Учебный курс по WIZnet W5100

Фред ИДИ (Fred EADY)
Перевод: Дмитрий ИОФФЕ
dsioffe@yandex.ru

Вы готовы присоединиться к интернет-революции? Если да, то настало время работать с сетевым контроллером с аппаратной реализацией стека TCP-IP W5100 фирмы WIZnet. В этой статье автор рассказывает, как начать ваш первый проект на W5100.
Перевод статьи Фреда Иди (Fred Eady) "iEthernet Bootcamp", Circuit Cellar, Issue 208, опубликованной в ноябре 2007 года.

Недавно я получил письмо от читателя, который спрашивал, почему нет подробных статей о реализации TCP-IP с практическими рекомендациями. Честно говоря, я никогда не думал об этом, потому что обычно предпочитаю формальному стеку TCP-IP небольшие, понятные, пригодные для работы в домашних условиях Ethernet-драйверы. Как журнальный автор, я в первую очередь предположил, что нехватка литературы по стеку TCP-IP вызвана победой стоимости над интересом. Мне приходилось рассматривать много коммерческих предложений реализации стека TCP-IP, и, опираясь на свой опыт, я могу утверждать, что вы получаете то, за что платите. Мои читатели просто не могут позволить себе оплатить полнофункциональный стек TCP-IP для своих приложений и проектов. Таким образом, почему я должен предлагать им читать о том, что они не могут позволить себе использовать?

Вторая причина непопулярности статей о стеке TCP-IP в журнале — его сложность. Кто писал для журналов, тот знает, что количество слов в статье ограничено. Для объяснения всего того, что надо знать о стеках TCP-IP, потребуется целая серия статей.

Должен признать, что раньше я предлагал в своих статьях некоторые довольно дорогие вещи. Но сейчас, в силу указанных причин, стараюсь этого избегать. Однако когда я вижу что-то полезное для типичного читателя технических журналов, то стараюсь следить за этим. Например, я обнаружил, что среди читателей Circuit Cellar очень популярны микросхемы для Ethernet с поддержкой свободных стеков TCP-IP. Также я увидел, что многие читатели, разрабатывающие однокристальные Ethernet-устройства, даже не используют стек TCP-IP. Вместо этого они, так же как и я, используют простые драйверы протоколов, специально написанные для однокристальных Ethernet-устройств, которые они применяют в своих проектах. Я практикую то, что исповедую, и то, о чем собираюсь вам рассказать, — это самый лучший в мире «гаражный» драйвер Ethernet и TCP-IP стек. Вы хотите использовать в своем устройстве однокристальное Ethernet-решение, обеспечивающее всю мощь коммерческих полнофункциональных TCP-IP стеков в простом драйвере Ethernet? Тогда читайте дальше.

WIZnet W5100

Микросхема WIZnet W5100 — это однокристальное Ethernet-решение со встроенным стеком TCP-IP. Такой стек часто называют «зашитым». Все замечательные возможности для доступа в Интернет, которые предоставляет W5100, помещаются в компактном 80-выводном корпусе LQFP. Кроме встроенного стека TCP-IP, W5100 содержит встроенную IEEE 802.3 10Base-T и 802.3u 100Base-TX совместимую реализацию MAC и PHY уровней. Можно ожидать, что раз стек TCP-IP обеспечивает все необходимое, то устройство можно сразу подключить к сети Ethernet. Стек TCP-IP в W5100 поддерживает протоколы TCP, UDP, ICMP и ARP. Этого достаточно для большинства встроенных сетевых Ethernet-приложений, разрабатываемых обычными пользователями. Кроме того, устройство W5100 поддерживает PPPoE, что позволяет использовать его в приложениях ADSL.

Если вы когда-либо экспериментировали со встроенными Ethernet-устройствами, то знаете, что нехватка буферной памяти при приеме или передаче может оказаться болезненной и уменьшить производительность вашего устройства. Производители встроенных устройств для Ethernet знают об этом. Большинство однокристальных Ethernet-решений, предлагаемых в настоящее время, содержит достаточное количество специальной буферной памяти для приема и передачи. Микросхема W5100 не является исключением и имеет 16 кбайт внутренней буферной памяти.

Для обеспечения работы с небольшими микроконтроллерами в W5100 предусмотрено три способа обмена информацией с управляющим микроконтроллером: через SPI и при помощи прямого или косвенного доступа к памяти. Для лучшей совместимости с большинством современных микроконтроллеров напряжение питания W5100 сделано равным 3,3 В. Это позволяет непосредственно подключаться к малопотребляющим микроконтроллерам, которые тоже питаются от этого напряжения. Микросхема W5100 может также работать с разработанными ранее системами с напряжением питания 5 В, потому что ее подсистема ввода/вывода может выдерживать такое напряжение.

W5100 поддерживает до четырех одновременно активных сокетов. Таким образом, все, что пользователь должен знать, — это основы программирования сокетов, потому что W5100 разработана так, чтобы как можно удобнее предоставить все необходимое для работы встроенного устройства с Ethernet. Единственное, что W5100 не сделает для пользователя, — это написание его собственного кода и обработку фрагментации IP-пакетов.

Случайно у меня оказалось несколько микросхем W5100. Давайте создадим устройство на W5100 с нуля. Когда у нас получится аппаратная часть, мы добавим немного кода драйвера для микроконтроллера PIC18LF8722 фирмы Microchip Technology, установленного на нашей отладочной плате.

Устройство отладочной платы

Для удобства я предлагаю проект печатной платы на W5100 (рис. 1), разработанный для фирмы EDTP Electronics. У PIC18LF8722 достаточно способностей, чтобы обеспечить доступ к регистрам W5100 и буферной памяти в любом из режимов на выбор: прямой доступ, косвенный доступ, SPI. При использовании PIC18LF8722, кроме того, в наше распоряжение попадает мощный набор средств разработки от фирмы Microchip. Файл проекта лежит на FTP-сервере Circuit Cellar в формате ExpressPCB. Я выбрал ExpressPCB потому, что этот сервис относительно недорог и доступен каждому.

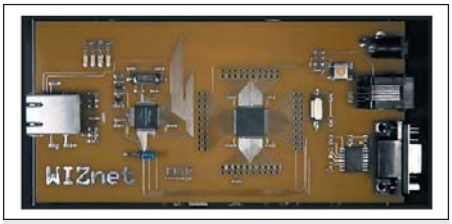

Рис. 1. Отладочная плата с WIZnet W5100 на микроконтроллере PIC18LF8722 фирмы Microchip Technology

Программное обеспечение ExpressPCB загружается бесплатно, и качество получаемых печатных плат очень высокое. Другой плюс, связанный с использованием ExpressPCB, — это то, что вам не понадобится разрабатывать плату для W5100 с нуля. Можно использовать мой шаблон ExpressPCB и доработать его под свою задачу. Если у вас уже есть любимая САПР для подготовки печатных плат, можно легко перенести предлагаемый проект в вашу САПР, используя мой чертеж как руководство.

Проект для EDTP Electronics основан на микроконтроллере PIC18LF8722 в стандартной конфигурации, который подключен к базовой конфигурации W5100. Как можно увидеть на рис. 2, у PIC18LF8722 с запасом хватает выводов, чтобы подключиться к 15-разрядной адресной шине W5100, 8-разрядной шине данных и всем управляющим сигналам (*RD, *WR, *CS и *INT). В дополнение к соединениям для подключения к W5100 в режиме прямого доступа (A0:A14, D0:D7 и управляющие сигналы) я подключил порт SPI и вывод разрешения SPI у W5100 к интерфейсу ввода/вывода SPI-микроконтроллера PIC18LF8722. Это позволит нам получить доступ к внутренностям W5100 в режиме SPI. Режим косвенной адресации, для которого требуются только две адресные линии, все восемь линий данных и все управляющие сигналы, в проекте EDTP тоже легко организовать, так как адресные линии притянуты к «земле» внутри W5100.

Все 80 линий ввода/вывода и питания PIC18F8722 разведены группами по 20 выводов со стандартным шагом 0,1″. Доступ к микроконтроллеру обеспечивается через сертифицированный фирмой Microchip модуль программирования/отладки с тактовой частотой 20 МГц и порт RS-232. Я не включил в комплект никаких источников питания, потому что в продаже более чем достаточно готовых стабилизаторов с выходным напряжением 3,3 В, которые требуются для питания платы с W5100, и внешней аппаратуры для программирования и отладки.

Рядом с микросхемой W5100 на отладочной плате EDTP расположен необходимый для нее кварцевый резонатор на 25 МГц, а также экранированный разъем для сетевого кабеля со встроенным трансформатором (рис. 3). Я выбрал разъем RDI-125BAG1A фирмы U. D. Electronic из следующих соображений. Во-первых, посадочное место под RDI-125BAG1A в точности совпадает с посадочным местом под импульсный трансформатор, которое у меня уже было в библиотеке ExpressPCB. Во-вторых, в нем, кроме пары импульсных трансформаторов для приема и передачи и нагрузочных резисторов, есть еще пара встроенных светодиодных индикаторов. Если вы когда-либо работали с отладочными платами для Ethernet от EDTP, сделанными на ASIX или Microchip, то заметите, что соединения физического уровня W5100 очень похожи на те, что используются в EDTP Electronics ASIX и Microchip ENC29J60.

Вы можете спросить, почему для встроенного в W5100 источника напряжения 1,8 В не предусмотрено никаких развязывающих компонентов. Этот вопрос часто задают службе поддержки. Однако если мы посмотрим на схему включения W5100 из фирменного руководства, то там тоже не увидим развязки по питанию 1,8 В.

Так как отладочная плата для WIZnet W5100 от EDTP предназначена для того, чтобы помочь максимально быстро создать свой проект и запустить его, я подключил светодиоды ко всем индикаторным линиям W5100. Пара светодиодов, встроенных в разъем RDI-125BAG1A, подключена к линиям состояния LINKLED и RXLED. К линиям TXLED, COLLED, FDXLED и SPDLED я подключил дискретные светодиоды, которые можно видеть на рис. 1 среди компонентов типоразмера 0805, окружающих WIZnet W5100. Я также предусмотрел джампер для выбора режима SPI в W5100, если вам этот режим потребуется. Единственная причуда, на которую я должен обратить ваше внимание, — это два резистора сброса номиналом 12,3 кОм, показанные на рис. 3, которые подключены к выводу RSET_BG W5100. Общий вид той части отладочной платы EDTP, которая относится

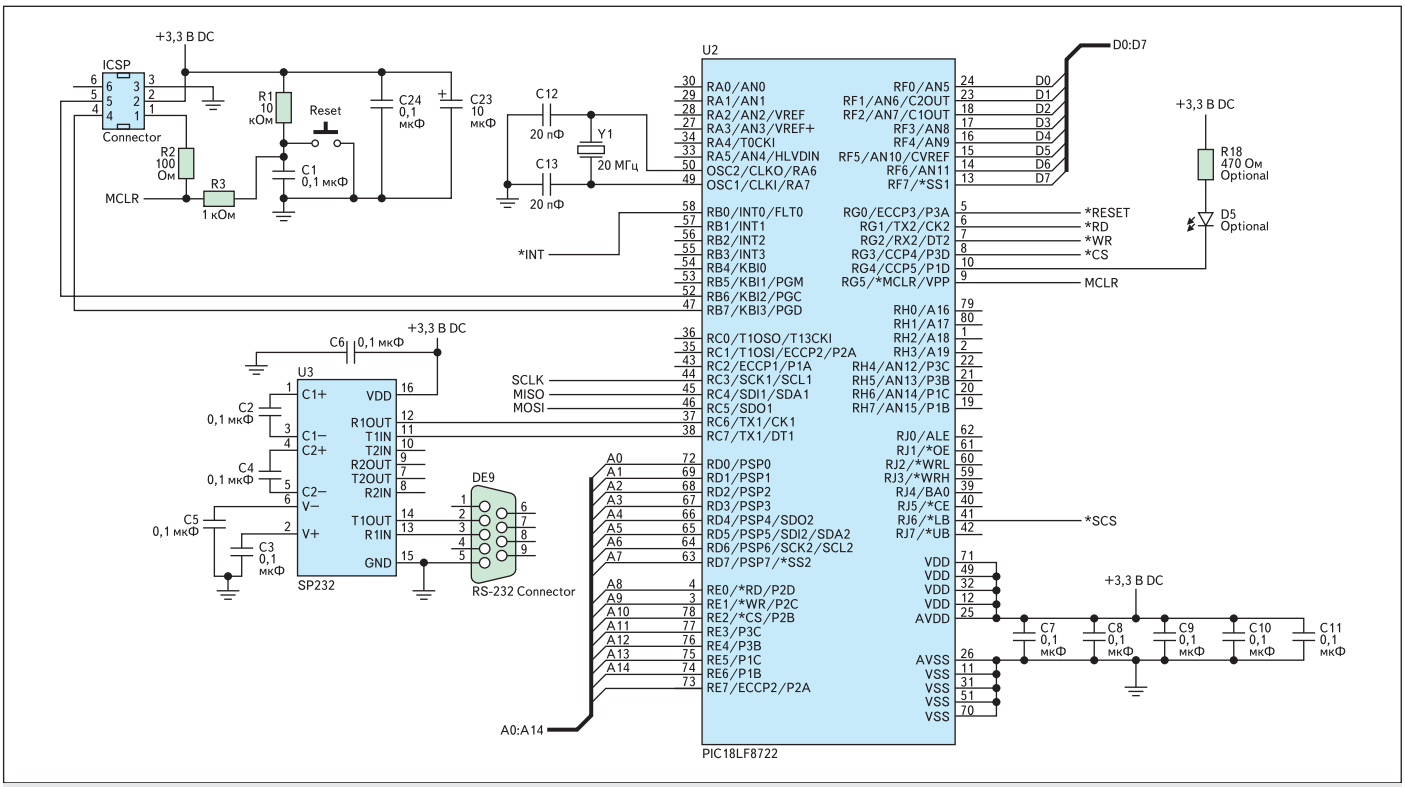

Рис. 2. Схема микроконтроллера PIC18LF8722 в стандартной конфигурации, который подключен к базовой конфигурации W5100

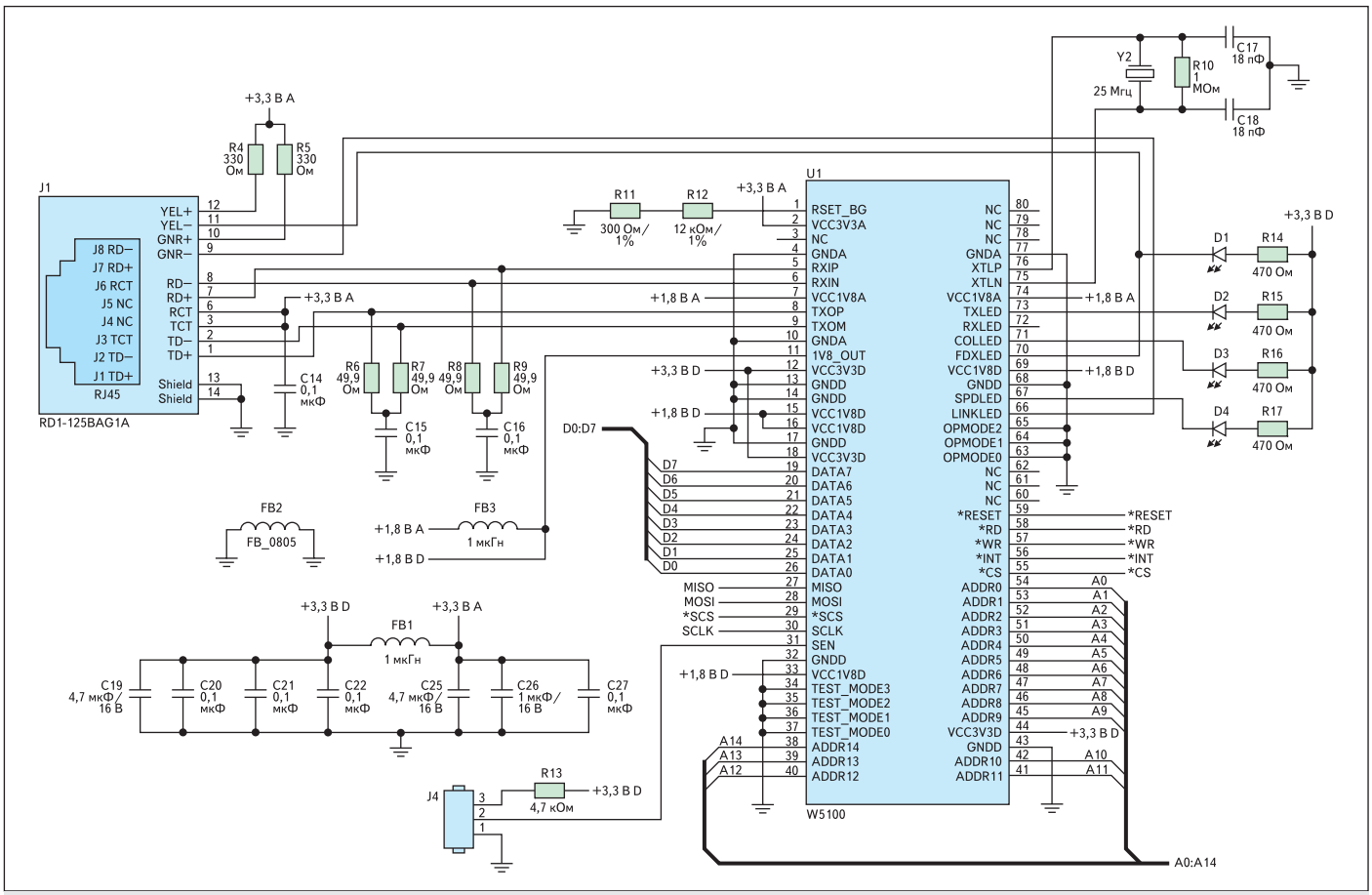

Рис. 3. Структура отладочной платы EDTP

к W5100, показан на рис. 4. Здесь нет ничего такого, с чем бы вы не могли справиться. За исключением пары резисторов 12,3 кОм, другие компоненты рядом с W5100 — это элементы фильтров и развязки. Компоненты физического уровня выстроены в ряд около импульсного трансформатора. Обратите внимание на статусные светодиоды слева и джампер для выбора SPI справа.

Как можно видеть, аппаратная часть устройства с W5100 очень проста. Перед тем, как перейти к программированию W5100, заме-

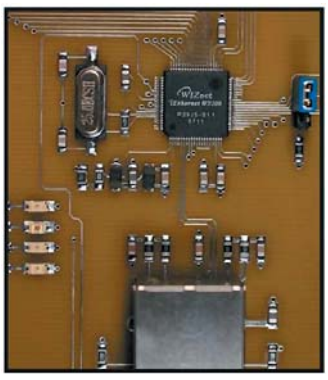

Рис. 4. Общий вид той части отладочной платы EDTP, которая относится к W5100

тим, что на рис. 1 нет светодиода, который я подключил к выводу RG4 PIC18LF8722. Я использовал его для проверки работы программы микроконтроллера. Он вспыхивал с частотой 1 Гц по прерываниям от таймера 3.

«Гаражный» код для WIZnet W5100

С точки зрения программиста W5100 состоит из регистров общего назначения, регистров сокетов, памяти передатчика и памяти приемника. Основное содержание регистров общего назначения — локальные IP и MAC адреса W5100. Кроме того, в них находятся размеры областей памяти для приема и передачи и параметры PPP/PPPoE. Можно сразу убить двух зайцев: использовать регистры общего назначения для проверки линий ввода/вывода PIC18LF8722, написав для него программу, которая будет записывать данные в регистры W5100 и считывать их оттуда. Я буду использовать компилятор HI-TECH PICC-18 C совместно со средой проектирования MPLAB и технологией REAL ICE от Microchip в качестве инструментов для работы с W5100.

Но я не смог заставить мой W5100 правильно общаться с PIC18LF8722. При поверхностном осмотре платы с W5100 никаких проблем не обнаружилось. Тогда я вернулся к коду на C, чтобы попытаться обнаружить ошибку. Оказалось, что большая часть выводов PIC18LF8722, предназначенных для работы с адресом и данными, просто не была припаяна к печатной плате. Для пайки микросхем с малым шагом выводов, подобных W5100, я обычно использую фен и делаю это так часто, что считаю такой процесс совершенно безупречным. Но в этот раз фен меня подвел.

Перейдем к написанию кода для ввода/вывода данных W5100. Официально поставляемый производителем код драйвера, которым я располагал, написан для устройств AVR. Вместо того чтобы писать свой собственный include-файл W5100 для PIC, я адаптировал фирменный include-файл. Непосредственно сейчас мне нужен include-файл с кодом, который извлекает адреса из внутренних регистров W5100. Фирменный include-файл содержит также определения для всего содержимого регистров W5100, которые обязательно пригодятся потом. Я потерял уйму времени на устранение дефектов собственноручной пайки, но вернул часть этих потерь, переделав оригинальные файлы для AVR под PIC.

Первым действием моего официального встроенного ПО был аппаратный сброс W5100 (см. листинг 1). Затем я перевел интерфейс W5100 в режим непосредственной адресации. Это означает, что я не должен трогать регистр режима W5100, который является первым ре-

гистром общего назначения. Таким образом, мы можем начать наш первый тест ввода/вывода W5100/PIC18LF8722 с установки регистров адреса шлюза, расположенных в диапазоне адресов 0x0001:0x0004. Эти регистры определены в преобразованном мною include-файле как GAR0:GAR3. Преобразованный файл получил имя w5100_pic.h. Как можно видеть в листинге 1, я расположил рядом некоторые основные подпрограммы ввода/вывода PIC18LF8722 для чтения и записи регистров W5100. Затем я написал содержимое массива gwayipaddrc для набора регистров адреса шлюза W5100. Чтобы убедиться, что запись в регистры общего назначения выполнена, читаю содержимое GAR0:GAR3 в массив с именем svrmacaddrc. Можете себе представить, как я был доволен, наблюдая IP-адрес шлюза в шестнадцатеричном формате в отладочном окне Watch среды MPLAB (рис. 5). Затем я проверил, используя свою подпрограмму чтения, пару регистров общего RTR0, которая по умолчанию должна содержать 0x07D0, и следующий регистр RCR, по умолчанию содержащий 0x08. Все прошло хорошо. Затем я инициализировал регистры шлюза, MAC-адреса, маски подсети и IP-адреса W5100. В ходе записи IP-адреса шлюза в WIZnet и чтения его оттуда мы отладили базовые операции чтения и записи внутренних регистров W5100.

Следующим шагом на нашем пути подключения платы с W5100 к сети будет запись

```
char gwayipaddrc[4] = {192,168,0,1};
char svrmacaddrc[6];
#define make8(var,offset) ((unsigned int)var >> (offset * 8)) & 0x00FF
#define TO_WIZ TRISF = 0x00
#define FROM_WIZ TRISF = 0xFF
****************************************
void wr_wiz_addr(unsigned int addr)
{
addr_hi = (make8(addr,1));
addr_lo = addr & 0x00FF;
}
void wr_wiz_reg(char reg_data,unsigned int reg_addr)
{
TO_WIZ;
wr_wiz_addr(reg_addr);
data_out = reg_data;
clr_WR;
NOP();
set_WR;
FROM_WIZ;
}
char rd_wiz_reg(unsigned int reg_addr)
{
char data;
wr_wiz_addr(reg_addr);
clr_RD;
NOP();
data = data_in;
set_RD;
return(data);
}
****************************************
clr_RSET;
msecs_timer2 = 0;
while(msecs_timer2 < 2);
set_RSET;
addri = GAR0;
for(i8=0;i8<4;++i8)
wr_wz_reg(gwayipaddrc[i8],addri++);
addri = GAR0;
for(i8=0;i8<4;++i8)
svrmacaddrc[i8] = rd_wz_reg(addri++);
```

Листинг 1. Вы не найдете кода этого уровня в примерах из справочных данных W5100. Но без этих базовых подпрограмм ввода/вывода регистров ничего работать не будет

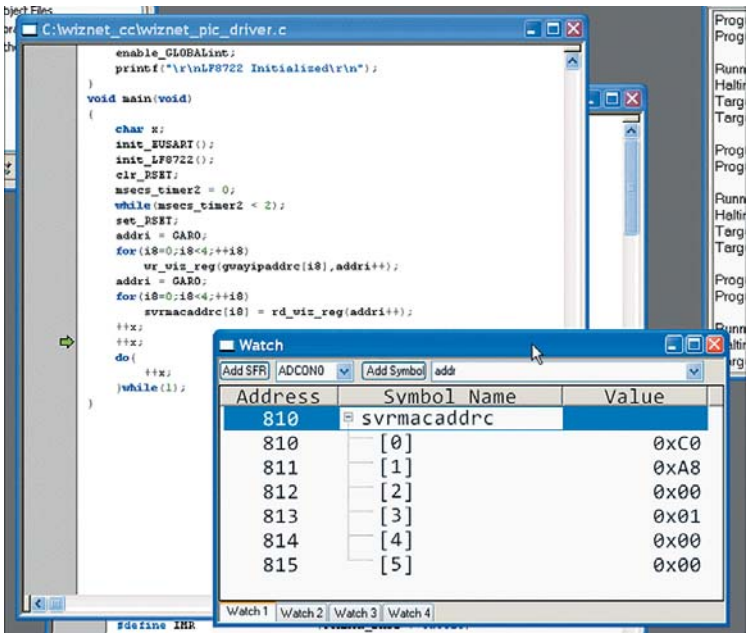

Рис. 5. IP-адрес шлюза в отладочном окне Watch среды MPLAB

информации в память сокетов. Используем отведенные по умолчанию 2 кбайта на сокет. Это означает, что мы не должны трогать регистры RMSR (размер памяти приемника) и TMSR (размер памяти передатчика), которые по умолчанию содержат значения 0x55. Как можно видеть в листинге 2, все, что нам на самом деле нужно сделать, — это установить границы областей памяти приемника и передатчика для каждого из четырех сокетов, поддерживаемых W5100. После распределения памяти для сокетов мы можем сосредоточиться на том, что нам нужно для работы с сокетами W5100.

```
#define chip_base_address 0x0000
#define RX_memory_base_address 0x6000
#define gS0_RX_BASE chip_base_address + RX_memory_base_address
#define gS0_RX_MASK 0x0800 — 1
#define gS1_RX_BASE gS0_RX_BASE + (gS0_RX_MASK + 1)
#define gS1_RX_MASK 0x0800 — 1
#define gS2_RX_BASE gS1_RX_BASE + (gS1_RX_MASK + 1)
#define gS2_RX_MASK 0x0800 — 1
#define gS3_RX_BASE gS2_RX_BASE + (gS2_RX_MASK + 1)
#define gS3_RX_MASK 0x0800 — 1
#define TX_memory_base_address 0x4000
#define gS0_TX_BASE chip_base_address + RX_memory_base_address
#define gS0_TX_MASK 0x0800 — 1
#define gS1_TX_BASE gS0_TX_BASE + (gS0_TX_MASK + 1)
#define gS1_TX_MASK 0x0800 — 1
#define gS2_TX_BASE gS1_TX_BASE + (gS1_TX_MASK + 1)
#define gS2_TX_MASK 0x0800 — 1
#define gS3_TX_BASE gS2_TX_BASE + (gS2_TX_MASK + 1)
#define gS3_TX_MASK 0x0800 — 1
```

Листинг 2. В справочных данных на W5100 это описано при помощи псевдокода

Самая важная часть листинга 3 — это код, который открывает UDP сокет W5100. Первым делом нужно сообщить W5100, с каким типом сокетов мы хотим работать. В настоящий момент мы работаем с UDP. Таким образом, я загружаю в регистр режима сокета 0 значение, задающее UDP. У меня уже есть приложение (EDTP Internet test panel), которое будет посылать ASCII символы на известный порт 7, и, как можно видеть в листинге 3, я загружаю в регистр порта источника сокета 0 число 0x0007. Мы уже загрузили информацию о наших IP и MAC адресах. Таким образом, добавление значения порта источника UDP позволяет нам открыть сокет UDP. По большому количеству нулей в коде инициализации сокета в листинге 3 видно, что мы открываем сокет 0 W5100 в режиме UDP.

```
//SOCKET INTI**************************************************
do{
wr_wiz_reg(Sn_MR_UDP,Sn_MR(0)); //protocol = UDP
wr_wiz_reg(0x00,Sn_PORT0(0)); //well-known ECHO port
wr_wiz_reg(0x07,Sn_PORT1(0));
wr_wiz_reg(Sn_CR_OPEN,Sn_CR(0)); //give the open command
if(rd_wiz_reg(Sn_SR(0)) != SOCK_UDP) //wait for the socket to come
online
wr_wiz_reg(Sn_CR_CLOSE,Sn_CR(0));
}while(rd_wiz_reg(Sn_SR(0)) != SOCK_UDP);
//RECEIVE*****************************************************
do{
//look for incoming UDP datagrams
i16 = rd_wiz_reg(Sn_IR(0));
}while(i16 == 0);
wr_wiz_reg(0x04,Sn_IR(0));
//get the datagram size
hi_byte = rd_wiz_reg(Sn_RX_RSR0(0));
lo_byte = rd_wiz_reg(Sn_RX_RSR1(0));
get_size = make16(hi_byte,lo_byte);
//get the datagram's buffer offset
hi_byte = rd_wiz_reg(Sn_RX_RD0(0));
lo_byte = rd_wiz_reg(Sn_RX_RD1(0));
get_offset = make16(hi_byte,lo_byte) & gS0_RX_MASK;
//calculate the datagram's starting buffer address
get_start_address = gS0_RX_BASE + get_offset;
//UDP header size
header_size = 8;
//store the UDP header information
addri = get_start_address;
for(i8=0;i8<header_size;++i8)
{
packet[ip_destaddr+i8] = rd_wiz_reg(addri++);
++get_offset;
}
//store the UDP data
get_start_address = gS0_RX_BASE + get_offset;
udp_data_size = get_size — header_size;
addri = get_start_address;
for(i8=0;i8<udp_data_size;++i8)
{
```

```
packet[UDP_data+i8] = rd_wiz_reg(addri++);
++get_offset;
}
//update the receive buffer pointers
wr_wiz_reg(Sn_CR_RECV,Sn_CR(0));
//TRANSMIT*****************************************************
//load destination IP address
addri = Sn_DIPR0(0);
for(i8=0;i8<4;++i8)
wr_wiz_reg(packet[ip_destaddr+i8],addri++);
//load destination port address
addri = Sn_DPORT0(0);
for(i8=0;i8<2;++i8)
wr_wiz_reg(packet[UDP_srcport+i8],addri++);
//get transmit buffer offset
hi_byte = rd_wiz_reg(Sn_TX_WR0(0));
lo_byte = rd_wiz_reg(Sn_TX_WR1(0));
get_offset = make16(hi_byte,lo_byte) & gS0_TX_MASK;
//calculate transmit data buffer start address
get_start_address = gS0_TX_BASE + get_offset;
//load data into transmit buffer
addri = get_start_address;
for(i8=0;i8<udp_data_size;++i8)
{
wr_wiz_reg(packet[UDP_data+i8],addri++);
++get_offset;
}
//update transmit buffer pointer
wr_wiz_reg((make8(get_offset,1)),Sn_TX_WR0(0));
wr_wiz_reg((make8(get_offset,0)),Sn_TX_WR1(0));
//send data
wr_wiz_reg(Sn_CR_SEND,Sn_CR(0));
while(rd_wiz_reg(Sn_CR(0)));
```

Листинг 3. Задумайтесь об этом. Все, что должно делать любое коммуникационное устройство — это прием и передача. Я получил логику этого кода из псевдокода в справочных данных W5100

После открытия сокета мы имеем возможность посылать и принимать пакеты UDP. Существует два способа обнаружить пришедший пакет UDP: можно опрашивать регистр Received Size этого сокета или бит RECV его регистра прерываний (Interrupt). Как видно из кода приема пакета UDP, который находится в середине листинга 3, я выбрал последний способ.

Входящий пакет UDP устанавливает бит RECV регистра прерываний сокета. Наша первая реакция на это — сбросить бит RECV, записав 1 в соответствующую этому биту позицию в регистре прерываний. W5100 сама заботится о контрольной сумме, и мы как программисты никогда не смотрим на информацию о ней в пакете UDP. Размер входящего пакета UDP автоматически помещается в регистр Receive Size сокета. Мы читаем содержимое этого регистра и помещаем полученное значение в переменную get_size. Я использую имена переменных из текста в справочных данных на W5100, где это возможно, чтобы облегчить сравнение моего кода драйвера с псевдокодом обработки UDP, приведенным в примере из этого текста. Значение указателя чтения приемного буфера хранится в регистре Read Pointer сокета. Мы будем использовать это значение, чтобы сформировать базу для переменной get_offset. Значение этой переменной используем в сочетании с базовым адресом приемного буфера сокета для того, чтобы вычислить адрес начала заголовка пакета UDP. Предлагаемый в W5100 заголовок пакета UDP состоит из четырех байтов IP-адреса назначения, двух байтов адреса порта назначения и двух байтов информации о размере данных. Таким образом, значение переменной header_size равно восьми. После завершения расчета адреса мы можем использовать нашу подпрограмму чтения регистров W5100, чтобы сохранить данные в ОЗУ PIC18LF8722 для дальнейшего использования.

Если рассуждать логически, то, что не является заголовком, должно быть данными, потому что мы защищены от расчета контрольных сумм архитектурой W5100. При этом мы можем сделать вывод о том, что переменная udp_data_size содержит количество байтов данных, которое нам нужно извлечь и сохранить. Далее, используя наш самодельный код ввода/вывода для W5100, читаем данные из приемного буфера W5100 и сохраняем их в соответствующей области ОЗУ PIC18LF8722. Извлечение пакета UDP закончено. Мы завершаем сеанс приема записью команды RECV в регистр команд сокета, обновляя тем самым указатели приемного буфера. Чтобы правильно сориентироваться в псевдокоде из справочных данных W5100, обратите внимание: переменная udp_data_size не из этого псевдокода. Это переменная от Фреда.

Отправка пакета UDP еще проще, чем его прием. Мы повторно используем полученную ранее информацию и пошлем UDP-сообщение обратно отправителю. Обратимся еще раз к полученному заголовку пакета UDP, который содержит IP-адрес назначения и порт назначения UDP. Мы не напрасно побеспокоились заранее и сохранили оба числа из заголовка. Теперь все, что от нас требуется, — это извлечь их из ОЗУ PIC18LF8722 и загрузить в соответствующие регистры сокета W5100. Я начинаю мою передачу пакета UDP так, как показано в нижней части листинга 3. Используя указатель записи буфера передачи сокета, я рассчитываю, где в W5100 находится этот буфер, и заполняю его данными, которые хочу передать. Затем передаю сохраненные ранее в ОЗУ PIC18LF8722 данные из пакета UDP в буфер передачи W5100. А W5100 передаст данные, расположенные между указателем чтения передатчика (transmit read pointer) и указателем записи передатчика (transmit write pointer) сокета. Таким образом, я должен обновить указатель записи передатчика, добавив к нему число байтов, которое мне надо передать. После того, как я это сделаю, я посылаю команду SEND и жду сигнала успешной передачи, который очистит регистр команд сокета. Теперь я могу послать команду CLOSE в регистр команд сокета, чтобы закрыть сокет или же отправить или принять следующий пакет UDP.

Заключение

Вы закончили учебный курс по W5100. Теперь у вас есть примеры кода для чтения и записи регистров W5100, кода для передачи и отправки пакетов UDP и универсальная отладочная плата для работы.

Вот подсказка, которая поможет вам в самом начале проектирования стендов с W5100: выполните только код до загрузки IP-адреса. Загрузите адрес шлюза, MAC-адрес, маску подсети и IP-адрес. Не открывайте другие сокеты. Сделав все это, вы сможете послать команду PING вашему устройству с W5100. Если получите ответ, то это будет означать, что аппаратура физического уровня и код чтения и записи регистров работают нормально. Тем самым вы также проверите линии адреса, данных и управляющих сигналов W5100 и убедитесь в их исправности. Поднять UDP — приятный и простой способ познакомиться с W5100. EDTP Internet Test Panel — это UDP-программа, которая работает на вашем персональном компьютере [1].

Исходные коды проекта можно загрузить с ftp-сервера журнала Circuit Cellar по адресу **ftp://ftp.circuitcellar.com/pub/Circuit_Cellar/2007/208**.

Литература

1. EDTP Internet test panel EDTP Electronics, Inc. **www.edtp.com**
2. HI-TECH PICC-18 C Compiler HI-TECH Software. **www.htsoft.com**
3. PIC18LF8722 Microcontroller, REAL ICE, and MPLAB Microchip Technology, Inc. **www.microchip.com**
4. RDI-125BAG1A Pulse transformer U.D. Electronic Corp. **www.ude-corp.com**
5. W5100 TCP/IP Ethernet controller WIZnet, Inc. **www.ewiznet.com**