. 10. После этого проверяем передачу с обеих сторон, например, передавая сообщение Application started (рис. 10). Наличие этого сообщения в окнах говорит об успешной передаче с одной и другой сторон.

Итак, в статье были рассмотрены Ethernet-контроллер W7100 компании WIZnet и основные отладочные утилиты и модули для него. Стоит отметить, что контроллер очень прост в освоении. Наличие готовых примеров позволяет быстро проверить его возможности для решения конкретных задач. А наличие таких модулей на его базе, как рассмотренный Wiz220IO, позволяет полностью решить конкретную задачу. Для мелкосерийных изделий такой путь гораздо эффективнее, чем полный цикл разработки и производства аппаратной и программной частей изделия.

Литература

1. Кривченко И. В. Новая продукция WIZnet для приложений Embedded Internet // Компоненты и технологии. 2007. № 4.
2. Долгушин С. А. Работаем с Vinculum II и W5100. USB от FTDI + Ethernet от WIZnet // Компоненты и технологии. 2011. № 12.
3. Internet Embedded MCU W7100A. Datasheet.
4. www.WIZnet.co.kr
5. www.efo.ru
6. http://www1.mymcu.ru:8080/content/articles/WIZnet/SToE.rar