http://www.microchip.com/

Микрокристалл TCP/IP Кипы

Автор: Nilesh Rajbharti Microchip Technology Inc.

ВВЕДЕНИЕ

Примечание:
Это прикладное примечание первоначально было написано для Микрокристалла TCP/IP Кипы выпущенного в 2002; кипа корректируется много раз с тех пор. Самая последняя информация API теперь предусмотрена как Windows файл Подсказки, Стек TCPIP Help.chm, который распространен самого последнего TCP/IP Стека, что может быть загружено из http://www.microchip.com/tcpip.
Стек теперь поддерживает 8, 16 и 32- бит PIC и dsPIC устройства. Это прикладное примечание все еще полезное как материал ссылки.

Есть ничто новый о выполнении TCP/IP (Передача Управляющего Протокола Protocol/Internet) в Микрокристалле microcontrollers. Заинтересованные разработчики могут легко найти много коммерческие и не-коммерческие реализации TCP/IP для продуктов Микрокристалла. Это прикладное примечание описывает подробно собственную свободно доступную реализацию Microchip s Стека TCP/IP.

Микрокристалл TCP/IP Стека является блоком программ, которые обеспечивают услуги в стандарт TCP/основавший приложения IP (Сервер HTTP, Клиент Почты, и т.п.), или могут быть использованы в обычае TCP/основавшем приложение IP. Для того, чтобы лучше иллюстрировать это, полное приложение Сервера HTTP описано в конце этого документа и включено стек исходным кодовым архивом.

Микрокристалл TCP/IP Стека осуществлен в модульном способе, со всеми услугами, создающими очень абстрагировавшие слои. Потенциальным пользователям не нужно знать, что все сложности TCP/IP спецификации. Фактически, те, которые только заинтересованы в сопутствующем приложении Сервера HTTP не нужно специфическое знание TCP/IP. (Специфическая информация о Сервере HTTP начинается на странице 77.)

Это прикладное примечание не обсуждает протоколы TCP/IP в глубине. Те, которые заинтересованы в деталях протоколов могут изучить индивидуально в (RFC) документах. Частичный список ключевых чисел RFC может быть обнаружен в конце этого документа.

АРХИТЕКТУРА СТЕКА

Много TCP/IP реализаций следуют за программной архитектурой именовавшей TCP/IP модели.
Программное обеспечение основанное в этой модели подразделено на многочисленные слои, где слои сложены на верх друг друга (таким образом имя

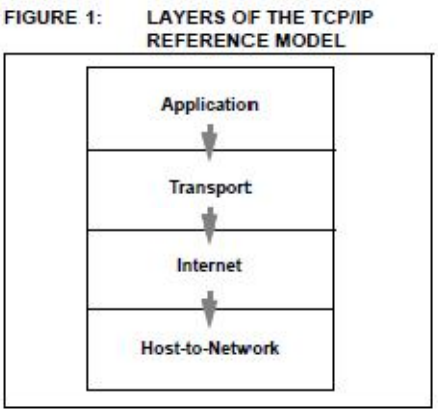
http://www.microchip.com/

TCP/IP Стека) и каждый слой услуги доступов из одного или более слоев непосредственно ниже это. Простая версия TCP/IP модели Стека показана на Рисунке 1.

За спецификацию, многих TCP/IP слоев живы, в значении, которое они не только действуют когда их услуга требуется, но также когда происходят события подобно задержке или новому прибытию пакета. Система с большой памятью данных и программной памятью могут легко включить эти требования. Многозадачная операционная система может обеспечить дополнительное средство и следовательно, может сделать реализацией модульной. Но задача становится трудной, когда система, применяя только 8- бит microcontroller, с несколько сот байтами RAM и ограничившего программную память использован. Кроме того, без доступа к многозадачной операционной системе, пользователь должен обратить специальное внимание, чтобы делать стеком независимым основного приложения. TCP/IP Стека, который плотно интегрирован в свое основное приложение сравнительно легкое осуществлять, и может даже быть очень эффективным. Но такое специализировавшееся стек может предлагать уникальные проблемы как все больше и больше новые приложения интегрированы.

Стек записан в языке программирования C, предназначен как для Microchip C18 так и HI-TECH PICC-18 компиляторы C. В зависимости, от которого использованы, исходные файлы автоматически делают необходимыми изменениями.
Микрокристалл TCP/IP Стека предназначен работать на семействе Microchip s PIC18 microcontrollers только. Кроме того, эта конкретная реализация особо нацелена, чтобы работать на демонстрационной плате Microchip s PICDEM.netTM Internet/Ethernet.
Тем не менее, может легко быть перенастроено в любые аппаратные средства оснащенные microcontroller PIC18.

РИСУНОК 1: СЛОИ МОДЕЛИ ССЫЛКИ TCP/IP



http://www.microchip.com/

Слои Стека

Подобно TCP/IP модели ссылки, Микрокристалл TCP/IP Стека делит Стек TCP/IP во многочисленные слои (Рисунок 2). Код, осуществляющий каждый слой находится в отдельном исходном файле, тогда как услуги и APIs (Приложение, программирующее Интерфейсы), определено через заголовок/включать файлы. В отличие от TCP/IP модели ссылки, многих слоев в Микрокристалле TCP/IP непосредственно доступа Стека один или более слоев, которые - не непосредственно ниже это. Решение как то когда слой должен обходить свой смежный модуль для услуг, ему нужно, делал первоначально на сумме наверху и независимо данной услуге нужна интеллектуальная обработка прежде, чем может быть пройден в следующий слой или нет.

Дополнительное основное отправление из традиционной TCP/IP реализации Стека - дополнение двух новых модулей: StackTask и ARPTask . StackTask управляет операциями стека и всех модулей, пока ARPTask управляет услугами Протокола Адреса Резолюции (ARP) слоя.

Как упомянуто раньше, TCP/IP Стека является стеком живой;
некоторые слои должны быть способными выполнить немного синхронизированные операции асинхронно. Чтобы быть способным удовлетворить это требование и все еще остаться сравнительно независимым основного приложения, использовавшего услуги, Микрокристалл TCP/IP Стека использует широко известную технику вызвавшую кооперативную многозадачную систему. В кооперативной многозадачной системе, есть более чем одна задача; каждый выполняет свою работу и возвращает свое управление чтобы следующая задача может выполнить свою работу. В этом контексте, StackTask и ARPTask - кооперативные задачи.

Обычно кооперативный multitasking (или любой другой тип multitasking, что касается этого), осуществлена также операционная система, или само основное приложение. Микрокристалл TCP/IP Стека предназначен быть независимым любой операционной системы и таким образом, осуществляет свой собственный кооператив многозадачной системы. В результате, может быть использовано в любой системе, независимо от того, что это использует multitasking операционную систему или нет. Тем не менее, приложение, использовавшее Микрокристалл TCP/IP Стека должно также использовать кооператив multitasking сам метод. Это сделан посредством также деля работу во многочисленные задачи, или организовывая основную работу в Конечную Государственную Машину (FSM) и деление длинной работы на многочисленные меньшие рабочие места.
Сервер HTTP, обсужденный ниже в этом документе, следует за последней парадигмой и иллюстрирует как кооперативное приложение может быть осуществлено.

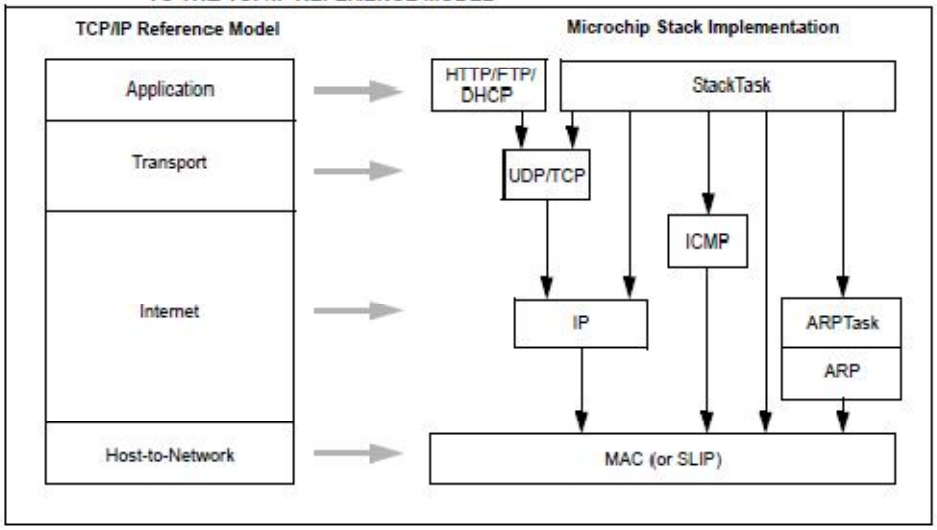
Обратите внимание как Микрокристалл TCP/IP Стека не осуществлял все модули, которые нормально присутствуют в Стеке TCP/IP. Хотя они не присутствуют, они могут всегда осуществлены как отдельная задача или модуль, если потребовалось.

Микрокристалл осуществит дополнительные протоколы основанные в этом стеке.

http://www.microchip.com/

РИСУНОК 2: СРАВНЕНИЕ МИКРОКРИСТАЛЛА TCP/IP СТРУКТУРЫ СТЕКА В TCP/IP МОДЕЛИ ССЫЛКИ HTTP/FTP/ StackTask UDP/TCP ICMP IP
MAC (ИЛИ СКОЛЬЖЕНИЕ) ARPTask ARP

FIGURE 2: COMPARING THE MICROCHIP TCP/IP STACK STRUCTURE TO THE TCP/IP REFERENCE MODEL



КОНФИГУРАЦИЯ СТЕКА

Чтобы снижать процесс конфигурации, стек использует компилятор C.
Для того, чтобы приспосабливаться, выведите из строя или устанавливайте конкретный параметр. Кооперативный multitasking допускает пользователю основной appli- параметр, пользователь изменяет одно или более их cation, чтобы выполнять свои собственные задачи без необходимости человек-определяется. Наиболее их определены в файле заголовка, StackTsk.h . Как уже отмечено, StackTsk.h . Некоторые определяют, что определены в другом, достигающее это функциональное назначение означает, что все файлы приложений показаны соответствующим файловым именем. Как только используя Микрокристалл TCP/IP Стека должно также записано этот файл модифицирован, пользователь должен создать вновь кооператив appliin multitasking способ. Дополнительно к проекту cation, чтобы включать изменения. Определяется - кооператив multitasking проект, пользователь должен сначала указанное в Таблице 1.
поймите немного основные детали конфигурации.

ТАБЛИЦА 1: ОПРЕДЕЛЕНИЯ КОНФИГУРАЦИИ СТЕКА

http://www.microchip.com/

TABLE 1: STACK CONFIGURATION DEFINITIONS

Define	Values	Used By	Purpose
CLOCK_FREQ (compiler.h)	Oscillator Frequency (Hz)	Tick.c	Define system oscillator frequency to determine tick counter value
TICKS_PER_SECONDS	10-255	Tick.c	To calculate a second
TICK_PRESCALE_VALUE	2, 4, 8, 16, 32, 64, 128, 256	Tick.c	To determine tick counter value
MPFS_USE_PGRM	N/A	MP File System (MPFS.c)	Uncomment this if program memory will be used for MPFS storage
MPFS_USE_EEPROM	N/A	MPFS.c	Uncomment this if external serial EEPROM will be used for MPFS storage
MPFS_RESERVE_BLOCK	0-255	MPFS.c	Number of bytes to reserve before MPFS storage starts
EEPROM_CONTROL	External Data EEPROM Control Code	MPFS.c	To address external data EEPROM
STACK_USE_ICMP	N/A	StackTsk.c	Comment this if ICMP is not required
STACK_USE_SLIP	N/A	SLIP.c	Comment this if SLIP is not required
STACK_USE_IP_GLEANING	N/A	StackTsk.c	Comment this if IP Gleaning is not required
STACK_USE_DHCP	N/A	DHCP.c, StackTsk.c	Comment this if DHCP is not required
STACK_USE_FTP_SERVER	N/A	FTP.c	Comment this if FTP Server is not required
STACK_USE_TCP	N/A	TCP.c, StackTsk.c	Comment this if TCP module is not required. This module will be automatically enabled if there is at least one high-level module requiring TCP.
STACK_USE_UDP	N/A	UDP.c, StackTsk.c	Comment this if UDP module is not required. This module will be automatically enabled if there is at least one high-level module requiring UDP.
STACK_CLIENT_MODE	N/A	ARP.c, TCP.c	Client related code will be enabled
TCP_NO_WAIT_FOR_ACK	N/A	TCP.c	TCP will wait for ACK before transmitting next packet
MY_DEFAULT_IP_ADDR_BYTE? MY_DEFAULT_MASK_BYTE? MY_DEFAULT_GATE_BYTE? MY_DEFAULT_MAC_BYTE?	0-255	User Application	Define default IP, MAC, gateway and subnet mask values. Default values are 10.10.5.15 for IP address 00:04:163:00:00:00 for MAC 10.10.5.15 for gateway 255.255.255.0 for subnet mask

ТАБЛИЦА 1: ОПРЕДЕЛЕНИЯ КОНФИГУРАЦИИ СТЕКА (НЕПРЕРЫВНЫЕ)

определение	велечины	пользовательские	
CLOCK_FREQ (compiler.h)	Генератора Частота (Гц)	Tick.c	Определяет системную частоту генератора, чтобы определять встречная величина метки
TICKS_PER_SECONDS	10-255	Tick.c,	чтобы вычислять секунду
TICK_PRESCALE_VALUE	2, 4, 8, 16, 32, 64, 128, 256	Tick.c,	Чтобы определять встречная величина метки
MPFS_USE_PGRM		MP File System (MPFS.c)	разкомментируйте это если программная память будет использована для хранения MPFS
MPFS_USE_EEPROM		MPFS.c	Разкомментируйте это если внешний последовательный EEPROM будет использован для хранения MPFS
MPFS_RESERVE_BLOCK	0-255	MPFS.c	Количество байтов, чтобы резервироваться прежде, чем хранение MPFS начнется
EEPROM_CONTROL	External Data EEPROM Control Code	MPFS.c	Чтобы адресовать внешние данные EEPROM
STACK_USE_ICMP		StackTsk.c	закомментируйте это если ICMP не потребовался
STACK_USE_SLIP		SLIP.c	закомментируйте это если SLIP не потребовался
STACK_USE_IP_GLEANING		StackTsk.c	закомментируйте это если Отбирать IP тщательно не потребовалось

STACK_USE_DHCP		DHCP.c, StackTsk.c	закомментируйте это если DHCP не потребовался
STACK_USE_FTP_SERVER		FTP.c	Закомментируйте это если Сервер FTP не потребовался
STACK_USE_TCP		TCP.c, StackTsk.c	Закомментируйте это если модуль TCP не потребовался. Этот модуль будет авто-matically приспособленное если есть по крайней мере один высокоуровневый модуль, требующий TCP
STACK_USE_UDP		UDP.c, StackTsk.c	Закомментируйте это если модуль UDP не потребовался. Этот модуль будет авто-matically приспособленное если есть по крайней мере один высокоуровневый модуль, требующий UDP.
STACK_CLIENT_MODE		ARP.c, TCP.c	Клиент связавший код будет приспособлен
TCP_NO_WAIT_FOR_ACK		TCP.c	TCP подждет ACK перед передающим следующим пакетом
MY_DEFAULT_IP_ADDR_BYTE? MY_DEFAULT_MASK_BYTE? MY_DEFAULT_GATE_BYTE? MY_DEFAULT_MAC_BYTE?	0-255	User Application	Определите невыполнение IP, MAC, межсетевые и величины маски подсети. Значение по умолчанию: 10.10.5.15 для адреса IP 00:04:163:00:00:00 для MAC 10.10.5.15 для шлюза 255.255.255.0 для маски подсети

MY_IP_BYTE? MY_MASK_BYTE? MY_GATE_BYTE? MY_MAC_BYTE?	0-255	MAC.c, ARP.c, DHCP.c, User Application	Фактический IP, MAC, межсетевые и величины маски подсети как сохранено/определялся приложением. Если DHCP приспособлен, эти величины отражают текущий сервер DHCP назначивший конфигурацию.
MAX_SOCKETS	1-253	TCP.c	Чтобы определять общее число предусмотренных sockets (ограниченное доступным RAM). Компиляция-время чека сделана, чтобы убеждаться, что достаточно sockets доступны для выбранных приложений TCP.
MAX_UDP_SOCKETS	1-254	UDP.c	Чтобы определять общее число sockets перенесших глоток (ограниченное доступным RAM). Компиляция-время чека сделана, чтобы убеждаться, что достаточно sockets доступны для выбранных приложений UDP.
MAC_TX_BUFFER_SIZE	201-1500	TCP.c, MAC.c	Чтобы определять буферный размер индивидуальной передачи
MAX_TX_BUFFER_COUNT	1-255	MAC.c	Чтобы определять общее число буферов передачи. Это число ограничено доступным буферным размером MAC.
MAX_HTTP_CONNECTIONS	1-255	HTTP.c	Чтобы определять

			максимальное число связей HTTP допущенных в любое время
MPFS_WRITE_PAGE_SIZE (MPFS.h)	1-255	MPFS.c	Чтобы определять writable страничный размер для текущего носителя хранения MPFS
FTP_USER_NAME_LEN (FTP.h)	1-31	FTP.c	Чтобы определять максимальную длину строки имени потребителя FTP
MAX_HTTP_ARGS (HTTP.c)	1-31	HTTP.c	Чтобы определять максимальное число областей формы HTML включая имя формы HTML
MAX_HTML_CMD_LEN (HTTP.c)	1-128	HTTP.c	Чтобы определять максимальную длину формы HTML строки URL

ИСПОЛЬЗОВАНИЕ СТЕКА

Файлы, сопровождающие это прикладное примечание содержат полному источнику для Микрокристалла TCP/IP Стека, HTTP и серверов FTP, и DCHP и IP, отбирающий тщательно модули.
Также включен - образец приложения, которое использует стек.

Есть несколько MPLAB файлы проекта разработанные, чтобы иллюстрировать все другие конфигурации в которых стек может быть использован. Эти описаны в Таблице 2.

С каждые модули, включающие стек находится в своем собственном файле, пользователи должны, несомненно включат все подходящие файлы в их проект для правильной компиляции. Полный список модулей и необходимых файлов представлен в Таблице 3 (следующая страница).

http://www.microchip.com/

TABLE 2: STACK PROJECT FILES

Project Name	Purpose	"Defines" Used
HtNICEE.pjt	Microchip TCP/IP Stack using Network Interface Controller (NIC) and external serial EEPROM – HI-TECH® C compiler. Uses IP Gleaning, DHCP, FTP Server, ICMP and HTTP Server.	MPFS_USE_EEPROM, STACK_USE_IP_GLEANING, STACK_USE_DHCP, STACK_USE_FTP_SERVER, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
HtNICPG.pjt	Microchip TCP/IP Stack using NIC and internal program memory – HI-TECH C compiler. Uses IP Gleaning, DHCP, ICMP and HTTP Server.	MPFS_USE_PGRM, STACK_USE_IP_GLEANING, STACK_USE_DHCP, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
HtSlEE.pjt	Microchip TCP/IP Stack using SLIP and external serial EEPROM – HI-TECH Compiler. Uses FTP Server, ICMP and HTTP Server.	STACK_USE_SLIP, MPFS_USE_EEPROM, STACK_USE_FTP_SERVER, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
HtSlPG.pjt	Microchip TCP/IP Stack using SLIP and internal program memory – HI-TECH Compiler. Uses ICMP and HTTP Server.	STACK_USE_SLIP, MPFS_USE_PGRM, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
MPNICEE.pjt	Microchip TCP/IP Stack using NIC and external serial EEPROM – Microchip C18 Compiler. Uses ICMP and HTTP Server.	MPFS_USE_EEPROM, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
MPNICPG.pjt	Microchip TCP/IP Stack using NIC and internal program memory – Microchip C18 Compiler. Uses ICMP and HTTP Server.	MPFS_USE_PGRM, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
MPSlEE.pjt	Microchip TCP/IP Stack using SLIP and external serial EEPROM – Microchip C18 Compiler. Uses ICMP and HTTP Server.	STACK_USE_SLIP, MPFS_USE_EEPROM, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
MPSlPG.pjt	Microchip TCP/IP Stack using SLIP and internal program memory – Microchip C18 Compiler. Uses ICMP and HTTP Server.	STACK_USE_SLIP, MPFS_USE_PGRM, STACK_USE_ICMP, STACK_USE_HTTP_SERVER

ТАБЛИЦА 2: ФАЙЛЫ ПРОЕКТА СТЕКА

Имя проекта		**Определяется пользователем**
HtNICEE.pjt	Микрокристалл TCP/IP Стека, использовавший Сетевого Диспетчера Интерфейса (NIC) и внешний последовательный EEPROM ПРИВЕТ-TECH компилятор C. Использует IP, отбирающий тщательно, DHCP, FTP Сервера, ICMP и Сервера HTTP.	MPFS_USE_EEPROM, STACK_USE_IP_GLEANING, STACK_USE_DHCP, STACK_USE_FTP_SERVER, STACK_USE_ICMP, STACK_USE_HTTP_SERVER

HtNICPG.pjt	Микрокристалл TCP/IP Стека, использовавший NIC и внутренняя программная память ПРИВЕТ-компилятор TEX C. Использование s IP Glea nin g, DHCP, ICMP и Сервер HTTP	MPFS_USE_PGRM, STACK_USE_IP_GLEANING, STACK_USE_DHCP, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
HtSIEE.pjt	Микрокристалл TCP/IP Стека, использовавший СКОЛЬЖЕНИЕ и внешний последовательный EEPROM ПРИВЕТ-Компилятор TEX. Использует FTP Сервера, ICMP и Сервера HTTP.	STACK_USE_SLIP, MPFS_USE_EEPROM, STACK_USE_FTP_SERVER, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
HtSIPG.pjt	Микрокристалл TCP/IP Стека, использовавший СКОЛЬЖЕНИЕ и внутренняя программная память ПРИВЕТ-Компилятор TEX. Использует ICMP и Сервер HTTP.	STACK_USE_SLIP, MPFS_USE_PGRM, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
MPNICEE.pjt	Микрокристалл TCP/IP Стека, использовавший NIC и внешний последовательный EEPROM Компилятор Microchip C18. Использует ICMP и Сервер HTTP.	MPFS_USE_EEPROM, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
MPNICPG.pjt	Микрокристалл TCP/IP Стека, использовавший NIC и внутренняя программная память Компилятор Microchip C18. Использует ICMP и Сервер HTTP.	MPFS_USE_PGRM, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
MPSIEE.pjt	Микрокристалл TCP/IP Стека, использовавший СКОЛЬЖЕНИЕ и внешний последовательный EEPROM Компилятор Microchip C18. Использует ICMP и Сервер HTTP.	STACK_USE_SLIP, MPFS_USE_EEPROM, STACK_USE_ICMP, STACK_USE_HTTP_SERVER
MPSIPG.pjt	Микрокристалл TCP/IP Стека, использовавший СКОЛЬЖЕНИЕ и	STACK_USE_SLIP, MPFS_USE_PGRM, STACK_USE_ICMP,

http://www.microchip.com/

	внутренняя программная память Компилятор Microchip C18. Использует ICMP и Сервер HTTP.	STACK_USE_HTTP_SERVER

TABLE 3: MICROCHIP TCP/IP STACK MODULES AND FILE REQUIREMENTS

Module	Files Required	Purpose
MAC	MAC.c Delay.c	Media Access Layer
SLIP	SLIP.c	Media Access Layer for SLIP
ARP	ARP.c ARPTsk.c MAC.c **or** SLIP.c Helpers.c	Address Resolution Protocol
IP	IP.c MAC.c **or** SLIP.c Helpers.c	Internet Protocol
ICMP	ICMP.c StackTsk.c IP.c MAC.c **or** SLIP.c Helpers.c	Internet Control Message Protocol
TCP	StackTsk.c TCP.c IP.c MAC.c **or** SLIP.c Helpers.c Tick.c	Transmission Control Protocol
UDP	StackTsk.c UDP.c IP.c MAC.c **or** SLIP.c Helpers.c	User Datagram Protocol
Stack Manager	StackTsk.c TCP.c IP.c ICMP.c ARPTsk.c ARP.c MAC.c **or** SLIP.c Tick.c Helpers.c	Stack Manager ("StackTask"), which coordinates the other Microchip TCP/IP Stack modules
HTTP Server	HTTP.c TCP.c IP.c MAC.c **or** SLIP.c Helpers.c Tick.c MPFS.c XEEPROM.c(1)	HyperText Transfer Protocol Server
DHCP Client	DHCP.c UDP.c IP.c MAC.c Helpers.c Tick.c	Dynamic Host Configuration Protocol

Note 1: Required only if the external serial EEPROM for MPFS Storage option (MPFS_USE_EEPROM definition) is enabled. If selected, the corresponding MPFS image file must be included. (Refer to "MPFS Image Builder" (page 84) for additional information.)

http://www.microchip.com/

Таблица 3: МОДУЛИ СТЕКА MICROCHIP TCP/IP И ФАЙЛОВЫЕ ТРЕБОВАНИЯ

Module	**Требуемые файлы**	
MAC	MAC.c Delay.c	Слой Доступа Носителя
SLIP	SLIP.c	Слой Доступа Носителя для SLIP
ARP	ARP.c ARPTsk.c MAC.c или SLIP.c Helpers.c	Протокол Адреса Резолюции
IP	IP.c MAC.c or SLIP.c Helpers.c	Протокол Internet
ICMP	ICMP.c StackTsk.c IP.c MAC.c or SLIP.c Helpers.c	Управляющий Протокол Сообщения Internet
TCP	StackTsk.c TCP.c IP.c MAC.c or SLIP.c Helpers.c Tick.c	Управляющий Протокол Передачи
UDP	StackTsk.c UDP.c IP.c MAC.c or SLIP.c Helpers.c	Протокол Датаграммы Пользователя
Stack Manager	StackTsk.c TCP.c IP.c ICMP.c ARPTsk.c ARP.c MAC.c or SLIP.c Tick.c Helpers.c	Менеджер Стека (StackTask), который координирует другой Стек Microchip TCP/IP модуля
HTTP Server	HTTP.c TCP.c IP.c MAC.c or SLIP.c Helpers.c Tick.c MPFS.c XEEPROM.c (1)	Сервер протокола Передачи HyperText
DHCP Client	DHCP.c UDP.c	Динамический Главный Протокол Конфигурации

	IP.c MAC.c Helpers.c Tick.c	
IP Gleaning	StackTsk.c ARP.c ARPTsk.c ICMP.c MAC.c or SLIP.c	Чтобы конфигурировать узловой адрес IP только
FTP Server	FTP.c TCP.c IP.c MAC.c or SLIP.c	Сервер Протокола Файловой Передачи

Примечание 1:
Требовавшееся только если внешний последовательный EEPROM для опции Памяти MPFS (определение MPFS_USE_EEPROM) приспособлен. Если выбрано, соответствующий файл образа MPFS должен быть включен. (Посмотрите "Разработчика Образа MPFS" (страница 84) для дополнительного information.)

Как только проект установлен с включенными подходящими файлами, основной прикладной исходный файл должен быть модифицирован, чтобы включать программирующие предложения показанные в Примере 1. Для полной работы примера, сошлитесь на исходный код для Websrvr.c. Этот исходный файл, вместе с другими файлами стека, осуществляет Сервер HTTP.

Для того, чтобы понимать как специализированная логика выполнена кооперативным multitasking, ссшлитесь на исходный код для HTTP.c (задача Сервера HTTP). Без кооперативного multitasking, приложения, использовавшие стек должны поделить их логику в несколько меньшие задачи, с каждым ограниченное из монополизации CPU на расширенные периоды. Код для HTTP.c демонстрирует государственному машинному методу деления длинного приложения в меньшие государственные машинные состояния, возврат управления в основное приложение всякий раз, когда логика должна подождать определенное событие.

ПРИМЕР 1: КОДОВЫЕ ДОПОЛНЕНИЯ В ОСНОВНОЕ ПРИЛОЖЕНИЕ

```
// Объявите этот файл как основной прикладной файл
#define THIS_IS_STACK_APPLICATION


#include  StackTsk.h
 #include  Tick.h
#include  dhcp.h  // Только если DHCP использован.
#include  http.h  // Только если HTTP использован.
```

```
#include  ftp.h  // Только если FTP использован.
// Другой специализированный включать файлы
...


// основная точка входа
void main(void)
 {
// Выполните специализированную инициализацию
...


// Инициализируйте компоненты Стека.
// Если StackApplication использован, инициализируйте это тоже.

TickInit();
StackInit();
HTTPInit(); // Только если HTTP использован.
FTPInit(); // Только если FTP использован.


// Ввод в бесконечный программный цикл
while(1)
{

// Счет метки Коррекции. Может быть сделано через прерывание.
TickUpdate();

// Позволившее Менеджера Стека выполнять свою задачу.
StackTask();

// Позволенно любому приложению Стека выполнять свою задачу.
HTTPServer(); // Только если HTTP использован.
FTPServer(); // Только если FTP использован.

// Логика Приложения находится здесь.
DoAppSpecificTask();
}
}
```

МОДУЛИ СТЕКА И APIs

Микрокристалл TCP/IP Стека состоит из многих модулей.
Для того, чтобы использовать модуль, пользователь должен просмотреть и понимать свою цель и APIs. Общий обзор каждого модуля, вместе с описанием их APIs, приведен в следующих секциях.

http://www.microchip.com/

Доступ Носителя Управляющего Слоя (MAC)

Версия Микрокристалла TCP/IP Стека включенного в это прикладное примечание особо записана, чтобы использовать Сетевого Диспетчера Интерфейса Realtek RTL8019AS (NIC). RTL8019AS - NE2000 совместимый NIC, который осуществляет как медицинский осмотр Ethernet (PHY) так и слои MAC. Если другой NIC должен быть использован, пользователи должны модифицировать или создавать новый MAC.cfile, чтобы осуществлять доступ. Так же долго (длиной) как услуги, предусмотренные MAC.c не измениться, все другие модули останется неизменным.

Стек использует on-чип SRAM доступный на NIC как хранение буфера, пока более высокий модуль уровня не прочитает это. Это также выполняет необходимые вычисления контрольной суммы IP в буфере NIC SRAM. Дополнительно к буферу приемника FIFO управляемому NIC собой, слоем MAC управляет своей собственной очередью передачи. Используя эту очередь, вызывающий оператор может передать сообщение и запрашивать MAC, чтобы резервировать это чтобы то же сообщение может быть передано повторно, если потребовалось. Пользователь может определить размеры для буфера передачи, передавать очередь и очередь приемника, использовавшие C определяет (смотри Таблицу 1, "Определения Конфигурации Стека" более подробно).

Последовательный Протокол Строки Internet (SLIP)

SLIP использует последовательный кабель как основной носитель связи, а не кабель ethernet. SLIP не требует NIC, и таким образом предложения очень простая и недорогая связь IP. SLIP является обычно взаимно-однозначной связью, где главный компьютер действует как клиент. Модуль SLIP предназначен действовать с базирующимся компьютером Windows , все-же может быть модифицирован, чтобы работать с другими операционными системами с лишь некоторыми изменениями. C модульным проектом стека, только необходимо должно связывать модуль SLIP (SLIP.c), чтобы использовать это.
APIs Предусмотренное модулем SLIP по существу такие же как и те использованные MAC (посмотрите описания API для MAC на следующих страницах относительно деталей).

SLIP использует прерывание управлявшее последовательной передачей данных, по сравнению с опрашивающим методом использованным NIC MAC.
Основное приложение должно объявить вручителя прерывания и вызывать вручителя прерывания SLIP, MACISR. Для дополнительных деталей, сошлитесь на исходный код для Websrvr.c, программа примера сервера Web включена в файловый архив.

Для того, чтобы подключать к Микрокристаллу TCP/IP Стека, использовавшему модуль SLIP, главная система должна быть сориентирована на связь SLIP. Посмотрите "Демонстрационные Приложения" (страница 87) более подробно.

http://www.microchip.com/

//**

MACInit

Эта функция инициализирует слой MAC. Это инициализирует внутренние буферы и сбрасывает NIC в известное состояние.

Syntax

void MACInit()

Parameters

None

Return Values

None

Pre-Condition

None

Side Effects
Побочные Эффекты

Вся незаконченная передача и приемы отвергнуты.

Пример

//**

MACIsTxReady

Эта функция указывает независимо, по крайней мере, один буфер передачи MAC - пустой или нет.

Syntax

BOOL MACIsTxReady()

Parameters

http://www.microchip.com/

None

Return Values
ИСТИНА: Если по крайней мере один буфер передачи MAC пустой.
ЛОЖЬ: Если все буферы передачи MAC полные.

Pre-Condition

None

Side Effects

None

Remarks

None

Example

```
// Проверьте готовность передачи MAC...
If ( MACIsTxReady() )
{


// Буфер Передачи пустой, передает сообщение.
...


//******************************************************************
```

MACGetHeader

Эта функция проверяет буфер приемника MAC; если любой пакет обнаружен, он возвращает дистанционную главную и информацию пакета данных.

Syntax

```
BOOL MACGetHeader(MAC_ADDR *remote, BYTE *type)
```

Parameters

Remote [out]
Дистанционный [out]

Дистанционный адрес MAC

наберите [out]

Тип пакета Данных

Возможные величины для этого параметра:

Значение	Величины
MAC_IP	пакет данных IP получен
MAC_ARP	пакет данных ARP получен
MAC_UNKNOWN	неизвестный или неподдерживаемый пакет данных получен

Return Values
ИСТИНА: Если пакет данных получен и оказывается быть в силе. Все параметры заполнены.
ЛОЖЬ: Если никакой пакет данных не получен или оказывается быть недействительно.
.

Pre-Condition

None

Side Effects

None

Remarks

Как только пакет данных будет выбран разговором MACGetHeader, полный пакет данных должен быть выбран (использование MACGet) и отвергнутое (использование MACDiscardRx). Пользователи не могут вызвать многочисленное время MACGetHeader, чтобы получать многочисленные пакеты и выбирать данные позже.

Example

```
// Получите возможный пакет данных инфо.

 if ( MACGetHeader(&RemoteNodeMAC, &PacketType) )
 {
// A Пакет получен, обработан это.
...

// Как только сделано, отвергните это.

MACDiscardRx();
...
```

http://www.microchip.com/

//**

MACGet

Эта функция возвращает следующему байту из активного буфера передачи или приемника.

Syntax

BYTE MACGet()

Parameters

None

Return Values
Data byte

Pre-Condition

MACGetHeader, MACPutHeader, MACSetRxBuffer or MACSetTxBuffer вызван.

Side Effects

None

Remarks

Активный буфер (независимо передача или приемник), определены предшествующими вызовами в MACGetHeader, MACPutHeader, MACSetRxBuffer, или функции MACSetTxBuffer. Например, если MACGetHeader следует за MACGet, буфер приемника становится активным буфером. Но если MACPutHeader следует за MACGet, буфер передачи становится активным буфером.

Example 1

```
// Получите возможный пакет данных инфо.

 if ( MACGetHeader(&RemoteNode, &PacketType) )
 {
                 // A Пакет получен, обработан это.
                     data = MACGet();
                          ...
                 // Когда сделано, отвергните это.
                     MACDiscardRx();
    ...
```

http://www.microchip.com/

Example 2

```
// Загрузка пакет для передачи

 if ( MACIsTxReady() )
{
                // Заголовок Загрузки MAC.
                  MACPutHeader(&RemoteNode, MAC_ARP);

                // Получите активный буфер передачи.
                  Buffer = MACGetTxBuffer();

               // Загрузка все данные.
                    ...


          // Мы хотим вычислить контрольную сумму целого пакета.
          // Установившее указатель буферного доступа передачи в начало пакета.

             MACSetTxBuffer(buffer, 0);

// Прочитавшее буферное содержимое передачи, чтобы вычислять контрольную
сумму.
            checksum += MACGet();
              ...
              ...


//**************************************************************************
```

MACGetArray

Эта функция выбирает массив байт из активного буфера передачи или приемника.

Syntax

```
WORD MACGetArray(BYTE *val, WORD len)
```

Parameters

val [out]
Указатель в байтовый массив

len [in]
Количество байтов, чтобы выбираться

Return Values

Общее число байтов выбиралось.

Pre-Condition
MACGetHeader, MACPutHeader,MACSetRxBuffer or MACSetTxBuffer вызван.

Side Effects

None

Remarks

Вызывающий оператор должен убедиться, что общее число байтов данных выбирало не превышает доступные данные в текущем буфере. Эта функция не проверяет на наличие буфера выходить за границы условие.

Example

```
// Получите возможный пакет данных инфо.
if ( MACGetHeader(&RemoteNode, &PacketType) )
{

        // A Пакет получен, обработан это.
            actualCount = MACGetArray(data, count);
              ...

//******************************************************************
```

MACDiscardRx

Эта функция отвергает буферные данные активного приемника и выделяет, который буферизует как свободный.

Syntax
void MACDiscardRx()

Parameters
None

Return Values
None

http://www.microchip.com/

Pre-Condition
MACGetHeader вызван и возвращан ИСТИНА.

Side Effects
None

Remarks

Вызывающий оператор должен убедиться, что есть по крайней мере один пакет данных получал перед разговором этой функции. MACGetHeader должен быть использован, чтобы определять должен буфер приемника быть отвергнут или нет.

Example

```
// Получите возможный пакет данных инфо.
 if ( MACGetHeader(&RemoteNode, &PacketType) )
{
         // A Пакет получен, обработан это.
          actualCount = MACGetArray(data, count);
           ...

        // Сделанным обрабатывая это. Отвергните это.
        MACDiscardRx();
          ...
```

//**

MACPutHeader

Эта функция собирает заголовок MAC и загружает это в активный буфер передачи.

Syntax
void MACPutHeader(MAC_ADDR *remote, BYTE type, WORD dataLen)

Parameters
remote [in]

Дистанционный узловой адрес MAC

type [in]

Тип пакета данных, посылаемых

Возможные величины для этого параметра:

Значение	Величины
MAC_IP	пакет данных IP должен быть передан
MAC_ARP	пакет данных ARP должен быть передан

Return Values
None

Pre-Condition
MACIsTxReady вызван и возвращан ИСТИНА.

Side Effects
None

Remarks
Эта функция просто загружает заголовок MAC в активный буфер передачи. Вызывающий оператор все еще должен загружать более данные и/или краску буфер для передачи. Вызовите MACFlush, чтобы вводить передачу текущего пакета.

Example

```
// Check to see if at least one transmit buffer is empty
if ( MACIsTxReady() )
{
        // Assemble IP packet with total IP packet size of 100 bytes
        // including IP header.
      MACPutHeader(&RemoteNodeMAC, MAC_IP, 100);
    ...

//******************************************************************
```

MACPut

Эта функция загружает данному байту данных в активный буфер передачи или приемника.

Syntax

```
void MACPut(BYTE val)
```

Parameters
val [in]
Байт Данных, который нужно записывать

Return Values
None

Pre-Condition
MACGetHeader, MACPutHeader, MACSetRxBuffer ИЛИ MACSetTxBuffermust вызван.

Side Effects
None

Remarks

Эта функция может быть использована, чтобы модифицировать или буфер передачи или буфер приемника какой угодно к настоящему времени активно.

Example

```
// Проверьте если по крайней мере один буфер передачи пустой если

if ( MACIsTxReady() )
{
// Соберите пакет IP с общим размером пакета IP 100 байтов
 // включая заголовок IP.
 MACPutHeader(&RemoteNodeMAC, MAC_IP, 100);

// Теперь помещенный фактические байты данных IP
        MACPut(0x55);
          ...


//*********************************************************************
```

MACPutArray

Эта функция записывает массив байтов данных в активный буфер передачи или приемника.

Syntax
void MACPutArray(BYTE *val, WORD len)

Parameters
val [in]
Байты Данных, которые нужно записывать

len [in]
Общее число байтов, чтобы записывать

Return Values
None

Pre-Condition

Side Effects
None

Remarks
None

Example

```
// Проверьте если по крайней мере один буфер передачи пустой
 if ( MACIsTxReady() )
 {
// Соберите пакет IP с общим размером пакета IP 100 байтов
 // включая заголовок IP.

// Теперь помещенный фактические байты данных IP
MACPut(0x55);
```

MACPutArray(data, count);

...

```
  //************************************************************************
```

MACFlush

Эта функция выделяет активный буфер передачи как готовый для передачи.

Syntax
void MACFlush()

Parameters
None

Return Values
None

Pre-Condition
MACPutHeader ИЛИ MACSetTxBuffer вызван.

Side Effects
None

Remarks
None

Example

```
// Проверьте если по крайней мере один буфер передачи пустой
if ( MACIsTxReady() )
{
// Соберите пакет IP с общим размером пакета IP 100 байтов
 // включая заголовок IP.
 MACPutHeader(&RemoteNodeMAC, MAC_IP, 100);

// Помещенный фактические байты данных IP
MACPut(0x55);
MACPutArray(data, count);
...

// Теперь передача это.

MACFlush();
 ...


//*****************************************************************************
```

MACDiscardTx
Эта функция отвергает буферное содержимое данной передачи и марок это как свободный.

Syntax
void MACDiscardTx(BUFFER buffer)

Parameters
buffer [in]
Буфер, чтобы быть отвергнут

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
Буферный идентификатор передачи, который был получен разговором MACGetTxBuffer должно быть использовано.

Example

```
// Проверьте если по крайней мере один буфер передачи пустой
 if ( MACIsTxReady() )
 {
// Соберите пакет IP с общим размером пакета IP 100 байтов
 // включая заголовок IP.

 // Получите текущий буфер передачи
buffer = MACGetTxBuffer();

// Резерв это.
MACReserveTxBuffer (Buffer);

// Помещенный фактические байты данных IP
 ...

// Теперь передача это.
 MACFlush();

// Больше не нужно этот буфер
MACDiscardTx(buffer);
 ...
```

//**********

MACSetRxBuffer

Эта функция устанавливает позицию доступа для активного буфера приемника.

Syntax
void MACSetRxBuffer(WORD offset)

Parameters
offset [in]
Позиция (что касается начала буфера) где следующий доступ должен происходить

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
Пользователи должны убедиться, что поставленное смещение не превосходит буферным содержимым текущего приемника. Если смещение выходят за границы текущий буфер приемника, весь последующий доступ к этому буферу должен давать неправильные данные.

Example

```
// Получите возможный пакет данных инфо.
 if ( MACGetHeader(&RemoteNode, &PacketType) )
 {
                 // A Пакет получен, обработан это.
                    actualCount = MACGetArray(data, count);
                   ...
                 // Данные Выборки, начинаемые в смещении 20

                MACSetRxBuffer(20);
                data = MACGet();
 ...
```

//**********

MACSetTxBuffer

Эта функция устанавливает позицию доступа для данного буфера передачи и делает этой передачей буферной активной.

Syntax
void MACSetTxBuffer(BUFFER buffer, WORD offset)

Parameters
buffer [in]
Буфер передачи где это смещение доступа приложено

offset [in]
Позиция (что касается начала буфера) где следующий доступ должен происходить

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
Пользователи должны убедиться, что поставленное смещение не превосходит буферным содержимым текущей передачи. Если смещение выходят за границы текущий буфер передачи, весь последующий доступ к этому буферу должен давать неправильные данные.

Example

```
// Проверьте если по крайней мере один буфер передачи пустой
if ( MACIsTxReady() )
 {
// Соберите пакет IP с общим размером пакета IP 100 байтов
 // включая заголовок IP.
MACPutHeader(&RemoteNodeMAC, MAC_IP, 100);

// Получите текущий буфер передачи
buffer = MACGetTxBuffer();

// Помещенный фактические байты данных IP
 ...

// Вычислите контрольную сумму пакета данных, которые передаются
 ...

// Теперь скорректируйте контрольную сумму в этом пакете.
// Для того, чтобы корректировать контрольную сумму, установившее буферный
доступ передачи к контрольной сумме
MACSetTxBuffer(buffer, checksumLocation);
 ...

// Теперь передача это.
MACFlush();
 ...
```

http://www.microchip.com/

//***

MACReserveTxBuffer

Эта функция резервирует данный буфер передачи и выделяет это как недоступно. Эта функция полезная для слоя TCP где сообщение должно быть поставлено в очередь пока оно правильно не будет подтверждено дистанционным хозяином.

Syntax
void MACReserveTxBuffer(BUFFER buffer)

Parameters
buffer [in]
Буфер передачи, чтобы резервироваться
Эта величина должна быть буферным идентификатором правильной передачи как возвращано функцией MACGetTxBuffer

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
None

Example

```
// Проверьте если по крайней мере один буфер передачи пустой
if ( MACIsTxReady() )
 {
// Пакет Передачи IP с общим размером пакета IP 100 байтов
// включая заголовок IP.
MACPutHeader(&RemoteNodeMAC, MAC_IP, 100);

// Получите текущий буфер передачи
buffer = MACGetTxBuffer();


// Резерв это, чтобы быть отвергнут когда ACK получен.
 MACReserveTxBuffer(buffer);
```

```
// Помещенный фактические байты данных IP
 ...

// Вычислите контрольную сумму пакета данных, которые передаются
...

// Теперь скорректируйте контрольную сумму в этом пакете.
// Для того, чтобы корректировать контрольную сумму, установившее буферный
доступ передачи к контрольной сумме
MACSetTxBuffer(buffer, checksumLocation);
 ...

// Теперь передача это.
 MACFlush();
...


//**********************************************************************
```

MACGetFreeRxSize
Эта функция возвращает буферный размер общего приемника доступный для будущих пакетов данных.

Syntax

```
WORD MACGetFreeRxSize()
```

Parameters
None

Return Values
Общее число байтов доступных для будущих пакетов данных.

Pre-Condition
None

Side Effects
None

Remarks
None

Example

```
// Получите буферному размеру доступный приемник
freeRxBuffer = MACGetFreeRxSize();
```

//**

MACGetRxBuffer
Это макро возвращает буферный идентификатор текущего приемника.

Syntax
BUFFER MACGetRxBuffer()

Parameters
None

Return Values
Буферный идентификатор Активного приемника.

Pre-Condition
None

Side Effects
None

Remarks
None

Example
```
// Получите возможный пакет данных инфо.
if ( MACGetHeader(&RemoteNode, &PacketType) )
{

// Получите активный буфер приемника id.
buffer = MACGetRxBuffer();

// Выберите контрольную сумму непосредственно не выбирая другие данные.
 MACSetRxBuffer(checkLocation);
checksum = MACGet();
 ...
```

//**

MACGetTxBuffer

Это макро возвращает буферный идентификатор текущей передачи.

Syntax
BUFFER MACGetTxBuffer()

http://www.microchip.com/

Parameters
None

Return Values
Буферный идентификатор Активной передачи.

Pre-Condition
None

Side Effects
None

Remarks
None

Example
```
// Если есть комната, загрузите новое сообщение.
if ( MACIsTxReady() )
 {
          // Заголовок Загрузки MAC
            MACPutHeader(&RemoteNode, MAC_ARP, 100);

          // Получите активный буфер передачи id.
            buffer = MACGetTxBuffer();

         // Модифицируйте байт данных #20 в буфер передачи
             MACSetTxBuffer(buffer, 20);
             MACPut(0x55);
       ...
```

```
//******************************************************************
//******************************************************************
```

Протокол Адреса Резолюции (ARP и ARPTask)

Слой ARP Микрокристалла TCP/IP Стека действительно осуществлен двумя модулями: ARP и ARPTask.
ARP реализован файлом ARP.c , создает примитивы ARP. ARPTask, реализованный файлом ARPTsk.c, использует примитивы и обеспечивает полные услуги ARP.

ARPTask реализован как кооперативная государственная машина, реагирующая на запросы ARP из дистанционного хозяина. Это также поддерживает одноуровневый

кеш, чтобы загружать ответ ARP и возврат на более высокий уровень когда подходящие вызовы сделаны. ARPTask не осуществляет механизм повторной попытки, так что верхние модули уровня или приложений должны обнаружить условия задержки и отвечать соответственно.

ARP ФУНКЦИИ:

//*

ARPIsTxReady

Это - макро, которое вызывает MACIsTxReady в свою очередь.

Syntax
BOOL ARPIsTxReady()

Parameters
None

Return Values
ИСТИНА: Если есть по крайней мере одна пустая буферная передача.
ЛОЖЬ: Если нет пустой буферной передачи.

Pre-Condition
None

Side Effects
None

Remarks
ARPTask действует в двух режимах: режим Сервера и режима Server/Client. В режиме Server/Client, часть кода приспособлена и компилирована, чтобы генерировать запросы ARP из локального хозяина себя. В режиме Сервера, код запроса ARP не компилирован. Обычно, если стек использован приложениями сервера (как например, Сервер HTTP или FTP serve)r, ARPTask должен быть компилирован в режим Сервера, чтобы уменьшать кодовый размер.

Example

```
// Если буфер передачи ARP готовый, передайте пакет ARP
if ( ARPIsTxReady() )
{

        // Пакет Передачи ARP.
      ARPPut(&RemoteNode, ARP_REPLY);
 ...
```

http://www.microchip.com/

//**

ARPGet

Эта функция выбирает полный пакет ARP и необходимую информацию возврата.

Syntax
BOOL ARPGet(NODE_INFO *remote, BYTE *opCode)

Parameters
дистанционный [out]

Дистанционная узловая информация как например, MAC и адресы IP

opCode [out]

КОД ARP

Возможные величины для этого параметра:

Значение	**Величины**
ARP_REPLY	пакета Ответа ARP получен
ARP_REQUEST	Запроса ARP получен
ARP_UNKNOWN	неизвестный пакет ARP получен

Return Values
ИСТИНА: Если был выбран правильный пакет ARP, который был адресован локальному хозяину; дистанционный и opCode содержит правильные величины.

ЛОЖЬ: Или неизвестный код ARP был выбран или этот пакет не был адресован локальному хозяину.

Pre-Condition
MACGetHeader уже вызван AND
Получен MAC пакет тип == MAC_ARP

Side Effects
None

Remarks
Эта функция допускает что активный буфер приемника содержит пакет ARP и имеет доступ к указателю к этому буферу - в начале пакета ARP. Обычно более высокие слои уровня должны проверять буфер MAC и вызывать эту функцию только если пакет ARP был обнаружен. Эта функция выбирает полный пакет ARP и выпускает активный буфер приемника.

Example

```
// Если пакет MAC получен...
if ( MACGetHeader(&RemoteNode, &PacketType) )
{

// Если это - пакет ARP, выберите это.

      If ( PacketType == MAC_ARP )

      {

         // Это - пакет ARP.

          ARPGet(&RemoteNode, &ARPCode);

...
```

```
 //************************************************************************
```

ARPPut

Эта функция загружает буфер MAC с правильным пакетом ARP.

Syntax
void ARPPut(NODE_INFO *remote, BYTE opCode)

Parameters
remote [in]

Дистанционная узловая информация как например, MAC и адресы IP

opCode [in]

КОД ARP

Возможные величины для этого параметра:

Значение	Величины
ARP_REPLY	Передает этот пакет как Ответ ARP
ARP ARP_REQUEST	Передавать этот пакет как Запрос ARP

Return Values
None

Pre-Condition
ARPIsTxReady == TRUE

Side Effects
None

Remarks
Эта функция собирает полный пакет ARP и передает это.

Example

```
// Проверьте если буфер передачи доступен

if ( ARPIsTxReady() )
{

// Передача это

ARPPut(&RemoteNode, ARP_REQUEST);
...
```

```
//****************************************************************************
//****************************************************************************
```

ARPTask ФУНКЦИОНИРУЕТ

```
//****************************************************************************
```

ARPInit

Эта функция инициализирует государственную машину ARPTask и подготавливает это, чтобы оперировать запросы ARP и ответы.

Syntax

void ARPInit()

Parameters
None

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
В режиме Server/Client, эта функция также инициализирует внутренний одноуровневый кеш.

Example

```
// Инициализируйте ARPTask

ARPInit();

...


//**********************************************************************
```

ARPResolve
Эта функция посылает запрос ARP дистанционному хозяину.

Syntax
void ARPResolve(IP_ADDR *IPAddr)

Parameters
IPAddr [in]
    IP адрес дистанционного хозяина, который нужно решаться

Return Values
None

Pre-Condition
ARPIsTxReady == TRUE

Side Effects
None

Remarks
Эта функция доступна когда STACK_CLIENT_MODE определен.

Example

```
// Проверьте если буфер передачи доступен
if ( ARPIsTxReady() )
{

        // Пошлите запрос ARP
          ARPResolve(&RemoteNodeIP);
...


//*********************************************************************
```

ARPIsResolved

Эта функция проверяет внутренний кеш и возвращает информацию главного адреса сопоставления.

Syntax
BOOL ARPIsResolved(IP_ADDR *IPAddr, MAC_ADDR *MACAddr)

Parameters
IPAddr [in]
IP адрес дистанционного хозяина, который должен быть решен

MACAddr [out]
Буфер, чтобы держать Адрес MAC дистанционного хозяина, который должен быть решен

Return Values
ИСТИНА: Если сопоставление Адреса IP было обнаружено во внутреннем кеше, соответствующем Адресу MAC скопирован на MACAddr.
ЛОЖЬ: Если нет сочетаясь Адрес IP во внутреннем кеше; MACAddr НЕ заполнен.

Pre-Condition
None

Side Effects
Вход, который сочетается с внутренним кешем удален из кеша и объявлен как решено.

Remarks

Поскольку ARPTask поддерживает только один уровень кеша, более высокий слой уровня должен убедиться, что запрос секунды ARP не послан пока предшествующий запрос не будет решен. Эта функция доступна когда STACK_CLIENT_MODE определен.

Example

```
// Проверьте если буфер передачи доступен

if ( ARPIsResolved(&RemoteIPAddr, &RemoteMACAddr) )

{
// ARP РЕШЕН. Продолжите дальнейшую связь...
...

}
else
{

// Не решенное. Ожидание...
...
}
```

//**********

Протокол Internet (IP)

Слой IP Микрокристалла TCP/IP Стека осуществлен файлом ***IP.c*** . Файл заголовка ***IP.h*** определяет услуги предусмотренные слоем.

В этой архитектуре, слой IP пассивный; это не реагирует на пакеты данных IP. Взамен, более высокие слои уровня используют примитивы IP и выбирают пакет IP, интерпретируют это и берет подходящее действие.
Спецификация IP требует, чтобы локальный хозяин генерировал уникальный идентификатор пакета для каждого пакета переданного этим. Идентификатор позволяет хозяина, чтобы идентифицировать двойные пакеты и отвергать их. Микрокристалл TCP/IP Stack s IP слоя поддерживает частную 16- битовую переменную, чтобы прослеживать идентификаторы пакета.

//**********

IPIsTxReady

Это - макро, которое вызывает MACIsTxReady в свою очередь.

Syntax
BOOL IPIsTxReady()

Parameters
None

Return Values
ИСТИНА: Если есть по крайней мере один передача буферная пустая.
ЛОЖЬ: Если нет пустого буфера передачи.

Pre-Condition
None

Side Effects
None

Remarks
Это макро предусмотрено, чтобы создавать абстракцию только. А не вызывая MACIsTxReady непосредственно, верхний слой, который использует услуги IP должно вызывать это макро.

Example

```
// Если буфер передачи IP готовый, передайте пакет IP


if ( IPIsTxReady() )

{
// Соберите пакет IP.
IPPutHeader(&Remote, MAC_TCP, IPPacketLen);

...


//******************************************************************************
```

IPSetTxBuffer
Это - макро, чтобы допускать более высокий слой уровня установивший указатель буферного доступа передачи. Это макро берет заголовок IP на счет перед разговором MACSetTxBuffer.

Syntax
void IPSetTxBuffer(BUFFER buffer, WORD offset)

Parameters
buffer [in]

Буферный идентификатор Передачи, чей указатель доступа должен быть установлен

offset [in]
Смещение что касается Данных IP

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
Слои, которые используют услуги IP должно вызывать это макро, чтобы устанавливать указатель доступа для данного буфера передачи. Это макро регулирует данное смещение длиной заголовка IP.

Example

```
// Если буфер передачи IP готовый, передайте пакет IP
if ( IPIsTxReady() )
{


// Соберите пакет IP.
IPPutHeader(&Remote, MAC_TCP, IPPacketLen);


// Получите текущий буфер передачи id.
buffer = MACGetTxBuffer();

// Данные передачи Загрузки
...

// Вычислите контрольную сумму checkHi:checkLo
...

// Коррекция контрольная сумма.
IPSetTxBuffer(buffer, checkLocation);
MACPut (checkHi);
MACPut (checkLo);
...


  //*****************************************************************
```

IPPutHeader
Эта функция собирает правильный заголовок IP и загружает это в активный буфер передачи.

Syntax
WORD IPPutHeader(NODE_INFO *remote, BYTE protocol, WORD len)

Parameters
remote [in]
Дистанционная узловая информация как например, MAC и адресы IP

protocol [in]
Протокол, чтобы использоваться для этого пакета данных

Возможные величины для этого параметра:

Значение	**Величины**
IP_PROT_ICMP	Собирает этот пакет как ICMP
IP_PROT_TCP	Собирает этот пакет как сегмент TCP
IP_PROT_UDP	Собирать этот пакет как сегмент UDP

len [in]

Общая длина байтов данных IP, исключающих заголовок IP

Return Values
None

Pre-Condition
IPIsTxReady == TRUE

Side Effects
None

Remarks
Эта функция собирается, пакет IP кончает с контрольной суммой заголовка и корректирует сетевой байтовый заказ. После этой функции, буферный доступ активного указателя передачи указывает на начало области данных IP.

Эта функция не вводит передачу пакета IP. Вызывающий оператор должен также данные загрузки IP вызывая функцию MACPut и/или разговор MACFlushto выделяют буфер как готовый передаться.

Example

```
// Проверьте если буфер передачи доступен
если ( IPIsTxReady() )
{
                    // Загрузка заголовок
        IPPutHeader(&RemoteNode, IP_PROT_ICMP, ICMP_HEADER_SIZE+dataLen);

        // Данные Загрузки ICMP
         IPPutArray(ICMPData, dataLen);

       // Марка это как готовый передан
         MACFlush();
...
```

```
//******************************************************************
```

IPPutArray
Это макро загружает массив байтов в активный буфер передачи.

Syntax
void IPPutArray(BYTE *buffer, WORD len)

Parameters
buffer [in]
Массив Данных, который - на загруженное

len [in]
Общее число пунктов в массиве данных

Return Values
None

Pre-Condition
IPIsTxReady == TRUE

Side Effects
None

Remarks
Это макро вызывает функцию MACPutArray. Это предусмотрен для абстракции только.

Example

```
// Проверьте если буфер передачи доступен
If ( IPIsTxReady() )
{
```

http://www.microchip.com/

```
            // Загрузка заголовок
          IPPutHeader(&RemoteNode, IP_PROT_ICMP, ICMP_HEADER_SIZE+dataLen);


           // Данные Загрузки ICMP
        IPPutArray(ICMPData, dataLen);


         // Марка это как готовый передан
          MACFlush();
...


//********************************************************************
```

IPGetHeader
Эта функция выбирает заголовок IP с активной передачи буферизовать и подтверждает это.

Syntax
BOOL IPGetHeader(IP_ADDR *localIP, NODE_INFO *remote, BYTE *protocol, WORD *len)

Parameters
localIP [out]
Локальная узловая информация как например, MAC и адресы IP

remote [out]
Дистанционная узловая информация как например, MAC и адресы IP

protocol [out]

Протокол связанный этим пакетом IP

Возможные величины для этого параметра:

Значение	**Величины**
IP_PROT_ICMP	Это - пакет ICMP
IP_PROT_TCP	Это - пакет TCP
IP_PROT_UDP	Это - пакет UDP
ICMP_ECHO_REQUEST	Неизвестные протоколы

len [out]

Общая длина данных IP в этом пакете

Return Values
ИСТИНА:Правильный пакет IP был получен. Дистанционный адрес IP, протокол пакета и параметров длины пакета заполнены.

ЛОЖЬ:Неправильный пакет IP был получен. Параметры не заполнены.

Pre-Condition
MACGetHeader == TRUE

Side Effects
None

Remarks
Эта функция допускает что буферный доступ активного указателя приемника спозиционирован в начало области Данных MAC.
Для того, чтобы удовлетворять это условие, более высокий слой уровня должен выполнить следующее чеков перед разговором этой функции:

If MACGetHeader == TRUE and PacketType == MAC_IP, call IPGetHeader

ELSE не вызывайте IPGetHeader

Как только пакет IP будет обработан и больше не нужно, вызывающий оператор должен отвергнуть это из MAC буферизует вызывая функцию **MACDiscardRx**. Посмотрите исходный код Менеджера Задачи Стека (**StackTsk.c**) фактической реализации.

IPGetHeader (Continued)

Example

```
// Проверьте если любой пакет готовый
If  ( MACGetHeader(&RemoteMACAddr, &PacketType) )
{

// Чек какой тип протокола это –
            If ( PacketType == MAC_IP )
{
          // Это - пакет IP. Выберите это.

IPGetHeader(&Local, &Remote, &IPProtocol, &IPLen);

// Процесс этот пакет IP.
...

// Когда сделано обработка этого пакета, отвергните это
```

```
MACDiscardRx();
}
else
{
// Это - не пакет IP. Прооперируйте этому
...
```

```
//******************************************************************
```

IPGetArray
Это макро выбирает массиву байтов из активного буфера передачи или приемника.

Syntax
WORD IPGetArray(BYTE *val, WORD len)

Parameters
val [out]
Указатель в буфер в байтовый массив

len [in]
Количество байтов, чтобы выбираться

Return Values
Общее число выбранных байтов

Pre-Condition
IPGetHeader, IPPutHeader, IPSetRxBuffer ИЛИ IPSetTxBuffer по-видимому вызван.

Side Effects
None

Remarks
Вызывающий оператор должен убедиться, что общее число байтов данных выбирало не превышает доступные данные в активном буфере. Это макро не проверяет на наличие буфера выходить за границы условия.

Example

```
// Проверьте если любой пакет готовый
If  ( MACGetHeader(&RemoteMACAddr, &PacketType) )
{
                // Чек какой тип протокола это
                  if ( PacketType == MAC_IP )
{
            // Это - пакет IP. Выберите это.
               IPGetHeader(&Remote, &IPProtocol, &IPLen);

           // Получите 20 байтов данных
```

```
            IPGetArray(IPData, 20);
              ...
            // Когда сделано обработка этого пакета, отвергните это
              MACDiscardRx();
}
else
{
// Это - не пакет IP. Прооперируйте этому
...
```

```
//******************************************************************
```

IPSetRxBuffer
Это макро позволяет более высокий слой уровня, чтобы устанавливать указатель буферного доступа приемника. Это берет заголовок IP на счет перед разговором MACSetRxBuffer.

Syntax
void IPSetRxBuffer(WORD offset)

Parameters
offset [in]
Смещение что касается Данных IP

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
Слои, которые используют услуги IP должно вызывать это макро, чтобы устанавливать указатель доступа для текущего буфера. Это макро регулирует данное смещение длиной заголовка IP.

Example

```
// Проверьте если любой пакет готовый
 if ( MACGetHeader(&RemoteMACAddr, &PacketType) )
{

        // Чек какой тип протокола это
```

```
            if ( PacketType == MAC_IP )
{
            // Это - пакет IP. Выберите это.
               IPGetHeader(&Remote, &IPProtocol, &IPLen);


            // Выберите 20-й байт в пределах данных IP.
               IPSetRxBuffer(20);
              data = MACGet();
                 ...

      // Когда сделано обработка этого пакета, отвергните это
       MACDiscardRx();
}
else
{
        // Это - не пакет IP. Прооперируйте этому
...
```

//**

Управляющий Протокол Сообщения Internet (ICMP)

Слой ICMP осуществлен файлом ICMP.c .
Файл заголовка ICMP.h определяет услуги предусмотренные слоем.

В этой архитектуре, слой ICMP является пассивным слоем; это не реагирует на пакет данных ICMP. Взамен, более высокие слои уровня используют примитивы ICMP и выбирают пакет ICMP, интерпретируют это и берет подходящее действие.

Нормально, ICMP использован, чтобы посылать и получать ошибку IP или диагностические сообщения. В Микрокристалле TCP/IP Стека, принадлежность ICMP примитивов ICMP, которая может быть использована, чтобы генерировать любое из сообщений ICMP. В вложенных приложениях, ICMP полезный для диагностической цели.
Когда приспособлено, ICMP может среагировать на пакеты “PING” , таким образом допуская дистанционного хозяина, чтобы определять локальное главное присутствие.

Слой Микрокристалла ICMP только реагирует на пакеты данных ping вплоть до 32 байтов; большие пакеты будут проигнорированы. Если необходимо должно оперировать большие пакеты, модифицируйте компилятор определять MAX_ICMP_DATA_LEN (в файле заголовка StackTsk.h).

//**

ICMPIsTxReady
Это макро определяется если по крайней мере один буфер передачи пустой.

Syntax
BOOL ICMPIsTxReady()

Parameters
None

Return Values
ИСТИНА: Если есть по крайней мере один передача буферная пустая.
ЛОЖЬ: Если нет пустого буфера передачи.

Pre-Condition
None

Side Effects
None

Remarks
Это макро предусмотрено, чтобы создавать абстракцию только. А не вызывая MACIsTxReady непосредственно, верхний слой, который использует услуги IP должно вызывать это макро.

Example

```
// Если буфер передачи IP готовый, передайте пакет IP
if ( ICMPIsTxReady() )
{
// Пакет Передачи ICMP.
...


//*********************************************************************
```

ICMPPut
Эта функция собирает правильный пакет ICMP и передает это.

Syntax

```
void ICMPPut( NODE_INFO *remote,
        ICMP_CODE code,
        BYTE *data,
        BYTE len,
        WORD id,
         WORD seq)
```

Parameters

remote [in]
Дистанционная узловая информация как например, MAC и адресы IP

code [in]
КОД ICMP, чтобы быть использованн для этого пакета ICMP

Возможные величины для этого параметра:

Значение	**Величины**
ICMP_ECHO_REPLY	Это - пакет ответа Эха
ICMP_ECHO_REQUEST	Это - пакет запроса Эха ICMP

data [in]
данные ICMP

len [in]
длина данных ICMP

id [in]
ICMP packet identifier

seq [in]
номер последовательности пакета ICMP

Return Values
None

Pre-Condition
IPIsTxReady == TRUE

Side Effects
None

Remarks
Эта функция спрашивает, чтобы слой IP собирал заголовок IP и собирает пакет ICMP, кончает с контрольной суммой заголовка и корректирует сетевой байтовый заказ. Один основное различие между этим и другим слоем функций Поместившим - то, что эта функция собирает и передает пакет в одну функцию. Вызывающий оператор поставляет полный буфер данных и нет должно поставлять данные как отдельный функциональный вызов.

Example

```
// Проверьте если буфер передачи доступен
If  ( ICMPIsTxReady() )
{
```

// Пакет Передачи ICMP.
ICMPPut(&RemoteNode, ICMP_ECHO_REPLY, data, datalen, id, seq);

// Сделанным. ICMP ПОМЕЩЕН в очередь передачи.
...

//**

ICMPGet
Эта функция выбирает заголовок ICMP с активной передачи буферизовать и подтверждает это.

Syntax
void ICMPGet(
NODE_INFO *remote,
ICMP_CODE *code,
BYTE *data,
BYTE *len,
WORD *id,
WORD *seq)

Parameters

remote [out]
Дистанционная узловая информация как например, MAC и адресы IP

code [out]
КОД ICMP для полученного пакета ICMP

Возможные величины для этого параметра:

Значение	**Величины**
ICMP_ECHO_REPLY	REPLY пакет ответа Эха ICMP получен
ICMP_ECHO_REQUEST	пакет запроса Эха ICMP получен
Все другие	Неизвестный/неподдерживаемый пакет получен

data [out]
данные ICMP

len [out]
длина данных ICMP

id [out]
идентификатор пакета ICMP

seq [out]

номер последовательности пакета ICMP

Return Values
ИСТИНА: правильный пакет ICMP был получен. Все параметры заполнены.
ЛОЖЬ: неправильный пакет ICMP был получен. Параметры не заполнены.

Pre-Condition
IPGetHeader == TRUE and PacketType == IP_PROT_ICMP

Side Effects
None

Remarks
Эта функция допускает что буферный доступ активного указателя приемника спозиционирован в начало области Данных IP. Для того, чтобы удовлетворять это условие, более высокий слой уровня должен выполнить следующее чеков перед разговором этой функции:

If IPGetHeader == TRUE and PacketType == IP_PROT_ICMP, call ICMPGet

Else

Не вызывайте ICMPGet

Как только пакет IP будет обработан и больше не нужно, вызывающий оператор должен отвергнуть это из MAC буферизует вызывая функцию MACDiscardRx.

ICMPGet (Continued)

Example

```
// Проверьте если любой пакет готовый

If  ( IPGetHeader(&Remote, &IPProtocol, &IPLen) )
{
            // Чек какой тип протокола это
          if ( IPProtocol == IP_PROT_ICMP )
{
       // Это - пакет ICMPP. Выберите это.
        ICMPGet(&ICMPCode, data, &dataLen, &id, &seq);

// Процесс этот пакет ICMP.
...

// Когда сделано обработка этого пакета, отвергните это
MACDiscardRx();
}
```

http://www.microchip.com/

```
else
{
// Это - не пакет ICMP. Прооперируйте этому
...


//*********************************************************************
```

Управляющий Протокол Передачи (TCP)

Слой TCP Микрокристалла TCP/IP Стека осуществлен файлом TCP.c . Файл заголовка TCP.h определяет услуги предусмотренные слоем. В этой архитектуре стека, TCP - активный слой. Это выбирает пакеты TCP и реагирует на дистанционный хозяина согласно государственной машине TCP. Модуль TCP также осуществлен как кооперативная задача, выполняющая автоматические операции без знания основного приложения.

TCP.h ОБЕСПЕЧИВАЕТ услуги socket TCP и прячет весь пакет TCP обрабатываясь из вызывающего оператора. Слой допускает от 2 до 253 socket TCP, число ограничивалось только доступной памятью и компилятор использовался. Более чем с одной socket, более высокие приложения уровня могут поддержать многочисленные одновременные связи TCP и могли быть более, чем одно приложение, использовавшее этот слой.
Это средство полезное когда Сервер HTTP использован. Важно должно узнать, что каждая socket поглотит приблизительно 36 байтов (контрольный исходный файл для фактического потребления) и комбинезон увеличения TCP обрабатывающего времени.

В отличие от других TCP/IP реализаций, все socket в Микрокристалле TCP/IP Складывают акцию один или более общих буферов передачи. Этот метод уменьшает общие требования RAM, но это может создать потенциальную проблему, где несколько розеток резервируют все доступные буферы передачи и не выпускают их вовремя для других socket, чтобы использоваться. При этих условиях, дистанционные хозяева и/или локальные приложения не быть способным обратиться к стеку. Для того, чтобы избегать это, пользователи должны убедиться, что есть достаточно буферов передачи для всех socket.

На стороне приемника, есть только один буфер приемника. Если socket получает свои данные, владелец этой socket должен выбрать и отвергать буфер приемника в одном времени задачи для того, чтобы другие socket, чтобы получать их данные. Эти проектные мандаты, что как только задача обнаружит пакет в котором она заинтересована, это должно поглотить полный пакет в одном времени задачи. Задача не может выбрать часть пакета в течение одного времени задачи и ожидать, что выбрать остальную часть пакета позже.

Как потребовалось спецификацией TCP, каждый сегмент TCP содержит контрольную сумму, которая покрывает целый пакет TCP, включая область данных. Для того, чтобы уменьшать требования RAM, слой TCP использует буфер MAC в NIC как

память и выполняет вычисление контрольной суммы в буфере MAC себе. Если NIC использован как MAC, NIC SRAM использован как буферное пространство. Но если SLIP использовано как MAC, microcontroller s внутреннее PAM данных использовано.

Слой TCP Микрокристалла TCP/IP Стека осуществляет наиболее государственный машинный состояния TCP предложившее RFC793. Это также осуществляет автоматическую повторную попытку и синхронизированные операции, какие пользователи могут приспособиться или выводиться из строя определением времени компиляции TCP_NO_WAIT_FOR_ACK.
Когда автоматическая повторная попытка приспособлена, каждый буфер передачи розетки зарезервирован пока подтверждение из дистанционного хозяина не будет получено. Этот проект эффективно создает окно передачи одного сегмента TCP. Таким образом, производительность данных должна быть значительно ниже, чем, что в режиме Без Повторной попытки. Если только приложение Сервера HTTP использовано, пользователь может вывести из строя автоматическую повторную попытку и эффективно увеличивать производительность. Если основная приложение логика требует, чтобы каждый пакет был признан прежде, чем новый может быть передан, пользователь должен приспосабливаться режим Автоматической Повторной попытки . С Автоматически
Повторная попытка приспособленная, немного открытые связи не могут получать served вовремя, и дистанционный хозяин может получить задержку или Восстановленные ошибки.

//**

TCPInit
Эта функция инициализирует государственную машину TCP и подготавливает это ко многочисленным связям TCP.

Syntax
```
void TCPInit()
```

Parameters
None

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
Эта функция вызывается только как только в прикладном пуске.

Example

http://www.microchip.com/

```
// Инициализируйте TCP
 TCPInit();


//********************************************************************
```

TCPListen
Эта функция назначает одну из доступных sockets, чтобы слушать в данном порту TCP.

Syntax
TCP_SOCKET TCPListen(TCP_PORT port)

Parameters
port [in]
НОМЕР Порта TCP к чему слушать

Return Values
Правильный идентификатор socket если было по крайней мере один свободная socket.
INVALID_SOCKET если нет socket доступной.

Pre-Condition
None

Side Effects
None

Remarks
None

Example

```
...


switch(smState)
 {
 case SM_LISTEN:

// Слушайте для запросов HTTP.

httpSocket = TCPListen(80);

if ( httpSocket == INVALID_SOCKET )

{
```

```
// socket не является доступной
 // ошибкой Возврата.
...


}
else
smState = SM_LISTEN_WAIT;

return;

case SM_LISTEN_WAIT:

// связь Ожидания...
...


//*********************************************************************
```

TCPConnect
Эта функция вводит запрос связи дистанционному хозяину в данном дистанционном порту.

Syntax
TCP_SOCKET TCPConnect(NODE_INFO *remote, TCP_PORT port)

Parameters
remote [in]
Дистанционный хозяин, который должен быть связан

port [in]
НОМЕР Порта TCP на дистанционном хозяине, чтобы соединять, чтобы

Return Values
Правильный идентификатор socket если было по крайней мере один свободная socket и запрос связи был послан.
INVALID_SOCKET если нет socket доступной.

Pre-Condition
None

Side Effects
None

Remarks
Эта функция доступна только когда STACK_CLIENT_MODE определен (смотри "Конфигурацию Стека" (страница 3)).

http://www.microchip.com/

Example

...

```
switch(smState)
{
 case SM_CONNECT:

// Подключите к дистанционному серверу FTP.

ftpSocket = TCPConnect(&RemoteNode, 21);

Если ( ftpSocket == INVALID_SOCKET )

{
// socket не является доступной
// ошибкой Возврата.

}
else
smState = SM_CONNECT_WAIT;
return;

case SM_CONNECT_WAIT:
        // связь Ожидания...
...
```

//**********

TCPIsConnected
Эта функция определяет подключена данная socket к дистанционному хозяину или нет.

Syntax
BOOL TCPIsConnected(TCP_SOCKET socket)

Parameters
socket [in]
Идентификатор socket для которого связь должна быть проверена

Return Values
ИСТИНА: Если данная розетка подключена к дистанционному хозяину.
ЛОЖЬ: Если данная розетка не подключена к дистанционному хозяину.

Pre-Condition

http://www.microchip.com/

None

Side Effects
None

Remarks
None

Example
...

```
switch(smState)
{
case SM_CONNECT:

// Подключите к дистанционному серверу FTP.
ftpSocket = TCPConnect(&RemoteNode, 21);

if ( ftpSocket == INVALID_SOCKET )
{
// Розетка не доступна
// Обратная ошибка.
}
else

smState = SM_CONNECT_WAIT;
return;

case SM_CONNECT_WAIT:
 // связь Ожидания...
```

if (TCPIsConnected(ftpSocket))

```
smState = SM_CONNECTED;
return
...
```

//**********

TCPDisconnect
Эта функция запрашивает дистанционного хозяина, чтобы разъединяться.

Syntax
void TCPDisconnect(TCP_SOCKET socket)

Parameters
socket [in]

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
None

Example

```
...

switch(smState)
{
case SM_CONNECT:

// Подключите к дистанционному серверу FTP.
ftpSocket = TCPConnect(&RemoteNode, 21);

if  ( ftpSocket == INVALID_SOCKET )
{

// Розетка не доступна
// Обратная ошибка.
}
else

smState = SM_CONNECT_WAIT;
return

case  SM_CONNECT_WAIT:
 // связь Ожидания...
If ( TCPIsConnected(ftpSocket) )

smState = SM_CONNECTED;
return

case  SM_CONNECTED:
// Посылать данные
...
// Разъединитесь

TCPDisconnect(ftpSocket);
```

http://www.microchip.com/

...

//**

TCPIsPutReady
Эта функция определяется если розетка готовая передаться. Розетка готовая передаться когда она подключена к дистанционному хозяину и буфер передачи пустой.

Syntax
BOOL TCPIsPutReady(TCP_SOCKET socket)

Parameters
socket [in]
Идентификатор Socket, который должен быть проверен

Return Values
ИСТИНА: Если данная розетка готовая передаться.
ЛОЖЬ: Если данная розетка не связана или нет буфера передачи готового.

Pre-Condition
None

Side Effects
None

Remarks
None

Example
...

```
switch(smState)
 {
case SM_CONNECT:

// Подключите к дистанционному серверу FTP.
ftpSocket = TCPConnect(&RemoteNode, 21);
if ( ftpSocket == INVALID_SOCKET )
{
// Розетка не доступна
// Обратная ошибка.
}
else

smState = SM_CONNECT_WAIT;
return
```

```
case SM_CONNECT_WAIT:
      // связь Ожидания...
if ( TCPIsConnected(ftpSocket) )

smState = SM_CONNECTED;
return

case SM_CONNECTED: // Посылать данные

if ( TCPIsPutReady(ftpSocket) )

{
// Пошлите данные
...


//*********************************************************************
```

TCPPut
Эта функция загружает байт данных в буфер передачи для данной розетки.

Syntax
BOOL TCPPut(TCP_SOCKET socket, BYTE byte)

Parameters

socket [in]
Идентификатор Socket, который должен быть проверен

byte [in]
Байт Данных, который нужно загружаться

Return Values
ИСТИНА: Если данный байт данных успешно был загружен в буфер передачи и есть место для более данных.

ЛОЖЬ: Если данный байт данных успешно был загружен в буфер передачи и нет места для более данных.

Pre-Condition
TCPIsPutReady == TRUE

Side Effects
None

Remarks

Как только оказывается, что Socket будет готовой передаться, пользователь должен загрузить все данные, или до нет более комнаты в буфере Socket. Пользователь не может загрузить часть данных в одну Socket и загружать другой буфер Socket.

Важно должно поминать это когда данные Socket загружены используя эту функцию, фактическая передача может или не может запускать, в зависимости от общего числа байтов данных загруженных. Если количество загруженных байтов - менее чем буферный размер доступная Socket, пользователь должен явно сбросить буфер передачи, использовавший функцию TCPFlush. Если пользователь пытается загружать больше байтов, тогда доступный буферный размер, эта функция автоматически начинает передачу и возвращает ЛОЖЬ, так что пользователь может пытаться снова. Обычно, это - хорошая практика, чтобы сбрасывать буфер Socket в конце концов узнавший, что данные загружены в буфер, независимо от того, что буфер был полным или нет.

TCPPut (НЕПРЕРЫВНЫЙ)

Example

```
...

switch(smState)
 {
case SM_CONNECT:

// Подключите к дистанционному серверу FTP.
ftpSocket = TCPConnect(&RemoteNode, 21);

if  ( ftpSocket == INVALID_SOCKET )
{
// Розетка не доступна
// Обратная ошибка.
}
else

        smState = SM_CONNECT_WAIT;
return;

case SM_CONNECT_WAIT:
 // связь Ожидания...

If  ( TCPIsConnected(ftpSocket) )


smState = SM_CONNECTED;
return;
```

```
case SM_CONNECTED:
 // Посылать данные

If  ( TCPIsPutReady(ftpSocket) )
{
// Пошлите данные
TCPPut(ftpSocket, dataByte);
...


//*********************************************************************
```

TCPFlush
Эта функция выделяет буфер передачи данной розетки как готовый передан.

Syntax
void TCPFlush(TCP_SOCKET socket)

Parameters
socket [in]
Идентификатор Розетки, что нужно на переданное

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
Эта функция выделяет текущий буфер передачи как готовый передаться; фактическая передача не может запускаться немедленно.
Пользователь нет должен проверять статус передачи. NIC Сделает новую попытку передавая сообщение вплоть до 15 раз (для RTL8019AS; проверьте документацию на наличие специфического NIC если RTL8019AS не используется). Если розетка уже сброшена, другая краска должна быть проигнорирована.

Example

```
...

switch(smState)
{
case SM_CONNECT:
```

```
// Подключите к дистанционному серверу FTP.
ftpSocket = TCPConnect(&RemoteNode, 21);

if  ( ftpSocket == INVALID_SOCKET )
{
          // Розетка не доступна
          // Обратная ошибка.
}
else

           smState = SM_CONNECT_WAIT;
            return;

case    SM_CONNECT_WAIT:
        // связь Ожидания...

         If ( TCPIsConnected(ftpSocket) )

        smState = SM_CONNECTED;
          return;

case   SM_CONNECTED:
                 // Посылать данные

               If  ( TCPIsPutReady(ftpSocket) )
              {
              // Пошлите данные
                TCPPut(ftpSocket, dataByte);
               ...

             // Теперь передача это.

               TCPFlush(ftpSocket);
                 ...


//*********************************************************************
```

TCPIsGetReady
Эта функция определяется если данная розетка содержит данные приемника.

Syntax
BOOL TCPIsGetReady(TCP_SOCKET socket)

Parameters
socket [in]

Return Values

ИСТИНА: Если данная розетка содержит данные приемника.
ЛОЖЬ: Если данная розетка не содержит любые данные.

Pre-Condition
None

Side Effects
None

Remarks
None

Example

```
...

switch(smState)
{
case SM_LISTEN:

                // Слушайте розетку HTTP
                  httpSocket = TCPListen(&RemoteNode, 80);
                 if ( httpSocket == INVALID_SOCKET )
                   {
                  // Розетка не доступна
                  // Обратная ошибка.
                   }
                  else

                  smState = SM_LISTEN_WAIT;
                  return;

case SM_LISTEN_WAIT:
               // связь Ожидания...
              If ( TCPIsConnected(httpSocket) )
                smState = SM_CONNECTED;
               return;

case SM_CONNECTED:
             // данные Выборки

              if ( TCPIsGetReady(httpSocket) )
               {
              // Данные Выборки
              ...


//*********************************************************************
```

TCPGet
Эта функция выбирает один байт данных из буфера приемника данной розетки.

Syntax
BOOL TCPGet(TCP_SOCKET socket, BYTE *byte)

Parameters
socket [in]
Идентификатор Розетки, который должен быть выбран

byte [out]
Байт Данных, который был прочитан

Return Values
ИСТИНА: Если байт был прочитан.
ЛОЖЬ: Если никакой байт не был прочитан.

Pre-Condition
TCPIsGetReady == TRUE

Side Effects
None

Remarks
Когда оказывается, что розетка будет содержать данные приемника, пользователь должен выбрать один или более байтов данных (если потребовалось) в одном времени задачи и было отвергнуто буфер розетки. Данные не могут быть выбраны из другой розетки пока буферное содержание розетки для сначала не будет отвергнуто.

Example

```
...
switch(smState)
 {
case SM_LISTEN:

                    // Слушайте розетку
                   HTTP httpSocket = TCPListen(&RemoteNode, 80);
                   if ( httpSocket == INVALID_SOCKET )
                  {
                   // Розетка не доступна
                   // Обратная ошибка.
                     }
                  else

                 smState = SM_LISTEN_WAIT;
                  return
```

http://www.microchip.com/

```
case  SM_LISTEN_WAIT:
                  // связь Ожидания...
                 if ( TCPIsConnected(httpSocket) )

                 smState = SM_CONNECTED;
                  return;

case SM_CONNECTED:
              // данные Выборки
              If   ( TCPIsGetReady(httpSocket) )
              {

             // Данные Выборки
               TCPGet(httpSocket, &dataByte);

               ...
```

```
 //*********************************************************************
```

TCPGetArray
Эта функция выбирает массив данных из буфера приемника данной розетки.

Syntax
```
WORD TCPGetArray(TCP_SOCKET socket,
                                      BYTE *byte,
                                     WORD count)
```

Parameters
socket [in]
Идентификатор Розетки, который должен быть выбран

byte [out]
Массив Данных, который был прочитан

count [out]
Общее число байтов, чтобы читаться

Return Values
Общее число байтов данных было прочитано.

Pre-Condition
TCPIsGetReady == TRUE

Side Effects
None

Remarks
Когда оказывается, что розетка будет содержать данные приемника, пользователь должен выбрать один или более байтов данных (если потребовалось) и было отвергнуто буфер розетки. Данные не могут быть выбраны из другой розетки пока буферное содержание розетки для сначала не будет отвергнуто.

Эта функция не проверяет запрос против доступных байтов данных в буфере приемника. Пользователь должен убедиться, что текущий буфер приемника не выхожен за границы.

TCPGetArray (Continued)

Example

```
...

switch(smState)
 {
case SM_LISTEN:

                  // Слушайте розетку HTTP
                 httpSocket = TCPListen(&RemoteNode, 80);
                if  ( httpSocket == INVALID_SOCKET )
{
// Розетка не доступна
// Обратная ошибка.
}
Else

                   smState = SM_LISTEN_WAIT;
                  return

case SM_LISTEN_WAIT:
               // связь Ожидания...
              If  ( TCPIsConnected(httpSocket) )
                     smState = SM_CONNECTED;
            return

case  SM_CONNECTED:
                // данные Выборки
              if ( TCPIsGetReady(httpSocket) )
                {
                     // Выборка 20 байтов данных
                     TCPGetArray(httpSocket, buffer, 20);
...


//*********************************************************************
```

http://www.microchip.com/

TCPDiscard
Эта функция выпускает буфер приемника связанный данной розеткой.

Syntax
BOOL TCPDiscard(TCP_SOCKET socket)

Parameters
socket [in]
Идентификатор Розетки, что нужно на переданное

Return Values
ИСТИНА: Если буфер приемника для данного успешно был отвергнут.
ЛОЖЬ: Если буфер приемника для данного буфера уже был отвергнут.

Pre-Condition
None

Side Effects
None

Remarks
None

Example

```
...

switch(smState)
{

 case SM_LISTEN:

                  // Слушайте розетку HTTP
              httpSocket = TCPListen(&RemoteNode, 80);
              if ( httpSocket == INVALID_SOCKET )
{
// Розетка не доступна
// Обратная ошибка.
}
else
                smState = SM_LISTEN_WAIT;
return


case   SM_LISTEN_WAIT:
           // связь Ожидания...

             if ( TCPIsConnected(httpSocket) )
              smState = SM_CONNECTED;
```

```
            return;


case SM_CONNECTED:
          // данные Выборки
            If ( TCPIsGetReady(httpSocket) )
             {

                // Выборка 20 байтов данных
                TCPGetArray(httpSocket, буфер, 20);

                 // Данные Процесса.
                 ...

               // Версия буфер.

                TCPDiscard(httpSocket);


...


//**********************************************************************
```

TCPProcess

Эта функция действует как TCPTask . Это выбирает уже полученный пакет TCP и выполняет Государственную машину TCP для сопоставления розеток. Эта функция должна вызываться только когда пакет TCP получен.

Syntax
BOOL TCPProcess(NODE_INFO *remote, WORD len)

Parameters
remote [in]
Дистанционный узел из которого текущий пакет TCP был получен

len [out]
Общая длина длины пакета TCP, включая заголовок TCP

Return Values
ИСТИНА: Если эта функция (задача) полностью обработала бы текущий пакет.
ЛОЖЬ: Если эта функция (задача) частично обработала бы текущий пакет.

Pre-Condition
IPGetHeader == TRUE and IPProtocol = IP_PRO_TCP

Side Effects

http://www.microchip.com/

None

Remarks
Как упомянуто выше, эта функция осуществляет государственную машину TCP. Пользователи должны вызвать эту функцию когда правильный пакет данных TCP получен. Как только пакет будет обнаружен, эта функция выбирает и оперирует пакет. Обратная величина из этой функции указывается вызывающему оператору если государственное машинное состояние StackTask должно быть изменено или нет.

В своей текущей реализации, эта функция всегда возвращает ИСТИНУ. Посмотрите задачу Менеджера исходного кода Стека (StackTsk.c) в фактической реализации.

Example

```
...

Swith (smState)
{
Case  SM_STACK_IDLE:

          If  ( MACGetHeader(&RemoveMAC, &MACFrameType) )
         {
                    if ( MACFrameType == MAC_IP )
                              smState = SM_STACK_IP;
                  ...
              return;

case   SM_STACK_IP:
                  if ( IPGetHeader(&RemoteNode, &IPFrameType, &IPDataCount) )
                  {
                            If ( IPFrameType == IP_PROT_TCP )
                                      smState = SM_STACK_TCP;
                       ...
                         return;

case SM_STACK_TCP:
                 if ( TCPProcess(&RemoteNode, IPDataCount) )
                           smState = SM_STACK_IDLE;
                             return;
                             ...


 //*********************************************************************
```

http://www.microchip.com/

TCPTick
Эта функция действует как другое TCPTask дополнительно к TCPProcess. Эта функция проверяет на наличие условий задержки для всех розеток и пытается восстанавливаться из них.

Syntax
void TCPTick()

Parameters
None

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
Эта функция осуществляет обработку задержки. Пользователь должен вызвать эту функцию периодически, чтобы гарантировать, который задерживает условия прооперированы в своевременном способе.

Example

```
TCPTick();
```

```
//******************************************************************
```

Протокол Дейтаграммы Пользователя (UDP)

Слой UDP Микрокристалла TCP/IP Стека осуществлен файлом UDP.c . Файл заголовка UDP.h определяет услуги предусмотренные слоем. В этой архитектуре стека, UDP - активный слой. Это выбирает пакеты UDP и уведомляет соответствовать розетки UDP о прибытии данных или передачи. Модуль UDP осуществлен как кооперативная задача, выполняющая автоматические операции без знания основного приложения.

UDP.h ОБЕСПЕЧИВАЕТ услуги розетки UDP и прячет весь пакет UDP обрабатываясь из вызывающего оператора. Слой допускает вплоть до 254 розеток UDP (число ограничивалось только доступной памятью и компилятор использовал). Более чем с одной розеткой, более высокие приложения уровня могут поддержать

многочисленные одновременные связи UDP; более, чем одно приложение могло бы использовать этот слой. Важно должно узнать, что каждая розетка поглотит приблизительно 19 байтов (чек файла UDP.h для фактического потребления) и комбинезон увеличения UDP обрабатывающего времени.

В отличие от других реализаций розетки, все розетки в Микрокристалле TCP/IP Складывают акцию один или более общих буферов передачи. Этот метод уменьшает общие требования PAM, но это может создать потенциальную проблему, где несколько розеток резервируют все доступные буферы передачи и не выпускают их вовремя для других розеток, чтобы использоваться. При этих условиях, дистанционные хозяева и/или локальные приложения не быть способным обратиться к стеку. Для того, чтобы избегать это, пользователи должны убедиться, что есть достаточно буферов передачи для всех розеток.

На стороне приемника, есть только один буфер приемника. Если розетка получает свои данные, владелец этой розетки должен выбрать и отвергать буфер приемника в одном времени задачи для того, чтобы другие розетки, чтобы получать их данные. Эти проектые мандаты, что как только задача обнаружит пакет в котором она заинтересованна, это должно поглотить полный пакет в одном времени задачи. Задача не может выбрать часть пакета в течение одного времени задачи и ожидать, что выбрать остальную часть пакета позже.

Спецификация UDP не делает мандатом, который вычисление контрольной суммы выполнено в пакетах UDP. Чтобы уменьшать общие программные и требования памяти данных, Микрокристалл TCP/IP Стека не осуществляет вычисление контрольной суммы UDP; взамен, это устанавливает области контрольной суммы в нуль, чтобы указывать, что никакое вычисление контрольной суммы не выполнено. Это проектое решение требует, что все модули, использовавшие модуль UDP гарантируют их целостность данных.

//**

UDPInit
Эта функция инициализирует модуль UDP и подготавливает это ко многочисленным связям UDP.

Syntax
void UDPInit()

Parameters
None

Return Values
None

Pre-Condition
None

http://www.microchip.com/

Side Effects
None

Remarks
Эта функция вызывается только как только в прикладном пуске.

Example

```
// Инициализируйте UDP
UDPInit();


//*********************************************************************
```

UDPOpen
Эта функция подготавливает следующую доступную розетку UDP в данном порту для возможной передачи данных. Или локальный или дистанционный узел может ввести передачу данных.

Syntax
UDP_SOCKET UDPOpen(UDP_PORT localPort, NODE_INFO *remoteNode, TCP_PORT remotePort)

Parameters

localPort [in]
Локальный номер порта UDP на котором передача данных произойдет

remoteNode [in]
Дистанционный хозяин, который содержит remotePort

remotePort [in]
НОМЕР Порта UDP на дистанционном хозяине, чтобы передавать данные на и из

Return Values
Правильный идентификатор розетки если было по крайней мере один свободная розетка.
INVALID_UDP_SOCKET если нет розетки доступной.

Pre-Condition
None

Side Effects
None

Remarks
Эта функция может быть использована как для локальных введенных главных так и дистанционных введенных главных передач данных. Нет явной связи входить UDP.

Как только розетка UDP будет открыта, она готовая передать или получить данные. Когда говорят, что розетка будет открыта, она просто означает, что соответствующая розетка является установкой, чтобы получать и передавать один или более пакетов данных пока эта розетка не будет закрыта.

Example

```
...

Swith (smState)
{
case  SM_OPEN:

                // Разговор с дистанционным сервером DHCP.

                DHCPSocket = UDPOpen(68, &DHCPServerNode, 67);

                if ( DHCPSocket == INVALID_UDP_SOCKET )

{
                    // Розетка не является доступной
                     // ошибкой Возврата.
}

Else

// Транслируйте сообщение Передачи DHCP.
break;
...


//***********************************************************************
```

UDPClose

Эта функция закрывает данную розетку UDP и объявляет этому как свободная розетка.

Syntax

void UDPDisconnect(UDP_SOCKET socket)

Parameters

socket [in]
Идентификатор розетки, которая должна быть закрыта

Return Values

None

http://www.microchip.com/

Pre-Condition
None

Side Effects
None

Remarks
None

Example

```
Swith (smState)
{
case SM_OPEN:
                // Разговор с дистанционным сервером DHCP.
                 DHCPSocket = UDPOpen(68, &DHCPServerNode, 67);
               if ( DHCPSocket == INVALID_UDP_SOCKET )
{
                    // Розетка не доступна
                    // Обратная ошибка.
}
else
                  // Пошлите запрос DHCP...
                 ...
                // Закрытие розетка
```

UDPClose(DHCPSocket);

```
break;
...

//************************************************************************
```

UDPIsPutReady
Это макро определяется если данная розетка готовая передаться. Розетка готовая передаться когда по крайней мере один из буферов передачи MAC пустой. Это также устанавливает данную розетку как активную розетку UDP.

Syntax

```
BOOL UDPIsPutReady(UDP_SOCKET socket)
```

Parameters
socket [in]
Идентификатор розетки, которая должна быть проверена и установлена активным

Return Values

ИСТИНА: Если данная розетка готовая передаться.
ЛОЖЬ: Если нет буфера передачи готового.

Pre-Condition
None

Side Effects
None

Remarks
С тех пор как UDP - связь менее протокол, розетка всегда готовая передать так же долго (длиной) как по крайней мере буфер передачи MAC пустое. Дополнительно к проверке на готовность передатчика, UPDIsPutReady также устанавливает активную розетку UDP. Как только розетка установлена активные, последующие вызовы в другие функции UDP (как например, UDPGet, UDPPut, UDPFlush и UDPDiscard), используют активную розетку как текущую розетку. Следовательно, нет необходимости для пользователя, чтобы передавать розетку на каждый вызов.

Смотри UDPGet, UDPPut, UDPFlush и функции UDPDiscard для дополнительной информации.

Example

```
...

Swith (smState)
{
case  SM_OPEN:

                        // Разговор с дистанционным сервером DHCP.
                      DHCPSocket = UDPOpen(68, &DHCPServerNode, 67);
                     if ( DHCPSocket == INVALID_UDP_SOCKET )
{
                    // Розетка не доступна
                    // Обратная ошибка.
}
else

                 // Транслируйте сообщение Передачи DHCP.
                   smState = SM_BROADCAST;
            break;

case SM_BROADCAST:

      if ( UDPIsPutReady(DHCPSocket) )
{
             // Розетка готовая передаться. Передайте данные...
```

http://www.microchip.com/

...
}
break;
...

//***

UDPPut
Эта функция загружает байт данных в буфер передачи для активной розетки.

Syntax
BOOL UDPPut(BYTE byte)

Parameters
byte [in]
Байт Данных, который нужно загружаться

Return Values

ИСТИНА: Если данный байт данных успешно был загружен в буфер передачи и есть место для более данных.
ЛОЖЬ: Если данный байт данных успешно был загружен в буфер передачи и нет места для более данных.

Pre-Condition
UDPIsPutReady == TRUE

Side Effects
None

Remarks
Как только оказывается, что розетка будет готовой передаться, пользователь должен загрузить все данные, или до нет более комнаты в буфере розетки. Пользователь не может загрузить часть данных в одну розетку и разделяться в другой буфер розетки.

Важно должно поминать это когда данные розетки загружены используя эту функцию, фактическая передача может или не может запускать, в зависимости от общего числа байтов данных загруженных. Если количество загруженных байтов - менее чем буферный размер доступная розетка, пользователь должен явно сбросить буфер передачи, использовавший функцию UDPFlush. Если пользователь пытается загружать больше байтов, тогда доступный буферный размер, эта функция автоматически начинает передачу и возвращает ЛОЖЬ, так что пользователь может пытаться снова. Обычно, это - хорошая практика, чтобы сбрасывать буфер розетки в конце концов узнавший, что данные загружены в буфер, независимо от того, что буфер был полным или нет.

UDPPut (Continued)

http://www.microchip.com/

Example

```
...

swith (smState)
{

case SM_OPEN:

                        // Разговор с дистанционным сервером DHCP.
                        DHCPSocket = UDPOpen(68, &DHCPServerNode, 67);
                        If  ( DHCPSocket == INVALID_UDP_SOCKET )
{
// Розетка не доступна
// Обратная ошибка.
}
else

                  // Транслируйте сообщение Передачи DHCP.
                   smState = SM_BROADCAST;
break;

case SM_BROADCAST:
                   if ( UDPIsPutReady(DHCPSocket) )
{
                  // Розетка готовая передаться. Передайте данные...
                  // Отметьтесь, что есть параметр DHCPSocket в UDPPut.
                  // Этот вызов UDPPut использует активную розетку
                  // как установлено UDPIsPutReady() -, что - DHCPSocket.

                 UDPPut(0x55);
                ...
}
break;
...

//***********************************************************************
```

UDPFlush
Эта функция выделяет буфер передачи активной розетки как готовый передан.

Syntax
void UDPFlush()

Parameters

Return Values
None

Pre-Condition
UDPPut(), уже вызван, и желаемая розетка UDP установлена как активная розетка вызывая UDPIsPutReady().

Side Effects
None

Remarks
Эта функция выделяет текущий буфер передачи как готовый передаться; фактическая передача не может запускаться немедленно.
Пользователь нет должен проверять статус передачи. NIC Сделает новую попытку передачей вплоть до 15 раз (для RTL8019AS; проверьте документацию на наличие специфического NIC если RTL8019AS не используется). Если розетка уже сброшена, другая краска должна быть проигнорирована.

Example

```
...

swith (smState)
{

   case SM_OPEN:
                    // Разговор с дистанционным сервером DHCP.
                   DHCPSocket = UDPOpen(68, &DHCPServerNode, 67);

                   if ( DHCPSocket == INVALID_UDP_SOCKET )
{
                   // Розетка не доступна
                   // Обратная ошибка.
}
                  else

                   // Транслируйте сообщение Передачи DHCP.
                   smState = SM_BROADCAST;
break;

case SM_BROADCAST:
                  if ( UDPIsPutReady(DHCPSocket) )
                {
                    // Розетка готовая передаться. Передайте данные...
                    // Отметьтесь, что есть параметр DHCPSocket в UDPPut.
                    // Этот вызов UDPPut использует активную розетку
                    // как установлено  UDPIsPutReady() -, что - DHCPSocket.
```

```
            UDPPut(0x55);
                ...
             // Теперь передача это.

             UDPFlush();

        }
          break;
        ...
```

```
//*********************************************************************
```

UDPIsGetReady
Эта функция определяется если данная розетка содержит данные приемника. Это также устанавливает данную розетку как активную розетку.

Syntax
BOOL UDPIsGetReady(UDP_SOCKET socket)

Parameters
socket [in]
Идентификатор для розетки, что нужно на переданное и устанавливать активный

Return Values
ИСТИНА: Если данная розетка содержит данные приемника.
ЛОЖЬ: Если данная розетка не содержит любые данные.

Pre-Condition
UDPOpen(), уже вызван. Величина розетки должна быть, что возвращался UDPOpen вызова().

Side Effects
None

Remarks
None

Example

```
...
swith(smState)
{
case SM_OPEN:

                    // Разговор с дистанционным сервером DHCP.
                    DHCPSocket = UDPOpen(68, &DHCPServerNode, 67);
                    if ( DHCPSocket == INVALID_UDP_SOCKET )
```

```
                {
                   // Розетка не доступна
                   // Обратная ошибка.
                }
                     else

                  // Подождите ответ из сервера DHCP
                  smState = SM_WAIT_FOR_DATA;
break;

case SM_WAIT_FOR_DATA:

                  if ( UDPIsGetReady(DHCPSocket) )

{
        // Розетка содержит некоторые данные. Выберите это и обрабатывайте это.
          ...
}
break;
...


//*********************************************************************
```

UDPGet
Эта функция выбирает один байт данных из буфера приемника активной розетки.

Syntax
BOOL UDPGet(BYTE *byte)

Parameters
byte [out]
Байт Данных, который был прочитан

Return Values
ИСТИНА: Если байт был прочитан.
ЛОЖЬ: Если никакой байт не был прочитан.

Pre-Condition
UDPIsGetReady == TRUE

Side Effects
None

Remarks
Когда оказывается, что розетка будет содержать данные приемника, пользователь должен выбрать один или более байтов данных (если потребовалось) в одном

времени задачи и было отвергнуто буфер розетки. Данные не могут быть выбраны из другой розетки пока буферное содержание розетки для сначала не будет отвергнуто.

Example

```
...

Swith (smState)
{
case  SM_OPEN:

                // Разговор с дистанционным сервером DHCP.
                  DHCPSocket = UDPOpen(68, &DHCPServerNode, 67);
                  If  ( DHCPSocket == INVALID_UDP_SOCKET )
{
                 // Розетка не доступна
                 // Обратная ошибка.
}
                   else

                    // Подождите ответ из сервера DHCP
                    smState = SM_WAIT_FOR_DATA;
break;

case  SM_WAIT_FOR_DATA:
                  if ( UDPIsGetReady(DHCPSocket) )
          {
                // Розетка содержит некоторые данные. Выберите это все.
                 // буфер является указателем в БАЙТ.

                     while( UDPGet(buffer) )
                       buffer++;
                    // Процесс это.
                      ...
                     // Отвергните буфер розетки.
                        ...
             }
            break;
...


//*********************************************************************
```

UDPDiscard
Эта функция выпускает буфер приемника связанный активной розеткой.

Syntax
BOOL UDPDiscard()

Parameters
None

Return Values
ИСТИНА: Если буфер приемника успешно был отвергнут.
ЛОЖЬ: Если буфер приемника уже был отвергнут.

Pre-Condition
None

Side Effects
None

Remarks
None

Example
...

```
swith(smState)
{
case  SM_OPEN:

                // Разговор с дистанционным сервером DHCP.
                DHCPSocket = UDPOpen(68, &DHCPServerNode, 67);

               If ( DHCPSocket == INVALID_UDP_SOCKET )
        {
                    // Розетка не доступна
                    // Обратная ошибка.
        }
else


                  // Подождите ответ из сервера DHCP
                  smState = SM_WAIT_FOR_DATA;
break;

case  SM_WAIT_FOR_DATA:

                    if ( UDPIsGetReady(DHCPSocket) )
                {
                    // Розетка содержит некоторые данные. Выберите это все.
                    // буфер является указателем в БАЙТ.

                  while( UDPGet(buffer) )
```

```
                buffer++;
            // Процесс it..
            ...
            // Отвергните буфер розетки.

            UDPDiscard();

}
break;
...

//*********************************************************************
```

UDPProcess
Эта функция действует как UDPTask . Это выбирает уже полученный пакет UDP и назначает это в сопоставление розетки UDP.
Эта функция должна вызываться только когда пакет UDP получен.

Syntax
BOOL UDPProcess(NODE_INFO *remote, WORD len)

Parameters
remote [in]
Дистанционный узел из которого текущий пакет UDP был получен

len [in]
Общая длина длины пакета UDP, включая заголовок UDP

Return Values
ИСТИНА: Если эта функция (задача) полностью обработала бы текущий пакет.
ЛОЖЬ: Если эта функция (задача) частично обработала бы текущий пакет.

Pre-Condition
IPGetHeader == TRUE and IPProtocol = IP_PRO_UDP

Side Effects
None

Remarks
Как упомянуто выше, эта функция осуществляет задачу UDP. Пользователи должны вызвать эту функцию когда правильный пакет данных UDP получен. Как только пакет будет обнаружен, эта функция выбирает и оперирует пакет. Обратная величина из этой функции указывается вызывающему оператору если государственное машинное состояние StackTask должно быть изменено или нет. В своей текущей реализации, эта функция всегда возвращает ИСТИНУ.

http://www.microchip.com/

Посмотрите задачу Менеджера исходного кода Стека (StackTsk.c) в фактической реализации.

Example

```
...

Swith (smState)
{
case  SM_STACK_IDLE:

            if ( MACGetHeader(&RemoveMAC, &MACFrameType) )
{
                    if ( MACFrameType == MAC_IP )
                      smState = SM_STACK_IP;
                  ...
                 return;


case SM_STACK_IP:
              if  ( IPGetHeader(&RemoteNode, &IPFrameType, &IPDataCount) )
{
                                  if ( IPFrameType == IP_PROT_UDP )
                                    smState = SM_STACK_UDP;
                ...
            return;

case SM_STACK_UDP:

            if ( UDPProcess(&RemoteNode, IPDataCount) )
              smState = SM_STACK_IDLE;
return;
...


//*********************************************************************
```

Динамический Главный Протокол Конфигурации (DHCP)

Слой DHCP Микрокристалла TCP/IP Стека осуществлен файлом dhcp.c . Файл заголовка dhcp.h определяет услуги предусмотренные слоем.
DHCP - активный слой, который транслирует запросы DHCP и автоматически получает и декодирует ответы DHCP. Основные характеристики включают:

Конфигурация IP и межсетевой маски адресов и подсети
Автоматически время аренды DHCP и управление восстановления аренды
Полностью автоматическая операция без прикладного вмешательства пользователя

Модуль DHCP осуществлен как кооперативная задача, выполняющая автоматические операции без знания основного приложения. Фактическая интеграция DHCP и управление сделаны посредством Менеджера Стека; это оперирует все необходимые операции как часть своей стандартной задачи, использовавшей DHCP APIs, чтобы управлять module s поведением. Пользователь не нужно знать о DHCP для того, чтобы использовать это.

Приложение пользователя может также решить вызывать некоторые APIs, чтобы непосредственно управлять операцией DHCP, как например, независимо DHCP сконфигурирован или нет, и навсегда останавливать DHCP. Нормально, user s приложение не должно нужно непосредственно взаимодействовать с DHCP совсем.

Для того, чтобы использовать модуль DHCP, файлы проекта пользователя должны модифицироваться следующим образом:

1. Uncomment STACK_USE_DHCP В файле заголовка StackTsk.h .

2. Включите dhcp.c и файлы udp.c в проект.

3. Увеличьте MAX_UDP_SOCKETS одним (по крайней мере один розетка UDP должна быть доступна для DCHP; отрегулируйте количества розеток основанных на UDP и DCHP нужно).

Когда DHCP осуществлен, приложение пользователя не должно пытаться сетевая связь пока DHCP не будет сконфигурироваться правильно. Нормально, если приложение пользователя содержит одно или более приложений клиента, которые требуют связь на включении питания или Восстанавливают, приложение должно проверить, что DHCP сконфигурирован перед передачей любых данных, использовавших более низкие модули слоя.
Это может быть удовлетвориться функцией DHCPIsBound (страница 75).

Официальная спецификация DHCP (RFC1541) требует клиента DHCP, чтобы заменять свою конфигурацию IP прежде, чем время аренды истечет. Для того, чтобы прослеживать время аренды, приложение пользователя должно убедиться, что TickUpdate(), вызван как потребовалось, и, что разумная точность времени поддержана (посмотрите исходный кодовый файл websrvr.c для рабочего примера). Необходимое разрешение времени - 15 минут, плюс или минус, который означает, что TickUpdate(), может быть вызвано в том же низком приоритете.

Для модуля DHCP, чтобы автоматически конфигурировать адресы и маска подсети, должна быть по крайней мере один сервер DHCP в сети. Это - user s ответственность, чтобы осуществлять некоторый метод для издания конфигурации потенциальным пользователям. Опции колеблются чтобы вручную читать информацию с дисплея в каждом узле, на хранение информации в центральном сервере. DHCP КОРРЕКТИРУЕТ величины MY_IP_BYTE?, MY_GATE_BYTE? и MY_MASK_BYTE? в результате автоматической конфигурации.

http://www.microchip.com/

//***

DHCPTask
Это - основная функция задачи DHCP, которая выполняет необходимые действия протокола.

Syntax
void DHCPTask()

Parameters
None

Return Values
None

Pre-Condition
None

Side Effects
None

Remarks
Эта функция должна вызываться периодически.

Example

```
// Введите задачу DHCP
DHCPTask();
```

//***

DHCPDisable
Это макро выводит из строя модуль DHCP, даже прежде, чем задача будет введена.

Syntax
void DHCPDisable()

Parameters
None

Return Values
None

Pre-Condition
Должно быть вызвано прежде, чем DHCPTask будет вызван.

Side Effects
None

Remarks
Эта функция полезная если приложение требует опцию, чтобы выводить из строя режим DHCP как часть своей конфигурации. Нормально, если DHCP не требуется совсем, это не должно включано в проект. Но в случае, если приложение требует время выполнения выбора DHCP, эта функция может быть использована, чтобы выводить из строя DHCP. Микрокристалл TCP/IP Демонстрационного приложения Стека использует эту функцию, чтобы демонстрировать время выполнения блокировки модуля DHCP.

Эта функция должна быть вызвана прежде, чем функция DHCPTask будет вызвана. Как только DHCPTaskis будет вызван, используйте DHCPAbort (страница 74), чтобы выводить из строя DHCP. Если DHCPDisable вызван после того, как DHCPTask был выполнен, розетка DHCP UDP станет сиротой. Любые новые ответы DHCP полученные этим узлом будут загружены в розетку DHCP UDP, но это не будет извлечено любым приложением; в конечном счете, никакое приложение или слой не будет способным получить данные совсем.

Example

```
void main()
{
        // Инициализируйте приложение и стек
       ...
         StackInit();

  // Если потребовалось выводить из строя DHCP здесь.
        If  ( DHCPIsDisabled )

            DHCPDisable();

     // Теперь ввод в основной цикл.
    while(1)
            {
             // Выполните необходимые шаги.
              ...
            }
}


 //***********************************************************************
```

DHCPAbort
Эта функция прерывает уже активную задачу DHCP и выводит из строя это для жизни приложения.

Syntax
void DHCPAbort()

Parameters
None

Return Values
None

Pre-Condition
Должно быть вызвано после DHCPTaskis вызванный по крайней мере, один раз.

Side Effects
None

Remarks
Эта функция может быть использована, чтобы выводить из строя или прерывать DHCP после того, как DHCPTask будет вызван. Важно должно отмечаться, что, как только DHCPTask будет запущен, узел может уже получить конфигурацию DHCP. Если DHCP прерван после получать конфигурации IP, конфигурация не автоматически будет заменена. Невозможнось заменять конфигурацию может закончиться узлом не в состоянии связаться в сети. Это может также закончиться тем же адресом IP, назначанным в другой узел, вызывающим ненадежную связь. Стек Микрокристалла не обеспечивает любую обратную связь относительно достоверности конфигурации IP.

Example

```
void main() {

            // Инициализируйте приложение и стек

          ...
          StackInit();

      // Теперь ввод в основной цикл.
while(1)
            {
    // Выполните необходимые шаги.
       ...
    // После некоторой итерации, приложению нужно DHCP прерванный.

      DHCPAbort();

            }
}
```

http://www.microchip.com/

//**

DHCPIsBound
Это макро проверяется если модуль DHCP получил бы конфигурацию IP (связан).

Syntax
BOOL DHCPIsBound()

Parameters
None

Return Values
ИСТИНА: Если DHCP обязанный конфигурация IP.
ЛОЖЬ: Если DHCP еще не связан.

Pre-Condition
None

Side Effects
None

Remarks
Используйте эту функцию, чтобы определяться если DHCP связан или нет. Эта функция полезная когда приложение пользователя состоит из одного или более приложений клиента, что нужно связываться на включении питания или Восстанавливаться. Перед попыткой связываться, приложение пользователя должно ожидаться пока DHCP не будет обязанным конфигурация IP.

Когда DHCP обязанный конфигурация IP, соответствующие величины MY_IP_BYTE?, MY_GATE_BYTE? и MY_MASK_BYTE? скорректированы.

Example

```
void main()
                {
                   // Инициализируйте приложение и стек
                    ...
            StackInit();

          // Теперь ввод в основной цикл.
       while(1)
    {
           // Выполните необходимые шаги.
           ...
           // Проверьте если DHCP связан. Если да, отобразите адрес IP

          if ( DHCPIsBound() )
                          {
                // Отобразите MY_IP_BYTE? величина.
```

http://www.microchip.com/

```
                DisplayIPAddress();
                        }
                }
        }
```

//**

IP, ОТБИРАЮЩИЙСЯ тщательно для Конфигурации Адреса IP

Как тощая альтернатива для DCHP, Микрокристалл TCP/IP Стека также осуществляет простой метод, известный как IP, отбирающий тщательно, чтобы дистанционно установившее адрес IP узла Стека Микрокристалла. Этот метод - не стандарт Протокола Internet, и нет соответствуя документу RFC.
ОТБИРАТЬ IP тщательно позволяет адрес IP, чтобы быть установленн.
Для полной конфигурации IP, DCHP должен быть использован.

ОТБИРАТЬ IP тщательно не требует любые дополнительные программные модули. Взамен, это использует ARP и модули ICMP.
Для того, чтобы использовать это, файл icmp.c должен быть включен в проект, и компилятор определяет STACK_USE_IP_GLEANING должен uncommented в файл заголовка StackTsh.h.

С IP, отбирающий тщательно разблокированным, Стек Микрокристалла входит в специальный режим Конфигурации IP в Сбросе. В течение этого режима, это принимает любого ICMP сообщения (звон), которое обречено в этот узел и устанавливает адрес node s IP в то самое message s расположение звона. Как только правильный пакет звона будет получен, стек выходит из Конфигурации и входит в нормальный режим.

В течение конфигурации, приложение пользователя не должно пытаться связываться; это может вызвать функцию StackIsInConfigMode(), чтобы определять во всяком случае стек - в режиме Конфигурации.
StackIsInConfigMode() == ИСТИНА указывает, что стек - в режиме Конфигурации.

Чтобы дистанционно установившее адрес IP узла стека, необходимо должно выпускать последовательность команд из дистанционной машины. Для этого примера, допустите что узел, выполняющий Стек Микрокристалла должен быть назначен адрес IP 10.10.5.106. Адрес MAC этого узла (шестнадцатеричный) - 0a-0a-0a-0a-0a-0a. После сброса узла, другой системы в сети (мы примем компьютер, выполняющий Microsoft Windows), выпускает команды:

```
> arp -s 10.10.5.106 0a-0a-0a-0a-0a-0a
>ping  10.10.5.106
```

http://www.microchip.com/

Это должно сгенерировать стандартную серию ответов ping с 10.10.5.106, указание узла успешно назначено адрес IP.

Менеджер Стека

Как уже отмечено, Микрокристалл TCP/IP Стека является инкассированием других модулей. Некоторые модули (как например, IP, TCP, UDP и ICMP), должно быть вызвано когда соответствующий пакет получен. Любое приложение, использовавшее Микрокристалл TCP/IP Стека должно выполнить определенные шаги, чтобы проверять, что модули вызваны в подходящем времени. Эта задача остатков модулей управляющего стека тот же, независимо от основной прикладной логики.

Для того, чтобы облегчать основное приложение из бремени, управляющего индивидуальными модулями, Микрокристалл TCP/IP Стека использует специальный прикладной модуль слоя узнанный как StackTask , или Менеджер Стека. Этот модуль осуществлен исходным файлом StackTsk.c . StackTask ОСУЩЕСТВЛЕН как кооперативная задача; когда данное обрабатывающее время, это опрашивает слой MAC для правильных пакетов данных. Когда один получен, это декодирует это и разгромы это в подходящий модуль для продвигать обработка.

Важно должно отмечать, что Менеджер Стека является не часть Микрокристалла TCP/IP Стека. Обеспечено стеком, так что основное приложение нет должно управлять модулями стека, дополнительно к своей собственной работе. Прежде, чем задача Менеджера Стека может быть помещена, чтобы работать, она должна быть инициализирована разговором StackInit()function. Эта функция инициализирует переменные Менеджера Стека и индивидуальных модулей в правильном заказе. Как только StackInit функции(), будет вызван, основное приложение должно вызвать StackTask() периодически, чтобы проверять, что все поступающие пакеты оперируются вовремя, вместе с любой задержкой и обработкой условия ошибки.

Точные шаги использованные StackTask показаны в алгоритме в Примере 1.

ПРИМЕР 1: АЛГОРИТМ МЕНЕДЖЕРА СТЕКА

Если пакет данных получал бы,
тогда

Получите тип протокола пакета данных

Если тип пакета - IP,
тогда

Заголовок Выборки IP пакета

Получите тип пакета IP

если тип пакета IP - ICMP, тогда

http://www.microchip.com/

Модуль Вызова ICMP

еще если тип пакета IP - TCP, тогда

Вызовите модуль TCP

Еще

если тип пакета IP - UDP, тогда

Вызовите модуль UDP еще

Ручка не поддерживала

Конец протокола

Если Конец

Если

Еще если тип пакета - ARP, тогда

Вызовите модульный Конец ARP

Если Конец Если

//----------

EXAMPLE 1: THE STACK MANAGER ALGORITHM

If a data packet received then

Get data packet protocol type

If packet type is IP then

Fetch IP header of packet

Get IP packet type

if IP packet type is ICMP then

Call ICMP module

else if IP packet type is TCP then

Call TCP module

else if IP packet type is UDP then

http://www.microchip.com/

```
            Call UDP module
        else
                Handle not supported protocol
            End If
        End If
    Else if packet type is ARP then
            Call ARP module
        End If
End If
//*********************************************************************
```

СЕРВЕР МИКРОКРИСТАЛЛА HTTP

Сервер HTTP включенный этим прикладным примечанием осуществлен как кооперативная задача, что со-существует с Микрокристаллом TCP/IP Стека и user s основное приложение.
Сам Сервер осуществлен в исходном файле HTTP.c , с приложением пользователя, осуществляющим две функции возврата. Демонстрационный прикладной исходный файл файла Websrvr.c должен быть использован как приложение шаблона, чтобы создавать необходимые интерфейсы.

Сервер HTTP предусматривал здесь не осуществляет все функциональное назначение HTTP; это - минимальный сервер нацеленный для вложенной системы. Пользователь может легко добавить новое функциональное назначение как потребовалось.

- Сервер HTTP включает эти основные характеристики:
- Опоры многочисленных связей HTTP
- Содержит простую файловую систему (MPFS)
- страницы Опор Web располагались в или внутренняя программная память или внешний последовательный EEPROM

- Включает базирующуюся программу PC, чтобы создавать образы MPFS из данного директория Поддерживает метод HTTP ПОЛУЧАТЬ (другие методы могут легко быть добавлены)

- Опоры модифицировать Общий Межсетевой Интерфейс (CGI), чтобы вводить встроенные функции из дистанционного окна просмотра

- Поддерживает страничное содержимое динамического поколения веб, которое сервер состоит из следующих основных компонентов:

- Разработчик Образа MPFS

- Библиотека Доступа MPFS

- Программа Загрузки MPFS (осуществленное основным приложением)

- Задача Сервера HTTP

Для того, чтобы внедрять Сервер HTTP в приложение пользователя, сделайте следующим:

1. Uncomment STACK_USE_HTTP_SERVER В файле заголовка StackTsk.h , чтобы приспосабливаться Сервер HTTP связавший код.
2. Установите желаемую величину MAX_HTTP_CONNECTIONS в файле заголовка StackTsk.h .
3. Включите файлы http.c и mpfs.c в проект.
4. Зависеть где страницы веб загружены, uncomment также MPFS_USE_PGRM, или MPFS_USE_EEPROM. Если внешние данные EEPROM использованы как носитель памяти, включите файл xeeprom.c также.
5. Модифицируйте основу() функцию приложения, чтобы включать сервер HTTP (смотри кодовый пример в Примере 1 (страница 8)).

Это также будет необходимо генерировать любые страницы Web заранее и преобразовывать их в совместимый формат для памяти. Для Стека Микрокристалла и Сервера HTTP, конкретный формат - MPFS (смотри секцию в "Файловой Системе Микрокристалла (MPFS)" (страница 83) относительно деталей).
Если образ MPFS должен быть загружен во внешние данные EEPROM, ПРОГРАММИРУЮЩИЙ метод может должно быть включено в приложение, особенно если содержимое ожидается изменяется. Есть три основных опции для программирования внешних EEPROMs:

1. Используйте приложение программиста или устройства поставленные поставщиком данных EEPROM, чтобы программировать образ MPFS. Всегда убедитесь, что образ MPFS начинается после Резервного Блока .

http://www.microchip.com/

2. Включите программу загрузки в основное приложение, которое может принять данные из внешнего источника данных (то есть, из компьютера через последовательную связь данных) и программа это в EEPROM. Программа примера предусмотрена Стеком Микрокристалла исходного кода (смотри "Демонстрационную Программу Загрузки MPFS" (страница 88)).

3. Включите модуль Сервера FTP в проект и программируйте образ MPFS дистанционно через сеть, использовавшую FTP. Смотри "Загрузку Образа MPFS Используя Клиента FTP" (страница 85) более подробно.

Сервер HTTP использует файл index.htm как по умолчанию страницу Web. Если дистанционный клиент (окно просмотра) имеет доступ к Серверу HTTP своим адресом IP или доменное имя только, index.htm - обслуженная по умолчанию страница.
Это требует, чтобы все приложения включали файл называл index.htm как часть их образа MPFS. Если необходимо, имя этого по умолчанию файла может быть изменено модифицирующим определением компилятора HTTP_DEFAULT_FILE_STRING в файле http.c .

Очень важно должно убедиться, что ни одно из страничных файловых имен Web не содержат любое из следующего не-текстовые символы:

- единственные или двойные цитаты (символ и)
- левые или правые скобки угла (< and >)
- фунтовый знак (#) проценты знака (%)
- левые или правые скобки или скобы ([,{,] и })
- труба (|)
- обратная косая черта (\ )
- символ ^ (^)
- тильда (~)

Если файл содержит любой из этих символов, соответствующая страница Web станет недоступной. Никакое предшествующее предупреждение не будет дано.

Сервер HTTP также поддерживает список файловых типов, которые он поддерживает. Это использует эту информацию, чтобы советовать дистанционное окно просмотра о том как интерпретировать конкретный файловый, базирующийся в file s трех письмо расширении. По умолчанию, Сервер Микрокристалла HTTP поддерживает .txt , .htm , .gif , .cgi , .jpg , .cla и файлы .wav . Если приложение использует файловые типы, что не включены в этот список, пользователь может

http://www.microchip.com/

модифицировать таблицу перечислений httpFiles , вместе с соответствующими httpContents в файле http.c .

//**

ДИНАМИЧЕСКОЕ СТРАНИЧНОЕ ПОКОЛЕНИЕ HTTP

Сервер HTTP может динамически изменить страницы и заменять информацию в реальном времени, как например, ввод/выходной статус. Для того, чтобы включать эту информацию в реальном времени, соответствующий файл CGI (*.cgi), должно содержать текстовую строку %xx , где % символ служит в качестве управляющего кода и xx представляет двух цифровой переменный идентификатор.

Переменная величина имеет дипазон 00-99. Когда Сервер HTTP сталкивается с этой текстовой строкой, он удаляет % символ и называет функцию HTTPGetVar. Если страница требует % как дисплейный символ, она должна быть следована за другим % символ. Например, чтобы отображать 23% на странице, помещенной 23%% .

HTTPGetVar

Эта функция является возвратом из HTTP. Когда сервер HTTP сталкивается со строкой %xx на странице CGI, которую он обслуживает, это вызывает эту функцию. Эта функция осуществлена основным приложением пользователя и использована, чтобы передавать специализированный переменный статус на HTTP.

Syntax
WORD HTTPGetVar(BYTE var, WORD ref, BYTE *val)

Parameters
var [in]

Переменный идентификатор, чей статус должен быть возвращан

ref [in]

Ссылка Вызова. Эта величина ссылки указывается если это - тот же вызов первого. После вызов первого, эта величина строго поддержана основным приложением. HTTP ИСПОЛЬЗУЕТ обратную величину этой функции, чтобы определять призывать эту функцию снова к более данным. Если только один байт передается за один раз с этим возвратом, величина ссылки позволяет основное приложение, чтобы следить своей передачи данных. Если переменный статус требует более, чем байты, основное приложение может использовать ref как индекс в массив данных, который нужно возвращаться. Каждый раз байт послан, скорректированная величина ref возвращана как обратная величина; та же величина пройдена в следующий возврат. В конце, когда последний байт послан, приложение должно возвращать HTTP_END_OF_VAR как обратную величину. HTTP ПРОДОЛЖИТ вызывать эту функцию пока это не получит HTTP_END_OF_VAR как обратную величину.

Возможные величины для этого параметра:

Value	Meaning
HTTP_START_OF_VAR	Это - сам первый возврат для данной переменной для текущего примера. Если много-байтовая передача данных потребовалась, эта величина должна быть использована, чтобы условно инициализировать индекс в много-байтовый массив, который будет передан для текущей переменной.
Для всей другой	Основной специализированной величины.

val [out]

Один байт данных, которые должны быть перевестися

Return Values
Новая величина ссылки как определено основным приложением. Если эта величина - кроме HTTP_END_OF_VAR, HTTP вызовет эту функцию снова с обратной величиной с предшествующего вызова.
Если HTTP_END_OF_VAR возвращан, HTTP не вызовет эту функцию и допускает что переменная передача величины завершена.
Возможные величины для этого параметра:

Значение	Величины
HTTP_END_OF_VAR	Это - последний байт данных для данной переменной. HTTP НЕ вызовет эту функцию пока другая переменная величина не будет не нужно.
Для всей другой	Основной специализированной величины.

Pre-Condition
None

HTTPGetVar (Continued)

Side Effects
None

Remarks
Хотя эта функция запрашивает переменную величину из основного приложения, приложение нет должно возвращать величину. Фактическая переменная величина могла быть массивом байтов, которые могут или не может быть переменной величиной. Какая информация в возврат полностью зависимая от основного приложения и связанная страница Web. Например, переменная 50 могут означать фрейм JPEG пикселей 120 x 120. В этом случае, основное приложение может

использовать ссылку как индекс в фрейм JPEG и возвращать один байт от до HTTP. HTTP ОСТАНЕТСЯ вызывать эту функцию пока она не получит HTTP_END_OF_VAR как обратную величину этой функции.

Если эта функция имеет обратную величину 16 битов, вплоть до 64 Кбайтов данных может быть передано как одна переменная величина.
Если более длина - нужно, два или больше переменных могут устанавливаться рядом, чтобы создавать массив передачи больших данных.

Example

Рассматривайте страницу status.cgi , которая - served HTTP.
status.cgi содержит следующее строки HTML:

…

<td>S3=%04</td><td>D6=%01</td><td>D5=%00</td>
…

В течение обработки этого файла, HTTP сталкивается с %04 строк. После синтаксического анализа это, HTTP делает возврат HTTPGetVar(4, HTTP_START_OF_VAR, &величина). Основное приложение пользователя осуществляет HTTPGetVar следующим образом:

```
WORD HTTPGetVar(BYTE var, WORD ref, BYTE *val)
{
// Идентифицируйте переменную.
// Это RB5?
если ( var == 4 )
{
            // Мы просто возвращаем  1 если RB5 - высокий,
            // или  0 если низкий уровень.
          If  ( PORTBbits.RB5 )
                *val =  1;
        else
                *val =  0;
           // Сообщите HTTP, что это - последний байт
           // переменной величины.
          return HTTP_END_OF_VAR;
           }
    else
// Чек других переменных...
...
}
```

Для более детали, сошлитесь на Webpages*.cgi файлы и соответствующий возврат в исходном файле Websrvr.c .

http://www.microchip.com/

//**

HTTPGetVar (Continued)

Example 2

Допустите что странице status.cgi нужно отображать серийный номер устройства сервера HTTP Web.
Страница status.cgi served HTTP содержит следующее строки HTML:

…
<td>Serial Number=%05</td>
 …

При обработке этого файла, HTTP сталкивается с %05 строк. После синтаксического анализа это, HTTP делает возврат HTTPGetVar(4, HTTP_START_OF_VAR, &величина). Основное приложение осуществляет HTTPGetVar следующим образом:

```
WORD HTTPGetVar(BYTE var, WORD ref, BYTE *val)
{
// Идентифицируйте переменную.
// Это RB5 ?.
// Если да, прооперируйте величину RB5 - быть подобным Примеру 1.
// Это переменная  SerialNumber ?
если ( var == 5 )
{
        // Серийный Номер является НЕДЕЙСТВИТЕЛЬНОЙ расторгнутой строкой.
       // Прежде всего определите, если это - очень вызов первого.
      if ( ref == HTTP_START_OF_VAR )
                {
               // Это - первый вызов. Инициализируйте индекс в строку SerialNumber
              // Мы используем ref как наш индекс.
               ref = (BYTE)0;
                }
// Байт Теперь доступа в текущем индексе и сохраняемый это в буфере.
*val = SerialNumberStr[(BYTE)ref];
// Мы достигали конца строки?
if ( *val == '\0' )
         {
      // Да, мы сделаны передавая строку.
      // Возврат с HTTP_END_OF_VAR, чтобы уведомлять сервер HTTP, который мы
      // завершены передавая величину.
        Return  HTTP_END_OF_VAR;
       }
   // Или иначе, индекс массива приращения и возврата это в сервер HTTP.
      (BYTE)ref++;
      // С тех пор как величина ref - не HTTP_END_OF_VAR, сервер HTTP вызовет
       // он снова для отдыха величины.
```

http://www.microchip.com/

```
    return ref;

else
// Чек других переменных...
...
}
```

Для более детали, ссшлитесь на файлы Webpages*.cgi и соответствующий возврат в исходном файле Websrvr.c .

//***

HTTP CGI

Сервер HTTP осуществляет модифицировать версию CGI. С этим интерфейсом, клиент HTTP может ввести функцию в пределах HTTP и получать результаты в форме страницы Web. Дистанционный клиент вводит функцию HTML ПОЛУЧАТЬ метод более чем с одним параметром.
Посмотрите RFC1866 (языковая спецификация HTML 2.0) более подробно.

Когда дистанционное окно просмотра выполняет метод GET более чем с одним параметром, сервер HTTP выполняет грамматический разбор это и вызывает основному приложению с фактическим кодом метода и параметра. Для того, чтобы оперировать этот метод, основное приложение должно осуществить функцию возврата с подходящим кодом.

Сервер Микрокристалла HTTP не выполняет URL, декодировавший . Это означает, что если любой из текста области формы содержит определенные специальные не-текстовые символы (как например, <, >, , #, %, и т.п.), фактическая величина параметра должна содержать %xx (xx двух цифровая шестнадцатеричная величина символа ASCII) вместо фактического символа. Например, вход <Name> должен возвращать %3CName%3C . Смотри " Сервер Микрокристалла HTTP" (страница 77) для полного списка символов.

Файл, который содержит форму HTML, должно иметь .cgi как файловое расширение.

HTTPExecCmd

Эта функция является возвратом из HTTP. Когда сервер HTTP получает метод GET более чем с одним параметром, он называет эту функцию. Эта функция возврата осуществлена основным приложением. Эта функция должна декодировать данный код метода и брать подходящие действия. Такие действия могут включить поставку нового страничного имени Web, чтобы быть возвращанн и/или выполнение задачи B/B.

Syntax
void HTTPExecCmd(BYTE **argv, BYTE argc)

Parameters
argv [in]
Список командных аргументов строки. Первая строка (argv[0]), представляет действие формы, тогда как остальные (argv[1..n]), - командные параметры.

argc [in]
Общее число параметров, включая действие формы.

Return Values
Основное приложение возможно нужно модифицировать argv[0] со страничным именем правильного веб, чтобы быть использованн как командный результат.

Pre-Condition
None

Side Effects
None

Remarks
Это - возврат от HTTP до основного приложения в результате дистанционного вызова. Могло быть одновременным вызовом (один после другое) данного метода. Основное приложение должно решить эти одновременные вызовы и действовать соответственно.

По умолчанию, количество аргументов (или области формы) и итог длин строки аргумента (или строка формы URL), ограниченное 5 и 80, соответственно. Предел областей формы включает имя действия формы. Если приложение требует формы более чем с четырьмя областями и/или общей строкой URL более, чем 80 символов, соответствующие определения MAX_HTTP_ARGS и MAX_HTML_CMD_LEN (определенное в http.c), должно быть возрасти.

HTTPExecCmd (Continued)

Example

Consider the HTML page “Power.cgi”, as displayed by a remote browser:

```
<html>
<body><center>

<FORM METHOD=GET action=Power.cgi>
<table>
<tr><td>Power Level:</td>
<td><input type=text size=2 maxlength=1 name=P value=%07></td></tr>
<tr><td>Low Power Setting:</td>
<td><input type=text size=2 maxlength=1 name=L value=%08></td></tr>
```

```
 <tr><td>High Power Setting:</td>
 <td><input type=text size=2 maxlength=1 name=H value=%09></td></tr>
<tr><td><input type=submit name=B value=Apply></td></tr>
</table>
</form>
</body></html>
```

Эта страница отображает стол с этикетками в первой колонне и текстовых величинах ящика во второй колонне. Первая колонка, первая ячейка колонны содержит строку "Силового Уровня"; вторая колонна является текстовым ящиком, чтобы отображать и модифицировать силовую величину уровня. Последняя колонка содержит кнопку помеченную "Относиться". Потребитель, рассматривающий эту страницу имеет способность модифицировать величине в текстовом ящике Силового Уровня и щелчка на кнопке "Прилагать", чтобы подвергать новую силовую величину уровня Кипе Микрокристалла.

Допустите что потребитель вводит величинам кнопки '5', '1 и '9 в Мощности Уровня, Low-Power, устанавливающее и High-Power, устанавливающее xt ящики te соответственно затем щелкает на "Прилагать". Окно просмотра будет создать строку просьбы HTTP "Power.cgi?P=5&L=1&H=9" и посылать это в сервер HTTP. Сервер в свою очередь называет HTTPExecCmd с следующими параметрами:

```
 argv[0] = “Power.cgi”, argv[1] = “P”, argv[2] = “5”, argv[3]=“L“, argv[4]=“1”, argv[5]=“H“,

argv[6]=“9“
 argc = 7
```

Основное приложение осуществляет HTTPExecCmd как ниже:

```
void HTTPExecCmd(BYTE *argv, BYTE argc)

 {

BYTE i;

                // Пройдите через каждые параметры для текущей команды формы.
                // Мы пропускаем имя действия формы, так что я начинаю в 1...

            for (i = 1; i < argc; i++ )
{
// Идентифицируйте параметр.

                if ( argv[i][0] == ‘P’ ) // Is this power level?
                   {

                   // Следующий параметр является величиной уровня Мощности.
                    PowerLevel = atoi(argv[++i]);              }

            else if ( argv[i][0] == ‘L’ ) // - эта Низкая Мощность, устанавливающая?
```

```
                    LowPowerSetting = atoi(argv[++i]);}

            else if ( argv[i][0] == 'H' ) // - эта Высокая Мощность, устанавливающая?

                    HighPowerSetting = atoi(argv[++i]); }

 // Если другая страница должна быть отображена в результате этой команды, копия
 // имя верхнего регистра в argv[0]
// strcpy(argv[0], "RESULTS.CGI"); }
}
```

Примечание: Для этого примера, общее число аргументов превышает встроенный предел 5. Для того, чтобы этот пример, чтобы функционировать правильно, величина MAX_HTTP_ARGS (расположенное в "http.c"), должно быть установлено на в 7.

Для более детали, ссшлитесь на файлы "Webpages*.cgi" и соответствующий возврат в исходном файле "Websrvr.c".

```
//*********************************************************************
```

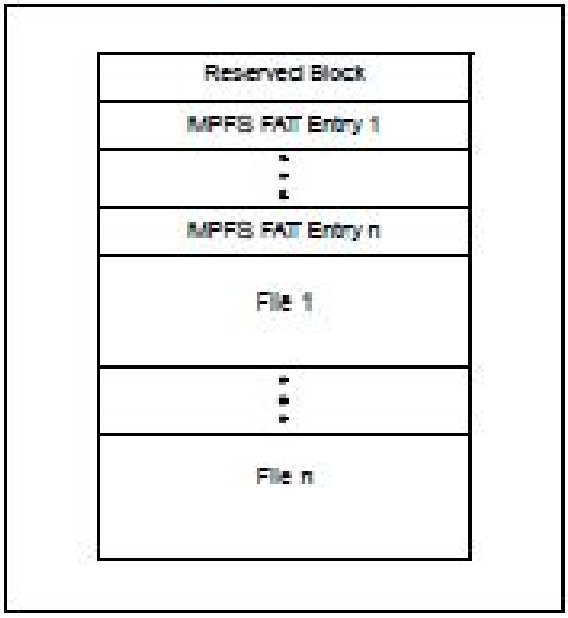
ФАЙЛОВАЯ СИСТЕМА МИКРОКРИСТАЛЛА (MPFS)

Сервер Микрокристалла HTTP использует простую файловую систему (Файловая Система Микрокристалла, или MPFS), чтобы загружать страницы Web. Образ MPFS может быть загружен в в-миниатюрной программной памяти или внешнем последовательном EEPROM. MPFS СЛЕДУЕТ за специальным форматом, чтобы загружать многочисленные файлы в данный носителя памяти, которые суммированы на Рисунке 3.

РИСУНОК 3: ФОРМАТ ОБРАЗА MPFS

http://www.microchip.com/

FIGURE 3: MPFS IMAGE FORMAT



Длина Резервного Блока определена MPFS_RESERVE_BLOCK. Резервный блок может быть использован основным приложением, чтобы хранить простые configuration величины. ПАМЯТЬ MPFS начинает с одного или более MPFS данных FAT (Файловая Таблица Распределения), сопровожденное одним или более файловыми данными. Вход FAT описывает файловое имя, позицию и статус. Формат для входа FAT показан на Рисунке 4.

FIGURE 4: MPFS FAT ENTRY FORMAT

Flag (8 bits)	Address (16 or 24 bits)	File Name (8-byte + 3-byte format)

Флаг указывает используется текущий вход, удаленное, или в конце FAT.

Каждый вход FAT содержит или 16- бит или 24-битовая величина адреса. Длина адреса определена типом памяти использованной, а также модель размера памяти. Если внутренняя программная память использована, и Микрокристалл TCP/IP проекта Стека компилирован небольшой моделью памяти, 16- битовые адресы использованы. Если внутренняя программная память и большая модель памяти выбраны, 24- битовый адрес использован. 16- Бит, указывающий схему всегда использован для внешних устройств EEPROM, независимо от модели размера памяти. Это подразумевает максимальный размер образа MPFS 64 Кбайтов для этих устройств.

MPFS ИСПОЛЬЗУЕТ файловые имена перемычки формата 8 + 3 (8 байтов для фактического файлового имени и 3 байтов для расширения, или NNNNNNNN.EEE). 16- Битовый адрес дает начало первого файлового блока данных. Все файловые

имена загружены в верхний регистр, чтобы делать файлом называть сравнения легче.

Адрес на каждом входе FAT указывает в свою очередь на блок данных, что содержит фактические файловые данные. Блочный формат данных показан на Рисунке 5. Блок расторгнутый со специальным 8- битовым флагом вызвавшим EOF (Конец Файла), сопровожденное FFFFh (для 16- битовой адресации), или FFFFFFh (24- битовая адресация). Если часть данных блока содержит символ EOF, она набита специальным символом перехода, DLE (Переход Связи Данных). Сам Любой случай DLE также набит DLE.

FIGURE 5: MPFS DATA BLOCK FORMAT

РИСУНОК 5: БЛОЧНЫЙ ФОРМАТ ДАННЫХ MPFS

Данные
(переменная длина) EOF (8 битов) FFFFh или (16 или 24 битов) FFFFFFh

Флаг Адреса (8 битов) (16 или 24 битов) Файловое Имя
(8- байт + 3-байтовый формат)

//**********

РАЗРАБОТЧИК Образа MPFS

Это прикладное примечание включает специальную базирующуюся программу PC (MPFS.exe), что может быть использовано, чтобы формировать образ MPFS из установки файлов. В зависимости от где MPFS в конце концов будет загружен, пользователь имеет опцию, чтобы генерировать или файл данных C или двоичного файла, представляющие образ MPFS.

Полный командный синтаксис строки для MPFS.exe

mpfs [/?] [/c] [/b] [/r<Block>]
<InputDir> <OutputFile>

где

/? подсказка командной строки дисплеев
/c генерирует файл данных C
/b генерирует двоичный файл данных (по умолчанию выход),
/r резервирует блока памяти в начале файла

(правильный только в Двоичном Выходном режиме, со значением по умолчанию 32 байтов)

<InputDir> - директорий, содержащий файлы для создания образа

<OutputFile> - выходное файловое имя

Например, команда

mpfs /c <Your WebPage Dir> mypages.c

генерирует образ MPFS как файл данных C, ***mypages.c***, с содержания директория "Ваши Страницы Web" . На контрасте, команда

mpfs <Your WebPage Dir> mypages.bin

генерирует двоичный файл образа с 32- байтом зарезервировавшим блока (оба двоичных кода форматируют и 32- байтовый блок - устанавливает по умолчанию), пока

mpfs /r128 <Your WebPage Dir> mypages.bin

генерирует тот же файл с 128- байтом зарезервировавшим блока.

Примечание:
Использование резервного блочного размера кроме по умолчанию 32 байтов требует изменение в компилятор определять MPFS_RESERVE_BLOCK в файле заголовка StackTsk.h .

Прежде, чем образ MPFS будет создан, пользователь должен создать все страницы Web и связавшие файлы и сохраняемые в единственном директории. Если файловое расширение - htm , полосы Разработчика Образа весь перевод строки и перевод строки символов, чтобы уменьшать общий размер образа MPFS.

Если образ MPFS должен быть загружен во внутреннюю программную память, сгенерированный проект файла данных C должен быть связан Websrvr . Если образ должен быть загружен во внешний последовательный EEPROM, двоичный файл должен загружаться там. Более подробно, сошлитесь на " Сервер Микрокристалла FTP" (страница 85).

Разработчик Образа MPFS не проверяет на наличие размера ограничений.
Если двоичный формат данных выбран, проверьте, что общий размер образа не превышает доступное пространство памяти MPFS.

БИБЛИОТЕКА Доступа MPFS

Исходный файл MPFS.c осуществляет программы требовавшиеся, чтобы иметь доступ к памяти MPFS. Пользователи не нужно понимать детали MPFS для того, чтобы использовать HTTP. Все имеют доступ к MPFS выполнен HTTP, без любой подсказки из основного приложения.

http://www.microchip.com/

Текущая версия библиотеки MPFS не учитывает дополнение или удаление индивидуальных файлов в существующий образ; все файлы, включающие единственный образ MPFS добавляются в любой момент. Любые изменения требуют создание нового файла образа.

Если внутренняя программная память использована для памяти MPFS, **MPFS_USE_PGRM** должен быть определен. Аналогично, если внешние данные EEPROM использованы для памяти MPFS, **MPFS_USE_EEPROM** должен быть определен. Только одно из этих определений могут присутствовать в **StackTsk.h** ; компиляция-время чека убеждается, что только одна опция выбрана.

В зависимости от типа устройства памяти использованной, страничный буферный размер изменится. По умолчанию установка страничного буферного размера (как определено **MPFS_WRITE_PAGE_SIZE** в файле заголовка (MPFS.h)), - 64 байтов. Если другой буферный размер потребовался, измените величину **MPFS_WRITE_PAGE_SIZE** соответственно.

Примечание:
Эта версия библиотеки доступа MPFS использует файл xeeprom.c для доступа ко внешним данным EEPROMs. Когда файл прочитан или записанный, MPFS исключительно регулирует шину I2C и не допустит любого другого раба I2C или овладевает устройством, чтобы связываться.
Пользователи, создающие приложения со многочисленными устройствами I2C нужно иметь это в виду.

//**

СЕРВЕР МИКРОКРИСТАЛЛА FTP

Сервер FTP включенный этим прикладным примечанием осуществлен как кооперативная задача, что со-существует с Микрокристаллом TCP/IP Стека и user s основное приложение.
Сам Сервер осуществлен в исходном файле FTP.c .

Сервер FTP предусматривал здесь не осуществляет все функциональное назначение определенное в RFC 959; это - минимальный сервер нацеленный для вложенной системы. Это включает эти основные характеристики:

- Один связь FTP, удостоверенная приложением пользователя
- Автоматическое взаимодействие с файловой системой (MPFS)
- Дистанционная программируемость целого образа MPFS с одной командой поместившей
- Никакой индивидуальный файл не загружает или извлекает поддержку

Потребитель может добавить новое функциональное назначение как потребовалось. Сервер действительно состоит из двух компонентов: сам сервер FTP, и возвраты аутентификации FTP осуществлялись приложением пользователя. Пользователь должен осуществить возврат с функцией **FTPVerify**, которая проверяет как имя пользователя FTP так и пароль. (Смотри функцию **FTPVerify** на следующей странице для более детали.) Демонстрационный прикладной исходный файл **Websrvr**.c должен быть использован как приложение шаблона, чтобы создавать необходимые интерфейсы. Посмотрите Демонстрационное Приложение для рабочего примера того как, чтобы внедрять сервер FTP в приложение пользователя. Чтобы внедрять Сервер FTP в приложение пользователя, сделайте следующим:

1. разкоментируйте STACK_USE_FTP_SERVER В файле заголовка StackTsk.h .

2. Увеличьте количество определенных розеток к двум (в зависимости от числа уже доступного).

3. Включите файлы FTP.c и mpfs.c в проект.

4. разкоментируйте Или MPFS_USE_PGRM или MPFS_USE_EEPROM, в зависимости от где страницы Web должны быть загружены. Если внешние данные EEPROM использованы как носитель памяти, включите xeeprom.c в проекте.

5. Модифицируйте основу() функцию приложения, чтобы включать сервер HTTP (смотри кодовый пример в Примере 1 (страница 8)).

Сервер Микрокристалла FTP допускает только один связь FTP за один раз. Каждая связь FTP требует две розетки TCP.

СЕРВЕР FTP использует по умолчанию задержку приблизительно 180 секунд для обоих загружает и загружается. Если дистанционная связь FTP остается ОЖИДАНИЕМ в течение более, чем 180 секунд, она автоматически разъединена. Это гарантирует, что сиротская связь или проблема в течение файла не загружать связывается сервер FTP неопределенно.

Загрузка Образа MPFS, использовавшие Клиента FTP

Основная цель Сервера FTP в Стеке Микрокристалла - дистанционно модернизировать образ MPFS. Текущая версия доступна только при использовании внешних данных EEPROM для памяти MPFS; это не будет работать если опция MPFS_USE_PGRM выбрана.

Типичный обмен между пользователем и узел, выполняющими Стек Микрокристалла с Сервером FTP показан в Примере 2. В данном случае, образ MPFS загружается от компьютера до узла. Для иллюстрации, это -, который пользователь должен видеть используя командное окно из компьютера, выполняющего Microsoft Windows; другие

системы и терминальная эмуляция могут измениться немного. Фактическое имя пользователя FTP и пароль зависит от приложения пользователя; демонстрационное приложение Сервера Web (страница 87) использует показанные величины. ДЕЙСТВИЯ Клиента FTP (то есть, ручной ввод из пользователя), показан в жирном шрифте. Система подсказывается и ответы сервера FTP - в простое лицо.

ПРИМЕР 2: ИСПОЛЬЗОВАНИЕ ОБРАЗА ЗАГРУЗКИ MPFS FTP

EXAMPLE 2: UPLOADING AN MPFS IMAGE USING FTP

```
c:\ ftp 10.10.5.15
220 ready
User (10.10.5.15: (none)): ftp
331 Password required
Password: microchip
230 Logged in
ftp> put mpfsimg.bin
200 OK
150 Transferring data...
226 Transfer Complete
ftp: 16212 bytes transferred in
0.01Seconds 16212000.00Kbytes/sec.
ftp> quit
221 Bye
```

c:\ ftp 10.10.5.15

220 готовый
Пользователь (10.10.5.15: (ничто)):

331 Пароль ftp требовал

Пароль: **microchip**

230 Регистрировался в

ftp> поместившее mpfsimg.bin

200 ok
150 Передающих данных...

226 Передач Полный

ftp> 16212 байтов переданных в

0.01Seconds 16212000.00Kbytes/sec.

http://www.microchip.com/

ftp> выход

221 Пока

Примечание:
Сервер FTP НЕ эхо возвращает пароль как типы пользователя это в. В данном случае выше, показано, чтобы иллюстрировать, который пользователь должен вводить.

//**

FTPVerify

Эта функция является возвратом из Сервера FTP. Сервер вызывает эту функцию когда он получает соединять запрос из дистанционного клиента FTP. Эта функция осуществлена основным приложением пользователя и использована, чтобы удостоверять дистанционного пользователя FTP.

Syntax
BOOL FTPVerify(char *login, char *password)

Parameters
login [in]
Символьная строка, которая содержит имя пользователя

password [in]
Символьная строка, которая содержит пароль

Return Values
ИСТИНА: Если вход и спичка пароля вход и переменные величины пароля определялись бы в приложении потребителя

ЛОЖЬ: Если или вход или пароль не сочетается

СЕРВЕР FTP использует обратную величину из этой функции, чтобы допускать или отвергать доступ дистанционным пользователем FTP.

Pre-Condition
None

Side Effects
None

Remarks

http://www.microchip.com/

Длина имени пользователя может поколебаться с 0 по 9. Если более длинное имя пользователя потребовалось, модифицируйте величину FTP_USER_NAME_LEN в файл заголовка ftp.h .

Длина строки максимального пароля и командной длины итога FTP встроенное, чтобы быть 31. Если более длинный пароль и/или команда потребовались, модифицируйте величину MAX_FTP_CMD_STRING_LEN в FTP.c .

Example

Этот пример демонстрирует как возврат FTP будет осуществлен.

```
ROM char FTPUserName[] = “ftp”;
#define FTP_USER_NAME_LEN   (sizeof(FTP_USER_NAME)-1)
ROM char FTPPassword[] = “microchip”;
 #define FTP_USER_PASS_LEN   (sizeof(FTP_USER_PASS)-1)

BOOL FTPVerify(char *login, char *password)
{
                if ( !memcmppgm2ram(FTP_USER_NAME, login, FTP_USER_NAME_LEN) )
          {
           if ( !memcmppgm2ram(FTP_USER_PASS, password, FTP_USER_PASS_LEN) )
                        return TRUE;
                                }
             return FALSE;
}
```

Для более детали, ссшлитесь на файлы Webpages*.cgi и соответствующий возврат в исходном файле Websrvr.c .

//**

ДЕМОНСТРАЦИОННЫЕ ПРИЛОЖЕНИЯ

Включенное Микрокристаллом TCP/IP Стека - полная работа приложения, которая демонстрирует всех TCP/IP модулей. Это приложение (Сервер Web), предназначен работать на демонстрационной плате Microchip s PICDEM.netTM Ethernet/Internet. Основной исходный файл для этого приложения - websrvr.c . Пользователи должны ссылаться на исходный код как отправной пункт для создание свои собственный приложения, использовавший другие аспекты Микрокристалла TCP/IP Стека. Проекты образца также включенные Микрокристаллом Складывают другие конфигурации иллюстрации в которых это демонстрационное приложение может работаться.

Выбор конфигурации Плата PICDEM.net и Сервер Web

http://www.microchip.com/

Чтобы запускать демонстрационное приложение, необходимо должно иметь ШЕСТНАДЦАТЕРИЧНЫЙ (HEX) кодовый файл. Одна опция должна сформировать одного из проектов образца основанных в компиляторе и конфигурации аппаратных средств, чтобы использованное; кроме того, используйте уже смонтированный ШЕСТНАДЦАТЕРИЧНЫЙ файл для соответствующего проекта (смотри Таблицу 2 на странице 5 для списка проектов включенных Стеком Микрокристалла). Последуйте за стандартной процедурой для вашего программиста устройства при программировании microcontroller.
Убедитесь, что следующие опции конфигурации установлены:

• Oscillator: HS
• Watchdog Timer: Disabled
• Low-Voltage Programming: Disabled

Когда программируемый microcontroller установлено на демонстрационной плате PICDEM.net и усилено по, ПРОВЕДЕННАЯ Система должна мигать, чтобы указывать, что приложение выполняет. Дисплей LCD покажет

v2.0 MCHPStack

в первой строке (номер версии может отличиться, в зависимости от уровня версии приложения), и или сообщение конфигурации или адрес IP во второй строке.

Как только запрограммировано, демонстрационное приложение может все еще должно конфигурироваться правильно прежде, чем оно будет установлено (накладывать) на реальную сеть.
Инструкции ниже характерные для Microsoft Windows и терминального эмулятора HyperTerminal;
ваша процедура может измениться если Вы используете другую операционную систему или терминальное программное обеспечение.

1. Запрограммируйте PIC18 microcontroller как упомянуто выше и было установлено это на плате PICDEM.net.

2. Подключите плату PICDEM.net к доступному последовательному порту в компьютере, используя стандартный кабель RS-232.

3. Запустите HyperTerminal (Запустите > Programs > Принадлежность).

4. В диалоговом блоке Описания Связи , вводить любой удобный называть в честь связи.
   Щелчок OK .

5. . В диалоговом блоке , Соединитесь, чтобы , выбираться порт COM на который плата PICDEM.net связана.
   Щелчок OK .

6. Сконфигурируйте последовательный порт подключенный к плате PICDEM.net:
   • 1 9200 bps,

• 8 data bits, 1 STOP bit and no parity
• no flow control
Щелчок OK , чтобы вводить связь.

7. Приложите мощность к совету при хранении ключа S3, или прессы и хранилище как Сброс так и ключи S3; затем, выпустите ключ Сброса. Показ LCD показывает сообщение

MCHPStack v2.0
Board Setup...

(Номер версии может отличиться в зависимости от уровня версии приложения.)
Выпустите S3.

Меню Конфигурации появляется в терминальном окне:

```
MCHPStack Demo Application v1.0
(Microchip TCP/IP Stack 2.0)
1: Change board Serial number.
2: Change default IP address.
3: Change default gateway address.
4: Change default subnet mask.
5: Enable DHCP and IP Gleaning.
6: Disable DHCP and IP Gleaning.
7: Download MPFS image.
8: Save & Quit.
Enter a menu choice (1-8):
```

ДЕМОНСТРАЦИОННОЕ Приложение MCHPStack v1.0
(Стек Microchip TCP/IP 2.0)

1: Серийный номер платы Изменения.
2: Изменение по умолчанию адреса IP.
3: По умолчанию межсетевой адрес Изменения.
4: Маска по умолчанию подсети Изменения.
5: Допустимый DHCP и Отбирать IP тщательно.
6: Выведите из строя DHCP и Отбирать IP тщательно.
7: Образ Загрузки MPFS.
8: Сохраняемый & Выход.

Введите выбор меню (1-8):

8. Выберитесь каждые пункты, которые должны config- ured и вводить новые величины (обычно, это только будет пунктами 2, 3, и 4). Пункт Выбора 8, чтобы сохранять изменения и выходить из конфигурации; новые адресы сохранены в

данные EEPROM. Приложение выходит из режима Конфигурации и выполняет Сервер Web.

Соединение в Сеть Ethernet

При прогоне Сервера Web демонстрационного приложения, плата PICDEM.net может непосредственно быть подключена к сети Ethernet без других модификаций. Конечно, конфигурация IP должна быть совместимой с той самой сетью. По умолчанию, приложение Сервера Web использует эти величины для конфигурации:

- АДРЕС IP: 10.10.5.15
- Межсетевой Адрес: 10.10.5.15
- Маска Подсети: 255.255.255.0

Даже если бы адрес IP совместимый, шлюз и маска не может быть. Если изменения потребовались, есть несколько путей ходить об этом.

//**********

АВТОМАТИЧЕСКАЯ КОНФИГУРАЦИЯ С DHCP

Если сеть использует конфигурацию DHCP, никакая дополнительная работа не - нужно. Когда плата подключена к сети и усилена по, она будет назначена конфигурация IP сервером DHCP. В течение этого процесса, дисплей LCD показывает сообщение

DCHP/Gleaning...

После нескольких секунд, дисплей показывает назначенный адрес IP, например:

100.100.100.1 1

Фактический адрес IP отображался в назначенном адресе на борту. Число в дальнем праве указывает раз аренда DHCP заменена. Это показан для информационных целей только. В зависимости от как сеть сконфигурирована, PICDEM.net адреса board s IP может измениться после того, как усиленный вниз на расширенный период (то есть, аренда board s DHCP истекла и старый адрес взят другим устройством). Всегда используйте адрес IP к настоящему времени отображенный, чтобы связываться с на борту.

ОПРЕДЕЛЕННЫЕ СЕТЕВЫЕ КОНФИГУРАЦИИ PRE

Некоторые сети могут трудно сконфигурирован; то есть, каждое устройство имеет адрес, который вручную назначен сетевым администратором. В этих случаях, плата PICDEM.net должна конфигурироваться вручную перед присоединением этой в сеть, с конфигурацией IP предусмотренной администратором. Сошлитесь обратная сторона на "Выбор конфигурации Плата PICDEM.net и Сервер Web" (страница 87) относительно деталей.

УСТАНОВКА АДРЕСА IP С ОТБИРАТЬ IP тщательно

Если плата подключена к сети и только потребовалась изменение адреса IP, Отбирать IP тщательно (страница 76), может быть использовано. Как уже упомянуто, этот метод может быть использован, чтобы конфигурировать адрес IP, но не межсетевая или маска подсети.

Для того, чтобы использовать IP, отбирающий тщательно, адрес MAC устройства должен известный. Это - всегда 6- байтовый шестнадцатеричный номер формата ***xx-xx-xx-xx-xx-xx*** . Для платы PICDEM.net, MAC - всегда ***00-04-A3-00-nn-nn***, где nn-nn - серийный номер платы в шестнадцатеричном формате. Таким образом, плата с серийным номером 1234 (или 04D2h), имеет адрес ***MAC 00-04-A3-00-04-D2***.

Как только адрес MAC и новый адрес IP устройства будет определен, адрес определен сбросом устройства, затем выпускаясь с дистанционного терминала **arp** и команды **ping**. Продолжающий пример выше, если мы захотели бы назначить прежде упомянутой платой новый адрес ***IP 10.10.5.50***, мы должны послать команды

```
> arp -s 10.10.5.50 00-04-a3-00-04-d2
> ping 10.10.5.50
```

Успешный ответ ping указывает, что адрес IP изменен.

ДЕМОНСТРАЦИОННАЯ Программа Загрузки MPFS

Если образ MPFS должен быть загружен во внешний последовательный EEPROM, он должен или быть pre- программируемым с образом MPFS или загруженным из другого приложения.
Демонстрационная принадлежность Сервера Web простая программа загрузки MPFS, которая принимает двоичный файл MPFS из терминального эмулятора, использовавшего протокол Xmodem.

Чтобы загружать двоичный файл в формат MPFS:

1. Если уже не сделано, установите плату PICDEM.net для конфигурации (смотри "Выбору конфигурации Плату PICDEM.net и Сервер Web", шаги 1 по 7).

http://www.microchip.com/

2. В меню Конфигурации, тип 7, чтобы запускать загрузку MPFS. Вы должны видеть Готовый загрузиться... сообщение. В на этот раз, Вы должны также видеть ПРОВЕДЕННОГО левого Пользователя (D6) мигая приблизительно дважды в сукунду.

3. Из HyperTerminal меню Передачи, выбор Посылает Файл. В диалоговом блоке, Пошлите Файл, просматривайтесь в директорий, содержащий файл mpfsimg.bin , и выбор. Выбор Xmodem как протокол.

4. Щелчок Посылает. Передача Данных должна запускаться автоматически. ПРОВЕДЕННЫЙ Пользователь мигнет так же быстро как данным получен из компьютера.

5. Когда файл полностью перевестися, нажмите 8, чтобы выходить из режима Конфигурации.
Приложение Сервера Web теперь работает, и сервер HTTP готовый обслужить страницы только что загруженные.

Демонстрационный Сервер HTTP

Когда сконфигурировано правильно и было предусмотрено образом MPFS, демонстрационный Сервер HTTP (страница 77) обслужит страницы Web. Страницы образца включенные Микрокристаллом Складывают исходную архивную иллюстрацию как модифицировать форма CGI (дистанционный вызов метода) так и динамическое страничное поколение.

Демонстрационный Сервер FTP

Это демонстрационное приложение обсуждается подробно запускаясь на страницу 85. Между прочим, это учитывает дистанционно обновляя образ MPFS через сеть. По умолчанию конфигурация использует имя пользователя FTP ftp и пароля микрокристалла (user name of “ftp” and password of “microchip”). Эти определены в демонстрационном приложении Сервера Web.

//**********

КОНФИГУРАЦИЯ СКОЛЬЖЕНИЯ ДЛЯ WINDOWS 95/98 СИСТЕМ

Любой персональный компьютер, выполняющий 32- битовые версии Microsoft Windows (то есть, Windows 95 или выше), может быть сконфигурирован, чтобы использовать связь SLIP для своих сетевых услуг связи. Эта секция очерчивает шаги требовавшиеся, чтобы ориентировать настольную систему на SLIP и создавать pre-определенную связь SLIP.

http://www.microchip.com/

Создание связи сделано в двух шагах: 1) создавая ложный модем в доступном порту COM, и 2) определяя наборный тип связи для этого устройства.

Описанная процедура здесь относится как к Windows 95 (все версии кроме оригинальной версии) так и 98; некоторые шаги могут отличиться, в зависимости от уровня операционной системы и исправленного издания. Краткости, процесс конфигурации для Windows NT и Профессионала Windows 2000 Desktop опущен. Заинтересованные пользователи на создании связи SLIP с этими операционными системами должны ссылаться на документацию Микрософт если нужна дополнительная информация, используя эти шаги как руководство.

Чтобы создавать ложный модем:

1. С Панели Управления (Запустите > Settings > Панель Управления), двойной щелчок на вводе Модемов.

2. Если никакой модем не установлен в системе, вводный диалоговый блок появится; проверьте подходящего блока, чтобы предохранять Windows от поиска устройства. Щелчок Затем.
Если модем уже установлен, лист Свойств Модема появится. Щелкните кнопку Добавлять.
В любых последующих диалоговых блоках, opt, чтобы вручную выбирать устройство; не сделайте позволять поиск Windows устройства или автоматически выбирайтесь драйверы.

3. На выборе устройства диалогового блока, выбора Стандартных Типов Модема из капли-вниз списка Изготовителей ; выберитесь любой из стандартных модемов (предподтительно 19.2 Kbaud или быстрее) из списка Моделей . Щелчок Затем .

4. В диалоговом блоке следующего, выбираться подходящий порт COM и ускорять (если запрошено). Щелчок Затем .

5. Следуя за установкой сообщения и краткая задержка, успешное сообщение установки появится. Щелчок Finish.

Как только устройство будет создано в соответствующем порту, оно может быть сориентировано на SLIP.

1. Запустите Наборного Волшебника Связи (Запустите > Settings > Наборные Связи > Добавлять Связь)

2. В первом , Сделайте Новой Связью общаться блок, выбирайтесь вновь созданный Стандартный Модем как устройство. Щелчок Затем .

3. Когда потребовалось номер телефона в следующем диалоге, введите любое номер телефона (это не будет использован). Используйте по умолчанию код области и позиция.
Щелчок Затем .

4. Щелчок Конца , чтобы создавать следующую связь. Это появится как новая иконка в папке Набора, связывающей .

5. Щелчок Права на новой иконке связи и выбора Свойств из меню.

6. На листе Свойств Связи, выберитесь таб. Генерала . Проверьте, что модем стандарта (манекена) выбран в капле-вниз меню Соединять Использование . Щелкните кнопку Конфигурировать .

7. На листе свойства конфигурации, установите показатель бода на 38,400. Выберитесь таб. Связи и проверяйте, что установочные параметры связи не являются 8 битов данных, 1 бита СТОП и никакой четности. Щелкните кнопку Передовой .

8. Отмените выбор Управление Потока Использования checkbox. Щелчок OK , чтобы закрывать диалогового блока, затем OK , чтобы закрывать лист свойства конфигурации (Вы - теперь на листе Свойств Связи).

9. Выберитесь таб. Типов Сервера . Выбор СКОЛЬЖЕНИЯ Связь Unix из капли-вниз списка Типа Сервера Dial-Up . Uncheck Всех контрольных блоков кроме внизу проверять блока, TCP/IP . Щелкните кнопку TCP/IP Установочных параметров .

10. Введите адрес IP 10.10.5.1 . Отмените выбор сжатие заголовка IP и опции по умолчанию шлюза Использования в дистанционной сети . Щелчок OK , чтобы возвращаться на лист Свойств.

Примечание 1: адрес использованный в этой конфигурации - для главной настольной системы для этой связи;
это - НЕ адрес целевой системы. Адрес IP 10.10.5.1 используется особо с Демонстрационной Платой PICDEM.net.

2:
Главная система и цель должны находиться в той же подсети.

11. Выберитесь таб. Описания затем щелкайте кнопку Просмотра , чтобы располагать директорий где файл empty.scp находится (включенное в Микрокристалл TCP/IP архивного файла Стека). Выберитесь файл и щелкайте OK .

12. Щелчок OK , чтобы завершаться.
Для того, чтобы использовать связь SLIP, двойной-щелчок на своей иконке. Используйте окно просмотра Web и любое приложение IP, чтобы связываться над связью.

//**********

ИСПОЛЬЗОВАНИЕ ПАМЯТИ

http://www.microchip.com/

Общая сумма памяти использованной для Микрокристалла TCP/IP Стека зависит от компилятора и модулей, использованных. Типичные размеры для различных модулей стека на момент публикации этого документа (Июнь 2002), используя HI-TECH PICC-18TM V8.11PL1 показаны в Таблице 4.

ТАБЛИЦА 4:
ИСПОЛЬЗОВАНИЕ ПАМЯТИ ДЛЯ РАЗЛИЧНЫХ МОДУЛЕЙ СТЕКА, ИСПОЛЬЗОВАВШИХ HI-TECH PICC-18 КОМПИЛЯТОР

TABLE 4: MEMORY USAGE FOR THE VARIOUS STACK MODULES USING HI-TECH® PICC-18™ COMPILER

Module	Program Memory (words)	Data Memory (bytes)
MAC (Ethernet)	906	5(1)
SLIP	780	12(2)
ARP	392	0
ARPTask	181	11
IP	396	2
ICMP	318	0
TCP	3323	42
HTTP	1441	10
FTP Server	1063	35
DHCP Client	1228	26
IP Gleaning	20	1
MPFS(3)	304	0
Stack Manager	334(4)	12+ICMP Buffer

Примечание
1: Как осуществлено RTL8019AS NIC.
2: Не включает размер передачи и получает буферы, которые - пользователь определялся.
3: Внутренняя программная память памяти.
4: Максимальный размер. Фактический размер может измениться.

ВЫВОД

Микрокристалл TCP/IP Стека не является первым приложением своего типа для семейства Микрокристалла PIC18 microcontrollers.
Что это обеспечивает, - космическая эффективная и модульная версия TCP/IP, осуществляющая кооперативный multitasking (без операционной системы) в небольшом следе. Со своей приспособляемостью, чтобы много приложений микропрограмм и доступности как никакое стоящее программное решение, этот стек должен оказаться полезным в проявлении приложений где вложенное управление требует сетевую связность.

//**********

http://www.microchip.com/

ПРИЛОЖЕНИЕ A: ИСХОДНЫЙ КОД

Полный исходный код для Микрокристалла TCP/IP Стека, включая демонстрационные приложения и необходимые файлы поддержки, доступен под никаким-стоящего лицензионного соглашения. Доступно для загрузки как единственный архивный файл из Микрокристалла корпоративного сайта Web, в

www.microchip.com.

После загрузки архива, всегда проверьте файлу version.log на наличие текущего уровня исправленного издания и история изменений в программное обеспечение.

ПРИЛОЖЕНИЕ B: ЧАСТИЧНЫЙ СПИСОК ДОКУМЕНТОВ RFC

APPENDIX B: PARTIAL LIST OF RFC DOCUMENTS

RFC Document	Description
RFC 826	Ethernet Address Resolution Protocol (ARP)
RFC 791	Internet Protocol (IP)
RFC 792	Internet Control Message Protocol (ICMP)
RFC 793	Transmission Control Protocol (TCP)
RFC 768	User Datagram Protocol (UDP)
RFC 821	Simple Mail Transfer Protocol (SMTP)
RFC 1055	Serial Line Internet Protocol (SLIP)
RFC 1866	Hypertext Markup Language (HTML 2.0)
RFC 2616	Hypertext Transfer Protocol (HTTP) 1.1
RFC 1541	Dynamic Host Configuration Protocol (DHCP)
RFC 1533	DHCP Options
RFC 959	File Transfer Protocol (FTP)

Полный список Internet RFCs и связанных документов доступен на много местах веб Internet.
Заинтересованные считыватели ссылались на www.faqs.org/rfcs и www.rfc-editor.org как отправные пункты.