ARP
BSD SOCKETS
DNS
DYNAMIC DNS
HASHES
HELPERS
HTTP2 SERVER
ICMP
MPFS2
NBNS
REFORMANSE TEST
SMTP
REBOOT
SNMP
SNTP
SSL
TCP
TFTP
TICK
UDP
WI FI

**3.1 Исполнения Стека**

Отметьте, что эта таблица не появится в версии PDF файла подсказки; смотри файл "Стека TCPIP Performance.htm" в папке документации TCPIP на подсказке Прикладной Библиотеки папки Микрокристалла.

**3.2 Использования Памяти**

Эти таблицы содержат PIC программные и требования памяти данных для стека TCP/IP. Первые две колонки указывают программное потребление памяти необходимых файлов стека ( смотри страницу 137), и каждая дополнительная колонка содержит дополнительную память требовавшуюся, чтобы осуществлять специфические модули. Эти величины являются аппроксимациями; программный размер памяти может возрасти в зависимости от прикладного кода, или уменьшение основывалось в оптимизации модулей с перекрытием кода. Модули, которые требуют осуществленные функции пользователя API (SNMP, HTTP), протестированы без дополнительного кода. Столбец памяти глобальных данных включается только PAM необходимое для необходимых структур в стеке; это не включает память использованную для распределения розетки ( смотри страницу 152).

Код C18 использует PIC18F97J60 семейного диспетчера Ethernet как чип MAC/PHY; C30 и размеры C32 сделаны используя диспетчера ENC28J60 Ethernet (размеры ENCX24J600 аналогичные). Все компиляторы включают отдельной строке Кода Необходимый Стек для приложений Wi-Fi, использовавших MRF24WB0M как сетевой диспетчер. Эти два Требовали, чтобы строки Кода Стека - взаимно исключительными -- не добавлять им вместе. Взамен, выберите линию представляя вашего сетевого диспетчера.

Эти величины являются аппроксимациями полученными из TCP/IP версии Стека 5.31. Отметьте, что эти таблицы не появится в версии PDF файла подсказки; смотри файлы "TCPIP Cxx Memory Usage.htm" в папке документации TCPIP на подсказке Прикладной Библиотеки папки Микрокристалла.

C18

C30

C32

### 3.3 Периферийных Использования

Несколько microcontroller периферийных устройств могут/должно быть использовано, чтобы осуществлять приложение стека TCP/IP.

| Типа | Specific/Configurable | Polled/Interrupt | |
|---|---|---|---|
| | Таймера 0 для PIC18, Таймера 1 в противном случае | Прерывание | Использованное, чтобы осуществлять таймер метки |
| | | | |
| | | | |
| | | | |

Таймер Цели.

SPI или PMP
Выбор через #define в HardwareProfile.h.
Смотри Конфигурацию Аппаратных средств ( смотри страницу 142).
Опрошенное. Модуль SPI использован, чтобы управлять ENC28J60 или MRF24WB0M. ENCX24J600 может быть проведен модулем SPI или модуль PMP.
ВЫБОР SPI через #define в HardwareProfile.h.
Смотри Внешнюю Память ( смотри страницу 142).
Опрошенное. Использованное, чтобы связываться в EEPROM или Последовательный чип Вспышки, как опция, чтобы загружать страницы веб для MPFS/MPFS2 структуры ( смотри страницу 271) или AppConfig ( смотри страницу 138).
ВЫБОР SPI через #define в HardwareProfile.h.
Смотри Внешнюю Память ( смотри страницу 142).
Опрошенное. Использованное, чтобы связываться в последовательное PAM как дополнительная розетка распределения ( смотри страницу 152) метод.

## 4 Кремниевых Решения

Один из первых выборов, чтобы делать при проектировании вашего приложения - какой аппаратный слой, чтобы использовать. Микрокристалл поддерживает множество аппаратных средств TCP/IP решений, каждый со встроенным MAC и/или PHY. ENC28J60 И ENCX24J600 - отдельные чипы диспетчера Ethernet, проявленный Технологией Микрокристалла. MRF24WB0M - отдельный 802.11b беспроволочный приемо-передатчик. PIC18F97J60 - PIC18 microcontroller со встроенным периферийным устройством Ethernet. Серия PIC32MX7XX/6XX 32- бита microcontrollers является высокими устройствами исполнения со встроенным периферийным устройством Ethernet MAC (MII/RMII интерфейса на внешний PHY).

Для информации о демонстрационной плате, которая использует эти устройства, смотри Демонстрационные Комплекты секции ( смотри страницу 68).

//----------
Технология Характеристики ENC28J60 ENCX24J600 PIC18F97J60 MRF24WB0M PIC32MX7XX/6XX Телеграфировала Ethernet Телеграфировавший Ethernet Телеграфировавший Ethernet 802.11
Телеграмма Телеграфировавшая Ethernet
MAC ВНУТРЕННИЙ Внутренний Внутренний Internal Internal PHY Внутренний (10-Base-T) Внутренний (10/100-Base-T) Внутренний (10-Base-T)
Внутреннего External PHY PAM (MII/RMII Интерфейса) Буфер (байты)
8,192 24,576 3,808 14,170 Перестраиваемых дескрипторов во Внутреннем PAM (128k Внутреннем PAM)
Интерфейс SPI SPI, 8 или 16 кусает мультиплексный или многоканальный параллельный интерфейс Ничто (встроенный Ethernet MAC/PHY)
SPI НИЧТО (встроенный Ethernet MAC)
Прикалывает 28 44, 64 64/80/100 36 64/100/121 Пакетов SOIC, SPDIP, SSOP, QFN (6x6 mm) TQFP, ПОВЕРХНОСТНАЯ Гора QFN TQFP модуля WiFi B/B TQFP, QFN (9x9 mm),
Шифровальное No Да Двигателей BGA No No No программировало Pre MAC адресует Да No(1) No(1) Да Да
//----------

1: Для устройств без pre- программируемого адреса MAC, Вы можете решать использовать EEPROM со встроенным адресом MAC, как например, семейство устройства описанное здесь ( смотри страницу 147).

## 5. Программных обеспечения

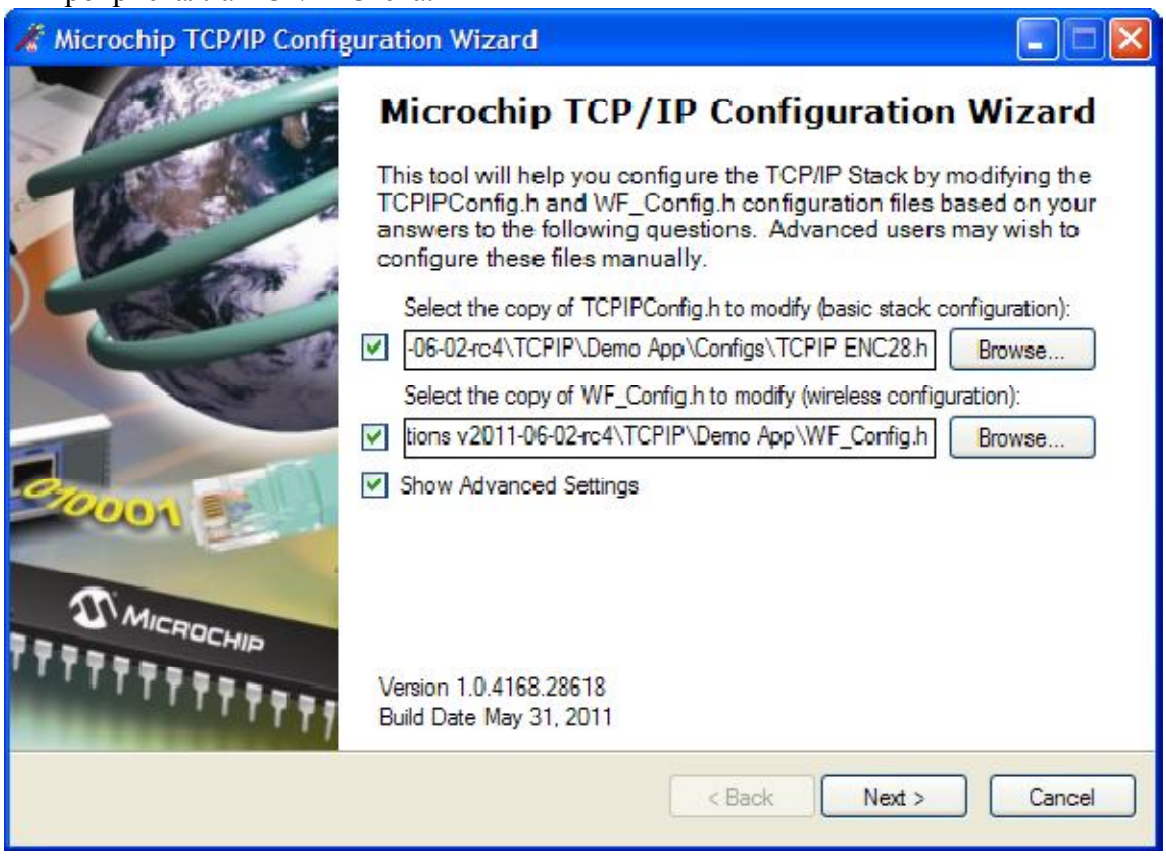
Эта секция обсудит компьютерные программные приложения включенные Микрокристаллом TCP/IP Стека.

Эти инструментальные средства осуществлены используя C# или языки программирования Java, или оба. # Инструментальные средства C (*.exe), потребует Microsoft Каркас .NET v2.0, чтобы быть установленн на локальном PC. Инструментальные средства Java (*. банка), требовать Время прогона Среды Java (JRE) 1.6 или позже, чтобы быть установленн в целевом компьютере.

### 5.1 TCP/IP ВОЛШЕБНИКА Конфигурации

TCP/IP Волшебника Конфигурации является легчайшим, safest путем устанавливать микропрограммы (и некоторые аппаратные средства) опции конфигурации. Это прочитает и выполняет грамматический разбор установочные параметры конфигурации из копии TCPIPConfig.h затем обеспечивает графический интерфейс пользавателя, который легко позволит Вам, чтобы рассматривать и модифицировать те установочные параметры. Кроме того, если характеристика, которую Вы приспосабливаете, потребует другой ресурс или представляет, чтобы действовать, дополнительные характеристики приспособится автоматически. TCP/IP Волшебника Конфигурации будет установлен в меню Начала когда TCP/IP Стека установлен.

Когда Вы запускаете волшебника конфигурации, Вы будете подсказаны, чтобы вводить путь в копию TCPIPConfig.h и в этом случае, чтобы модифицировать передовые установочные параметры конфигурации. Передовая установка опции даст более точное управление перегружать характеристики, но также потребует большее практическое знание Микрокристалла TCP/IP Стека.



### 5.2 Утилита MPFS2

Утилита MPFS2 упаковывает страницы веб в формат для эффективной памяти в вложенной системе. Это - графическая заявка на PCs, которая может сгенерировать образы MPFS2 для памяти во внешней памяти или программной памяти внутренней Вспышки.

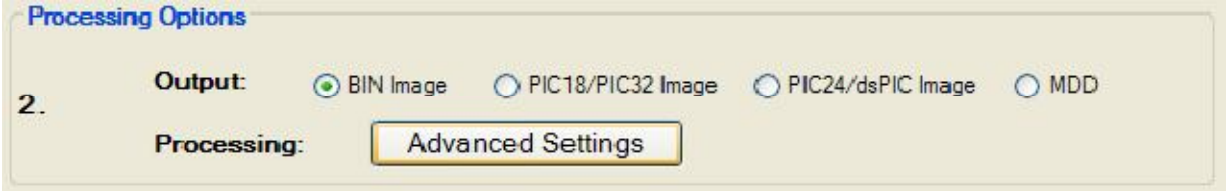
Когда использовано, чтобы формировать образы MPFS2, Утилита MPFS2 также индексирует обнаруженные динамические переменные. Это использует эту информацию, чтобы генерировать HTTPPrint.h, который проверяет, что соответствующие функции возврата введены как необходимый. Это также загружает эту индексную информацию вместе с файлом в образе MPFS2, которая облегчает задачу поиска с вложенного устройства.

Наконец, при проявлении приложения, которое использует внешнюю память, Утилита MPFS2 может загрузить образы во внешний загружающих устройство памяти смонтированного использования Образов Pre MPFS2 загружают функциональное назначение созданное в сервер веб HTTP2 или сервер FTP.
Исходный код для этого приложения включен в Библиотеки Приложений Микрокристалла installer.

#### 5.2.1 Формируя Образы MPFS2

Утилита MPFS2 имеет четыре шага, которые обозначаются слева передавать сторону диалога. Для того, чтобы формировать образ MPFS, выбор Начинает C: Директорий Webpage на этапе 1 и выбирает директорий в котором страницы веб загружены.

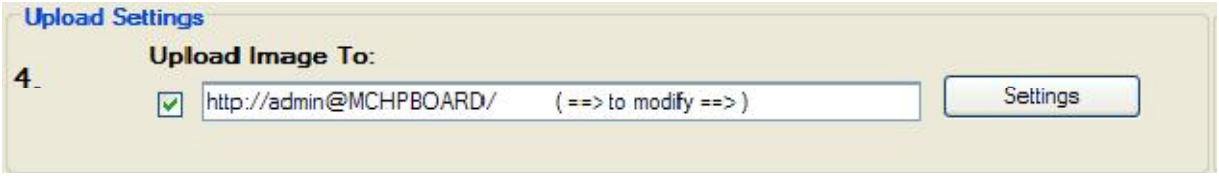
Шаг 2 выбирается выходной формат. При хранении страниц веб во внешней EEPROM или последовательной Вспышке, выберите выходной формат Образа BIN. Если внутренняя программная память будет использована, выберитесь Образ C18/C32 для использования с 8-битом и 32-битовыми частями, или Массивом ASM30 для 16- битовых целей. Для того, чтобы загружать страницы веб на устройство отформатированное файловой системой FAT не сжимая им в образ MPFS, выберитесь MDD (смотри Демонстрационный Прил. MDD, ПРИСТУПАЮЩЕЕ руководство более подробно).




Шаг 3 требует директорий проекта MPLAB IDE. Средство MPFS запишет образ файл в директорий проекта, и также скорректирует файл HTTPPrint.h там если нужно. Выберитесь правильный директорий чтобы файлы права были модифицированы.

Проведите 4 элемента управления загружать установочные параметры. Когда внешний EEPROM или последовательная вспышка использована для памяти, опция, чтобы загружать вновь созданный образ на плату доступно. Проверьте блока около Загружать Образ, чтобы, чтобы приспосабливаться эту характеристику. Целевое главное имя (или адрес IP), загружать протокол и загружать путь может должно быть изменено на один выбранное когда плата сначала была сконфигурирована. Вам возможно также нужно модифицировать имя пользователя и пароля использованные, чтобы иметь доступ к защищенному функциональному назначению в вашем приложении, подобно странице веб загружаться. Используйте кнопку Установочных параметров, чтобы редактировать эти величины.

Если внутренняя программная память используется, образ будет компилирован в с проектом и, так что направлять загружает, не доступны. Убедитесь, чтобы включать выходной исходный файл указанный на этапе 3 как часть проекта.

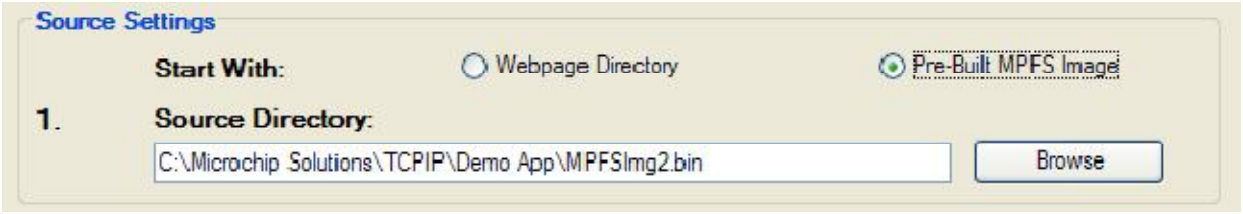



Как только все правильные установочные параметры выбраны, щелкните кнопку Generate, чтобы создавать образ. Если загружает, приспособлены, это также попытается загружать файл на устройство.

### 5.2.2 Загружая смонтированные Образы Pre MPFS2

Есть два пути загружать pre- смонтированный образ во внешнюю память. Сначала описан на ПОДГОТОВКЕ секции ( смотри страницу 78) и включен загрузка из окна просмотра непосредственно. Второй должен использовать Утилиту MPFS2, чтобы загружать образ. Вы можете выбрать HTTP или FTP, загружающий, чтобы соответствовать протоколу, который ваше приложение использует.

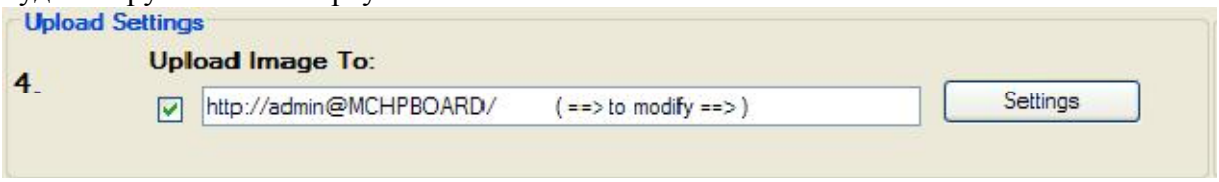
Для того, чтобы использовать Утилиту MPFS2, чтобы загружать образ, начните с выбора Начинать C: Образ Pre-Build MPFS на этапе 1 на верхе.
Решают файл образа, чтобы загружаться.




Шаги 2 и 3 не обязательны для pre- смонтированных образов. Приступите непосредственно к шагу четыре, чтобы загрузить установочные параметры.
Целевое главное имя (или адрес IP), загружать протокол и загружать путь может должно быть изменено на один выбранное когда плата сначала была сконфигурирована. Вам

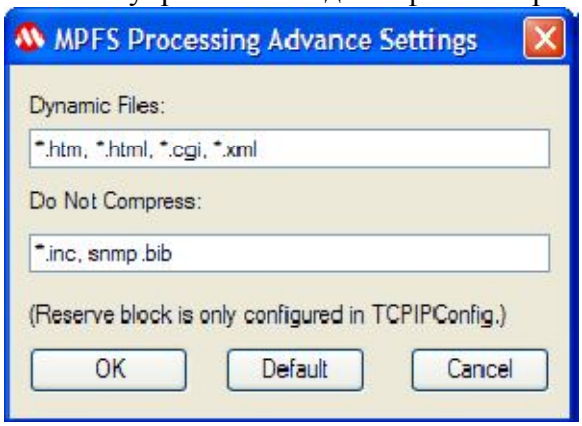
возможно также нужно модифицировать имя пользователя и пароля использованные, чтобы иметь доступ к защищенному функциональному назначению в вашем приложении, подобно странице веб загружаться. Используйте кнопку Установочных параметров, чтобы редактировать эти величины.

Как только все установочные параметры будут правильно, щелкните кнопку Upload. Образ будет загружен на на борту.




### 5.2.3 Предоставившее Установочные параметры MPFS2

Передовой диалог Установочных параметров обнаруженный на этапе 2 обеспечивает большее управление над как файлы обработаны.




Список **Dynamic Files** указывает какие файловые типы, чтобы выполнять грамматический разбор для динамических переменных. По умолчанию, все файлы с расширениями htm, html, cgi, или xml разобраны. Если приложение имеет динамические переменные в других файловых типах, эти типы должны быть добавлены к списку.
Эта область должна быть разделенным списком запятой расширений и файловых имен.

**Do Not Compress** указывает какие файловые типы не должно никогда сжат. Сжатие файлов с GZIP сохраняет как память космическое так и время передачи. Тем не менее, это только пригодное для статического содержимого как например, CSS или JavaScript. Любые файлы с динамическими переменными автоматически будут исключены. Кроме того, любой файл, что PIC возможно нужно обрабатывать непосредственно должно быть исключено. Файлы включались через ~inc:filename~ не должно сжимано, а также не должно любой файл НАГРУДНИКА использовался для модуля SNMP (если имеется). Дополнительные файловые типы могут быть добавлены к этому списку если заказное приложение получит MPFS.

Компрессор GZIP попытается уменьшать все файлы. В некоторых случаях, особенно с образами, плохо или вообще сжатие не достигнуто. Когда это происходит файл загружен как-- в образе MPFS.

### 5.2.4 Командных Опции Строки MPFS2

Чтобы облегчать пакетные файлы и автоматизацию, Утилита MPFS2 также поддерживает выполнение из командной строки. Синтаксис - следующим образом:

MPFS2.jar [options] <SourceDir> <ProjectDir> <OutputFile>

SourceDir, ProjectDir, И опции OutputFile потребовались и должны быть приложены в кавычках. Опция OutputFile будет относительно ProjectDir, и не мочь быть полным именем пути.

Различные ключи опции описаны в таблице ниже:

| **Switch** | **Short** | **Description** |
|---|---|---|
| BIN | /b | Output a BIN image (Default) |
| C18 C32 | /c | Output a C18 or XC32 image |
| ASM16 | /s | Output an ASM16 image |
| Mpfs2 | /2 | Use the MPFS2 format (Default) |
| Html | /h | File types to be parsed for dynamic variables (Default: "*.htm, *.html, *.cgi, *.xml") |
| xgzip | /z | File types to be excluded from GZIP compression (Default: "*.bib, *.inc") |

Короткого /b Выхода Description

/BIN образ BIN (Умолчание) /c Выход

/C18_C32 C18 или образ XC32 /s Выхода

/ASM16 образ ASM16

/mpfs2 /2 Использования формат MPFS2 (Умолчание)

/html "..." /h "..." Файл набирается, чтобы быть разобранн для динамических переменных (Умолчание: "*.htm, *.html, *.cgi, *.xml")

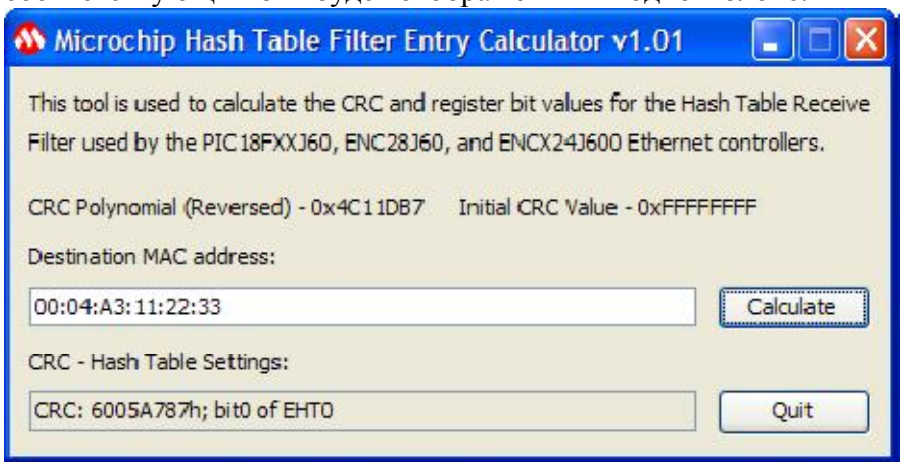
/xgzip "..." /z "..." Файл набирается, чтобы быть исключенн из сжатия GZIP (Умолчание: "*. нагрудник, *.inc")

Командная строка интерфейса не поддерживает образ загружается. Для пакета или производства загружает, использует средство как например, wget, чтобы загружать сгенерированный образ BIN.

**5.3 Входов Табличного Фильтра Калькулятора Хэша**

Этот фильтр приемника Таблицы Хэша на ENC28J60, ENCX24J600, и microcontroller семействе PIC18F97J60 выполняет вычисление CRC над шестью байтами адреса расположения в полученном пакете затем использует эту величину как указатель в регистры EHT0-EHT7. Если бит, что указатель указывает, чтобы установленн, пакет будет получен. Хэш Микрокристалла Табличного Калькулятора Входа Фильтра определит бит, который должен быть установлен в этом банке регистра для данного адреса расположения. Если у вас

есть фиксированный адрес MAC, узнанный в проектом времени, Вы можете установить ваш фильтр приемника Таблицы Хэша в вашем коде, использовавшем величину получал из этого средства; в противном случае, Вы должны использовать SetRXHashTableEntry функции(), чтобы устанавливать это в течение времени прогона. Для того, чтобы использовать это средство, определите адрес вашего устройства, щелчок вычисляет, и величина CRC и соответствующий бит будет отображен в выходном блоке.

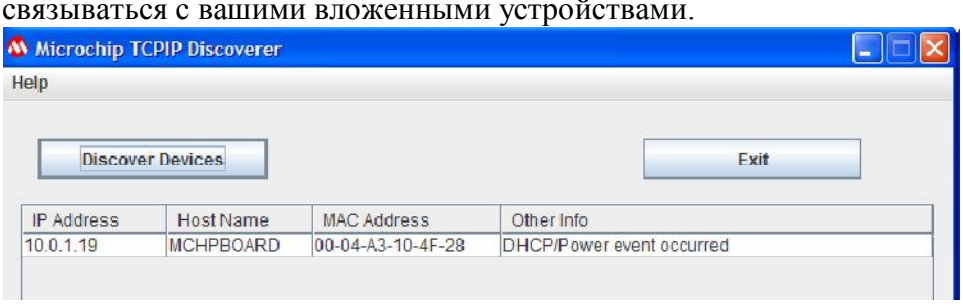



## 5.4 Microchip TCP/IP Discoverer

Микрокристалл TCP/IP проекта Открывателя PC протокола (fomerly узнанное как Вложенный Открыватель Устройства Ethernet), поможет в обнаружении устройства вложенного продукта (с Announce ( смотри страницу 155)) и продемонстрируйте как, чтобы записывать приложения PC, чтобы сообщать вложенные устройства.

Когда кнопка "Обнаруживает что Устройства" щелкнуты, это приложение будет передать широковещательный пакет UDP, содержащий сообщение, "Обнаружение:, который?" в локальной сети в порт 30303. Если любые вложенные устройства с Announce протокола ( смотри страницу 155) приспособленное подключены к сети, они укажут пакет UDP, содержащий свое главное имя (NBNS ( смотри страницу 290)) и адрес MAC.

Исходный код Java для этого приложения также включен. Этот исходный код должен обеспечить грубую идею того как, чтобы записывать базирующееся приложение PC, чтобы связываться с вашими вложенными устройствами.



# 6 Getting Started

Эта секция описывает шаги необходимые начинать использовать Микрокристалл TCP/IP Демонстрационных Приложений. Эта секция содержит специфической информации для установки и использования общего TCPIP Демонстрационный Прил. ( смотри страницу 86). Наиболее это информация установка может быть приложено, чтобы приступать с другими демонстрационными приложениями также.

## 6.1 Аппаратных Установки

Первый шаг, чтобы использовать стек - убеждаться что подходящая плата разработки сконфигурирована. Для того, чтобы приступать, выберитесь платформу из тем представленных ниже.

### 6.1.1 Дочерних Платы

Микрокристалл предлагает четыре дочерних платы, которые обеспечивают другое функциональное назначение Ethernet на доступную демонстрационную плату. Каждая плата разработана:

- PICtail Разъем, который приспосабливается интерфейс на PICDEM.сеть 2 плата ( смотри страницу 69) или PIC18 Explorer ( смотри страницу 71) (заполненное процессором PIC18) и/или

- PICtail Плюс разъем, который позволит, чтобы связываться в Explorer 16 разработки ( смотри страницу 72) плата (заполненное PIC24, dsPIC33, или процессором PIC32) или Комплект Стартера PIC32 ( смотри страницу 72).

Отметьте, что PICDEM.net 2 заполняется по умолчанию с ENC28J60 и PIC18F97J60.

Дочерняя Плата Ethernet PICtail

Плата Дочери Ethernet PICtail заполнена ENC28J60, разъемом RJ-45 (со встроенными магнитными), и несколькими другими компонентами необходимыми для операции Ethernet. Это обеспечивает 10-связь Base-T Ethernet для любой демонстрационной платы с разъемом PICtail. Эта дочерняя плата в основном, заменена PICDEM.net 2 ( смотри страницу 69) для

отладки приложений Ethernet, использовавшего PIC18. Посетите сайт веб Микрокристалла, чтобы рассматривать Страницу Продукта Ethernet PICtail.

Ethernet PICtail Плюс Дочерняя Плата

Ethernet PICtail Плюс Дочерняя Плата - PICtail Плюс версия Дочерней Платы Ethernet PICtail. Это допускает the свяжите ENC28J60 на любую демонстрационную плату с PICtail Плюс разъем. Посетите сайт веб Микрокристалла, чтобы рассматривать Ethernet PICtail Плюс Страница Продукта Дочерней Платы.

Быстрый 100Mbps Ethernet PICtail Плюс Дочерняя Плата

Быстрый 100Mbps Ethernet PICtail Плюс Дочерняя Плата обеспечивает метод для испытания и демонстрирующего Диспетчер ENC624J600 Ethernet. Плата разработана для гибкости и может быть подключено к PICtail или PICtail плюс разъем. Кроме того, предназначено допускать использование любого из параллельных или режимов связи SPI отображенной на ENC624J600 на PICtail Плюс разъем. Эта дочерняя плата обеспечивает 10/100-функциональное назначение Base-T. Посетите сайт веб Микрокристалла, чтобы рассматривать Быстрый 100Mbps Ethernet PICtailTM Плюс Страница Продукта Дочерней Платы.

Микрокристалл 802.11b WiFi PICtail Плюс Дочерняя Плата

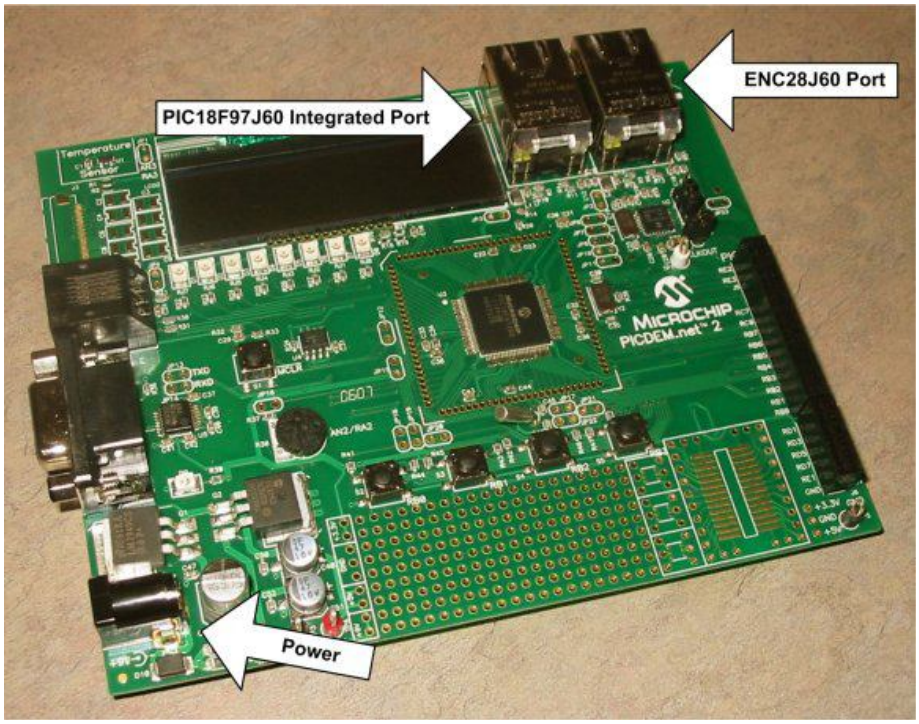
Micorchip 802.11b WiFi PICtail Плюс Дочерняя Плата - демонстрационная плата для оценки связности Wi-Fi на плате с PICtail или PICtail Плюс разъем. Плата представляет модуль Микрокристалла MRF24WB0MA, который включает приемо-передатчик Wi-Fi и связанные элементы цепи. Посетите Сайт Микрокристалла Web, чтобы рассматривать больше информации о Беспроволочных Решениях и 802.11b WiFi PICtail Product Page..

### 6.1.2 PICDEM.net 2

Посетите сайт веб Микрокристалла, чтобы рассматривать Страницу Продукта PICDEM.net 2.

Плата разработки PICDEM.net 2 приходит с прошитым PIC18F97J60 со встроенным диспетчером Ethernet, а также с автономным диспетчером ENC28J60 Ethernet. Встроенный диспетчер связывается налево гнездо Ethernet (ближайшее на LCD), и автономная часть связывается направо один. По умолчанию стек сконфигурирован, чтобы использовать встроенного диспетчера, так что левый порт должен быть подключен к сетевому кабелю. Никакая другая конфигурация платы не нужна.

Руководство Пользователя, которое отправляло с этим бортом разработки может ссылаться на более старую версию Стека TCP/IP. Этот документ корректирует большинство этой документации для версии 5.42.02.



По умолчанию, эта плата использует ENC28J60 или PIC18F97J60 для связи Ethernet. Тем не менее, соединяя Быстрый Ethernet PICtail в разъем PICtail на плате, Вы можете использовать это, чтобы тестировать ENC624J600. Для того, чтобы использовать Быстрый Ethernet PICtail, включите это как показано в снимке, с заголовком J4 на PICtail включенное в разъем J5 на демонстрационную плату.

Быстрый Ethernet PICtail предназначен использовать шину связи SPI когда связано через заголовок PICtail, так что установочные параметры перемычки неиспользованные в этой конфигурации, с одним исключением: перемычка JP2 на PICtail, помеченная ISENSE, должна быть шунтирована. pre- Скомпилированный и pre-сконфигурировавшее версии демонстрационного, который переписывается в эту установку уже записаны, чтобы приспосабливаться функциональное назначение ENC624J600; для ручной информации конфигурации, смотри ENCX24J600 конфигурации ( смотри страницу 144) страница.

**Использование Микрокристалла 802.11b WiFi PICtail**

PICDEM.net 2 Может быть использован, чтобы отлаживать беспровдное соиденение функциональное назначение соединяя PICtail как показ в снимке, с заголовком J1 на PICtail включенное в разъем J5 на демонстрационную плату.

Примечание если перемычка JP3 существует, она должна быть шунтирована между штырьками 2 и 3 когда использовано на этой платформе разработки.
Как только ваши аппаратные средства будут сконфигурированы, Вы можете запрограммировать вашу плату с вашим предпочтительным демонстрационным проектом. Следующих несколько тем ( смотри страницу 75) в ПРИСТУПАЮЩЕЙ секции этого файла

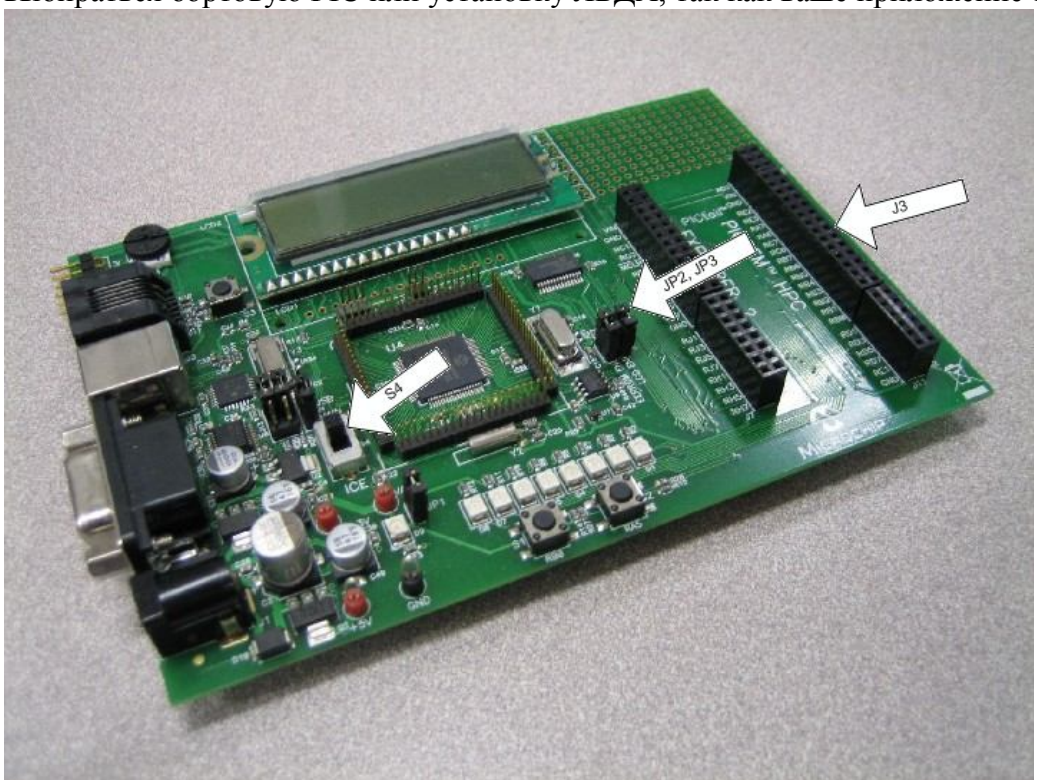
подсказки обеспечивать консультацию для установки общего демонстрационного приложения TCPIP.

### 6.1.3 PIC18 Explorer

Посетите сайт веб Микрокристалла, чтобы рассматривать Страницу Продукта PIC18 Explorer.

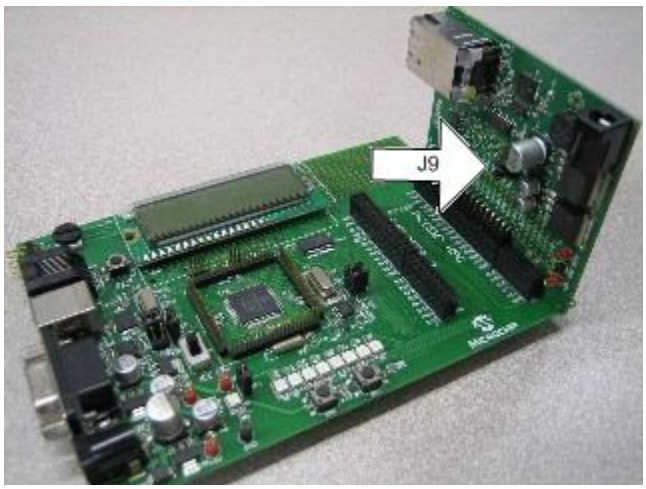
PIC18 Explorer - для оценки высоких штырьковых-microcontrollers счета PIC18. Соединяя TCP/IP дочерней платы этому, Вы можете протестировать и отлаживать функциональное назначение Ethernet с рядом PIC18s. семейство PIC18F97J60 включает встроенное периферийное устройство Ethernet, делающее это по умолчанию дешевую, платформу разработки PIC18 Ethernet; PICDEM.net 2 ( смотри страницу 69), - рекомендуемая плата разработки для этой части.

При использовании PIC18 Explorer, гарантируйте, что перемычки JP2 и JP3 шунтированы, чтобы приспосабливаться LCD и EEPROM, и ключ S4 сконфигурирован, чтобы правильно выбираться бортовую PIC или установку ЛЬДА, так как ваше приложение требуется.




**Использование Ethernet PICtail**

В отличие от PICDEM.net 2, PIC18 Explorer не включает ENC28J60 на на борту. Для того, чтобы приспосабливаться тестируя и отлаживая использование ENC28J60, Вы должны соединить ( смотри страницу 171) Ethernet PICtail, как показано в снимке (включите заголовок J2 в разъем J3 на демонстрационной плате).




При использовании этой конфигурации, коротких штырьков 2 и 3 в перемычке J9, чтобы указывать, что PIC18 Explorer обеспечивает полномочие 5V. pre- Скомпилированный и pre-сконфигурировавшее версии демонстрационного, который переписывается в эту установку уже записаны, чтобы приспосабливаться функциональное назначение ENC28J60; для ручной информации конфигурации, смотри ENC28J60 конфигурации ( смотри страницу 143) страница.

**Использование Быстрый Ethernet PICtail**

Соединяя Быстрый Ethernet PICtail в разъем PICtail на плате, Вы можете использовать это, чтобы тестировать ENC624J600. Для того, чтобы использовать Быстрый Ethernet PICtail, включите это как показано в снимке, с заголовком J4 на PICtail включенное в разъем J3 на демонстрационную плату.

Быстрый Ethernet PICtail предназначен использовать шину связи SPI когда связано через заголовок PICtail, так что установочные параметры перемычки неиспользованные в этой конфигурации, с одним исключением: перемычка JP2 на PICtail, помеченная ISENSE, должна быть шунтирована. pre- Скомпилированный и pre-сконфигурировавшее версии демонстрационного, который переписывается в эту установку уже записаны, чтобы приспосабливаться функциональное назначение ENC624J600; для ручной информации конфигурации, смотри ENCX24J600 конфигурации ( смотри страницу 144) страница.

**Использование Микрокристалла 802.11b WiFi PICtail**

PIC18 Explorer может быть использован, чтобы отлаживать беспроволочное функциональное назначение соединяя PICtail как показ в снимке, с заголовком J1 на PICtail включенное в разъем J3 на демонстрационную плату.

Примечание если перемычка JP3 существует, она должна быть шунтирована между штырьками 2 и 3 когда использовано на этой платформе разработки.

Как только ваши аппаратные средства будут сконфигурированы, Вы можете запрограммировать вашу плату с вашим предпочтительным демонстрационным проектом. Следующих несколько тем ( смотри страницу 75) в ПРИСТУПАЮЩЕЙ секции этого файла подсказки обеспечивать консультацию для установки общего демонстрационного приложения TCPIP.

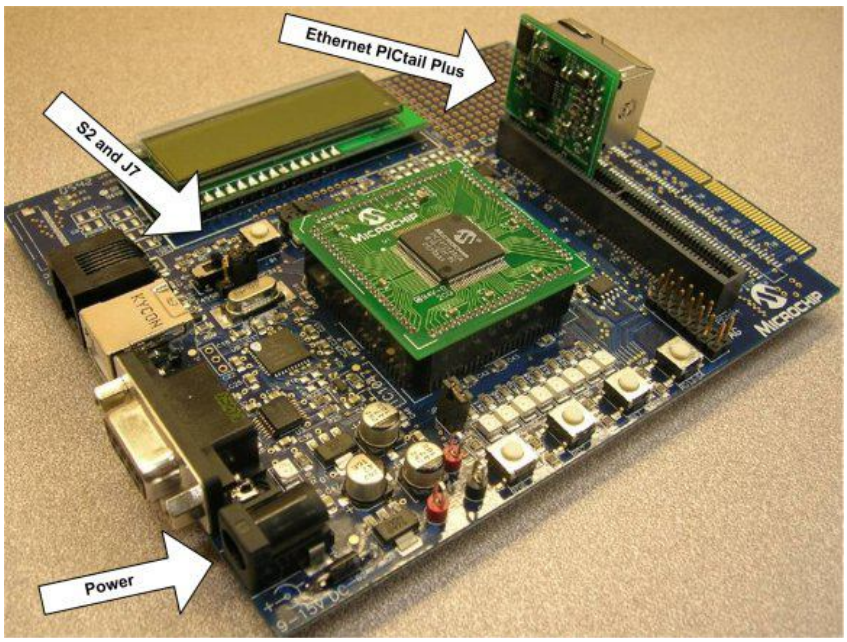
### 6.1.4 Explorer 16 и Комплект Стартера PIC32

Посетите сайт веб Микрокристалла, чтобы рассматривать Страницу Продукта Explorer 16 и Страница Продукта Комплекта Стартера PIC32.

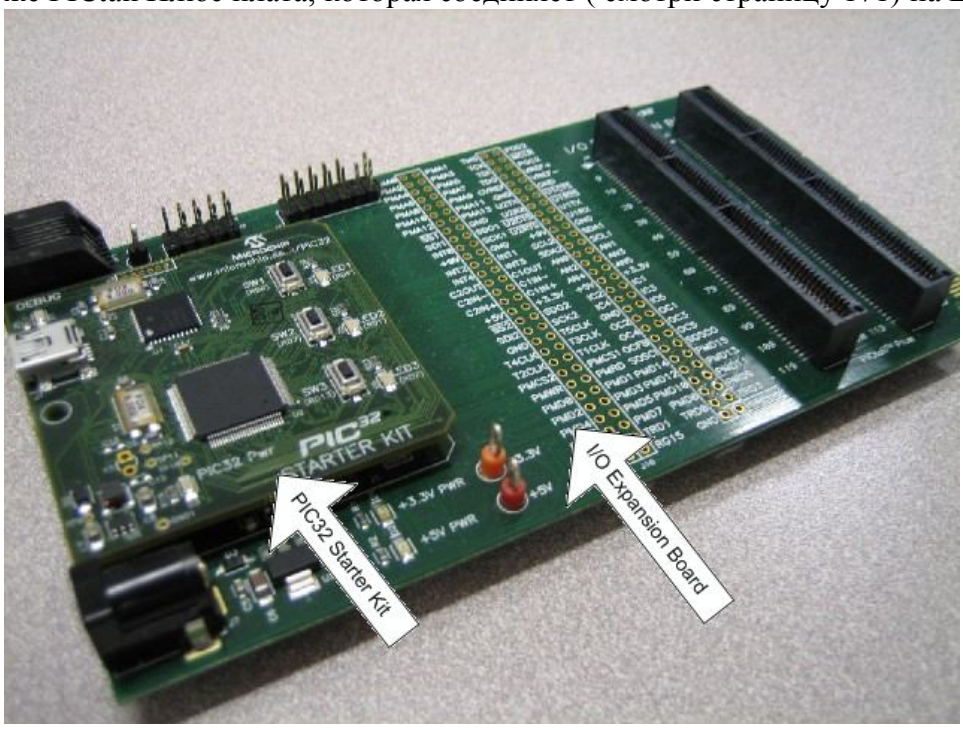
Плата Explorer 16 является универсальной демонстрационной и платой разработки для 16-бита и 32-битовые части. Может быть расширено для TCP/IP поддержки, использовавшей Ethernet PICtail Плюс, Быстрый 100Mbps Ethernet PICtail Плюс, или 802.11b WiFi PICtail Плюс дочерняя плата.

Перед использованием Explorer 16, проверьтесь, что:

1. Ключ S2 выбирается PIM
2. Перемычка J7 выбирается PIC24 (даже если этикетка читает PIC24, эта установка перемычки выбирается программирование сигналов на любой PIC на Explorer 16).




Комплект Стартера PIC32 выполняет аналогичную функцию для 32- битовых частей PIC32. Используя Плату Расширения PIC32 B/B Вы можете соединить ( смотри страницу 171) тот же PICtail Плюс плата, которая соединяет ( смотри страницу 171) на Explorer 16.




**Использование Ethernet PICtail Плюс**

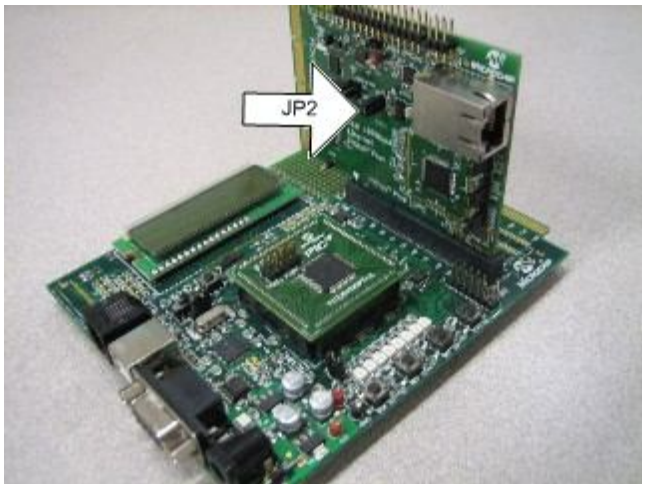
Чтобы приспосабливаться испытание и отладку ENC28J60 на этой плате, Вы должны соединить ( смотри страницу 171) Ethernet PICtail Плюс, как показано в снимке (включите заголовок J2 в верхний карта-краевой разъем J5 (Explorer 16) или J4 (Плата Расширения B/B)). Отметьте, что для немного демонстрационные, Ethernet PICtail Плюс должно быть включено в центр карта-краевой разъем PICtail Плюс разъем, чтобы использовать модуль SPI2. Смотри Демонстрационную Таблицу Совместимости ( смотри страницу 82) более подробно.

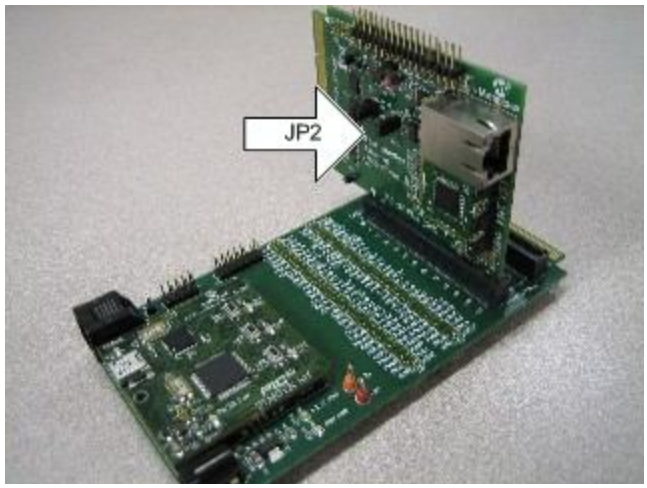
http://www.microchip.com/

pre- скомпилированный и pre-сконфигурировавшее версии демонстрационного, который переписывается в эту установку уже записаны, чтобы приспосабливаться функциональное назначение ENC28J60; для ручной информации конфигурации, смотри ENC28J60 конфигурации ( смотри страницу 143) страница.

**Использование Быстрый Ethernet PICtail Плюс**

Соединяя Быстрые 10/100 Ethernet PICtail Плюс на PICtail Плюс разъем на вашей плате, Вы можете использовать это, чтобы тестировать ENC624J600. Быстрый Ethernet PICtail Плюс может быть использовано этой платой в или последовательный (SPI) или параллельный режим связи. Для последовательного режима, соедините заголовок ( смотри страницу 171) J2 дочерний плата в разъем J5 (Explorer 16) или J4 (Плата Расширения B/B), как видно в снимках. Действуя в последовательном режиме, перемычки на Быстром Ethernet PICtail неиспользованные, с одним исключением: перемычка JP2 на PICtail, помеченная ISENSE, должна быть шунтирована.



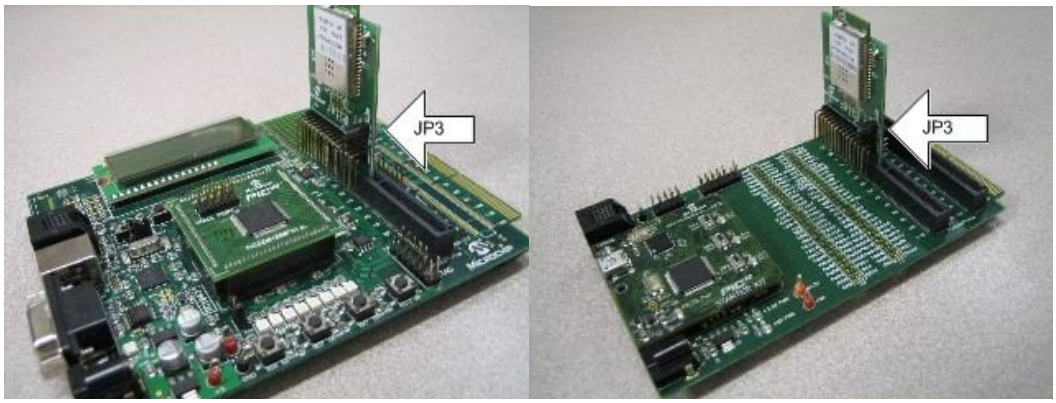




Для того, чтобы использовать Быстрый Ethernet PICtail Плюс плата в параллельном режиме, включите заголовок J1 в разъем J5 Explorer 16 или J4 Плата Расширение B/B, как видно в снимках. В этой конфигурации, перемычки должны быть шунтированы или открытое соответствовать в параллельный режим связи, использованный. Матрица, очерчивающая какие связи перемычки должна быть сделана для перемычек пометивших PSPCFG3, PSPCFG2, PSPCFG1&4, PMA на AD, и PMA на напечатан на обратной стороне дочерней платы.

pre- Скомпилированный и pre-сконфигурировавшее версии демонстрационного, который переписывается в эту установку уже записаны, чтобы приспосабливаться функциональное назначение ENC624J600; для ручной информации конфигурации, смотри ENCX24J600 конфигурации ( смотри страницу 144) страница.

**Использование Микрокристалла 802.11b WiFi PICtail**

Explorer 16 и Комплект Стартера PIC32 может быть использован, чтобы отлаживать беспроволочное функциональное назначение соединяя PICtail как показ в снимках, с заголовком J2 на PICtail включенное в верхний слот разъема J5 (Explorer 16) или J4 (Плата Расширения B/B) на демонстрационной плате.




если перемычка JP3 существует, она должна быть шунтирована между штырьками 1 и 2 когда использовано на этой платформе разработки.

Как только ваши аппаратные средства будут сконфигурированы, Вы можете запрограммировать вашу плату с вашим предпочтительным демонстрационным проектом. Следующих несколько тем ( смотри страницу 75) в ПРИСТУПАЮЩЕЙ секции этого файла подсказки обеспечивать консультацию для установки общего TCP/IP демонстрационного приложения.

**6.1.5 Платы Разработки PIC24FJ256DA210**

Посетите сайт веб Микрокристалла, чтобы рассматривать Страницу Продукта Комплекта Разработки PIC24FJ256DA210.

Комплект Разработки PIC24FJ256DA210 является низким уровнем стоимым и эффективного комплекта разработки, чтобы оценивать характеристики и исполнение PIC24FJ256DA210 со встроенной графикой, mTouch  и USB.

Вы можете добавить сетевую связность к этой демонстрационной плате включая Ethernet PICtail Плюс, Быстро Ethernet PICtail Плюс, или Микрокристалл 802.11b WiFi PICtail в PICtail Плюс разъем на демонстрационной плате. Метод для занятия это функционально идентично методу использованному для Explorer 16 и Комплекта Стартера PIC32 ( смотри страницу 72).

**6.2 Программируя и Сначала Работать**

Как только аппаратные средства будут сконфигурированы ( смотри страницу 68), Вы готовые запрограммировать устройство впервые.

**Установка Проекта**

Откройте сеанс MPLAB IDE.

1. Из меню "Файла", выбор "Импорта." Просмотрите Precompiled Шестнадцатеричный subdirectory в вашем демонстрационном директории проекта и выбора the * шестнадцатеричного файла., которые соответствуют вашей аппаратной установке. Шестнадцатеричные файловые имена описывают аппаратные средства для которых файл компилирован. Например, файл "Решений Микрокристалла v2011-06-02\TCPIP\Demo App\Precompiled Hex\C18-PICDN2_ETH97 18F97J60.hex" переписывается в общей TCP/IP Демонстрационной заявки на PIC18F97J60 в PICDEM.net 2, использовании PIC's внутреннего модуля Ethernet. Документ, перечисляющий сокращения использовался в шестнадцатеричном файле и файловые имена проекта доступно в Решениях Микрокристалла v20xx-xx-xx/директорий Подсказки.
2. Проверьте, что целевой выбор процессора MPLAB IDE и сценарий компоновщика (если один представляет), соответствуют части на вашей демонстрационной плате (экс: PIC18F97J60).
Отметьте, что проекты и исходный код использованные, чтобы формировать каждый шестнадцатеричный файл присутствовать в директории проекта. Аппаратные и файлы конфигурации микропрограмм использованные, чтобы формировать каждый проект включены в subdirectory Configs.

**Программирование**

Выберитесь вашего программиста устройства из меню Программиста в MPLAB, затем используйте Программную сокращенную кнопку или опция меню Программы, чтобы программировать код, который Вы импортировали на вашу плату.

**Клиринг EEPROM**

TCP/IP Стека загружает сетевые установочные параметры конфигурации (как например, главное имя, адрес MAC, по умолчанию статические адресы IP, SNMP, сетевого имени WiFi (SSID), и т.п.) во внешнем EEPROM на на борту. Демонстрационный проект обнаружится если значение по умолчанию изменено в EEPROM, и если так, используйте новые величины. Если нет, демонстрационная воля (завещание) использует значение по умолчанию сконфигурированное в TCPIPConfig.h и WF_Config.h. Контрольные суммы загруженные в EEPROM использованы, чтобы определяться если структуры загруженные в EEPROM в силе. Вручную клиринг EEPROM допустит демонстрационный, чтобы продолжать используя по умолчанию установочные параметры.

Используйте следующую процедуру, чтобы очищать EEPROM:

1. Убедитесь что плата разработки запрограммирована и не в отладочном режиме
2. Разъедините MPLAB  ICD 2/3 или MPLAB REAL ICETM из на борту
3. Нажатие и трюм BUTTON0 (RD13/S4 на Explorer 16 или RB3/S5 на PICDEM.netTM 2)
4. Нажмите и выпускайте кнопку MCLR
5. Продолжите удерживать BUTTON0 пока несколько СВЕТОДИОДОВ не высветятся указывая, что EEPROM очищен. Это берет около 4 секунд.
6. Версия BUTTON0
7. Нажатие и версия MCLR снова, чтобы восстанавливать программное обеспечение Как только Вы увидите LED0 мигания (право-наиболее ПРОВЕДЕННЫЙ), программное обеспечение выполняет и готовый для использования.

Если Вы используете MRF24WB0M WiFi PICtail, вам понадобится конфигурировать вашу беспроволочную точку доступа ( смотри страницу 76) сначала. Для всех устройств Ethernet, Соедините вашу Плату Разработки ( смотри страницу 77) в вашу сеть.

### 6.3 Конфигурируем вашу точку доступа WiFi

Чтобы запускать Wi-Fi демонстрационные с MRF24WB0M PICtail, Вы будете также нужно устанавливать беспроволочную точку доступа. Так как пример, это руководство пройдет через установку точки доступа Linksys WRT54G2.

ГРАФИЧЕСКИЙ интерфейс пользователя Окна просмотра Точки Доступа

Linksys, Вместе с много другими популярными марками маршрутизатора, использует встроенный webserver в маршрутизаторе, чтобы управлять сетью (как телеграфированное так и телеграмму). Пожалуйста обратитесь к документации, которая приходила с вашим маршрутизатором более подробно в конфигурации и установке. Для списка известных совместимых маршрутизаторов, ссшлитесь на секцию "Совместимости Точки Доступа" ( смотри страницу 700). Для того, чтобы приобретать доступ к этой странице веб, вам понадобится указывать ваше окно просмотра на http://192.168.1.1. По умолчанию, имя пользователя области оставлено пробел, и пароль является администрацией.

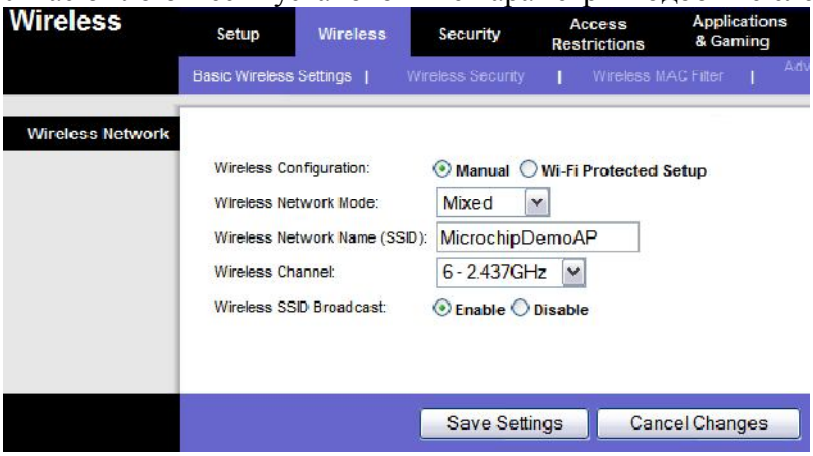
Вдоль верха webpage, там должно быть много таб. для всех других характеристик точки доступа. Одно из таб. должны читать "Телеграмму". После щелчка таб., Вы будете преподнесены Wi-Fi защитившее страницу установки. Вам понадобится щелкать ручное таб., чтобы быть способным ввести ваши собственные беспроволочные установочные параметры, чтобы соответствовать демонстрационному.

out of блок демонстрационный ищет AP с следующими параметрами (отметьте, что SSID - случай чувствительный):

| | |
|---|---|
| | |
| | |

БЕЗОПАСНОСТЬ SSID MicrochipDemoAP Никакого Канала Также 1, 6, или 11

У вас было бы есть установочные параметры подобные следующему:




Как только сеть явится установкой, Вы можете подключить ваше устройство к сети ( смотри страницу 77).

**6.4 Соединяясь в Сеть**

Все устройства в TCP/IP сети должны быть назначены адрес IP ( смотри страницу 148). Поскольку адрес MAC ( смотри страницу 147) - аппаратный адрес устройства, адрес IP является программным адресом. DHCP Темы (Динамический Главный Протокол Конфигурации), позволяет назначение, чтобы происходить автоматически (для более адреса информационные и опции конфигурации, смотри Адресы ( смотри страницу 147)).

Демонстрационное приложение приходит с как сервером DHCP так и клиент DHCP конфигурировал. Это позволяет плату, чтобы соединять ( смотри страницу 171), чтобы большинство сетей без конфигурации. Если свободный порт Ethernet доступен в соседнем маршрутизаторе, ключе, или пластине стены, плата может связываться непосредственно используя любой стандарт прямо-через кабель Ethernet. Под этой конфигурацией, плата попытается получать адрес IP из вашего сетевого сервера DHCP.

Если этот метод не возможный, перевернутый кабель Ethernet может быть использован, чтобы соединять ( смотри страницу 171) плата непосредственно в порт Ethernet. Используя эту конфигурацию, плата подействует как собственный сервер DHCP и назначит единственный адрес IP в компьютер. ( Быстрый 100Mbps Ethernet PICtail Плюс и немного

новый PCs не требует специальный перевернутый кабель, так что любой кабель Ethernet может быть used.)

Подключите плату разработки к сети и ждите ссылку LED на гнезде Ethernet, чтобы зажигать по. Плата - теперь на сетевом и способная передаваться с другими устройствами.

Если ссылка LED на гнезде Ethernet не зажигается, ваша плата не может связаться в сеть. Проверьте, что Вы выбрали соответствующий кабель и пробуйте переключиться из прямо-вплоть до перевернутого кабеля, или наоборот.

Теперь, когда плата - online, Вы можете Загрузить Демонстрационные Страницы Web ( смотри страницу 78).

**6.5 Загружая Страницы Web**

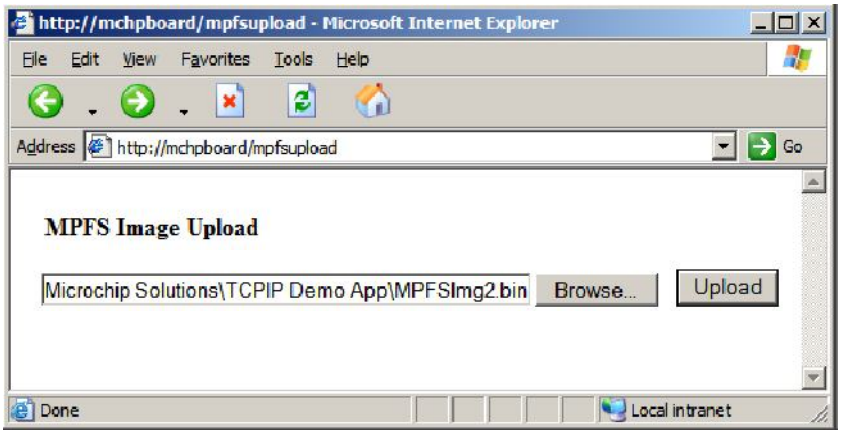
Страницы Web загружены как MPFS2 образа ( смотри страницу 271). Этот образ может быть установлен в или внешняя не-летучая память ( смотри страницу 142) (EEPROM или Вспышка SPI), или во внутренней Вспышке microcontroller's программной памяти. Для этого примера, чип EEPROM (25LC256) на вашей демонстрационной плате будет запрограммирован pre- смонтированным образом MPFS2 BIN. Эта позиция может быть изменена через компиляцию-время опции в TCPIPConfig.h.

Целевое приложение на плате разработки должно выполняться на этой процедуре, чтобы работать. Убедитесь право наиболее статус LED мигает.

Каждый шестнадцатеричный файл сконфигурирован, чтобы обеспечивать Хозяина Называть в честь вашей платы разработки. Это будет именем, которым ваша плата доступна. В по умолчанию шестнадцатеричных файлах, главное имя - mchpboard, так что Вы плата может быть доступна в http://mchpboard. Это главное имя использует Услугу Имени NetBIOS ( смотри страницу 290). Только доступно в вашей локальной подсети, и не будет доступно из Internet. Отметьте, что эта услуга не поддерживана всеми операционными системами. Если у вас есть трудность получая вашу плату, попробуйте использовать адрес IP показанный на экране LCD взамен (напр иметь доступ к плате в http://192.168.1.101). Вы можете также определить что IP адресует используя Микрокристалл TCP/IP Открывателя ( смотри страницу 66).

Откройте окно просмотра веб и имейте доступ к плате в http://mchpboard/mpfsupload. Эта форма допустит страницы веб загруженные на устройство, которое нужно корректироваться. Если Вы mistype это URL, плата обеспечит по умолчанию страницу ошибки HTTP 404 ссылкой на MPFS Загружать страницу. Это умолчание 404 страниц не появится если Вы сконфигурировали бы ваше окно просмотра, чтобы аннулировать заказные страницы ошибки

(напр проверяя "Показ дружественных сообщений ошибки HTTP" в Internet Explorer опций 7's меню internet). Выберитесь файл MPFSImg2.bin из TCPIP\Demo Appfolder как показано ниже.
Этот метод коррекции только доступен при использовании внешней памяти.

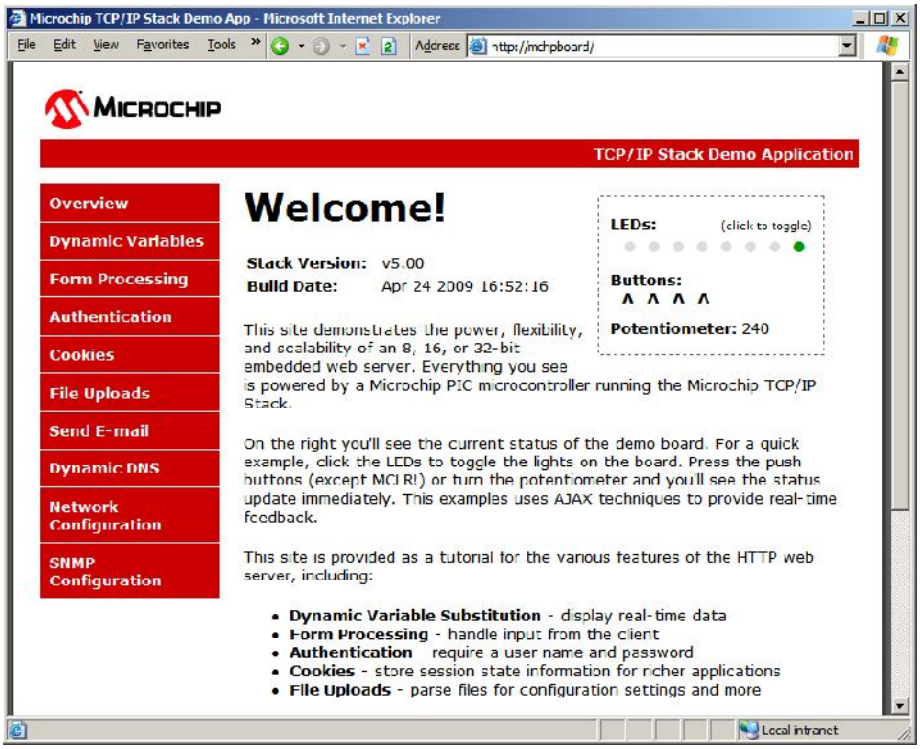



Когда кнопка Upload щелкнута, образ MPFS послан на плату и запрограммирован в EEPROM. Так как это случается, деятельность ПРОВЕДЕННАЯ на гнезде Ethernet мигнет. Как только окно просмотра сообщит, что загружать завершился, щелкните ссылку предусмотренную в пределах сообщения статуса, чтобы иметь доступ к страницам веб платы.

Вы можете теперь Иметь доступ к Демонстрационному Приложению ( смотри страницу 79).

### 6.6 Получая Демонстрационное Приложение

Плата теперь доступна в главном имени mchpboard или в адресе платы IP. Когда доступно в окне просмотра веб, коррекция в реальном времени элементов управления платы отображена. Демонстрационное приложение покрасуется несколько характеристик, и объяснит как, чтобы модифицировать страницы веб и приложения, чтобы подходить различным потребностям.




Если Вы пытаетесь иметь доступ к Сетевой Конфигурации или страницам веб Конфигурации SNMP из красного меню слева, Вы будете подсказаны для имени пользователя и пароль. По умолчанию имя пользователя является "администрацией" и по умолчанию пароль является

"микрокристаллом". Более информация доступна в Аутентификации веб ( смотри страницу 89) страница, или в аутентификации сервера HTTP2 подсказки ( смотри страницу 236) тема.

Некоторые характеристики по умолчанию демонстрационного приложения не могут быть доступны на определенных аппаратных платформах. Более подробно, смотри TCPIP Демонстрационных Характеристик Прил. Аппаратной Платформой темы ( смотри страницу 86). Для информации соберитесь использовать каждую характеристику TCP/IP Демонстрационного Приложения, обращаться к подтемам в TCP/IP Демонстрационных Прикладных Демонстрационных Модулей темы ( смотри страницу 87).

Как только Вы завершили изучая демонстрационное приложение, Вы можете приступить в Стек API секции ( смотри страницу 155), чтобы узнавать более о стеке и начинать проявлять ваше собственное приложение.

Если Вы изучаете демонстрационные приложения Wi-Fi и хотитесь установить безопасность, Вы можете получить больше информации о странице безопасности WLAN ( смотри страницу 80).

### 6.7 Компонуя Безопасность WiFi

MRF24WB0M Может быть сконфигурирован, чтобы соединять ( смотри страницу 171) в беспроволочные сети с шифрованием приспособленным. MRF24WB0M Поддерживает WEP (40- бит и 104-бит), а также WPA (TKIP) и WPA2 (TKIP/AES).

**Режимы Безопасности Устройства**

Установочные параметры Безопасности для MRF24WB0M расположены в файле WF_Config.h. Чтобы приспосабливаться безопасность, представляет #define определение препроцессора для MY_DEFAULT_WIFI_SECURITY_MODE должно быть определено как одна из следующих опций:

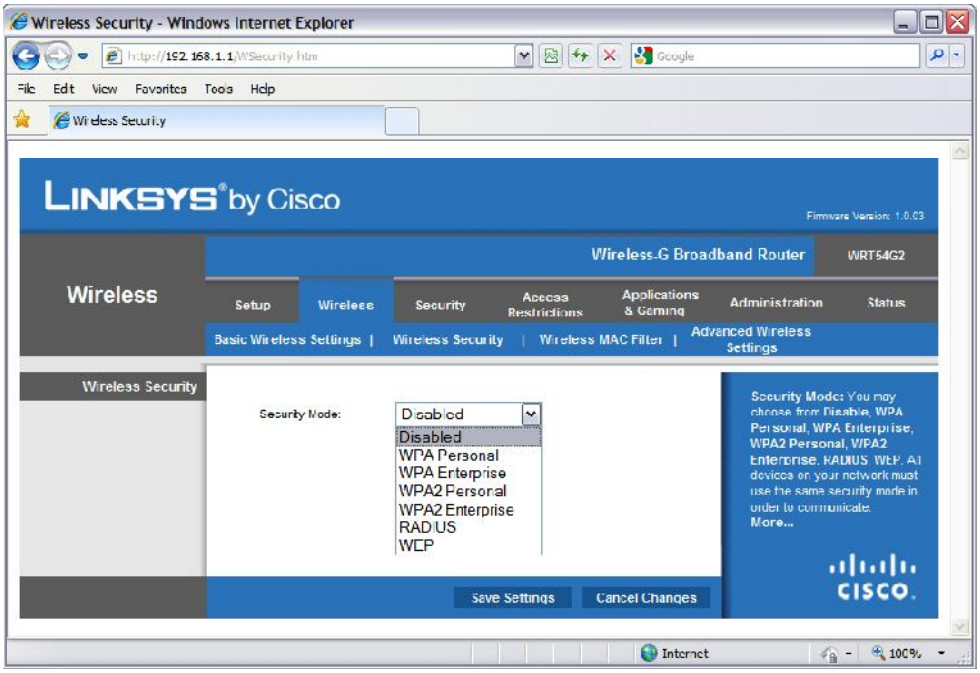
| | |
|---|---|
| WF_SECURITY_WEP_40 | 40-битовая безопасность WEP. Это приравнивается в 5 символов ASCII или 10 шестнадцатеричных цифр. MY_DEFAULT_WEP_KEYS_40 содержит вплоть до четырех клавиш, которые могут быть запрограммированы (умолчание является клавишей 0). |
| WF_SECURITY_WEP_104 | 104-битовая безопасность WEP. Это приравнивается в 13 символов ASCII или 26 шестнадцатеричных цифр. MY_DEFAULT_WEP_KEYS_104 содержит вплоть до четырех клавиш, которые могут быть запрограммированы (умолчание является клавишей 0). |
| WF_SECURITY_WPA_WITH_KEY<br>WF_SECURITY_WPA2_WITH_KEY | Использует 32 байта в MY_DEFAULT_PSK в качестве ключа для подключения к сети. |

| | |
|---|---|
| WF_SECURITY_WPA_AUTO_WITH_KEY | Эти значения генерируются из хэша имени SSID и WPA<br>фраза. Для целей демонстрации, 32-байт<br>MY_DEFAULT_PSK в WF_Config.h соответствовать SSID из<br>"MicrochipDemoAP" и пароль<br>"Microchip 802.11 Секрет PSK Пароль " |
| WF_SECURITY_WPA_WITH_PASS_PHRASE<br>WF_SECURITY_WPA2_WITH_PASS_PHRASE<br>WF_SECURITY_WPA_AUTO_WITH_PASS_PHRASE | Поручает MRF24WB0M генерировать PSK 32 байт с помощью<br>SSID и парольная фраза. По умолчанию в WF_Config.h соответствует<br>SSID из "MicrochipDemoAP" и фраза<br>"Microchip 802,11 секрет<br>PSK Пароль ". Обратите внимание, что это занимает около 30 секунд, MRF24WB0M вычислить это значение [1]. |

**Примечание**: Некоторая попытка маршрутизаторов, чтобы увеличивать произвольную природу клавиши WEP добавляя дополнительный слой, который преобразовает ASCII passphrase на шестнадцатеричную клавишу. MRF24WB0M PICtail Не Потребует шестнадцатеричную клавишу, никакой материал какой путь сгенерировано.

**Установочные параметры Безопасности Точки Доступа**

Точка доступа будет также должно быть измениться, чтобы соответствовать тем же установочным параметрам безопасности. Беспроволочные установочные параметры безопасности могут быть обнаружены в таб. "Беспроволочной Безопасности" под основой таб. "Телеграммы" (пример показывает Linksys WRT5G2). Капля-вниз блок для безопасности имеет все другие опции безопасности. Отметьтесь, что для WPA/WPA2, уровни персональной безопасности только опор MRF24WB0M (по сравнению с предприятием, которое не поддерживано).




[1]: Как только 32- байт PSK будет вычислен, он может быть извлечен хозяином из MRF24WB0M., хозяин может затем сохранять эту клавишу во внешнюю не-летучую память. В будущих попытках связи, хозяин может запрограммировать MRF24WB0M с опциями WF_SECURITY_*_WITH_KEY, обеспечивать сохраненную клавишу, и не должен ожидать 30 секунд, чтобы пересоединяться в сеть.

**Сгенерировавшее Pre PSK**

У вас также есть опция на pre-генерирует PSK и используешь 32- байт PSK непосредственно в исходном коде. Один удобное средство, чтобы генерировать PSK может быть обнаружен online в Фонде Wireshark http://www.wireshark.org/tools/wpa-psk.html. Wireshark website Может сгенерировать ожидаемую 32- байтовую клавишу PSK с именем SSID и passphrase. Вы можете затем использовать эти величины в переменной MY_DEFAULT_PSK в TCPIPConfig.h.

**Демонстрационных Информации**

Эта секция описывает Микрокристалл TCP/IP Демонстрационных проектов, включая информацию о демонстрационной-аппаратной совместимости. Для информации соберитесь загружать и конфигурировать демонстрационные, пожалуйста обратитесь к Getting Started.

**7.1 Демонстрационных Таблицы Совместимости**

Каждый демонстрационный проект стека приходит с несколько встроенными, протестированными конфигурациями. Смонтированные шестнадцатеричные файлы для каждого демонстрационного проекта доступны в Precompiled Шестнадцатеричный subdirectory в этой папке проекта демонстрационный (то есть файлы для Демонстрационного Прил. расположены в <устанавливать Решения directory>\Microchip v20xx-xx-xx\TCPIP\Demo App\Precompiled Шестнадцатеричный). Эта секция определит комбинации

демонстрационной платы, процессоров, MAC/PHY слоев, и связь везет в автобусе, что установлена, чтобы работать по умолчанию.

### TCPIP Demo App. ( смотри страницу 86)

**TCPIP Demo App (see page 86)**

| Demo Board | Processor | MAC/PHY Layer | Comm. Bus | Notes |
|---|---|---|---|---|
| PIC18 Explorer | 18F87J11 | ENC28J60 | SPI | Requires silicon revision A4 or later. |
| PIC18 Explorer | 18F87J11 | ENCX24J600 | SPI | |
| PIC18 Explorer | 18F87J11 | MRF24WB0M | SPI | |
| PIC18 Explorer | 18F87J50 | ENC28J60 | SPI | |
| PIC18 Explorer | 18F87J50 | ENCX24J600 | SPI | |
| PIC18 Explorer | 18F87J50 | MRF24WB0M | SPI | |
| PIC18 Explorer | 18F8722 | ENC28J60 | SPI | |
| PIC18 Explorer | 18F8722 | ENCX24J600 | SPI | |
| PIC18 Explorer | 18F8722 | MRF24WB0M | SPI | |
| PICDEM.net 2 | 18F97J60 | ETH97J60 | - | |
| PICDEM.net 2 | 18F97J60 | ENC28J60 | SPI | |
| PICDEM.net 2 | 18F97J60 | ENCX24J600 | SPI | |
| PICDEM.net 2 | 18F97J60 | MRF24WB0M | SPI | |
| Explorer 16 | 24FJ128GA010 | ENC28J60 | SPI | |
| Explorer 16 | 24FJ128GA010 | ENCX24J600 | SPI | |
| Explorer 16 | 24FJ128GA010 | MRF24WB0M | SPI | |
| Explorer 16 | 24FJ128GA010 | ENCX24J600 | PSP 5 Indirect | |
| Explorer 16 | 24FJ256GA110 | ENC28J60 | SPI2 | |
| Explorer 16 | 24FJ256GA110 | ENCX24J600 | SPI2 | |
| Explorer 16 | 24FJ256GA110 | ENCX24J600 | PSP 5 Indirect | |
| Explorer 16 | 24FJ256GA110 | MRF24WB0M | SPI2 | |

| | | | | |
|---|---|---|---|---|
| Explorer 16 | 24FJ256GB110 | ENC28J60 | SPI | |
| Explorer 16 | 24FJ256GB110 | ENCX24J600 | SPI | |
| Explorer 16 | 24FJ256GB110 | ENCX24J600 | PSP 5 Indirect | |
| Explorer 16 | 24FJ256GB110 | MRF24WB0M | SPI | |
| Explorer 16 | 24FJ256GB210 | ENC28J60 | SPI | |
| Explorer 16 | 24FJ256GB210 | ENCX24J600 | SPI | |
| Explorer 16 | 24FJ256GB210 | ENCX24J600 | PSP 5 Indirect Bitbang | |
| Explorer 16 | 24FJ256GB210 | MRF24WB0M | SPI | |
| Explorer 16 | 33FJ256GP710 | ENC28J60 | SPI | |
| Explorer 16 | 33FJ256GP710 | ENCX24J600 | SPI | |
| Explorer 16 | 33FJ256GP710 | ENCX24J600 | PSP 5 Indirect | |
| Explorer 16 | 33FJ256GP710 | MRF24WB0M | SPI | |
| Explorer 16 | 32MX360F512L | ENC28J60 | SPI | |
| Explorer 16 | 33EP512MU810 | ENC28J60 | SPI2 | |
| Explorer 16 | 24EP512GU810 | ENC28J60 | SPI2 | |
| Explorer 16 | 32MX360F512L | ENCX24J600 | SPI | |
| Explorer 16 | 32MX360F512L | ENCX24J600 | PSP 5 Indirect | |
| Explorer 16 | 32MX360F512L | ENCX24J600 | PSP 9 | |
| Explorer 16 | 32MX360F512L | MRF24WB0M | SPI | |
| Explorer 16 | 32MX460F512L | ENC28J60 | SPI | |
| Explorer 16 | 32MX460F512L | ENCX24J600 | SPI | |
| Explorer 16 | 32MX460F512L | MRF24WB0M | SPI | |
| Explorer 16 | 32MX795F512L | ENC28J60 | SPI | |
| Explorer 16 | 32MX795F512L | ENCX24J600 | SPI | |
| Explorer 16 | 32MX795F512L | MRF24WB0M | SPI | |
| PIC24FJ256DA210 Development Board | 24FJ256DA210 | ENC28J60 | SPI | |
| PIC24FJ256DA210 Development Board | 24FJ256DA210 | ENCX24J600 | SPI | |
| PIC24FJ256DA210 Development Board | 24FJ256DA210 | ENCX24J600 | PSP 5 Indirect Bitbang | |
| PIC24FJ256DA210 Development Board | 24FJ256DA210 | MRF24WB0M | SPI | |
| PIC32 General Purpose Starter Kit (DM320001) | 32MX360F512L | ENC28J60 | SPI | |
| PIC32 General Purpose Starter Kit (DM320001) | 32MX360F512L | ENCX24J600 | SPI | |
| PIC32 General Purpose Starter Kit (DM320001) | 32MX360F512L | ENCX24J600 | PSP 5 Indirect | |
| PIC32 General Purpose Starter Kit (DM320001) | 32MX360F512L | MRF24WB0M | SPI | |

Демонстрационный Board Processor MAC/PHY Layer Comm. Примечания Шины PIC18 Explorer 18F87J11 ENC28J60 SPI Требует кремниевое исправленное издание A4 или позже. PIC18 Explorer 18F87J11 ENCX24J600 SPI PIC18 Explorer 18F87J11 MRF24WB0M SPI PIC18 Explorer 18F87J50 ENC28J60 SPI PIC18 Explorer 18F87J50 ENCX24J600 SPI PIC18 Explorer 18F87J50 MRF24WB0M SPI PIC18 Explorer 18F8722 ENC28J60 SPI PIC18 Explorer 18F8722 ENCX24J600 SPI PIC18 Explorer 18F8722 MRF24WB0M SPI PICDEM.net 2 18F97J60

ETH97J60 - PICDEM.net 2 18F97J60 ENC28J60 SPI PICDEM.net 2 18F97J60 ENCX24J600 SPI
PICDEM.сеть 2 18F97J60 MRF24WB0M SPI Explorer 16 24FJ128GA010 ENC28J60 SPI
Explorer 16 24FJ128GA010 ENCX24J600 SPI Explorer 16 24FJ128GA010 MRF24WB0M SPI
Explorer 16 24FJ128GA010 ENCX24J600 PSP 5 Косвенный
Explorer 16 24FJ256GA110 ENC28J60 SPI2 Explorer 16 24FJ256GA110 ENCX24J600 SPI2
Explorer 16 24FJ256GA110 ENCX24J600 PSP 5 Косвенный
Explorer 16 24FJ256GA110 MRF24WB0M SPI2

7.1 Демонстрационных Подсказки Стека Совместимости Table Microchip TCP/IP Explorer 16
24FJ256GB110 ENC28J60 SPI Explorer 16 24FJ256GB110 ENCX24J600 SPI Explorer 16
24FJ256GB110 ENCX24J600 PSP Косвенный
5
Explorer 16 24FJ256GB110 MRF24WB0M SPI Explorer 16 24FJ256GB210 ENC28J60 SPI
Explorer 16 24FJ256GB210 ENCX24J600 SPI Explorer 16 24FJ256GB210 ENCX24J600 PSP
Indirect Bitbang
5
Explorer 16 24FJ256GB210 MRF24WB0M SPI Explorer 16 33FJ256GP710 ENC28J60 SPI
Explorer 16 33FJ256GP710 ENCX24J600 SPI Explorer 16 33FJ256GP710 ENCX24J600 PSP
Косвенный
5
Explorer 16 33FJ256GP710 MRF24WB0M SPI Explorer 16 32MX360F512L ENC28J60 SPI
Explorer 16 33EP512MU810 ENC28J60 SPI2 Explorer 16 24EP512GU810 ENC28J60 SPI2
Explorer 16 32MX360F512L ENCX24J600 SPI Explorer 16 32MX360F512L ENCX24J600 PSP
Косвенный
5
Плата Разработки Explorer 16 32MX360F512L ENCX24J600 PSP 9 Explorer 16
32MX360F512L MRF24WB0M SPI Explorer 16 32MX460F512L ENC28J60 SPI Explorer 16
32MX460F512L ENCX24J600 SPI Explorer 16 32MX460F512L MRF24WB0M SPI Explorer 16
32MX795F512L ENC28J60 SPI Explorer 16 32MX795F512L ENCX24J600 SPI Explorer 16
32MX795F512L MRF24WB0M SPI PIC24FJ256DA210
Плата Разработки 24FJ256DA210 ENC28J60 SPI PIC24FJ256DA210
Плата Разработки 24FJ256DA210 ENCX24J600 SPI PIC24FJ256DA210
24FJ256DA210 ENCX24J600 PSP Indirect Bitbang
5
Плата Разработки PIC24FJ256DA210
УНИВЕРСАЛЬНЫЙ Комплект Стартера 24FJ256DA210 MRF24WB0M SPI PIC32
(DM320001)
УНИВЕРСАЛЬНЫЙ Комплект Стартера 32MX360F512L ENC28J60 SPI PIC32 (DM320001)
УНИВЕРСАЛЬНЫЙ Комплект Стартера 32MX360F512L ENCX24J600 SPI PIC32
(DM320001)
32MX360F512L ENCX24J600 PSP Косвенный
5
УНИВЕРСАЛЬНЫЙ Комплект Стартера PIC32 (DM320001)
32MX360F512L MRF24WB0M SPI

7.1 Демонстрационных Подсказки Стека Стартера Совместимости Table Microchip TCP/IP
Комплекта PIC32 USB (DM320003_2) 32MX795F512L ENC28J60 SPI2 PIC32 USB

(DM320003_2) 32MX795F512L ENCX24J600 SPI2 PIC32 USB (DM320003_2) 32MX795F512L ENCX24J600 PSP 5 Косвенный
PIC32 USB (DM320003_2) 32MX795F512L ENCX24J600 PSP 9 PIC32 USB (DM320003_2) 32MX795F512L MRF24WB0M SPI2 PIC32 Ethernet Starter Kit 32MX795F512L Internal MAC, National DP83848C PHY
-
dsPIC33E USB Starter Kit 33EP512MU810 ENCX24J600 SPI2 dsPIC33E USB Starter Kit 33EP512MU810 ENCX24J600 PSP 5 dsPIC33E USB Starter Kit 33EP512MU810 ENCX24J600 PSP 5 Косвенный
dsPIC33E USB Starter Kit 33EP512MU810 MRF24WB0M SPI2 PIC24E USB Starter Kit 24EP512GU810 ENCX24J600 SPI2 PIC24E USB Starter Kit 24EP512GU810 ENCX24J600 PSP5 PIC24E USB Starter Kit 24EP512GU810 ENCX24J600 PSP 5 Косвенный
PIC24E USB Starter Kit 24EP512GU810 MRF24WB0M SPI2

TCPIP WebVend ( СМОТРИ страницу 126)

Демонстрационный Board Processor MAC/PHY Layer Comm.
Примечания Шины
PICDEM.net 2 18F97J60 ENC28J60 SPI PICDEM.net 2 18F97J60 ETH97J60 - Explorer 16 24FJ128GA010 ENC28J60 SPI Explorer 16 24FJ128GA010 ENCX24J600 SPI Explorer 16 24FJ128GA010 MRF24WB0M SPI Explorer 16 33FJ256GP710 ENC28J60 SPI Explorer 16 33FJ256GP710 ENCX24J600 SPI Explorer 16 33FJ256GP710 MRF24WB0M SPI Explorer 16 32MX360F512L ENC28J60 SPI Explorer 16 32MX360F512L ENCX24J600 SPI Explorer 16 32MX360F512L MRF24WB0M SPI Explorer 16 32MX460F512L ENC28J60 SPI Explorer 16 32MX460F512L ENCX24J600 SPI Explorer 16 32MX460F512L MRF24WB0M SPI Explorer 16 32MX795F512L ENC28J60 SPI Explorer 16 32MX795F512L ENCX24J600 SPI Explorer 16 32MX795F512L MRF24WB0M SPI

TCPIP WiFi EasyConfig ДЕМОНСТРАЦИОННЫЙ Прил. ( смотри страницу 133)

Демонстрационный Board Processor MAC/PHY Layer Comm.
Примечания Шины
Explorer 16 24FJ128GA010 MRF24WB0M SPI

7.1 Демонстрационных Подсказки Стека Совместимости Table Microchip TCP/IP Платы Explorer 16 33FJ256GP710 MRF24WB0M SPI Explorer 16 32MX360F512L MRF24WB0M SPI Explorer 16 32MX460F512L MRF24WB0M SPI Explorer 16 32MX795F512L MRF24WB0M SPI PIC24FJ256DA210
Разработка PIC24FJ256DA210 MRF24WB0M SPI

TCPIP WiFi КОНСОЛЬНЫЙ Демонстрационный Прил. ( смотри страницу 127)

Демонстрационный Board Processor MAC/PHY Layer Comm.
Примечания Шины
ПЛАТА PICDEM.net 2 18F97J60 MRF24WB0M SPI Explorer 16 24FJ128GA010 MRF24WB0M SPI Explorer 16 33FJ256GP710 MRF24WB0M SPI Explorer 16 32MX360F512L MRF24WB0M SPI Explorer 16 32MX460F512L MRF24WB0M SPI Explorer 16 32MX795F512L MRF24WB0M SPI PIC24FJ256DA210
Разработка PIC24FJ256DA210 MRF24WB0M SPI

РАДИО TCPIP Internet Прил. ( смотри страницу 126)

Демонстрационный Board Processor MAC/PHY Layer Comm. Шина Отмечает Internet Radio Board 18F67J60 ENC28J60 SPI

TCPIP Internet Bootloader ( СМОТРИ страницу 120)

Демонстрационная Плата
Процессор MAC/PHY Layer Comm.
Примечания Шины
N/A 18F66J60 ETH97J60 -
N/A 18F66J60 ETH97J60 -РАСШИРИВШЕЕ Instruction Mode N/A 18F66J65 ETH97J60 -
N/A 18F66J65 ETH97J60 -РАСШИРИВШЕЕ Instruction Mode N/A 18F67J60 ETH97J60 -
N/A 18F67J60 ETH97J60 -РАСШИРИВШЕЕ Instruction Mode N/A 18F86J60 ETH97J60 -
N/A 18F86J60 ETH97J60 -РАСШИРИВШЕЕ Instruction Mode N/A 18F86J65 ETH97J60 -
N/A 18F86J65 ETH97J60 -РАСШИРИВШЕЕ Instruction Mode N/A 18F87J60 ETH97J60 -
N/A 18F87J60 ETH97J60 -РАСШИРИВШЕЕ Instruction Mode N/A 18F96J60 ETH97J60 -
N/A 18F96J60 ETH97J60 -РАСШИРИВШЕЕ Instruction Mode N/A 18F96J65 ETH97J60 -
N/A 18F96J65 ETH97J60 -РАСШИРИВШЕЕ Instruction Mode N/A 18F97J60 ETH97J60 -

7.2 Доступных Демонстрационных Подсказок Стека Microchip TCP/IP Демонстрационный Прил. N/A 18F97J60 ETH97J60 -Расширившее Режим Инструкции

TCPIP MDD ДЕМОНСТРАЦИОННЫЙ Прил. ( смотри страницу 135)

Демонстрационный Board Processor MAC/PHY Layer Comm.
Примечания Шины
Explorer 16 PIC24FJ256GB110 ENC28J60 SPI Использует Поворотного Накопителя USB как средство памяти для страниц веб.
Explorer 16 PIC24FJ128GA010 ENC28J60 SPI Использует Карту SD как средство памяти для страниц веб.

## Карта Google

Для информации о демонстрационной совместимости Карты Google, смотри, что файл "ПОДГОТОВКИ -Прогон Графической Карты Google Демонстрационный" в Combo Demos/Google Mapdirectory в вашей установке Библиотеки Приложений директория Микрокристалла.

### 7.2 Доступных Демонстрационных

TCP/IP Стека приходит с несколькими приложениями примера. Эти приложения описаны в следующих секциях.

#### 7.2.1 Демонстрационных Прил.

Папка проекта TCPIP \ Демо приложение содержит основную демонстрационное приложение для Microchip Стек TCP / IP . Помимо примеров приложений, использующих веб-сервер , почтовый клиент , сервер SNMP , и многое другое , это приложение также включает в себя примеры для реализации слоев пользовательских приложений . Подробнее об этих приложениях описано здесь.

Для получения списка предварительно протестированных конфигураций демо аппаратных , пожалуйста, обратитесь к демонстрационная таблице (см. стр. 82) .

Произвольные аппаратные конфигурации могут быть также полезной с Демо App , но некоторые дополнительные настройки могут быть необходимо.

Некоторые демо отключены в некоторых проектов Демо App поддерживать связанную аппаратную платформу и TCP / IP контроллер . Пожалуйста, обратитесь к следующей таблице , чтобы определить, какие функции доступны на которые конфигурациях:

##### 7.2.1.1 TCPIP Demo App Особенности аппаратно платформы

Некоторые аппаратные платформы не могут поддерживать все функции, реализованные в TCP / IP Demo Application. Следующий таблице приведены , какие возможности доступны для каждой комбинации демонстрационной платы и MAC / PHY слоя встроенная поддержка TCP / IP Демо приложения . Обратите внимание, что эта таблица не появится в PDF версии файла справки ; см. " TCPIP Demo App Features.htm " файл в папке TCPIP документации в папке Библиотека Помощь Microchip Application

**ПАМЯТЬ NVM**

Плата с энергонезависимой памяти может изменять и сохранять свои переменные конфигурации во время выполнения. В TCP / IP Demo App , это позволяет изменить имя платы, IP-адрес (см. стр. 147) , беспроводной SSID, беспроводной безопасности или другую конфигурацию параметров через интерфейс веб-страницы . Данные будут записаны в SPI Flash, или EEPROM , а затем использовать для повторной инициализации доска , если она срабатывала . Доска без этой функции всегда будет использовать настройки по умолчанию после включения питания .

**Кнопки и светодиоды**

В TCP / IP стека - совместимые демо доски имеют переменное число кнопок и светодиодов. По умолчанию, TCP / IP Демо приложения настроен на отображение и принимаю ( см. стр. 169) вход от 8 светодиодов и 4 кнопки на индексной странице нашего примера ; кнопки и светодиоды , используемые в зависимости от того, что доступно на плате.

**7.2.1.2 Демо Модули**

**Модули**

**Имя Описание**

Веб-страницы Демос ( см. стр. 87 ) Содержит пример для построения пользовательского приложения HTTP с помощью http2 сервер и позволяет несколько других демо- функции, чтобы получить доступ и контроль через веб-интерфейс .

Электронная почта (SMTP) Демо ( см. стр. 96 ) Показывает, как использовать клиент электронной почты для отправки сообщений , когда события происходят . Это автономный демо ; для веб-страницы " Отправить письмо" демо см. Формы, используя POST (см. стр. 90) тему .

Общий Клиент TCP ( см. стр. 97 ) Показывает, как построить клиентское приложение TCP через клиента HTTP пример.

Общий Сервер TCP ( см. стр. 100 ) Демонстрирует построения серверного приложения TCP Пинг ( ICMP) Демо ( см. стр. 101 ) Демонстрирует , как построить клиента Ping.

Сервер SNMP (Агент) ( см. стр. 103 ) Описывает простой протокол управления сетью Demo.

**описание**

Несколько пользовательские модули используются в этой демонстрации . В этом разделе будут описаны компоненты и функциональные возможности этих модули .

**7.2.1.2.1 веб-страницы Демос**

Функции

Имя Описание

**HTTPPostSNMPCommunity**( См. стр. 92 ) Это функция HTTPPostSNMPCommunity .

**HTTPPostConfig** (см.стр. 93) Обработка формы конфигурации на конфигурации / index.htm

**HTTPPostDDNSConfig** (см. стр. 93) Разбор и сбор данных HTTP , полученные от формы HTTP .

**HTTPPostEmail** (см. стр.94) Процессы электронной почты форму на электронной почте / index.htm

**HTTPPostLCD** (см. стр.94) Процессы ЖК форму на forms.htm

**HTTPPostMD5** (см. стр.95) Обработка формы загрузки файла на upload.htm

Переменные

Описание Имени
**DDNSData** ( см. стр. 95 ) Оперативная память выделяется для параметров DDNS

**lastFailure** (см. стр. 96) переменную сообщение о состоянии Придерживайтесь. Смотреть lastSuccess (см. стр. 96) для деталей.

**lastSuccess** (см. стр. 96) переменную сообщение Важно статус. Это используется для указанная или нет Предыдущая операция POST прошла успешно. Приложение использует их для хранения Сообщения о состоянии, когда операция POST переадресацию. Это позволяет приложение обеспечить сообщения о состоянии после редиректа, когда данные экземпляр соединение имеет уже потеряли.

**описание**

Файл **CustomHTTPApp**.c демонстрирует, как построить собственное приложение HTTP поверх сервера http2. Все особенности веб-страниц TCPIP Demo App реализованы здесь. Примеры можно найти для обработки проверки подлинности ( см. стр. 89), обработки веб-форм (с помощью HTTP **GET** и **POST**), а также предоставление информации о состоянии через выход динамические переменные.

#### 7.2.1.2.1.1 динамические переменные

модуль

Веб-страницы Демос ( см. стр. 87 )

описание

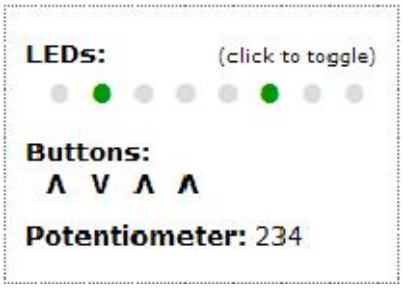
**обзор**

В этом разделе описано, как просмотреть динамические переменные в TCP / IP Демо App http2 демо. Для получения информации о том, как реализовать динамические переменные в вашей собственной странице , увидеть http2 динамических переменных тему (см. стр. 231 ) .

**инструкции**

1 . Программирование доску с демо- кода и загрузить веб-страницу демо приложение . Откройте веб-браузер и перейдите к веб-страницы в разделе ( **http://mchpboard** по умолчанию).

2 . Соблюдайте LED выход, состояние кнопки и потенциометра чтение в окне в правом верхнем углу веб-страницы. кнопка и значения потенциометров будет обновляться динамически на основе статуса кнопки на борту. Кроме того, если вам нажмите светодиоды для переключения их , их статус будет динамически обновляется на странице . Обратите внимание, что некоторые светодиоды или кнопки могут не быть реализован , в зависимости от настроек аппаратного обеспечения . Обратитесь к Особенности TCPIP Demo App на аппаратная платформа ( См. стр. 86 ) тему для получения дополнительной информации .




3 . Соблюдайте текущий стек версии и дата сборки в верхней центральной части Обзор Page .

4 . Перейдите на страницу динамических переменных , используя панель навигации на левой части страницы .

5 . Соблюдайте Телосложение даты и времени , LED состояние , версию стека , и Текущий IP-адрес - эти переменные выводятся на это страницы динамически , когда он загружен браузером.

**упражнения**

При необходимости можно выполнить упражнения , описанные на странице динамических переменных . Вы можете читать http2 динамические переменные тему (см. стр. 231) в первую очередь. Первое упражнение заключается в реализации отображение LED0 на динамический

Переменная демонстрационная страница .

1 . Начните с открытия dynvars.htm в папке " TCPIP Demo App \ WebPages2 " .

2 . Найдите динамических переменных на странице и замените вопросительный знак с динамической переменной для отображения значения LED 0 . Вы можете использовать другие светодиодные переменные в качестве шаблона , но указать 0 в качестве светодиода , чтобы открыть .

3 . В проекте MPLAB , открытый CustomHTTPApp.c и убедиться, что функция **HTTPPrint_led** (если вы использовали ~ привело (0) ~ в качестве динамической переменной), если написано в выходных данных при 0 передается в качестве параметра .

4 . Перестроить свою веб-страницу с MPFS2 Utility.

5 . Восстановленный ваш проект , и перепрограммировать ваш совет . Перейдите к динамической переменной странице и убедитесь , что поле LED0 отражает состояние светодиода на вашем правлении . Поскольку светодиод на плате мигает, возможно, потребуется обновить веб страницы , чтобы просмотреть свой нынешний статус .

Второе упражнение на этой странице , просто показывает, как динамически вставить файл в веб-страницу .

1 . Начните с открытия dynvars.htm в папке " TCPIP Demo App \ WebPages2 " .

2 . Найдите динамических переменных , которые включают header.inc и footer.inc . Соблюдайте разницу между объявление этих переменных и других переменных на странице.

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

### 7.2.1.2.1.2 Аутентификация

**модуль**

Веб-страницы Демос ( см. стр. 87 )

**описание**

обзор

В этом разделе описывается, как использовать демо аутентификации в TCP / IP Демо App http2 демо. Для получения информации о как реализовать аутентификацию в вашей собственной странице , см. в разделе http2 аутентификации (см. стр. 236 ) .

**инструкции**

1 . Программирование доску с демо- кода и загрузить веб-страницу демо приложение . Откройте веб-браузер и перейдите к веб-страницы в разделе ( http://mchpboard по умолчанию).

2 . Перейдите на страницу аутентификации с помощью навигационной панели слева от страницы.

3 . Обратите внимание на имя пользователя аутентификации ( " администратора " ) и пароль ( " микрочип " ) .

4 . Нажмите ссылку " ограниченного доступа Page " на странице аутентификации .

5 . Введите неправильную комбинацию имени пользователя и пароля. Браузер не будет продвигаться к Access Restricted Страница . После 3 неправильных комбинаций имя пользователя / пароль , браузер выдаст сообщение «несанкционированный ».

6 . Нажмите кнопку назад в вашем браузере. Нажмите на ссылку " ограниченного доступа Страница " и введите правильное имя пользователя и пароля .

7 . Вы будете продвигаться к « успешном входе " странице. Вашему браузеру будет хранить эту комбинацию имени пользователя / пароля , пока он не закрытия и повторного открытия .

**упражнение**

При необходимости можно выполнить упражнение , описанное на " Вход успешного " странице. В этом упражнении вы измените имя пользователя и пароль , которые вы используете , чтобы войти в эту страницу .

1. в TCP / IP Демо App MPLAB проекта. Открыть CustomHTTPApp.c .

2 . Найдите HTTPCheckAuth ( см. стр. 242 ) функция .

3 . Измените значения сравниваемых с функциональными вклада в имя пользователя и пароль из вас выбирая .

4 . Перестроить свой проект и запрограммировать ваш совет .

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

### 7.2.1.2.1.3 Формы используя Поиск

**модуль**

Веб-страницы Демос ( см. стр. 87 )

описание

**обзор**

В этом разделе описывается, как использовать веб-формы в TCP / IP Демо App http2 демо. Для получения информации о том, как реализовать форму в вашей собственной странице, см. в разделе http2 обработка формы (см. стр. 233) .

**инструкции**

1 . Программирование платы с демо- кодом и загрузите веб-страницу демо приложения. Откройте веб-браузер и перейдите к веб-страницы в разделе ( http://mchpboard по умолчанию).

2 . Соблюдайте LED состояние на доске. Нажмите на светодиодном индикаторе в поле в правом верхнем углу на странице Обзор. проверить что изменения Индикация состояния на борту. Обратите внимание, что некоторые светодиоды или кнопки не могут быть реализованы, в зависимости от вашей установки оборудования. Обратитесь к особенностям TCPIP Demo App на аппаратной платформе (см.стр.86) тема для большей информации.

3 . Перейдите на страницу Форма обработки с помощью навигационной панели слева от страницы.

4 . Выберите новые светодиодные состояния в выпадающих окнах. Нажмите кнопку " Сохранить" и заметим, что светодиодные состояния вашего правления изменена на соответствовать выбранные настройки.

**Упражнение**

При необходимости можно выполнить упражнение , описанное на странице " Форма обработки " . В этом упражнении вы измените Пример для поддержки LED5 . Вы можете прочитать эту тему http2 обработки формы (см. стр. 233) в первую очередь.

1 . Начните с открытия forms.htm в папке " TCPIP Demo App \ WebPages2 " .

2 . Найдите реализацию метода GET покажет светодиоды . Вы должны увидеть выберите формы для четырех светодиодов, которые уже реализованы . Каждая из них имеет два варианта: вариант Вкл. пошлет '1 ' на сервер при представлены и Выкл. вариант пошлет '0 ' , когда представил . Каждый из заявлений этих вариантов также использовать ledSelected динамической переменной , чтобы определить , какой вариант будет выбран по умолчанию , в зависимости от текущего состояния соответствующего Светодиода на плате . Это динамическая переменная принимает два аргумента: первый определяет, какие в настоящее время проверяются светодиод , и Вторая описывает состояние проверяется на . Так , например,~ ledSelected (4 , TRUE) ~ переменной будет заменен словом "избранных" , если LED4 горит, когда эта переменная функция обратного вызова вызывается . В этом случае ~ ledSelected (4 , ЛОЖЬ) ~ будет заменен ничего. Это приведет к опция «On» выбирают по умолчанию на странице .

3 . Создайте новый выберите вход для LED5 .

4 . Открыть CustomHTTPApp.c в проекте протокола TCP / IP Демо App MPLAB .

5 . Убедитесь, что функция **HTTPPrint_ledSelected** динамической переменной обратного вызова был реализован в течение LED5 .

6 . Найдите **HTTPExecuteGet** (см. 242 стр. ) функцию . Найдите раздел кода процессы GET аргументы для файла forms.htm .

7 . Добавить реализацию для поиска в " LED5 " строки аргумента в буфер данных GET , а затем установить LED5_IO на основе соответствующее значение .

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

### 7.2.1.2.1.4 Формы, использовавших POST

**модуль**

Веб-страницы Демос ( см. стр. 87 )

**описание**

**обзор**

В этом разделе описывается, как использовать веб-формы в TCP / IP Демо App http2 демо. Для получения информации о том, как реализации форм в вашей собственной странице , см. в разделе http2 обработки формы (см. стр. 233) .

**инструкции**

1 . Программирование доску с демо- кода и загрузить веб-страницу демо приложение . Откройте веб-браузер и перейдите к веб-страницы в разделе ( http://mchpboard по умолчанию).

2 . Перейдите на страницу Форма обработки с помощью навигационной панели слева от страницы.

3 . Введите текстовую строку в "LCD" текстового поля и нажмите " Сохранить". Убедитесь, что эта строка была отображена на ЖК- дисплее вашей демонстрационной плате.

4 . Перейдите на страницу загрузки файлов с помощью навигационной панели слева от страницы.

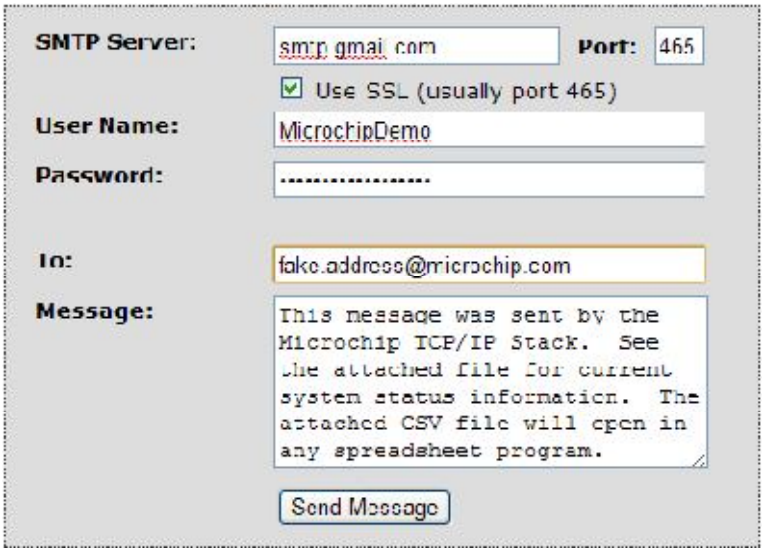
5 . Браузер для файла на вашем компьютере и нажмите кнопку " Получить MD5 . " Приложение будет читать файл с помощью серии POST переводы и рассчитать и дисплей и MD5 хеш содержания .

6 . Перейдите к отправить по электронной почте страницу с помощью навигационной панели слева от страницы.

7 . Заполните поля формы с соответствующей информацией.

1) Нет SSL - . Вам понадобится локальный SMTP -сервер , который не требует безопасного соединения. Введите адрес в SMTP Поле Сервер , установите порт на 25 , и введите свое имя пользователя и пароль для сервера. Установите " Кому: " поле в электронной почте получатель и нажмите " Отправить сообщение ".

2) SSL - . Введите адрес общественной SMTP сервер (например, smtp.gmail.com ) . Установите номер порта на 465 или 587 . Введите Информация учетной записи электронной почты (например, username@gmail.com и ваш пароль Gmail ) . Установите " Кому: " поле в электронной почте получатель и нажмите " Отправить сообщение ". Обратите внимание, что некоторые корпоративные подсети могут блокировать исходящий трафик безопасного на SMTP порт . Если это так, то вы должны будете установить VPN туннель вне этой сети или подключения (см. стр. 171 ) ваш совет к сети, не заблокирован этого типа брандмауэра. Необходимо установить микрочип данные Шифрование Библиотеки использовать SSL , а также поддержка клиента SSL должен быть включен . См. SSL API ( см. стр. 462 ) тему для больше информации .




8 . Убедитесь, что электронная почта была получена на получателя электронной почты .

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

### 7.2.1.2.1.5 Cookies

**Module**

Веб-страницы Демос ( см. стр. 87 )

**описание**

**обзор**

В этом разделе описывается, как использовать печенья демо в TCP / IP Демо App http2 демо. Для получения информации о том, как реализовать печенье в вашей собственной странице , см. в разделе http2 куки (см. стр. 238 ) .

**инструкции**

1 . Программирование доску с демо- кода и загрузить веб-страницу демо приложение . Откройте веб-браузер и перейдите к веб-страницы в разделе ( http://mchpboard по умолчанию).

2 . Перейдите на страницу Cookies используя панель навигации на левой части страницы .

3 . Укажите свое имя в текстовое поле " Имя " и щелкните "Установка Cookies. " Убедитесь, что имя было успешно читать и отображаются в поле "Имя" выходного области .

**Упражнение**

При необходимости можно выполнить упражнение , описанное на странице " Cookies " . В этом упражнении вы создадите печенье под названием " Любимая " со значением в любимой области в поле примера . Вы можете читать динамическую переменную http2 (см. стр.231) , GET (см. стр. 233) , и печенье (см. стр. 238 ) темы в первую очередь.

1 . Начните с открытия cookies.htm в папке " TCPIP Demo App \ WebPages2 " .

2 . Найдите код для примера поле, которое отображает имя и любимый архитектуры ПОС . Заменить (см. стр. 221 ) " не реализована" строку с динамической переменной для вывода данных из печенья .

3 . Найдите код для выбора окно формы ввода для любимой архитектуры. Обратите внимание на значение имени области выберите форма .

4 . Открыть CustomHTTPApp.c в проекте протокола TCP / IP Демо App MPLAB . Найдите HTTPExecuteGet (см. стр. 242 ) Функции и найти код, который обрабатывает GET метод входы от cookies.htm.

5 . Установить значение curHTTP.hasArgs чтобы указать, что два аргумента формы присутствуют в буфере данных .

6 . В CustomHTTPApp.c , создать функцию для вывода данных для динамической переменной , созданный на шаге 2. Название функция будет зависеть от имени переменной . Для переменная с именем ~ cookiefavorite ~ вы будет осуществлять Функция называется HTTPPrint_cookiefavorite . Эта функция должна искать через буфер данных curHTTP.data к попытаться найти пару имя / значение с именем , равным имени вашего выбора формы из пункта 3. Если он находит его, он должен написать значение для этой пары в буфер TCP ; в противном случае , он должен написать " не установлен. " Рассматриваем осуществление

HTTPPrint_cookiename для примера.

```
void HTTPPrint_cookiename(void)
{
BYTE *ptr;
ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE*)"name");
if(ptr)
TCPPutString(sktHTTP, ptr);
else
TCPPutROMString(sktHTTP, (ROM BYTE*)"not set");
return;
}
```

7 . Составьте свою веб-страницу с помощью MPFS2 Utility и загрузить его на свой борт. Вы можете получить предупреждение, что ваш динамические переменные на странице изменились.

8 . Перестроить свой проект и запрограммировать ваш совет .

9 . Убедитесь, что оба печенье может быть множество.

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

#### 7.2.1.2.1.6 Functions

| | |
|---|---|
| HTTPPostSNMPCommunity ( смотри страницу 92) | Это - функция HTTPPostSNMPCommunity. |
| HTTPPostConfig ( see page 93) | Обрабатывает форму конфигурации на config/index.htm |
| HTTPPostDDNSConfig ( see page 93) | Разбор и сбор данных HTTP , полученные от формы HTTP . |
| HTTPPostEmail ( see page 94) | Обрабатывает форму эл.почты на email/index.htm |
| HTTPPostLCD ( see page 94) | Обрабатывает форму LCD на forms.htm |
| HTTPPostMD5 ( see page 95) | Обработка формы загрузки файла на upload.htm |

**Module**

Web Page Demos ( see page 87)

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

#### 7.2.1.2.1.6.1 HTTPPostSNMPCommunity Function

**File**
CustomHTTPApp.c

**C**
**static** HTTP_IO_RESULT **HTTPPostSNMPCommunity**();

**Description**
Это - функция HTTPPostSNMPCommunity.

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

#### 7.2.1.2.1.6.2 HTTPPostConfig Function

**File**
CustomHTTPApp.c

**C**
**static** HTTP_IO_RESULT **HTTPPostConfig**();

**Description**
Принимает параметры конфигурации из формы, сохраняет их во временную папку в памяти, то в конечном итоге сохраняет данные в EEPROM или внешней Flash.
После завершения эта функция перенаправляет на **Config / reboot.htm**, который будет отображать информацию о повторного подключения к плате.
Эта функция создает теневую копию структуры **AppConfig** в оперативной памяти, а затем переписывает входящие данные там, как это прибывает. Для каждой пары имя / значение, имя впервые прочитал в **curHTTP**.**data** [0:5]. Далее, значение считывается в **newAppConfig**. После того как все данные считаны, новый **AppConfig** сохраняется обратно в EEPROM и браузер перенаправляется на **reboot**.**htm**.
Этот файл включает в себя вызов **AJAX** к **reboot**.**cgi**, которая выполняет фактическое перезагрузки машины.
Если IP-адрес не может быть разобрано, слишком много данных, размещенных или любой другой разбора ошибка, на перезагрузки браузера **config**.**htm** и выводит сообщение об ошибке на самом верху.

**Preconditions**
None

**Return Values**

HTTP_IO_DONE - все параметры были обработаны данные

HTTP_IO_NEED_DATA, - данные, необходимые этой функции еще не наступило

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

### 7.2.1.2.1.6.3 HTTPPostDDNSConfig Function

**File**
CustomHTTPApp.c

**C**
**static** HTTP_IO_RESULT **HTTPPostDDNSConfig**();

**Description**
Эта процедура будет , выполненного каждый раз, когда форма Настройка клиента динамического DNS представляется . Данные HTTP получен как строка из переменных , разделенными "& " символов в буфере TCP RX. Эти данные анализируются , чтобы прочитать требуемая значения конфигурации , и те значения заполняются в глобальном массиве ( DDNSData ( см. стр. 95 ) ) , отведенные для этого Цель . Как читать данные , DDNSPointers также заполняется так , что динамический DNS-клиент может выполнить с новыми параметрами.

**Preconditions**
curHTTP ( see page 240) is loaded.

**Return Values**

HTTP_IO_DONE - Завершенное процедурой
HTTP_IO_NEED_DATA - больше данных , необходимые для продолжения , перезвоните позже

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

### 7.2.1.2.1.6.4 HTTPPostEmail Function

**File**
CustomHTTPApp.c

**C**
**static** HTTP_IO_RESULT **HTTPPostEmail**();

**Description**

Эта функция посылает по электронной почте сообщение об помощью клиента SMTP и, возможно, шифрует соединение с SMTP- сервером с помощью SSL. Это демонстрирует использование клиентом SMTP , ожидая асинхронных процессов в HTTP обратного вызова , и , как отправить вложения электронной почты с использованием стека .

Сообщения с вложениями направляются использовании многотомных / смешанный кодировку MIME , который имеет три раздела . Не первый не имеет заголовки , и только для отображения старых клиентов, которые не могут интерпретировать формат MIME. (Подавляющее большинство эти клиенты были obseleted, но так называемый раздел " игнорируется " до сих пор используется.) Второй имеет несколько заголовков в указать, что он является основным органом сообщения в кодировке текстовом . Третий раздел имеет заголовки , указывающие прикрепленные файл , вместе с его именем и типом . Все разделы разделены границей строки , которые не могут появиться в другом месте в сообщение .

**Preconditions**
None

**Return Values**

HTTP_IO_DONE - сообщение было отправлено
HTTP_IO_WAITING - функцию ждет процесс SMTP для завершения

HTTP_IO_NEED_DATA - данные, необходимые этой функции еще не наступило

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

#### 7.2.1.2.1.6.5 HTTPPostLCD Function

**File**
CustomHTTPApp.c

**C**
```
static HTTP_IO_RESULT HTTPPostLCD();
```

**описание**

Находит параметр ' ЖК ' и использует его для обновления отображаемой на ЖК-дисплее платы текст.

Эта функция имеет четыре состояния . Первый считывает имя из строки данных, возвращаемых как часть запроса POST. Если имя не может быть найден , то возвращается , просить больше данных . В противном случае, если имя , как ожидается , он считывает соответствующее значение и записывает его в ЖК-дисплее. Если имя не ожидается , значение отбрасывается и следующий параметр имя читается .

В случае, когда ожидаемое строка никогда не встречается , эта функция не будет возвращать в конечном счете HTTP_IO_NEED_DATA при отсутствии данных осталось . В этом случае сервер http2 автоматически ловушка ошибку и выдать внутренняя ошибка сервера к браузеру.

**Preconditions**
None

**Return Values**

HTTP_IO_DONE - параметр обнаружен и сохранен
HTTP_IO_WAITING - функция останавливается, чтобы продолжать позже данным
HTTP_IO_NEED_DATA - данные, необходимые этой функции еще не наступило

**Section**
Прототипы функций и Глобализируемые памяти

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

### 7.2.1.2.1.6.6 HTTPPostMD5 Function

**File**
CustomHTTPApp.c

**C**
```
static HTTP_IO_RESULT HTTPPostMD5();
```

**Description**
Эта функция демонстрирует обработку загрузки файлов. Во-первых, функция находит данные файла , пропуская никаких заголовков что приехать. Во-вторых, он читает файл 64 байт в то время, и хэши эти данные. После того как все данные были получены , функция вычисляет сумму MD5 и сохраняет его в **curHTTP.data** .

После заголовков , первая линия от формы будет MIME сепаратор . После это больше заголовков о файле , которые отбросить . После очередного CRLFCRLF , начинается данные файла , и мы читаем его 16 байт в то время, и добавить, что в MD5 расчет . Чтение заканчивается, когда строка разделитель снова столкнулись на отдельной строке . Обратите внимание, что сам файл данные разгромили в этом процессе , что позволяет нам принимать (см. стр. 169 ) файлы произвольного размера , не ограниченные памяти. Также обратите внимание, что буфер данных используется в качестве произвольного массива для хранения результата. ~ **Uploadedmd5** ~ обратного вызова читает эти данные позже отправить обратно клиенту .

**Preconditions**

None

**Return Values**

HTTP_IO_DONE все параметры были обработаны

HTTP_IO_WAITING функцию во время паузы продолжить позже

HTTP_IO_NEED_DATA данные, необходимые этой функции еще не наступило

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

#### 7.2.1.2.1.7 Variables

**Module**
Web Page Demos ( see page 87)

**Variables**

**DDNSData** ( см. стр. 95 ) Оперативная память выделяется для параметров DDNS

**lastFailure** (см. стр. 96) переменную сообщение о состоянии Придерживайтесь . Смотреть lastSuccess (см. стр. 96) для деталей.

**lastSuccess** (см. стр. 96) переменную сообщение Важно статус . Это используется для указанная или нет Предыдущая операция POST прошла успешно. Приложение использует их для хранения Сообщения о состоянии , когда операция POST переадресацию. Это позволяет приложение обеспечить сообщения о состоянии после редиректа , когда данные экземпляр соединение имеет уже потеряли .

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

#### 7.2.1.2.1.7.1 DDNSData Variable

**File**
CustomHTTPApp.c

**C**
BYTE **DDNSData**[100];

**Description**
PAM распределенное для параметров DDNS

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

#### 7.2.1.2.1.7.2 lastFailure Variable

**File**
CustomHTTPApp.c

**C**
BOOL **lastFailure** = FALSE;

**Description**

Придерживайтесь переменную сообщения о статусе. Смотреть **lastSuccess** (см. стр. 96) для деталей

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

#### 7.2.1.2.1.7.3 lastSuccess Variable

**File**
CustomHTTPApp.c

**C**
BOOL **lastSuccess** = FALSE;

**Description**
Переменная сообщение Важно статус . Это используется для указано , была ли предыдущая операция POST успешно.

Приложение использует их для хранения сообщений о состоянии , когда операция POST перенаправляет . Это позволяет приложению предоставить статус сообщения после редиректа , когда данные экземпляра соединение уже потерял .

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

### 7.2.1.2.2 E-mail (SMTP) Demo
**Functions**

**Description**
**Overview**

Этот файл содержит два примера для использования клиентского модуля SMTP для отправки сообщений электронной почты. Первый передает краткое предупреждение Сообщения которых весь органы могут быть сохранены в памяти одновременно. Второй пример демонстрирует , как  генерировать Сообщения на лету , когда все тело не может быть выделено в оперативной памяти . ( Второй пример комментирует . Вы должны комментировать первый пример и уберите этот его использовать. )

Третий пример помощью клиента SMTP предоставляется в **HTTPPostEmail** (см. стр. 94 ) . Этот пример показывает, как отправить сообщения с вложениями , а также , как динамически настраивать получателя и сервера электронной почты во время выполнения .

Инструкции ( коротких сообщений Демо )

1 . Откройте свой проект в MPLAB и открытой **SMTPDemo**.c . Перейдите в случае MAIL_BEGIN в распределительном заявление в **SMTPDemo** (см. стр. 97) () функцию .

1 . Заменить (см. стр. 221 ) инициализатор RAMStringTo [ ] массив с целевой адрес электронной почты.

2 . Заменить (см. стр. 221 ) инициализатор структуры **SMTPClient**.**Server**.**szROM** элемента с адресом вашего почтовый сервер . Обратите внимание, что это демо не включает функции безопасности , так что вам потребуется почтовый сервер , который не требует SSL . Чтобы проверить эту функциональность с почтового сервера , что делает поддержку SSL ( в том числе большинства общедоступных серверов почтовых ) , пожалуйста, используйте **HTTPPostEmail** ( см. стр. 94) SMTP демо.

2 . Скомпилируйте код , запрограммировать ваш совет , и запустить демо.

3 . Помощью кнопок 2 и 3 на вашем борту для передачи сообщения электронной почты . СИД1 на доске будет указано, что сообщение является передается ; Индикатор 2 будет указано, что успешно отправлено. Проверьте BUTTON2_IO , BUTTON3_IO , LED1_IO ,и LED2_IO макросы в копии HardwareProfile.h , что соответствует вашему проекту , чтобы определить, какие кнопки и Светодиоды используются для настроек аппаратного обеспечения .

4 . Убедитесь, что сообщение было получено от учетной записи электронной почты , указанный в **RAMStringTo** [ ] массива.

**описание**

**SMTPDemo** коротких сообщений ( см. стр. 97 ) функция задача реализует государственную машину четыре состояния . Когда плата будет питание, государственная машина инициализируется государства SM_HOME , в котором он ждет кнопок 2 и 3 , чтобы быть нажата . один раз они прижаты , задача будет войти в состояние MAIL_BEGIN .

В состоянии MAIL_BEGIN , задача будет пытаться реквизировать SMTP модуль . Как только это в состоянии сделать это , он будет заполнить **SmtpClient** ( см. стр. 304) структуру с параметрами сообщений и передать сообщение . Он будет затем введите MAIL_SMTP_FINISHING состояние .

В состоянии MAIL_SMTP_FINISHING , задача будет проверить функцию обратного вызова (**SMTPIsBusy** (см. стр. 305 ) ) , чтобы определить, когда модуль закончена. Затем он отказаться от контроля над SMTP модуля и переключения светодиодов на основе успешной эксплуатация SMTP модуля. Государственная машина будет войти в состояние MAIL_DONE, который будет ждать не менее 1 секунды перед переходом к MAIL_HOME, позволяя другому электронной почты для отправки .

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

#### 7.2.1.2.2.1 SMTPDemo Function

**File**
MainDemo.h

**C**
```
void SMTPDemo();
```

**Module**
E-mail (SMTP) Demo ( see page 96)

**Returns**
None

**Description**
Эта функция демонстрирует использование клиентом SMTP . Функция вызывается периодически стеком , и проверяет, нажаты ли одновременно Button2 и Button3. Если они нажаты, он пытается отправить сообщение по электронной почты с помощью параметров жестко в функции ниже.

В то время как клиент выполняет, LED1 будет использоваться в качестве насыщенного индикатора. LED2 будет гореть , когда передача была успешно завершена. Если оба светодиода не горят, произошла ошибка.

В качестве примера отправки более длинное сообщение (тот, который не существует в памяти все сразу ), см.прокомментировал вторичный Реализация этой функции в этом файле ( **SMTPDemo**.c ) ниже. В качестве примера отправки сообщения с использованием параметров собрались во время выполнения, и / или сообщение с вложениями, увидеть реализацию **HTTPPostEmail** (см. стр. 94) в **CustomHTTPApp**.c .

**Preconditions**
The SMTP client is initialized.

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

#### 7.2.1.2.3 Generic TCP Client

**Functions**

**GenericTCPClient** - ( страница 98)
Реализует простой HTTP клиента ( по TCP ) .

**RemoteURL** - ( см. стр. 99 ) Определяет URL , чтобы быть запрошена этой HTTP клиента

**ServerName** ( см. стр. 99 ) Определяет сервер , чтобы получить доступ к приложению

**ServerPort** ( см. стр. 99 ) Заметим, что если HTTPS используется ,ServerName ( см. стр. 99 ) и URL должны изменить к SSL включен сервер.

**Description**
**Overview**

Клиент Generic TCP приводится пример , как построить HTTP клиента ( или любого другого клиента TCP) с помощью Microchip Стек TCP / IP . Это выведет результаты из запроса поисковик, чтобы модуля UART в ПОС . Данные результат может быть просмотре на ПК терминал .

**инструкции**

1 . Подключите запрограммированный демо доска к маршрутизатору , который подключен к Интернету.

2 . Подключите компьютер к демонстрационной платы с помощью кабеля RS-232. Откройте терминал программы, как HyperTerminal , и настроить его следующими параметрами: 19200 , 8 бит данных, без проверки четности, 1 стоповый бит, без управления потоком .

3 . Нажмите кнопку 1 на демонстрационной платы ( проверить BUTTON1_IO макрос в копии HardwareProfile.h , что соответствует Ваш проект , чтобы определить, какая кнопка Кнопка1).

4 . Соблюдайте Результаты поиска для " Microchip " на www.microchip.com на своем терминале .

**описание**

Демо Generic TCP Client реализует функцию задач с пяти государств. Когда плата включено, начальное состояние будет быть установлен в SM_DONE . Это состояние будет ждать пока пользователь не нажмет кнопку 1 ; при возникновении кнопку - пресс-конференция , государство будет переключиться на SM_HOME . В состоянии SM_HOME , задача будет пытаться открыть клиентский TCP сокет ( см. стр. 152 ) . это сокет будет использовать тип TCP_PURPOSE_GENERIC_TCP_CLIENT сокета ( см. стр. 152 ) из структуры сокета TCP ( см. стр. 153) , что был инициализирован в файлах конфигурации. Целевая сервер будет поисковая система Google, а также порт сервера будет 80, порт, используемый для HTTP соединений. Задача переключится государственную машину, чтобы SM_SOCKET_OBTAINED состояние.

Задача будет ждать в состоянии SM_SOCKET_OBTAINED пока соединение не будет установлено с помощью Google или 5 - секундной задержкой истекает. Если происходит таймаут, государство будет закрыть сокет и изменения состояния обратно в SM_HOME. В противном случае, он будет ждать пока буфер TCP не может принять (см. стр. 169 ) 125 байт данных, а затем использовать HTTP GET (см. стр. 233 ) для поиска для слова " Microchip " на месте " microchip.com . " Как только GET был отправлен , государство будет переключиться на SM_PROCESS_RESPONSE .

В состоянии SM_PROCESS_RESPONSE , задача будет ждать, покане будет получен ответ или сокет был отключен. Если ответ получен , он будет печатать его в UART. В любом

случае , задача будет перейти в состояние SM_DISCONNECT , где он будет закрыть клиентский сокет и вернуться в состояние SM_DONE.

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

#### 7.2.1.2.3.1 GenericTCPClient Function

**File**
MainDemo.h

**C**
```
void GenericTCPClient();
```

**Module**
Generic TCP Client ( see page 97)

**Returns**
None
**Description**

Эта функция реализует простой HTTP -клиент , который работает над TCP. Функция называется периодически стека , и ждет Button1 , чтобы быть нажата . При нажатии на кнопку , приложение открывает соединение TCP к Интернету Поисковая система , выполняет поиск слова " Microchip " на " microchip.com " , и печатает полученный HTML страницу в UART .

Этот пример может быть использован в качестве модели для многих приложений TCP и клиентских HTTP .

**Preconditions**
TCP is initialized.

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

#### 7.2.1.2.3.2 Variables

**Module**
Generic TCP Client ( see page 97)

**Variables**

**RemoteURL** ( см. стр. 99 ) Определяет URL , чтобы быть запрошена этой HTTP клиента

**ServerName** ( см. стр. 99 ) Определяет сервер , чтобы получить доступ к приложению

**ServerPort** ( см. стр. 99 ) Заметим, что если HTTPS используется ,ServerName ( см. стр. 99 ) и URL должны изменить к SSL включен сервер.

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

#### 7.2.1.2.3.2.1 RemoteURL Variable

**File**
GenericTCPClient.c

**C**
ROM BYTE **RemoteURL**[] = "/search?as_q=Microchip&as_sitesearch=microchip.com";

**Description**
Определяет URL, чтобы быть запрошен этим клиентом HTTP

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

#### 7.2.1.2.3.2.2 ServerName Variable

**File**
GenericTCPClient.c

**C**
BYTE **ServerName**[] = "www.google.com";

**Description**
Определяет сервер , чтобы получить доступ к приложению

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

#### 7.2.1.2.3.2.3 ServerPort Variable

**File**
GenericTCPClient.c

**C**
WORD **ServerPort** = HTTP_PORT;

**Description**
Заметим, что если HTTPS используется, ServerName (см. стр. 99) и URL должны изменить к SSL включен сервер.

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

#### 7.2.1.2.4 Generic TCP Server

**Functions**
**GenericTCPServer** - ( see page 101) - Реализует простой ToUpper TCP Server.

**Macros**

SERVER_PORT - Определяет, какой порт сервер будет слушать (см. стр. 175)
**Description**

**Overview**
Пример Generic TCP Server демонстрируется построения серверного приложения TCP . После подключения (см. стр. 171 ) в демо-сервере , он будет эхо нажатия клавиш обратно к вам после преобразования символов в верхнем регистре.

**инструкции**

1 . Подключите запрограммированный демо доска к компьютеру напрямую или через маршрутизатор . Для Ethernet, прямое подключение может потребовать перекрестного кабеля; для WiFi, доска возможно, должны быть в режиме AdHoc чтобы установить прямое соединение.

2 . Определить IP адрес демо-платы . Это может быть сделано несколькими различными способами.

1 . Если вы используете демо плату с ЖК-дисплеем (например, Explorer, 16 или PICDEM.net 2), IP- адрес должен быть отображается на второй строке дисплея .

2 . Откройте Microchip TCP / IP Discoverer из стартового меню . Нажмите кнопку «Открой Устройства ", чтобы увидеть адреса и имена хостов всех устройств cОбъявить ( см. стр. 155 ) Протокол включен в вашей сети. Возможно, вам придется настроить брандмауэр компьютера, чтобы предотвратить его блокирование UDP порт 30303 для этого решения .

3. Если ваш совет подключается непосредственно к компьютеру с помощью кросс-кабеля. :

1 . Откройте командную / DOS и наберите " IPCONFIG ". Найти сетевой адаптер, подключенный к плате. IP-адрес совета находится в поле ' Default Gateway "

2 . Откройте состояние сети для сетевого адаптера, который соединяет два устройства. Это может быть сделано по праву нажав на значок сетевого соединения в папке сетевых настроек и выберите ' статус ' из меню . Найти " Основной шлюз " поле.

3 . Откройте командную / DOS . Тип " Telnet ip_address 9760 ", где ip_address является IP- адрес, который вы получили от шаг 2 и 9760 здесь порту выбрали для реализации Generic TCP Server.

4 . По мере ввода символов, они будут передаваться обратно в окне командной строки в верхнем регистре.

5 . Нажмите Побег в конец демо.

**описание**

**GenericTCPServer** (см. стр. 101) демо реализующий функцию задач с 3-х государств . В первом состоянии , SM_HOME , Задача будет пытаться открыть сокет сервера TCP ( см. стр. 152 ) . Этот разъем будет использовать Тип TCP_PURPOSE_GENERIC_TCP_SERVER разъем ( см. стр. 152 ) из структуры сокета TCP ( см. стр. 153 ) , который был инициализирован в файлах конфигурации . Он также будет слушать (см. стр. 175 ) на TCP порт 9760 (определяется с помощью макроса SERVER_PORT ( см. стр. 101 )) .

После того, как сокет был успешно открыт , функция Задание перейдет в состояние SM_LISTENING. В этом состоянии, задача всегда будет возвращать, если клиент не подключен к нему ( путем установления соединения телнет на порт 9760 ). Как только клиент имеет подключен к серверу, сервер будет читать полученные данные из приемного буфера TCP сокета, преобразовать его в верхний регистр, и написать ему TX буфера TCP сокета.

При получении Побег характер , сервер перейдет в состояние SM_CLOSING . В этом состоянии , он закроет сокет сервера чтобы прервать текущее соединение . Сервер будет повторно войти в состояние SM_HOME , где он будет вновь открыть TCP_PURPOSE_GENERIC_TCP_SERVER сокет слушать (см. стр. 175) для новых подключений .

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

#### 7.2.1.2.4.1 GenericTCPServer Function

**File**
MainDemo.h

**C**
```
void GenericTCPServer();
```

**Module**
Generic TCP Server ( see page 100)

**Returns**
None

**Description**
Эта функция реализует простой сервер TCP . Функция вызывается периодически стека слушать (см.стр.175 ) для входящих соединений . Когда соединение установлено, сервер читает все входящие данные , преобразует их в верхний регистр , и отправляет обратно .

Этот пример может быть использован в качестве модели для многих серверных TCP приложений.

**Preconditions**

TCP is initialized.

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

#### 7.2.1.2.4.2 Macros

**Macros**
SERVER_PORT ( see page 101) - смотри Определяет какой порт сервер будет слушать ( смотри страницу 175)

**Module**
Generic TCP Server ( see page 100)

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

#### 7.2.1.2.4.2.1 SERVER_PORT Macro

**File**
GenericTCPServer.c

**C**
```
#define SERVER_PORT 9760
```

**Description**
Определяет , какой порт сервер будет слушать (см. стр. 175) на

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

#### 7.2.1.2.5 Ping (ICMP) Demo

**Functions**

PingDemo ( см. стр. 102 ) Демонстрирует использование клиентом ICMP ( пинг ) .

**Macros**

HOST_TO_PING - смотри Адрес ( смотри страницу 147), что клиент ICMP будет **Ping**. Если модуль клиента DNS не доступен в стеке, тогда этот **hostname** проигнорировано и локальный межсетевой адрес IP будет pinged взамен.

**Description**
**Overview**
Демонстрация **Ping** объясняет как, использовать клиента ICMP, чтобы проверяться если дистанционный узел - reachable. Если проект с этим демонстрационным включает модуль DNS, PIC будет звоном "ww1.microchip.com." В противном случае, это будет звоном локальный шлюз. Этот демонстрационный только доступно в аппаратных установках с дисплеями LCD (напр.. Explorer 16 или PICDEM.net 2).

**Instructions**
1. Нажмите Кнопку 0 на вашу демонстрационную плату. Кнопка 0 - обычно самая правая или самая верхняя кнопка на плате (проверьте макро BUTTON0_IO в копии HardwareProfile.h, которая переписывается в ваш проект, чтобы определять, что точно какая кнопка - Кнопка 0).
2. Когда устройство получает ответ эха из дистанционного узла или когда время звона, LCD будет скорректирован подходящей информацией.

**Description**

**PingDemo** ( см. стр. 102 ) Функция задача реализует государственную машину двух государств . Задача будет ждать в SM_HOME состояние , пока пользователь не нажмет кнопку 0 . После того, как кнопка нажата , то задача будет пытаться получить право собственности на ICMP Модуль с ICMPBeginUsage (см. 264 стр. ) функцию . Если это произойдет, это будет отправить пинг по указанному адресу и перейти в состояние SM_GET_ICMP_RESPONSE .

В состоянии SM_GET_ICMP_RESPONSE , задача будет вызывать ICMPGetReply (см. стр. 266 ) функция обратного вызова и принять действия в зависимости от возвращаемого значения :

Значение Действие

-2 Остаются в этом состоянии и продолжают ждать ответ.

-1 Написать сообщение для ЖК указывая, что пинг истекло. Изменение состояния в SM_HOME .

-3 Написать сообщение для ЖК указывая, что модуль DNS не мог решить целевой адрес . Изменение государство SM_HOME .

Другое Преобразование время отклика в виде текстовой строки и печатать его в ЖК-дисплее. Изменение состояния в SM_HOME .

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

#### 7.2.1.2.5.1 PingDemo Function

**File**
MainDemo.h

**C**
```
void PingDemo();
```

**Module**
Ping (ICMP) Demo ( see page 101)

**Returns**
None

**Description**

Эта функция реализует простой клиент ICMP . Функция называется периодически стека , и он проверяет , если **BUTTON0** была нажата . При нажатии на кнопку , функция посылает ICMP Echo Request ( Ping ) на веб-сервер Microchip . отображается время передачи на UART , когда принимается ответ .

Эта функция может быть использована в качестве модели для приложений, требующих возможности Ping для проверки хоста.

**Preconditions**
TCP is initialized.

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

#### 7.2.1.2.5.2 Macros

**Macros**

HOST_TO_PING - Адрес (см. стр. 147 ) , что клиент ICMP будет пинг . Если клиентский модуль DNS является нет в наличии в стеке , то это имя хоста игнорируется и локального шлюза IP -адрес будет пингуется вместо этого.

**Module**
Ping (ICMP) Demo ( see page 101)

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

#### 7.2.1.2.5.2.1 HOST_TO_PING Macro

**File**
PingDemo.c

**C**
**#define HOST_TO_PING** "ww1.microchip.com" / / адрес, который клиент ICMP будет пинг . Если Клиентский модуль DNS не доступен в стеке , то это имя хоста игнорируется, и Локальный IP-адрес шлюза будет пинговаться вместо этого.

**Description**
Адрес ( смотри страницу 147), что клиент ICMP будет пинг . Если клиентский модуль DNS не доступен в стеке , то это Имя хоста игнорируется и местный IP -адрес шлюза будет пингуется вместо этого.

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

### 7.2.1.2.6 SNMP Server (Agent)

**Functions**

Имя Описание

**SendNotification** (см. стр.116)

Подготовка , проверка удаленного узла , который будет получать ловушку и отправить перехваты PDU.

**SNMPGetTimeStamp** (см.стр. 116)

Получает текущее значение Tick для отметки времени SNMP.

**макрос**

**Имя Описание**

**MAX_TRY_TO_SEND_TRAP (**см. стр. 118 )

**SNMP_MAX_NON_REC_ID_OID**( См. стр. 118 )

Обновите Non значение OID Индентификационный который является частью CustomSnmpDemo.c

**файл**

**STACK_USE_SMIV2** (см.стр. 118)По умолчанию STACK_USE_SMIV2 включена . Для Stack V5.31 ,

**STACK_USE_SMIV2** должны быть отключены.

Переменные

Имя Описание

**gSendTrapSMstate** (см.стр. 117)Это переменная gSendTrapSMstate .

**gSnmpNonMibRecInfo** (см. стр. 117)СТАРЫЙ файл snmp.mib стандарту SMIv1

**gSnmpv3UserSecurityName**(См. стр. 117 )Это переменная gSnmpv3UserSecurityName .

**описание**

Сервер Microchip SNMP является многоязычным реализация , которая поддерживает функции сервера SNMPv1, v2c и v3 одновременно. Сервер SNMP реализуется для удовлетворения потребностей встраиваемых приложений . Поддержка SNMPv3 добавляют TCPIP Stack Version 5.31. SNMPv1 и V2c включены с односпальными макро

**#define STACK_USE_SNMP_SERVER** . Сервер SNMPv3 может быть выборочно включен с независимыми макро

**#define STACK_USE_SNMPV3_SERVER** . Как SNMPv3 стеки должны поддерживать SNMPv1 и SNMPv2c , что позволяет SNMPv3 Сервер автоматически включает серверы SNMPv1 и SNMPv2c в агенте Microchip TCPIP стека SNMP. Эти макросы определены в TCPIP ( MAC / PHY ) ч Файл расположен на <installation path> : . \ Microchip Solutions ГЛАВНАЯ \ TCPIP \ Демо App \ конфигурации \ .

Эта серия вопросов будут рассмотрены приложения - и демо - конкретной реализации сервера SNMP входит в Демо приложение TCP / IP . Для информации, описывающей модуль SNMP в целом , можно найти в разделе SNMP API .

V2c реализуется с поддержкой конфигурации нескольких имен сообщества , которые хранятся в отдельных энергонезависимой памяти ( SPI EEPROM или SPI Flash) . Имена сообщества можно настроить с помощью мастера TCP / IP Configuration или через веб-интерфейс HTTP/MPFS2 . Веб-страницы доступа ограничивается снабжен демонстрационного приложения , чтобы позволить динамического конфигурирования SNMP общин .

SNMPv3 RFC определяет различные типы механизма доступа , модель безопасности пользователя (USM) , аутентификации и конфиденциальности протоколов. Сервер Microchip SNMPv3 реализуется с поддержкой USM, AES 128 CFB 128 протокол конфиденциальности , и MD5 и SHA1 сообщение протоколы аутентификации . Демо Реализация сервера настроен с 3-х различных типов имена пользователей с соответствующими аутентификации и конфиденциальности учетных данных и типов аутентификации . Эти учетные данные и другие информация о пользователе хранятся в глобальном массиве . Пользователь стека SNMPv3 может решать вопрос о числе имен пользователей в база данных пользователя, чтобы хранить с сервером . В соответствии с рекомендацией SNMPv3 , сервер SNMPv3 не должны быть сконфигурирован с аутентификации и конфиденциальности паролей. Вместо может быть настроен с соответствующими локализованных ключей пароля . Microchip SNMPv3 агент получает информацию паролем в базе для " Получение Started " и для понимания только цель. Рекомендуется, чтобы стек SNMPv3 должен быть изменен , чтобы ограничить доступ к паролю OIDs объявленных в базе данных пользователя.

Обратите внимание, что несмотря на то, SNMPv3 также требует SNMPv1 и SNMPv2c , слой в стеке SNMP предотвратит доступ к переменные , которые должны быть обеспеченные SNMPv3 . SNMP переменные структуры в дереве в информационной базе управления (MIB) . Доступ к части этого дерева определяются версии . Например, системные переменные типа могут быть доступны независимо от SNMP версии , SNMPv2c запросы могут получить доступ к части дерева , и идентифицированные запросы SNMPv3 может доступ к полной дерево.

Вам потребуется два пакетов микропрограмм для запуска сервера демо SNMPv3 . Каждый из них должен быть установлен в указанном порядке , как некоторые файлы будут перезаписаны. Библиотека шифрования программное обеспечение не требуется, если SNMPv3 услуги не предназначены .

1 . Микрочипа Прикладные библиотеки , в том числе TCP / IP стека v5.31 или более поздней версии . Этот код доступен в www.microchip.com / MLA .

2 . Шифрование данных Библиотеки Microchip. Это демо использует уровень безопасности SSL для связи с Google. Для использования SSL со стеком TCP / IP , вам потребуется эти библиотеки . Они доступны в компакт- основе или загружаемом формате с MicrochipDirect за небольшую плату ( требуется соблюдать США скрининга криптографической с ограничением экспорта ) . Вы должны

выполнить команду " Microchip TCPIP стека Vx.xx Шифрование дополнений on.exe " установки установить файлы, необходимые демо.

ARCFOUR.c / ч и RSA.c / ч криптографические файлы в этом установщик будет заменить фиктивные версии , найденные с умолчанию Установка Стек TCP / IP .

Чтобы включить функциональность SNMPv3 , необходимо добавить библиотеку PIC32_AES.a от шифрования данных библиотек на свой демо Проект .

Примечание: Для существующих Microchip SNMP v1 и v2c пользователей .

• пользователи SNMP v1/v2c желающие обновить Microchip TCP / IP стек с более старых версий до последней версии и продолжить использовать протокол SNMP v1/v2c может получить услуги SNMP v1/v2c от этого агента , при условии, что они не изменить настройки по умолчанию модуль SNMP в v5.25 вперед.

•Форма реализации для V1 и V2c остается тем же , за исключением нескольких новых возможностей и функций , за исключением . имена и параметры некоторых функций были изменены. Пользователи v1/v2c , возможно, придется вносить изменения в их конкретного приложения код . Там не должно быть никаких изменений в коде SNMP стека , если пользователи не включены Код приложения в стеке SNMP.

• Пользователи должны построить новый MPFS изображение, используя MPFS File System Generator утилиту и загрузить его на выбранный EEPROM или флэш-памяти, как структура AppConfig обновляется , чтобы разместить имена сообществ в V2c и SNMP двигатель ID для SNMPv3 .

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

**7.2.1.2.6.1 Файлы MIB**

СЕРВЕР SNMP (Агент) ( смотри страницу 103)

**Описание**

SNMP описывает иерархическую хранение объектов управления (называемый с идентификаторами объектов или OID) с управлением Информационная база управления (MIB ) файлы . Сервер демо Microchip SNMP состоит из двух файлов MIB :

• mchip.mib - Это Абстрактная синтаксическая нотация версии один (ACH.1 ) отформатирован MIB файл, содержащий информацию о переменных используется в демо .

• snmp.mib - Это обычай формате файл, который может быть разобран для создания веб-страницы и заголовка ресурсы , которые могут быть Доступ с ПОС микроконтроллера.

IP- стек TCP / включает mib2bib утилиту , которая будет компилировать обычай Microchip MIB сценария ( snmp.mib ) для создания два файла называемые snmp.bib и mib.h. Файл snmp.bib представляет собой сжатый запись объектов управления, которые будут храниться с веб-страниц и файлов mib.h содержит C определяет для каждого OID. Эти файлы включены в соответствующие каталоги для TCP / IP Демо приложения , но для пользовательского приложения необходимо скопировать snmp.bib в каталог веб-страницы , скопируйте mib.h к вашему каталог приложения и включить его в проект , восстановить ваш проект, а затем восстановить и повторно загрузить веб-страницы . это будет расслоение файл BIB в ваш веб-страницы изображения, что позволит агент SNMP для поиска требуемой переменной Информация с MPFS файловой системы.

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

**7.2.1.2.6.2 MIB Браузеры**

СЕРВЕР SNMP (Агент) ( смотри страницу 103)

Описание

Несколько браузеров SNMP MIB доступны . Пользователи также могут установить индивидуальные MIB -браузер , характерные для их применения.

В этом файле справки описывает с помощью браузера iReasoning MIB запустить демонстрационное приложение . IReasoning MIB -браузер может быть получить по адресу: http://www.ireasoning.com/downloadmibbrowserlicense.shtml . Сценарий загрузки MIB , структура дерева MIB дисплей, и процедуры механизма SNMP запросов варьируется от браузера к браузеру .

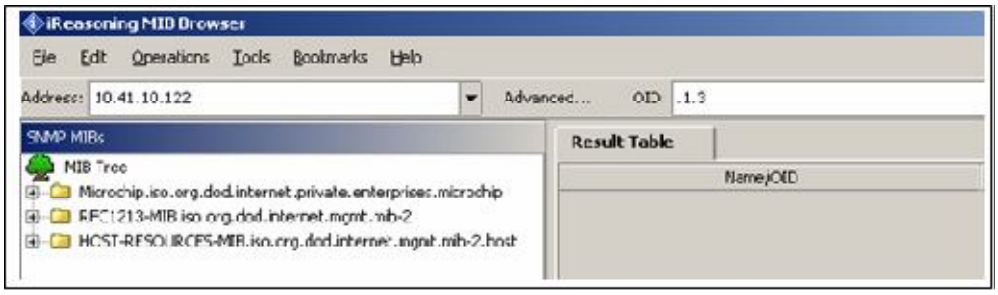
Обратите внимание, что использование браузера MIB или других инструментов сторонних может требовать, чтобы пользователи ознакомиться и согласиться с условиями лицензия . Ссылка компании Microchip в браузер iReasoning MIB для удобства пользователей . Это ответственность пользователя , чтобы получить информацию о , и соблюдать условиями , любыми применимыми лицензий.

После установки вашего браузера была завершена, выполните следующие действия :

1. Скопируйте файл mchip.mib в каталог MIB файла Вашем браузере (например, " C : \ Program Files \ ireasoning \ MibBrowser \ MIBS " ) .

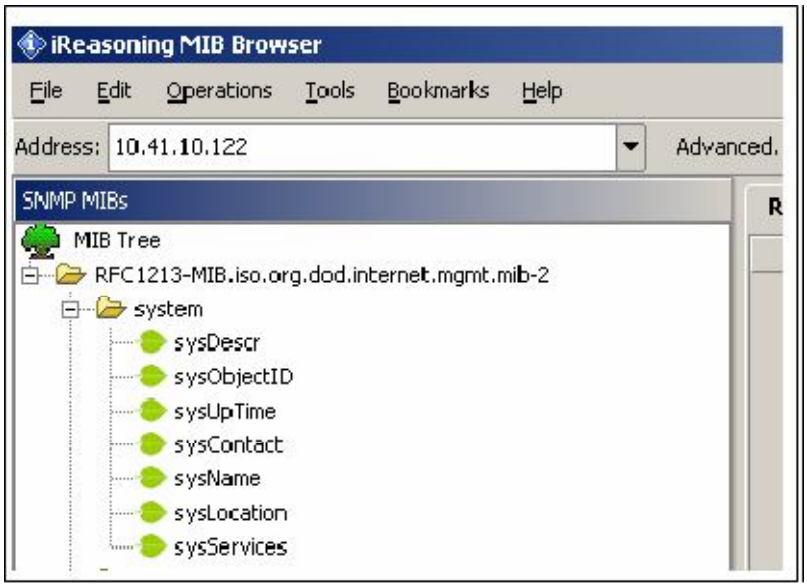
2 . Откройте iReasoning браузера выберите Файл - > Загрузить базы MIB и выберите mchip.mib , RFC1213.mib и SNMP - БАЗА - MIB.mib ( Если сервер SNMPv3 включена ) файл.

Каталог Microchip MIB будет отображаться в панели SNMP MIB .



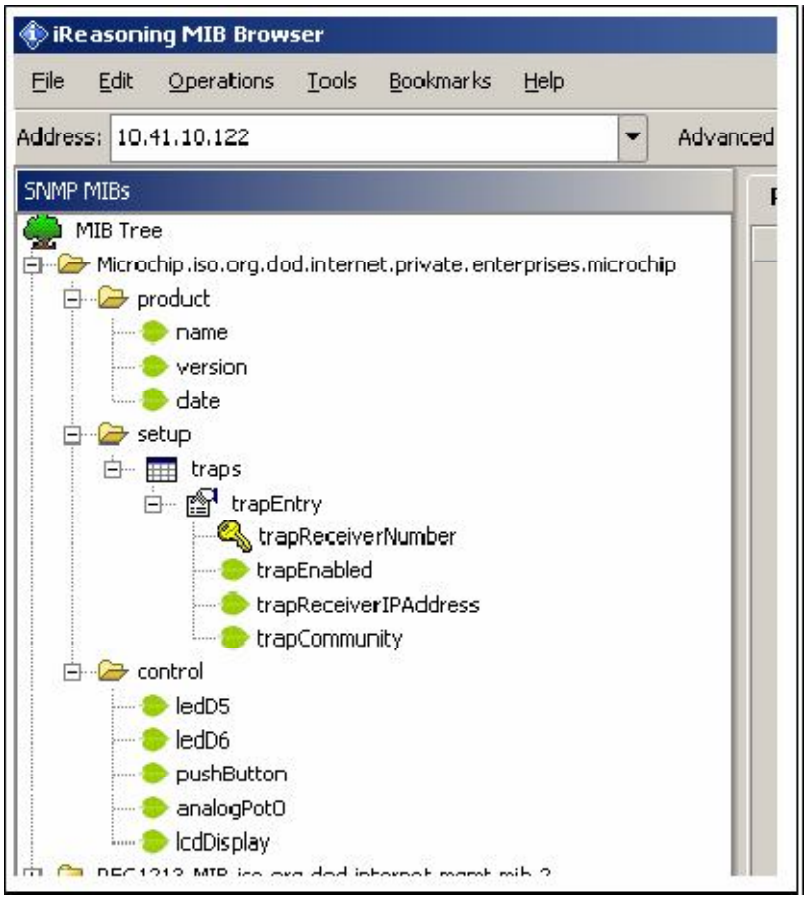

Минимальный набор RFC 1213 MIB2 переменных , которые необходимы для идентификации Microchip узел в качестве SNMP узла сети реализованы . Эти переменные могут быть доступны любым браузером SNMP с «общественной» имени сообщества типа .

См. AN870 - " SNMP v2c Agent для Microchip Стек TCP / IP " для более подробной информации о сценарии MIB , имена сообществ и демо SNMP MIB переменная структура дерева . На следующем рисунке показаны переменные , реализованные в Microchip SNMP Агент.




Формат файла mchip.mib ASN.1 определяется с собственной переменной структурой дерева для переменных MIB. Также mchp.mib является добавил с числом OIDs которые могут быть доступны только с SNMPv3 запросу. Браузер может получить доступ каждую переменную в база данных MIB условии матчи имя сообщества. Доступ к переменным MIB ограничивается типа просьба. В RFC1213 MIB переменные могут быть доступны с SNMPv2c/v3 запросу. Но SNMP - БАЗА - MIB.mib переменные могут быть доступны только с SNMPv3 reqeust , если учетные данные сопоставляются и сообщение подлинности . к изменить эти MIB переменные , соответствующие изменения должны быть внесены в обоих сценариях MIB ( snmp.mib и mchip.mib ).

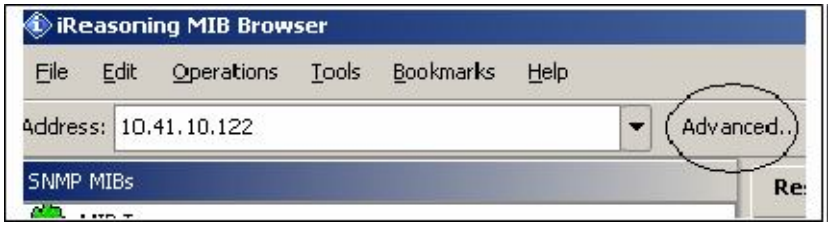
Следующий рисунок показывает личное MIB переменной древовидную структуру Microchip в браузере.



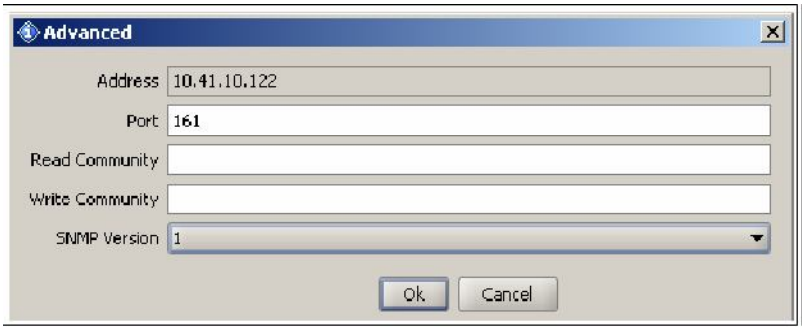

**Настройка браузера**

Чтобы настроить браузер iReasoning MIB :

1 . Выберите вкладку "Дополнительно" в браузере.




Следующее окно конфигурации будет отображено:




2 . Если требуется V2C услуги , выберите версию SNMP V2c , настроить на чтение и запись сообщество браузере.

•V2c агент будет реагировать только на запросы SNMP MIB браузеров, использующих один и тот же сообщество. То есть, V2c агент и браузер должны быть членами одного сообщества .

• Если поля общины остаются пустыми , менеджер отправляет запрос SNMP с именем сообщества , как " общественности".

•V2c агент по умолчанию настроена с 3 Читать общин ( "общественность" , "чтение" , "") и 3 Написать общин ( "частный" , " запись " , "общественность" ) .

• Максимальная длина сообщества по умолчанию составляет 8 символов .

• Как общины по умолчанию также содержат "общественное " имя сообщества , агент будет реагировать на все браузеры с просьбой «общественные» сообщество.

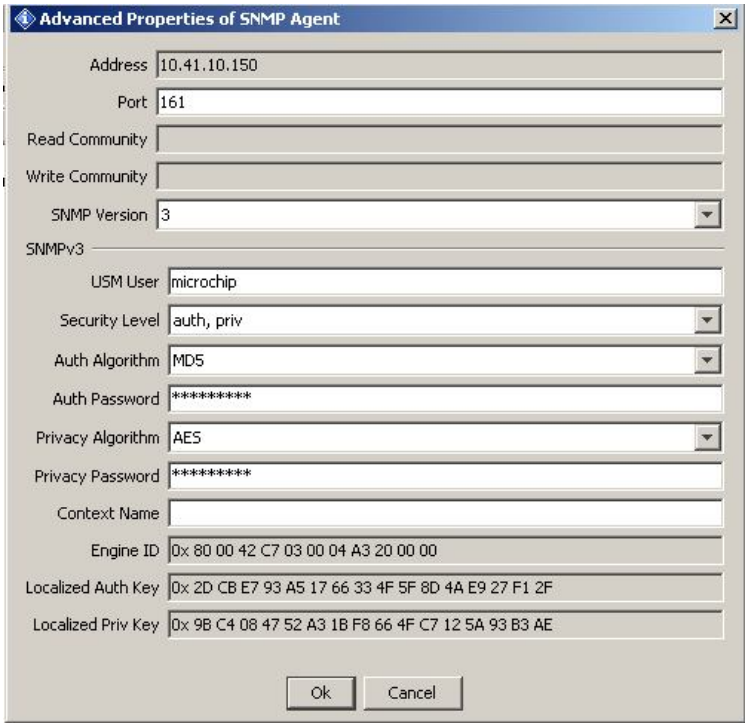
• МастерКонфигурация TCP / IP ( см. стр. 63) можно использовать для настройки имен групп SNMP по умолчанию. в во время выполнения , имена сообщества могут быть динамически настроить с помощью HTTP -интерфейс для SNMP имени сообщества

**Конфигурация** .

Если агент V2c получает запрос SNMP с неизвестным именем сообщества , агент будет генерировать подлинности ( см. стр. 89) ловушки .

Особенностью V2c агента поддержка нескольких сообщество позволяет приложению пользователя , чтобы обеспечить ограниченный доступ к запрашивающая браузер, основанный на имени сообщества обозреватель использует для доступа к переменным базы данных MIB агента.

3 . Если требуются услуги SNMPv3 , выберите SNMP версии " V3 " в "Дополнительно" вкладке браузера SNMP MIB .



Окно следующая конфигурация будет отображаться :

4 . Если требуются услуги SNMPv3 , SNMPv3 требуется браузер должен быть настроен с именем пользователя , аутентификации и пароль конфиденциальности , тип аутентификации сообщения хэш , тип протокола конфиденциальности . Сервер SNMP будет реагировать только тогда, когда один из учетных данных и параметров безопасности пользователей в таблице ниже настроен на менеджера . В приведенной ниже таблице является хранятся в глобальной структуре с стека сервера SNMPv3 . Сервер SNMPv3 бы отвечать только если запрос Полномочия браузера MIB соответствует тому, что хранимой базе данных пользовательского сервера SNMP.

USER 1 USER 2 USER 3

USM Пользователь микрочип SnmpAdmin корень

Уровень безопасности аутентификации , собств авторизации , нет собств нет авторизации , нет

собств

Авт Алгоритм MD5 SHA1

Авт пароль auth12345 ChandlerUS

Защита Алгоритм AES

Защита пароля priv12345

5 . Стек Microchip SNMPv3 ли поддерживать только одну Context двигателя ID с сервером. Оставьте опцию " Контекстное имя " на вкладке "Дополнительно" пуст . Он игнорируется на сервере.

6 . Согласно пользователем и AUTH и конфиденциальности протоколов настроенных с браузером SNMP ,UDP проверку подлинности и зашифрованное сообщение будет осуществляться обмен между сервером и клиентом .

• Если пользователь 1 значения , как в таблице выше , настроены в браузере MIB , обмен данными между клиентом и сервером шифровании и проверке подлинности . PDU может быть захвачен в анализатор пакетов Ethernet как WireShark и исследованы.

Какданные зашифрованы и подлинности ,целостности данных и конфиденциальности достигается .

• Если USER 2 значения , как в таблице выше , настроены в браузере MIB , обмен данными между клиентом и сервером подлинности . Целостность данных будет проверяться один разданные получены на обоих концах. Аутентификации сообщения механизм защищает от возможных данных нюхают и модификации угрозы, и АЛОС гарантирует, что будет полученаданные от авторизованного и гарантированного источника.

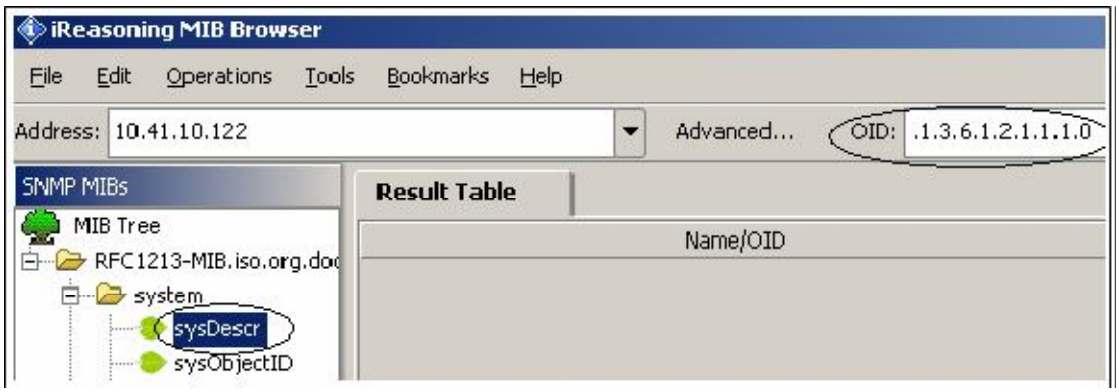
• если значения USER3 , как в таблице выше , настроены в браузере MIB , обмен данными между клиентом и сервером ни подлинности , ни в зашифрованном виде.

• Принимая во внимание выше трех пользовательских конфигураций , если сервер SNMP должен быть доступен через глобальную сеть , в интернет- облаке ,

Данные тха следует шифрования и аутентификации , имеют самый высокий уровень конфиденциальности и целостности данных.

7 . Настройка IP- адрес агента SNMP на " Адрес (см. стр. 147 ) поля .

7 . Выберите переменную, которая доступна из формы агента базы данных MIB из панели SNMP MIBS . OID выбранной переменной можно увидеть на вкладке OID на следующем рисунке .




8 . Выберите Get операцию SNMP на вкладке операций .

9 .SNMPv3 сервер демо MIB входит в RFC1213 SNMPv2 MIB переменных , частных переменных MIB и SNMP - БАЗА - MIB переменные . Если запрос SNMPv2C с подтверждены имени сообщества генерируется из MIB Браузер , только набор нескольких переменных доступен . Доступ к переменным MIB ограничивается типа SNMP версии получен запрос . Если запрос SNMPv3 с правильными учетными данными генерируется из браузера MIB , полное MIB доступ предоставляется .

10 . Пользователь потребует , чтобы решить, на котором должны быть обязаны часть MIB должен быть ограничен в зависимости от SNMP версии типа . Дизайн MIB является одним из важным шагом в определении структуры дерева MIB и переменную быть размещены соответствующим образом.

11 .SNMP сервер демо MIB добавляется с статической переменной OID назван " snmpv3PvtObject » с OID ценности как 43.6.1.4.1.17095.6.1 . Эта переменная находится в частном ветви MIB , создавая независимую ветвь. Все другие переменные в частном филиала доступны на SNMPv2c запросу. Доступ к этой статической переменной ограничен по SNMP версии типа . Только просьба SNMPv3 с правильными учетными данными может доступ к этой переменной .

**Изучение Demo**

После того как сценарий MIB загружается в браузере SNMP , структура дерева MIB будет отображаться в браузере. Любой из переменные в этом дереве можно получить (с помощью операции SNMP ) от агента , если агент поддерживает эти переменные .

браузер и агент должны быть членами одного сообщества . Чтобы узнать больше о деятельности SNMP, типов PDU, и терминология, обратитесь к AN870 - " SNMP v2c Agent для Microchip Стек TCP/IP. "

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

### 7.2.1.2.6.3 ОПЕРАЦИИ SNMP

**Module**
SNMP Server (Agent) ( see page 103)

**Описание**

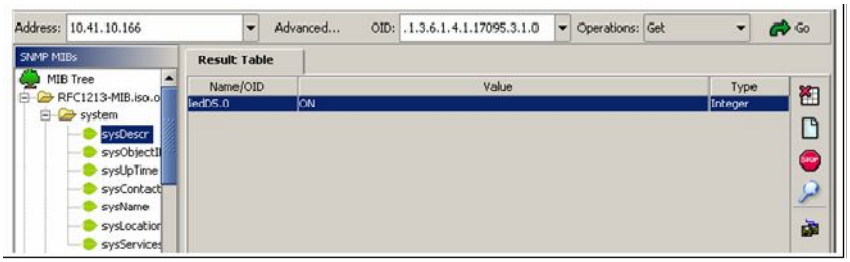
**GET**

1 . Выберите вкладку "Дополнительно" и настроить версию SNMP в '1 'и сообщества прочитанными в " общественной " .

2 . Выберите " Получить " из меню операций .

3 . Выберите переменную **sysDescr** от MIB дерева .

Таблица результатов отображает переменную информацию **sysDescr** . Повторите эту процедуру для любой переменной MIB . Для SNMP V2c ,повторите ту же процедуру , заменив '2 ' вместо '1' в конфигурации версии.




Как указывалось выше, агент V2c настроен с тремя читать и писать сообщества по умолчанию. Настройте браузер,использовать любой из этих общин и попробуйте получить доступ переменные MIB. Вы должны иметь возможность получить доступ к некоторым из переменных MIB даже ссообщения прочитанными сконфигурирован как любой из «писать» сообщества по умолчанию. Для GET операций , если чтение или запись сообщество матчей , агент обрабатывает запрос . Для множественных операций , полученные имена сообществ должно соответствовать любое из имен «писать» сообщества.

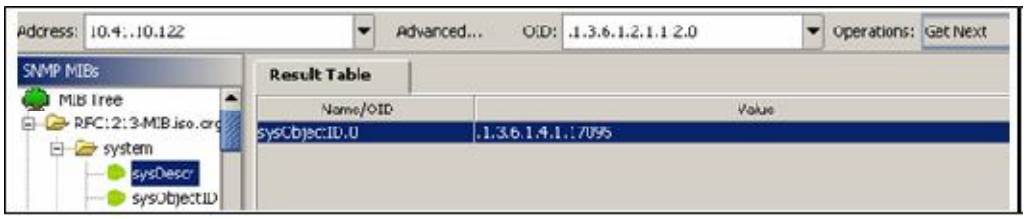
Для SNMP V3, заменяющего '3 ' вместо '1' в конфигурации версии на вкладке "Дополнительно" . Настройка другого пользователя на основе авт и собств полномочия , как описано в разделе « MIB Браузеры " .

При наличии соответствующих полномочий , все переменные MIB доступны . Выберите любой из переменных MIB в MIB дерева и сделать GET операции.

**get_next**

1 . Повторите процесс для GET . Выберите переменную sysDescr от MIB дерева. Выберите " Получить Next" от операций Меню . Таблица результатов будет отображать sysObjectID переменной информации .

2 . Повторите эти действия для дополнительных переменных MIB , чтобы получить информацию для соответствующего следующей переменной .




3 . Установите браузер версии SNMP MIB в v1/v2c . Попробуйте получить доступ к закрытой переменной MIB " snmpv3PvtObject " с OID значение как 43.6.1.4.1.17095.6.1 . Доступ должен быть ограничен. Установите Verison до V3 , configre на credentails , снова попробовать Операция get_next для переменной SAE . Доступ должен быть предоставлен .

**Get_Bulk**

Эта операция поддерживается в SNMP V2c и SNMP V3. Get_Bulk позволяет коллекцию объемной информации от агента с одного запроса от менеджера.

1 . Настройте версию SNMP для '2 ' или '3' в браузере SNMP.

2 . Если версия настроена на '2 ' , установите все сообщения прочитанными чтобы "общественность" или "читать" .

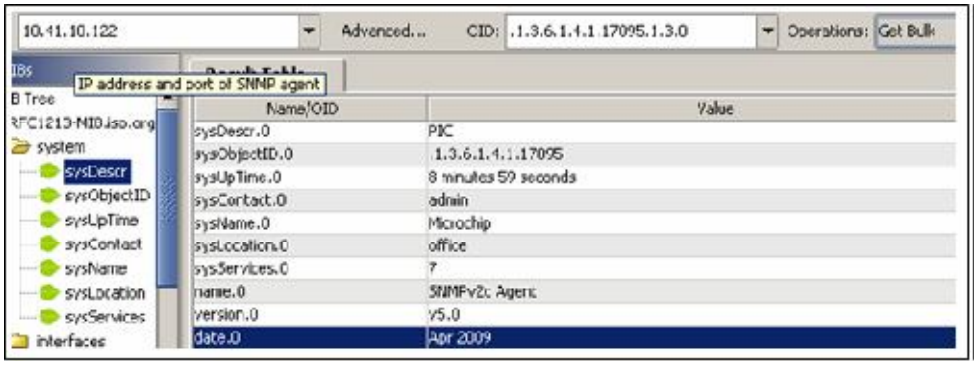
3 . Если версия настроена на '3 ' , настроить соответствующие учетные данные V3 .

4 . Выберите переменную sysDescr от MIB дерева.

5 . Выберите Get операцию массового из меню операций .

Таблица результатов будет отображать информацию в течение 10 MIB переменных в одном запросе ( если Max- Повторения = 10 и Номера для Ретрансляторы = 0 настроен ) . Эти переменные являются лексикографические преемники переменной sysDescr . число переменных, которые агент будет реагировать с может быть сконфигурирован в браузере через меню :

" Сервис-> Параметры- > Non- ретрансляторы " и " Сервис-> Параметры -> Max- Повторения ". Не- ретрансляторы и Max- Повторения Номера извлекаются агентом протокола SNMP из принятого запроса и Get_Bulk числа переменных , которые будут включен в PDU ответа вычисляется . Более подробную информацию о расчете количества переменных , не ретрансляторов , и Max- Повторения , обратитесь к RFC 3416 .




**Установка**

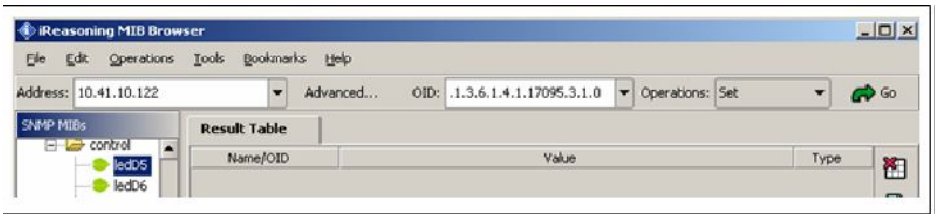
Команда установка обновляет переменную информацию базы данных MIB в агента. Команда Набор может быть выполнена только на тех переменных , которые объявлены как « ReadWrite 'в скриптах MIB , и только если имя сообщества матчи любой из «писать» имен сообщества настроенных с агентом .

1 . Выберите переменную ledD5 от MIB дерева.

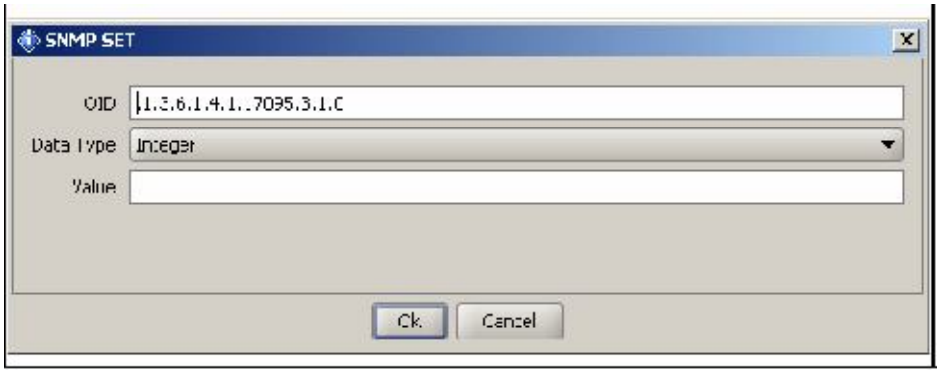
2 . Настройте версию SNMP в '1 'или '2 ". Настройте записи сообществу «общественного », « записи », или «частный» .

3 . Если версия настроена на '3 ' , настроить соответствующие учетные данные V3 .

4 . Выберите 'Установить ' в меню Operations .




Окно SNMP SET появится. Введите значение для браузера в OID области .




Появится сообщение об успешном .

' Get' операция по той же переменной должен теперь вернуть новую 'Set ' значение для этой переменной . LED5 на демо-плате Теперь должен быть включен . Повторите процедуру, чтобы установить LED5 ВЫКЛ. LED6 также может быть установлен или выключается.

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

**7.2.1.2.6.4 Прерывания SNMP**

**модуль**

Сервер SNMP (Агент) ( см. стр. 103 )

**описание**

SNMP агент в версии 5.25 , а затем и компании Microchip TCP / IP стек поддерживает SNMP v1 и v2c отформатированные ловушки . Ловушки это уведомления от агента до менеджера , которые используются , когда предварительно определенная событие происходит на агента.

Несколько макрос препроцессора в файле заголовка вариант **TCPIPConfig**.h можно использовать для включения или отключения ловушки в агента.

Комментируя и снимите комментируя эти макросы в файле будут иметь разные последствия. SNMP_TRAP_DISABLED макро отключит ловушки полностью, если оно не прокомментировал:

**#define SNMP_TRAP_DISABLED**

Пользователь должен настроить ожидаемому формату ловушки на SNMP-менеджером . SNMPv2 организаций, действующих в качестве агента должно быть способны генерировать и передавать SNMP V2 ловушки PDU, когда менеджер настроен на полученной процесс и SNMP V2 ловушку

БРП . Чтобы настроить формат ловушки , комментировать или раскомментируйте следующие макросы в файле заголовка TCPIPConfig.h :

**#defiine SNMP_STACK_USE_V2_TRAP**

Если макрос был комментарий, агент SNMP пошлет V1 отформатированный ловушки PDU, ; в противном случае , он будет посылать V2 отформатирован ловушка щиты . По умолчанию

агент SNMP настроен на отправку V2 отформатированные ловушки . Обратите внимание, что агент SNMP V2c должны только отправить V2 отформатированные ловушки .

Чтобы включить ловушки в SNMPv3 , # **defiine** SNMP_V1_V2_TRAP_WITH_SNMPv3 макрос должен снять комментарий .

В следующей таблице показано, как включить / отключить ловушки для разных версий SNMP :

SNMPv1 SNMPv2c SNMPv3

TRAP v1

( Enabled по по умолчанию)

• Закомментируйте # **defiine** SNMP_TRAP_DISABLED макрос

• закомментируйте # **defiine** SNMP_TRAP_DISABLEDмакрос

• закомментируйте # **defiine** SNMP_TRAP_DISABLED макро

• Раскомментировать # **defiine** SNMP_V1_V2_TRAP_WITH_SNMPV3 макрос

TRAP v2 ( инвалидов по по умолчанию) (Не поддерживается )

• закомментируйте # **defiine** SNMP_TRAP_DISABLED макрос

• Раскомментировать # **defiine** SNMP_STACK_USE_V2_TRAP макрос

• закомментируйте # **defiine** SNMP_TRAP_DISABLED макро

• Раскомментировать # **defiine** SNMP_V1_V2_TRAP_WITH_SNMPV3 макрос

• Раскомментировать # **defiine** SNMP_STACK_USE_V2_TRAP макро

**демос**

Два ловушки демо включены в TCP / IP стека. Функции задача для этих демонстраций называются в главном приложении функция :

• **SNMPTrapDemo** ( ) - Этот API демонстрирует V1 или V2 форматы ловушки (в зависимости от статуса

SNMP_STACK_USE_V2_TRAP макро) . Ловушка PDU будет иметь только один демо переменной связывания в списке VarBind .

• **SNMPV2TrapDemo** ( ) - Этот API предоставляет уведомления формата V2 с несколькими ( 3 ) привязок переменных . Пользователь должен изменять или использовать эту процедуру в качестве эталона для отправки уведомлений формата V2 ловушки с несколькими привязками в списке VarBind.

Пользователь должен включить только один демонстрационный SNMP API одновременно. По умолчанию SNMPTrapDemo () API включен и SNMPV2TrapDemo ( ) закомментирован .

V1/V2 форматированные Ловушки с одной переменной Binding В заголовке файла TCPIPConfig.h :

• Раскомментируйте # **defiine** SNMP_TRAP_DISABLED

• Комментарий / / # **defiine** SNMP_STACK_USE_V2_TRAP

Для Trap демонстрации , два события определяются в агента V2c :

• Если значение аналогового потенциометра превышает 512 , которое может отправить агент ловушку каждые 5 секунд, чтобынастроить ' trapReceiverIPAddress .'

• При нажатии кнопки 3 на демо-плате , организация конкретных PUSH_BUTTON ловушка будет отправлен .

Текущая реализация агента V2c также генерирует стандартный «Проверка подлинности ( см. стр. 89 ) Отказ Trap " :

• Если запрос получен изменить (комплект) личное переменную MIB , или

• Если значение переменной запрашивается ( получить ) с помощью браузера с именем неправильный сообщества .

Порядок действий:

1 . Откройте "Дополнительно" в меню конфигурации , настройки версию SNMP '2 ' и настроить запись сообщества к "общественный ", " запись " или " личное " .

2 . Выберите переменную ' trapEnabled.0 ' от MIB дерева.

3 . Выберите 'Set' из меню операций .

4 . Введите '1 ' в поле значения окна SNMP SET .

5 . Выберите ' trapReceiverIPAddress.0 ' из MIB дерева.

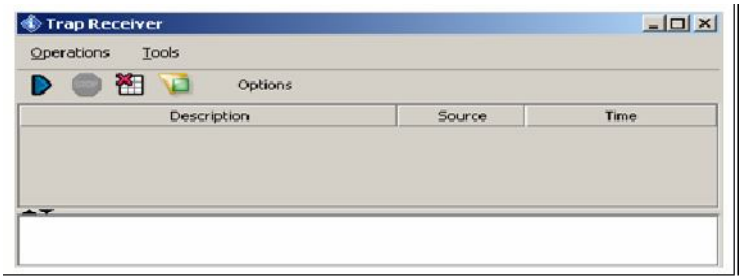
6 . Установите значение в IP адрес компьютера , на котором установлен браузер SNMP и эксплуатации.

7 . Выберите ' trapCommunity.0 ' из MIB дерева.

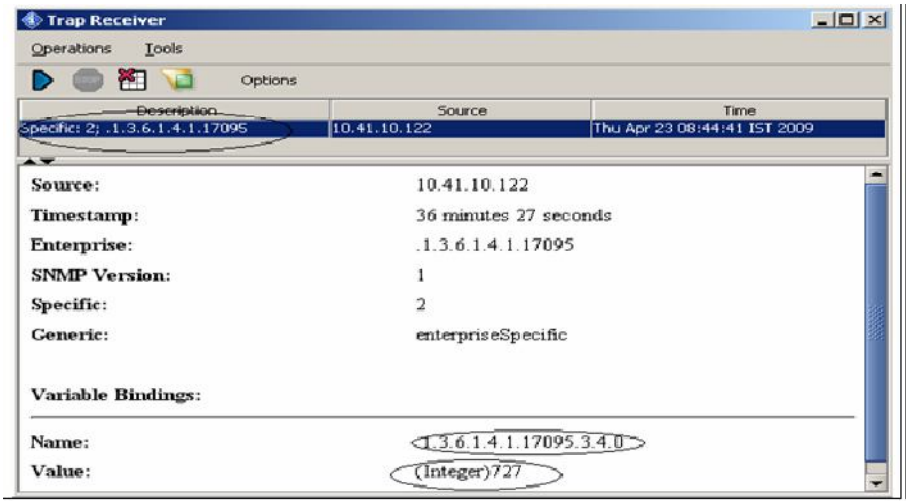
8 . Установить имя сообщества браузера SNMP ( сообщество по умолчанию , если не установлено, является "общественность" ) . ' TrapCommunity ' имя будет работать в качестве фильтра для SNMP -браузеров на сервере ловушка - мониторинга.

9 . Откройте " Trap Receiver ' утилита, которая была установлена с браузером iReasoning MIB ( Пуск-> Программы - > iReasoning - > MIB Браузер - > Trap Receiver ) .

MIB (Получатель Start->Programs->iReasoning->MIB Browser->Trap).




Для проверки аналогового потенциометра ловушку , отрегулируйте потенциометр на демо-плате , так что значение превышает 512 ( включите его по часовой стрелке) . Это предприятие по конкретным ловушка. SNMP -менеджер получит IP -адрес источника , идентификатор объекта ( в качестве имени переменной ) , то значение , штамп и т.д. для каждого события . Браузер будет интерпретировать данные как переменной AnalogPot информация на основе OID имени.



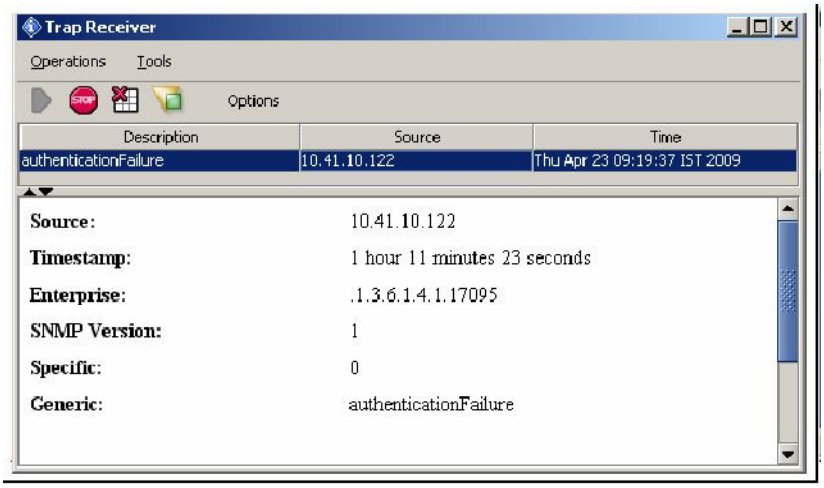

Для того, чтобы тестировать прерывание кнопки, нажмите подходящую кнопку на плату разработки (RB0 на PICDEM.net 2 или S3 на плате Explorer 16).

Для того, чтобы тестировать Аутентификацию Неудачи ( смотри страницу 89) прерывание, сконфигурируйте Чтению Общество в вашем окне просмотра в общественное имя, что не поддерживан агентом ( по умолчанию предусмотренные имена являются 'публикой и 'прочитанное'). Например:

1. Сконфигурируйте 'mchp в качестве имени сообщения прочитанными в браузере.

2 . Выберите личное MIB переменной LED5 от MIB дерева и выдать операцию 'Get ' из браузера .

В результате таблица в браузере, не будет отображаться никакого результата , но Trap Receiver получит и аутентификации ( смотри страницу 89) прерывание.




Это - intimation от агента до Менеджера SNMP, что была несанкционированная попытка, чтобы иметь доступ к частной переменной MIB в базе данных.

**V2 Форматировал Прерывания со Многочисленными Переменными Связями**

В заголовке файла TCPIPConfig.h :

- Uncomment #define SNMP_TRAP_DISABLED
- Uncomment #define SNMP_STACK_USE_V2_TRAP

Прерывание SNMP V2 структуры PDU:

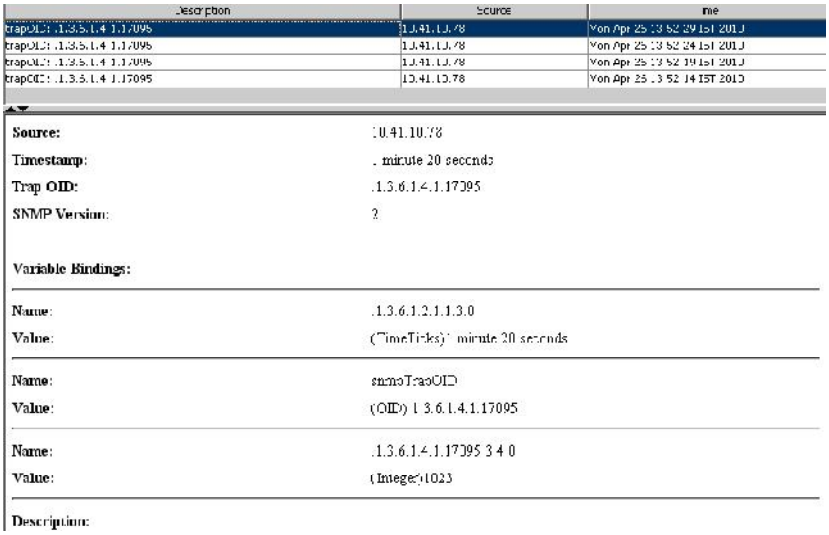
Версия (2) | общество | pdu- тип SNMP-PDU (TRAP=0xA7) | запрос-id | ошибка-статус | err-индекс | varbind Список

Первые две переменных varbinds в переменной, связывающей список SNMPv2-ПРЕРЫВАНИЯ-PDU - sysUpTime.0 и snmpTrapOID.0, соответственно. Если любые дополнительные переменные должны быть включены, тогда каждый из им varbind структур должно быть скопировано в переменную, связывающую список.

Для многочисленной переменной ПРЕРЫВАНИЯ SNMPv2 varbind демонстрация, ANALOG_POT0 использован, чтобы генерировать событие и передавать v2 Прерывание SNMP PDU. Отрегулируйте аналоговый потенциометр в величину больше, чем 512 (поворот это по часовой стрелке) на демонстрационной плате.
Дополнительно к sysUpTime.0 и snmpTrapOID.0 varbinds, дополнительные varbinds, что включены прерыванием PDU:

- PUSH-BUTTON
- LED0_IO
- ANALOG_POT0

Следующее цифры показывает V2 отформатировавшее прерывание с ANALOG_POT0 как переменная, связывающая, чтобы быть извещен.




На следующем рисунке показано многостраничного переменной VarBind при прерывании PDU SNMP V2 , с тремя дополнительными привязок переменных :

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

## 7.2.1.2.6.5 Конфигурации HTTP

**Module**
SNMP Server (Agent) ( see page 103)

**Description**

Если сервер http2 используется с Microchip Стек TCP / IP , можно динамически конфигурировать чтение и запись имена сообществ через веб- странице конфигурации SNMP . Следуйте инструкциям в начале раздела загрузить веб-страницы в энергонезависимой памяти , то доступ к веб- страницу конфигурации SNMP через навигационной панели. Используйте "Admin" имя пользователя и " микрочип " для пароля.

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

**7.2.1.2.6.6 Functions**

**Функций**

**SendNotification** ( смотри страницу116) - Подготовка , проверка удаленного узла , который будет получать ловушку и отправить перехваты PDU

**SNMPGetTimeStamp** ( СМОТРИ страницу 116) - Получает текущее значение Tick для отметки времени SNMP.

**Module**
SNMP Server (Agent) ( see page 103)

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

7.2.1.2.6.6.1 **SendNotification Function**

**File**
CustomSNMPApp.c

**C**
```
static BOOL SendNotification(
BYTE receiverIndex,
SNMP_ID var,
SNMP_VAL val,
UINT8 targetIndex
);
```

**Description**
Эта программа подготавливает уведомление прерывания pdu, посылает ARP и получает дистанционный адрес устройства MAC на который уведомление на посланное, посылает уведомление. Государственная машина Notofication корректироваться если есть любое разрешение неудачи ARP для адреса расположения прерывания perticular.

**Remarks**
None.

**Preconditions**
SNMPTrapDemo() is called.

**Parameters**
receiverIndex - индекс массива , где удаленный адрес IP хранится.
var - SNMP var ID, идентификатор, который должен быть использован в уведомлении

val - Величине уведомления о var. Только величина БАЙТА, СЛОВА или DWORD может быть послана.
targetIndex - snmpv3 Целевой индекс

Обратные Величины

ИСТИНА - Если уведомление отправить успешно.
ЛОЖЬ - Если уведомление отправить не удалось.

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

### 7.2.1.2.6.6.2 SNMPGetTimeStamp Function

**File**
CustomSNMPApp.c

**C**
```
static DWORD SNMPGetTimeStamp();
```

**Description**

Эта функция получает абсолютные измерения времени для SNMP времени. Используйте **TickGet** (смотри страницу 603) и **TickGetDiv64K** ( смотри страницу 604), собрать все 48bits внутреннего **Tick** таймера.

**Remarks**
None.

**Preconditions**
None

**Return Values**
Отметка времени - DWORD timevalue

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

7.2.1.2.6.7 **Variables**

**Module**
SNMP Server (Agent) ( see page 103)

**Variables**

**gSendTrapSMstate** ( seepage 117)- Это - переменная gSendTrapSMstate.

**gSnmpNonMibRecInfo** (see page 117) - OLD snmp СТАРЫЙ snmp.mib файл со стандартом SMIv1

**gSnmpv3UserSecurityName**( see page 117) - переменная gSnmpv3UserSecurityName.

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

7.2.1.2.6.7.1 **gSendTrapSMstate Variable**

**File**
CustomSNMPApp.c

**C**
UINT8 **gSendTrapSMstate** = 0;

**Description**
Это - переменная gSendTrapSMstate.

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

#### 7.2.1.2.6.7.2 gSnmpNonMibRecInfo Variable

**File**
CustomSNMPApp.c

**C**
SNMPNONMIBRECDINFO **gSnmpNonMibRecInfo**[SNMP_MAX_NON_REC_ID_OID] = {
{{43,6,1,4,1,0x81,0x85,0x47,6},SNMP_V3}, {{43,6,1,2,1,1},SNMP_V2C},
{{43,6,1,4,1,0x81,0x85,0x47,0x1},SNMP_V2C}, };

**Description**
OLD СТАРЫЙ snmp.mib файл со стандартом SMIv1

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

#### 7.2.1.2.6.7.3 gSnmpv3UserSecurityName Variable

**File**
CustomSNMPApp.c

**C**
BYTE **Snmpv3UserSecurityName**[SNMPV3_USER_SECURITY_NAME_LEN_MEM_USE];

**Description**
Это - переменная gSnmpv3UserSecurityName.

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

#### 7.2.1.2.6.8 Macros

**Macros**

**MAX_TRY_TO_SEND_TRAP** ( смотри страницу 118)

**SNMP_MAX_NON_REC_ID_OID** ( смотри страницу 118)- Обновите Non значение OID Индентификационный который является частью CustomSnmpDemo.c

**STACK_USE_SMIV2** ( смотри страницу 118) - По умолчанию STACK_USE_SMIV2 включена . Для Stack V5.31 , STACK_USE_SMIV2 должны быть отключены.

**Module**
SNMP Server (Agent) ( see page 103)

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

#### 7.2.1.2.6.8.1 MAX_TRY_TO_SEND_TRAP Macro

**File**
CustomSNMPApp.c

**C**
```
#define MAX_TRY_TO_SEND_TRAP (10)
```

**Section**
Global Variables

**********
Этот макрос используется , чтобы обеспечить максимальную попытку для адреса сервера отказ Trap

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

#### 7.2.1.2.6.8.2 SNMP_MAX_NON_REC_ID_OID Macro

**File**
CustomSNMPApp.c
**C**
```
#define SNMP_MAX_NON_REC_ID_OID 3
```

**Description**
Обновите Non значение OID Индентификационный который является частью CustomSnmpDemo.c файл

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

#### 7.2.1.2.6.8.3 STACK_USE_SMIV2 Macro

**File**
CustomSNMPApp.c

**C**
```
#define STACK_USE_SMIV2
```

**Description**
По умолчанию STACK_USE_SMIV2 включена . Для Stack V5.31 , STACK_USE_SMIV2 должны быть отключены.

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

## 7.2.1.2.7 ОБЗОРОВ Моста UART-TCP

**Характеристика**
Функция UART - на - TCP мост из TCP / IP Demo App передает все входящие байты TCP на сокете изПИК

Модуль UART и все входящие UART байт из сокета TCP .

**инструкции**

1 . Составьте свой MPLAB проект и запрограммировать демо доска .

2 . Подключите порт RS-232 на демо борту к RS-232 порт на вашем компьютере. На новой компьютера вам может понадобиться RS-232 для преобразователя кабеля USB . На компьютере откройте программу терминала (например, HyperTerminal ) . Установите его использовать COM-порт подключен ваш совет , чтобы , по крайней 19200 бод , 8 бит данных, без проверки четности , 1 стоповый бит , без контроля потока .

3 . Подключите запрограммированный демо доска к компьютеру напрямую или через маршрутизатор . Для Ethernet, прямое подключение может потребовать перекрестного кабеля; для WiFi , доска возможно, должны быть в режиме AdHoc чтобы установить прямое соединение .

4 . Определить IP адрес демо-плате . Это может быть сделано несколькими различными способами .

1 . Если вы используете демо установку с ЖК-дисплеем (например, Explorer, 16 или PICDEM.net 2) ,IP- адрес должен быть отображается на второй строке дисплея .

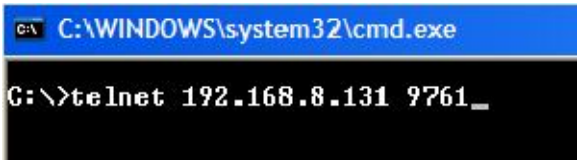
2 . Откройте Device Discoverer Microchip Ethernet из стартового меню . Нажмите кнопку " Поиск устройств ", чтобы увидеть адреса и имена хостов всех устройств сОбъявить ( см. стр. 155 ) Протокол включен в вашей сети. вы Возможно, придется настроить брандмауэр компьютера , чтобы предотвратить его блокирование UDP порт 30303 для этого решения .

3. Если ваш совет подключается непосредственно к компьютеру с помощью кросс-кабеля . :

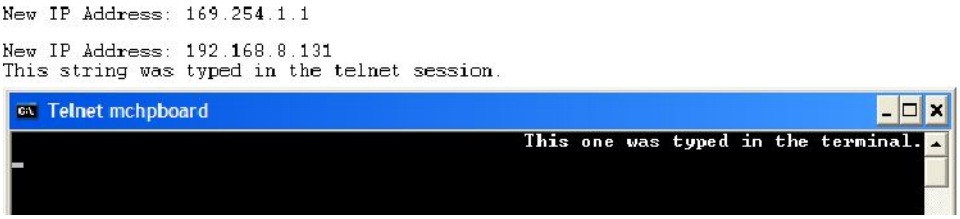
1 . Откройте командную / DOS и наберите " IPCONFIG " . Найти сетевой адаптер , подключенный к плате . IP-адрес совета находится в поле ' Default Gateway "

2 . Откройте состояние сети для сетевого адаптера, который соединяет два устройства. Это может быть сделано по праву нажав на значок сетевого соединения в папке сетевых настроек и выберите ' статус ' из меню . Найти " Основной шлюз " поле .

5 . Откройте командную / DOS . Тип " Telnet ip_address 9761 " , где ip_address является IP-адрес, который вы получили от шаг




6 . По мере ввода символов в командной строке , то они будут переданы по **Telnet** (см. стр. 572) TCP порта ПИК , а затем передаются из UART в ПОС , чтобы появиться на терминальной программы . По мере ввода символов в терминал программа , они будут переданы в PIC через модуль UART , а затем повторно передается через TCP подключение появляться в командной строке Telnet ( см. стр. 572) сессии.




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

### 7.2.1.2.8 без настройки ( ZeroConf )

Нулевая конфигурация ( Zeroconf ) , предоставляет механизм , чтобы облегчить конфигурацию устройства в сети. Он также предусматривает более человек-как именования , а не полагаться на IP -адреса в одиночку. Zeroconf также известен под именами Bonjour (Apple) и Avahi (Linux) , и является IETF стандартом.

**Включение**

Zeroconf можно включить, установив следующие два определяет, в TCPIPConfig.h :

• STACK_USE_ZEROCONF_LINK_LOCAL

• STACK_USE_ZEROCONF_MDNS_SD

В настоящее время , использование Zeroconf ограничивается WiFi демонстрационных приложений (и модуля MRF24WB0M ). Будущие версии стек должен включить поддержку Zeroconf во всех решений Ethernet.

**Ссылка Часовой**

Первый компонент Zeroconf является возможность самостоятельно назначить IP -адрес каждого члена сети. Как правило, DHCP Сервер будет обрабатывать такие ситуации. Однако в тех случаях, когда нет DHCP -сервер не существует , Zeroconf включен устройства переговоры уникальный IP- адреса между собой .

**mDNS**

Второй компонент Zeroconf является возможность самостоятельно назначить человеческого понимания хостов для себя. Multicast DNS предоставляет локальной сети возможность иметь

черты DNS сервера. Пользователи могут использовать легко запоминающиеся имена хостов , чтобы получает доступ к устройствам в сети . В случае, если устройства избирать использовать то же имя хоста , как и в IP- адрес разрешение, каждое из устройств будет автоматическое согласование новых имен для себя (обычно путем добавления ряд  до конца название ) .

**Service Discovery**

Последний компонент **Zeroconf** является открытие сервис . Все устройства **Zeroconf** может транслировать какие услуги они предоставляют . для Например, принтер может транслировать , что он имеет полиграфические услуги , доступные . Термостат может транслировать , что он имеет HVAC управления услугами . Другие заинтересованные стороны в сети , которые ищут определенные услуги могут увидеть список устройств какие имеют возможность предоставить  услуги и подключения (см. стр. 171 ) непосредственно к нему . Это дополнительно исключает необходимость знать , существует ли что-то в сети (и что это IP или имя хоста ) . В качестве конечного пользователя , все, что вам потребуется сделать, будет запрашивать из сети , если некий сервис существует, и легко подключиться (см. стр. 171 ) к нему.

**демонстрация**

Демо , когда включен , показывает все три пункта выше , работающих вместе. Каждый комплект разработчика в сети предполагает Имя хоста MCHPBOARD - x.local , где хчисло приращение от 1 (только в том случае , когда несколько комплекты запрограммирован для сети). Каждая плата будет транслировать это услуга, которая является DemoWebServer .

**Zeroconf Включено Среды**

Все продукты Apple, включена Zeroconf по умолчанию. В Windows , вы должны загрузить веб-браузер Safari , и В ходе инсталляции , включите поддержку Bonjour. Обратите внимание, что в браузере Safari , вы можете просматривать и увидеть список всех Bonjour совместимыми устройствами и перейти на них автоматически.

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

### 7.2.2 Интернет загрузчика

Интернет загрузчика является самостоятельным приложением позволяет новая прошивка приложение для загрузки непосредственно в Flash, память о PIC18F микроконтроллера по сети Ethernet или через Интернет. Для других ПОС и DSPIC архитектур , сторонних TCP / IP загрузчики могут быть получены из http://www.brushelectronics.com/ . Это загрузчик приложений Интернет реализует свой собственный UDP / IP стек , а также упрощенного протокола передачи файлов (TFTP ) сервера . Загрузчик работает независимо от основного приложения и не может обновлять себя . Гарантии реализованы внутри , чтобы свести к минимуму риск неустранимая удалось обновления .

Важные атрибуты Интернет загрузчика включают в себя:

• Автономный TFTP , UDP, IP , ARP, и обращение протокол Ethernet

• Выполняет на сброса по включению питания а не во время основного приложения

• Ожидает около 4 секунд , прежде чем начать основное приложение

• Требуется 8 Кбайт программного флэш

• Требуется 0B оперативной памяти ( все привыкли ОЗУ обложен основного приложения )

• Не требует процессорного времени при выполнении основного приложения

• Требуется минимальное или никаких изменений в коде главной приложений и компоновщика сценария

• Не мешает с векторными приложение прерывания местах или добавить задержки при обработке прерываний

• Может перепрограммировать конфигурации слова

• Может повторно MAC и IP -адрес, предоставленный основного приложения

• обновление Клиентское программное обеспечение уже доступно на большинстве компьютеров

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

**7.2.2.1 загрузчика Дизайн**

Загрузчик Вступление

Загрузчик представляет собой сервер TFTP , который автоматически запускается на сброса по включению питания (POR) . Он может быть расположен в любом месте в пределах памяти программ . Чтобы вызвать автоматический запуск , загрузчик прозрачно выполняет замену инструкцией (ы) в местах памяти программ 000000h - 000003h . . Шестнадцатеричный файл для программирования на чип с помощью загрузчика будет нормально содержат инструкции GOTO по адресу 000000H которая разветвляется к основному приложению . Вместо того чтобы писать оригинал Инструкция по адресу нулевой , загрузчик создает новую инструкцию GOTO , который всегда филиалы в начальный адрес код загрузчика . Оригинальный Инструкция по применению с нулевого адреса перемещается в таблицу переходов , которая позже под названием для выхода загрузчик . Таблица переходов также содержит GOTO 000004h инструкцию для обеспечения нормальной работы приложений , если первый Инструкция не был GOTO .

Если устройство программируется только с загрузчиком (без приложений ), адрес 000000H через начальный адрес код загрузчика будет в незапрограммированном состоянии ( FFFFh ) . Эти инструкции NOP , которая не будет быстро выполнять до тех пор, Программный счетчик достигает начало загрузчика . Это гарантирует въезд в загрузчик для обоих запрограммированы и незапрограммированные части.

**Загрузчик Re - запись**

Если работает приложение хочет повторно загрузчик , он должен очистить RCON <NOT_POR> немного, а затем выполнить СБРОС инструкция . Когда загрузится возвращает управление основного приложения ,NOT_POR бит будет в установленном состоянии . Если приложение должно быстро сбросить , не дожидаясь тайм-аут загрузчика , он должен оставить этот бит в установленном состоянии . Это воля вызвать загрузчик , чтобы пропустить его нормальной работы и немедленно вернуться в приложение.

До выполнения команды RESET, чтобы повторно загрузчик , основное приложение может указать адрес MAC и IP для загрузчик для использования. Для этого:

1 . Выберите случайную ячейку памяти , где 12 байт может быть записан

2 . Скопируйте MAC -адрес выбранной ячейки памяти по смещению 0

3 . Скопируйте IP -адрес выбранной ячейки памяти по смещению 6

4 . Вычислить IP контрольную сумму MAC-адреса и IP хранится в ячейках памяти от 0 до 9

5 . Напишите IP контрольной суммы в выбранной ячейке памяти по смещению 10

. 6 Храните адрес выбранной ячейки памяти в PRODH : регистры Предль

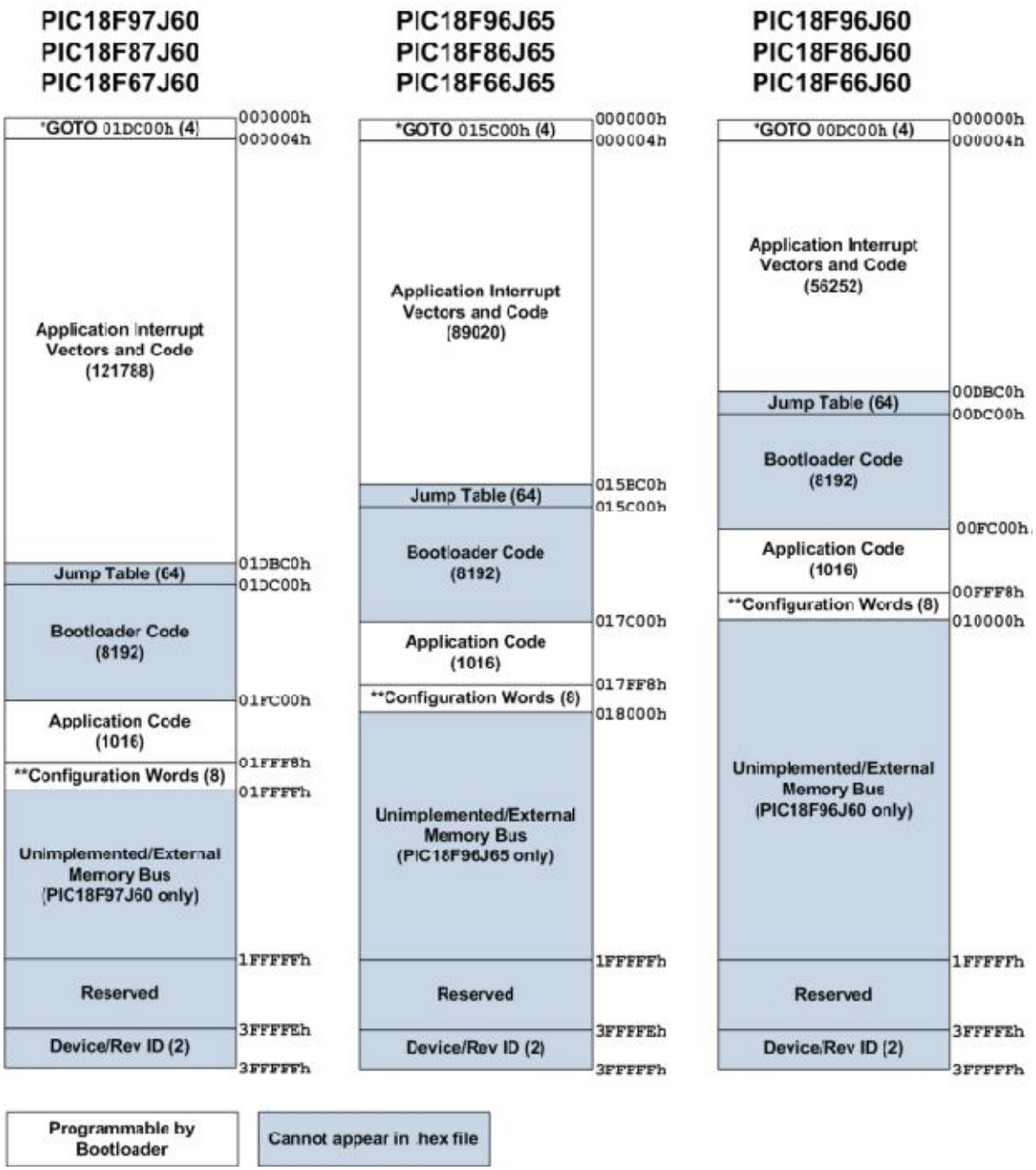
7 . Снимите RCON <NOT_POR> немного и выполнить команду RESET, чтобы войти в загрузчик

После вступления , загрузчик обнаружит , что бит RCON <NOT_RI> ясно , что указывает на загрузчик был введен с Основное применение вместо подлинного события ПОР . В этом случае загрузчик будет разыменовать PRODH : PRODL указатель , проверки контрольной суммы , и если действует , используйте MAC и IP -адрес, указанный . Если контрольная сумма является недействительным, загрузчик будет использовать это по умолчанию скомпилирован MAC и IP- адрес .

Библиотека Microchip Стек TCP / IP обеспечивает Reboot ( см. стр. 317) модуль, который может выполнять описанную выше процедуру при обнаружении пакета TFTP. Когда Перезагрузка (см. стр. 317 ) модуль используется ,IP- адрес приложения (возможно, получены автоматически через DHCP) может быть использован в качестве мишени TFTP загрузчика . Если IP- адрес, используемый основного приложения является Интернет маршрутизации , то сам загрузчик будет доступен через Интернет.

**Карта памяти**

Вся карта памяти программ при использовании загрузчика показан ниже.




*: ИНСТРУКЦИЯ GOTO автоматически сгенерирована bootloader при записи. Прикладная инструкция по адресу 000000h перемещена на Таблицу Перехода Bootloader.

**: Некоторые опции конфигурации не поддерживаны и автоматически будет изменен bootloader перед миганием. bootloader Требует HS или режим генератора HS+PLL и не поддерживает 1:1 и 1:2 режима Watchdog Timer Postscale.
Переключаясь между Расширенным и не-Расширившее режим не поддерживан также. ( bootloader Должно быть перекомпилировано, чтобы изменять modes.)

Загрузчик использует блок 8256 байт памяти программ. Чтобы запретить приложению непреднамеренно использовать это блок , вы должны изменить сценарий компоновщика в вашем приложении перед компиляцией . Например, для MPLAB ® C18 C компилятор , сценарий линкер будет содержать CODEPAGE строку, описывающую доступной флэш-

памяти в устройстве. Для PIC18F97J60 продукт с 128 КБ памяти программ , сценарий линкер ( 18f97j60i.lkr ) будет содержать строку , таких как:

CODEPAGE NAME=page START=0x2A END=0x1FFF7

Эта линия указывает, что компоновщик может разместить констант приложения и код где угодно между 00002Ah и 01FFF7h . это линия должна быть разделена на две CodePage линий для описания разрыва в доступной памяти программ , занятой загрузчика . Пример:

CODEPAGE NAME=page START=0x2A END=0x1DBBF

CODEPAGE NAME=page2 START=0x1FC00 END=0x1FFF7

Приведенный выше пример удаляет таблицу переходов и код загрузчика блоки для PIC18F97J60 с 128 КБ флэш памяти . Другие устройства с меньшим флэш-памяти нужно будет использовать разные начальные и конечные значения в соответствии с Перейти таблиценачальный адрес и код загрузчика конечный адрес описано в карте памяти рисунке выше .

Сервер TFTP выполняет " объемную стирание " перед началом любой TFTP положить операции (запись) . Стирания не является истинным стирайте что загрузчик и конфигурации слова остаются нетронутыми. Однако все другие местоположения вернулась к их запрограммирован состояние . Процедура стирания начинается со страницей флэш содержащей таблицу переходов и продолжает назад в памяти к адресной 000000H . После адрес 000000h стирается,страница последняя программа памяти, содержащая конфигурации устройств слова стирается. Например, предполагая PIC18F97J60 с 128 КБ Flash, процедура стирания

выполните следующие действия:

1. Стирание 01D800h-01DBFFh
2. Стирание 01D400h-01D7FFh
3. Стирание 01D000h-01D3FFh
4....
5. Стирание 000400h-0007FFh
6. Стирание 000000h-0003FFh
7. Стирание 01FC00h-01FFFFh

После последняя страница, содержащая слово конфигурации будет выстирана, слово конфигурации немедленно перепрограммируют в их предшествующую величину. Этот алгоритм обеспечивает очень прочную операцию чрезвычайно низкой вероятностью разрушающего доступа к bootloader из-за неожиданного события (экс: полномочие или сетевая связность потерены bootloading). Неожиданные события оставит первую инструкцию GOTO по адресу 000000h целую, проверяя, что bootloader запустит снова. Поскольку слово конфигурации выстираны последним, там не будут любыми средствами, обходящими внутренний код защищать характеристику пока кодовые неподвижные остатки приложения на устройстве.

**Операции Программы**

Операции Программы выполняются последовательно запускаясь по адресу 000000h и выращивая вверх, как представлено в .hex файле, который нужно программироваться. Слово конфигурации устройства являются обычно последними величинами сталкивался в .hex файле. Поскольку процедура стирания включает очищая слово конфигурации и затем немедленно перепрограммируя он, слово конфигурации уже будут запрограммированы когда слово конфигурации столкнулись в .hex файле. Следовательно, если .hex файл содержит другие слово конфигурации из которых уже загружены в память Вспышки, bootloader должно выполнить новую операцию стирания на последней странице до программирования новых слова конфигурации. Это дополнительное стирание/запись цикла уменьшит общую выносливость Вспышки на последней странице по сравнению с остальной частью устройства. Тем не менее, bootloader не выполнит это стирание/записывать если слово конфигурации не изменились бы. Эта характеристика сохраняет выносливость для большинства прикладной модернизации микропрограмм, какое обычно не требует другие опции конфигурации, которые нужно программироваться.

**Прочитайте Операции**

Для того, чтобы сохранять кодовое пространство, bootloader к настоящему времени только опоры, читающие сервер TFTP как двоичные данные. Вместо получения .hex файла из TFTP, получите операцию, bootloader возвращает двоичный файл измеренное на сумму внутренней памяти Вспышки доступной (128kB для PIC18F97J60, 64kB для PIC18F66J60, и т.п..). bootloader Проверяет код немедленно после программирования устройств, так что характеристика чтения - первоначально для отладки только.

Прочитайте операции выведены из строя если к настоящему времени программируемое приложение имеет PIC  microcontroller Код Защищает характеристику включенную.

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

**7.2.2.2 Используя Операцию Bootloader**

После того, как bootloader запустилось, код приспособится модуль Ethernet и начинает выполнять частного UDP/IP стека. Это использует следующие по умолчанию адресы из Полномочия-в Сбросе:

- АДРЕС IP ( смотри страницу 147): 192.168.97.60
- Адрес MAC ( смотри страницу 147): 00-04-A3-00-00-00

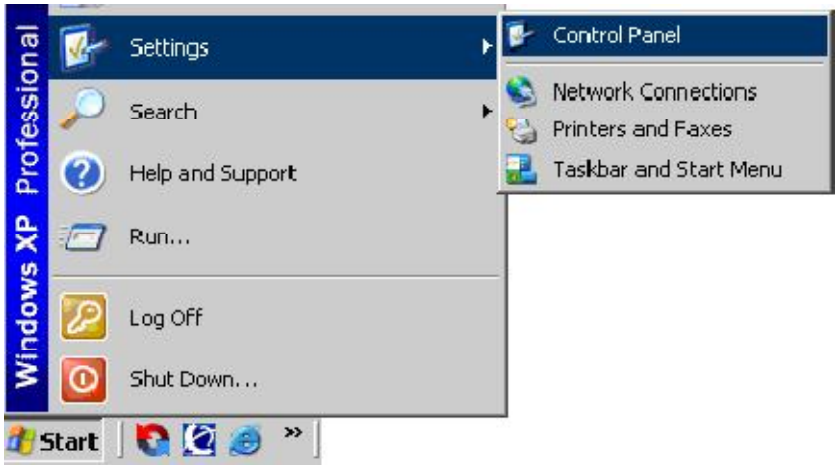
Этих по умолчанию адресов статически определены и могут только изменены перекомпиляцией bootloader собой. Тем не менее, если bootloader вызвано из основного приложения, как например, с Перезагрузкой модуля ( смотри страницу 317), тогда bootloader использует приложение назначившее IP и адресы MAC (если предусмотрено).

Единственные услуги, которые доступны в течение bootloader операции - TFTP и ARP. ICMP (ЗВОН) и другие услуги не осуществлены.

**Выбор конфигурации Ваше PC (Полномочие-на входе Сброса)**

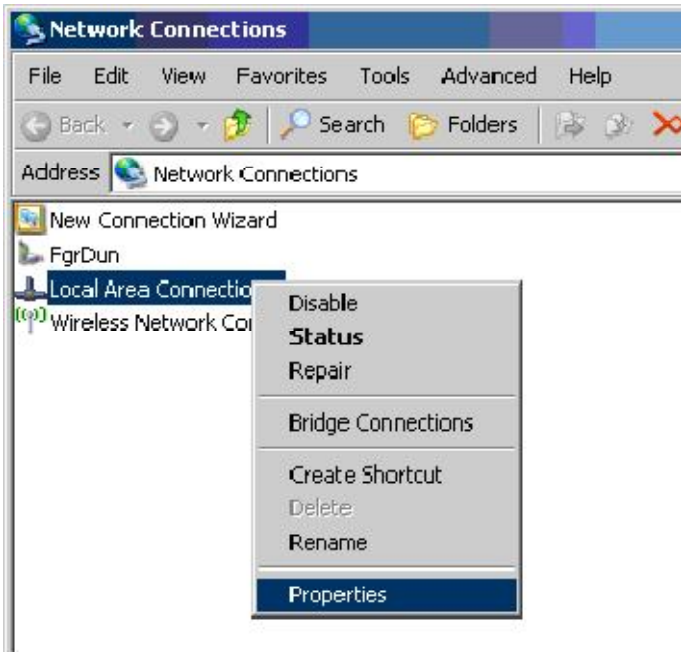
Чтобы получить доступ к загрузчик , IP- адрес загрузчика должна быть в той же подсети , что и компьютер . Для умолчанию 192.168.97.60 IP адрес , необходимо временно изменить настройки на вашем компьютере . Если IP- адрес загрузчик был указанное приложение и уже в той же подсети , что и ваш компьютер , то этот раздел должен быть пропущен.

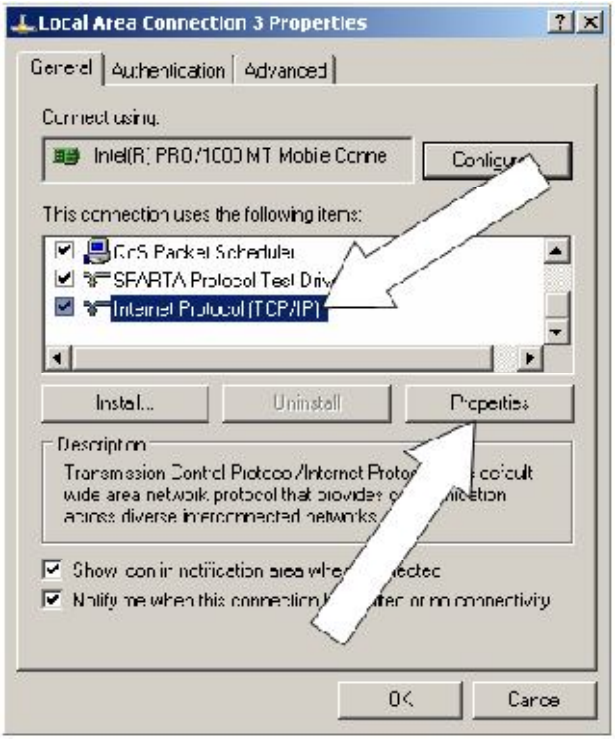
Следующие инструкции предполагают, что Вы используете Microsoft ® Windows ® XP и будет отличаться для других операционных систем.




1 . Открыть Сетевые подключения .

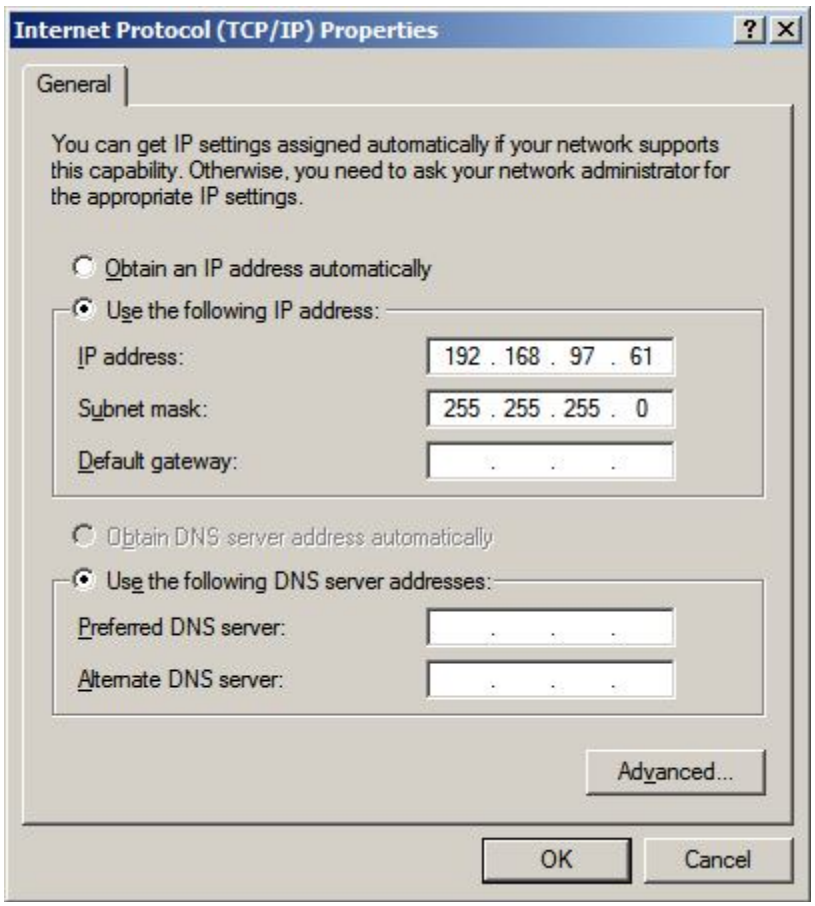
2 . Щелкните правой кнопкой мыши на сетевом адаптере , который вы используете для связи с загрузчиком и выберите Свойства.




3 . Выберите Протокол Интернета (TCP / IP) и нажмите Свойства.

4 . Выберите Использовать следующий IP- адрес , а затем введите IP -адрес 192.168.97.61 .




5 . Нажмите кнопку ОК, а затем Закрыть в предыдущем диалоговом окне , чтобы закрыть их и установить новый адрес .

**ОПЕРАЦИЯ TFTP (**включение питания вступления Reset )

Наиболее операционные системы приходят с клиентом TFTP встроенным в. В Microsoft Windows, эта утилита названа tftp.exe. Эта утилита является тем же простым консольным приложением, которое может быть использовано, чтобы загружать ваш прикладной .hex файл над сетью в bootloader.
Для выполнения Flash, модернизировать с помощью клиента tftp.exe , выполните следующие действия :

1 . В консоли , введите следующую команду , но не выполнять его. Сделать соответствующие изменения Путь к шестнадцатеричный файл . . TFTP 192.168.97.60 пут "C: \ Microchip Решения \ TCPIP Demo App \ TCPIP Demo App- C18.hex "

2 . Выключите и включите целевая плата или если устройство имеет MCLR  кнопка сброса , нажмите на нее .

3 . Быстро нажмите клавишу ВВОД , чтобы выполнить команду TFTP . Если прошивка уже в устройстве, загрузчик автоматически прекратить после примерно 4 секунд , так что вы должны выполнить команду TFTP рамках второго окна 4 .

4 . В случае успеха , клиент TFTP будет указано, как долго трансфер занял . Фактическое время программирования будет меняться в зависимости от многочисленные факторы , в том

числе необходимости стереть вспышки первый , . размер файла Hex , . сложности шестигранной файлов и внутреннего программирования Время . Сообщил Скорость передачи Поэтому не очень хорошая метрика производительности сети во встраиваемых приложениях.

bootloader же данные считываются обратно проверка вскоре после написания и не нужен второй шаг, чтобы прочитать его содержимое флэш . Если ошибка проверка происходит , ошибка будет немедленно сообщил клиенту TFTP.

Наиболее вероятной причиной проверки Отказ не является проблемой вспышки выносливость, а скорее , инвалидом. шестнадцатеричный файл в качестве ввода . Как показано в загрузчике Карта памяти , . шестигранные файлы не могут содержать никаких данных в пределах зоны 8 кб флэш были загрузчик хранится .

загрузчика внутренне маскирует этом регионе вспышкой и рассматривает его как только для чтения , чтобы предотвратить коррупцию загрузчика . В результате если шестнадцатеричный файл . содержит данные в регионе только для чтения,записи не будет выполнена и проверка покажет несоответствие .

5 . После успешного записи , загрузчик будет тайм-аут после примерно 4 секунд и начать выполнение основной приложение, которое было только что загрузили .

После завершения процесса загрузки TFTP , восстановить настройки IP- адрес вашего компьютера , чтобы позволить нормальную сетевую активность и доступ к приложению вы bootloaded .

**TFTP Операция (запись Применение )**

При использовании приложение, которое автоматически определяет пакеты TFTP и входит в загрузчик при необходимости , например, перезагрузки (см. 317 модуль страница ) в Microchip Стек TCP / IP , то не будет вообще никакой необходимости перенастроить компьютер или пройти через

чувствительным ко времени процесс питания на велосипеде . Вместо этого, вы можете выполнить операцию TFTP непосредственно на устройстве без каких-либо интерактивные шаги .

1 . В консоли , введите следующую команду и выполнить ее . Сделать соответствующие IP adddress / имя хоста и путь изменений.

TFTP mchpboard поставить " **C: \ Microchip Решения \ TCPIP Demo App \ TCPIP Demo App- C18.hex "**

Если процесс Bootload прерван из-за сбоя сети или отмены пользователя , вы можете просто повторите команду TFTP .

Загрузчик не будет пытаться запустить частично bootloaded приложения. Указано Приложение MAC и IP- адрес будет на неопределенный срок сохраняться , пока устройство не выключении и повторном включении или перезагрузки .

Если операция Bootload прерван из-за сбоя питания , загрузчик начнет резервное копирование с помощью сброса по включению питания по умолчанию MAC и IP- адресов. В этом случае , вы должны следовать Сброс направления ввода питания - на восстановить .

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

### 7.2.3 WebVend

TCPIP WebVend App пример веб - включен автомат приложение . Он используется серии TCP / IP вебинар :

1 TCP / IP-сети Часть 1: . Web-Based Мониторинг Статус (просмотр)

2 TCP / IP-сети Часть 2: . Управление Web-Based (просмотр)

3 TCP / IP-сети Часть 3: Расширенный контроль по сети (просмотр)

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

### 7.2.4 Радио Internet

Не ВАЖНО: Из-за изменений в протоколе SHOUTcast ,демонстрационное приложение интернет-радио больше не в состоянии выполнять свои функции . Это демо сейчас существует только в качестве примера кода TCP / IP стека .

Интернет-радио приложение демонстрирует использование TCP / IP стека для автономного встроенного приложения . Это приложение способен связаться с различными SHOUTcast серверов и воспроизведения аудио поток с парой стереодинамиков .

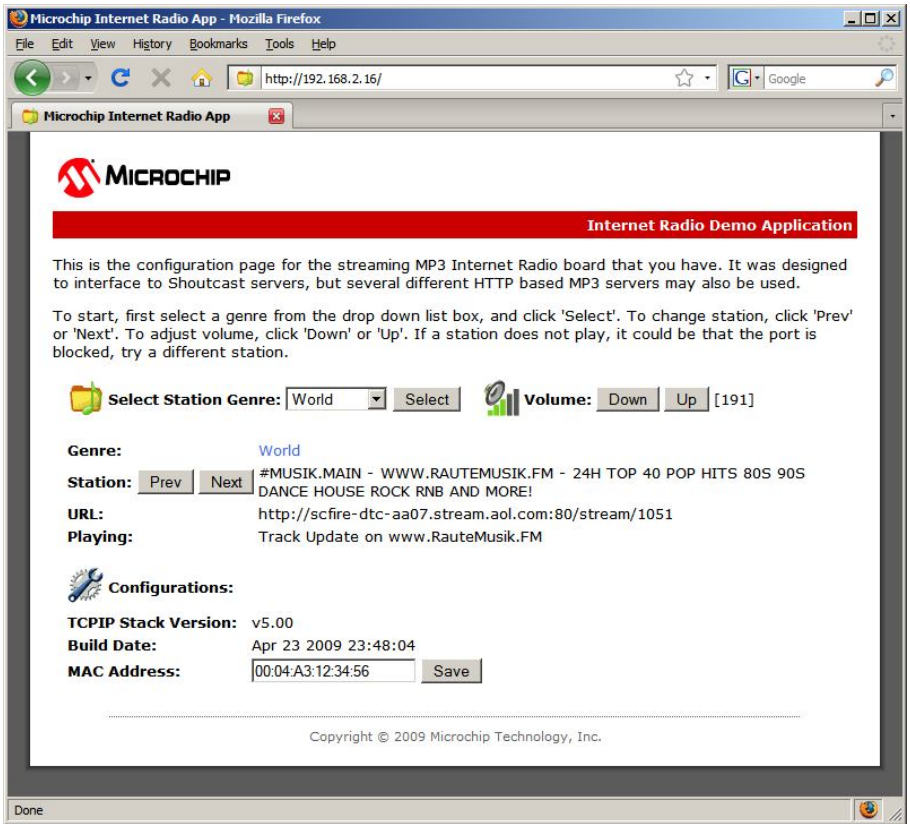
Демо требует демонстрационная доска Интернет Радио . PIC18F67J60 используется для обработки интерфейса Ethernet , в то время как внешний декодер MP3 обрабатывает воспроизведение. Указания по применению AN1128 " TCP / IP-сети : интернет-радио

Использование OLED-дисплей и MP3 Audio Decoder ( DS01128 ) " описывает применение Интернет-радио в деталях.

Для запуска демо, сначала убедитесь, что Интернет-радио доска имеет правильную прошивку запрограммированный . Затем подключите (см. стр. 171) на плате к интернету , подключить аудио гарнитуры или громкоговорителя. По умолчанию , программа не будет играть радиостанцию автоматически пока не будет выбран жанр . Следуйте OLED экранного на экране меню , чтобы изменить жанр, станция, и объем.

Совет может также управляться через интерфейс веб-браузера. Для подключения (см. стр. 171 ) , чтобы веб-сервера платы, использования IP адрес отображается на OLED дисплее платы. Ниже показан снимок экрана веб-страницы. Для начала , сначала выберите жанр из выпадающего списка и нажмите кнопку "Выбрать" . Для смены канала , нажмите кнопку " Назад " или " Далее" . Для регулировки громкости , нажмите "вниз" или «Вверх» . Если станция не играет , это может быть , что порт заблокирован , попробуйте другую станцию .

Каждый интернет-радио доска также имеет наклейку , содержащую уникальный MAC- адрес . Этот уникальный MAC- адрес может быть сохранен на плата с помощью конфигураций раздел веб- интерфейса.




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

### 7.2.5 КОНСОЛИ WiFi

### 7.2.5 Wi-Fi консоли

TCPIP WiFi Консоль Демо App ( ранееTCPIP WiFi Iperf Демо App) является интерфейс командной строки (CLI) , чтобы MRF24WB0M . Это позволяет для командной строки отладки и настройки сетевой информации для беспроводной локальной сети. Она также имеет **Iperf** построен в для ведения WLAN тестирования пропускной способности . Это приложение предназначено , скорее, как развитие для отладки , так и должно быть отключена в приложениях конечных пользователей .

#### 7.2.5.1 Автономные Команды

Эти команды CLI не связаны с беспроводной или сетей интерфейс напрямую.

**help**- Содержит список всех доступных команды консоли для MRF24WB0M .

**getwfver** - Включает микропрограммы WiFi и главные драйверные номера версии.

**reset**- Проблемы сброс хоста.

**cls** - Сброс строки.

**kill iperf** - Убийство работающей **Iperf** сессии. См. раздел по **Iperf** (см. стр. 131 ) для получения дополнительной информации .

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

#### 7.2.5.2 Команды iwconfig

Обратите внимание, что большинство из этих пунктов не должны изменяться , когда устройство находится в подключенном состоянии к сети .

команды **iwconfig** принять следующую структуру:

iwconfig [ ssid <name> ]
[ mode <idle|managed|adhoc> ]
[ channel <channel list|all> ]
[ power <reenable|disable|unicast|all> ]
[ domain <name> ]
[ rts <length> ]
[ txrate <0|1|2> ]

[ scan ]
[ hibernate ]
[ wakeup ]

Примечание: **iwconfig** без опций отобразит беспроволочный статус.

**ssid** - имя - 1-32 символа ASCII. К настоящему времени не принимает пробелы ( смотри страницу 169) в имени SSID.

**idle** - режим- ожидание Заставляет MRF24WB0M, чтобы отключать от любой к настоящему времени связанной сети (специальной или инфраструктуры).

управлявшим MRF24WB0M соединит ( смотри страницу 171) на SSID в режиме инфраструктуры. Отметьте, что все сетевые параметры должны быть правильно прежде, чем эта команда будет вызвана.

специальный MRF24WB0M соединит ( смотри страницу 171) на SSID в специальном режиме. Отметьте, что все сетевые параметры должны быть правильно прежде, чем эта команда будет вызвана.

канал - канал указывает запятую разделившую список всех каналов, чтобы сканироваться. все Комплекты MRF24WB0M, чтобы сканировать все каналы в данную полномочную область.

полномочие
восстановите Приспосабливать все полномочие, сохраняющее характеристики (PS_POLL) MRF24WB0M. выводить из строя Выводить из строя любые силовые характеристики расходов. MRF24WB0M Всегда будет в активном силовом состоянии.

одноадресная MRF24WB0M будет в это самое глубокое состояние сна , только проснувшись через определенные промежутки времени , чтобы проверить для однонаправленных данных . MRF24WB0M не проснется от периода DTIM для вещания или многоадресной трафика . все MRF24WB0M проснется для проверки всех типов трафика ( одноадресных , многоадресных и широковещательных ) .

**домен**

каналы Соединенных Штатов фед.ком.по эл-связи 1-11.
ic Канадские каналы 1-11.
etsi Европейские каналы 1-13.
испания Испанских каналов 10-11.
франция Французских каналов 10-13.
japana Японский канал 14.
japanb Японские каналы 1-11.

rts длина Устанавливает запрошенное количество байтов, чтобы посылать. Умолчание max - 2347.

**txrate**

0 MRF24WB0M сделает автоматическую адаптацию скорости от 1 до 2 Mbps.

1 Устанавливает MRF24WB0M к фиксированной скоростью передачи данных 1 Мбит .

2 Устанавливает MRF24WB0M к фиксированной скоростью передачи данных 2 Мбит .

**сканирование**

Указывает MRF24WB0M выполнять активное сканирование сайта. Результаты сканирования будут отображаться в выходной терминал.

**зимовать**

Удаляет питание MRF24WB0M .

**пробуждения**

Восстанавливает питание MRF24WB0M и восстанавливает соединение .

**Примечание**: сканирование поддерживается только демо WiFi EZConfig .

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

### 7.2.5.3 ifconfig Команды

Обратите внимание, что эти предметы не должны быть изменены , когда устройство находится в подключенном состоянии к сети .

команды Ifconfig принять следующую структуру:

ifconfig [ <IP address> ]
[ <MAC address> ]
[ netmask <IP address> ]
[ gateway <IP address> ]
[ auto-dhcp <start|drop> ]

Примечание **Ifconfig** само по себе даст состояние сети.

**IP адрес**

Используйте статический IP -адрес. IP -адрес должен быть в дот- десятичной системе счисления .

Обратите внимание, что эта команда вернет неверный параметр, если клиент DHCP включен. Впервые отключить попытки DHCP ( **Ifconfig** падение авто - DHCP) до запуска этой команды .

**MAC-адрес**

Пересмотрите устройства MAC -адрес. MAC адрес должен быть указан в шестнадцатеричном формате.

Эта команда может быть выдана только когда **MRF24WB0M** находится в режиме ожидания . Это на друга раз может привести к неожиданным результатам .

**маска**

IP адрес Используйте только указанный IP -адрес для маски сети. Значение маски задается в десятичной системе.

**шлюз**

IP адрес Настройка адрес шлюза . Значение шлюз, указанный в десятичной системе счисления.

**авто - DHCP**

начало запуске клиента DHCP .

Действительно только если модуль DHCP был составлен дюйма DHCP клиента запускается по умолчанию .

Падение Останавливает клиента DHCP . Статический IP- адрес нужно будет назначить устройству.

Действительно только если модуль DHCP был составлен дюйма

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

### 7.2.5.4 iwpriv Команды

Обратите внимание, что эти предметы не должны быть изменены , когда устройство находится в подключенном состоянии к сети .

**iwpriv** команды принимают следующую структуру:

iwpriv [ enc <none|wep|wpa-psk|wpa- фраза> ] [ клавиша <[1]|[2]|[3]|[4]> <оценивает> ] [ psk <value> ] [ phrase <величину> ]

Примечание iwpriv само по себе будет отображать параметры безопасности сети .

**enc**
никакое **MRF24WB0M** не не использует любое шифрование, чтобы соединять ( смотри страницу 171) в определенную сеть.

**wep**

MRF24WB0M использует или WEP-40 (перемычка) или WEP-104 шифрования (долго (длиной)), чтобы соединять ( смотри страницу 171) в определенную сеть.

**wpa-**
psk MRF24WB0M использует определенный 32- байт PSK, чтобы соединять ( смотри страницу 171) в сеть

**WPA/WPA2.**
MRF24WB0M возьмет данные 1-32 символа ASCII passphrase, вместе с SSID и вычислять необходимый 32- байт PSK для сети.

Обратите внимание, что сделать это занимает около 30 секунд, чтобы завершить расчет .

**ключ**

[1] [2]

[3] [4]

Поручает **MRF24WB0M** использовать этот ключ для подключения к WEP шифруются сети.

Обратите внимание, что единственный ключ 1 считается безопасным для использования среди различных поставщиков AP . Ключи 2-4 может иметь реализации конкретных записей, которые не могут быть совместимы между точками доступа, .

**значение** Если указано значение , это будет поручить MRF24WB0M использовать указанный номер ключа , а также программировать устройство с этим ключевое значение . Для WEP- 40 сетей , это означает, ключевым является либо 5 символов ASCII символов 10 шестнадцатеричных символов . Для WEP- 104 сетей , это означает, ключ либо 13 символов ASCII или 26 шестнадцатеричных символов .

Консоль принимает только ключи WEP шестнадцатеричные. Таким образом, пользователь должен сделать ASCII проклясть преобразования для их ключей ASCII.

**PSK**

ценим значение шестигранную 32 байт для PSK.

Это значение можно рассчитать по следующей сайт, размещение на веб-сайте **Wireshark** .

**фраза**

ценим 8-63 ASCII символов фразу ( с разделителями с кавычками , если использовать пробелы ) . Эта фраза будет использоваться вместе с SSID , чтобы генерировать значение PSK 32 байт для сети.

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

**7.2.5.5 iperf Примеры**

**iperf** - сетевое средство, которое помогает измерять связь ширины полосы частот и исполнение. Консольное демонстрационное приложение имеет встроенное **iperf** приложение, которое может подействовать как клиент так и как сервер для теста. **iperf** имеют способность тестировать как UDP так и TCP. В случае UDP, Вы можете определить размер дейтаграмм UDP. Для TCP, **iperf** измеряет производительность полезной нагрузки.

Для того, чтобы запускать **iperf**, Вам нужно PC, что имеет приложение **iperf** на этом также. Есть открытая исходная версия, которая поддержана, а также много другие варианты через интернет. **iperf** захотеть быть прогоном в командной строке. Тем не менее, если ГРАФИЧЕСКИЙ интерфейс пользователя желательный, вариант был вызван **jperf** может быть использовано.

В случае демонстрационного приложения, **iperf** данные исполнения мер в однонаправленном формате. Следовательно, сторона, что сервер работает на считается получателем и обеспечен наиболее точные величины исполнения.

**Командный Краткий обзор**

iperf

[ -s|-c <IP addr> ] [ -u ] [ -я <sec> ] [ -b <ширина полосы частот> ] [ -t <time> ]

-s Выполняет сервер iperf. Никакой адрес IP не должен быть определен.

-c <IP addr> Прогоны клиент iperf. Адрес IP является адресом IP сервера.

u Сервера только. Посылает дейтаграммы UDP.

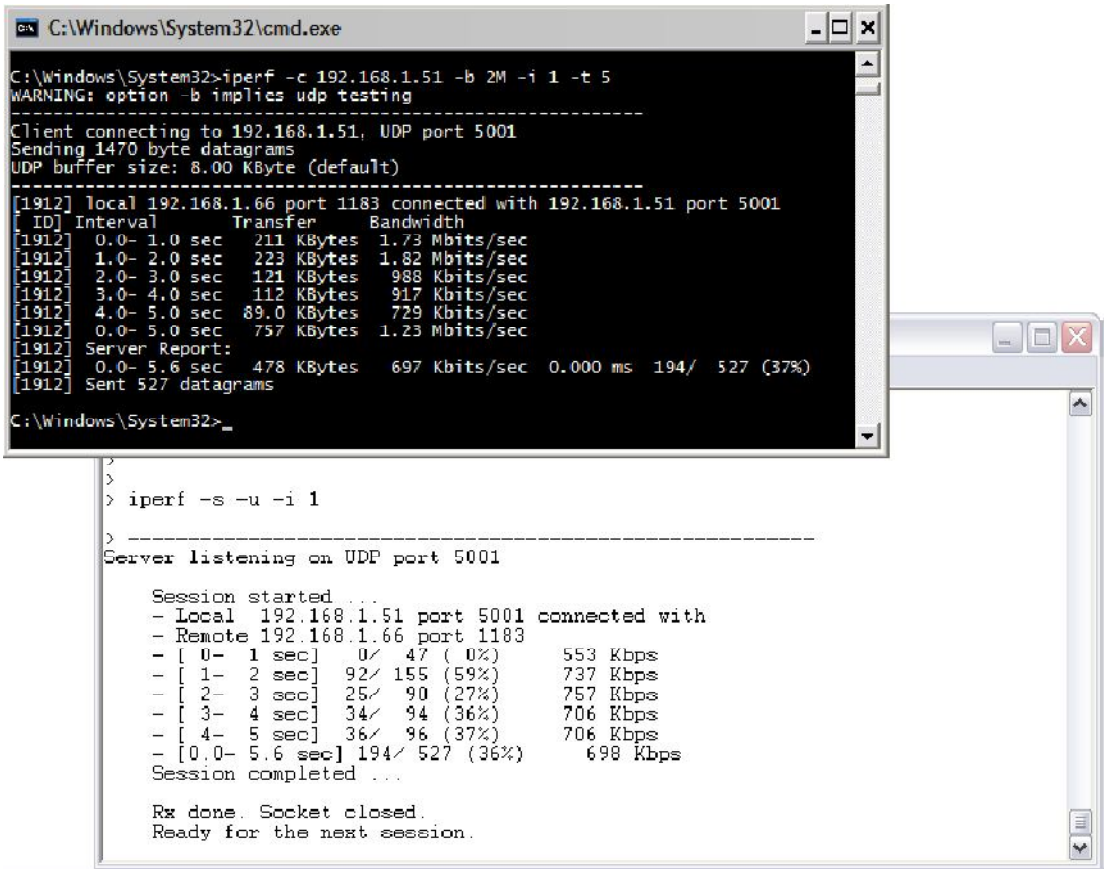
-я <sec> Определивший интервал времени, в течение секунд, что дисплей будет скорректирован.

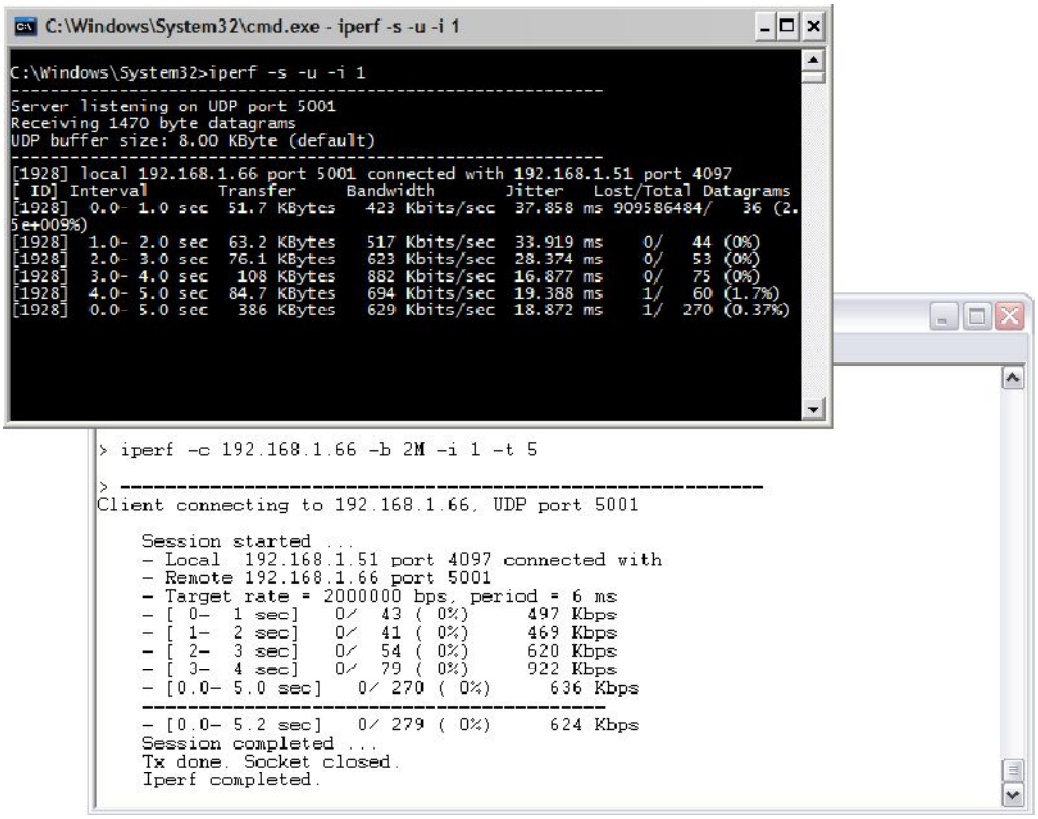
-b <bandwidth> Определяет сумму данных, чтобы пытаться и посылать. Эта опция только правильная с дейтаграммами UDP.

-t <time> Время, в течение секунд, чтобы запускать тест.

**Прогон Демонстрационного**

После включать платы разработки и коллеги с вашей беспроволочной сетью, вам понадобится запускать сторону сервера **iperf** приложение first. если Вы запускаете **iperf** как сервер на плате разработки на консоли, тогда это подразумевает, который Вы пытаетесь измерять исполнение получателя **MRF24WB0M**. Если Вы запускаете сервер **iperf** на PC, тогда Вы измерите исполнение передачи **MRF24WB0M**. Ниже - два образа, которые показывают получателя и исполнения передатчика, соответственно.


C:\Windows\System32\cmd.exe
C:\Windows\System32>iperf -c 192.168.1.51 -b 2M -i 1 -t 5
WARNING: option -b implies udp testing
Client connecting to 192.168.1.51, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
[1912] local 192.168.1.66 port 1183 connected with 192.168.1.51 port 5001
[ ID] Interval Transfer Bandwidth
[1912] 0.0- 1.0 sec 211 KBytes 1.73 Mbits/sec
[1912] 1.0- 2.0 sec 223 KBytes 1.82 Mbits/sec
[1912] 2.0- 3.0 sec 121 KBytes 988 Kbits/sec
[1912] 3.0- 4.0 sec 112 KBytes 917 Kbits/sec
[1912] 4.0- 5.0 sec 89.0 KBytes 729 Kbits/sec
[1912] 0.0- 5.0 sec 757 KBytes 1.23 Mbits/sec
[1912] Server Report:
[1912] 0.0- 5.6 sec 478 KBytes 697 Kbits/sec 0.000 ms 194/ 527 (37%)
[1912] Sent 527 datagrams
C:\Windows\System32>_
> iperf -s -u -i 1
Server listening on UDP port 5001
Session started ...
- Local 192.168.1.51 port 5001 connected with
- Remote 192.168.1.66 port 1183
- [ 0- 1 sec] 0/ 47 ( 0%) 553 Kbps
- [ 1- 2 sec] 92/ 155 (59%) 737 Kbps
- [ 2- 3 sec] 25/ 90 (27%) 757 Kbps
- [ 3- 4 sec] 34/ 94 (36%) 706 Kbps
- [ 4- 5 sec] 36/ 96 (37%) 706 Kbps
- [0.0- 5.6 sec] 194/ 527 (36%) 698 Kbps
Session completed ...
Rx done. Socket closed.
Ready for the next session.


C:\Windows\System32\cmd.exe - iperf -s -u -i 1
C:\Windows\System32>iperf -s -u -i 1
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 8.00 KByte (default)
[1928] local 192.168.1.66 port 5001 connected with 192.168.1.51 port 4097
[ ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[1928] 0.0- 1.0 sec 51.7 KBytes 423 Kbits/sec 37.858 ms 909586484/ 36 (2.5e+009%)
[1928] 1.0- 2.0 sec 63.2 KBytes 517 Kbits/sec 33.919 ms 0/ 44 (0%)
[1928] 2.0- 3.0 sec 76.1 KBytes 623 Kbits/sec 28.374 ms 0/ 53 (0%)
[1928] 3.0- 4.0 sec 108 KBytes 882 Kbits/sec 16.877 ms 0/ 75 (0%)
[1928] 4.0- 5.0 sec 84.7 KBytes 694 Kbits/sec 19.388 ms 1/ 60 (1.7%)
[1928] 0.0- 5.0 sec 386 KBytes 629 Kbits/sec 18.872 ms 1/ 270 (0.37%)
> iperf -c 192.168.1.66 -b 2M -i 1 -t 5
Client connecting to 192.168.1.66, UDP port 5001
Session started ...
- Local 192.168.1.51 port 4097 connected with
- Remote 192.168.1.66 port 5001
- Target rate = 2000000 bps, period = 6 ms
- [ 0- 1 sec] 0/ 43 ( 0%) 497 Kbps
- [ 1- 2 sec] 0/ 41 ( 0%) 469 Kbps
- [ 2- 3 sec] 0/ 54 ( 0%) 620 Kbps
- [ 3- 4 sec] 0/ 79 ( 0%) 922 Kbps
- [0.0- 5.0 sec] 0/ 270 ( 0%) 636 Kbps
- [0.0- 5.2 sec] 0/ 279 ( 0%) 624 Kbps
Session completed ...
Tx done. Socket closed.
Iperf completed.

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

### 7.2.6 WiFi EZConfig

**обзор**

WLAN -сети обеспечивают уникальный вызов для настройки встроенных беспроводных без естественного пользовательского интерфейса. В отличие от проводной сети , беспроводные сети требуют уникальные предметы , такие как SSID и тип сети и ключи , которые должны быть направлены Устройство в той или иной форме . Традиционно, это означает, что пользователь будет вводить эту информацию с помощью клавиатуры и дисплея.

**EasyConfig** представляет собой механизм для обеспечения конфигурации встроенного устройства в беспроводной сети. Он использует Интернет сетевой сервер стека протокола TCP / IP , а такжебеспроводной АПЧРК ( IBSS) , чтобы позволить пользователю ввести желаемое сеть Информация с браузером клиента , а затем перезагрузить устройство для подключения (см. стр. 171 ) до нужной сети.

**EasyConfig** демо работает в примерно следующим образом :

1 . После включения питания устройство, оно вещает АПЧРК сеть с SSID " EasyConfig " .

2 .Клиентское устройство (ноутбук, Ipod Touch / iPhone / IPad ) могут подключаться (см. стр. 171 ) к сети EasyConfig .

3 . После подключения , пользователь может затем использовать стандартный веб-браузер , чтобы перейти на IP- адрес демо ( http://169.254.1.1 ) .

4 . Пользователь затем будут представлены с некоторыми веб-страниц с веб-сервера . На веб-странице index.htm имеет некоторые дополнительные информация о EasyConfig , а также показывает постоянно обновляет статус светодиодов , кнопок и потенциометра на Совет по развитию . На странице configure.htm позволит пользователю для поиска сетей и подключения (см. стр. 171) , чтобы сеть их выбору.

5 . Устройство затем сбросить себя , используя параметры для новой сети. Для того , чтобы продолжить использование демо клиента Устройство теперь нужно будет подключиться к той же сети , что Совет по развитию находится на.

Обратите внимание, что демо будет всегда пытаться подключиться (см. стр. 171) по последнему известному сети. Если пользователь хочет, чтобы сбросить демо к запуску в специальном режиме снова, то кнопка S3 на борту развития Explorer 16 должен быть нажата в течение 4 секунд.

**Adhoc сети**

После запуска демо , сеть будет либо подключить (см. стр. 171) на другой AdHoc сети , или начнет свой собственный, если один не найден . Adhoc сети равный- одноранговых сетей, без централизованного координатора сети. Все устройства в сети разделять ответственность по поддержанию сети собирается.

Один падение AdHoc сетей является то, что, как правило, безопасность не работают на них. Модуль **MRF24WB0M** может обеспечить сети с WEP ( 40-bit/104-bit ) безопасности , так же

как большинство ноутбуков и Adhoc устройства . Почти нет устройств на рынке может обеспечить себе АПЧРК сеть с безопасностью уровня WPA из-за огромной головой в этом .

Демо начинает АПЧРК сеть без защиты . Это означает, что вся информация в сети , что не будет настроена на устройство будет более- воздух под открытым небом. Для большинства приложений , если кто-то специально пытается подслушать в этой сети , должна быть не оказывают существенного влияния на безопасность. Тем не менее, для приложений, которые требуют некоторого базового уровня безопасности , то WEP может быть использован в сети. SSL также могут быть использованы для шифрования трафика между веб-сервером и браузер клиента . Кроме того, некоторые другие формы безопасности данных на уровне могут быть использованы , чтобы скрыть сеть ASCII Информация доводится до устройства.

**Параметры сети**

Ниже приводится некоторая информация о параметрах , которые в настоящее время отправленных через HTTP POST с клиентского браузера к устройству. все эта информация будет анализироваться и обрабатываться в функции **HTTPPostWifiConfig** ( ) в **CustomHTTPApp**.c .

WLAN Тип Либо АПЧРК или инфраструктуры

SSID Название сети ( 1-32 символов ASCII )

**Тип безопасности**

• Ни

• WEP -40 ( 5 символов ASCII или 10 шестнадцатеричных чисел )

• WEP -104 ( 13 символов ASCII или 26 шестнадцатеричных чисел )

• WPA/WPA2 фраза ( 8-63 символов ASCII )

**Настроен против ненастроенным государства**

Когда демо работает в ненастроенной состоянии (т.е. , служа по умолчанию EasyConfig SSID в специальном режиме ), то сердцебиение светодиод ( LED0 ) будет мигать два раза в секунду , чтобы показать , что она не настроена. После того, как сеть была сконфигурирован, то сердцебиение SSID изменит мигать один раз в секунду , аналогичным образом на другой TCP / IP демонстрации приложений.

**EasyConfig Демо Дополнительные возможности**

Есть четыре определяет , позволяющие **EasyConfig** а также расширить его с природными особенностями .

***STACK_USE_EZ_CONFIG*** высшем уровне определить для включения функций **EasyConfig**.

***EZ_CONFIG_SCAN*** Добавляет дополнительную способность инструктировать модуль MRF24WB0M для поиска доступных сетей поблизости. Это может быть сделано , когда вы уже подключены к сети.

***EZ_CONFIG_STORE*** магазин информация о конфигурации для новой сети в энергонезависимой памяти.

в событие, которое уровень безопасности WPA/WPA2 используется ,PSK 32- байт будет сохранен в NVM .

***EZ_CONFIG_STALL*** Перед включением сетей, заставляет конфигурации государственную машину, чтобы приостановить. это дает клиентское устройство дополнительное время требовать ресурсы от развития Платформа , прежде чем он пытается подключиться (см. стр. 171 ) к новой сети .

**EZ_CONFIG_SCAN** MRF24WB0M имеет возможность сканировать соседнии сети. Это похоже на ноутбук, который может показать доступные беспроводные сети, к которым можно подключится. Результаты сканирования сохраняются в модуле MRF24WB0M , и могут быть получены по одному время от устройства. Это помогает уменьшить воздействие хранения всех результатов сканирования на рабочей машине .

Сканирование может выполняться в режиме ожидания или при подключении к AdHoc или инфраструктурной сети .

**EZ_CONFIG_STORE** Новые сетевые параметры также могут быть сохранены в энергонезависимой памяти . Для демо платы Explorer 16 , это информация хранится в 32kB памяти EEPROM на плате .

Одна чрезвычайно полезная функция хранения информации поверхностей при подключении к сети с WPA/WPA2 безопасности.

Вычисление PSK 32- байт довольно тяжелое, и может занимать MRF24WB0M до 30 секунд , чтобы вычислить ключ . В обычном приложении , было бы неприемлемым ждать 30 секунд каждый раз, когда устройство запускается перед созданаием подключения к сети .

***EZ_CONFIG_STORE*** помогает облегчить делать расчеты каждый раз, сохраняя PSK 32-байтовую NVM .

При этом есть только один 30-секундный ударил в самый первый раз ключ только рассчитывается . Последовательные подключения к сети будет значительно быстрее .

**EZ_CONFIG_STALL** Государственная машина конфигурация , которая контролирует сетевые соединения внутри EasyConfig может использовать состояние ожидания между переключения сетей. Из опыта конечного пользователя , это становится жизненно важным . Если переключатель между различными сетями было мгновенно, клиентский браузер никогда не получили бы указание, что HTTP сессия была закрыта после информации POST был отправлен . Конечный пользователь будет видеть это как браузер, который был постоянно в ожидании , что в конечном итоге тайм-аут .

Чтобы на переключатель еще естественной и полной , EZ_CONFIG_STALL добавляет дополнительное время , чтобы позволить клиенту получить Остальная информация веб-страницы . Для демонстрации, это включает в себя HTTP перенаправления на страницу , которая выделяет новую сеть информация .

**Текущие Несовместимость**

JavaScript используется в EasyConfig не совместим с Internet Explorer 7. EasyConfig делает работу со многими другими ароматы браузера на различных архитектурах , не ограничиваясь Internet Explorer 8 , Mozilla Firefox , Apple Safari и Google Хром . Несовместимость является то, что находится в стадии расследования , и должны быть установлены в будущем выпуске стека.

### 7.2.7 Демо App MDD

TCPIP MDD Демо приложение является вариант TCPIP Demo App который использует SD - карту или USB флэш-накопитель для хранения веб-страниц .

Для получения дополнительной информации см.TCPIP MDD Демо App Руководство по началу работы , установлены в папке с документацией в стека .

### 7.2.8 Google PowerMeter

Google, Inc имеет устарела Google PowerMeter и выразил свое намерение удалить доступ к нему 16 сентября 2011 года.

Из-за этого развития , Microchip Technology были удалены PowerMeter демо проекты Google от Microchip Прикладные библиотеки распределение . Для получения справочной PowerMeter Google реализацию Microchip , пожалуйста, скачать установка архивные Microchip Прикладные библиотеки с июня 2011 года с www.microchip.com / OMC.

### 7.2.9 Энергия Мониторинг

Это демо реализует приложения для мониторинга сила, которая добавления данных в Google PowerMeter . В этом применении , фактический Потребление электроэнергии получается из мониторинга PIC18F87J72 энергии PICtail Плюс дочерней плате.

Google, Inc имеет устарела Google PowerMeter и выразил свое намерение удалить доступ к нему 16 сентября 2011 года. Из-за этого развития , Microchip Technology были удалены PowerMeter демо проекты Google от Microchip Прикладные библиотеки распределение . Для получения справочной PowerMeter Google реализацию Microchip , пожалуйста, скачать установка архивные Microchip Прикладные библиотеки с июня 2011 года с www.microchip.com / OMC.

Эта энергия Мониторинг демо был изменен, чтобы вывести свои Google PowerMeter функции загрузки. Однако энергия Измерения , сделанные PICtail еще можно посмотреть на демо- платы из организаторов веб-странице.

ПК графический интерфейс для PICtail энергии мониторинга можно получить по адресу:

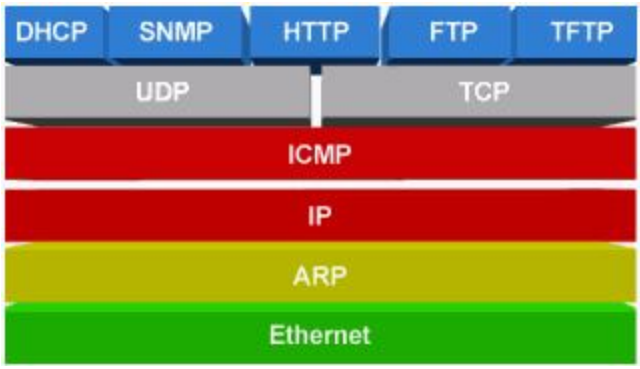
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en550456

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

# 8 Использование стека

В этом разделе описывается, как использовать компании Microchip TCP / IP стека .

## 8.1 Архитектуры Стека




Стек TCP / IP имеет модульную структуру и написан на языке 'C'. Стек в настоящее время поддерживает модули TCP и UDP транспортного уровня, в IPv4 (и часть ICMP ) Интернет слоя модули , модули ARP канального уровня, а также ряд модулей прикладного уровня . Большая часть канального уровня управления доступом к среде функциональность обеспечивается аппаратным MAC / PHY фишек ( см. стр. 62 ), используемых со стеком .

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

## 8.2 Как Стек работает

Этот раздел содержит сведения о том, как работает стек , что требуется чтобы использовать стек, и, как ваш код может быть построена так, чтобы слаженно работать со стеком TCP / IP.

### 8.2.1 Требуемые файлы

Есть несколько базовых файлы, которые должны быть включены в каждом проекте с использованием TCP / IP стек Microchip. К ним относятся:

•Главный файл (см. стр. 138) - это файл с кодом приложения в нем ..

• ARP.c и ARP.h - Эти файлы используются в стеке , чтобы обнаружить MAC-адрес , связанный с данным IP -адрес.

• **Delay**.c и **delay**.h - Эти файлы используются для обеспечения задержки для некоторых функций стека. Обратите внимание, что было бы лучше , чтобы Не используйте эти задержки в собственном коде , так как они создают блокирующие условия .

• Файлы физического уровня - Эти файлы используются для того, чтобы задействовать указанный физический уровень . Более подробную информацию о том, какие файлы включить, можно найти в разделе аппаратной конфигурации (см. стр. 142).

• **Helpers**.c и **Helpers**.h - Эти файлы содержат вспомогательные функции , используемые для разных задач стека.

• **IP**.c и **IP**.h - Эти файлы обеспечивают функциональность интернет слоя для стека .

• **StackTsk**.c и **StackTsk**.h - Эти файлы содержат код для инициализации стека и выполняют обратные вызовы , которые держат стек собирается .

• **Tick**.c и **Tick**.h - Эти файлы реализуют таймер, который используется для реализации некоторых функций времени в рамках стека .

• **HardwareProfile**.h - Этот конфигурационный файл используется для настройки параметры оборудования.

• **TCPIPConfig**.h - Этот конфигурационный файл используется для настройки параметров прошивки.

• **MAC**.h - Этот файл заголовка предоставляет макросы и структуры , связанные с аппаратным MAC слоем.

• **TCPIP**.h - Это основной файл заголовка для стека.

Вы можете выбрать , чтобы включить дополнительные файлы для поддержки дополнительных протоколов и функций. Список протоколов и их необходимые файлы можно найти в Макросы Протоколов и файлов (см. стр. 150) тему в настройках протокола (см. стр. 149) темы .

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

### 8.2.2 APP_CONFIG Структура

Большинство переменных приложения стека, связанных хранятся в структуре APP_CONFIG. К ним относятся адреса (см.стр. 147) , флаги, и струны NBNS / SSID имя . Вам нужно будет объявить один из этих структур ( под названием " **AppConfig** " ) для ваше приложение и инициализировать его со значениями по умолчанию , определенных в **TCPIPConfig**.h . Например, вы должны установить байты поле **MyIPAddr** для значений MY_DEFAULT_IP_ADDR_BYTE макросов в **TCPIPConfig**.h .

**InitAppConfig** Функция в файле MainDemo.c проекта TCPIP Demo App демонстрируется полностью заполнить эту структуру .

Полный список параметров в структуре APP_CONFIG определяется в файле StackTsk.h .

В начале большинстве демонстрационных стек приложений , код будет проверить ЭСППЗУ , чтобы определить , если она содержит действительный Образ структуры APP_CONFIG . Если это так, он будет читать изображение и использовать его для заполнения экземпляр

**AppConfig** в демо Проект . В противном случае , она будет загружать переменные приложения из ваших статически определенных значений и / или настроить их на основе прикладные протоколы ( DHCP / AutoIP ). Это позволяет доска сохранить свои настроенные параметры , даже если приложение теряет власть.

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

### 8.2.3 Основное файла

Потому что есть огромное разнообразие способов, которыми вы могли написать свое приложение , в этом разделе будет предоставлять схему того, что ваш главный файл должен содержать . Это также обеспечивает некоторую описание работы стека , и лучшие практики программирования методы по предотвращению проблем стека .

#### 8.2.3.1 Инициализация

Вы должны начать с инициализации аппаратного обеспечения . Это включает в себя PPS флажки , осцилляторы , индикаторы , ЖК-дисплеи , любой SPI или PMP модули вы используете для управления вашего оборудования MAC / PHY чип и т.д.

Затем вызвать функции аппаратной инициализации для библиотеки. **TickInit** (см. стр. 604 ) должен быть вызван в первую очередь; он будет инициализировать таймер, который управляет вашим временем стека. Тогда звоните никаких дополнительных функций инициализации, которые требуют оборудования инициализации. Например, **MPFSInit** (см. стр. 284 ) функция  нужно будет инициализировать порт SPI для связи с запоминающее (см. стр. 142 ) устройство для хранения веб- страниц, так что следует называть теперь.

После того, как ваше оборудование инициализации, вы можете начать настройку свой стек. Большинство переменных приложения стека, связанных являются хранится в **AppConfig** (см. стр. 138) структуру. На данный момент, вы должны инициализировать структуру **AppConfig** с вашим значения по умолчанию, или предоставить другое средство инициализации структуры **AppConfig** .

Наконец, вы можете инициализировать стек, вызвав функцию **StackInit** ( ). Эта функция автоматически вызовет инициализацию  функции для других протоколов прошивки, если они были включены в **TCPIPConfig**.h (т.е. **TCPInit** (см. стр. 554) ( ) для Протокол TCP , **HTTPInit** (см. стр. 250 ) ( ) для http2 , ... ). После **StackInit** ( ) был вызван, вы можете позвонить другу прикладных функций инициализации прошивки .

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

#### 8.2.3.2 Главный цикл

После того, как ваша программа была инициализирована, вы должны войти в бесконечный цикл, который будет обрабатывать ваши прикладных задач. В рамках этой цикл, есть две функции, которые необходимо вызвать регулярно: **StackTask** и **StackApplications**.

Функция **StackTask** будет выполнять любые приуроченые операции , какие стеку требуется, и будет обрабатывать передачу и прием пакетов данных. Эта функция также будет маршрутизировать любые пакеты, которые были получены в соответствующее приложение Функция на уровне протокола с ним обращаться.

Функция **StackApplications** вызовет загруженные модули приложения . Например, если приложение с использованием http2 сервер , **StackApplications** автоматически вызывает **HTTPServer** (см. стр. 251 )и будет обрабатывать любые Http2 задачи, которые ставятся в очередь .

Большинство подзадач в рамках **StackTask** и **StackApplications** реализованы как государство -машина управляется кооператива многозадачность функции. Так как эти подзадачи состоит из нескольких этапов ( которые могут возникнуть в разное время ) Такой отзыв Система гарантирует, что ни одна задача не будет монополизировать управление процессора .

В рамках этого основного цикла , вы также можете опрашивать любые изменения ввода / вывода в вашем приложении и вызвать любые приложения конкретных задач, которые вы реализовали . Чтобы не вызывать переполнение буфера в вашем оборудовании или нарушения синхронизации протокола вы должны попытаться реализовать собственные прикладные задачи в функции обратного вызова с временными на основе триггеров. Способ как это реализовать описывается в соседней теме.

Вы должны сделать один вызов в **StackTask** для каждого вызова **StackApplications** но вы не обязаны вызывать эти функции с какой-либо конкретной частотой. Вызов **StackTask** слишком редко может ограничивать пропускную способность, хотя, как каждый вызов **StackTask** можете получить один пакет (самое большее) из пакетного буфера. Аналогичным образом, прикладные задачи, которые зависят от времени ( как ответ пинг ICMP ) может привести к нежелательным результатам, если **StackApplications** не вызывается достаточно часто.

Количество времени на основной цикл , необходимое для завершения одной итерации зависит от нескольких факторов. Если данные готовы быть передается, или если был получен пакет данных, функция **StackTask** займет больше времени, чем это было бы в противном случае. Каждый дополнительный протокол включен в вашем приложении вызовет основной цикл, чтобы взять дополнительное время, а количество времени для каждого варьируется от длины самой короткой автоматной государства в задачи , чтобы самая длинная .

Как только ваша заявка будет завершена, вы можете настроить тестовый пример для определения мин / средний / приблизительное максимальное время, которое ваш цикл займет. Вы можете настроить свой код, чтобы использовать внутренний таймер для измерения длительности каждой итерации Основного цикла, или вы можете установить код до запуска выходной вывод каждый раз когда основной цикл завершается, и использовать осциллограф для захвата времяни выполнения в сети. Вы можете внести вклад приложений или дополнительного сетевого трафика с помощью программы PC ( и другие картинки ), чтобы имитировать реальные условия эксплуатации.

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

### 8.2.4 кооперативную многозадачность

При реализации стека TCP / IP с использованием кооперативную многозадачность подход , необходимо проводить периодические звонки к ответу функции передавать / принимать пакеты и поддерживать функциональные возможности протокола. Для предотвращения конфликтов с стека , вы должны написать ваш собственные пользовательские задачи таким образом, что позволит им отказаться от процессора , если он не нужен . При создании протокола или Приложение задача с несколькими шагами , это может быть выгодно , чтобы разделить их между государствами. Затем вы можете использовать глобальные или Статическая переменная для отслеживания состояния и периодически вызывать эту функцию задачи для перемещения по государственной машины.

Следующий пример содержит пример приложения для передачи данных от машины некоторого типа с внешним цели.

Она включает в себя функцию задач под названием **ApplicationTask** который имеет состояний ждать входов кнопки , обновить экран , и передавать данные из устройства. Функции в примере используются для представления других действий : **ButtonPressDetected-** Представляет код нужен для проверки ввода от пользователя, **LCDdisplay-** представляет код, необходимый для обновления показаний на машине, **SampleData**- получает данные от машины, **DataBufferIsFull**- указывает, что буфер, используемый для провести выборки данных должен быть отправлен, и **TransferData-** это функция, которая записывает данные в открытом TCP или UDP сокетах.

В перерывах между каждым из этих государств, функция **ApplicationTask** возвращается в основной цикл, и **StackTask** и **StackApplications** функции вызываются. Этот поток позволит функция **StackApplications** поддерживать любой модуль задачи. Функция **StackTask** будет периодически передавать данные из буферов сокета к месту назначения, который позволит предотвратить буферов передачи от переполнения.

```
unsigned char gAppState; // // Состояние, прослеживающее переменную
int main (void)
{
/ / Функция Псевдо- инициализации

InitializeCode();

/ / Состояние приложения установки

gAppState = STATE_DISPLAY_MENU;

/ / Главный цикл

while (1)
{
StackTask();
StackApplications();
ApplicationTask();
```

```
}
}
void ApplicationTask (void)
{
switch (gAppState)
{
case STATE_DISPLAY_MENU:
LCDDisplay (stringMainMenu);
gAppState = STATE_MAIN_MENU;
break;
case STATE_MAIN_MENU:
if (ButtonPressDetected (BUTTON_1) ) / / Проверка вход
gAppState = STATE_MONITOR_MACHINERY;
break;
case STATE_MONITOR_MACHINERY:
LCDDisplay (stringTransferringData);
/ / Генерация или отправить данные

if (DataBufferIsFull())
{
TransferData();
}
else
{
SampleData();
}
if (ButtonPressDetected (BACK_BUTTON) )
gAppState = STATE_DISPLAY_MENU;
break;
}
}
```

Некоторые из государств в вашем приложении может быть время, на основе . Предположим, например, что наш пример приложения необходимо отправить Данные в течение 5 секунд каждый раз, когда вход обнаружено. Проблемы **Stack** может произойти, если приложение, используемое цикл задержки ждать 5 секунд, пока не пришло время остановиться, так эта функциональность должна быть реализована с использованием встроенного **Tick** таймера стека. Когда запрос на передачу данных получен, код будет получать текущее время Tick с помощью функции **TickGet** (см. стр. 603 ), добавить достаточно клещей наверстать 5 секунд, сохранить его в статической переменной называемой **tickCounter**, а затем переключиться в состояние передачи.

Каждый раз, когда функция **ApplicationTask** вызывается, он войдет в это состояние в государственной машине, вызовите снова **TickGet** (см.страница 603), а затем сравнить его со значением, содержащимся в этой статической переменной. Если текущее время позже, чем первоначальное время плюс задержка, код будет восстановить дисплей и вновь войти в основное состояние меню.

```
void ApplicationTask
(void) {
```

```
    static DWORD tickCounter;
    switch (gAppState) {
        case STATE_DISPLAY_MENU:
            LCDDisplay (stringMainMenu); gAppState =
        STATE_MAIN_MENU; break; case STATE_MAIN_MENU:
            if (ButtonPressDetected (BUTTON_1) ) / / Проверка вход
                gAppState =
            STATE_MONITOR_MACHINERY;
            break;
        case STATE_MONITOR_MACHINERY:
            LCDDisplay (stringTransferringData);
/ / Сохранить текущее время, и добавить 5 секунд к нему

            tickCounter = TickGet() + (5 * TICK SECOND);
              gAppState =
              STATE_CONTINUE_MONITORIN
              G; break;
          case STATE_CONTINUE_MONITORING:
               if ((long)(TickGet() - tickCounter) > 0) gAppState = STATE_DISPLAY_MENU;
                                                        else {
/ / Генерация или отправить данные

                    if
                    (DataBufferIsFull()
                    ) {
                        TransferData();
                    }
                                                            else {
                        SampleData();
                    }
                }
                break;
        }
  }
```

Есть три галочки сроки макросы объявлены помочь с задержками: TICK_SECOND ( см. стр. 602) определяет число тиков в секунду, TICK_MINUTE ( см. стр. 602) определяет количество тактов в минуту, и TICK_HOUR ( см. стр.602) определяет количество тактов в час. С помощью tick таймера для реализации задержки, вы можете гарантировать, что ваш код не будет блок критических функций слишком долго.

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

## 8.2.5 OCPB

В качестве альтернативы реализации приложения стека в кооперативной многозадачности формате , вы можете интегрировать стек в Real-Time Operating System . Для получения дополнительной информации см. Application Note 1264 на веб-сайте Microchip .

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

# 9 Настройка Stack

Существует широкий спектр опций конфигурации доступных для компании Microchip TCP / IP стека. В этой теме будут обсуждаться функциональность этих вариантов , и как их можно реализовать .

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

## 9.1 Конфигурация оборудования

Саму аппаратную конфигурацию выполняют, комментируя , удаляя или определяя серию макросов в одном из вариантов заголовка файла **HardwareProfile**.h. Вы можете увидеть примеры версии, как установить эти параметры в копии **HardwareProfile**.h, что включено в демо проект стека (см. стр. 82 ).

В большинстве случаев, макрос для включения устройства такой же, как макрос который используется для определения в микросхеме выбраных устройств. По умолчанию в копии **HardwareProfile**.h входящей в демо проект, есть примеры, определенные для большинства демо плат, ограниченные по препроцессору отчетности. Например, в разделе для Explorer 16 начинается с макро " # Elif определена ( EXPLORER_16 ) " и продолжается до следующего заявленного препроцессора на демо плате. Если вы используете один из этих файлов в качестве основы для вашего проекта, убедитесь, что вы изменяете макросы в правильном разделе.

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

### 9.1.1 тактовая частота

Многие операции TCP / IP зависят от времени. Указав частоту генератора, которую вы используете в вашем приложении для стека, вы можете включить автоматическую обработку этих операций. Для установки **Hte** значение времени, подставить свое частоту генератора ( в герцах ) для значения в следующем макроса в **HardwareProfile.h** :

```
#define GetSystemClock ( ) xxxxxxxxxxx
```

Есть также два других макросов часов. **GetInstructionClock()** и **GetPeripheralClock()** обеспечивают частоту значения для часов инструкций и периферических часов в вашем микроконтроллере. Эти значения, как правило, быть установлен в качестве фракции системные часы (т.е. **GetInstructionClock** ( ) будет определяет как ( **GetSystemClock** ( ) / 2) для PIC24F процессоры ).

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

### 9.1.2 Внешние накопители

Есть несколько особенностей, в стеке TCP / IP , которые используют внешние накопители для поддержания структуры или веб-страниц. Поддержка несколько устройств хранения

входит в стек; опорные файлы могут быть использованы в качестве шаблона писать драйверы для других  устройств, а также. Определения контактный **HardwareProfile**.h примерно эквивалентны для каждого устройства хранения, для первого слова, кроме макроса, который указывает, какой тип запоминающего устройства применительно к (например EEPROM_CS_IO против SPIFLASH_CS_IO ).

Там три различных носителей информации .

**EEPROM**

**EEPROM** может быть использован для хранения **MPFS2** (см. стр. 271 ) веб-страницы изображения и пользовательской структуры приложения. Чтобы указать, в стеке , что он должен использовать **EEPROM** для хранения **MPFS2** изображения, определите макрос **MPFS_USE_EEPROM** в  **TCPIPConfig.h** файл заголовка . По умолчанию стек включает в себя драйвер для 25LC256 EEPROM семьи компании Microchip (использовать 1 Мбит **EEPROM**, необходимо также определить макрос **USE_EEPROM_25LC1024** в **TCPIPConfig.h** ). Макросы для управления связью с **EEPROM** будет предваряться строкой **EEPROM_** в этом случае. Для установления связи, определить **EEPROM_CS_TRIS** и включают в себя **SPIEEPROM**.c файлов и **XEEPROM**.h в вашем приложении. Эти файлы могут потребовать некоторых изменений для поддержки дополнительных **EEPROM** устройств.

**Серийный флэш-памяти**

Для хранения MPFS изображений и пользовательских структур также доступна на серийных устройств флэш-памяти ( протестированы с SST 25VF016B и Spansion 25FL040A ). Чтобы указать, что стек должен использовать последовательную флэш-память для хранения веб-страниц , определить **MPFS_USE_SPI_FLASH** в **TCPIPConfig**.h . В communicaiton макросы будут предваряться строкой **SPIFLASH_** в этом случае. Чтобы включить функции связи, определить **SPIFLASH_CS_TRIS** и включите файлы **SPIFlash.c** и **SPIFlash.h** в ваше приложение. Эти файлы могут потребовать внесения некоторых изменений для поддержки дополнительных карт памяти. Есть несколько макросов, включенные в " **SPIFlash.h** ", которые также должны быть определены, в том числе макросы для определения сектора и размера страниц и макросы для описания, используется ли Флэш устройство SST или Spansion .

**SRAM**

Последовательная оперативная память может быть использована для хранения FIFO блоков и TCP Блоков управления для розеток ( см. стр. 152 ) (проверено с AMT N256S0830HDA полупроводника). Макросы будет предваряться строкой **SPIRAM_** в этом случае. Чтобы использовать это функцию, определите **EEPROM_CS_TRIS** и включите в себя файлы " **SPIRAM.c** " и " **SPIRAM.h** " в приложении. Эти файлы могут потребовать внесения некоторых изменений для поддержки дополнительных RAM устройств .

**Макро Цель Пример значения**

**xxxxxx_CS_IO** Определяет LAT (или порт, где это применимо ) бита регистра, который соответствует выбора микросхемы контактный. Определение этого макроса указывают, что стек должен использовать указанный тип внешних накопителей.

LATDbits.LATD12

**xxxxxx_CS_TRIS** Определяет TRIS бит, который соответствует выбора микросхемы вывод.
TRISDbits.TRISD12

**xxxxxx_SCK_TRIS** Определяет TRIS бит, который соответствует тактовой штифта модуля SPI подключен к устройству.

TRISGbits.TRISG6

**xxxxxx_SDI_TRIS** Определяет TRIS бит, который соответствует данных в выводе модуля SPI подключен к устройству .

TRISGbits.TRISG7

**xxxxxx_SDO_TRIS** Определяет TRIS бит, который соответствует данным отъезда штифта модуля SPI подключен к устройству .

TRISGbits.TRISG8

**xxxxxx_SPI_IF** Указывает на флага прерывания для модуля SPI подключенного к устройству.

IFS2bits.SPI2IF

**xxxxxx_SSPBUF** указывает на буфер регистра SPI для модуля SPI подключенного к устройству .

SPI2BUF

**xxxxxx_SPICON1** очки в элемент управления SPI зарегистрируйтесь для модуля SPI подключенного к устройству .

SPI2CON1

**xxxxxx_SPICON1** Обеспечивает побитовое доступ к регистру управления SPI для модуля SPI подключен к устройству . В ____ бит регистры , обычно определяются в файлы заголовков процессора.

SPI2CON1bits

**xxxxxx_SPICON2** очки на второй регистре управления SPI для модуля SPI , подключенного к Устройство. Если ваше устройство не имеет SPICON2 регистр ( например PIC32 ) просто опустить это определение.

SPI2CON2

**xxxxxx_SPISTAT** очки в состояние SPI зарегистрируйтесь для модуля SPI подключенного к устройству .

SPI2STAT

**xxxxxx_SPISTATbits** Обеспечивает побитовое доступ к регистре состояния SPI для модуля SPI подключен к устройству .

SPI2STATbits

**xxxxxx_SPIBRG** Указывает на SPI Скорость передачи генератора зарегистрируйтесь для модуля SPI , подключенного к устройству. Если ваше устройство не имеет модуль SPI BRG основе , просто не пишите это определение .

SPI2BRG

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

### 9.1.3 ENC28J60 Config

Для использования ENC28J60 в вашем проекте , включают файлы ' ENC28J60.c "и" ENC28J60.h " в вашем проекте и раскомментируйте Следующий макрос в **HardwareProfile**.h:

#define ENC_CS_TRIS XXXXXXXXXXXXXXX

Несколько макросы должны быть сопоставлены с регистрирует или зарегистрируйтесь биты при использовании ENC28J60 . Они включают в себя :

**Макро Цель Пример значения**

**ENC_CS_IO** Определяет LAT (или порт, где это применимо ) бита регистра , который соответствует выбора микросхемы контактный . Определение этого макроса также показывают, что стек должен использовать ENC28J60 .

LATDbits.LATD14

**ENC_CS_TRIS** Определяет TRIS бит, который соответствует выбора микросхемы штифта. TRISDbits.TRISD14

**ENC_RST_IO** Определяет LAT (или порт, где это применимо ) бита регистра , который соответствует контакт сброса . Если вы оставите контактный сброса не связанных между собой в вашем дизайне , комментировать это макро вне.

LATDbits.LATD15

**ENC_RST_TRIS** Определяет TRIS бит , соответствующий штифт сброса. TRISDbits.TRISD15

**ENC_SPI_IF** Указывает на флага прерывания для модуля SPI, подключенного к микросхеме.

IFS0bits.SPI1IF

**ENC_SSPBUF** указывает на буфер SPI зарегистрируйтесь для модуля SPI, подключенного к микросхеме.

SPI1BUF

**ENC_SPISTAT** очки в состояние SPI зарегистрируйтесь для модуля SPI, подключенного к микросхеме.

SPI1STAT

**ENC_SPISTAT** Обеспечивает побитовое доступ к регистре состояния SPI для модуля SPI, подключенного на чипе. В ____ бит регистры, обычно определяются в заголовке процессора файлы .

SPI1STATbits

**ENC_SPICON1** очки в регистре управления SPI для модуля SPI, подключенного к микросхеме. SPI1CON1

**ENC_SPICON1** Обеспечивает побитовое доступ к регистру управления SPI для модуля SPI, подключенного с чипом (см. запись ENC_SPISTATbits ).

SPI1CON1bits

**ENC_SPICON2** очки на второй регистре управления SPI для модуля SPI, подключенного к чип. Если ваше устройство не имеет регистр SPICON2 (например PIC32 ) просто не пишите это определение.

SPI1CON2

**ENC_SPIBRG** Указывает на SPI Скорость передачи генератора зарегистрируйтесь для модуля SPI, подключенного к чип. Если ваше устройство не имеет модуль SPI BRG основе, просто не пишите это определение.

SPI1BRG

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

### 9.1.4 ENCX24J600 Config

Для использования ENC624J600 или- 424J600 в вашем проекте , включают " ENCX24J600.c " и " ENCX24J600.h " и раскомментируйте Следующий макрос в HardwareProfile.h :

#define ENC100_INTERFACE_MODE 0

Параметр '0 ' указывает, что вы будете использовать устройство в режиме SPI.
Потенциальные используемые параметры включают в себя :

Параметр Функциональность

0 Режим SPI с использованием CS , СХК, SI и так флажки

1 8 бит демультиплексируются PSP Режим 1 с РД и булавки WR

2 8-битный демультиплексируются PSP Режим 2 с R / ~ W и EN контакты

3 16-битный демультиплексируются PSP Режим 3 с РД , WRL и Wrh флажки

4 16-битный демультиплексируются PSP режима 4 с R / ~ W, B0SEL , и булавки B1SEL

5 8-битный мультиплексирован PSP режим 5 с двигателем РД и булавки WR

6 8-битный мультиплексирован PSP MODE 6 с R / ~ W и EN контакты

9 16-битный мультиплексирован PSP режим 9 с AL , RD, WRL и Wrh флажки

10 16-битный Мультиплексированные PSP Режим 10 с AL , R / ~ W, B0SEL , и булавки B1SEL

Более подробную информацию о функциональности каждого режима доступно в ENC624J600 семьи данные. Однако следует отметить, что 44 -контактный ENC424J600 только поддерживает связь в режиме SPI и PSP Режимы 5 и 6. Кроме того, из-за борту конфликты, PSP Режимы 2 , 4, 6, и 10 не должны использоваться с Explorer 16 ( и PSP Режим 3 может вызвать на шине с 25LC256 EEPROM ) .

Несколько макросы должны быть сопоставлены с регистрирует или зарегистрируйтесь биты при использовании ENCX24J600 также. Кроме того, некоторые возможности можно включить / отключить для этого устройства , определив отдельные макросы . Макросы включают в себя:

**Макро Цель Пример значения**

**ENC100_INTERFACE_MODE** Указывает, какой режим связи стек должен использовать для взаимодействия с чипом . Этот макрос также показывают, что стек должен использовать ENCX24J600 .

0

**ENC100_PSP_USE_INDIRECT_ - RAM_ADDRESSING**

Un - комментируя этот макрос позволит стек косвенно обратиться к оперативной памяти в ENCX24J600 ( сохранить некоторые адрес провода) . Для режима SPI или PSP режимов 9 и 10 , это параметр будет игнорироваться .

N / A

**ENC100_TRANSLATE_TO_PIN - _addr ( A)** Этот макрос будет на самом деле переназначить адреса , переданные в параллельный интерфейс , чтобы соответствовать конфигурации выводов (если вы с помощью косвенной адресации ) .

(((( а) и 0x100 ) << 6) | ( (и 0x00FF ) )

**ENC100_MDIX_IO** Если вы разработать Авто - кроссовер ( Auto-MDIX ) схему в ваше доска, этот макрос будет определятьконтактный использовать для этого. Смотреть Быстрый 100Mbps Ethernet PICtail / PICtail Плюс плата схематическое для примера схемы .

LATBbits.LATB3

**ENC100_MDIX_TRIS** Определяет немного TRIS использовать для цепи Авто - кроссовера. TRISBbits.TRISB3

**ENC100_INT_IO** Определяет I / O булавку, чтобы использовать для прерывания сигнала штифта чипа.

Эта функция в настоящее время не используется стеком .

PORTAbits.RA13

**ENC100_INT_TRIS** Определяет Трис немного, чтобы использовать для прерывания сигнала штифта чипа. TRISAbits.TRISA13

**ENC100_CS_IO** Определяет порт бит для использования с чипом выберите штифта. дополнительно на PSP режимы .

LATAbits.LATA5

**ENC100_CS_TRIS** Определяет Трис немного, чтобы использовать для выбора микросхемы штифта. TRISAbits.TRISA5

**ENC100_POR_IO** Определяет порт немного, чтобы использовать с обесточить цепь . если ваше приложение не имеет эта функция реализована , закомментируйте этот бит .

LATCbits.LATC1

**ENC100_POR_TRIS** Определяет немного TRIS использовать с обесточить цепь . TRISCbits.TRISC1

**ENC100_SO_WR_B0SEL_EN_IO** Определяет штифт используется для связи . Функциональность этот вывод зависит от того, какой режим связи в выбран.

Это может быть эквивалентно последовательному из штифта ENCX24J600 , Параллельный режим WR стробоскоп ,B0SEL контактный илиEN контактный .

LATDbits.LATD4

**ENC100_SO_WR_B0SEL_EN_TRIS** Определяет немного ТРИС для использования с

ENC100_SO_WR_B0SEL_EN_IO контактный .

TRISDbits.TRISD4

**ENC100_SI_RD_RW_IO** Определяет штифт используется для связи . Функциональность этот вывод зависит от того, какой режим связи в выбран.

Это может быть эквивалентно ENCX24J600 сериала в выводе , Параллельный режим RD строб, илиR / ~ W контактный .

LATDbits.LATD5

**ENC100_SI_RD_RW_TRIS** Определяет немного ТРИС использовать с ENC100_SI_RD_RW_IO контактный .

TRISDbits.TRISD5

**ENC100_SCK_AL_IO** Определяет штифт используется для связи . Функциональность этот вывод зависит от того, какой режим связи в выбран.

Это может быть эквивалентно ENCX24J600 серийный тактовой штифта или параллельный режим защелка адреса стробоскоп .

LATDbits.LATDB15

**ENC100_SCK_AL_TRIS** Определяет немного ТРИС использовать с ENC100_SCK_AL_IO контактный .

TRISDbits.TRISD15

**ENC100_ISR_ENABLE** Указывает на некоторое время, чтобы разрешить прерывание длявывода на основе ввода /

**ENCX24J600** -срабатывает прерывание . Эта функция в настоящее время не реализованы .

IEC1bits.INT2IE

**ENC100_ISR_FLAG** Указывает на флаг прерывания длявывода на основе ввода /

ENCX24J600 -срабатывает прерывание . Эта функция в настоящее время не реализованы .

IFS1bits.INT2IF

**ENC100_ISR_POLARITY** Указывает на битом полярности прерывания длявывода на основе ввода / ENCX24J600 -срабатывает прерывание . Эта функция в настоящее время не реализованы .

INTCON2bits.INT2EP

**ENC100_ISR_PRIORITY** Указывает на приоритетной бит прерывания длявывода на основе ввода / ENCX24J600 -срабатывает прерывание . Эта функция в настоящее время не реализованы .

IPC7bits.INT2IP

**ENC100_SPI_ENABLE** очки в модуль SPI позволяют немного, если используется режим SPI. SPI1STATbits.SPIEN

**ENC100_SPI_IF** Указывает на прерывания флагом для модуля SPI , если режим SPI является используется .

IFS0bits.SPI1IF

**ENC100_SSPBUF** указывает на буфер SPI зарегистрируйтесь для модуля SPI , если SPI режим используется .

SPI1BUF

**ENC100_SPISTAT** очки в состояние SPI зарегистрируйтесь для модуля SPI , если SPI режим используется .

SPI1STAT

**ENC100_SPISTAT** Обеспечивает побитовое доступ к регистре состояния SPI для SPI если используется режим SPI . В ____ бит регистры , как правило, определены в файлах заголовков процессора.

SPI1STATbits

**ENC100_SPICON1** очки в регистре управления SPI для модуля SPI , если SPI режим используется .

SPI1CON1

**ENC100_SPICON1** Обеспечивает побитовое доступ к регистру управления SPI для Модуль SPI , если используется режим SPI (см. ENC_SPISTATbits запись ) .

SPI1CON1bits

**ENC100_SPICON2** очки на второй регистре управления SPI для модуля SPI если используется режим SPI . Если ваше устройство не имеет SPICON2 реестр (например, PIC32 ) просто не пишите это определение.

SPI1CON2

**ENC100_SPIBRG** Указывает на SPI Скорость передачи генератора зарегистрироваться для участия в SPI Модуль если используется режим SPI . Если ваше устройство не имеет Модуль SPI BRG основе , просто не пишите это определение.

SPI1BRG

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

### 9.1.5 PIC18F97J60 Config

18F97J60 можно использовать в вашем приложении , выбрав его в качестве процессора в MPLAB , гарантируя, что ENC_CS_TRIS макро закомментирован , и в том числе файлы " ETH97J60.c " и " ETH97J60.h ". Там никаких дополнительных макросов для определения для 97J60 ; так как он использует свой собственный внутренний MAC и PHY для связи все имена регистров и битовых имен исправлена.

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

### 9.1.6 PIC32MX7XX Config

Для использования PIC32MX795 в вашем проекте , включают ETHPIC32IntMac.c файлов и ETHPIC32ExtPhy.c в вашем проекте.

Вы также должны будете добавить определенный файл реализации PHY (по умолчанию ETHPIC32ExtPhyDP83848.c предоставляется ) в зависимости от фактического отбора внешний PHY .

Обновите следующие определения в **HardwareProfile**.h :

**Макро Образец Цель**

**значение**

PHY_RMII Определите эту макрокоманду , если внешний PHY работает в режиме RMII . Комментарий это, если вы используя PHY MII .

-

PHY_CONFIG_ALTERNATE Определить этот символ , если PIC32MX7XX использует альтернативные флажки конфигурации в подключения (см. стр. 171 ) к PHY . Комментарий это для конфигурации по умолчаниюбулавки .

-

PHY_ADDRESS обновление с MIIM адрес внешнего PHY , который вы используете (адрес на которых PHY реагирует на Miim сделок. См. PHY спецификацию) .

0x1

Обновите следующие определения в TCPIPConfig.h :

**Макро Образец Цель**

**значение**

ETH_CFG_LINK Установите 0, чтобы использовать характеристики соединения по умолчанию ( зависит от выбранного PHY ) .

Установите 1, чтобы настроить ссылку Ethernet на следующие конкретные параметры .

Автоматическое согласование всегда будет включен , если это поддерживается PHY .

0

ETH_CFG_AUTO Установите 1 для использования автоматическое согласование. Настоятельно рекомендуется . 1

ETH_CFG_10 использования / рекламируют 10 Мбит возможности . 1

ETH_CFG_100 использования / рекламировать возможности 100 Мбит . 1

ETH_CFG_HDUPLEX использования / рекламировать полудуплексный возможности . 1

ETH_CFG_FDUPLEX использования / рекламировать полного дуплекса возможности . 1

ETH_CFG_AUTO_MDIX использования / рекламируют возможность автоматического MDIX (вступает в силу только тогда, когда ETH_CFG_AUTO включена ) . 1

ETH_CFG_SWAP_MDIX Использование местами MDIX если они определены . В противном случае используйте обычную MDIX. 1

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

### 9.2 Адрес

TCP / IP приложения нужно будет иметь и управления доступом к среде (MAC) адрес и Интернет -протокола (IP) адрес.

Есть несколько методов для получения или установки этих адресов.

#### 9.2.1 MAC-адрес

6- байт MAC-адрес предоставляет решения для уровня протокола управления доступом к среде стека TCP / IP. MAC адреса являются постоянными адресами, привязанные к оборудованию. Блоки MAC-адресов, продаются организациям и частным лицам по IEEE; если вы не используете Microchip устройство с встроенным MAC-адрес, вам нужно будет приобрести один из этих блоков назначить MAC -адреса к вашей продукции.

MAC-адрес определен в заголовке конфигурации прошивки " **TCPIPConfig.h** ". Есть шесть макросов, которые должны быть определены в этом файле, чтобы установить MAC -адрес. К ним относятся:

**Макро Пример значения**

MY_DEFAULT_MAC_BYTE1 ( 0x00)

MY_DEFAULT_MAC_BYTE2 ( 0x04 )

MY_DEFAULT_MAC_BYTE3 ( 0xA3 )

MY_DEFAULT_MAC_BYTE4 ( 0x00)

MY_DEFAULT_MAC_BYTE5 ( 0x00)

MY_DEFAULT_MAC_BYTE6 ( 0x00)

Каждый из этих макросов представляет собой байт MAC-адрес (заметим, что 00:04 : A3 : xx: xx: xx является блок MAC-адресов зарезервированы для Microchip позиций). После получения вашего блок адресов, вам нужно будет указать уникальный адрес для каждого устройства которое вы вы произведете.

"TCP / IP Демо App" демонстрационный проект описан способ с использованием MPLAB компании Microchip PM3 программисту серийно запрограммировать диапазон MAC-адресов на несколько частей без повторной компиляции проекта.

ENCX24J600 , MRF24WB0M и PIC32MX7XX/6XX оснащены предварительно запрограммированным MAC-адресом (от компании Microchip Адресный блок ). Если вы используете одно из этих устройств в вашем проекте, вы можете определить свой MAC адрес " 00:04 : A3 : 00:00:00 " и стек будет автоматически использовать предварительно запрограммированный адрес со стороны для вашего приложения.

Microchip также предоставляет семейство EEPROMs, которые включают уникальный, запрограммированный заранее EUI -48 (MAC) или EUI- 64 -адрес.

При использовании одного из этих устройств, вы можете написать свой **AppConfig** (см. стр. 138 ) код инициализации так она получит MAC-адрес из одного из этих EEPROMs вместо адреса по умолчанию MAC устройства.

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

### 9.2.2 IP-адрес

IP- адрес используется для адресации узлов в сети интернет-протокола. Вам нужно будет настроить приложение с IP адресом, или включить метод, чтобы получить один. Вы также можете определить несколько других параметров, которые описывают, как ваш устройство будет пытаться вписаться в свою сеть, по умолчанию.

Макросы , которые вам нужно будет определить , включают:

**Макро Образец недвижимости**

**значение**

IP адрес MY_DEFAULT_IP_ADDR_BYTE1 умолчанию байт 1 **192ul**

IP адрес MY_DEFAULT_IP_ADDR_BYTE2 умолчанию байт 2 **168ul**

IP адрес MY_DEFAULT_IP_ADDR_BYTE3 умолчанию байт 3 **1 ul**

IP адрес MY_DEFAULT_IP_ADDR_BYTE4 умолчанию байт 4 **100 ul**

Маска подсети MY_DEFAULT_MASK_BYTE1 умолчанию байт 1 **255ul**

Маска подсети MY_DEFAULT_MASK_BYTE2 умолчанию байт 2 **255ul**

Маска подсети MY_DEFAULT_MASK_BYTE3 умолчанию байт 3 **255ul**

Маска подсети MY_DEFAULT_MASK_BYTE4 умолчанию байт 4 **0ul**

MY_DEFAULT_GATE_BYTE1 Шлюз по умолчанию байт 1 **192ul**

MY_DEFAULT_GATE_BYTE1 Шлюз по умолчанию байт 2 **168ul**

MY_DEFAULT_GATE_BYTE1 Шлюз по умолчанию байт 3 **1 ul**

MY_DEFAULT_GATE_BYTE1 Шлюз по умолчанию байт 4 **1 ul**

Адрес подсети представляет собой битовую маску, которая определяет сферу охвата сети. Если ваш IP- адрес **192.168.5.100**, и указать маску подсети **255.255.255.0**, стек будет предполагать, что адреса в диапазоне **192.168.5.x** находятся в одной подсети, что и Вы, и что пакеты, отправленные на любой из этих адресов не должны быть направлены в друге место.

Шлюз по умолчанию является IP- адресом узла в сети, что ваше приложение будет отправлять пакеты, если он не знает, как маршрут их по адресу он хочет отправить их. Если ваше приложение на **192.268.5.x** подсети, если она хочет, чтобы отправить пакет для **198.175.253.160** и не знаю точно, как туда добраться, он пошлет его на шлюз по умолчанию .

Обратите внимание, что если вы написать свой собственный код вместо начиная с демо-приложения, вам нужно будет заполнить ваш **AppConfig** (см. стр. 138) структуру с этими ценностями. Также обратите внимание, что это только значения по умолчанию. Другие протоколы (или приложение Сам ) может изменить любое из полей **APP_CONFIG**, которые представляют эти параметры .

Есть три способа , которые можно использовать, чтобы установить или получить IP адрес: статический, DHCP или AutoIP .

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

**Статический IP-адресация**

Использование статического адреса позволит вам указать набор IP -адрес. Это может быть сделано во время компиляции, установив IP -адрес по умолчанию к значению вы хотели бы использовать и при помощи демо- код (который населенный ваша структура **AppConfig** автоматически), или во время выполнения, путем программирования приложения, чтобы установить IP- адрес в вашей структуре **AppConfig** основе на некоторых входных сигналов. Если вы хотели бы включать код в ваш проект для приобретения адреса DCHP и AutoIP, но до сих пор используют статический адресации, вы можете вызывать функции DHCP и AutoIP, отключающие те модули, чтобы предотвратить их от перезаписи ваш IP адрес. Использование статических адресов , как правило, работают только если сервер настроен для поддержки этот адрес.

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

**DHCP**

Модуль DHCP -клиент позволит приложению динамически получить IP -адрес от DHCP - сервера в той же самой сети. Делать это сбросить IP -адрес, маску подсети, адрес шлюза, а также некоторые другие параметры конфигурации в вашей структуре **AppConfig**. Для

использования DHCP, включают файлы DHCP.c, DHCPs.c и DHCP.h в проекте, а также раскомментируйте это определение " #define STACK_USE_DHCP_CLIENT " в **TCPIPConfig.h** . IP- стек TCP / также включает в себя просто DHCP -сервер, который может предоставлять IP -адрес для одного клиента DHCP. Чтобы включить эту функцию, добавить макрос " #define STACK_USE_DHCP_SERVER " в **TCPIPConfig.h** .

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

**AutoIP**

Модуль **AutoIP** позволит вашему приложению, чтобы выбрать IP -адрес и утверждают, что само за себя. Эти адреса локальной связи, то есть они не могут быть направлены, и будет работать только на локальном ссылку. Эта функция основана на спецификации для выделения динамических адресов локальной связи, но изменяется принимать форму, используемую APIPA локальный адрес компании Microsoft схема распределения. Чтобы включить эту функцию, включать файлы **AutoIP.c** и **AutoIP.h** и добавить макрос " #define STACK_USE_AUTO_IP " в **TCPIPConfig**.h .

**IP-адрес ( см. стр. 147 ) Модуль Взаимодействие**

Можно настроить по умолчанию статический адрес и включить **DHCP** и **AutoIP** в то же время. Если вы не отключить один или с другой стороны, модуль **AutoIP** сразу выбрать адрес в указанном диапазоне адресов и начинают попытке требовать этого. **DHCP** также начнет отправлять сообщения попытаться арендовать **DHCP** IP -адрес от **DHCP**- сервера. В большинстве случаев модуль **DHCP** завершит все свои сделки до **AutoIP** заканчивается утверждая его адрес. В этом случае **DHCP** адрес будет скопирован в структуру **AppConfig** и модуль **AutoIP** перестанет пытаться претендовать на адрес. С маршрутизируемым **DHCP** адрес всегда предпочитал к локальной связи **AutoIP** адрес, стек также немедленно начать использовать **DHCP** -адрес, если он становится доступным, даже если адрес **AutoIP** уже используется (т.е. если вы включите **DHCP** после **AutoIP** уже заявил, адрес ). Это может привести к существующей розетки потерять связь; они должны быть вновь открыты с новым адресом. В этой ситуации, вы также можете использовать статический адрес, если вы отключите **DHCP** и **AutoIP** и установить статический обратиться в структуре **AppConfig** .

Если **AutoIP** используется в сочетании с модулем сервера **DHCP**, модуль **AutoIP** сгенерирует адрес в **169.254.xx** подсети, а затем служить другой адрес в той же подсети для клиента **DHCP**, подключенного к плате.

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

**9.3 Конфигурация протокола**

Есть несколько шагов, которые вы должны предпринять, чтобы включить каждый протокол в вашем приложении. Большая часть этой конфигурации выполняется, установив параметры в одном из вариантов заголовка файла **TCPIPConfig.h** . Почти все протоколы потребуют от

вас, чтобы позволить их использовать, определить один или несколько макросов в **TCPIPConfig.h** . Кроме того, необходимо будет включить все необходимые файлы на ваши протоколы в ваш проект. Некоторые протоколы потребует от вас, чтобы определить розетки для их использования в **TCPIPConfig.h**, и выделить память для них.

**Мастер настройки TCP / IP**, входит в стек, позволит вам выбрать функции, которые вы хотите при обращении наиболее сложной конфигурации прошивки, автоматически. Из-за этого, это самый простой ( и самый безопасный ) способ настроить ваш прикладные протоколы.

Тема модуль API, имеет описание каждого из модулей.

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

### 9.3.1 Протокол макросы и файлы

Вам нужно будет определить некоторые макросы в **TCPIPConfig.h** и включить некоторые файлы в вашем проекте:

Модуль Макро Функция Требуемые файлы

**ICMP** (см. стр.263)STACK_USE_ICMP_SERVER Обеспечивает возможность запрашивать и реагировать на пинги. ICMP.c, ICMP.h

**ICMP** (см. стр.263)STACK_USE_ICMP_CLIENT Обеспечивает возможность передачи пинги. ICMP.c, ICMP.h

**http2** (см. стр.230)STACK_USE_HTTP2_SERVER Предоставляет функциональные HTTP сервера с динамические переменные, POST, Cookies (см.стр. 91), аутентификации ( см. стр. 89 ), и другие особенности HTTP2.c, HTTP2.h, TCP.c, TCP.h, CustomHTTPApp.c и HTTPPrint.h (см. http2 (см. страница 230 ) раздел для информация о них файлы )

**SSL** ( см. стр. 462) STACK_USE_SSL_SERVER Обеспечивает поддержку сервера SSL сокетов. SSL.c, SSL.h, ARCFOUR.c, ARCFOUR.h, BigInt.c ,BigInt.h, Random.c, Random.h, RSA.c, RSA.h

**SSL** (см. стр.462)STACK_USE_SSL_CLIENT Обеспечивает поддержку клиентских SSL сокетов. SSL.c, SSL.h, ARCFOUR.c, ARCFOUR.h, BigInt.c ,BigInt.h, Random.c, Random.h, RSA.c, RSA.h

**FTP** STACK_USE_FTP_SERVER Обеспечивает возможность удаленного загрузить MPFS2 изображений на http2 серверов через FTP FTP.c, FTP.h, TCP.c, TCP.h

**SMTP** (см. стр.297) STACK_USE_SMTP_CLIENT Обеспечивает возможность отправки электронной почты SMTP.c , SMTP.h , TCP.c , TCP.h , Helpers.c , Helpers.h

**SNMP** (см. стр.318) STACK_USE_SNMP_SERVER Обеспечивает на основе сети машины Протокол управления / мониторинга SNMP.c (см. стр. 449 ) , SNMP.h (см.страница 451 ) , udp.c , UDP.h

**TFTP** (см. стр.576) STACK_USE_TFTP_CLIENT Обеспечивает загрузки ненадежных файлов / скачать услуги TFTPc.c , TFTPc.h , TCP.c , TCP.h

**Telnet** ( см. стр.572) STACK_USE_TELNET_SERVER Обеспечивает телнет услуги. Telnet.c , Telnet.h , TCP.c , TCP.h объявлять (см. страница 155 )

**Announce** STACK_USE_ANNOUNCE Обеспечивает имя хоста устройства и IP -адрес Открытие в локальной подсети Ethernet Announce.c , Announce.h

**DNS** (см. стр.184) STACK_USE_DNS Предоставляет возможность разрешения имен в IP адреса DNS.c , DNS.h , udp.c , UDP.h

**NBNS** ( см. стр.290) STACK_USE_NBNS Предоставляет возможность разрешения имен в IP адреса в той же подсети . NBNS.c , NBNS.h , Udp.c , UDP.h

**SNTP** ( см. стр.457) STACK_USE_SNTP_CLIENT Обеспечивает возможность получить дату / время от Интернет SNTP.c , SNTP.h , Udp.c , UDP.h

**динамический DNS** ( см. стр.193) STACK_USE_DYNAMICDNS_CLIENT Предоставляет возможность разрешения имен в IP адреса , которые часто изменяются . DynDNS.c , DynDNS.h , TCP.c , TCP.h

**MPFS2** (см.страница 271 ) STACK_USE_MPFS2 Обеспечивает MPFS2 услуги по обычаю приложений. Эта функция будет включен / требуется автоматически протоколы стека на базе, которые требуют MPFS2. MPFS2.c , MPFS2.h

**TCP** (см. стр.526) STACK_USE_TCP Предоставляет услуги транспортного уровня TCP для пользовательские протоколы . Эта функциональность автоматически включается / с требованиями протоколы стека на базе , которые требуют TCP розетки. TCP.c , TCP.h

**UDP** (см. стр.607) STACK_USE_UDP Обеспечивает UDP услуги транспортного уровня для пользовательские протоколы . Эта функциональность автоматически включается / с требованиями протоколы стека на базе , которые требуют UDP розетки. Udp.c , UDP.h

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

### 9.3.2 Дополнительные возможности

IP- стек TCP / включает в себя некоторые дополнительные функции , которые можно включить в TCPIPConfig.h .

**Функция Макрос Описание Требуемые файлы**

**UART Демо** STACK_USE_UART Применение демо помощи UART для IP- адреса дисплее и конфигурация стека . Uart.c , uart.h

**UART - на - TCP мост** STACK_USE_UART2TCP_BRIDGE UART для TCP моста пример приложения UART2TCPBridge.c , UART2TCPBridge.h

**IP GLEANING** STACK_USE_IP_GLEANING Позволяет присвоение IP- решения по приему ICMP пакет с действительным IP в режиме конфигурации

**REBOOT_SERVER** (см. стр. 317) STACK_USE_REBOOT_SERVER Позволяет ПИК для сброса удаленно (полезно для загрузчики ) . Reboot.c , Reboot.h

**UDP**

**Performance Test** (См. стр. 293 ) STACK_USE_UDP_PERFORMANCE_TEST UDP тест производительности . Монитор локальную сеть для UDP пакетов с анализатор пакетов . Этот тест передавать 1024 пакетов. использование временные метки из первой и последние пакеты для расчета Пропускная . **UDPPerformanceTest**.c , **UDPPerformanceTest**.h

**TCP**

**Performance Test** (См. стр. 293 ) STACK_USE_TCP_PERFORMACE_TEST производительность TCP тест . Подключите демо доску ПК через UART , выполните Код , и контролировать пропускная на ПК терминал . **TCPPerformanceTest**.c , **TCPPerformanceTest**.h

**BERKELEY_API** (см. стр. 167 ) STACK_USE_BERKELEY_API Обеспечивает Беркли Sockets ( См. стр. 152 ) API уровень абстракции . BerkeleyAPI.c , BerkeleyAPI.h

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

### 9.3.3 Розетки / Sockets

Большинство ваших прикладных протоколов потребует от вас выделить память для каждого соединения (Sockets), что у вас открыт. подобно другие параметры конфигурации микропрограмм, это контролируется по определению макросов в **TCPIPConfig**.h. Для TCP сокетов, Вы должны будете указать четыре параметра инициализации для каждого сокета, в том числе с целью этого сокета, тип память сокет следует хранить в, размер FIFO передачи, а размерполучать FIFO. Стек будет потом инициализации сокетов с этой информацией, и создать блок TCP управления ( УТС) для каждого контролировать свою деятельность. Эта тема будут определены функциональные конфигурации разъем в образце версии **TCPIPConfig**.h , который включен с TCP / IP Демонстрационный проект приложения.

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

#### 9.3.3.1 Распределение памяти

```
#define TCP_ETH_RAM_SIZE ( 3900ul )

#define TCP_PIC_RAM_SIZE ( 0ul )

#define TCP_SPI_RAM_SIZE ( 0ul )

#define TCP_SPI_RAM_BASE_ADDRESS ( 0ul )
```

Первые четыре макросы в разделе гнезда используются для описания общее количество памяти, используемой для хранения сокеты. когда данные передаются из сокета TCP, он сначала будут скопированы в передачи FIFO сокета, а затем в MAC / PHY передачи буфера. Точно так же, полученные данные будут прочитаны с чипа MAC / PHY в приемный FIFO. Эти буферы FIFO, а такжеУТС, могут быть сохранены в 3 местах.

**TCP_ETH_RAM_SIZE** используется для определения оперативной памяти, доступной для розеток о фактическом TCP / IP MAC / PHY чип. Это не будет так же, как в общей памяти на чипе; некоторый объем памяти должны быть зарезервированы для пакетов, переданных и полученных. по по умолчанию ~ 1518 байт ( максимальный размер передачи одного пакета ) будут зарезервированы для TX пакетов на Microchip частей.

количество зарезервированы для приемного буфера пакетов будет равняться сумме оставшегося после выделения памяти для буфера TX и памяти для гнезда. Вы можете получить предупреждение во время компиляции, если буфер RX неоправданно мало.

**TCP_PIC_RAM_SIZE** используется для определения оперативной памяти доступны для разъемов на PIC микроконтроллера , который вождения Приложение.

**TCP_SPI_RAM_SIZE** определяет объем доступной оперативной памяти для гнезда на внешнем SPI памяти (см. Внешние накопители (см. стр.142 ) ). Вы можете указать необходимый адрес в этом RAM чипа для использования с **TCP_SPI_RAM_BASE_ADDRESS** макроса.

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

### 9.3.3.2 патрона Виды

При создании списка инициализации для ваших розеток, вам придется указать тип сокета. Этот параметр будет определять, какие протоколы может использовать сокет. Вы можете создавать и удалять типы сокетов, сколько потребуется. В примере версии **TCPIPConfig.h** следующие типы определены:

```
#define TCP_SOCKET_TYPES

#define TCP_PURPOSE_GENERIC_TCP_CLIENT 0

#define TCP_PURPOSE_GENERIC_TCP_SERVER 1

#define TCP_PURPOSE_TELNET 2

#define TCP_PURPOSE_FTP_COMMAND 3

#define TCP_PURPOSE_FTP_DATA 4

#define TCP_PURPOSE_TCP_PERFORMANCE_TX 5

#define TCP_PURPOSE_TCP_PERFORMANCE_RX 6

#define TCP_PURPOSE_UART_2_TCP_BRIDGE 7

#define TCP_PURPOSE_HTTP_SERVER 8

#define TCP_PURPOSE_DEFAULT 9

#define TCP_PURPOSE_BERKELEY_SERVER 10

#define TCP_PURPOSE_BERKELEY_CLIENT 11

#define END_OF_TCP_SOCKET_TYPES
```

Типы сокетов TCP_PURPOSE_GENERIC_TCP_CLIENT и TCP_PURPOSE_GENERIC_TCP_SERVER используются общий клиент TCP и серверные примеры (см. GenericTCPClient.c и GenericTCPServer.c ). Эти файлы используются

Например, о том, как создать нового, пользовательского TCP клиента или серверное приложение.

Если вы пытаетесь открыть Telnet ( см. стр. 572) соединение, стек будет пытаться использовать TCP_PURPOSE_TELNET гнездо.

Типы сокетов TCP_PURPOSE_FTP_COMMAND и TCP_PURPOSE_FTP_DATA используются для получения команд FTP и данных.

Эти два типа сокетов TCP_PERFORMANCE_x используются исключительно для проведения тестирования производительности TCP .

Тип TCP_PURPOSE_UART_2_TCP_BRIDGE разъем используется для примера моста UART - на - TCP.

Тип TCP_PURPOSE_HTTP_SERVER разъем используется для сокетов на HTTP -серверов, которые слушают (см. стр. 175) для веб- показов страниц запросы.

Тип TCP_PURPOSE_DEFAULT розетку можно использовать для разных приложений, или для приложений, которые только нуждаются в розетки temporairly. Динамические соединения DNS и соединения SMTP использовать сокеты умолчанию и обертки наследия Реализация для **TCPListen** (см. стр. 542 ) и **TCPConnect** (см. стр. 532 ) функции попытке открыть им .

Типы сокетов TCP_PURPOSE_BERKELEY_SERVER и TCP_PURPOSE_BERKELEY_CLIENT показывают, что сокет доступны для использования API Беркли ( см. стр. 167) слой (см. также BSD-сокетов (см. стр. 154 ) ).

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

### 9.3.3.3 Структура инициализации

В заголовке файла **TCPIPConfig**.h, необходимо также определить массив структур для объявления и инициализации любых инструментов, которые вы нужно. Структура выборки:

```
#define TCP_CONFIGURATION ROM struct {

BYTE vSocketPurpose , BYTE vMemoryMedium , СЛОВО wTXBufferSize , СЛОВО
wRXBufferSize }

TCPSocketInitializer [ ] =

{

{ TCP_PURPOSE_GENERIC_TCP_CLIENT , TCP_ETH_RAM , 125, 100} ,

{ TCP_PURPOSE_GENERIC_TCP_SERVER , TCP_ETH_RAM , 20, 20} ,

{ TCP_PURPOSE_TELNET , TCP_ETH_RAM , 200, 150} ,
```

/ / { TCP_PURPOSE_TELNET , TCP_ETH_RAM , 200, 150} ,

/ / { TCP_PURPOSE_TELNET , TCP_ETH_RAM , 200, 150} ,

/ / { TCP_PURPOSE_FTP_COMMAND , TCP_ETH_RAM , 100, 40} ,

/ / { TCP_PURPOSE_FTP_DATA , TCP_ETH_RAM , 0 , 128} ,

{ TCP_PURPOSE_TCP_PERFORMANCE_TX , TCP_ETH_RAM , 200, 1} ,

/ / { TCP_PURPOSE_TCP_PERFORMANCE_RX , TCP_ETH_RAM , 40 , 1500 } ,

{ TCP_PURPOSE_UART_2_TCP_BRIDGE , TCP_ETH_RAM , 256, 256} ,

{ TCP_PURPOSE_HTTP_SERVER , TCP_ETH_RAM , 200, 200} ,

{ TCP_PURPOSE_HTTP_SERVER , TCP_ETH_RAM , 200, 200} ,

{ TCP_PURPOSE_DEFAULT , TCP_ETH_RAM , 200, 200} ,

{ TCP_PURPOSE_BERKELEY_SERVER , TCP_ETH_RAM , 25, 20} ,

/ / { TCP_PURPOSE_BERKELEY_SERVER , TCP_ETH_RAM , 25, 20} ,

/ / { TCP_PURPOSE_BERKELEY_SERVER , TCP_ETH_RAM , 25, 20} ,

/ / { TCP_PURPOSE_BERKELEY_CLIENT , TCP_ETH_RAM , 125, 100} ,

} ;

#define END_OF_TCP_CONFIGURATION

Как вы можете видеть из параметров структуры, четыре параметра вам нужно включить в каждый из ваших заявлений сокетов  являются:

• Разъем цель / Тип

• место хранения ОЗУ

• TX FIFO размер буфера

• RX FIFO размер буфера

Несколько деклараций пример гнездо перечислены. Цель сокет для каждого соответствует одному из типов сокетов (см. стр. 152). Хранение ОЗУ для каждого примера сокета задает расположение TCP_ETH_RAM (чип RAM MAC / PHY). другой варианты TCP_PIC_RAM (собственный ОЗУ ПОС ), и TCP_SPI_RAM ( внешний SPI ОЗУ устройства). Наконец, Техас и RX FIFOs объявлены. Каждый буфер RX должен содержать не менее одного байта, чтобы справиться с SYN и FIN сообщения необходимые TCP.

Каждое гнездо объявлении потребует до 48 байт ПОС RAM и 40 + (Техас размера FIFO) + (размер RX FIFO ) с объемом оперативной памяти носитель, что вы выберите.

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

#### 9.3.3.4 сокетах UDP

UDP сокеты несколько легче объявить, чем TCP сокетов. Поскольку передачи UDP не должны быть обработаны в Конкретный порядок и ответы не требуется отправителем, вы не обязаны декларировать отдельные буферы для этих гнезд.

Есть два варианта, чтобы определить при использовании UDP:

#define MAX_UDP_SOCKETS ( 10U)

/ / #define UDP_USE_TX_CHECKSUM

Определение MAX_UDP_SOCKETS определяет размер массива UDP_SOCKET_INFO (см. стр. 625 ) структур.

Эти структуры содержат два шестнадцать -битные идентификаторы для удаленного узла и номера портов местного узла UDP и 10- байт Структура используется для хранения MAC-адрес удаленного узла и IP -адрес ( эти структуры используют упакованный атрибут, поэтому Фактический размер UDP_SOCKET_INFO (см. стр. 625 ) структура может очень незначительно в зависимости от архитектуры ПОС Вы используете).

Определение UDP_USE_TX_CHECKSUM вызовет стек для генерации контрольных сумм для передаваемых данных, и включить их с переданных пакетов. Это может обеспечить некоторую проверку целостности данных, но также снижает производительность TX почти 50%, если не используетсяENCX24J600 ( в ENCX24J600 чипы включают контрольной суммы аппаратные калькуляторы).

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

#### 9.3.3.5 Использование BSD-сокетов

Параметр конфигурации API сокет Беркли (см. стр. 151 ) потребует Беркли сокеты. Каждый из них внутренне использует один TCP или UDP сокет, определяемый **TCPSocketInitializer** [ ] массива ( см. стр. 153 ) и **MAX_UDP_SOCKETS** (См. стр. 154) четкости. Из-за этого, число Беркли розетки вы объявляете должно быть меньше или равно сумма числа розеток UDP вы объявить и количества TCP Беркли типа сокетов вы объявляете.

**TCPIPConfig.h** макро определить количество Беркли сокетов:

#define BSD_SOCKET_COUNT ( 5u )

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

## 10. Stack API

Модули

Описание Имени
**Announce** ( см. стр. 155 ) Обеспечивает функцию адрес объявление UDP MAC.

**ARP** ( см. стр. 157 ) Обеспечивает адрес (см. стр. 147 ) поддержки протокола разрешения.

**Беркли** (BSD) Розетки (см. стр. 167 ) Обеспечивает сокета обертку BSD на микрочип Стек TCP / IP.

**DNS-клиент** (см. стр. 184 ) Обеспечивает разрешение DNS.

**Динамический DNS-клиент** (см. стр. 193 ) Обновляет внешний IP -адрес в службе динамического DNS.

**Hashes** (см. стр. 202 ) Рассчитывает MD5 и SHA-1 хэш суммы.

**Helpers** (см. стр. 212 ) предоставляет несколько вспомогательную функцию для работы стека.

**Http2 Сервер** ( см. стр. 230 ) Предоставляет развитый встроенный веб-сервер.

**ICMP** ( см. стр. 263 ) Предоставляет функциональные Ping.

**MPFS2** (см. стр. 271 ) Обеспечивает файловую систему облегченный.

**NBNS** (см. стр. 290 ) описывает протокол NetBIOS Name Service.

**Performance Tests - Тесты производительности** ( см. стр. 293 ) Тесты TCP и UDP производительность приложения.

**SMTP** Client (см. стр. 297 ) Отправляет сообщения электронной почты через Интернет.

**Reboot** (см. стр. 317 ) Предоставляет службу для удаленного перезагрузки ПИК.

**SNMP** (см. стр. 318 ) Обеспечивает Простой агент протокол управления сетью.

**Клиент SNTP** (см. стр. 457)Получает абсолютные отметки времени из пула серверов сетевых времени.

**SSL** (см. стр. 462 ) Реализует SSL шифрование для соединений TCP.

**TCP** (см. стр. 526 ) Реализует протокол TCP транспортного уровня.

**Telnet** (см. стр. 572 ) Описывает работу модуля Telnet.

**TFTP** (см. стр. 576 ) Описывает модуль TFTP.

**Tick Module** ( см. стр. 600 ) Обеспечивает точные возможности времени поддержанию .

**UDP** (см. стр. 607 ) Реализует UDP протокол транспортного уровня.

**описание**

Microchip Стек TCP / IP реализован в виде набора модулей. Каждый модуль существует на отдельном слое в слое TCP / IP модель , и имеет свой собственный набор API. Эти API описаны в этом разделе

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

## 10.1 Announce

Этот модуль будет способствовать обнаружение устройств на DHCP включен сетей посредством передачи UDP сообщение на порт 30303 всякий раз, когда локальные изменения IP- адрес. Вы можете изменить порт, используемый объявить модуля, изменив следующие макроопределение в Announce.c.

```
#define ANNOUNCE_PORT 30303
```

Протокол Анонс предназначен для использования с TCP / IP Discoverer ( см. стр. 66) программы PC.

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

### 10.1.1 Announce Stack Members

**Functions**

**Name Description**
AnnounceIP ( смотри страницу 156) -Передает Announce пакета ( смотри страницу 155).

DiscoveryTask ( СМОТРИ страницу156) - Заявите возврат ( смотри страницу 155) задача.

**Module**
Announce ( see page 155)

**Description**
Следующие функции и переменные являются открытыми, но предназначены только обратиться к нему по самой стека. приложения должны, как правило, не называют эти функции или модифицировать эти переменные.

//*-*-*-*-*----------

#### 10.1.1.1 AnnounceIP Function

**File**

Announce.h

**C**
**void AnnounceIP**();

**Side Effects**
None

**Returns**
None

**Description**
AnnounceIP открывает UDP сокет и передает пакет вещания в порт 30303. Если компьютер находится в той же подсети и утилита ищет пакетов на порт UDP, он будет принимать вещание. Для этого приложения, оно используется, чтобы объявить изменение IP -адреса этого платы. Сообщения могут быть просмотрены с помощью программного инструмента TCP / IP Discoverer .

**Замечания**

socket должны быть доступны до эта функция вызывается. Это будет освобожден в конце функции. MAX_UDP_SOCKETS возможно, должны быть увеличены, если другие модули используют UDP сокеты.

**Preconditions**
Stack is initialized()

//*-*-*-*-*----------

### 10.1.1.2 DiscoveryTask Function

**File**
Announce.h

**C**
**void DiscoveryTask**();

**Side Effects**
None

**Returns**
None

**Описание**

Периодическое задача использовать для прослушивания (см. стр. 175) для сообщений Discovery на указанном ANNOUNCE_PORT. эти сообщения могут быть отправлены с использованием Microchip устройства Discoverer инструмента. Если кто-то получил, эта функция будет передавать ответ.

**Замечания**

Разъем UDP должны быть доступны до эта функция вызывается. Это будет освобожден в конце функции. MAX_UDP_SOCKETS возможно, должны быть увеличены, если другие модули используют UDP сокеты.

**Preconditions**
Stack is initialized()

//*-*-*-*-*----------

## 10.2 ARP

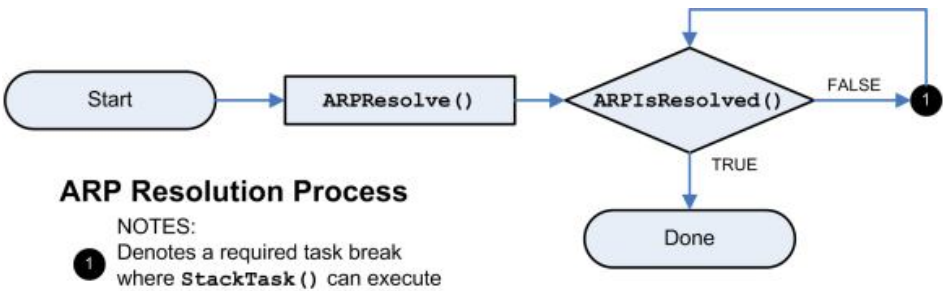
**Types**
ARP_PACKET ( see page166)  смотри страничную структуру пакета ARP

**Описание**

Адрес (см. стр. 147 ) Resolution Protocol , или ARP , является основой слой TCP / IP. Он переводит IP -адреса в физические MAC адреса, или находит шлюз, через который машина может находиться.

Приложению TCP и UDP не нужно будет получать доступ к ARP непосредственно. **TCPOpen** (см. стр. 542 ) и **UDPOpen** (см.страница 611 ) функции будут обрабатывать как **ARP** и операций **DNS** прозрачно.

Ответы на входящие запросы ARP обрабатываются автоматически. Постановление ARP запросов следует простое состояние машины, как указано на рисунке ниже.




//*-*-*-*-*----------

### 10.2.1 ARP Public Members

**Functions**

**ARPResolve** ( смотри страницу158) Передает запрос ARP, чтобы устранить ошибку IP - адрес.

**ARPIsResolved** ( СМОТРИ страницу159) Определяет, является ли запрос ARP была решена еще.

**ARPDeRegisterCallbacks** ( смотри страницу 159) Отмена регистрации обратных вызовов с модулем ARP, которые зарегистрированы ранее.

**ARPRegisterCallbacks** ( СМОТРИ страницу 160) Регистрация обратного вызова с модулем ARP, чтобы получить уведомление о certian событий.

**ARPSendPkt** ( СМОТРИ страницу 160) Передает запрос ARP / Ответ Инициировано по области применения или внешний модуль.

**Macros**
**ARP_REQ** Операции ( смотри страницу 161) код, указывающий Запрос ARP
**ARP_RESP** Операции ( смотри страницу 161) код операции , указывающий на ARP ответ
**MAX_REG_APPS** ( смотри страницу 161)НОМЕР MAX регистраций модулей / приложений Modules/Apps

**Module**
ARP ( see page 157)

**Structures**
**arp_app_callbacks** ( seepage 161)- смотри Это - запись arp_app_callbacks.

**Описание**
Следующие функции и переменные доступны для применения стека.

//*-*-*-*-*----------

#### 10.2.1.1 ARPResolve Function

**File**
ARP.h

**C**
**void ARPResolve**( IP_ADDR* **IPAddr** );

**Описание**
Эта функция передает и ARP запрос для определения аппаратный адрес данного IP -адреса.

**Замечания**
Эта функция требуется только, когда стек является клиентом, и поэтому только когда STACK_CLIENT_MODE является включен.

Чтобы получить результат запроса ARP, вызовите ARPIsResolved ()(см. стр. 159 ) функцию.

**Preconditions**
Ничто

**Parameters**
Описание

**IPAddr** - IP адрес, который будет решен. Адрес должен быть указан в сети байт порядок (Big Endian).

//*-*-*-*-*----------

### 10.2.1.2 ARPIsResolved Function

**File**
ARP.h

**C**
```
BOOL ARPIsResolved(
IP_ADDR* IPAddr,
MAC_ADDR* MACAddr
);
```

**Description**
Эта функция проверяет, если запрос ARP была решена еще, и если да, хранит разрешенный MAC -адрес в указатель предоставляется.

**Remarks**
Эта функция требуется только, когда стек является клиентом, и поэтому только когда STACK_CLIENT_MODE является включен.

**Preconditions**
ARP пакет готов в буфере MAC .

**Parameters**
**IPAddr** - IP адрес, который будет решен. Он должен соответствовать IP -адрес, предоставленный для вызова функции **ARPResolve**( ) (см. стр. 158).

**MACAddr** - буфер для хранения соответствующий MAC-адрес, полученный от запроса ARP.

**Return Values**

**TRUE -** адрес IP решен и поле адреса MACAddr MAC указывает ответ.

**FALSE** - адрес IP еще не решен. Попробуйте вызвать **ARPIsResolved**() снова впоследствии. Если вы не получите ответ по истечении периода ожидания приложений конкретные, вы можете позвонить **ARPResolve**() (см. стр. 158 ), чтобы передать другим ARP -запрос (в случае, если исходный запрос или ответ был потерян в сети ). если вы никогда не получите ответ ARP, это может свидетельствовать о том, что IP- адрес не в использовании.

//*-*-*-*-*----------

#### 10.2.1.3 ARPDeRegisterCallbacks Function

**File**
ARP.h

**C**
BOOL **ARPDeRegisterCallbacks**(
CHAR **id**
);

**Returns**
**ИСТИНА** - В случае успеха
**ЛОЖЬ** - Непредставление неверный reg_id

**Description**
Эта функция позволяет конечному пользователю - приложение отменить регистрацию с обратных вызовов, которые были зарегистрированы ранее . Это называется по Пользователь - приложение, когда его уже не интересует уведомлений от ARP -модуль. Это позволяет другое приложение, чтобы получить зарегистрировано ARP -модуль.

**Preconditions**
None

**Parameters**
**reg_id** Регистрация - ID вернулся в **ARPRegisterCallbacks** ( см. стр. 160 ) вызова

//*-*-*-*-*----------

#### 10.2.1.4 ARPRegisterCallbacks Function

**File**
ARP.h

**C**
CHAR **ARPRegisterCallbacks**(
**struct** arp_app_callbacks * **app**

);

**Returns**
ID > 0 - возврат неотрицательное значение, представляющее идентификатор регистрации и тот же идентификатор должен быть использован в отмене регистрации

-1 -Когда зарегистрированных приложений превышает MAX_REG_APPS (см. стр. 161 ) и нет свободного слота для регистрации

**Description**
Эта функция позволяет применение конечному пользователю зарегистрироваться в обратных вызовов , которые будут вызываться модулем ARP, чтобы дать уведомление Пользователь - приложение о событиях, произошедших в ARP слое. Для бывших : при получении ARP - пакет, который конфликтует с нашей собственный пара адресов (MAC - адрес (см. стр. 147) и IP- адрес ). Это расширение для протокола **Zeroconf** реализация ( ZeroconfLL.c )

**Preconditions**
None

**Parameters**
приложение ARP - приложений структура обратные вызовы поставляется пользовательского приложения

//*-*-*-*-*----------

10.2.1.5 ARPSendPkt Function

**File**
ARP.h

**C**
BOOL **ARPSendPkt**(
DWORD **SrcIPAddr**,
DWORD **DestIPAddr**,
BYTE **op_req**
);

**Returns**
ИСТИНА - Пакет ARP был создан правильно

ЛОЖЬ - не возможно возвращаемого значения

**Description**
Эта функция передает и ARP запрос / ответ , чтобы определить аппаратный адрес данного IP -адреса (или) объявляют (см. стр. 155 ) само- адрес для всех узлов в сети. Расширенный для протокола Zeroconf .

**Remarks**

Этот API, чтобы дать контроль над AR- пакета для внешних модулей.

**Preconditions**
ARP пакет готов в буфере MAC .

**Parameters**
**SrcIPAddr** Исходный адрес IP

**DestIPAddr** Расположение IP-Адрес

**op_req -** Операция Запрос ( ARP_REQ (см. стр. 161) / ARP_RESP (см. стр.161 ))

//*-*-*-*-*----------

#### 10.2.1.6 arp_app_callbacks Structure

**File**
ARP.h

**C**
```
struct arp_app_callbacks {
BOOL used;
void (* ARPPkt_notify)(DWORD SenderIPAddr, DWORD TargetIPAddr, MAC_ADDR*
SenderMACAddr,
MAC_ADDR* TargetMACAddr, BYTE op_req);
};
```

**Description**
Это - запись arp_app_callbacks.

//*-*-*-*-*----------

#### 10.2.1.7 ARP_REQ Macro

**File**
ARP.h

**C**
```
#define ARP_REQ 0x0001u // Операция кода, указывающая Запрос ARP
```

**Description**
Код операции указывая на запрос ARP

//*-*-*-*-*----------

#### 10.2.1.8 ARP_RESP Macro

**File**
ARP.h

**C**
**#define ARP_RESP** 0x0002u *// Операция кода, указывающая Ответ ARP*
**Description**
Код операции указывая на запрос ARP

//*-*-*-*-*----------

### 10.2.1.9 MAX_REG_APPS Macro

**File**
ARP.c

**C**
**#define MAX_REG_APPS** 2 *// MAX допустивших регистрации Modules/Apps*

**Description**
НОМЕР MAX позволили регистраций модулей / приложений

//*-*-*-*-*----------

## 10.2.2 ARP Stack Members

**Functions**
**ARPInit** ( see page 162) - Инициализирует модуль ARP.
**ARPProcess** ( see page162) - Обрабатывает поступающий пакет ARP.

**Module**
ARP ( see page 157)

**Description**
Следующие функции и переменные являются открытыми, но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные.

//*-*-*-*-*----------

### 10.2.2.1 ARPInit Function

**File**
ARP.h

**C**

**void ARPInit**();

**Returns**
None

**Description**
Инициализирует модуль ARP. Эта функция вызывается один раз при загрузке недействительным в кэше поиска.

**Remarks**
Эта функция требуется только, когда стек является клиентом, и поэтому только когда STACK_CLIENT_MODE является включен.

**Preconditions**
None

//*-*-*-*-*----------

### 10.2.2.2 ARPProcess Function

**File**
ARP.h

**C**
BOOL **ARPProcess**();

**Description**
Получает пакет ARP из буфера MAC и определяет, что это ответ на нашу просьбу ( в этом случае ARP является решен ), или если это просьба требуя наш ответ (в этом случае мы передаем друг. )

**Preconditions**
ARP пакет готов в буфере MAC .

**Return Values**
**TRUE** - Вся обработка этого пакета ARP завершена. Не вызывать снова, пока новый ARP пакет ждет в буфере RX.

**FALSE** - Эта функция должна быть вызвана снова. Требуется больше времени, чтобы отправить ARP ответ.

//*-*-*-*-*----------

### 10.2.3 ARP Internal Members

**Functions**

**ARPPut** ( see page 164) - Записывает пакет ARP на MAC.
**SwapARPPacket** ( seepage 164) - Меняет местами порядка байтов информации заголовка в пакете ARP.

**Macros**
ARP_OPERATION_REQ (see page 165) - Код операции указывая на запрос ARP

ARP_OPERATION_RESP (see page 165) - Код операции свидетельствует об ARP ответ

HW_ETHERNET ( seepage 165) - Тип оборудования как определено IEEE 802.3

ARP_IP ( see page 165) - ARP IP тип пакета , как это определено IEEE 802.3

**Module**
ARP ( see page 157)

**Variables**
**Cache** ( see page 165) - кэш для одного ответа ARP Cache for one ARP response

**reg_apps** - ( see page 166) Call-Backs хранения для MAX из двух модулей / приложений

// ARP typedef struct __attribute__((aligned(2), упакованное)) { WORD HardwareType; WORD Protocol; БАЙТ MACAddrLen; БАЙТ ProtocolLen; WORD Operation; MAC_ADDR SenderMACAddr; IP_ADDR SenderIPAddr; MAC_ADDR TargetMACAddr; IP_ADDR TargetIPAddr; } ARP_PACKET ( смотри страницу 166);

**Description**
Следующие функции и переменные определяются как внутренние, к модулю ARP.

//*-*-*-*-*----------

### 10.2.3.1 ARPPut Function

**File**
ARP.c

**C**
```
static BOOL ARPPut(
ARP_PACKET* packet
);
```

**Description**
Записывает пакет ARP на MAC.

**Preconditions**
None

**Parameters**
Указатель в ARP_PACKET структуры ( смотри страницу 166) с правильной операцией и цель pre configured.

**Return Values**
TRUE - пакет ARP был сгенерирован правильно
FALSE - Невозможно возвращаемого значения

**Section**
Помощник Прототипы функций

//*-*-*-*-*----------

### 10.2.3.2 SwapARPPacket Function

**File**
ARP.h

**C**
```
void SwapARPPacket(
ARP_PACKET* p
);
```

**Returns**
None

**Description**
endian- ness Меняет местами порядка байтов информации заголовка в пакете ARP.

**Preconditions**
None

**Parameters**
Пакет ARP будет заменен

//*-*-*-*-*----------

### 10.2.3.3 ARP_OPERATION_REQ Macro

**File**
ARP.h

**C**
```
#define ARP_OPERATION_REQ 0x0001u // Операция кода, указывающая Запрос ARP
```

**Description**
Код операции указывая на запрос ARP

//*-*-*-*-*----------

#### 10.2.3.4 ARP_OPERATION_RESP Macro

**File**
ARP.h

**C**
```
#define ARP_OPERATION_RESP 0x0002u //  Операция кода, указывающая Ответ ARP
```

**Description**
Код  операции свидетельствует об ARP ответ

```
//*-*-*-*-*-----------------------------------------
```

#### 10.2.3.5 HW_ETHERNET Macro

**File**
ARP.c

**C**
```
#define HW_ETHERNET (0x0001u) // Аппаратный тип ARP как определено IEEE 802.3
```

**Description**
ARP Тип оборудования как определено IEEE 802.3

```
//*-*-*-*-*-----------------------------------------
```

#### 10.2.3.6 ARP_IP Macro
**File**
ARP.c

**C**
```
#define ARP_IP (0x0800u) // тип пакета ARP IP как определено IEEE 802.3
```

**Description**
ARP IP тип пакета, как это определено IEEE 802.3

```
//*-*-*-*-*-----------------------------------------
```

#### 10.2.3.7 Cache Variable
Кеш Переменные

**File**
ARP.c

**C**
```
NODE_INFO Cache;
```

**Description**
Кеш для одного ответа ARP

//*-*-*-*-*----------

### 10.2.3.8 reg_apps Variable

**File**
ARP.c

**C**
```
struct arp_app_callbacks reg_apps[MAX_REG_APPS];
```

**Description**
ПАМЯТЬ Call-Backs для MAX двух Modules/Apps

// СТРУКТУРА пакета ARP typedef struct __attribute__((aligned(2), упакованное)) { WORD HardwareType; WORD Protocol; БАЙТ MACAddrLen; БАЙТ ProtocolLen; WORD Operation; MAC_ADDR SenderMACAddr; IP_ADDR SenderIPAddr; MAC_ADDR TargetMACAddr; IP_ADDR TargetIPAddr; } ARP_PACKET ( смотри страницу 166);

//*-*-*-*-*----------

## 10.2.4 Types

**Module**
ARP ( see page 157)

**Structures**
ARP_PACKET ( see page166)- Структура пакета ARP

//*-*-*-*-*----------

### 10.2.4.1 ARP_PACKET Structure

**File**
ARP.h

**C**
```
typedef struct {
WORD HardwareType;
WORD Protocol;
BYTE MACAddrLen;
BYTE ProtocolLen;
WORD Operation;
MAC_ADDR SenderMACAddr;
```

IP_ADDR **SenderIPAddr**;
MAC_ADDR **TargetMACAddr**;
IP_ADDR **TargetIPAddr**;
} ARP_PACKET;

**Members**
**WORD HardwareType**; - Тип протокола канального уровня (Ethernet 1 ) .
**WORD Protocol**; - Протокол верхнего уровня выдачи запроса ARP ( 0x0800 для IPv4 ) ..
**BYTE MACAddrLen;** - Длина адреса MAC (6).
**BYTE ProtocolLen**; - Длина адреса, используемые в протоколе верхнего уровня (4).
**WORD Operation;** - Операция отправитель выполняет **( ARP_REQ** (см. стр. 161 ) или **ARP_RESP** (см. стр. 161 )) .

**MAC_ADDR SenderMACAddr;** - Аппаратное отправителя (MAC ) адрес .

**IP_ADDR SenderIPAddr;** - IP -адреса отправителя .

**MAC_ADDR TargetMACAddr;** - Целевые узловые аппаратные средства (MAC) адрес.
**IP_ADDR TargetIPAddr;** - IP -адрес целевого узла .

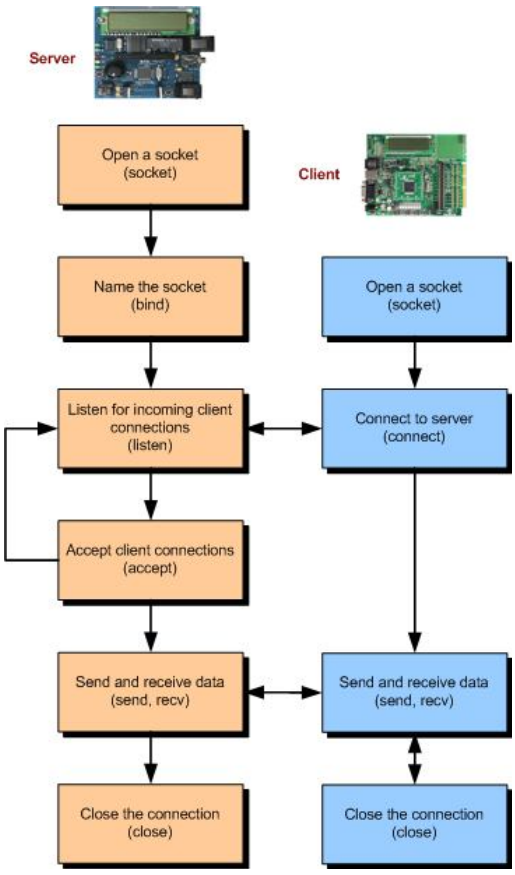
**Description**
СТРУКТУРА пакета ARP

//*-*-*-*-*----------

### 10.3 Berkeley (BSD) Sockets

Распределение Розетки Berkeley (BSD) API-интерфейсы обеспечивают BSD обертку к собственным Microchip TCP / IP API, стек. С помощью этого Интерфейс, программисты, знакомые с BSD-сокетов может быстро разрабатывать приложения с помощью компании Microchip TCP / IP стека.

На рисунке ниже показана типичная взаимодействия для сервера TCP или клиента, используя API, с сокетами.




//*-*-*-*-*----------

### 10.3.1 Общие принадлежности Суперобложки BSD

**Functions**
**Описание Имени**

**accept** ( смотри страницу 169) Эта функция принимает запросы связи поставленные в очередь для слушающей розетки.
**bind** ( смотри страницу 170) Эта функция присваивает имя в дескриптор сокета.
**closesocket** ( смотри страницу 171) Функция closesocket закрывает существующий сокет.
**connect** ( смотри страницу 171) Эта функция подключается к конечному сверстников связи точке.
**gethostname** ( смотри страницу172) Возвращает стандартное имя хоста для системы.

**listen** ( смотри страницу 175) Функция слушать задает указанный сокет в режиме прослушивания

**recv** ( смотри страницу 175 используется для получения входящих данных, которые были в очереди для розетки.

**recvfrom** ( смотри страницу 176) Функция recvfrom ( ) используется для получения входящих данных, которые были в очереди для сокета.

**send** ( смотри страницу 177) Функция отправить используется для отправки исходящих данных на уже подключен розетке.

**sendto** ( смотри страницу 177) Эта функция использованная, чтобы посылать данные как для связи сориентированной так и связь-менее розеток.
**socket** ( смотри страницу 180) Эта функция создает новый сокет Беркли ..

**Macros**

**AF_INET** ( see page 170) Адрес Internet ( смотри страницу 147) - UDP, TCP, и т.п..

**INADDR_ANY** ( see page 173) адрес IP для связи сервера.

**INVALID_TCP_PORT** ( seepage 174)Invalide TCP port

**IP_ADDR_ANY** ( see page 174) Адрес IP ( смотри страницу 147) для сервера связывания

**IPPROTO_IP** ( see page 174) Указывает IP псевдо-протокол.

**IPPROTO_TCP** ( see page 174) Указывает TCP для семейства адреса internet.

**IPPROTO_UDP** ( see page 174) Указывает UDP для семейства адреса internet.

**SOCK_DGRAM** ( see page 178) без установления соединения датаграмма сокета . Используйте UDP для интернет семейства адресов .

**SOCK_STREAM** ( see page178) Соединения на основе потоки байтов . Используйте TCP для интернет семейства адресов .

**SOCKET_CNXN_IN_PROGRESS**( see page 180) Socket ( смотри страницу 180) состояние связи.
**SOCKET_DISCONNECTED** (see page 181) Socket отключен

**SOCKET_ERROR** ( see page181) Socket error

**Module**
Berkeley (BSD) Sockets ( see page 167)

**Structures**
**BSDSocket** ( see page 170) - Berkeley Socket structure

**in_addr-** ( see page 173) in_addr structure

**sockaddr-** ( see page 178) общая структура адреса для всех семейств адреса

**sockaddr_in-** ( see page 179) В семействе адреса Internet

**Types**

**Name Description**
**SOCKADDR** ( see page179)- общая структура адреса для всех семейств адреса

**SOCKADDR_IN** ( see page179) - В адрес в Интернете семьи

**SOCKET** ( see page 180) Socket descriptor

**Description**
Следующие функции и переменные доступны для приложения стека.

//*-*-*-*-*----------

### 10.3.1.1 accept Function

**File**
BerkeleyAPI.h

**C**
```
SOCKET accept(
SOCKET s,
struct sockaddr* addr,
int* addrlen
);
```

**Returns**
Если функция принимает успешно, она возвращает неотрицательное целое число, которое дескриптор для сокета. В противном случае, значение INVALID_SOCKET (см. стр. 529 ) возвращается.

**Description**
Функция принимает используется для приема запросов на соединение в очередь для слушающего сокета. Если запрос на подключение еще не принято, принять удаляет запрос из очереди, и новый сокет для соединения. Оригинальный слушающий сокет остается открытым и продолжает стоять в очереди новых запросов на соединение. Розетка должна быть SOCK_STREAM (см. стр. 178 ) Тип сокета.

**Remarks**
None.

**Preconditions**
слушать (см. 175 стр. ) функция должна быть вызвана.

**Parameters**

**Parameters Description**
**s** - дескриптор Розетки вернулся из предыдущего вызова к гнезду. должен быть связан с локального имени и в режиме прослушивания.

**addr** - Дополнительный указатель на буфер, который получает адрес соединительной лица.
**addrlen** - Факультативный указатель на целое число, которое содержит длину адреса **addr**

```
int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

*sockfd* - это слушающий сокет. Пока просто. *addr* - это обычно указатель на локальную структуру *sockaddr_storage*. В ней будет информация о входящем соединении (с её помощью вы сможете узнать, кто именно и с каким исходящим портом соединялся с вами). *addrlen* - локальная переменная, целое число, которое должно быть устанволено в sizeof(struct sockaddr_storage) перед передачей в accept(). *accept()* не поместит в структуру обьём данных больше, чем вы укажете в этой переменной. Если поместит меньше - то отразит это в значении *addrlen*.

Угадаете, что дальше? *accept* возвращает -1 и устанавливает *errno* в значение ошибки, если что-то пойдёт не так.

//*-*-*-*-*----------

### 10.3.1.2 AF_INET Macro

**File**
BerkeleyAPI.h

**C**
```
#define AF_INET 2 // Internet Address Family - UDP, TCP, etc.
```

**Description**
Internet Address ( see page 147) Family - UDP, TCP, etc.

//*-*-*-*-*----------

### 10.3.1.3 bind Function

**File**
BerkeleyAPI.h

**C**
```
int bind(
SOCKET s,
const struct sockaddr* name,
int namelen
);
```

**Returns**
Если связывать успешно, значение 0 возвращается . Возвращенное значение SOCKET_ERROR (см. стр. 181 ) указывает на ошибку.

**Description**
Функция привязки назначает имя неназванного розетки. Название представляет собой локальный адрес связи конечная точка. Для сокетами типа SOCK_STREAM (см. стр. 178 ),

имя удаленной конечной точки присваивается при подключения (см. стр. 171) или принять (см. стр. 169) функция выполняется.

**Remarks**
None.

**Preconditions**
функция розетки должна быть вызвана.

**Parameters**
**Parameters Description**
**s-** дескриптор Socket возвращался от предшествующего вызова до розетки.
**name** - указатель имени в sockaddr ( розетки. смотри страницу 178) структура, содержащая локальный адрес
**namelen** длина sockaddr структуры ( смотри страницу 178).

```
int bind(int sockfd, struct sockaddr *my_addr, int addrlen);
```

*sockfd* - это файловый дескриптор сокета, возвращаемый вызовом *socket()*. *my_addr* - указатель на *struct soctaddr*, содержащую информацию о вашем адресе, а именно - IP и порте. *addrlen* - это длинна адреса в байтах.

```
//*-*-*-*-*-------------------------------------------
```

### 10.3.1.4 BSDSocket Structure

**File**
BerkeleyAPI.h

**C**
```
struct BSDSocket {
int SocketType;
BSD_SCK_STATE bsdState;
WORD localPort;
WORD remotePort;
DWORD remoteIP;
int backlog;
BOOL isServer;
TCP_SOCKET SocketID;
};
```

**Members**
**Members Description**
int **SocketType**; - Socket type

BSD_SCK_STATE **bsdState**; - Socket state

WORD **localPort**; - local port

WORD **remotePort**; - удаленный порт

DWORD **remoteIP**; - удаленный IP-

int **backlog**; - Максимальное количество или клиент соединение

BOOL **isServer**; - server/client check

TCP_SOCKET **SocketID**; - Socket ID

**Description**
Berkeley Socket structure

//*-*-*-*-*----------

### 10.3.1.5 closesocket Function

**File**
BerkeleyAPI.h

**C**
**int closesocket**(SOCKET **s**);

**Returns**
Если closesocket успешно, значение 0 возвращается . Возвращенное значение SOCKET_ERROR (см. стр. 181 ) (-1) указывает ошибку .

**Description**
Функция **closesocket** закрывает существующий сокет. Эта функция освобождает дескриптор сокета **s**. Любые данные, сохраненные в Гнездо отбрасывается. Если больше не требуется сокет **s**, **closesocket**() должен быть вызван для того, чтобы освободить все ресурсы связаны с **s**.

**Remarks**
None.

**Preconditions**
None.

**Parameters**
**Parameters Description**
**s** - Дескриптор Розетки вернулся из предыдущего вызова до розетки

```
close(sockfd);
```

*Это предотвратит дальнейшее чтение и передачу в сокет. Если кто-то на другом конце попытается что-то прочитать или передать в него, получит ошибку.*

//*-*-*-*-*----------

### 10.3.1.6 connect Function

**File**
BerkeleyAPI.h

**C**
**int connect**(SOCKET **s**,
**struct** sockaddr* **name**,
**int namelen** );

**Returns**
Если функция **Connect ()** завершается успешно, она возвращает 0. Иначе, значение **SOCKET_ERROR** (см. стр. 181 ) возвращается указывает на наличие ошибок. Для потока на основе сокета, если соединение еще не установлено, подключите возвращается **SOCKET_CNXN_IN_PROGRESS** (см.стр.180).

**Description**
Соединение функция присваивает адрес конечной точки одноранговые связи. Для потоковых сокетов, соединение создана между конечными точками. Для датаграммных сокетов адрес фильтр не установлен между конечными точками, пока изменилась с другим подключения функции ( ).

**Remarks**
None.

**Preconditions**
функция розетки должна быть вызвана.

**Parameters**
**Parameters Description**
**s** - дескриптор Розетки вернулся из предыдущего вызова к розетки.
**name** - указатель имени в sockaddr ( и номер порта. смотри страницу 178) структура
**namelen** - содержащая равный адрес длина sockaddr структуры ( смотри страницу 178).

```
int connect(int sockfd, struct sockaddr *serv_addr, int addrlen);
```

*sockfd* - наш файловый дескриптор, который любезно вернул нам *socket()*. *serv_addr* - структура *SOCKADDR*, содержащая порт назначения и IP-адрес; *addrlen* - длинна этой структуры в байтах.
Вся эта информация может быть почерпнута из результатов вызова *getaddrinfo()*.

//*-*-*-*-*-----------------------------------------

### 10.3.1.7 gethostname Function

**File**
BerkeleyAPI.h

**C**
```
int gethostname(
char* name,
int namelen
);
```

**Returns**
Успех будет возвращать значение 0. Если имя слишком коротка, чтобы проводить имя хоста или другая ошибка происходит, SOCKET_ERROR (см. стр. 181 ) (-1), будут возвращены. В случае ошибки * имя будет немодифицированные и настройка нулевой терминатор не будет.

**Description**
Эта функция возвращает стандартное имя хоста системы, который вызывает эту функцию. Возвращенный зовут нулем.

**Remarks**
None.

**Preconditions**
None.

**Parameters**
**Parameters Description**
**name** - Указатель на буфер, который получает имя локального хоста.

**namelen** - размер массива имени.

//*-*-*-*-*-----------------------------------------

### 10.3.1.8 in_addr Structure

**File**
BerkeleyAPI.h

**C**
```
struct in_addr {
        union {
          struct
          {BYTE s_b1, s_b2, s_b3, s_b4;} S_un_b;
```

```
        struct
       {WORD s_w1, s_w2;} S_un_w;
    DWORD S_addr;
  } S_un;
};
```

**Members**
**Members Description**
union {
struct {
BYTE s_b1, s_b2, s_b3, s_b4;
} S_un_b;
struct {
WORD s_w1, s_w2;
} S_un_w;
DWORD S_addr;
} S_un;
union of IP address
struct {
BYTE s_b1, s_b2, s_b3, s_b4;
} S_un_b;
IP address in Byte
struct {
WORD s_w1, s_w2;
} S_un_w;
IP address in Word
DWORD S_addr; IP address

**Description**
in_addr structure

```
//*-*-*-*-*-----------------------------------------
```

### 10.3.1.9 INADDR_ANY Macro

**File**
BerkeleyAPI.h

**C**
```
#define INADDR_ANY 0x00000000u // адрес IP для связи сервера.
```

**Description**
IP address для связи сервера.

```
//*-*-*-*-*-----------------------------------------
```

### 10.3.1.10 INVALID_TCP_PORT Macro

**File**
BerkeleyAPI.h

**C**
```
#define INVALID_TCP_PORT (0L) //Invalide TCP port
```

**Description**
Invalide TCP port

```
//*-*-*-*-*-----------------------------------------
```

### 10.3.1.11 IP_ADDR_ANY Macro

**File**
BerkeleyAPI.h

**C**
```
#define IP_ADDR_ANY 0u // адрес IP для связи сервера.
```

**Description**
IP Address ( see page 147) для связи сервера.

```
//*-*-*-*-*-----------------------------------------
```

### 10.3.1.12 IPPROTO_IP Macro

**File**
BerkeleyAPI.h

**C**
```
#define IPPROTO_IP 0 // Указывает IP псевдо-протокол
```

**Description**
Указывает IP псевдо-протокол .

```
//*-*-*-*-*-----------------------------------------
```

### 10.3.1.13 IPPROTO_TCP Macro

**File**
BerkeleyAPI.h

**C**
```
#define IPPROTO_TCP 6 // Указывает TCP для семейства адреса internet.
```

**Description**
Указывает TCP для семейства адреса internet.

//*-*-*-*-*----------

### 10.3.1.14 IPPROTO_UDP Macro

**File**
BerkeleyAPI.h

**C**
```
#define IPPROTO_UDP 17 // Указывает UDP для семейства адреса internet.
```

**Description**
Указывает UDP для интернет семейства адресов .

//*-*-*-*-*----------

### 10.3.1.15 listen Function

**File**
BerkeleyAPI.h

**C**
```
int listen(
SOCKET s,
int backlog
);
```

**Returns**
Возвращает 0 в случае успешного завершения, иначе вернуться **SOCKET_ERROR** (см. стр. 181 ).

**Description**
Эта функция устанавливает указанный сокет в режим прослушивания. Вызов функции слушать указывает, что приложение готово к принять (см. стр. 169 ) запросы на соединение прибытии в сокет типа **SOCK_STREAM** (см. стр. 178 ).

запрос на подключение в очередь ( если это возможно), пока не принято спринять (см. 169 стр. ) функцию. Параметр отставание определяет максимальное количество ожидающих соединений, которые могут быть поставлены в очередь.

**Remarks**
None

**Preconditions**
свяжите(), по-видимому посещен s розетка сначала.

**Parameters**
**Parameters Description**
**s** Socket- идентификатор вернулся из предварительного розетки() вызова.

**backlog** - Максимальное количество запросов на соединение, которые могут быть помещены в очередь . Заметим, что каждый отставание требует типа **TCP_PURPOSE_BERKELEY_SERVER TCP** сокет быть выделены в **TCPSocketInitializer** [ ] в **TCPIPConfig**.h . Кроме того, убедитесь , **что BSD_SOCKET_COUNT** (также в **TCPIPConfig**.h ) больше, чем отставание по по крайней мере, 1 ( больше, если у вас есть другие BSD-сокетов в использовании ).

```
int listen(int sockfd, int backlog);
```

*sockfd* - это обычный дескриптор файла, полученный вызовом *socket()*, а *backlog* - число возможных соединений во входящей очереди. Что это значит? Входящиее соединения должны ожидать в очереди, пока их не примут (accept()), и это как раз лимит размера очереди ожидающих соединений. Вам скорее всего хватит в большинстве случаев хватит значения от 5 до 10.

Опять же, как обычно, *listen()* возвращает -1 при ошибке и присваивает *errno* её значение.

//*-*-*-*-*----------

### 10.3.1.16 recv Function

**File**
BerkeleyAPI.h

**C**
```
int recv(
SOCKET s,
char* buf,
int len,
int flags
);
```

**Returns**
Если Recv успешно, количество байт, скопированных в буфер приложения BUF возвращается. Нулевое значение указывает на отсутствие доступных данных. Возвращенное значение **SOCKET_ERROR** (см. стр. 181 ) (-1) указывает состояние ошибки . Возвращаемое значение **SOCKET_DISCONNECTED** (см. стр. 181 ) указывает на связь больше не существует .

**Description**
Функция **Recv** ( ) используется для получения входящих данных, которые были в очереди для сокета . Эта функция может быть использована как с датаграммным и потоковым

сокетом. Если имеющиеся данные слишком велики, чтобы уместиться в предоставленный буфер приложения BUF, избыточные байты отбрасываются в случае **SOCK_DGRAM** (см. стр. 178 ) типа сокета. Для **SOCK_STREAM** (см. стр. 178 ) типов, данные буферизируются внутренне так что приложение может извлечь все данные по нескольким вызовов recvfrom (см. стр. 176 ).

**Remarks**
None.

**Preconditions**
Функция подключения (см. стр. 171 ) должна быть вызвана для TCP и UDP сокетов. Серверная часть , принять (см. стр. 169 ) функция должна быть вызвана .

**Parameters**
**Parameters Description**
**s** Socket - дескриптор вернулся из предыдущего вызова к розетки.

**buf** - приемник прикладных данных буфера.

**len** - длина буфера в байтах.
**flags** - не никакого значения в этой реализации

```
int recv(int sockfd, void *buf, int len, int flags);
```

*sockfd* - это дескриптор сокета, из которого мы читаем данные. *buf* - буфер, в который мы читаем информацию, а *len* - максимальный размер буфера приёма. Флаги вновь могут быть установлены в 0.

*recv()* возвращает число бафтов, прочитанных и записанных в буфер, или -1 при ошибке.

Подождите! *recv() может вернуть и 0. Это может означать только одно: удалённая сторона закрыла канал связи с вами. Значение 0, возвращённое recv(), даёт вам знать об этом.*

//*-*-*-*-*----------

### 10.3.1.17 recvfrom Function

**File**
BerkeleyAPI.h

**C**
```
int recvfrom(
SOCKET s,
char* buf,
int len,
int flags,
struct sockaddr* from,
```

**int*** **fromlen**
);

**Returns**
Если **recvfrom** успешно, количество байт, скопированных в буфер приложения **BUF** возвращается. Нулевое значение указывает на отсутствие доступных данных. Возвращенное значение **SOCKET_ERROR** (см. стр. 181 ) (-1) указывает состояние ошибки.

**Description**
Функция **recvfrom** ( ) используется для получения входящих данных, которые были в очереди для сокета. Эта функция может использоваться с как дейтаграммами и socket типа поток. Если имеющиеся данные слишком велики, чтобы уместиться в предоставленный буфер приложения BUF, избыток байт отбрасываются в случае **SOCK_DGRAM** (см. стр. 178 ) типа сокета. Для **SOCK_STREAM** (см. стр. 178 ) типы, данные заносятся в буфер внутренне так что приложение может извлечь все данные по нескольким вызовов **recvfrom**.

**Remarks**
None.

**Preconditions**
socket function должна быть вызвана.

**Parameters**
**Parameters Description**
**s Socket** - Дескриптор Розетки возвращался от предшествующего вызова до розетки.
**buf** - буфера приемника прикладных данных
**len** - длина буфера в байтах.
**flags message flags**. - В данный момент это не поддерживается.

**from** - указатель в sockaddr структуры ( смотри страницу 178), структуру, которая будет заполнена с адрес назначения .

**fromlen** - размер буфера указывался из.

```
int recvfrom(int sockfd, void *buf, int len, unsigned int flags,
        struct sockaddr *from, int *fromlen);
```

*Опять же, так же, как и recv(), добавляется пара полей. from - указатель на локальную структуру sockaddr_storage, которая будет заполнена IP-адресом и портом удалённой машины. fromlen - указатель на локальное int, которое должно быть инициализировано как sizeof *from или как sizeof(struct sockaddr_storage). После вызова функции fromlen будет содержать размер сохранённой информации об удалённом адресе.*

*recvfrom() возвращает число полученных байт или -1 при ошибке.*
//*-*-*-*-*----------

**10.3.1.18 send Function**

**File**
BerkeleyAPI.h

**C**
```
int send(
SOCKET s,
const char* buf,
int len,
int flags
);
```

**Returns**
В случае успеха, отправить возвращает количество отправленных байтов. В случае ошибки, возвращает SOCKET_ERROR (см. стр. 181). ноль указывает на отсутствие отправки данных.

**Description**
Функция **send** используется для отправки исходящих данных на уже подключенный сокет. Эта функция используется для отправки надежным, упорядоченным потоком байтов данных на сокет типа **SOCK_STREAM** (см. стр. 178 ), но также может быть использован для отправки дейтаграммы на сокет типа **SOCK_DGRAM** (см. стр. 178).

**Remarks**
None.

**Preconditions**
Функция подключения (см. стр. 171 ) должна быть вызвана для TCP и UDP сокетов. Серверная часть, принять (см. стр. 169 ) функция должна быть вызвана.

**Parameters**
**Parameters Description**
**s** Socket - возвращался от предшествующего вызова до розетки.
**buf** - Буфер данных приложение, содержащее данные для передачи .

**len** - длина данных в байтах.
**flags** message flags. - К настоящему времени эта область не поддерживана.

```
int send(int sockfd, const void *msg, int len,int flags);
```

***sockfd*** - как обычно, дескриптор сокета, через который вы хотите переслать данные. ***msg*** - указатель на данные, которые вы хотите передать; ***len*** - длинна этих данных в байтах. **Флаги** просто устанавливаем в 0 (см. man send для более подробной информации о флагах).

```
//*-*-*-*-*-----------------------------------------
```

### 10.3.1.19 sendto Function

**File**
BerkeleyAPI.h

**C**
**int sendto**(
SOCKET **s**,
**const char*** **buf**,
**int len**,
**int flags**,
**const struct** sockaddr* **to**,
**int tolen**);

**Returns**
В случае успеха , SendTo возвращает количество отправленных байтов. В случае возвращает ошибку SOCKET_ERROR (см. стр. 181)

**Description**
Функция **SendTo** используется для отправки исходящих данных на сокете. Адрес назначения дается в и Tolen. Оба датаграмный и потоковый сокет поддерживается.

**Remarks**
None.

**Preconditions**
функция розетки должна быть вызвана.

**Parameters**
**Parameters Description**
**s** - сокет дескриптор вернулся из предыдущего вызова сокета.
**buf** - Буфер данных, содержащее данные для передачи.

**len** - длина данных в байтах.
**flags** - сообщения флагов. К настоящему времени эта область не поддерживана.
**to** - Дополнительный указатель в **sockaddr** структуры ( смотри страницу 178), содержащий адрес назначения. Если NULL, в настоящее время связаны удаленный порт и IP- адрес используются в качестве пункта назначения.

**tolen** - длина **sockaddr** структуры ( смотри страницу 178).

//*-*-*-*-*----------

### 10.3.1.20 SOCK_DGRAM Macro

**File**
BerkeleyAPI.h

**C**
**#define SOCK_DGRAM** 110 //Дейтаграмма Без установления логического соединения розетки. Используйте UDP для интернет- адрес семья.

**Description**

Установления соединения датаграммный сокет. Используйте UDP для интернет семейства адресов.

//*-*-*-*-*----------

### 10.3.1.21 SOCK_STREAM Macro

**File**
BerkeleyAPI.h

**C**
```
#define SOCK_STREAM 100 // подключения на основе потока байтов. Используйте TCP для интернет- адрес  семья.
```

**Description**
Соединения на основе потоки байтов. Используйте TCP для интернет семейства адресов.

//*-*-*-*-*----------

### 10.3.1.22 sockaddr Structure

**File**
BerkeleyAPI.h

**C**
```
struct sockaddr {
unsigned short sa_family;
char sa_data[14];
};
```

**Members**
**Members Description**
unsigned short sa_family; address family
char sa_data[14]; до 14 байт прямого обращения

**Description**
общая структура адреса для всех адресных семей

//*-*-*-*-*----------

### 10.3.1.23 SOCKADDR Type

**File**
BerkeleyAPI.h

**C**
**typedef struct** sockaddr **SOCKADDR**;

**Description**
общая структура адреса для всех адресных семей

//*-*-*-*-*----------

### 10.3.1.24 sockaddr_in Structure

**File**
BerkeleyAPI.h

**C**
**struct sockaddr_in** {
**short sin_family**;
WORD **sin_port**;
**struct** in_addr **sin_addr**;
**char sin_zero**[8];
};

**Members**
**Members Description**
**short sin_family**; Адрес семейства ( смотри страницу 147); должно быть AF_INET ( смотри страницу 170).

**WORD sin_port**; Интернет -протокола (IP ) порт .
**struct in_addr sin_addr**; IP -адрес в сетевом порядке байт.

**char sin_zero**[8]; Заполнение сделать структуру того же размера, SOCKADDR (см. стр. 179 ).

**Description**
В семействе адреса Internet

//*-*-*-*-*----------

### 10.3.1.25 SOCKADDR_IN Type

**File**
BerkeleyAPI.h

**C**
**typedef struct** sockaddr_in **SOCKADDR_IN**;

**Description**
В семействе адреса Internet

//*-*-*-*-*----------

#### 10.3.1.26 socket Function

**File**
BerkeleyAPI.h

**C**
```
SOCKET socket(
int af,
int type,
int protocol
);
```

**Returns**
Новый дескриптор сокета. INVALID_SOCKET (см. стр. 529 ) в случае ошибки.

**Description**
Эта функция создает новый BSD сокет для стека микрочип TCPIP. Дескриптор возвращение разъем используется для последующие операции BSD.

**Remarks**
None.

**Preconditions**
функция **BerkeleySocketInit** (см. 181 стр. ) должна быть вызвана.

**Parameters**
**Parameters Description**
**af** Семейство адреса Описания Параметров - AF_INET ( смотри страницу 170).

**type** socket типа SOCK_DGRAM ( смотри страницу 178) или SOCK_STREAM ( смотри страницу 178).

**protocol** IP protocol IPPROTO_UDP ( смотри страницу 174) или IPPROTO_TCP ( смотри страницу 174).

//*-*-*-*-*----------

#### 10.3.1.27 SOCKET Type

**File**
BerkeleyAPI.h

**C**
```
typedef BYTE SOCKET;
```

**Description**
Socket descriptor

//*-*-*-*-*-----------------------------------------

### 10.3.1.28 SOCKET_CNXN_IN_PROGRESS Macro

**File**
BerkeleyAPI.h

**C**
```
#define SOCKET_CNXN_IN_PROGRESS (-2) //Socket connection state.
```

**Description**
Socket Состояние связи

//*-*-*-*-*-----------------------------------------

### 10.3.1.29 SOCKET_DISCONNECTED Macro

**File**
BerkeleyAPI.h

**C**
```
#define SOCKET_DISCONNECTED (-3) //Socket disconnected
```

**Description**
Socket disconnected

//*-*-*-*-*-----------------------------------------

### 10.3.1.30 SOCKET_ERROR Macro

**File**
BerkeleyAPI.h

**C**
```
#define SOCKET_ERROR (-1) //Socket error
```

**Description**
Socket error

//*-*-*-*-*-----------------------------------------

## 10.3.2 BSD Wrapper Stack Members

**Functions**

**Name Description**
BerkeleySocketInit ( see page 181) - смотри Инициализирует структурный массив розетки Berkeley.

**Module**
Berkeley (BSD) Sockets ( see page 167)

**Description**
Следующие функции и переменные являются открытыми, но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные.

//*-*-*-*-*----------

### 10.3.2.1 BerkeleySocketInit Function

**File**
BerkeleyAPI.h

**C**
```
void BerkeleySocketInit();
```

**Returns**
None

**Description**
Эта функция инициализирует массив сокетов Беркли. Эта функция должна быть вызвана перед любой вызов сокета BSD.

**Remarks**
None.

**Preconditions**
None.

//*-*-*-*-*----------

### 10.3.3 BSD Wrapper Internal Members

**Enumerations**

**Name Description**
BSD_SCK_STATE ( see page 182) -
смотри Розетку Berkeley (BSD) состояния

**Functions**
**Name Description**

**HandlePossibleTCPDisconnection** ( смотри страницу 183) -
Внутренняя функция, которая проверяет состояние и наличие асинхронного соединения TCP изменения и resynchs государственные дескриптор BSD сокета, чтобы соответствовать.

**Module**
Berkeley (BSD) Sockets ( see page 167)

**Variables**
**Name Description**
Имя Description

**BSDSocketArray** ( смотри страницу 183) -
Массив **BSDSocket** элементов ( смотри страницу 170); используется для отслеживания всех состояний сокета и информации о соединении.

**gAutoPortNumber** ( СМОТРИ страницу 183)-
Содержит следующий локальный номер порта, чтобы соединяться с розеткой.

**Description**
Содержит следующий номер локального порта для связи с сокетом .

//*-*-*-*-*----------

### 10.3.3.1 BSD_SCK_STATE Enumeration

**File**
BerkeleyAPI.h

**C**
```
typedef enum {
SKT_CLOSED,
SKT_CREATED,
SKT_BOUND,
SKT_BSD_LISTEN,
SKT_LISTEN,
SKT_IN_PROGRESS,
SKT_EST,
SKT_DISCONNECTED
} BSD_SCK_STATE;
```

**Описание Участников**

**SKT_CLOSED** Socket закрыт состояние указывает на бесплатный дескриптор
**SKT_CREATED** Socket создано государство для TCP и UDP сокетов
**SKT_BOUND** Socket связанное состояние для TCP и UDP сокетов
**SKT_BSD_LISTEN** Прослушивание состояние для TCP BSD слушатель ручкой " сервер
**SKT_LISTEN TCP** Соединение с клиентом

**SKT_IN_PROGRESS** TCP в состоянии выполнения Клиент

**SKT_EST TCP** Клиент или сервер состоявшееся государство

**SKT_DISCONNECTED** TCP или сервер больше не подключен к удаленному хосту (но было исторически )

**Description**
Berkeley Socket (BSD) states

//*-*-*-*-*----------

### 10.3.3.2 BSDSocketArray Variable

**File**
BerkeleyAPI.c

**C**
**struct** BSDSocket **BSDSocketArray**[BSD_SOCKET_COUNT];

**Description**
Массив **BSDSocket** (см. стр. 170 ) элементы; используется для отслеживания всю государственную розетки и подключения информацию.

//*-*-*-*-*----------

### 10.3.3.3 gAutoPortNumber Variable

**File**
BerkeleyAPI.c

**C**
WORD **gAutoPortNumber** = 1024;

**Description**
Содержит следующий номер локального порта для связи с сокетом.

//*-*-*-*-*----------

### 10.3.3.4 HandlePossibleTCPDisconnection Function

**File**
BerkeleyAPI.c

**C**
**static** BOOL **HandlePossibleTCPDisconnection**(
SOCKET **s**

);

**Returns**
**ИСТИНА** - Разъем отсоединен

**ЛОЖЬ** - сокет

**Description**
Внутренняя функция, которая проверяет наличие изменений асинхронных государственных соединение TCP и resynchs состояние дескриптора BSD сокета, чтобы соответствовать.

**Preconditions**
None

**Parameters**
**Parameters Description**
**s** - сокет дескриптор типа TCP вернулся из предыдущего вызова к сокету. Этот сокет должен быть в **SKT_LISTEN**, **SKT_IN_PROGRESS**, **SKT_EST** или **SKT_DISCONNECTED** государства.

//*-*-*-*-*----------

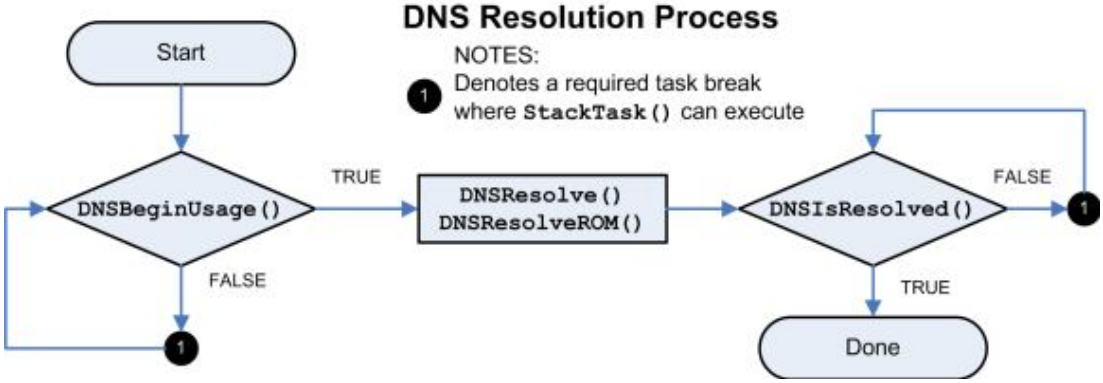
## 10.4 DNS Client

**Доменная Услуга**
Domain Name Service связывает имена хостов (например, www.microchip.com ) с IP- адресов (например, 10.0.54.2 ).

Модуль DNS-клиент обеспечивает разрешающую способность DNS в стек.

Приложения TCP не нужно использовать модуль DNS. Любые операции, необходимые DNS могут быть обработаны в **TCPOpen** (см. 542 функцию страницы). Приложения, созданные с помощью UDP может потребоваться использовать DNS, когда IP- адрес удаленного сервера неизвестно.

Операции в более высоком разрешении DNS следовать простой государственную машину, как указано на рисунке ниже.



//*-*-*-*-*----------

### 10.4.1 DNS Public Members

**Функции**

**Имя Описание**

**DNSBeginUsage** (см.страница 185 ) Требования доступ к модулю DNS.

**DNSEndUsage** (см. стр.185) Релизы управления модуля DNS.

**DNSResolve** (см. стр.186) Начинается разрешение адреса.

**DNSResolveROM** (см.страница 186 ) Начинается разрешение адреса.

**DNSIsResolved** (см. стр. 187) Определяет, является ли разрешение DNS является полной и обеспечивает IP.

**макрос**

**Имя Описание**

**DNS_TYPE_A** (см. стр.187) Константа для типа записи в **DNSResolve** (см. стр. 186). Указывает на A (стандартный адрес) запись.

**DNS_TYPE_MX** (см. стр.188) Константа для типа записи в **DNSResolve** (см. стр. 186). Указывает на MX( почтового обменника ) запись.

**модуль**

DNS-клиент (см. стр. 184 )

**описание**

Следующие функции и переменные доступны для применения стека.

//*-*-*-*-*----------

#### 10.4.1.1 DNSBeginUsage Function

**File**
DNS.h

**C**
BOOL **DNSBeginUsage**();

**Description**
Эта функция действует как семафор, чтобы получить использование модуля DNS. Вызовите эту функцию, и убедитесь, что она возвращает TRUE, перед вызовом любых других интерфейсов API DNS. Вызвав **DNSEndUsage** (см. стр. 185), когда данное приложение больше не нуждается в DNS модуле так, чтобы другие приложения могли использовать его.

**Замечания**

Убедитесь, что **DNSEndUsage** (см. стр. 185 ) всегда вызывается что только ваша заявка получила контроль над DNS Модулем. Если это не будет сделано, стек будет висеть для всех будущих приложений, требующих доступа DNS.

**Preconditions**
Stack is initialized.

**Return Values**
**Return Values Description**
**ИСТИНА** Никакие другие резолюции DNS не ведутся и вызывающее приложение имеет успешно приняты собственности модуля DNS

**ЛОЖЬ** Модуль DNS используется в настоящее время . Доходность к стека и попытаться этот призыв позже

//*-*-*-*-*----------

### 10.4.1.2 DNSEndUsage Function

**File**
DNS.h

**C**
BOOL **DNSEndUsage**();

**Description**
Эта функция действует как семафор , чтобы получить использование модуля DNS. Эта функция вызывается, когда данному приложению больше не нужен модуль DNS так, чтобы другие приложения могут использовать его.

**Замечания**

Убедитесь, что **DNSEndUsage** всегда вызывается что только ваша заявка получила контроль над модулем DNS. Если этого не сделать, стек будет висеть для всех будущих приложений, требующих доступа к DNS .

**Preconditions**
**DNSBeginUsage** ( СМОТРИ страницу 185) возвращавшее ИСТИНУ на предшествующем вызове.

**Return Values**
**Return Values Description**

**ИСТИНА** адрес к имени хоста успешно решены.

**ЛОЖЬ** DNS не смог или адрес не существует .

//*-*-*-*-*----------

### 10.4.1.3 DNSResolve Function

**File**
DNS.h

**C**
```
void DNSResolve(
BYTE* HostName,
BYTE Type
);
```

**Returns**
None

**Description**
Эта функция пытается определить имя хоста к IP адресу. При вызове, он начинает государственную машину DNS. вызов **DNSIsResolved** (см. стр. 187) несколько раз, чтобы определить, является ли разрешение завершена.

Только один DNS Resoultion могут быть выполнены одновременно. Имя хоста не должны быть изменены в памяти, пока резолюция не является завершена.

**Замечания**

Эта функция требует доступ к одному UDP сокету. Если ни один не доступен, **MAX_UDP_SOCKETS** возможно должны быть увеличены.

**Preconditions**
**DNSBeginUsage** ( СМОТРИ страницу 185) возвращавшее ИСТИНУ на предшествующем вызове.

**Parameters**
**Parameters Description**
**HostName** Имя хоста Указатель на нуль завершающим нулем **specifiying** хозяина, для которого необходимо решить IP.

**Type** Тип Запись ( см. стр. 191**) DNS_TYPE_A** (см. стр. 187 ) или **DNS_TYPE_MX** ( см. стр. 188 ) в зависимости от того, какой резолюции записи желательно .

//*-*-*-*-*----------

#### 10.4.1.4 DNSResolveROM Function

**File**
DNS.h

**C**
```
void DNSResolveROM(
ROM BYTE* Hostname,
BYTE Type
);
```

**Returns**
None

**Description**
Эта функция пытается определить имя хоста к IP адресу. При вызове, он начинает государственную машину DNS. Вызов **DNSIsResolved** (см. стр. 187) несколько раз, чтобы определить, является  ли разрешение завершена.

Только один DNS Resoultion могут быть выполнены одновременно. Имя хоста не должны быть изменены в памяти, пока резолюция не является завершена.

**Remarks**
Эта функция требует доступ к одному UDP сокету. Если ни один не доступен, **MAX_UDP_SOCKETS** возможно должен быть увеличен.

Эта функция является псевдонимом для **DNSResolve** (см. стр. 186) на не- PIC18 платформ.

**Preconditions**
**DNSBeginUsage** ( СМОТРИ страницу 185) возвращавшее ИСТИНУ на предшествующем вызове.

**Parameters**
**Parameters Description**
**Hostname**  Имя хоста Указатель на нуль завершающим нулем **specifiying**  хозяина, для которого необходимо решить IP.

**Type**  Тип Запись ( см. стр. 191) **DNS_TYPE_A** (см. стр. 187 ) или **DNS_TYPE_MX** ( см. стр. 188 ) в зависимости от того, какой резолюции записи желательно.

```
//*-*-*-*-*-------------------------------------------
```

#### 10.4.1.5 DNSIsResolved Function

**File**

DNS.h

**C**
BOOL **DNSIsResolved**(
IP_ADDR* **HostIP**
);

**Description**
Вызовите эту функцию, чтобы определить, если разрешение DNS из адреса была завершена. Если да, то разрешен адрес будет предусмотрено в **HostIP**.

**Preconditions**
**DNSResolve** ( СМОТРИ страницу 186) или **DNSResolveROM** ( смотри страницу 186), вызван.

**Parameters**
**Parameters Description**
**HostIP** Указатель на структуру **ip_addr** в которой будут храниться разрешенный IP -адрес раз разрешение будет завершена

**Return Values**
**Return Values Description**
**ИСТИНА** клиент DNS уже получил IP, или процесс DNS обнаружена Ошибка. **HostIP** будет 0.0.0.0 в случае ошибки. Возможные ошибки включают тайм-аут сервера (т.е. DNS-сервер не доступен), имя хоста не в DNS , или ошибки DNS-сервера.

**ЛОЖЬ** Процесс разрешения все еще продолжается .

//*-*-*-*-*----------

### 10.4.1.6 DNS_TYPE_A Macro

**File**
DNS.h

**C**
**#define DNS_TYPE_A** (1u) // Константа для типа записи в DNSResolve . Указывает на A

(стандартный адрес) запись .

//*-*-*-*-*----------

### 10.4.1.7 DNS_TYPE_MX Macro

**File**
DNS.h

**C**

**#define DNS_TYPE_MX** (15u) / / Константа для типа записи в DNSResolve. Указывает

MX (почтовый обменник ) запись.

//*-*-*-*-*----------

### 10.4.2 DNS Internal Members

**Функции**

**Имя Описание**

**DNSPutString** (см. стр.189) Записывает строку в сокет DNS.

**DNSPutROMString** (см.страница 189 ) Записывает ROM строку в сокет DNS.

**DNSDiscardName** (см.стр. 192 ) Читает строку имени или указателя на строку из сокета DNS и отбрасывает его.

**макрос**

**Имя Описание**

**DNS_PORT** (см. стр.190) Порт по умолчанию для разрешений DNS

**DNS_TIMEOUT** (см. стр.190) Прошедшее время, после чего разрешение модулю DNS считается истекло.

**DNS-клиент** (см. стр. 184 )

**структуры**

**Имя Описание**

**DNS_HEADER** (см. стр.192) Структура для заголовка DNS

**Переменные**

**Имя Описание**

**DNSHostName** (см. стр.190) Имя хоста в оперативной памяти для поиска

**DNSHostNameROM** (см.стр. 190 ) Имя хоста в ПЗУ для поиска

**Flags** ( см. стр. 191 ) Сохраняет различные флаги для UDP модуля

**RecordType** (см. стр.191) Тип записи существо сомнению

**ResolvedInfo** (см. стр.191) Узел информация о разрешенном узла

**smDNS** (см. стр. 191) Государственная машина для запроса DNS

**описание**

Следующие функции и переменные обозначены как внутренний в модуль DNS.

//*-*-*-*-*-----------------------------------------

### 10.4.2.1 DNSPutString Function

**File**
DNS.c

**C**
```
static void DNSPutString(
BYTE* String
);
```

**Returns**
None

**Description**
Эта функция записывает строку в сокет DNS, проверяя, что она правильно отформатирована.

**Preconditions**
UDP сокет получены и готовы к записи.

**Parameters**
**Parameters Description**
**String** строка, чтобы записываться в сокет UDP.

**Section**
Function Prototypes

//*-*-*-*-*-----------------------------------------

### 10.4.2.2 DNSPutROMString Function

**File**
DNS.c

**C**
```
static void DNSPutROMString(
ROM BYTE* String
);
```

**Returns**

None
**Description**

Эта функция записывает строку в сокет DNS, проверяя, что она правильно отформатирована.

**Remarks**
Эта функция - aliased на **DNSPutString** ( смотри страницу 189) на платформах non-PIC18.

**Preconditions**
сокет UDP получена и готовый для записи.

**Parameters**
**Parameters Description**
**String** - строка, чтобы записываться в сокет UDP.

//*-*-*-*-*----------

### 10.4.2.3 DNS_PORT Macro

**File**
DNS.c

**C**
```
#define DNS_PORT 53u  // По умолчанию порт для разрешения DNS
```

**Description**
По умолчанию порт для разрешения DNS

//*-*-*-*-*----------

### 10.4.2.4 DNS_TIMEOUT Macro

**File**
DNS.c

**C**
```
#define DNS_TIMEOUT (TICK_SECOND*1) ) / / Прошедшее время, после чего разрешение DNS считается истекло
```

//*-*-*-*-*----------

#### 10.4.2.5 DNSHostName Variable

**File**
DNS.c

**C**
BYTE * **DNSHostName**;

**Description**
Имя хоста в оперативной памяти для поиска

//*-*-*-*-*----------

#### 10.4.2.6 DNSHostNameROM Variable

**File**
DNS.c

**C**
ROM BYTE * **DNSHostNameROM**;

**Description**
Имя хоста в ПЗУ для поиска

//*-*-*-*-*----------

#### 10.4.2.7 Flags Variable

**File**
UDP.c

**C**
```
struct {
unsigned char bFirstRead : 1;
unsigned char bWasDiscarded : 1;
} Flags;
```

**Members**
**Members Description**
unsigned char **bFirstRead** : 1 ; Данные не считаны из этого сегмента еще
unsigned char **bWasDiscarded** : 1; The Данные в этом сегменте было сброшено

**Description**
Загружает различные флаги для модуля UDP

//*-*-*-*-*----------

#### 10.4.2.8 RecordType Variable

**File**
DNS.c

**C**
BYTE **RecordType**;

**Description**
Тип записи существо сомнению

//*-*-*-*-*----------

#### 10.4.2.9 ResolvedInfo Variable

**File**
DNS.c

**C**
NODE_INFO **ResolvedInfo**;

**Description**
Узел информация о разрешенном узла

//*-*-*-*-*----------

#### 10.4.2.10 smDNS Variable

**File**
DNS.c

**C**
```
enum {
DNS_START = 0,
DNS_ARP_START_RESOLVE,
DNS_ARP_RESOLVE,
DNS_OPEN_SOCKET,
DNS_QUERY,
DNS_GET_RESULT,
DNS_FAIL,
DNS_DONE
} smDNS;
```

**DNS_START** = 0  Исходное состояние для сброса переменных состояния клиента

**DNS_ARP_START_RESOLVE** Отправить разрешение ARP из DNS -сервера или шлюза MAC-адрес

**DNS_ARP_RESOLVE** Подождите ответ на ARP запрос

**DNS_OPEN_SOCKET** Откройте сокет UDP

**DNS_QUERY** Отправить DNS запрос, чтобы DNS -сервер

**DNS_GET_RESULT** Дождитесь ответа от DNS -сервера

**DNS_FAIL ARP** или DNS- сервер не отвечает

**DNS_DONE** запрос DNS закончена

**Description**
Государственная машина для запроса DNS

//*-*-*-*-*----------

### 10.4.2.11 DNS_HEADER Structure

**File**
DNS.c

**C**
```
typedef struct {
WORD_VAL TransactionID;
WORD_VAL Flags;//  Сигнализирует
WORD_VAL Questions;  //  Спрашивает;
WORD_VAL Answers;    //  Отвечает;
WORD_VAL AuthoritativeRecords;
WORD_VAL AdditionalRecords;

} DNS_HEADER;
```

**Description**
Структура для заголовка DNS

//*-*-*-*-*----------

### 10.4.2.12 DNSDiscardName Function

**File**
DNS.c

**C**
```
static void DNSDiscardName();
```

**Returns**
None

**Description**
Эта функция читает строку имени из розетки DNS. Каждая строка состоит из серии этикеток. Каждая метка состоит из Длина префикса байт, за которой следуют байты этикеток. В конце строки, метка нулевой длины находится как прекращения. Если имя сжатия используется, эта функция автоматически определяет указатель и выбросьте его.

**Preconditions**
UDP socket получена и готовый для чтения имени DNS

//*-*-*-*-*----------

## 10.5 Dynamic DNS Client

Клиентский модуль Dynamic DNS предоставляет способ обновления динамический IP -адрес в службе общественного DDNS. эти услуги могут быть использованы, чтобы обеспечить отображение имени хоста DNS для устройств, которым маршрутизаторы, брандмауэры и / или в сетях, могут динамически назначать IP- адреса.

Заметим, что это только решает одну из двух задач для связи с устройствами на локальных подсетей из Интернета. в то время как Динамический DNS может помочь найти устройство, маршрутизатор или брандмауэр он сидит сзади, должен еще правильно перенаправлять входящие запрос на подключение. Как правило, это требует перенаправление портов должен быть настроен для маршрутизатора, за которой устройство находится.

Клиент Dynamic DNS поддерживает популярный интерфейс, используемый DynDNS.org, No-IP.com и DNS -O- Matic.com.

ВАЖНО: Динамические услуги DNS предусматривают, что обновления не следует не чаще, чем через 10 минут, и только тогда, когда IP- адрес изменился. Обновления, сделанные чаще, чем, которые считаются оскорбительными, и в конечном итоге может привести ваш аккаунт отключен. Производственные устройства, которые получают часто перезагружается, возможно, потребуется, чтобы сохранять последний известный IP в энергонезависимой памяти. Вы также не должны позволять этот модуль во время тестирования остальную часть вашего приложения.

//*-*-*-*-*----------

### 10.5.1 Dynamic DNS Public Members

**Перечисления**

**Имя Описание**

**DDNS_SERVICES** (см.стр. 195 )Динамические услуги DNS. Должен поддерживать DynDNS (см. стр. 193 ) API ( Auxlang ) и соответствуют ddnsServiceHosts (см. стр. 200 ) и ddnsServicePorts (см. стр. 200) в DynDNS.c.

**DNS_STATUS** (см. стр.195)Сообщение о состоянии для DynDNS ( см. стр. 193) клиента. ХОРОШЕЕ и NOCHG являются хорошо, но НАРУШЕНИЯХ через 911 являются фатальными. БЕЗ ИЗМЕНЕНИЙ через недействительные, локально определена.

**Функции**

**Имя Описание**

**DDNSForceUpdate** (см.стр. 196 ) Принудительное немедленное обновление DDNS

**DDNSGetLastIP** (см. стр.197) Возвращает последний известный внешний IP -адрес устройства.

**DDNSGetLastStatus** (см.страница 197 ) Возвращает состояние последнего обновления.

**DDNSSetService** (см.страница 197 ) Выбор предварительно настроенной динамической службы DNS

**модуль**

Динамический DNS-клиент (см. стр. 193 )

**структуры**

**Имя Описание**

**DDNS_POINTERS** (см.страница 194)Параметры конфигурации для Клиента динамического DNS

**Переменные**

**Имя Описание**

**DDNSClient** (см. стр. 196 ) параметры конфигурации для модуля

**описание**

Эти функции и переменные предназначены для вызова приложением стека.

//*-*-*-*-*----------

### 10.5.1.1 DDNS_POINTERS Structure

**File**
DynDNS.h
**C**

```
typedef struct {
```

```
union {
BYTE * szRAM;
ROM BYTE * szROM;
} CheckIPServer;
WORD CheckIPPort;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} UpdateServer;
WORD UpdatePort;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} Username;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} Password;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} Host;
struct {
unsigned char CheckIPServer : 1;
unsigned char UpdateServer : 1;
unsigned char Username : 1;
unsigned char Password : 1;
unsigned char Host : 1;
} ROMPointers;
} DDNS_POINTERS;
```

**Description**
Эта структура указателей настраивает клиента динамического DNS. Изначально все указатели будут иметь законной клиент будет отключен.

Установите **DDNSClient** ( см. стр. 196 ). [ Имя поля ]. **SzRAM** использовать строку, хранящуюся в памяти, или **DDNSClient** (см. стр. 196 ). [ имя поля ]. **szROM** использовать строку, хранящуюся в ПЗУ. ( Где [ имя поля ] является одним из параметров ниже).

Если диск строка определяется, **DDNSClient**.**ROMPointers**. [ Имя поля ] также должен быть установлен в 1, чтобы указать, что это поле должны быть извлечены из ПЗУ вместо ОЗУ.

**Parameters**
**Parameters Description**

**CheckIPServer** сервера, используемого для определения внешнего IP -адреса

**CheckIPPort** Порт на выше -сервер для подключения (см. стр. 171), чтобы

**UpdateServer** сервера, на котором обновления должны быть вывешены

**UpdatePort** Порт на выше -сервер для подключения (см. стр. 171), чтобы

**Username** Имя пользователя для динамического сервера DNS

**Password** Пароль поставлять при выполнении обновлений

**Host** имя хоста, который вы хотите обновить

**ROMPointers** Указывает, какие параметры для чтения из ПЗУ вместо оперативной памяти.

//*-*-*-*-*----------

#### 10.5.1.2 DDNS_SERVICES Enumeration

**File**
DynDNS.h

**C**
```
typedef enum {
DYNDNS_ORG = 0u,
NO_IP_COM,
DNSOMATIC_COM
} DDNS_SERVICES;
```

**Members**
**Members Description**
DYNDNS_ORG = 0u www.dyndns.org
NO_IP_COM www.no-ip.com
DNSOMATIC_COM www.dnsomatic.com

**Description**
Динамические услуги DNS. Должен поддерживать **DynDNS** ( см. стр. 193) API ( Auxlang ) и соответствуют **ddnsServiceHosts** ( см. стр. 200) и **ddnsServicePorts** (см. стр. 200) в DynDNS.c.

//*-*-*-*-*----------

#### 10.5.1.3 DDNS_STATUS Enumeration

**File**
DynDNS.h

**C**
```
typedef enum {
DDNS_STATUS_GOOD = 0u,
DDNS_STATUS_NOCHG,
```

**DDNS_STATUS_ABUSE**,
**DDNS_STATUS_BADSYS**,
**DDNS_STATUS_BADAGENT**,
**DDNS_STATUS_BADAUTH**,
**DDNS_STATUS_NOT_DONATOR**,
**DDNS_STATUS_NOT_FQDN**,
**DDNS_STATUS_NOHOST**,
**DDNS_STATUS_NOT_YOURS**,
**DDNS_STATUS_NUMHOST**,
**DDNS_STATUS_DNSERR**,
**DDNS_STATUS_911**,
**DDNS_STATUS_UPDATE_ERROR**,
**DDNS_STATUS_UNCHANGED**,
**DDNS_STATUS_CHECKIP_ERROR**,
**DDNS_STATUS_INVALID**,
**DDNS_STATUS_UNKNOWN**
} DDNS_STATUS;

**Members**
**Members Description**
**DDNS_STATUS_GOOD** = 0u обновление успешно, имя хоста теперь обновляется

**DDNS_STATUS_NOCHG** обновление не изменилось дополнительную настройку и считается оскорбительным. Дополнительная ' nochg ' обновления вызовет хоста будет заблокирован.

**DDNS_STATUS_ABUSE** Имя хоста указано заблокирован за злоупотребления обновления.

**DDNS_STATUS_BADSYS** Параметр система не действует. Должно быть DynDNS, statdns или обычаев.

**DDNS_STATUS_BADAGENT** Агент пользователя был заблокирован или не отправлены.

**DDNS_STATUS_BADAUTH** Имя пользователя и пароль пара не совпадают реального пользователя.

**DDNS_STATUS_NOT_DONATOR** опция доступна только для зачисленных пользователей (например, в автономном URL ) был указан, но пользователь не приписывают пользователь. Если несколько хостов были указаны, только один! даритель будет возвращен.

**DDNS_STATUS_NOT_FQDN** Имя хоста указано не полное доменное имя ( не в виде hostname.dyndns.org или domain.com ).

**DDNS_STATUS_NOHOST** Имя хоста указано не существует в этой учетной записи пользователя (или не в Сервис, указанный в параметре системы ).

**DDNS_STATUS_NOT_YOURS** Имя хоста указано не принадлежат к этой учетной записи пользователя.

**DDNS_STATUS_NUMHOST** Слишком много хозяев, указанные в обновлении.

**DDNS_STATUS_DNSERR** Неизвестная ошибка DNS столкнулись службой DDNS.

**DDNS_STATUS_911** Существует проблема или плановое техническое обслуживание со службой DDNS.

**DDNS_STATUS_UPDATE_ERROR** Ошибка связи с обновлением службы.

**DDNS_STATUS_UNCHANGEDIP** проверка показала, что в обновлении не было необходимости.

**DDNS_STATUS_CHECKIP_ERROR** Ошибка связи с обслуживанием CheckIP.

**DDNS_STATUS_INVALID DDNS** Клиентские данные является недопустимым.

**DDNS_STATUS_UNKNOWN DDNS** Клиент еще не был выполнен с такой конфигурацией.

**описание**

Сообщение о состоянии для **DynDNS** ( см. стр. 193) клиента. ХОРОШЕЕ и NOCHG в порядке, но НАРУШЕНИЯХ через 911 являются фатальными.

БЕЗ ИЗМЕНЕНИЙ через INVALID локально определены.

//*-*-*-*-*----------

#### 10.5.1.4 DDNSClient Variable

**File**
DynDNS.c

**C**
DDNS_POINTERS **DDNSClient**;

**Description**
Параметры Конфигурации для модуля

//*-*-*-*-*----------

#### 10.5.1.5 DDNSForceUpdate Function

**File**
DynDNS.h

**C**
**void DDNSForceUpdate**();

**Returns**
None

**Description**
Эта функция заставляет DDNS Client немедленно выполнить полное обновление. Любое сообщение об ошибке очищается, и обновление будет выполнено несмотря, изменился ли IP-адрес или нет. Эта функция вызывается каждый раз, когда параметры **DDNSClient** ( см. стр. 196 ) были изменены.

**Preconditions**
**DDNSInit** ( СМОТРИ страницу 198), по-видимому вызван.

//*-*-*-*-*----------

### 10.5.1.6 DDNSGetLastIP Function

**File**
DynDNS.h

**C**
IP_ADDR **DDNSGetLastIP**();

**Returns**
Последний известный внешний IP- адрес устройства.

**Description**
Эта функция возвращает последний известный внешний IP -адрес устройства.

**Preconditions**
None

//*-*-*-*-*----------

### 10.5.1.7 DDNSGetLastStatus Function

**File**
DynDNS.h

**C**
DDNS_STATUS **DDNSGetLastStatus**();

**Returns**
DDNS_STATUS (см. стр. 195 ), что указывает код статуса для последнего обновления.

**Description**
Эта функция возвращает статус последнего обновления. См. DDNS_STATUS ( см. стр. 195) перечисление для  Возможные коды.

**Preconditions**

None

//*-*-*-*-*-----------------------------------------

#### 10.5.1.8 DDNSSetService Function

**File**
DynDNS.h

**C**
```
void DDNSSetService(
DDNS_SERVICES svc
);
```

**Returns**
None

**Description**
Эта функция выбирает динамическую службу DNS на основе параметров, настроенных в **ddnsServiceHosts** (см. стр. 200 ) и **ddnsServicePorts** (см. стр. 200). Эти массивы должны соответствовать **DDNS_SERVICES** (см. стр. 195) перечисления.

**Preconditions**
None

**Parameters**
**Parameters Description**
**svc** - один из DDNS_SERVICES элементов ( смотри страницу 195), чтобы указывать выбранную услугу

//*-*-*-*-*-----------------------------------------

### 10.5.2 Dynamic DNS Stack Members

**Functions**
**Name Description**
**DDNSInit** ( СМОТРИ страницу 198), Инициализирует Динамический модуль DNS.
**DDNSTask** ( СМОТРИ страницу 198) Динамический DNS задача клиент / состояние машины.
**Module**
Динамический DNS-клиент ( смотри страницу 193)

**Description**
Следующие функции и переменные являются открытыми, но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные.

//*-*-*-*-*-----------------------------------------

#### 10.5.2.1 DDNSInit Function

**File**
DynDNS.h

**C**
**void DDNSInit**();

**Returns**
None

**Description**
Эта функция инициализирует клиента динамического DNS. Она очищает **DDNSClient** ( см. стр. 196 ) структуру указатели, и говорит модуль попытка первое обновление через 15 секунд прошло ( так, чтобы позволить конфигурация DHCP в stabalize ).

**Remarks**
Эта функция вызывается только один при жизни приложения.

**Preconditions**
None

//*-*-*-*-*-----------------------------------------

#### 10.5.2.2 DDNSTask Function

**File**
DynDNS.h

**C**
**void DDNSTask**();

**Returns**
None

**Description**
Эта функция выполняет фоновые задачи Клиента динамического DNS. После того, как структура **DDNSPointers** настроен, эта попытка задача обновить **Dynamic** DNS имя хоста на периодической графику.

Задача первая обращается к серверу **CheckIP**, чтобы определить текущий внешний IP -адрес устройства. Если IP- адрес имеет изменения, она выдает команду обновления в службе динамического DNS для распространения изменений. Эта последовательность выполняется

всякий раз, когда **dwUpdateAt** (см. стр. 200) истекает, который по умолчанию является каждые 10 минут, или когда обновление вынужденные.

**Remarks**
Эта функция действует как задачи (по аналогии с одним в OCPB ). Он выполняет свою задачу в кооперативном образом, и главное приложение должно периодически вызывать эту функцию для того, чтобы ее задачи выполнялись своевременно.

**Preconditions**
**DDNSInit** ( see page 198)() has been called.

**Section**
Function Prototypes

//*-*-*-*-*----------

### 10.5.3 Dynamic DNS Internal Members

**макрос**

**Имя Описание**

**DDNS_CHECKIP_SERVER**(См. стр. 201 )По умолчанию **CheckIP** сервер для определения текущего IP -адреса

**DDNS_DEFAULT_PORT** (см. стр. 202)Порт по умолчанию для сервера **CheckIP**

**модуль**

Динамический DNS-клиент (см. стр. 193 )

**Переменные**

**Имя Описание**

**bForceUpdate** (см. стр.200) Указывает, что обновление должно быть сделано независимо от того изменился или нет IP. Используйте этот флаг , когда пользователь / пасс / хоста изменились.

**ddnsServiceHosts** (см.стр. 200 ) Имена для различных динамических услуг DNS хоста

**ddnsServicePorts** (см.стр. 200 ) Номера портов для различных динамических услуг DNS

**dwUpdateAt** (см. стр.200) Указывает, что на следующий **CheckIP** должно быть сделано

**lastKnownIP** (см. стр.200) Последний известный IP- адрес этого устройства

**LastStatus** ( см. стр. 201) ответ статуса с последнего обновления

**_checkIpSrvrResponse** (см. стр. 201) Разделитель , чтобы найти IP -адрес с сервера **CheckIP**

**_updateIpSrvrResponse** (см. стр. 201)  Коды ответа от **DynDNS** (см. стр. 193 ) Обновление сервера

**описание**

Следующие функции и переменные определяются как внутренние, в Dynamic модуля DNS.

//*-*-*-*-*----------

#### 10.5.3.1 bForceUpdate Variable

**File**
DynDNS.c

**C**
BOOL **bForceUpdate**;

**Description**
Указывает, что обновление должно быть сделано независимо от того изменился или нет IP. Используйте этот флаг, когда  USER / PASS / хоста изменились.

//*-*-*-*-*----------

#### 10.5.3.2 ddnsServiceHosts Variable

**File**
CustomHTTPApp.c

**C**
ROM **char** * ROM **ddnsServiceHosts**[];

**Description**
Имена для различных динамических услуг DNS хоста

//*-*-*-*-*----------

#### 10.5.3.3 ddnsServicePorts Variable

**File**
DynDNS.c

**C**
ROM WORD **ddnsServicePorts**[] = { 80, 80, 80, };

**Description**
Номера портов для различных динамических услуг DNS

//*-*-*-*-*----------

### 10.5.3.4 dwUpdateAt Variable

**File**
DynDNS.c

**C**
DWORD **dwUpdateAt**;

**Description**
Указывает, что на следующий **CheckIP** должно быть сделано

//*-*-*-*-*----------

### 10.5.3.5 lastKnownIP Variable

**File**
DynDNS.c

**C**
IP_ADDR **lastKnownIP**;

**Description**
Последний известный IP- адрес этого устройства

//*-*-*-*-*----------

### 10.5.3.6 lastStatus Variable

**File**
DynDNS.c

**C**
DDNS_STATUS **lastStatus**;

**Description**
Ответ статуса с последнего обновления

//*-*-*-*-*----------

### 10.5.3.7 _checkIpSrvrResponse Variable

**File**
DynDNS.c

**C**
ROM BYTE **_checkIpSrvrResponse**[] = "Address:";

**Description**
Разделитель, чтобы найти IP -адрес с сервера **CheckIP**

//*-*-*-*-*----------

### 10.5.3.8 _updateIpSrvrResponse Variable

**File**
DynDNS.c

**C**
```
ROM char* _updateIpSrvrResponse[] = { "good", "nochg", "abuse", "badsys", "badagent",
"badauth", "!donator", "notfqdn", "nohost", "!yours", "numhost", "dnserr", "911", };
```

**Description**
Коды ответа от **DynDNS** (см. стр. 193 ) Обновление сервера

//*-*-*-*-*----------

### 10.5.3.9 DDNS_CHECKIP_SERVER Macro

**File**
DynDNS.h

**C**
```
#define DDNS_CHECKIP_SERVER (ROM BYTE*)"checkip.dyndns.com" / / По умолчанию
CheckIP  сервер для определения текущего IP -адреса
```

//*-*-*-*-*----------

### 10.5.3.10 DDNS_DEFAULT_PORT Macro

**File**
DynDNS.h

**C**
```
#define DDNS_DEFAULT_PORT (80u) // По умолчанию порт для сервера CheckIP
```

//*-*-*-*-*----------

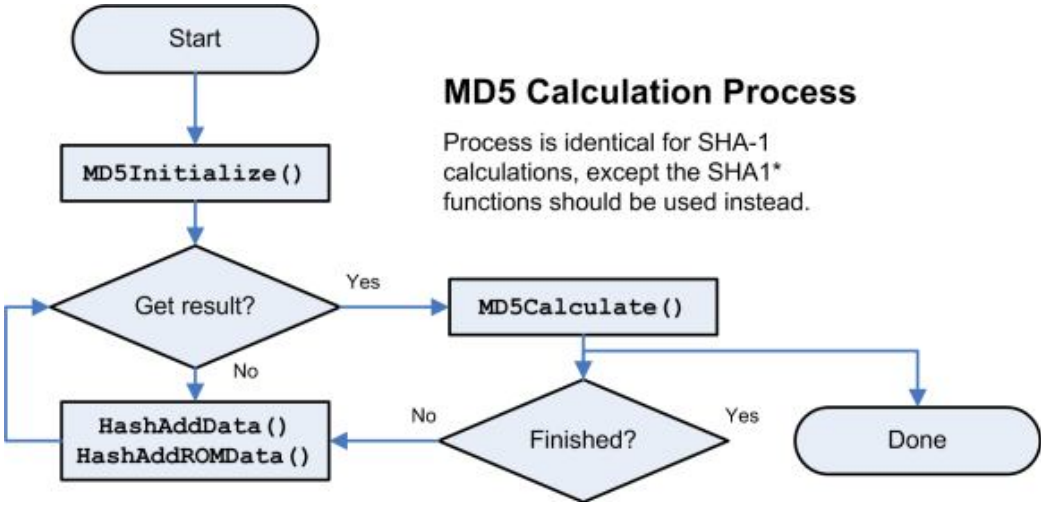
## 10.6 Hashes

Модуль **Hashes** вычисляет MD5 и / или SHA- 1 хэш суммы данных. Хэш суммы односторонним переварить функции, то есть что исходное сообщение не может быть выведено из хэша сообщения. Столкновения , в то время чрезвычайно редки, все же существуют.

Тем не менее, они очень трудно создать.

Хэш-функции, как правило, используются для целей целостность сообщения и аутентификации. Они широко используются протоколы шифрования, такие как SSL, чтобы убедиться, что сообщение не было изменено с во время транспортировки.

Следующая диаграмма потока показывает, как использовать этот модуль.




Чтобы использовать хэш-функций, в первую объявить HASH_SUM (см. стр. 206 ) структуру и передать указатель на него либо **MD5Initialize** (см. стр. 205 ) или **SHA1Initialize** (см. стр. 205 ). Затем вызовите **HashAddData** (см. стр. 203 ) или **HashAddROMData** (см. стр. 203 ) столько раз, сколько необходимо, чтобы обеспечить все данные в хэш. Позвоните **MD5Calculate** (См. стр. 204 ) или **SHA1Calculate** (см. стр. 205) в любое время, чтобы получить хеш до текущей точки. после расчет, продолжать добавление данных и Повторяя этот процесс столько раз, сколько необходимо.

//*-*-*-*-*-----------------------------------------

### 10.6.1 Hashes Public Members

**Structures**
**Name Description**
**HASH_SUM** ( see page
206)
Context storage for a hash operation

**Функции**

**Имя Описание**

**HashAddData** (см. стр.203) Добавляет данные в хэш- суммы.

**HashAddROMData** (см.страница 203) Добавляет данные в хэш- суммы.

**MD5Calculate** (см. стр.204) Рассчитывает MD5 хэш.

**MD5Initialize** (см. стр.205) Инициализацию нового MD5-хэш.

**SHA1Calculate** (см. стр.205) Рассчитывает SHA-1 хэш.

**SHA1Initialize** (см. стр.205) Инициализацию нового SHA-1 хэш.

**модуль**

**Hashes** (см. стр. 202 )

**структуры**

**Имя Описание**

**HASH_SUM** (см. стр.206) Контекст хранения для операции хэш.

**описание**

Следующие функции и переменные доступны для применения стека.

//*-*-*-*-*----------

### 10.6.1.1 HashAddData Function

**File**
Hashes.h

**C**
```
void HashAddData(
HASH_SUM* theSum,
BYTE* data,
WORD len
);
```

**Returns**
None

**Description**
Добавляет данные в хэш- суммы.

**Remarks**
Эта функция вызывает соответствующую функцию хэширования на основе хэш типизированные определены в **theSum**.

**Preconditions**
Хэш сумма уже инициализирована

**Parameters**
**Parameters Description**
**theSum** hash context state
**data** the data to be added to the hash sum
**len** length of data

**theSum** хэш контекст государство
**data**, чтобы данные были добавлены в хэш- суммы

**len** длина данных

//*-*-*-*-*----------

### 10.6.1.2 HashAddROMData Function

**File**
Hashes.h

**C**
```
void HashAddROMData(
HASH_SUM* theSum,
ROM BYTE* data,
WORD len
);
```

**Returns**
None

**Description**
Добавляет данные в хэш- суммы.

**Remarks**
Эта функция вызывает соответствующую функцию хэширования на основе хэш типизированные определены в **theSum**.

Эта функция является псевдонимом для **HashAddData** (см. стр. 203) на не- PIC18 платформ.

**Preconditions**
Хэш сумма уже инициализирована

**Parameters**
**Parameters Description**

**theSum** hash context state
**data** the data to be added to the hash sum
**len** length of data

**theSum** хэш контекст государство

**data**, чтобы данные были добавлены в хэш- суммы

**len** длина данных

//*-*-*-*-*----------

#### 10.6.1.3 MD5Calculate Function

**File**
Hashes.h

**C**
```
void MD5Calculate(
HASH_SUM* theSum,
BYTE* result
);
```

**Returns**
None

**Description**
Эта функция вычисляет хэш- сумму всех входных данных до сих пор. Это неразрушающего к контексту хэш, поэтому больше данных может быть добавлен после вызова этой функции.

**Preconditions**
Контекст хэш был правильно инициализирован.

**Parameters**
**Parameters Description**
**theSum** текущий контекст хэша привести 16 массив байтов, в котором хранится полученный хэш

//*-*-*-*-*----------

#### 10.6.1.4 MD5Initialize Function

**File**
Hashes.h

**C**
```
void MD5Initialize(
```

HASH_SUM* **theSum**
);

**Returns**
None

**Description**
Инициализирует новый хэш MD5.

**Preconditions**
None

**Parameters**
**Parameters Description**
**theSum** - указатель на размещенного **HASH_SUM** объекта ( смотри страницу 206) объект для инициализации как MD5

//*-*-*-*-*----------

### 10.6.1.5 SHA1Calculate Function

**File**
Hashes.h

**C**
**void SHA1Calculate**(
HASH_SUM* **theSum**,
BYTE* **result**
);

**Returns**
None

**Description**
Эта функция вычисляет хэш- сумму всех входных данных до сих пор. Это неразрушающего к контексту хэш, поэтому больше данных может быть добавлен после вызова этой функции.

**Preconditions**
Контекст хэша правильно инициализирован.

**Parameters**
**Parameters Description**
**theSum** текущий контекст хэша

**result** 20 массив байтов, в котором хранится полученный хэш

//*-*-*-*-*----------

### 10.6.1.6 SHA1Initialize Function

**File**

Hashes.h

**C**
```
void SHA1Initialize(
HASH_SUM* theSum
);
```

**Returns**
None

**Description**
Инициализирует новый хэш SHA-1.

**Preconditions**
None

**Parameters**
**Parameters Description**
**theSum** - указатель на размещенную **HASH_SUM** ( см. стр. 206) объект для инициализации как SHA-1n раздел

**Section**
Function Prototypes

//*-*-*-*-*----------

### 10.6.1.7 HASH_SUM Structure

**File**
Hashes.h

**C**
```
typedef struct {
DWORD h0;
DWORD h1;
DWORD h2;
DWORD h3;
DWORD h4;
DWORD bytesSoFar;
BYTE partialBlock[64];
HASH_TYPE hashType;
} HASH_SUM;
```

**Members**
**Members Description**
DWORD h0; Hash state h0
DWORD h1; Hash state h1
DWORD h2; Hash state h2
DWORD h3; Hash state h3
DWORD h4; Hash state h4

DWORD bytesSoFar; Total number of bytes hashed so far
BYTE partialBlock[64]; Beginning of next 64 byte block
HASH_TYPE hashType; Type of hash being calculated

DWORD h0 ;

DWORD h1 ;

DWORD h2 ;

DWORD h3 ;

DWORD h4 ;

DWORD bytesSoFar ;

BYTE partialBlock [64] ;

HASH_TYPE hashType ;

} HASH_SUM ;

**Пользователи**

**Пользователи Описание**

DWORD h0 ; Хэш государственные h0

DWORD h1 ; Хэш государственные h1

DWORD h2 ; Хэш государственные h2

DWORD h3 ; Хэш государственные h3

DWORD h4 ; Хэш государственные h4

DWORD bytesSoFar ; Общее число байтов хэш сих пор

BYTE partialBlock [64] ; Начало следующего блока 64 байт

HASH_TYPE hashType ; Тип хэш вычисляется

**описание**

**Контекст хранения для операции хэш**

**10.6.2 Хэши Стек Пользователи**

**Функции**

**Имя Описание**

**MD5AddROMData** (см.страница 207 ) Добавляет данные в расчете хэш MD5.

**SHA1AddROMData** (см.страница 208) Добавление данных на SHA- 1 хэш расчета.

**SHA1AddData** (см. стр.208) Добавление данных на SHA- 1 хэш расчета.

**MD5AddData** (см. стр.209) Добавляет данные в расчете хэш MD5.

**модуль**

**Hashes** ( see page 202)

**описание**

Следующие функции и переменные являются открытыми, но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные.

**Description**
Контекстная память для операции хэша

//*-*-*-*-*----------

**Module**

**Description**
Следующее функционирует и переменные являются публикой но предполагаются, чтобы быть доступно стеком себя. Приложения должны обычно не вызывать эти функции или модифицировать эти переменные.

//*-*-*-*-*----------

### 10.6.2.1 MD5AddROMData Function

**File**
Hashes.h

**C**
```
void MD5AddROMData(
HASH_SUM* theSum,
ROM BYTE* data,
WORD len
);
```

**Returns**
None

**Description**
Добавляет данные к вычислению хэша MD5.

**Remarks**
Эта функция является псевдонимом для MD5AddData (см. стр. 209) на не- PIC18 платформ.

**Preconditions**
Контекст хэша уже инициализирован.

**Parameters**
**Parameters Description**
**theSum** указатель на структуру контекста хэш
**data** данные для добавления в хеш
**len** длина данных, чтобы добавить

//*-*-*-*-*-----------------------------------------

### 10.6.2.2 SHA1AddROMData Function

**File**
Hashes.h

**C**
```
void SHA1AddROMData(
HASH_SUM* theSum,
ROM BYTE* data,
WORD len
);
```

**Returns**
None

**Description**
Добавляет данные к вычислению хэша SHA-1.

**Remarks**
Эта функция является псевдонимом для SHA1AddData (см. стр. 208) на не- PIC18 платформ.

**Preconditions**
Контекст хэша уже инициализирован.

**Parameters**
**Parameters Description**

**theSum** указатель на структуру контекста хэш

**data** данные для добавления в хеш

**len** длина данных , чтобы добавить

//*-*-*-*-*-----------------------------------------

### 10.6.2.3 SHA1AddData Function

**File**
Hashes.h

**C**
```
void SHA1AddData(
HASH_SUM* theSum,
BYTE* data,
WORD len
);
```

**Returns**
None

**Description**
Добавляет данные к вычислению хэша SHA-1.

**Preconditions**
Контекст хэша уже инициализирован.

**Parameters**
**Parameters Description**

**theSum** указатель на структуру контекста хэш

**data** данные для добавления в хеш

**len** длина данных , чтобы добавить

//*-*-*-*-*-----------------------------------------

### 10.6.2.4 MD5AddData Function

**File**
Hashes.h

**C**
```
void MD5AddData(
HASH_SUM* theSum,
BYTE* data,
WORD len
);
```

**Returns**
None

**Description**
Добавляет данные к вычислению хэша MD5.

**Preconditions**
Контекст хэша уже инициализирован.

**Parameters**
**Parameters Description**

**theSum** указатель на структуру контекста хэш

**data** данные для добавления в хеш

**len** длина данных , чтобы добавить

//*-*-*-*-*----------

### 10.6.3 Hashes Internal Members

**HASH_TYPE** (см. стр.210)Тип хэш вычисляется

**Функции**

**Имя Описание**

**SHA1HashBlock (**см. стр.211)Рассчитывает SHA-1 хеш блока .

**MD5HashBlock** (см. стр.211)Рассчитывает MD5 хеш блока .

**модуль**

Хэши (см. стр. 202 )

**Переменные**

**Имя Описание**

**_MD5_k** ( См. стр. 210 ) массив заранее определенных значений K для MD5

**_MD5_r** (См. стр. 210 ) Массив предопределенных R долин для MD5

**lastBlock** (см. стр. 210 ) Сохраняет копию последнего блока с требуемым заполнения

**описание**

Следующие функции и переменные определяются как внутренние, к хэшей (см. стр. 202) модуля.

//*-*-*-*-*----------

#### 10.6.3.1 _MD5_k Variable

**File**
Hashes.c
**C**
ROM DWORD **_MD5_k**[64] = { 0xD76AA478, 0xE8C7B756, 0x242070DB, 0xC1BDCEEE,
0xF57C0FAF,
0x4787C62A, 0xA8304613, 0xFD469501, 0x698098D8, 0x8B44F7AF, 0xFFFF5BB1,
0x895CD7BE,
0x6B901122, 0xFD987193, 0xA679438E, 0x49B40821, 0xF61E2562, 0xC040B340, 0x265E5A51,
0xE9B6C7AA, 0xD62F105D, 0x02441453, 0xD8A1E681, 0xE7D3FBC8, 0x21E1CDE6,
0xC33707D6,
0xF4D50D87, 0x455A14ED, 0xA9E3E905, 0xFCEFA3F8, 0x676F02D9, 0x8D2A4C8A,
0xFFFA3942,
0x8771F681, 0x6D9D6122, 0xFDE5380C, 0xA4BEEA44, 0x4BDECFA9, 0xF6BB4B60,
0xBEBFBC70,
0x289B7EC6, 0xEAA127FA, 0xD4EF3085, 0x04881D05, 0xD9D4D039, 0xE6DB99E5,
0x1FA27CF8,
0xC4AC5665, 0xF4292244, 0x432AFF97, 0xAB9423A7, 0xFC93A039, 0x655B59C3,
0x8F0CCC92,
0xFFEFF47D, 0x85845DD1, 0x6FA87E4F, 0xFE2CE6E0, 0xA3014314, 0x4E0811A1,
0xF7537E82,
0xBD3AF235, 0x2AD7D2BB, 0xEB86D391 };

**Description**
Массив заранее определенных значений K для MD5

//*-*-*-*-*----------

#### 10.6.3.2 _MD5_r Variable

**File**
Hashes.c

**C**
ROM BYTE **_MD5_r**[64] = {7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 5, 9,
14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16,
23, 4, 11, 16, 23, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21};

**Description**
Массив предопределенной R Valeš для MD5

//*-*-*-*-*----------

#### 10.6.3.3 lastBlock Variable

**File**

Hashes.c

**C**
BYTE **lastBlock**[64];

**Description**
Сохраняет копию последнего блока с требуемым заполнения

//*-*-*-*-*-----------------------------------------

### 10.6.3.4 HASH_TYPE Enumeration

**File**
Hashes.h

**C**
```
typedef enum {
HASH_MD5 = 0u,
HASH_SHA1
} HASH_TYPE;
```

**Members**
**Members Description**
HASH_MD5 = 0u MD5 вычисляется,
HASH_SHA1 SHA-1 в процессе расчета

**Description**
Тип хэш вычисляется

//*-*-*-*-*-----------------------------------------

### 10.6.3.5 SHA1HashBlock Function

**File**
Hashes.c

**C**
```
static void SHA1HashBlock(
BYTE* data,
DWORD* h0,
DWORD* h1,
DWORD* h2,
DWORD* h3,
DWORD* h4
);
```

**Returns**
None

**Description**
Эта функция вычисляет SHA-1 хеш над блоком и обновляет значения h0 -H3 со следующего контексте.

**Preconditions**
Указатель данные должны быть WORD выровнены по 16-битных частей и DWORD выровненных на 32-разрядных Тихого океана. Если выравнивание не является правильным, выравнивание памяти исключение происходит.

**Parameters**
**Parameters Description**
Данные Блок 64 байт хэш

h0 текущее значение хэш контекст h0

h1 текущее значение хэш контекст h1

h2 текущее значение хэш контекст h2

h3 текущее значение хэш контекст h3

h4 текущее значение хэш контекст h4

**Section**
Функции и переменные необходимые для SHA-1

//*-*-*-*-*----------

### 10.6.3.6 MD5HashBlock Function

**File**
Hashes.c

**C**
```
static void MD5HashBlock(
BYTE* data,
DWORD* h0,
DWORD* h1,
DWORD* h2,
DWORD* h3
);
```

**Returns**
None

**Description**

Эта функция вычисляет MD5 хеш над блоком и обновляет значения h0 -H3 со следующего контексте.

**Preconditions**
Указатель данные должны быть WORD выровнены по 16-битных частей и DWORD выровненных на 32-разрядных Тихого океана. Если выравнивание не является правильным, выравнивание памяти исключение происходит.

**Parameters**
**Parameters Description**
Данные Блок 64 байт хэш

h0 текущее значение хэш контекст h0

h1 текущее значение хэш контекст h1

h2 текущее значение хэш контекст h2

h3 текущее значение хэш контекст h3

//*-*-*-*-*----------

## 10.7 Helpers

**Функции**

**Имя Описание**

**LFSRRand** ( см. стр. 228 ) Возвращает псевдослучайное 16- разрядное число без знака в диапазоне от 0 до 65535( 0x0000 до 0xFFFF ).

**LFSRSeedRand** (см. стр.228) Семена **LFSR** генератор случайных чисел вызывается функцию **LFSRRand** () (см.страница 228). До семян возвращается.

**strncpy_m** ( см. стр. 229) Копии нескольких строк в пункт назначения

**Переменные**

**Имя Описание**

**dwLFSRRandSeed** (см.страница 230 ) По умолчанию генератор случайных чисел семя. 0x41FE9F9E соответствует вызову.

**LFSRSeedRand** ( см. стр. 228 ) (1)

**описание**

Этот модуль содержит несколько вспомогательных функций, используемых во всем TCP / IP стека. Некоторые из этих дубликата функциональности уже реализованы в библиотеках компилятора по умолчанию. В тех случаях, версия компилятора используется и версии стека в опущено.

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

### 10.7.1 Помощники открытые члены

### 10.7.1 Helpers Public Members

**Functions**
**Name Description**
**Base64Decode** (см. стр.214) Декодирует массив 64 в его буквальном представительства.

**Base64Encode** (см. стр.214) Кодирует двоичный массив в Base- 64.

**btohexa_high** (см. стр.215) Преобразует верхнюю клев двоичного значения в шестнадцатеричное байта ASCII.

**btohexa_low** (см. стр.215) Преобразует нижний разряд двоичного значения в шестнадцатеричное байта ASCII.

**CalcIPChecksum** (см.страница 216 ) Вычисляет значение IP контрольной суммы.

**ExtractURLFields** (см.страница 216 ) Извлечение всех параметров из URL строки (например:

" http://admin:passwd @ www.microchip.com:8080/myfile.gif " разбивается на

{ PROTOCOL_HTTP, " admin ", " PASSWD ", " www.microchip.com ", 8080," / myfile.gif " }.

**FormatNetBIOSName** (см.страница 219 ) Форматы строку в допустимое имя NetBIOS.

**GenerateRandomDWORD**( См. стр. 220 ) Генерирует случайное DWORD.

**hexatob** (см. стр. 220 ) Преобразует шестнадцатеричную строку в один байт.

**leftRotateDWORD** (см.страница 221 ) Левый поворот DWORD.

**Replace** (см. стр. 221 ) Заменяет все экземпляры конкретной подстроки с новой строки

**ROMStringToIPAddress** (см. стр. 222) Преобразует строку в IP -адрес

**stricmppgm2ram** (см.страница 223 ) Без учета регистра сравненивает строку в оперативной памяти со строкой в ROM.

**StringToIPAddress** (см.страница 224 ) Преобразует строку в IP -адрес

**strupr** (см. стр. 224 ) Преобразует строку в верхний регистр.

**strnchr** ( см. стр. 225 ) Ищет строку до указанного числа символов для конкретной характер.

**swapl** (см. стр. 225 ) Меняет местами порядок байтов из DWORD.

**swaps** (см. стр. 226 ) Меняет местами порядок байтов в слове.

**uitoa** (см. стр. 226 ) Преобразует целое число без знака десятичную строку.

**ultoa** (см. стр. 227 ) Преобразует целое число без знака в десятичную строку.

**UnencodeURL** (см. стр.227) Декодирует URL - строку в кодировке.

**макрос**

**Имя Описание**

**leftRotateDWORD** (см.страница 221 ) Повороты являются более эффективными в C30 и C32

**ROMStringToIPAddress** (см. стр. 223) Номера -ROM вариант для C30 и C32

**модуль**

Помощники (см. стр. 212 )

**описание**

Следующие функции и переменные доступны для применения стека.

//*-*-*-*-*----------

### 10.7.1.1 Base64Decode Function

**File**
Helpers.h

**C**
WORD **Base64Decode**(
BYTE* **cSourceData**,
WORD **wSourceLen**,
BYTE* **cDestData**,
WORD **wDestLen**
);

**Returns**
Количество декодированными байтов, записанных в **cDestData**.

**Description**
Декодирует массив 64 в его буквальном представительства.

**Remarks**
Эта функция двоичные и будет игнорировать недопустимые символы ( CR, LF , и т.д. ). Если **cSourceData** равна **cDestData**, данных будут преобразованы на месте. Если **cSourceData** не равна **cDestData**, но области перекрываются, поведение не определено.

Расшифрованные данные - всегда по крайней мере 1/4 меньше чем исходные данные.

**Preconditions**
None

**Parameters**
**Parameters Description**

**cSourceData** Указатель на строку Base- 64 кодированных данных

**wSourceLen** Длина Base- 64 источника данных Максимальная длина, которая может быть записана в **cDestData**

**cDestData** Указатель написать декодированные данные

//*-*-*-*-*----------

### 10.7.1.2 Base64Encode Function

**File**
Helpers.h

**C**
WORD **Base64Encode**(
BYTE* **cSourceData**,
WORD **wSourceLen**,
BYTE* **cDestData**,
WORD **wDestLen**
);

**Returns**
Количество закодированных байтов, записанных в **cDestData**. Это всегда будет кратно 4.

**Description**
Кодирует двоичный массив в Base- 64.

**Remarks**
Кодирование не может быть выполнена на месте. Если **cSourceData** перекрывается с **cDestData**, поведение не определено.

Закодированных данных - всегда по крайней мере 1/3 больше чем исходных данных. Это может быть на 1 или 2 байта больше.

**Preconditions**
None

**Parameters**
**Parameters Description**

**cSourceData** Указатель на строку из двоичных данных

**wSourceLen** Длина двоичного источника данных. Максимальная длина, которая может быть записана **cDestData**

**cDestData** Указатель написать Base- 64 кодированных данных

//*-*-*-*-*----------

### 10.7.1.3 btohexa_high Function

**File**
Helpers.h

**C**
BYTE **btohexa_high**(
BYTE **b**
);

**Returns**
Верхняя шестнадцатеричное ASCII байт '0 ' - '9 ' или ' A' 'F'.

**Description**
Преобразует верхний разряд двоичного значения в шестнадцатеричный байт ASCII. Например, **btohexa_high** ( 0xAE ) вернет 'A'.

**Preconditions**
None

**Parameters**
**Parameters Description**
**b -** байт, для преобразовываия

//*-*-*-*-*----------

### 10.7.1.4 btohexa_low Function

**File**
Helpers.h

**C**
BYTE **btohexa_low**(
BYTE **b**
);

**Returns**
Нижняя шестнадцатеричное ASCII байт '0 ' - '9 ' или ' A' 'F'.

**Description**
Преобразует нижний разряд двоичного значения в шестнадцатеричный байт ASCII. Например, **btohexa_high** (см. стр.215) ( 0xAE ) вернет ' E'.

**Preconditions**
None

**Parameters**
**Parameters Description**
**b**- байт, чтобы преобразовываться

//*-*-*-*-*----------

### 10.7.1.5 CalcIPChecksum Function

**File**
Helpers.h

**C**
WORD **CalcIPChecksum**(
BYTE* **buffer**,
WORD **len**
);

**Returns**
Рассчитанная контрольная сумма.

**Description**
Эта функция вычисляет IP контрольную сумму массива входных данных. Контрольная сумма представляет собой 16- битное дополнение по своему дополнить сумму всех слов в данных ( с нулями, если нечетное число байтов суммируются ). Это контрольная сумма определено в RFC 793.

**Internal**
Эта функция может быть улучшена, чтобы сделать 32-разрядные средства на PIC32 платформ.

**Preconditions**
буфер WORD выровнены (даже адрес памяти ) на 16 - и 32 - бите PICs.

**Parameters**
**Parameters Description**

**buffer** указатель на буфер данных, с контрольной суммой

**count** количество байтов, которые будут контрольной суммой

//*-*-*-*-*----------

### 10.7.1.6 ExtractURLFields Function

**File**
Helpers.h

**C**
BYTE **ExtractURLFields**(
BYTE * **vURL**,
PROTOCOLS * **protocol**,
BYTE * **vUsername**,
WORD * **wUsernameLen**,
BYTE * **vPassword**,
WORD * **wPasswordLen**,
BYTE * **vHostname**,
WORD * **wHostnameLen**,
WORD * **wPort**,
BYTE * **vFilePath**,
WORD * **wFilePathLen**
);

**Returns**
Ноль - успех. Отличное значение от нуля указывает код ошибки. Если ненулевой код ошибки возвращается, ни один из возвращенных буферов или значения указателя должны рассматриваться как действительные, но некоторые из них, возможно, были записаны в. Ниже приведены все возможное возвращение значения.

**0** нет ошибок

**1** Протокол неизвестно (дополнительный код должен быть добавлен в **ExtractURLFields** ( ) и перечисление ПРОТОКОЛЫ должен быть обновляться, если вы хотите, чтобы декодировать URL-адреса этого типа протокола.

**2** URL сформирован. Незаконное или неизвестный формат URL столкнулись.

**3** Буфер слишком мал. Один из размеров входной буфер слишком мал, чтобы содержать параметр URL.

**описание**

Извлечение всех параметров из URL строки (например: " http://admin:passwd @ www.microchip.com:8080/myfile.gif " разбивается на { PROTOCOL_HTTP , " админ ", " PASSWD ", " www.microchip.com " , 8080 , " / myfile.gif " } .

URL строка может быть нулем, или в качестве альтернативы может быть прекращено возврата каретки или перевода строки.

Если протокол не распознан или протокол признается но URL имеет неверный формат, чем ошибка благополучно вернулся. для Более подробную информацию о URL / URI интерпретации см. RFC 2396.

**Предпосылки**

Эта функция по умолчанию закомментирована, чтобы экономить кодовое пространство, потому что она не используется никакими текущими функциями стека.

Однако, если вы хотите использовать её, раскомментируйте ее. Она была протестирована, так что ( следовательно ) работает правильно.

**Parameters**
**Parameters Description**
**vURL** Указатель на оканчивающихся нулем URL для декодирования и извлечь из. Этот параметр требуется, и должен иметь минимальный RFC 1738 компонентов в нем ( протокола и имя хоста ).

Протокол Дополнительный указатель на PROTOCOLS перечисления для получения декодированного тип протокола.

Если этот параметр не нужен, укажите указатель NULL. Протокол является требуется часть URL, поэтому он всегда должен присутствовать. Протокол также определяет, что схема все остальные параметры декодируется с использованием, так Функция будет выполнена, если непризнанным протокол предоставляется. Протоколы Члены перечисления показать все поддерживаемые в настоящее время протоколы для этой функции.

Для примера URL предоставленной в функции описания, PROTOCOL_HTTP будут возвращены для этого поля.

**vUsername** Дополнительный указатель на буфер, чтобы написать декодированного имя пользователя часть URL.

Если URL не содержит имя пользователя или указатель NULL поставляется, то это поле игнорируется.

Для примера URL предоставленной в описании функции, "администратора" будет вернулся для этого поля.

**wUsernameLen** На звонок: Дополнительный указатель на WORD указанием максимальную длину

**vUsername** буфер, включая нулевой символ характера.

По возвращении: Если **wUsernameLen** и **vUsername** не являются NULL,

*** wUsernameLen** СЛОВО обновляется с фактическим количеством символов записывается в буфер **vUsername**, включая нулевой символ характера. если **vUsername** является NULL, но **wUsernameLen** не является NULL, то никакие символы не являются скопированы, но *** wUsernameLen** вернет количество символов, чтобы соответствовать полный имя пользователя строкой. Если **wUsernameLen** является NULL, то поле имя пользователя в URL, если он присутствует, игнорируется и указатель **vUsername** не используется.

Если ноль символов были написаны, это означает, что URL не содержит Пользователь Поле. Если один символ был написан это означает, что поле имя пользователя присутствовал, но был строкой нулевой символ (например: "").

Для примера URL, представленной в описании функции, 6 ( 0x0006 ), будет вернулся для этого поля.

**vPassword** Дополнительный указатель на буфер, чтобы написать декодированную пароль часть URL. если URL не содержит пароль или указатель NULL поставляется, то это поле игнорируется.

Для примера URL предоставленной в описании функции, " PASSWD " будет вернулся для этого поля.

**wPasswordLen** На звонок: Дополнительный указатель на WORD указанием максимальную длину

**vPassword** буфер, включая нулевой символ характера.

По возвращении: Если **wPasswordLen** и **vPassword** не являются NULL,

*** wPasswordLen** WORD обновляется с фактическим количеством символов, написанных в буфер **vPassword**, включая нулевой символ характера. Если **vPassword** является NULL но **wPasswordLen** не является NULL, то никакие символы не копируются, но *** wPasswordLen** вернет количество символов, чтобы соответствовать в полной мере Строке пароля. Если **wPasswordLen** является NULL, то поле пароля в URL, если он присутствует, игнорируется и указатель **vPassword** не используется.

Если ноль символов были написаны, это означает, что URL не содержит поля пароля. Если один символ был написан, это означает, что поле пароля присутствует, но был строкой нулевой символ (например: "").

Для примера URL, представленной в описании функции, 7 ( 0x0007 ), будет вернулся для этого поля.

**vHostname** Дополнительный указатель на буфер, чтобы написать декодированного хоста часть URL.

Все интернет URL-адреса должны содержать имя хоста или IP- адрес, однако, если NULL указатель поставляется, то это поле игнорируется.

Для примера URL, представленной в описании функции, " www.microchip.com " будут возвращены для этого поля. Если URL был " http://192.168.0.1 ", то это поле будет возвращен как " 192.168.0.1 ". IP- адрес не будет расшифрован в DWORD ( использовать **StringToIPAddress** (см. стр. 224) ( ) вспомогательную функцию для этого ).

**wHostnameLen** На звонок: Дополнительный указатель к слову указанием максимальную длину **vHostname** буфер, включая нулевой символ характера.

По возвращении: Если **wHostnameLen** и **vHostname** не являются NULL,

* **wHostnameLen** WORD обновляется с фактическим количеством символов, написанных в буфер **vHostname**, включая нулевой символ характера. Если **vHostname** является NULL, но **wHostnameLen** не без NULL, то никакие символы копируются , но

* **wHostnameLen** вернет количество символов, чтобы соответствовать в полной мере Имя хоста строка. Если **wHostnameLen** является NULL, то поле Имя хоста в URL, игнорируется и указатель **vHostname** не используется.

Для примера URL предоставленной в функции описания , 18 ( 0x0012 ) будет быть возвращены для этой области. Если URL был " http://192.168.0.1 ", то это поле будут возвращены как 12 ( 0x000C ).

**wPort** Дополнительный указатель на WORD указанием TCP или UDP порт, что сервер прослушивает. Если поле порт отсутствует в URL, то этот параметр будет указать порт по умолчанию для протокола. Например, " http://www.microchip.com " приведет к 80, являющегося возвращении в качестве указанного порта.

Если указатель **wPort** является NULL, то поле порт в URL игнорируется, если он присутствует.

**vFilePath** Дополнительный указатель на буфер, чтобы написать декодированную путь к файлу часть URL. Если NULL указатель поставляется, то это поле игнорируется. Если путь к файлу не присутствует в URL, то "/" будет возвращен в этой области.

Для примера URL предоставленной в описании функции " / myfile.gif " будет вернулся для этого поля.

**wFilePathLen** На звонок: Дополнительный указатель на WORD указанием максимальной длины.

**vFilePath** буфер, включая нулевой символ характера.

По возвращении: Если **wFilePathLen** и **vFilePath** не являются NULL,

* **wFilePathLen** WORD обновляется с фактическим количеством символов, записанных в **vFilePath** буфер , включая нулевой символ характера. Если **vFilePath** является NULL, но

**wFilePathLen** не является NULL, то никакие символы не копируются, но

* **wFilePathLen** вернет количество символов, чтобы соответствовать файл в строку пути. если **wFilePathLen** является NULL, то поле путь к файлу в URL, если он присутствует, игнорируется и указатель **vFilePath** не используется.

Эта функция всегда возвращает " / ", если путь к файлу не присутствует, поэтому * **wFilePathLen** будет также быть не менее 2 символов ( ' / ' и нулевое окончание ), если указатель не является NULL.

Для примера URL предоставленной в описании функции, 12 ( 0x000C ) будет быть возвращены для этой области.

//*-*-*-*-*-----------------------------------------

### 10.7.1.7 FormatNetBIOSName Function

**File**
Helpers.h

**C**
```
void FormatNetBIOSName(
BYTE Name[16]
);
```

**Returns**
None

**Description**
Эта функция форматирует строку в допустимое имя **NetBIOS**. Имена будет ровно 16 символов, как определено в **NetBIOS** спецификация 16-й персонаж будет 0x00 байт, в то время как другие 15 будут предоставленная строка, пробелами в случае необходимости.

**Preconditions**
None

**Parameters**
**Parameters Description**
Назовите строку форматирования как имя **NetBIOS**. Этот параметр должен иметь по крайней мере 16 байтов, выделенных.

//*-*-*-*-*-----------------------------------------

### 10.7.1.8 GenerateRandomDWORD Function

**File**
Helpers.h

**C**

DWORD **GenerateRandomDWORD**();

**Побочные эффекты**

Эта функция использует A/D конвертер ( и поэтому вы должны отключить прерывания, если вы используете A/D преобразуется в ISR ).

функция **LFSRRand**( )( см. стр. 228 ) будет повторно заполнить и **Timer0** ( PIC18 ) и **Timer1** будет использоваться ( PIC24 , КЛУБА , и PIC32 ). TMR # H : ПМР # L будет иметь новое значение. Обратите внимание, что это то же самое таймер используется модулем **Tick**.

**Returns**
Случайное 32-разрядное число.

**Description**
Эта функция генерирует случайное 32-разрядное целое. Он собирает случайности путем сравнения A/D преобразователя внутренний R/C тактовый генератор с нашим главным системными часами. Передавая собранную энтропию к **LFSRSeedRand**( ) (см. стр.228 ) / функции **LFSRRand** ( ) ( см. стр. 228 ), выход нормализуется ( Компенсация сдвига ) в надежде встретить статистическая случайность тесты.

**Замечания**

Раз Эта функция после 1 секунды в попытке сгенерировать случайный DWORD. В таком случае, выход может быть действительно случайный. Как правило, эта функция выполняет примерно 500 000 циклов команд.

Цель этой функции является создание статистически случайное и криптографически безопасного случайного числа. Будь или не это верно на все ( или любой ) устройства / напряжения / температуры не проверяется.

**Preconditions**
None

//*-*-*-*-*----------

### 10.7.1.9 hexatob Function

**File**
Helpers.h

**C**
BYTE **hexatob**(
WORD_VAL **AsciiChars**
);

**Returns**

В результате упакованы байт 0x00 - 0xFF.

**Description**
Преобразует ASCII шестнадцатеричную строку из двух символов в один упакованный байт.

**Preconditions**
None

**Parameters**
**Parameters Description**
**AsciiChars** WORD_VAL где. V [0] это значение ASCII для нижнего клев и. V [1] Значение ASCII для верхней клев. Каждый должен находиться в диапазоне от '0 ' - '9', 'A' 'F', или 'a'-' e'.

//*-*-*-*-*----------

### 10.7.1.10 leftRotateDWORD Function

**File**
Helpers.h

**C**
DWORD **leftRotateDWORD**(
DWORD **val**,
BYTE **bits**
);

**Returns**
Вращавшее величину DWORD.

**Description**
Эта функция вращает биты в 32-разрядной DWORD оставленной определенного количества битов.

**Remarks**
Эта функция реализована только на 8 -битных платформах в настоящее время. 8-битовые компиляторы создают чрезмерный код для этой функции, в то время как C30 и C32 уже генерируют компактный код. Эти компиляторы обслуживаются макросами, определенными в Helpers.h.

**Preconditions**
None

**Parameters**
**Parameters Description**
**Val DWORD** для поворота биты количество битов, с помощью которого можно сдвинуть

//*-*-*-*-*----------

#### 10.7.1.11 leftRotateDWORD Macro

**File**
Helpers.h

**C**
```
#define leftRotateDWORD(x, n) (((x) << (n)) | ((x) >> (32-(n))))
```

**Description**
Вращения более эффективные в C30 и C32

```
//*-*-*-*-*-----------------------------------------
```

#### 10.7.1.12 Replace Function

**File**
Helpers.h

**C**
```
SHORT Replace(
BYTE * vExpression,
ROM BYTE * vFind,
ROM BYTE * vReplacement,
WORD wMaxLen,
BOOL bSearchCaseInsensitive
);
```

**Returns**
Если нуль или больше, указывает на счета, сколько замены были сделаны. Если меньше нуля ( отрицательный результат ), указывает что **wMaxLen** была слишком мала, чтобы сделать необходимые замены. В этом случае, не было сделано никаких замен.

**Description**
Поиск строку ( **vExpression** ) и заменяет все экземпляры конкретной подстроки ( **vFind** ) с новой строки ( **vReplacement** ). Смещение пуска, чтобы быть поиск и максимальное количество замещений может быть указан. поиск может быть выполнена в случае чувствительной или без учета регистра образом.

**Замечания**

Если длина замена строка короче, чем или равно длине строки поиска и строка поиска происходит в нескольких перекрывающиеся места (например: выражение " aaa ", найти "aa", и заменить "bb"), то первая находка матч, возникающая при поиск слева направо будут заменены. (например: выход выражение будет "bba") .

Однако, если длина замена строка длиннее, чем длина строки поиска, поиск будет происходить, начиная с конца строки и перейдите к началу (справа налево поиска ). В этом случае, если выражение было " aaa ", находка была " aa ", и замена была " bbb ", то выражение конечный результат будет " abbb ".

**Предпосылки**

Эта функция по умолчанию закомментирована, чтобы экономить кодовое пространство, потому что она не используется никакими текущих функций стека.

Однако, если вы хотите использовать его, идти вперед и раскомментировать ее. Он был протестирован, так что работает правильно.

**Параметры**

**Параметры Описание**

**vExpression** Null прекращается строку для поиска и сделать замены в.

**vFind** Null завершающим нулем для поиска.

**vReplacement** Null завершающим нулем, чтобы заменить все экземпляры **vFind**.c.

**wMaxLen** Максимальная длина выходного **vExpression** строки, если раскрытие строки собирается происходят (длина замена длиннее найти длину ). Если замены будет причиной этого максимальная длина строки должна быть превышена, то никакие замены не будет сделаны и отрицательный результат будет возвращен, что указывает на недостаточность. Если Длина замена короче или равна длине поиска, то этот параметр игнорируется.

**bSearchCaseInsensitive** Логическое значение, указывающее, если поиск должен быть выполнен в случае нечувствительной образом. Укажите **ИСТИНА** для чувствительно к регистру поисках ( медленнее) или **ЛОЖЬ** для чувствительны к регистру поиск (быстрее).

//*-*-*-*-*----------

### 10.7.1.13 ROMStringToIPAddress Function

**File**
Helpers.h

**C**
```
BOOL ROMStringToIPAddress(
ROM BYTE* str,
IP_ADDR* IPAddress
);
```

**Description**
Эта функция анализирует точечной нотации десятичную строку IP адреса в структуре **ip_addr**. В результате выход от старшего к младшему.

**Remarks**
Эта функция является псевдонимом для **StringToIPAddress** (см. стр. 224) на не- PIC18 платформ.

**Preconditions**
None

**Parameters**
**Parameters Description**
**str** Указатель на точечной нотации IP адреса строки IP-адрес Указатель на

**ip_addr-** в котором для храненится результат

**Return Values**
**Return Values Description**
**ИСТИНА** IP -адрес был успешно распознан

**ЛОЖЬ** нет IP- адрес не может быть найден, или формат был неправильным

//*-*-*-*-*----------

### 10.7.1.14 ROMStringToIPAddress Macro

**File**
Helpers.h

**C**
```
#define ROMStringToIPAddress(a,b) StringToIPAddress((BYTE*)a,b)
```
**Description**
Non-ROM variant for C30 and C32

//*-*-*-*-*----------

### 10.7.1.15 stricmppgm2ram Function

**File**
Helpers.h

**C**
```
signed char stricmppgm2ram(
BYTE* a,
ROM BYTE* b
);
```

**Description**

Выполняет сравнение без учета регистра строки в оперативной памяти в строку в ROM. Эта функция выполняет идентично **strcmppgm2ram**, за исключением того, что сравнение не зависит от регистра.

**Preconditions**
None

**Parameters**
**Parameters Description**
**a** Pinter to tring in RAM
**b** Указатель на строку в ROM

**Return Values**
**Return Values Description**
-1 a < b
0 a = b
1 a > b

//*-*-*-*-*----------

### 10.7.1.16 StringToIPAddress Function

**File**
Helpers.h

**C**
BOOL **StringToIPAddress**(
BYTE* **str**,
IP_ADDR* **IPAddress**
);

**Description**
Эта функция анализирует точечной нотации десятичную IP адрес строку в структуре **ip_addr**. В результате выход от старшего к младшему.

**Preconditions**
None

**Parameters**
**Parameters Description**
**str** Указатель на точечной нотации IP адреса строки IP-адрес Указатель на **ip_addr**, в котором хранениться результат

**Return Values**
**Return Values Description**
**TRUE** - адрес был успешно распознан

**FALSE** - адрес не может быть найден, или формат был неправильным

//*-*-*-*-*----------

### 10.7.1.17 strupr Function

**File**
Helpers.h

**C**
```
char * strupr(
char* s
);
```

**Returns**
Указатель на начальной строке.

**Description**
Эта функция преобразует строки в верхний регистр на платформах, которые еще не имеют этой функции определены. Все строчные символы преобразуются, А.Н. символов, не включенных в ' A' 'Z' осталось как есть.

**Preconditions**
None

**Parameters**
**Parameters Description**
**s** - с нулевым символом в конце должны быть преобразованы.

//*-*-*-*-*----------

### 10.7.1.18 strnchr Function

**File**
Helpers.h

**C**
```
char * strnchr(
const char * searchString,
size_t count,
char c
);
```

**Возвращает**

Указатель на первом вхождению символа **c** в строке **SearchString**. Если символ не найден или достигается максимальное значение счетчика, возвращается указатель NULL.

**описание**

Поиск строку до указанного числа символов для конкретного характера. Строка поиск вперед и первый происшествием расположение возвращается. Если поиск символ не присутствует в строке, или, если достигли максимальное количество символов, а затем возвращается указатель NULL.

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**SearchString** Указатель на строку завершенная строка для поиска. Если ранее, чем размер строки, то строка не должна быть нулем.

рассчитывать Максимальное количество символов для поиска до отмены.

**s**- Персонаж искать

//*-*-*-*-*-----------------------------------------

### 10.7.1.19 swapl Function

**File**
Helpers.h

**C**
DWORD **swapl**(
DWORD **v**
);

**Returns**
Подкачанная версия v.

**Description**
Подкачивает endian- ness DWORD.

**Preconditions**
None

**Parameters**
**Parameters Description**
DWORD, чтобы подкачиваться

//*-*-*-*-*-----------------------------------------

### 10.7.1.20 swaps Function

**File**
Helpers.h

**C**
WORD **swaps**(
WORD **v**
);

**Returns**
Местами версия ст

**Description**
Меняет местами порядка байтов в слове.

**Preconditions**
None

**Parameters**
**Parameters Description**
WORD чтобы поменять
//*-*-*-*-*-----------------------------------------

### 10.7.1.21 uitoa Function

**File**
Helpers.h

**C**
**void uitoa**(
WORD **Value**,
BYTE* **Buffer**
);

**Returns**
None

**Description**
Преобразует 16-разрядное целое число без знака с нулем в конце десятичной строки.

**Preconditions**
None

**Parameters**
**Parameters Description**
**Value** - число которое нужно преобразовать

**Buffer** - указатель, в котором хранится преобразованная строка

//*-*-*-*-*----------

### 10.7.1.22 ultoa Function

**File**
Helpers.h

**C**
```
void ultoa(
DWORD Value,
BYTE* Buffer
);
```

**Returns**
None
**********
HI-TECH PICC -18 PRO 9.63 , C30 v3.25 и C32 v1.12 уже есть ultoa ( функция ) библиотека C18 уже имеет ultoa ( ) функция, которая более или менее матчи этот C32 < 1,12 и C30 < v3.25 нужна эта функция

**описание**

C32 < 1,12 и C30 < v3.25 нужен этот 2 параметра стека реализован функцию

Преобразует 32-разрядное целое число без знака с нулем в конце десятичной строки.

**Preconditions**
None

**Parameters**
**Parameters Description**
Параметры Описание

**Value** - число которое нужно преобразовать

**Buffer** - указатель, в котором хранится преобразованная строка

//*-*-*-*-*----------

### 10.7.1.23 UnencodeURL Function

**File**
Helpers.h

**C**

**void UnencodeURL(**
BYTE* **URL**
);

**Returns**
None

**описание**

Эта функция устарела для использования с HTTP Классик исключением. Она пытается декодировать URL строку в кодировке, превращая все шестигранные управляющие последовательности в буквальном байта. Однако это неэффективно в течение длинных строк и не обрабатывает URL - закодированные данные строки (' & ' и ' =').

**Preconditions**
None

**Parameters**
**Parameters Description**
URL на строку с завершающим нулем для декодирования

//*-*-*-*-*----------

### 10.7.2 Functions

**Functions**
**Name Description**
**LFSRRand** ( см. стр. 228 ) Возвращает псевдослучайное 16- разрядное число без знака в диапазоне от 0 до 65535 ( 0x0000 до 0xFFFF ).

**LFSRSeedRand** (см. стр.228) Семена **LFSR** генератор случайных чисел вызывается LFSRRand (см. страница 228) () функцию. До семян возвращается.

**strncpy_m** ( см. стр. 229) Копии нескольких строк в пункт назначения

**Module**
Helpers ( see page 212)

//*-*-*-*-*----------

#### 10.7.2.1 LFSRRand Function

**File**

Helpers.h

**C**
WORD **LFSRRand**();

**Side Effects**
Внутреннее семя LFSR скорректировано чтобы следующий вызов на **LFSRRand**(), возвращал другое произвольное число.

**Returns**
Случайное 16- разрядное беззнаковое целое число.

**Description**
Возвращает псевдослучайных 16- разрядное число без знака в диапазоне от 0 до 65535 ( 0x0000 до 0xFFFF ). Случайное число генерируется с использованием линейной обратной связью регистра сдвига ( LFSR ) тип псевдослучайных алгоритм число генератора. LFSR может быть посеяны, вызвав функцию **LFSRSeedRand**( ) (см. стр. 228 ) для создания такого же последовательность случайных чисел как предварительного строки звонков. Внутренняя **LFSR** повторит через 2 ^ 32-1 итераций.

**Remarks**
None

**Preconditions**
None

//*-*-*-*-*----------

### 10.7.2.2 LFSRSeedRand Function

**File**
Helpers.h

**C**
DWORD **LFSRSeedRand**(
DWORD **dwSeed**
);

**Side Effects**
None

**Returns**
Последний семян в использовании. Это могут быть сохранены и восстановлены последующим вызовом **LFSRSeedRand** ( ), если вы хотите использовать **LFSRRand**() ( см. стр. 228 ) в нескольких контекстах, не нарушая последовательность случайных чисел от альтернативы контекст. Например, если приложение 1 нуждается в заданной последовательности случайных чисел, чтобы выполнить тест, если вы сохранить и восстановить семян в App 2, это возможно для App 2 не нарушить случайную

последовательность чисел, предоставленную App 1, даже если число раз App 2 вызывает **LFSRRand** ( ) ( см. стр. 228 ) изменяется.

**Description**
Засевает произвольного генератора числа **LFSR** введенного функции **LFSRRand**() ( смотри страницу 228). Предшествующее семя возвращано.

**Remarks**
При первоначальном включении питания, внутреннее семя инициализируется 0x1. Использование **dwSeed** значение 0x0 вернется и ту же последовательность случайных чисел, как с помощью семя 0x1.

**Preconditions**
None

**Parameters**
**Parameters Description**
**dwSeed** Новый 32-битное значение семян для назначения LFSR.

//*-*-*-*-*----------

### 10.7.2.3 strncpy_m Function

**File**
Helpers.h

**C**
```
size_t strncpy_m(
char* destStr,
size_t destSize,
int nStrings,
...
);
```

**Returns**
Длина строки назначения, заканчивающийся 0 (если есть), не включенные Копии нескольких строк в пункт назначения, но не копирует более **destSize** символов. Полезная где назначение на самом деле массив и дополнительный 0 не будет добавлен к переполнить буфер

**Предпосылки**

• правильными указателями строк

• **destSize** должно быть > 0

**Параметры**

**Параметры Описание**

**destStr** Указатель на строку для инициализации с несколькими строками, предоставляемых в качестве

//*-*-*-*-*----------

### 10.7.3 Variables

**Module**
Helpers ( see page 212)

**Variables**
**Name Description**
**dwLFSRRandSeed** (см.страница 230 ) По умолчанию генератор случайных чисел семя. 0x41FE9F9E соответствует вызову **LFSRSeedRand** ( см. стр. 228 ) (1)

//*-*-*-*-*----------

#### 10.7.3.1 dwLFSRRandSeed Variable

**File**
Helpers.c

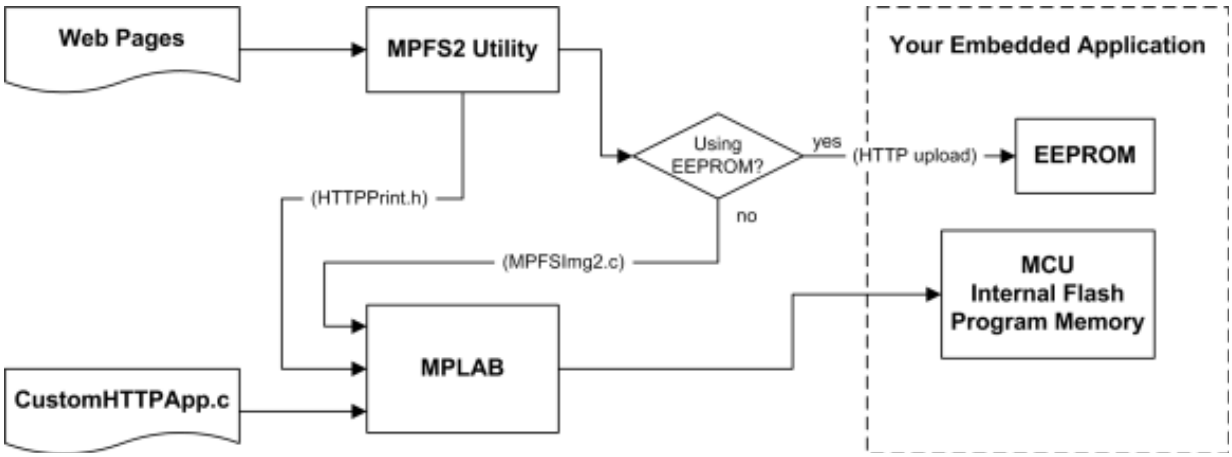
**C**
DWORD **dwLFSRRandSeed** = 0x41FE9F9E;

**Description**
По умолчанию генератор случайных чисел семя. 0x41FE9F9E соответствует вызову **LFSRSeedRand** (см. стр. 228 ) (1)

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

## 10.8 Сервер HTTP2

Модуль http2 веб-сервер и связанный с ним модуль MPFS2 файловая система позволит плате работать в качестве веб-сервера. это облегчает простой способ для просмотра состояния информационные и управляющие приложения с помощью любого стандартного веб-браузера.
Три основные компоненты необходимы, чтобы понять, как работает веб-сервер http2: веб-страницы, MPFS2 Утилита, и исходные файлы CustomHTTPApp.c и HTTPPrint.h.
Обзор всего процесса показан ниже.




**веб-страницы**
Они включают в себя все HTML и связанные с ними изображения, CSS стили и файлы JavaScript, необходимые для отображения веб-сайта.
Пример приложения в том числе всех этих компонентов находится в папке **WebPages2** .

**Утилита MPFS2**

Эта программа, поставляемые Microchip, пакеты веб-страницы в формате, который может эффективно храниться в любой внешней энергонезависимой памяти, или во внутренней флэш -памяти программ.
Эта программа также индексирует динамические переменные обнаруженные на веб страницах и корректирует **HTTPPrint**.h с этими индексами.
Если внешняя память используется, Утилита MPFS2 выполняет файл BIN и может загрузить этот файл непосредственно на плату. Если данные загружается в настоящее время во флэш-память программ, Утилита MPFS2 сгенерирует исходный файловый образ на C, который нужно включить в проект.

Когда динамические переменные добавлены или удалены из вашего приложения, Утилита MPFS2 скорректирует **HTTPPrint**.h. Когда это случается, проект должен быть перекомпилирован в MPLAB IDE, чтобы проверить, что все новые переменные индексы добавились в приложение.

**CustomHTTPApp.c**

Этот файл осуществляет приложение веб. Это описывает выход для динамических переменных возвратов (через **HTTPPrint_varname** ( смотри страницу 246)), данные грамматических разборов поданные через формы (в **HTTPExecuteGet** ( смотри страницу 242) и **HTTPExecutePost** ( смотри страницу 243)) и подтверждайте верительные грамоты разрешения (в **HTTPAuthenticate**). Точное функциональное назначение этих возвратов описано в пределах демонстрационных прикладных страниц веб и также подтверждено в пределах **CustomHTTPApp**.c пример, который распространен стеком.

**HTTPPrint**.h

Этот файл генерируется автоматически Утилитой MPFS2. Он индексирует все динамические переменные и обеспечивает "клей" между переменными расположенными на веб страницах

и их **HTTPPrint_varname** связанного возврата ( смотри страницу 246) функции определялись в **CustomHTTPApp**.c. Этот файл не требует модификации программистом.

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

### 10.8.1 Модуль Характеристик HTTP2

**Module**
HTTP2 Server ( see page 230)

**Описание**

Модуль веб-сервер http2 имеет много возможностей. В следующих разделах будут расмотрены эти функции и приведены примеры.

#### 10.8.1.1 http2 динамические переменные

Один из самых основных потребностей является предоставление информации о состоянии вашего веб-приложения обратно пользователю. Сервер HTTP предоставляет эту возможность с использованием динамических переменных обратные вызовы замещения. Эти команды в вашем HTML коде будет предупреждать, чтобы сервер выполнил функцию обратного вызова в тот момент, который разработчик создает для записи данных в веб-страницы. Динамические переменные следует рассматривать выход вашего приложения.

**Основы использования**
Чтобы создать динамическую переменную, просто заключите имя переменной внутри пары тильды (~) символы внутри сети HTML исходный код страниц. (например: ~MYVARIABLE~) При запуске MPFS2 утилита для генерации веб-страниц, он будет автоматически индексировать эти переменные в **HTTPPrint**.h. Этот индекс будет инструктировать приложение для вызова функции **HTTPPrint_myVariable** когда эта строка встречается.
Вот пример использования динамической переменной, чтобы вставить дату сборки вашего приложения в веб-страницы:

```
<div class="examplebox code">~builddate~</div>
```

Соответствующий обратного вызова будет выводить значение в веб-страницы:

```
void HTTPPrint_builddate(void)
{
TCPPutROMString(sktHTTP,(ROM void*)__DATE__);
}
```

**Прохождение Параметров**

Вы также можете передавать параметры динамических переменных, помещая числовые значения внутри скобок после имени переменной.

Например, ~ LED (2) ~ выведет значение второго светодиода. Числовые значения передаются как значения слово в функция обратного вызова. Вы можете передать столько параметров, сколько вы хотите этих функций, и если ваш C код имеет константы определяется, тем, будет разбираться, а также. **(ex: ~pair(3,TRUE)~)**

Следующий код вставляет значение из кнопок в веб-страницы, все с помощью того же функцию обратного вызова:

<div **class**="examplebox code">btn(3)~ btn(2)~ btn(1)~ btn(0)~

Этот связанный возврат напечатает величину запрошенной кнопки на странице веб:

```
void HTTPPrint_btn(WORD num)
{
// Determine which button
switch(num)
{
case 0:
num = BUTTON0_IO;
break;
case 1:
num = BUTTON1_IO;
break;
case 2:
num = BUTTON2_IO;
break;
case 3:
num = BUTTON3_IO;
break;
default:
num = 0;
}
// Print the output
if(num == 1)
TCPPutROMString(sktHTTP, "up");
else
TCPPutROMString(sktHTTP, "down");
}
```

**Более длинные выходы**

Протокол HTTP работает в фиксированном буфере памяти для передачи, поэтому не все данные могут быть отправлены одновременно. Уход должен быть принят внутри вашей функции обратного вызова, чтобы избежать переполнения буфера.
Веб-сервер http2 проверяет, что в этом буфере по крайней мере 16 байт свободны перед вызовом функции обратного вызова. Для коротких выходов ( менее чем 16 байт ), обратные вызовы нужно только вызвать соответствующюю функцию возвращения **TCPPut** (см. стр.

545 ). Для более длинных выходов, обратного вызова функции должны проверить, сколько свободного места осталось, напишите до этого много байт, а затем вернуться. Обратный вызов будет вызывается снова, когда много свободного места.
Для управления состояние выхода, обратные вызовы должны использовать **curHTTP**.**callbackPos**. Это значение **DWORD** установлено равным нулю когда функция обратного вызова вызывается в первый раз. Если обратный вызов только пишу часть своей продукции, она должна установить это поле в ненулевым ценность для указать, что он должен снова вызывается, когда много свободного места. Это значение будет доступно для обратного вызова во время следующего вызова, который позволяет функции возобновить выход, где она была прервана. Обычное использование для хранения количество записанных байтов, или оставшееся для записи, в этой области. После того, как функция обратного вызова закончит писать свой выход, она должен установить **curHTTP**.**callbackPos** обратно к нулю, чтобы указать завершение.
В качестве примера, этот код выводит текущее значение на ЖК-дисплее, который отбражает 32 байта на многих платах Microchip:

<div **class**="examplebox code">~lcdtext~

Следующая функция возврата оперирует выход и управляет своим состоянием для многочисленных вызовов:

```
void HTTPPrint_lcdtext(void)
{
WORD len;
// Определите сколько байтов, который мы можем записать
len = TCPIsPutReady(sktHTTP);
// Если просто запуская, установившее callbackPos
if (curHTTP.callbackPos == 0)
curHTTP.callbackPos = 32;
// Запись байт за один раз пока мы все еще можем
// Это может занять до 12 байтов, чтобы записывать символ
 // (пробелы и концы строки более длинные)
while (len > 12 && curHTTP.callbackPos)
{
// После 16 записей байтов конец строки
if (curHTTP.callbackPos == 16)
len -= TCPPutROMString(sktHTTP, (ROM BYTE*)"<br />");

if (LCDText[32-curHTTP.callbackPos] == ' ' || LCDText[32-curHTTP.callbackPos] ==
'\0')
len -= TCPPutROMString(sktHTTP, (ROM BYTE*)" ");
else
len -= TCPPut(sktHTTP, LCDText[32-curHTTP.callbackPos]);
curHTTP.callbackPos--;
}
}
```

Первоначальный вызов **TCPIsPutReady** (см. стр. 541) определяет, сколько байт можно записать в буфер прямо сейчас.
**TCPPut** (см. стр. 545) функции возвращают количество записанных байтов, поэтому мы можем вычесть это значение из **len** отслеживать сколько буферного пространства осталось.

Когда буферное пространство будет исчерпано, функция выходит и ждет, чтобы быть вызвана снова. для последующие вызовы, стоимость **curHTTP**.**callbackPos** точно как мы оставили его. Функция возобновляет свой выход в этой точке.

**Включение файлов**

Часто это полезно включить все содержимое другого файла в вашем выводе. Большинство веб-страниц, по крайней мере некоторую часть, что не изменяется, например, заголовок, меню ссылок и подпись. Эти разделы можно абстрагировать в отдельные файлы что делает их легче в управлении и экономит место при хранении.
Чтобы включить все содержимое другого файла, используйте динамическую переменную, которая начинается с " **inc**: ", например, ~ **Inc: header.inc** ~.
Эта последовательность вызовет файл **header.inc** для чтения из файловой системы и вставит в данном месте.
В следующем примере показано, как включить стандартное меню **бар** в каждой странице:

** ~inc:menu.inc~ **

В это время, динамические переменные не являются рекурсивными, поэтому любые переменные, находящиеся внутри файлов, включенных в этой манере не анализируются.

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

**10.8.1.2 http2 Обработка форм**

Многие приложения должны принять (см. стр. 169 ) данные от пользователя. Обычным решением является представление формы пользователю в веб-странице, то есть значения технологического устройства, представленные через эту форму. Веб-формы, как правило, представлены с помощью одного из двух метододов (GET и POST), и веб-сервера.

**Метод GET**

Метод **GET** добавляет данные в конце URI. Эти данные следует знак вопроса (? ) В адрес браузера бар. (**например: http://mchpboard/form.htm?led1=0&led2=1&led3=0** ) Данные, передаваемые через **GET** автоматически декодируется и сохраняется в **curHTTP**.**data** массиве. Так как они будут храниться в памяти, эти данные ограничены размером **curHTTP**.**data**, которая по умолчанию 100 байт. Однако, как правило, легче обрабатывать данные, полученные этим способом.

Функция обратного вызова **HTTPExecuteGet** ( см. стр. 242 ) предназначена разработчиком приложения для обработки этих данных и выполнять любые необходимые действия. Функции **HTTPGetArg** (см. стр. 244 ) и **HTTPGetROMArg** (см. стр. 245 ) обеспечивают простой способ для получения переданных значений для обработки.

Следующий пример демонстрирует форму контроля нескольких светодиодов .

```
<form method="get" action="leds.htm">
LED 1: <input type="checkbox" name="led1" value="1" /><br />
LED 2: <input type="checkbox" name="led2" value="1" /><br />
LED 3: <input type="checkbox" name="led3" value="1" /><br />
```

```
<input type="submit" value="Set LEDs" />
</form>
```

Предположим, пользователь выбирает флажки для светодиодов LED 1 и LED3. Следующая строка будет отправлена на сервер:

GET / leds.htm ? Led1 = LED 1 & LED3 = 1 HTTP/1.1

Веб-сервер **http2** будет разбирать этот запрос и хранить следующую строку в **curHTTP.data**:

" led1 \ 01 \ 0led3 \ 01 \ 0 \ 0 "

Затем он будет вызвать **HTTPExecuteGet** ( см. стр. 242 ), чтобы обработать этот вход. Для обработки этих данных, что обратного вызова нужно сделать несколько вещей. Во-первых, он должен вызвать **MPFSGetFilename** (см. стр. 277), чтобы проверить, какая была представлена форма. (Этот шаг может можно опустить, если только одна форма предоставляется в приложении. ) Далее, поскольку контроль флажок был использован состояние по умолчанию беспрепятственно следует предположить. Наконец , функция обратного вызова должны искать для каждого аргумента он ожидает, сравнить значение, и установите светодиодные штифты соответственно. В следующем примере удовлетворяются все эти требования:

```
HTTP_IO_RESULT HTTPExecuteGet(void)
{
BYTE *ptr, filename[20];
// Загрузка файловое имя (filename[] должно быть большой достаточно, чтобы проводить
// самое длинное файловое имя ожидало)
MPFSGetFilename(curHTTP.file, filename, 20);
// Проверьте файловое имя
if(!strcmppgm2ram(filename, (ROM char*)"leds.htm"))
{
// Примите по умолчанию состояние
LED1_IO = 0;
LED2_IO = 0;
LED3_IO = 0;
// Поиск каждого LED параметр и процесс
ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE*)"led1");
if(ptr)
LED1_IO = (*ptr == '1');
ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE*)"led2");
if(ptr)
LED2_IO = (*ptr == '1');
ptr = HTTPGetROMArg(curHTTP.data, (ROM BYTE*)"led3");
if(ptr)
LED3_IO = (*ptr == '1');
}
// Укажите завершение
return HTTP_IO_DONE;
}
```

**POST Метод**
Метод POST передает данные после того как все заголовки запросов были отправлены. Эти данные не отображается в адресной строке браузера, и можно увидеть только с помощью инструмента захвата пакетов. Он тем не менее использует тот же метод кодирования URL.

Сервер http2 не выполняет никакого предварительного разбора этих данных. Все данные POST остаются в буфере TCP, так обычно приложению нужен будет доступ к буферу TCP непосредственно для  его получения и расшифровки. Функции **HTTPReadPostName** (см. страница 247 ) и **HTTPReadPostValue** (см. стр. 248 ) были предоставлены для оказания помощи этим требованиям. Однако эти функции могут использовать только целые переменные, и чтобы они уместились в буфере TCP сразу.

Большинство функций постобработки будет осуществляться в виде конечных автоматов, чтобы использовать эти функции. Переменная **curHTTP.smPost** доступно для хранения текущего состояния. Эта переменная состояния машины сбрасывается в ноль при каждом новом запросе.

Функции должны вообще реализовать состояние, чтобы прочитать имя переменной, а другие, чтобы прочитать ожидаемое значение. Дополнительные состояния могут быть полезны в зависимости от применения.

В следующем примере форма принимает адрес электронной почты, тему и текст сообщения. Поскольку эти данные, скорее всего составит более 100 байт, он должен быть представлен через POST.

```
<form method="post" action="/email.htm">
To: <input type="text" name="to" maxlength="50" /><br />
Subject: <input type="text" name="subject" maxlength="50" /><br />
Message:<br />
<textarea name="msg" rows="6"></textarea><br />
<input type="submit" value="Send Message" /></div>
</form>
```

Предположим, что пользователь вводит следующие данные в следующем виде:

```
To: joe@picsaregood.com
Subject: Sent by a PIC
Message: I sent this message using my development board!
```

Сообщение: Я послал это сообщение с помощью моей платы!

Функция **HTTPExecutePost** ( смотри страницу 243) функция будет вызываться со следующими данными в буфере TCP:

```
to=joe%40picsaregood.com&subject=Sent+by+a+PIC
```

&msg=I+sent+**this**+message+**using**+my+development+board%21

Для использования модуля электронной почты, приложение должно читать в адрес и тему, сохранить их в памяти, а затем отправить сообщение. Однако, так как не гарантируется, что сообщение поместиться в памяти все сразу, оно должно быть прочитано когда пространство будет доступно и передаться модулю электронной почты. Государственная машина, в сочетании с функцией **HTTPReadPostName** (см. стр. 247) и **HTTPReadPostValue** ( см. стр. 248 ) могут сильно это упростить.

В следующем примере функция обратного вызова будет правильно разбирать этот вход. В этом примере предполагается, что это единственный сформировать совет принимает, поэтому проверка имя файла не выполняется. Адрес будет сохранен на **curHTTP.data**[0:49], и вопрос будет храниться при **curHTTP.data**[50:99]. Это не самое оптимальное решение, но служит в качестве простого примера .

```
HTTP_IO_RESULT HTTPExecutePost(void)
{
BYTE *dest, temp[16];
// Определите государственные машинные величины
#define SM_READ_NAME (0u)
#define SM_READ_VALUE (1u)
#define SM_START_MESSAGE (2u)
#define SM_SEND_MESSAGE (3u)

switch(curHTTP.smPost)
{
//---------------------------------------
case SM_READ_NAME:

// Прочитанный следующее переменное имя. Если полное имя -
// не обнаруженное, запросите более данные. Эта функция будет
 // автоматически исключать неправильные данные, чтобы предохранять
 // буфер от переполнения.
if(HTTPReadPostName(temp,16) == HTTP_READ_INCOMPLETE)
return HTTP_IO_NEED_DATA;
// Сохранять величин ", чтобы" на curHTTP.data[0:49]
if(!strcmppgm2ram((char*)temp, (ROM char*)"to"))
dest = curHTTP.data;
// Сохранять величин "предмета" на curHTTP.data[50:99]
else if(!strcmppgm2ram((char*)temp, (ROM char*)"subject"))
dest = curHTTP.data + 50;
// Когда "msg" столкнулось с, стартовая посылка
else if(!strcmppgm2ram((char*)temp, (ROM char*)"msg"))
{
curHTTP.smPost = SM_START_MESSAGE;
break;
}
// Когда "msg" столкнулось с, стартовой посылкой
else
dest = NULL;
```

```
// Перемещение на следующее состояние, но не ломается пока
curHTTP.smPost = SM_READ_VALUE;

//-----------------------------------------
case SM_READ_VALUE:
// Прочитанный следующая величина. Если полная величина -
// не обнаруженное, запросите более данные. Эта функция будет
// автоматически исключать неправильные данные, чтобы предохранять
// буфер от переполнения.
if(HTTPReadPostValue(dest,50) == HTTP_READ_INCOMPLETE)
return HTTP_IO_NEED_DATA;
// Возврат должен читать новое имя
curHTTP.smPost = SM_READ_NAME;
break;

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

case SM_START_MESSAGE:

// TODO: ВЫПОЛНИТЕ необходимые задачи, чтобы начинать посылать сообщение.
// Перейдите на посылку тела сообщения
curHTTP.smPost = SM_SEND_MESSAGE;
break;

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

case SM_SEND_MESSAGE:
// Сообщение может быть более длинным чем буфер TCP может провести
// сразу. Для того, чтобы избегать ошибок, прочитайте часть данных
// частью и посылайте это в модуль эл.почты. Это требует
// использование функций TCP непосредственно.
// Пошлите все остальные данные
while(curHTTP.byteCount > 0)
{
// Сначала проверьте, если данные готовы
if(TCPIsGetReady(sktHTTP) == 0)
return HTTP_IO_NEED_DATA;
// TODO: ПРОЧИТАЙТЕ данные с TCPGetArray и посылайте
// это в модуль эл.почты.
}
// Процесс завершен
return HTTP_IO_DONE;
}
// Примите возврат для государственной машинны удобства.
// Не возвращайте HTTP_IO_NEED_DATA здесь по умолчанию, поскольку
// занятие так когда более данные не прибудут, - причина для
// сервера HTTP2, чтобы возвращать ошибку пользователю.
return HTTP_IO_WAITING;
}
```

Предыдущий пример использует **HTTPReadPostName** (см. стр. 247 ) и **HTTPReadPostValue** (см. стр. 248 )функций, а также демонстрирует необходимость использования **TCPIsGetReady** (см. стр. 541 ), **TCPget** (см. стр. 537 ) и **TCPGetArray** (см. стр. 538 ), когда значения более длинные чем ожидалось. Для приложений, которые будут получать и реагировать на параметры сразу и не имеют необходимости в государственной машине, просто пока петля может быть написана около **HTTPReadPostPair** (см.страница 248 ) для выполнения обратного вызова. **HTTPPostLCD** (см. стр. 94) функция в TCPIP Demo App предоставляет этот простой пример.

Дополнительные примеры см. в **CustomHTTPApp**.c в проекте TCPIP Demo App .

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

### 10.8.1.3 http2 Аутентификация

Протокол HTTP предоставляет метод для серверов, запросить имя пользователя и пароль от клиента перед предоставлением доступа на страницу. Сервер http2 поддерживает этот механизм аутентификации, что позволяет разработчикам реализовать для доступа к некоторым или всем страницам, запрос ввести действительные учетные данные. Функциональность утентификации ( см. стр. 89 ) поддерживается двумя предоставленными пользователю функциями обратного вызова. Первая, **HTTPNeedsAuth** (См. стр. 245 ), определяет, если запрашиваемая страницы требует ввести действительные учетные данные, чтобы продолжить. Вторая, **HTTPCheckAuth** (см. стр. 242 ), проверяет имя пользователя и пароль от принятого списка и определяет, следует ли разрешить или запретить доступ. Этот раскол между двумя функциями обратного вызова диктуется природой протокола HTTP и низкой памяти архитектуры сервера http2. В случаях, когда различные учетные данные или наборы учетных данных могут быть приняты для различных страниц, эти две функции взаимодействуют друг с другом с помощью одного байта, хранящейся в **curHTTP.isAuthorized**.

#### Требование проверки подлинности

Когда сделали запрос, вызывается функция **HTTPNeedsAuth** (см. стр. 245 ), чтобы определить, защищена ли страница паролем. Эта функция возвращает значение для http2 сервера, как действовать. Самый старший бит показывает, является ли получение доступа. То есть, значения 0x80 и выше разрешить доступ безоговорочно, в то время как значения 0x79 и ниже потребует имя пользователя и пароль на более позднем этапе. Значение, возвращенное хранится как **curHTTP.isAuthorized** так что она может быть доступна в будущих функций обратного вызова.

Следующий пример является наиболее простым случаем, в котором все файлы требуют пароля для доступа:

```
BYTE HTTPNeedsAuth(BYTE* cFile)
{
return 0x00;
}
```

В некоторых случаях, только определенные файлы должны быть защищены. Второй пример требует пароль для любого файла расположенного в /treasure папке:

```
BYTE HTTPNeedsAuth(BYTE* cFile)
{
// Сравнение с папкой "/ treasure ". Не используйте strcmp здесь, поскольку
// cFile Имеет дополнительный путь инфо как например, "/treasure/gold.htm"
if(!memcmppgm2ram((void*)cFile, (ROM void*)"treasure", 8))
return 0x00;
return 0x80;
}
```

Для более комплексного использования, можно было потребовать имя административного пользователя, чтобы иметь доступ к /admin папке, тогда как любой подлинный пользователь может иметь доступ к остальной части сайта. Третий пример требует другую установку имени пользователя и комбинаций пароля для /admin папки против остальной части сайта:

```
#define HTTP_AUTH_OTHER (0x01)
BYTE HTTPNeedsAuth(BYTE* cFile)
{
// Возврат специфический код для пользователей администрации
if(!memcmppgm2ram((void*)cFile, (ROM void*)"admin", 5))
return HTTP_AUTH_ADMIN;
return HTTP_AUTH_OTHER;
}
```

**Пользователи учетных данных**

**HTTPCheckAuth** (см. стр. 242 ) функция определяет, является ли указанное имя пользователя и пароль действительными для получения доступа к этому ресурсу. Опять же, самый старший бит показывает, является ли получение доступа. Значение, возвращенное также сохраняется как **curHTTP.isAuthorized** так что она может быть доступна в будущих функций обратного вызова.

Следующий пример является наиболее простой случай, в котором один пользователь / пароль пара принимается для всех страниц :

```
BYTE HTTPCheckAuth(BYTE* cUser, BYTE* cPass)
{
if(!strcmppgm2ram((char*)cUser, (ROM char*)"AliBaba") &&
!strcmppgm2ram((char*)cPass, (ROM char*)"Open Sesame!") )
return 0x80;
return 0x00;
}
```

В некоторых случаях, возможно, придется создать несколько пользователей с различными уровнями доступа. В следующем третьем примере удовлетворяются эти потребности **HTTPNeedsAuth** (см. стр. 245 ):

```
BYTE HTTPCheckAuth(BYTE* cUser, BYTE* cPass)
{
/ / Проверяем, для администратора первым

if(curHTTP.isAuthorized == HTTP_AUTH_ADMIN &&
!strcmppgm2ram((char*)cUser, (ROM char*)"admin") &&
!strcmppgm2ram((char*)cPass, (ROM char*)"s3cREt") )
return 0x80;
if(!strcmppgm2ram((char*)cUser, (ROM char*)"kate") &&
!strcmppgm2ram((char*)cPass, (ROM char*)"puppies!") )
return 0x80;
return 0x00;
}
```

Более сложные виды использования, конечно, возможны. Многие приложения могут выбрать для хранения имен пользователей и паролей в EEPROM или в другой энергонезависимой памяти, чтобы они могли быть обновлены конечного пользователя. Некоторые приложени , возможно, пожелает вернуться различные значения выше 0x80 в **HTTPCheckAuth** ( см. стр. 242 ), чтобы потом функции обратного вызова может определить, какой пользователь зашел. Гибкость этих функций предусматривает гораздо больше возможностей, которые не документированны здесь, но может быть разработаны в течение нескольких часов.

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

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

**HTTP cookie**

**Ку́ки** (от англ. *cookie* — печенье) — небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя. Веб-клиент (обычно веб-браузер) всякий раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб-серверу в составе HTTP-запроса. Применяется для сохранения данных на стороне пользователя, на практике обычно используется для:

- аутентификации пользователя;
- хранения персональных предпочтений и настроек пользователя;
- отслеживания состояния сеанса[en] доступа пользователя;
- ведения статистики о пользователях.

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

**10.8.1.4 HTTP2 Cookies**

В соответствии с проектом , HTTP является сессия - менее и государственно- менее протокол; каждое соединение является независимым сессия без всякой связи с другой. **Cookies** ( см. стр. 91 ) добавляли к описанию протокола для решения этой проблемы. Эта функция позволяет веб- серверу для хранения небольших кусочков текста в браузере пользователя. Эти значения будут возвращены на сервер при каждом запросе, что позволяет сервер связать переменные сессии с просьбой. ***Cookies*** ( см. стр. 91 ), как правило, используется для более продвинутых системы аутентификации**.**

Лучшая практика, как правило, для хранения основной части данных на сервере, и хранить только уникальный идентификатор с браузером. это сокращает накладные расходы данных и гарантирует, что пользователь не может изменить значения, хранящиеся с сессии. Тем не менее, логика должны быть реализована когда на сервере истекают старые сеансы и выделяется память для новых. Если конфиденциальные данные хранятся, важно также, что идентификатор быть достаточно случайный так, чтобы предотвратить кражи или подмены **Cookies** другого пользователя.

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

**Получение Cookies**

В сервере http2, **Cookies** извлекаются автоматически. Они хранятся в **curHTTP.data**, так же, как любой другой форме GET аргумент или будет параметр URL. Наиболее подходящее место для разбора этих значений, следовательно, в **HTTPExecuteGet** (см. страница 242 ) обратного вызова с помощью **HTTPGetArg** (см. стр. 244 ) или функции **HTTPGetROMArg** (см. стр. 245 ), чтобы найти значения.

Эта модель потребляет некоторые из ограниченного пространства, доступного для параметров URL. Убедитесь, что **Cookies** не потребляют больше места, чем доступно (как определено **HTTP_MAX_DATA_LEN** (см. стр. 254) ) и что они будут соответствовать после любых данных, которые могут быть представлены через форму GET. Если есть достаточно места отсутствуют, **Cookies** будет усечен.

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

**Установка Cookies**

**Cookies** ( см. стр. 91 ) может быть установлен в **HTTPExecuteGet** ( см. стр. 242 ) или **HTTPExecutePost** (см. стр. 243 ). Для установки **Cookies**, хранить пары имя / значение в **curHTTP.data** в виде серии строки с нулевым символом. Затем установите, **curHTTP.hasArgs** равно числу пар имя / значение, которые будут установлены. Например, следующий код устанавливает куки, указывающий пользователя предпочтение типа **Cookies**:

```
void HTTPExecuteGet(void)
{
```

```
...
// Set a cookie
strcpypgm2ram((char*)curHTTP.data, (ROM char*)"flavor\0oatmeal raisin");
curHTTP.hasArgs = 1;
...
}
```

После этого, все будущие запросы от этого браузера будет включать в себя параметр «изюминку» в **curHTTP.data**, наряду с связано значение " овсяное с изюмом ".

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

#### 10.8.1.5 Сжатие HTTP2

Все современные веб-браузеры могут получить файлы, закодированные с компрессией GZIP. Для статических файлов ( тех, кто не динамический переменные), это может уменьшить количество данных, передаваемых на целых 60 %. MPFS2 Утилита автоматически определить, какие файлы могут воспользоваться сжатия GZIP, и будет хранить сжатый файл на изображении, если это возможно MPFS2. Как правило, это включает в себя все JavaScript и CSS файлы. ( Изображения обычно уже сжаты, поэтому MPFS2 Утилита обычно решают, что лучше хранить их в несжатом. ) Это HTTP Сервер будет легко вернуть этот сжатый файл в браузере. Менее энергонезависимой место для хранения будут необходимы для образ MPFS2, и быстрее переводы обратно клиенту приведет. Никаких специальных настроек не требуется для этой функции.

Чтобы исключить некоторые расширения из сжимается, использовать диалог Дополнительные настройки в MPFS2 Utility.

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

### 10.8.2 Общественных Перечислений Участников HTTP2

**Имя Description**

**HTTP_IO_RESULT** ( смотри страницу 241)
Результат состояния для выполнения обратного вызова

**HTTP_READ_STATUS** ( смотри страницу 241)
Результат состояния для **HTTPPostReadName** и **HTTPPostReadValue**

**Функции**

Имя Description

**HTTPCheckAuth** ( смотри страницу 242) Выполняет проверку имени пользователя и пароля .

**HTTPExecuteGet** ( СМОТРИ страницу 242) Процессы **GET** переменные формы полевых и cookies.

**HTTPExecutePost** ( СМОТРИ страницу 243) Процессы **POST** переменные формы и данных.

**HTTPGetArg** ( СМОТРИ страницу244) Находит значение поля формы в данном массиве данных .

**HTTPGetROMArg** ( СМОТРИ страницу 245) Находит значение поля формы в данном массиве данных .

**HTTPNeedsAuth** ( СМОТРИ страницу 245) Определяет, требует ли имя файла аутентификации.

**HTTPPrint_varname** ( смотри страницу 246) Вставка динамического контента в веб-страницу.

**HTTPReadPostName** ( смотри страницу 247) Читает название от URL кодируются строки в буфере TCP .

**HTTPReadPostValue** ( СМОТРИ страницу 248) Читает значения из URL кодируются строки в буфере TCP .

**HTTPURLDecode** ( СМОТРИ страницу 249) Анализирует строку из кодировки URL для простого текста .

**Макро**

**Имя Description**

**HTTPReadPostPair** ( смотри страницу 248) Читает имя и значение пара из URL кодируются строки в буфере TCP .

**sktHTTP** ( СМОТРИ страницу 250), Доступ к текущей розетке Сервер HTTP2 (смотри страницу 230)

**Структуры**

**Имя Description**

**HTTP_CONN** (240) смотри страничную продлен данные о состоянии для каждого соединения

**Переменные**

**Описание Имени**

**curHTTP** ( СМОТРИ страницу 240) Текущее состояние HTTP соединение

**Описание**
Следующие функции и переменные доступны или реализованы путем применения стека.

//*-*-*-*-*----------

### 10.8.2.1 curHTTP Variable

**File**
HTTP2.c

**C**
HTTP_CONN **curHTTP**;

**Description**

Текущее состояние HTTP соединение

//*-*-*-*-*----------

### 10.8.2.2 HTTP_CONN Structure

**File**
HTTP2.h
**C**
```
typedef struct {
DWORD byteCount;
DWORD nextCallback;
DWORD callbackID;
DWORD callbackPos;
BYTE * ptrData;
BYTE * ptrRead;
MPFS_HANDLE file;
MPFS_HANDLE offsets;
BYTE hasArgs;
BYTE isAuthorized;
HTTP_STATUS httpStatus;
HTTP_FILE_TYPE fileType;
BYTE data[HTTP_MAX_DATA_LEN];
BYTE smPost;
} HTTP_CONN;
```

**Members**
**Members Description**

DWORD **byteCount**; Сколько байт прочитано до сих пор
DWORD **nextCallback**; Байт Индекс следующего обратного вызова

DWORD **callbackID**; Ответный ID для выполнения, также используется в качестве сторожевого таймера

DWORD **callbackPos**; Ответный индикатор положения

BYTE * **ptrData**; Указывает на первой свободный байт в данных

BYTE * **ptrRead**; Указывает на текущее местоположение чтения

MPFS_HANDLE **file**; Указатель для файла, обслуживаемых файла

MPFS_HANDLE **offsets**; Указатель для любого смещения Информация используется файл

BYTE **hasArgs**; Правда, были ли получить или печенья аргументы

BYTE **isAuthorized**;  0x00-0x79 на неудачу, 0x80-0xff на перевал

HTTP_STATUS  httpStatus; Метод запроса / статус

HTTP_FILE_TYPE **fileType**; Тип файла, чтобы вернуться с Content-Type

BYTE **data**[HTTP_MAX_DATA_LEN];  Общего назначения буфер данных

BYTE **smPost**;  POST переменная состояния машины

//*-*-*-*-*----------

#### 10.8.2.3 HTTP_IO_RESULT Enumeration

**File**
HTTP2.h
**C**

```
typedef enum {
HTTP_IO_DONE = 0u,
HTTP_IO_NEED_DATA,
HTTP_IO_WAITING
} HTTP_IO_RESULT;
```

**Members**
**Members Description**
**HTTP_IO_DONE** = 0u Завершенные процедурой
**HTTP_IO_NEED_DATA** больше данных, необходимые для продолжения, вызовите позже
**HTTP_IO_WAITING** Ожидание асинхронного завершения процесса, вызовите позже.

**Description**
Результат состояния для выполнения обратного вызова

//*-*-*-*-*----------

#### 10.8.2.4 HTTP_READ_STATUS Enumeration

**File**
HTTP2.h
**C**

```
typedef enum {
```

**HTTP_READ_OK** = 0u,
**HTTP_READ_TRUNCATED**,
**HTTP_READ_INCOMPLETE**
} HTTP_READ_STATUS;
**Members**
**Members Description**
**HTTP_READ_OK** = 0u Читать успешно
**HTTP_READ_TRUNCATED** предотвратить путем усечения значение
**HTTP_READ_INCOMPLETE** весь объект еще не в буфере. Повторите попытку позже.

**Description**
Результат состояния для **HTTPPostReadName** и **HTTPPostReadValue**

//*-*-*-*-*----------

### 10.8.2.5 HTTPCheckAuth Function

**File**
HTTP2.h
**C**
BYTE **HTTPCheckAuth**(
BYTE* **cUser**,
BYTE* **cPass**
);

**Description**
Эта функция реализована разработчиком приложения в **CustomHTTPApp**.c. Его функция заключается в определении, если имя пользователя и пароль поставляется клиентом являются приемлемыми для данного ресурса.

Значение **curHTTP**.**isAuthorized** будет установлено в предыдущем возвращаемом значении **HTTPRequiresAuthorization**. Этот обратный вызов функция может проверить это значение, чтобы определить, если только конкретные имена пользователей и пароли будут приняты для этого ресурса.

Возвращаемые значения **0x80 - 0xFF** указывают, что полномочия не были приняты, в то время как значения от **0x00 до 0x79** показывают, что удалось авторизироваться. В то время как большинство приложений будет использовать только одно значение, чтобы предоставить доступ, гибкость обеспечивается в магазине несколько значений, чтобы указать, какой пользователь (или группу пользователя ) зашли Возвращаемое значение этой функции сохраняется как **curHTTP.isAuthorized**, и будут доступны для будущих обратных вызовов, в том числе любой из **HTTPExecuteGet** ( см. стр. 242 ), **HTTPExecutePost** (см. стр. 243), или **HTTPPrint_varname** (см. стр. 246 ) обратные вызовы.

**Remarks**
Эта функция вызывается только когда заголовок Авторизация встречается.

Эта функция может написать в буфер TCP.

**Internal**
См. документацию в API TCP / IP стека или HTTP2.h для деталей.

**Preconditions**
None

**Parameters**
**Parameters Description**
**cUser** имени пользователя, предоставленной клиентом
**cPass** пароль предоставленной клиентом

**Return Values**
**Return Values Description**

**<= 0x79**, полномочия были отклонены
>= **0x80** Доступ 0x80 предоставляется на связи

//*-*-*-*-*----------

### 10.8.2.6 HTTPExecuteGet Function

**File**
HTTP2.h

**C**
HTTP_IO_RESULT **HTTPExecuteGet**();

**Description**
Эта функция реализована разработчиком приложения в **CustomHTTPApp**.c. Его целью является, чтобы анализировать данные, полученные от параметров URL ( метод GET формы) и печенье и выполнять любые специфические для приложения задачи в ответ на них входы. Любой требуется аутентификация уже была подтверждена.

Когда эта функция вызывается, **curHTTP.data** содержит последовательные пары имя /значение строк, представляющих данные, полученные. в этот формат, **HTTPGetArg** (см. стр. 244 ) и **HTTPGetROMArg** (см. стр. 245 ) можно использовать для поиска конкретных переменных на входе. Если буфер данных пространство, ассоциированное с этой связи требуется, **curHTTP.data** могут быть перезаписаны здесь После того как приложение будет сделано со значениями. Любые данные помещены туда будут доступны для будущих обратных вызовов для этого соединения, в том числе **HTTPExecutePost** (см. стр. 243 ) и любой **HTTPPrint_varname** (см. стр. 246 ) динамические замены.

Эта функция может также выдавать перенаправления, установив **curHTTP.data** к имени файла назначения или URL, и **curHTTP.httpStatus** чтобы **HTTP_REDIRECT**.

Наконец, эта функция может устанавливать куки. Установите **curHTTP**.**data** к серии струнных пар имя / значение ( в том же формате, в котором параметры приехать), а затем установить **curHTTP.hasArgs** равные числу пар имя печенья / значение. Печенье будет передаются в браузер, и все последующие запросы будут иметь те значения, доступные в **curHTTP.data.**

**Remarks**
Эта функция вызывается, только если переменные получил через параметры URL или аргументов **Cookie**. Эта функция может написать в буфер TCP.

Эта функция может обслуживать несколько запросов HTTP одновременно. Соблюдайте осторожность при использовании глобальных или статических переменных внутри этой рутины. Используйте **curHTTP.callbackPos** или **curHTTP.data** для хранения, связанного с индивидуальным запросам.

**Internal**
Смотри документацию в TCP/IP Стека API или HTTP2.h относительно деталей.

**Preconditions**
None

**Return Values**
**Return Values Description**
**HTTP_IO_DONE** сделан обработка
**HTTP_IO_NEED_DATA** это значение не может быть возвращен, потому что больше данных не будут доступны

**HTTP_IO_WAITING** приложение ждет асинхронного завершения процесса, и это функция должна быть вызвана позже

//**-*-*-*-*----------**

### 10.8.2.7 HTTPExecutePost Function

**File**
HTTP2.h

**C**
HTTP_IO_RESULT **HTTPExecutePost**();

**Description**
Эта функция реализована разработчиком приложения в **CustomHTTPApp**.c. Его целью является, чтобы анализировать данные, полученные от **POST** форм и выполнять любые специфические для приложения задачи в ответ на эти входы. Любой требуется аутентификация имеет уже была подтверждена перед эта функция вызывается.

Когда эта функция вызывается, данные **POST** будет ждать в буфере **TCP**. **curHTTP.byteCount** будет указать количество байт остальные должны быть получены до запрос браузера завершена.

Поскольку данные все еще находится в буфере TCP, приложение должно вызвать **TCPget** (см. стр. 537 ) или **TCPGetArray** (см. стр. 538 ) в для того, чтобы извлечь байт. Когда это будет сделано, **curHTTP.byteCount** должны быть обновлены, чтобы отразить, сколько байтов до сих пор остаются.

Функции **TCPFind** (см. стр. 534 ), **TCPFindString**, **TCPFindROMString** , **TCPFindArray** (см. стр. 534 ), и **TCPFindROMArray** (см. стр. 536 ) можут быть полезными для поиска данных в буфере TCP.

В общем, данные, представленные в веб-формах через **POST** являются **URL** закодированными. **HTTPURLDecode** (см. 249 стр.) функция может быть использована для декодирования этой информации обратно в стандартную строку, если требуется. Если буфер данных пространство, ассоциированное с этой связи требуется, **curHTTP.data** могут быть перезаписаны здесь как только приложение выполняется со значениями. Любые данные размещены там будет быть доступны для будущих обратных вызовов для этой связи, в том числе **HTTPExecutePost** и любой **HTTPPrint_varname** (см. стр.246) динамические замены.

Всякий раз, когда обрабатывается **POST** формы рекомендуется выдавать редирект обратно в браузер, либо к странице состояния или та же форма страница, которая была размещена. Это предотвращает случайные повторяющиеся представления (нажав обновления или назад / вперед ) и позволяет избежать предупреждения браузера о " повторной данные форм ". Перенаправление может быть выдан в браузер, установив **curHTTP.data** в файл назначения или **URL**, и **curHTTP.httpStatus** к **HTTP_REDIRECT**.

Наконец, эта функция может устанавливать ***cookie***. Установите **curHTTP.data** к серии струнных пар имя / значение ( в том же формате, в котором параметры приехать), а затем установить **curHTTP.hasArgs** равные числу пар имя / значение ***cookie***. ***Cookie*** будет передаются в браузер, и все последующие запросы будут иметь те значения, доступные в **curHTTP.data**.

**Remarks**
Эта функция вызывается только когда метод запроса **POST**, и используется только при **HTTP_USE_POST** определяется. это метод может написать в буфер TCP.

Эта функция может обслуживать несколько запросов HTTP одновременно. Соблюдайте осторожность при использовании глобальных или статических переменных внутри этой рутины. Используйте **curHTTP.callbackPos** или **curHTTP.data** для хранения, связанного с индивидуальным запросам.

**Internal**
Смотри документацию в TCP/IP Стека API или HTTP2.h относительно деталей.

**Preconditions**
None

**Return Values**
**Return Values Description**
**HTTP_IO_DONE** сделан обработка
**HTTP_IO_NEED_DATA** больше данных необходимо продолжать, и эта функция должна быть вызвана позже
**HTTP_IO_WAITING** приложение ждет асинхронного завершения процесса, и это функция должна быть вызвана позже

//*-*-*-*-*----------

### 10.8.2.8 HTTPGetArg Function

**File**
HTTP2.h

**C**
BYTE* **HTTPGetArg**(
BYTE* **cData**,
BYTE* **cArg**
);

**Returns**
Указатель на величина аргумент, или **NULL** если не обнаружено.

**Description**
Просматривает массив данных, чтобы находить величину связанную данным аргументом. Может быть использовано, чтобы находить величины области формы в данных полученных над **GET** или **POST**.

Конец данных принят, чтобы быть достигнут когда недействительный параметр имени столкнулся. Это требует строку, чтобы иметь ровное число недействительный расторгнутых строк, сопровожденных дополнительным недействительным терминатором.

**Preconditions**
Массив данных имеет правильную серию недействительного расторгнутого имени/величины пар.

**Parameters**
**Parameters Description**
**cData** -Параметров буфер, чтобы искать
**cArg-** имя аргумента, чтобы оказываться

//*-*-*-*-*----------

### 10.8.2.9 HTTPGetROMArg Function

**File**
HTTP2.h

**C**

BYTE* **HTTPGetROMArg**(
BYTE* **cData**,
ROM BYTE* **cArg**
);

**Returns**
Указатель на значение аргумента, или NULL если он не найден.

**Description**
Поиск через массив данных, чтобы найти значение, связанное с данным аргументом. Он может быть использован, чтобы найти значения полей формы в данные, полученные в течение GET или POST.

Конец данных предполагается быть достигнуто, когда нулевой параметр имени встречается. Это требует, чтобы иметь строку четное число строки с нулевым символом, с последующим дополнительным нулевое окончание.

**Remarks**
Эта функция является псевдонимом для **HTTPGetArg** (см. стр. 244) на не- PIC18 платформ.

**Preconditions**

Массив данных имеет действительный серию нулевых прекращается пар имя / значение.

**Parameters**
**Parameters Description**
**cData** -буфер для поиска

**cArg-** имя аргумента , чтобы найти

**//*-*-*-*-*----------**

### 10.8.2.10 HTTPNeedsAuth Function

**File**
HTTP2.h

**C**
BYTE **HTTPNeedsAuth**(
BYTE* **cFile**
);

**Description**
Эта функция реализована разработчиком приложения в **CustomHTTPApp**.c. Его функция заключается в определении, если файл быть просил требует аутентификации для просмотра.

Имя пользователя и пароль, если имеются, прибудет позже с просьбой заголовки, и их обработка происходит в то время.

Возвращаемые значения 0x80 - 0xFF показывают, что аутентификация не требуется, в то время как значения от 0x00 до 0x79 указывают, что требуются ввести имя пользователя и пароль прежде чем продолжить. В то время как большинство приложений будет использовать только одно значение для предоставления доступа и другой, чтобы требовать разрешения, диапазон позволяет использовать несколько "области " или наборы страниц должны быть защищены, с другими учетными данными требования к каждой из них.

Возвращаемое значение этой функции сохраняется как **curHTTP.isAuthorized** , и будут доступны для будущих обратных вызовов, в том числе **HTTPCheckAuth** ( см. стр. 242 ) и любой из **HTTPExecuteGet** ( см. стр. 242 ), **HTTPExecutePost** (см. стр.243 ), или **HTTPPrint_varname** (см. стр. 246 ) обратные вызовы.

**Remarks**
Эта функция может написать в буфер TCP .

**Internal**
Смотри документацию в TCP/IP Стека API или **HTTP2**.h относительно деталей.

**Preconditions**
None

**Parameters**
**Parameters Description**
**cFile** имя файла запрашиваемого

**Return Values**
**Return Values Description**
<= 0x79 правильная аутентификация потребовался
>= 0x80 доступу предоставляют для этой связи

//*-*-*-*-*----------

### 10.8.2.11 HTTPPrint_varname Function

**File**
HTTP2.h

**C**
```
void HTTPPrint_varname(
WORD wParam1,
WORD wParam2,
...
);
```

**Returns**

None

**Description**
Функции в этом стиле реализованы разработчиком приложения в **CustomHTTPApp**.c. Эти функции генерации динамический контент для вставки в веб-страниц и других файлов, возвращенных сервером http2.

Функции такого типа вызываются, когда динамическая переменная находится на веб-странице. (т.е. ~ имяПеременной ~ ) Название между символов тильды ' ~' добавляется к имени базовой функции. В этом примере, обратный вызов будет называться **HTTPPrint_varname**.

Прототип функции находится в **HTTPPrint**.h вашего проекта , который автоматически генерируемого MPFS2 Utility.

Прототип будет иметь параметры WORD включены для каждого параметра, переданного в динамической переменной . Например, переменная " ~ туАггау (2,6 ) ~ " будет генерировать прототип " недействительным HTTPPrint_varname ( Word, Microsoft Word ) ; " .

При вызове эта функция должна написать свой выход непосредственно в розетку TCP , используя любую комбинацию **TCPIsPutReady** (см.страница 541) , **TCPPut** (см. стр. 545 ) , **TCPPutArray** (см. стр. 545 ) , **TCPPutString** (см. стр. 547 ) , **TCPPutROMArray**(См. стр. 546 ) , и **TCPPutROMString** (см. стр. 546 ) .

Перед вызовом , сервер http2 гарантирует, что по крайней мере **HTTP_MIN_CALLBACK_FREE** (см. стр. 254 ) байт (по умолчанию до 16 байт ) свободны в выходной буфер . Если функция пишет меньше этой суммы , он должен просто записать данные в розетка и возвращение .

В ситуациях, когда функция должна написать еще эту сумму , он должен управлять его исходное состояние с помощью **curHTTP.callbackPos** .

Это значение будет равным нулю , прежде чем функция называется . Если функция управления его исходное состояние , он должен установить это ненулевое значение , прежде чем вернуться . Как правило, это используется для отслеживания , сколько байт было написано , или сколько остаются написано . Если **curHTTP.callbackPos** не равен нулю , то функция будет снова вызывается, когда более буферное пространство доступно . После обратного вызова завершается, установить это значение на ноль , чтобы возобновить нормальное обслуживание запроса.

//**********//
***Со́кеты*** *(англ. socket — разъём) — название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет — абстрактный объект, представляющий конечную точку соединения.*
//**********//

**Remarks**
Эта функция может обслуживать несколько запросов HTTP одновременно , особенно при управлении его исходное состояние .

**упражнение**

Будьте осторожны при использовании глобальных или статических переменных внутри этой рутины . Используйте **curHTTP.callbackPos** или **curHTTP.data** для хранения связаны с индивидуальным запросам .

**Preconditions**
None

**Parameters**
**Parameters Description**
**wParam1** Первый параметр, передаваемый в динамической переменной ( если таковые имеются)

**wParam2** Второй передаваемый параметр в динамической переменной ( если таковые имеются)

... Дополнительные параметры , как необходимо

//*-*-*-*-*----------

### 10.8.2.12 HTTPReadPostName Function

**File**
HTTP2.h
**C**

```
HTTP_READ_STATUS HTTPReadPostName(
BYTE* cData,
WORD wLen
);
```

**Description**

Читает название от URL кодируются строки в буфере TCP . Эта функция предназначена для вызова из **HTTPExecutePost**( См. стр. 243 ) обратный вызов с целью облегчения разбор поступающих данных . Эта функция также предотвращает переполнение буфера путем заставляя программиста , чтобы указать , сколько байтов , как ожидается . По крайней мере, 2 дополнительных байта необходимы в **cData** над Максимальная длина данных , которые предполагается читать.

Эта функция будет читать до следующего '=' символ, что указывает на конец параметра имени. Она предполагает, что передняя часть буфера является началом имени paramter для чтения.

Эта функция правильно обновляет **curHTTP**.**byteCount** путем уменьшения его на количество считанных байтов . Он также удаляет ограничивающую '=' из буфера.

**Preconditions**
Передний план буфера TCP является началом параметра имени, и остальная часть буфера TCP содержит закодированную строку URL с параметром имени завершенным '=' символ.

**Parameters**
**Parameters Description**
**cData** где, чтобы загружать имя как только оно прочитано
**wLen** сколько байтов может быть записано на **cData**

**Return Values**
**Return Values Description**
**HTTP_READ_OK** имя было успешно прочитаны
**HTTP_READ_TRUNCTATED** полное имя не могли поместиться в буфере , поэтому значение были усечены и данных имеет был потерян

**HTTP_READ_INCOMPLETE** полное имя еще не было в буфере , поэтому вызывать эту функцию позже , чтобы получить

//*-*-*-*-*----------

### 10.8.2.13 HTTPReadPostPair Macro

**File**
HTTP2.h

**C**
```
#define HTTPReadPostPair(cData, wLen) HTTPReadPostValue(cData, wLen)
```

**Description**

Читает имя и значение пара из URL кодируются строки в буфере TCP . Эта функция предназначена для вызова из **HTTPExecutePost** ( см. стр. 243 ) обратный вызов с целью облегчения разбор поступающих данных . Эта функция также предотвращает буфер переполняется , заставляя программиста , чтобы указать , сколько байтов , как ожидается . По крайней мере, 2 дополнительных байта необходимы в CDATA над максимальной длины данных, ожидаемого для чтения.

Эта функция не будет читать до следующего '**&**' символ,, что указывает на конец параметром значения. Она предполагает, что передняя часть буфер начало имя paramter для чтения.

Эта функция правильно обновляет **curHTTP.byteCount** путем уменьшения его на количество считанных байтов . Он также удаляет ограничивающую '**&**' из буфера.

После завершения двух строк будет существовать в буфере **cData**. Первым из них является имя параметра , которое было считано , а второй соответствующее значение .

**Remarks**
Эта функция является псевдонимом для **HTTPReadPostValue** (см. стр. 248) , так как они эффективно выполняют те же задачи. Имя предоставляется только для полноты картины.

**Preconditions**
Передний план буфера TCP является началом параметра имени, и остальная часть буфера TCP содержит закодированную строку URL с параметром имени завершенным символ '=' и параметр величины завершался '**&**'.

**Parameters**
**Parameters Description**
**cData** где хранить имя и значение строки , как только они читаются
**wLen** сколько байт может быть записан на **cData**

**Return Values**
**Return Values Description**
**HTTP_READ_OK** имя и значение были успешно прочитаны

**HTTP_READ_TRUNCTATED** полное имя и значение не могли поместиться в буфере , поэтому вход был усечен и данные были потеряны

**HTTP_READ_INCOMPLETE** полное имя и значение еще не было в буфере , поэтому вызывать эту функцию позже для получения

//*-*-*-*-*----------

### 10.8.2.14 HTTPReadPostValue Function

**File**
HTTP2.h

**C**
```
HTTP_READ_STATUS HTTPReadPostValue(
BYTE* cData,
WORD wLen
);
```

**Description**
Читает значения из URL кодируются строки в буфере TCP . Эта функция предназначена для вызова из **HTTPExecutePost**( См. стр. 243 ) обратный вызов с целью облегчения разбор поступающих данных . Эта функция также предотвращает переполнение буфера путем заставляя программиста , чтобы указать , сколько байтов , как ожидается . По крайней мере, 2 дополнительных байта необходимы в **cData** выше Максимальная длина данных , которые предполагается читать.

Эта функция не будет читать до следующего ' **&** ' характер , что указывает на конец параметром значения. Она предполагает, что передняя часть буфер начало стоимости **paramter** для чтения. Если **curHTTP.byteCount** указывает, что все ожидаемые байт в буфера , он предполагает, что все остальные данные значение и действует соответственно.

Эта функция правильно обновляет **curHTTP.byteCount** путем уменьшения его на количество считанных байтов . Завершающий ' **&** ' символ также удаляются из буфера .

**Preconditions**
Передний план буфера TCP является началом параметра имени, и остальная часть буфера TCP содержит закодированную строку URL с параметром имени завершенным '=' символ.

**Parameters**
**Parameters Description**
**cData** где хранить значения , когда он будет читать
**wLen** сколько байтов может быть записано на **cData**

**Return Values**
**Return Values Description**
**HTTP_READ_OK** значение было успешно прочитаны
**HTTP_READ_TRUNCTATED** вся стоимость не мог поместиться в буфере , поэтому значение были усечены и данных имеет был потерян

**HTTP_READ_INCOMPLETE** все значение еще не было в буфере , поэтому вызывать эту функцию позже , чтобы получить

//*-*-*-*-*----------

### 10.8.2.15 HTTPURLDecode Function

**File**
HTTP2.h

**C**
BYTE* **HTTPURLDecode**(
BYTE* **cData**
);

**Returns**
Указатель на последний недействительный терминатор в данных , что также первый свободный байт для новых данных.

**Description**
Анализирует строку из кодировки URL для простого текста . Следующие преобразования сделаны : '=' в '0 ',' & ' в '0', '+ ' до ' ' , и "% XX" к одному шестнадцатеричном байта.

После завершения , данные были декодируется и нулевой терминатор означает конец имени или значения. Второй нуль терминатор ( или нулевой параметр имени ) указывает на конец всех данных.

**Remarks**
Эта функция называется стеком для разбора **GET** аргументы и данные печенья . Пользовательские приложения могут использовать эту функцию для декодирования данных **POST** , но в первую очередь необходимо убедиться, что строка является нулем.

**Preconditions**

Параметр данные нулем и имеет по крайней мере один дополнительный байт бесплатный .

**Parameters**
**Parameters Description**
**cData** Строка , которая должна быть расшифрованы на месте.

//*-*-*-*-*----------

### 10.8.2.16 sktHTTP Macro

**File**
HTTP2.h

**C**
```
#define sktHTTP httpStubs[curHTTPID].socket // текущая socket
```

**Description**
Иметь доступ к текущей розетке

//*-*-*-*-*----------

## 10.8.3 HTTP2 Stack Members

**Functions**
**Name Description**
**HTTPInit** ( СМОТРИ страницу 250), Инициализирует модуль сервера HTTP.
**HTTPServer** ( СМОТРИ страницу 251)
Выполняет периодические задачи для модуля HTTP2.

**Module**
HTTP2 Server ( see page 230)

**Description**
Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные .

//*-*-*-*-*----------

### 10.8.3.1 HTTPInit Function

**File**
HTTP2.h

**C**

**void HTTPInit**();

**Returns**
None

**Description**
Устанавливает все HTTP розетки в состоянии прослушивания и инициализирует государственная машина и дескрипторов файлов для каждого соединения . Если SSL является включен , открывает сокет на этом порту , а также.

**Remarks**
Эта функция вызывается только один при жизни приложения.

**Preconditions**
TCP должен быть уже инициализирована.

**Section**
Функциональные Прототипы

//*-*-*-*-*----------

### 10.8.3.2 HTTPServer Function

**File**
HTTP2.h

**C**
**void HTTPServer**();

**Returns**
None

**Description**
Просматривает каждую открытую связь и пытается обработать все незавершенные операции.

**Remarks**
Эта функция действует как задачи (по аналогии с одним в OCPB ) . Он выполняет свою задачу в кооперативном образом , и главный Приложение должно вызвать эту функцию несколько раз, чтобы убедиться, что все открытые или новые соединения подаются своевременно .

**Preconditions**
HTTPInit ( СМОТРИ страницу 250)(), должно уже вызвано.

//*-*-*-*-*----------

### 10.8.4 HTTP2 Internal Members

**Enumerations**
**Name Description**
**HTTP_FILE_TYPE** ( seepage 253) Определения типов файлов

**HTTP_STATUS** ( see page254) Предусмотренные Команды и Коды Ответа Сервера
**SM_HTTP2** ( see page 262) Основной государственный HTTP соединения машины

**Functions**
**Name Description**
**HTTPHeaderParseAuthorization**( see page 256)
Анализирует " Авторизация: " заголовка для запроса и проверяет полномочия .

**HTTPHeaderParseContentLength**( see page 257) Анализирует"Content-Length : " заголовок для запроса .

**HTTPHeaderParseCookie** ( seepage 257) Анализирует "Cookie :" заголовки для запроса, и сохраняет их в качестве GET переменные .

**HTTPHeaderParseLookup** ( seepage 258) Звонки соответствующий парсер заголовка на основе индекса заголовка , что было прочитано из запроса .

**HTTPIncFile** ( see page 258) Записывает файл байт - на - байт для загруженного в данный момент гнездо TCP .

**HTTPLoadConn** ( see page 259) Выключатели текущий загруженный соединение для модуля http2 .

**HTTPMPFSUpload** ( see page259) Сохраняет список изображений, загруженных с помощью POST в качестве нового MPFS изображения в EEPROM или внешней Flash.

**HTTPProcess** ( see page 260) Выполняет все ожидающие операции для загруженного в данный момент связи HTTP

**HTTPReadTo** ( see page 260) не читает в буфер до указанного символа-ограничителя .

**HTTPSendFile** ( see page 262) Служит на следующий кусок **curHTTP** (см. стр. 240 ) 'ы файл , до)доступны TX FIFO пространство или б) следующий индекс обратного вызова , что наступит в первую очередь.

**Macros**
**Name Description**
**HTTP_CACHE_LEN** ( seepage 253) Макс жизни (сек) статических ответов как строки

**HTTP_MAX_DATA_LEN** (see page 254) Определить максимальную длину данных для чтения печенье и GET / POST Аргументы ( байт )

**HTTP_MAX_HEADER_LEN** (see page 254) Установите в длине длинной строки выше

**HTTP_MIN_CALLBACK_FREE**( see page 254) Определите минимальное число байтов свободных в TX FIFO перед выполнением обратные вызовы

**HTTP_PORT** ( see page 254) Определите порт прослушивания для сервера HTTP

**HTTP_TIMEOUT** ( see page 256) Максимальное время (сек ), где ожидает больше данных , прежде чем повторить и отключение socket

**HTTPS_PORT** ( see page 261) Определите порт прослушивания для сервера HTTPS (если STACK_USE_SSL_SERVER включена )

**smHTTP** ( see page 263) Доступ к текущему государственную машину

**RESERVED_HTTP_MEMORY** ( see page 263) Макро с указанием , сколько оперативной памяти выделить на контроллер Ethernet для хранения HTTP данные о состоянии .

**Module**
HTTP2 Server ( see page 230)

**Structures**
**Name Description**
**HTTP_STUB** ( see page255) Частичные данные государственные HTTP соединения Структура, магазины для каждого соединения , предназначенные для хранение в оперативной памяти быстрого доступа RAM

**Variables**
**Name Description**

**curHTTPID** ( см. стр. 252) идентификаторв настоящее время загружены HTTP_CONN (см. стр. 240 )

**httpContentTypes** (см.страница 256 ) Строки Content-Type , соответствующие

**HTTP_FILE_TYPE** (см. стр. 253 )

**httpFileExtensions** (см.страница 256 )

**HTTPRequestHeaders** (см. стр. 261) Строки заголовков , для которых мы хотели бы разобрать

**HTTPResponseHeaders** (см. стр. 261) Начальные строки реагирования ( В соответствии с HTTP_STATUS (см. стр. 254 ) )

**httpStubs** (см. стр. 262 ) HTTP заглушки с государственной машиной и розетки

**описание**

Следующие функции и переменные определяются как внутренние, к модулю http2 .

**Имя Description**

**HTTPHeaderParseAuthorization** ( смотри страницу 256)Выполняет грамматический разбор заголовок "Разрешения:" для запроса и проверяет верительные грамоты.

**HTTPHeaderParseContentLength** ( СМОТРИ страницу 257)Выполняет грамматический разбор заголовок "Content-Length:" для запроса.

**HTTPHeaderParseCookie** ( СМОТРИ страницу 257)Выполняет грамматический разбор заголовки "Печенья:" для запроса и загружает их как ПОЛУЧАТЬ переменные.

**HTTPHeaderParseLookup** ( СМОТРИ страницу 258)Вызывает подходящий синтаксический анализатор заголовка основанный в индексе заголовка, который был прочитан с запроса.

**HTTPIncFile** ( СМОТРИ страницу 258), Записывает файловый байт-для-байта в к настоящему времени загруженную розетку TCP.

**HTTPLoadConn** ( СМОТРИ страницу 259), Переключает к настоящему времени загруженную связь для модуля HTTP2.

**HTTPMPFSUpload** ( СМОТРИ страницу259) Сохраняет файл загруженный через ПОСТ как новый образ MPFS в EEPROM или внешней Flash.

**HTTPProcess** ( смотри страницу 260), Выполняет любые незаконченные операции для к настоящему времени загруженной связи HTTP.

**HTTPReadTo** ( СМОТРИ страницу 260), Читается в буфер до определенного символа разделителя.

**HTTPSendFile** ( СМОТРИ страницу 262), Обслуживается по следующем куске **curHTTP** ( смотри страничный 240)'s файл, вплоть до a) доступное пространство **TX FIFO** или b) следующий индекс возврата, какой угодно приходит сначала.

**Макро**

**Имя Description**

**HTTP_CACHE_LEN** ( смотри страницу 253)Целая жизнь Max (сек) статические ответы как строка

**HTTP_MAX_DATA_LEN** ( смотри страницу 254)Определите максимальную длину данных для чтения печенья и ПОЛУЧАЙТЕ/ПОСЫЛАТЬ аргументы (байты)

**HTTP_MAX_HEADER_LEN** ( смотри страницу 254)Установитесь на длину самой длинной строки выше

**HTTP_MIN_CALLBACK_FREE** ( смотри страницу 254)Определите минимальное количество байтов свободное в TX FIFO перед выполнять возвратов

**HTTP_PORT** ( смотри страницу 254), Определять слушание порта для сервера HTTP

**HTTP_TIMEOUT** ( смотри страницу256) Время Max (сек), чтобы ожидать более данные перед синхронизация и отключение розетки

**HTTPS_PORT** ( смотри страницу261) Определите слушание порта для сервера HTTPS (если STACK_USE_SSL_SERVER приспособлен)

**smHTTP** ( смотри страницу 263), Иметь доступ к текущей государственной машине

**RESERVED_HTTP_MEMORY** ( смотри страницу 263)Макро, указывающее сколько PAM, чтобы распределяться на диспетчере ethernet, чтобы загружать государственные данные HTTP.

**Структуры**

**Имя Description**

**HTTP_STUB** (255) смотри страничную Связь HTTP частичных государственных данных Загрузки Struct для каждой связи Предназначенной для памяти в быстром PAM доступа

**Переменные**

**Описание Имени**

**curHTTPID** ( СМОТРИ страницу 252) ID к настоящему времени загруженный

**HTTP_CONN** ( смотри страницу 240)

**httpContentTypes** ( смотри страницу 256)Довольный тип строк, соответствуемые на

**HTTP_FILE_TYPE** ( смотри страницу 253)
**httpFileExtensions** ( смотри страницу 256)

**HTTPRequestHeaders** ( смотри страницу 261)Строки Заголовка для которых мы хотели бы выполнять грамматический разбор

**HTTPResponseHeaders** ( смотри страницу 261) Начальные строки ответа (Соответствовать на HTTP_STATUS ( смотри страницу 254))

**httpStubs** ( смотри страницу 262) заглушки HTTP с государственной машиной и розеткой

**Description**
Следующее функционирует и переменные определены как внутренний в модуль HTTP2.

//*-*-*-*-*----------

### 10.8.4.1 curHTTPID Variable

**File**
HTTP2.c

**C**
BYTE **curHTTPID**;

**Description**

Идентификатор загруженной **HTTP_CONN** ( смотри страницу 240)

//*-*-*-*-*----------

### 10.8.4.2 HTTP_CACHE_LEN Macro

**File**
HTTP2.h

**C**
```
#define HTTP_CACHE_LEN ("600") //  целая жизнь Max (сек) статические ответы как строка
```

**Description**
Макс жизни Max (сек) статические ответы как строки

//*-*-*-*-*----------

### 10.8.4.3 HTTP_FILE_TYPE Enumeration

**File**
HTTP2.h

**C**
```
typedef enum {
HTTP_TXT = 0u,
HTTP_HTM,
HTTP_HTML,
HTTP_CGI,
HTTP_XML,
HTTP_CSS,
HTTP_GIF,
HTTP_PNG,
HTTP_JPG,
HTTP_JAVA,
HTTP_WAV,
HTTP_UNKNOWN
} HTTP_FILE_TYPE;
```

**Members**
**Members Description**
**HTTP_TXT** = 0u Файл является текстовым документа
**HTTP_HTM** - Файл HTML (расширение .htm)
**HTTP_HTML** - Файл HTML (расширение .html)
**HTTP_CGI** - Файл HTML (расширение .cgi)
**HTTP_XML** - Файл XML (расширение .xml)
**HTTP_CSS** - Файл  stylesheet (расширение .css)

**HTTP_GIF** Файл является образом GIF (расширение .gif)
**HTTP_PNG** - Файл образ PNG (расширение .png)
**HTTP_JPG** Файл является образом JPG (расширение .jpg)
**HTTP_JAVA** - Файл java (расширение .java)
**HTTP_WAV** Файл является звуком (расширение .wav)
**HTTP_UNKNOWN** тип Файла неизвестное

**Description**
Файловые определения типа

//*-*-*-*-*----------

### 10.8.4.4 HTTP_MAX_DATA_LEN Macro

**File**
TCPIP MRF24W.h

**C**
**#define HTTP_MAX_DATA_LEN** (100u)

**Description**
Определите максимальную длину данных для чтения печенья и ПОЛУЧАЙТЕ/ПОСЫЛАТЬ (GET/POST) аргументы (байты)

//*-*-*-*-*----------

### 10.8.4.5 HTTP_MAX_HEADER_LEN Macro

**File**
HTTP2.c

**C**
**#define HTTP_MAX_HEADER_LEN** (15u)

**Description**
Установите в длине длинной строки выше

//*-*-*-*-*----------

### 10.8.4.6 HTTP_MIN_CALLBACK_FREE Macro

**File**
TCPIP MRF24W.h

**C**

**#define HTTP_MIN_CALLBACK_FREE** (16u)

**Description**
Определите минимальное число байтов свободных в **TX FIFO** перед выполнением обратного вызова

//*-*-*-*-*----------

### 10.8.4.7 HTTP_PORT Macro

**File**
TCPIP MRF24W.h

**C**
**#define HTTP_PORT** (80u)

**Description**
Определите порт прослушивания для сервера HTTP

//*-*-*-*-*----------

### 10.8.4.8 HTTP_STATUS Enumeration

**File**
HTTP2.h

**C**
**typedef enum** {
**HTTP_GET** = 0u,
**HTTP_POST**,
**HTTP_BAD_REQUEST**,
**HTTP_UNAUTHORIZED**,
**HTTP_NOT_FOUND**,
**HTTP_OVERFLOW**,
**HTTP_INTERNAL_SERVER_ERROR**,
**HTTP_NOT_IMPLEMENTED**,
**HTTP_MPFS_FORM**,
**HTTP_MPFS_UP**,
**HTTP_MPFS_OK**,
**HTTP_MPFS_ERROR**,
**HTTP_REDIRECT**,
**HTTP_SSL_REQUIRED**
} HTTP_STATUS;

**Members**
**Members Description**
HTTP_GET = 0u GET command is being processed
HTTP_POST POST command is being processed
HTTP_BAD_REQUEST 400 Bad Request will be returned
HTTP_UNAUTHORIZED 401 Unauthorized will be returned

HTTP_NOT_FOUND 404 Not Found will be returned
HTTP_OVERFLOW 414 Request-URI Too Long will be returned
HTTP_INTERNAL_SERVER_ERROR 500 Internal Server Error will be returned
HTTP_NOT_IMPLEMENTED 501 Not Implemented (not a GET or POST command)
HTTP_MPFS_FORM Show the MPFS Upload form
HTTP_MPFS_UP An MPFS Upload is being processed
HTTP_MPFS_OK An MPFS Upload was successful
HTTP_MPFS_ERROR An MPFS Upload was not a valid image
HTTP_REDIRECT 302 Redirect will be returned
HTTP_SSL_REQUIRED 403 Forbidden is returned, indicating SSL is required

Http_get = команда 0u GET обрабатывается

Команда HTTP_POST POST обрабатывается

**HTTP_BAD_REQUEST** 400 Bad Request будут возвращены

**HTTP_UNAUTHORIZED** 401 Несанкционированное будут возвращены

**HTTP_NOT_FOUND** 404 Not Found будут возвращены

**HTTP_OVERFLOW** 414 Request-URI Too Long будут возвращены

**HTTP_INTERNAL_SERVER_ERROR** 500 Внутренняя ошибка сервера будут возвращены

**HTTP_NOT_IMPLEMENTED** 501 Не реализовано ( не GET или командный пункт )

**HTTP_MPFS_FORM** Покажите ГПСЛ Форма загрузки

**HTTP_MPFS_UP MPFS** Загрузить обрабатывается

**HTTP_MPFS_OK MPFS** Загрузить был успешным

**HTTP_MPFS_ERROR MPFS** Загрузить не правильный образ

**HTTP_REDIRECT** 302 Перенаправление будут возвращены

**HTTP_SSL_REQUIRED** 403 Forbidden возвращается , указывая SSL требуется

описание

**Description**
Предусмотренные Команды и Коды Ответа Сервера

//*-*-*-*-*----------

### 10.8.4.9 HTTP_STUB Structure

**File**
HTTP2.h

**C**
```
typedef struct {
SM_HTTP2 sm;
TCP_SOCKET socket;
} HTTP_STUB;
```

**Members**
**Members Description**
SM_HTTP2 sm; Текущее состояние соединения
TCP_SOCKET socket; Socket Разъем обслуживаемых

**Description**
HTTP Connection Struct Загружает частичные государственные данные состоянии для каждого соединения , предназначенные для хранения в оперативной памяти быстрого доступа

//*-*-*-*-*----------

### 10.8.4.10 HTTP_TIMEOUT Macro

**File**
HTTP2.h

**C**
```
#define HTTP_TIMEOUT (45u) // время Max (сек), чтобы ожидать более данные перед синхронизация и отключение розетки
```

**Description**
Max time (sec), чтобы ожидать больше данных перед синхронизация и отключение розетки

//*-*-*-*-*----------

### 10.8.4.11 httpContentTypes Variable

**File**
HTTP2.c

**C**
```
ROM char * ROM httpContentTypes[HTTP_UNKNOWN+1] = { "text/plain", "text/html",
"text/html",
"text/html", "text/xml", "text/css", "image/gif", "image/png", "image/jpeg",
"application/java-vm", "audio/x-wave", "" };
```

**Description**
Строки Content-Type , соответствующие HTTP_FILE_TYPE (см. стр. 253 )

//*-*-*-*-*----------

### 10.8.4.12 httpFileExtensions Variable

**File**
HTTP2.c

**C**
ROM **char** * ROM **httpFileExtensions**[HTTP_UNKNOWN+1] = { "txt", "htm", "html", "cgi", "xml",
"css", "gif", "png", "jpg", "cla", "wav", "\0\0\0" };

**Section**
Файл и Содержимое Набирают Установочные параметры
**********
Файловые расширения типа, соответствуемые на HTTP_FILE_TYPE

//*-*-*-*-*----------

### 10.8.4.13 HTTPHeaderParseAuthorization Function

**File**
HTTP2.c

**C**
**static void HTTPHeaderParseAuthorization**();

**Returns**
None

**Description**
Анализирует " Авторизация: " заголовок для запроса . Например, " BASIC YWRtaW46cGFzc3dvcmQ =" декодируется для пользователя наименование "Admin" и паролем " пароль". После того, как читать, HTTPCheckAuth ( см. стр. 242 ) вызывается из **CustomHTTPApp**.c чтобы определить , если учетные данные являются приемлемыми.

Возвращаемое значение **HTTPCheckAuth** ( см. стр. 242 ) сохраняется в **curHTTP**.**isAuthorized** для последующего использования приложением .

**Remarks**
Эта функция ony доступна когда HTTP_USE_AUTHENTICATION определен.

**Preconditions**
None

//*-*-*-*-*----------

### 10.8.4.14 HTTPHeaderParseContentLength Function

**File**
HTTP2.c

**C**
**static void HTTPHeaderParseContentLength**();

**Returns**
None

**Description**
Анализирует"Content-Length : " заголовок , чтобы определить, сколько байт данных POST ожидать после запроса. Это значение хранится в **curHTTP.byteCount**.

**Remarks**
Эта функция ony доступна когда HTTP_USE_POST определен.

**Preconditions**
None

//*-*-*-*-*----------

### 10.8.4.15 HTTPHeaderParseCookie Function

**File**
HTTP2.c

**C**
**static void HTTPHeaderParseCookie**();

**Returns**
None

**Description**
Выполняет грамматический разбор заголовки "Печенья:" для запроса. Например, "Печенье: name=Wile+E.+Coyote; order=ROCKET_LAUNCHER" декодирован на "name=Wile+E.+Coyote&order=ROCKET_LAUNCHER&" и загружен как любое другое ПОЛУЧАТЬ переменную в **curHTTP.data.**

Приложение пользователя может легко иметь доступ к этим величинам позже используя **HTTPGetArg** функций ( смотри страницу 244)() и **HTTPGetROMArg** ( смотри страницу 245)().

**Remarks**
Эта функция ony доступна когда HTTP_USE_COOKIES определен.

**Preconditions**
None

//*-*-*-*-*----------

### 10.8.4.16 HTTPHeaderParseLookup Function

**File**
HTTP2.c

**C**
```
static void HTTPHeaderParseLookup(
BYTE i
);
```

**Description**
Вызывает подходящий синтаксический анализатор заголовка основанный в индексе заголовка, который был прочитан с запроса.

**Preconditions**
None

**Parameters**
**Parameters Description**
i - индекс строки обнаруженной в **HTTPRequestHeaders** ( смотри страницу 261)

**Return Values**
**Return Values Description**
**TRUE** конец файла был достигнут и чтение сделано
**FALSE** больше данных остается для чтения

**Section**
Function Prototypes

```
//*-*-*-*-*-------------------------------------------
```

### 10.8.4.17 HTTPIncFile Function

**File**
HTTP2.h

**C**
```
void HTTPIncFile(
ROM BYTE* cFile
);
```

**Returns**
None

**Description**
Позволяет весь файл , которые будут включены в качестве динамической переменной , обеспечивая базовый систему шаблонов для веб-страниц HTML. это уменьшает ненужные дублирования визуальных элементов , таких как заголовки , меню и т.д.

Когда **curHTTP.callbackPos** равен 0, то файл будет открыт и столько байт , сколько возможно написаны. Текущее положение , то сохраняются в **curHTTP.callbackPos** и файл закрывается . При последующих вызовах , чтение начинается с сохраненной месте и продолжается. После того, как в конце входного файла достигается , **curHTTP.callbackPos** установлен на 0 , чтобы указать, к завершению.

**Remarks**
Пользователи не должны вызывать эту функцию непосредственно, но должны взамен добавить динамические переменные в форме ~**inc**:**filename**.**ext**~ в их коде HTML, чтобы включать (например) файл "**filename**.**ext**" в этом определял позицию. Утилита Генератора MPFS2 прооперируется остальные.

**Preconditions**
None

**Parameters**
**Parameters Description**
имя файла, который нужно посылать

//*-*-*-*-*-----------------------------------------

### 10.8.4.18 HTTPLoadConn Function

**File**
HTTP2.c

**C**
```
static void HTTPLoadConn(
BYTE hHTTP
);
```

**Returns**
None

**Description**
Сохраняет текущий загруженный соединение HTTP к буферной памяти Ethernet , затем загружает выбранное соединение в **curHTTP** (См. стр. 240) в местной памяти для обработки.

**Preconditions**
None

**Parameters**
**Parameters Description**
**hHTTP** связь ID, чтобы загружаться

//*-*-*-*-*-----------------------------------------

### 10.8.4.19 HTTPMPFSUpload Function

**File**
HTTP2.c

**C**
**static** HTTP_IO_RESULT **HTTPMPFSUpload**();

**Description**
Допускает образ MPFS в EEPROM или внешнюю Вспышку, которые будут обновляться через веб-страницы , принимая загрузить файл и хранить его к внешней памяти.

**Remarks**

Эта функция доступна только тогда, когда **MPFS** закачка разрешена , и изображение **MPFS** хранится в EEPROM.

**Internal**
После заголовков , первая линия от формы будет MIME сепаратор . После это больше заголовков о файле , которые отбрасываются. После другой пары CRLFCRLF начинается данные файла , который читается 16 байт за один раз и записаны на внешний памяти .

**Preconditions**
MPFSFormat ( see page 274)() has been called.

**Return Values**
**Return Values Description**
HTTP_IO_DONE в успехе
HTTP_IO_NEED_DATA если более данные все еще ожиданы

//*-*-*-*-*----------

### 10.8.4.20 HTTPProcess Function

**File**
HTTP2.c

**C**
**static void HTTPProcess**();

**Returns**
None

**Description**
Выполняет все ожидающие операции для загруженного в данный момент связи HTTP.

**Preconditions**

**HTTPInit** ( СМОТРИ страницу 250)() и **HTTPLoadConn** ( смотри страницу 259)(), вызван.

//*-*-*-*-*----------

### 10.8.4.21 HTTPReadTo Function

**File**
HTTP2.c

**C**
```
static HTTP_READ_STATUS HTTPReadTo(
BYTE delim,
BYTE* buf,
WORD len
);
```

**Description**
Читает из буфера TCP на **cData** пока либо **cDelim** не будет достигнуто , или пока **Wlen** - 2 байта не были прочитаны. Значение читать, сохраняются в **cData** и нулем . ( **Wlen** - 2 используется таким образом, что значение может быть передано в **HTTPURLDecode** (см. стр.249 ) позже , что требует нулевой терминатор плюс один дополнительный свободный байт . )

Символ разделителя удаляется из буфера , но не сохраняются в **cData**. Если все данные не могут вписаться в **cData**, она все равно будет удаляется из буфера , но не будут сохранены в любом месте.

Эта функция правильно обновляет **curHTTP.byteCount** путем уменьшения его на количество считанных байтов .

**cData**
**Preconditions**
None

**Parameters**
**Parameters Description**
**cDelim** Символ при котором прекратить читать , или NULL читать до конца буфера
**cData** где хранить читается данные
**wLen** сколько байтов может быть записано на **cData**

**Return Values**
**Return Values Description**
**HTTP_READ_OK** данные были успешно прочитаны
**HTTP_READ_TRUNCTATED** все данные не могли поместиться в буфере , поэтому данные были усечены и данных имеет был потерян

**HTTP_READ_INCOMPLETE** символ-разделитель не найден

//*-*-*-*-*----------

### 10.8.4.22 HTTPRequestHeaders Variable

**File**
HTTP2.c

**C**
ROM **char** * ROM **HTTPRequestHeaders**[] = { "Cookie:", "Authorization:", "Content-Length:" };

**Description**
Строки Заголовка для которых мы хотели бы выполнять грамматический разбор

//*-*-*-*-*----------

### 10.8.4.23 HTTPResponseHeaders Variable

**File**
HTTP2.c

**C**
ROM **char** * ROM **HTTPResponseHeaders**[] = { "HTTP/1.1 200 OK\r\nConnection: close\r\n", "HTTP/1.1 200 OK\r\nConnection: close\r\n", "HTTP/1.1 400 Bad Request\r\nConnection: close\r\n\r\n400 Bad Request: can't handle Content-Length\r\n", "HTTP/1.1 401 Unauthorized\r\nWWW-Authenticate: Basic realm=\"Protected\"\r\nConnection: close\r\n\r\n401 Unauthorized: Password required\r\n", "HTTP/1.1 404 Not found\r\nConnection: close\r\nContent-Type: text/html\r\n\r\n404: File not found<br>Use <a href=\"/" HTTP_MPFS_UPLOAD "\">MPFS Upload</a> to program web pages\r\n", "HTTP/1.1 404 Not found\r\nConnection: close\r\n\r\n404: File not found\r\n", "HTTP/1.1 414 Request-URI Too Long\r\nConnection: close\r\n\r\n414 Request-URI Too Long: Buffer overflow detected\r\n", "HTTP/1.1 500 Internal Server Error\r\nConnection: close\r\n\r\n500 Internal Server Error: Expected data not present\r\n", "HTTP/1.1 501 Not Implemented\r\nConnection: close\r\n\r\n501 Not Implemented: Only GET and POST supported\r\n", "HTTP/1.1 200 OK\r\nConnection: close\r\nContent-Type: text/html\r\n\r\n<html><body style=\"margin:100px\"><form method=post action=\"/" HTTP_MPFS_UPLOAD "\" enctype=\"multipart/form-data\">MPFS Image Upload<p><input type=file name=i size=40>
  <input type=submit value=\"Upload\"></form></body></html>", "", "HTTP/1.1 200 OK\r\nConnection: close\r\nContent-Type: text/html\r\n\r\n<html><body style=\"margin:100px\">MPFS Update Successful<p><a href=\"/\">Site main page</a></body></html>", "HTTP/1.1 500 Internal Server Error\r\nConnection: close\r\nContent-Type: text/html\r\n\r\n<html><body style=\"margin:100px\">MPFS Image Corrupt or Wrong Version<p><a href=\"/" HTTP_MPFS_UPLOAD "\">Try again?</a></body></html>", "HTTP/1.1 302 Found\r\nConnection: close\r\nLocation: ", "HTTP/1.1 403 Forbidden\r\nConnection: close\r\n\r\n403 Forbidden: SSL Required - use HTTPS\r\n" };

СИМВОЛ ROM * ROM HTTPResponseHeaders[] = { "HTTP/1.1 200 OK\r\nConnection: close\r\n", "HTTP/1.1 200 OK\r\nConnection: close\r\n", "HTTP/1.1 400 Плохо Request\r\nConnection:
close\r\n\r\n400 Плохой Запрос: не может оперировать Content-Length\r\n", "HTTP/1.1 401 Unauthorized\r\nWWW-Удостоверять: Basic realm=\"Protected\"\r\nConnection: close\r\n\r\n401 Несанкционированный: Пароль required\r\n", "HTTP/1.1 404 Не found\r\nConnection: close\r\nContent-Тип: text/html\r\n\r\n404: Файл не found<br>Use < href=\"/" HTTP_MPFS_UPLOAD "\">MPFS Upload</a>, чтобы программировать веб pages\r\n", "HTTP/1.1 404 Не found\r\nConnection: close\r\n\r\n404: Файл не found\r\n", "HTTP/1.1 414 Request-URI Слишком Long\r\nConnection: close\r\n\r\n414 Request-URI Слишком Долго (длиной): переполнение Буфера detected\r\n", "HTTP/1.1 500 Internal Server Error\r\nConnection: close\r\n\r\n500 Внутренняя Ошибка Сервера: Ожидаемые данные не present\r\n", "HTTP/1.1 501 Не Implemented\r\nConnection: close\r\n\r\n501 Не Осуществленное: Только ПОЛУЧИТЕ и ПОСЫЛАЙТЕ supported\r\n", "HTTP/1.1 200 OK\r\nConnection: close\r\nContent-Тип: text/html\r\n\r\n<html><body style=\"margin method=post действия=:100px\"><form\"/" HTTP_MPFS_UPLOAD "\" enctype=\"multipart/form-data\">MPFS Image Upload<p><input type=file name=i size=40>   <ввод type=submit value=\"Upload\"></form></body></html>", "", "HTTP/1.1 200 OK\r\nConnection: close\r\nContent-Тип: text/html\r\n\r\n<html><body style=\"margin:Коррекция 100px\">MPFS Successful<p><a href=\"/основа \">Site page</a></body></html>", "HTTP/1.1 500 Internal Server Error\r\nConnection: close\r\nContent-Тип: text/html\r\n\r\n<html><body style=\"margin:Образ 100px\">MPFS Испорченый или Неправильный Version<p><a href=\"/" HTTP_MPFS_UPLOAD "\">Попытка again?</a></body></html>", "HTTP/1.1 302 Found\r\nConnection: close\r\nLocation: ", "HTTP/1.1 403 Forbidden\r\nConnection: close\r\n\r\n403 Запрещенное: SSL Требовал - использование HTTPS\r\n" };

**Description**
Начальные строки реагирования ( В соответствии с HTTP_STATUS (см. стр. 254 ) )

//*-*-*-*-*----------

### 10.8.4.24 HTTPS_PORT Macro

**File**
TCPIP MRF24W.h

**C**
```
#define HTTPS_PORT (443u)
```

**Description**
Определите порт прослушивания для сервера HTTPS (если STACK_USE_SSL_SERVER включена )

//*-*-*-*-*----------

### 10.8.4.25 HTTPSendFile Function

**File**
HTTP2.c

**C**
```
static BOOL HTTPSendFile();
```

**Description**
Служит на следующий кусок **curHTTP** ( см. стр. 240 ) , изменив файл , до) доступного Техас FIFO пространство или б) следующий обратный вызов индекс , что наступит раньше .

**Preconditions**
**curHTTP**.**file** и **curHTTP**.**offsets** оба были открыт для чтения .

**Return Values**
**Return Values Description**
**TRUE** конец файла был достигнут и чтение сделано
**FALSE** больше данных остается для чтения

```
//*-*-*-*-*-----------------------------------------
```

### 10.8.4.26 httpStubs Variable

**File**
HTTP2.c

**C**
```
HTTP_STUB httpStubs[MAX_HTTP_CONNECTIONS];
```

**Description**
HTTP заглушки с государственной машиной и розетки socket

```
//*-*-*-*-*-----------------------------------------
```

### 10.8.4.27 SM_HTTP2 Enumeration

**File**
HTTP2.h

**C**
```
typedef enum {
SM_HTTP_IDLE = 0u,
SM_HTTP_PARSE_REQUEST,
SM_HTTP_PARSE_HEADERS,
SM_HTTP_AUTHENTICATE,
SM_HTTP_PROCESS_GET,
```

**SM_HTTP_PROCESS_POST**,
**SM_HTTP_PROCESS_REQUEST**,
**SM_HTTP_SERVE_HEADERS**,
**SM_HTTP_SERVE_COOKIES**,
**SM_HTTP_SERVE_BODY**,
**SM_HTTP_SEND_FROM_CALLBACK**,
**SM_HTTP_DISCONNECT**
} SM_HTTP2;

**Members**
**Members Description**
SM_HTTP_IDLE = 0u Socket is idle
SM_HTTP_PARSE_REQUEST Parses the first line for a file name and GET args
SM_HTTP_PARSE_HEADERS Reads and parses headers one at a time
SM_HTTP_AUTHENTICATE Validates the current authorization state
SM_HTTP_PROCESS_GET Invokes user callback for GET args or cookies
SM_HTTP_PROCESS_POST Invokes user callback for POSTed data
SM_HTTP_PROCESS_REQUEST Begins the process of returning data
SM_HTTP_SERVE_HEADERS Sends any required headers for the response
SM_HTTP_SERVE_COOKIES Adds any cookies to the response
SM_HTTP_SERVE_BODY Serves the actual content
SM_HTTP_SEND_FROM_CALLBACK Invokes a dynamic variable callback
SM_HTTP_DISCONNECT Disconnects the server and closes all files

Участники Description
SM_HTTP_IDLE = 0u Розетка холостой
SM_HTTP_PARSE_REQUEST Выполняет грамматический разбор первую строку для файлового имени и ПОЛУЧАЕТ арг.
SM_HTTP_PARSE_HEADERS Читает и выполняет грамматический разбор заголовки поочередно
SM_HTTP_AUTHENTICATE Подтверждает текущее состояние разрешения
SM_HTTP_PROCESS_GET Вводит возврат пользователя для ПОЛУЧАТЬ арг. или cookies
SM_HTTP_PROCESS_POST Вводит возврат пользователя для ОБЪЯВЛЕННЫХ данных
SM_HTTP_PROCESS_REQUEST Начинает процесс обратных данных
SM_HTTP_SERVE_HEADERS Посылать любые необходимые заголовки для ответа
SM_HTTP_SERVE_COOKIES Добавляет любые cookies в ответ
SM_HTTP_SERVE_BODY Обслуживает фактическое содержимое
SM_HTTP_SEND_FROM_CALLBACK Вводится, динамический переменный возврат
SM_HTTP_DISCONNECT Разъединяет сервер и закрывает все файлы

**Description**
Базовая HTTP соединения конечного автомата

Основная Государственная Машина Связи HTTP

//*-*-*-*-*----------

### 10.8.4.28 smHTTP Macro

**File**
HTTP2.c

**C**
```
#define smHTTP httpStubs[curHTTPID].sm //  текущая государственная машина
```

**Description**
Иметь доступ к текущей государственной машине

//*-*-*-*-*----------

### 10.8.4.29 RESERVED_HTTP_MEMORY Macro

**File**
HTTP2.h

**C**
```
#define RESERVED_HTTP_MEMORY ((DWORD)MAX_HTTP_CONNECTIONS *
(DWORD)sizeof(HTTP_CONN))
```

**Description**
Макро с указанием , сколько оперативной памяти выделить на контроллер Ethernet для хранения HTTP данные о состоянии .

//*-*-*-*-*----------

## 10.9 ICMP

Протокол Сообщение управления Интернет используется для отправки ошибок и статусных сообщений и запросов . Модуль ICMP реализует Echo тип Ответить сообщения ( как правило, называют пингом ) , который может использоваться , чтобы определить , является ли указанный

хоста в качестве IP- сети с устройства под управлением стек TCP / IP. Сервер ICMP также поддерживается в реагировать на проверок на других устройствах.

//*-*-*-*-*----------

### 10.9.1 ICMP Public Members

**Functions**
**Name Description**
ICMPBeginUsage ( see

page 264)
Claims ownership of the ICMP module.
ICMPSendPing ( see page
265)
None
ICMPSendPingToHost (
see page 265)
None
ICMPSendPingToHostROM
( see page 266)
Begins the process of transmitting an ICMP echo request. This normally
involves an ARP resolution procedure first.
ICMPGetReply ( see page
266)
None
ICMPEndUsage ( see page
267)
Gives up ownership of the ICMP module.

**Macros**
**Name Description**
ICMPSendPingToHostROM
( see page 267)
This is macro ICMPSendPingToHostROM.

**Module**
ICMP ( see page 263)

**Description ICMPBeginUsage** ( смотри страницу 264)Собственность Претензий на модуль ICMP.
**ICMPSendPing** ( СМОТРИ страницу265) Ничто
**ICMPSendPingToHost** ( СМОТРИ страницу 265) Ничто
**ICMPSendPingToHostROM** ( СМОТРИ страницу 266)Начинает процесс запроса эха передачи ICMP. Это нормально включает разрешение процедуры ARP сначала.
**ICMPGetReply** ( СМОТРИ страницу266) Ничто
**ICMPEndUsage** ( СМОТРИ страницу267) Оставляет собственность на модуль ICMP.

Макро

Имя Description ICMPSendPingToHostROM ( смотри страницу 267)
Это - макро ICMPSendPingToHostROM.

Модуль

ICMP ( СМОТРИ страницу 263)

**Description**
Следующие функции и переменные доступны или реализованы путем применения стека.

//*-*-*-*-*-----------------------------------------

#### 10.9.1.1 ICMPBeginUsage Function

**File**
ICMP.h

**C**
BOOL **ICMPBeginUsage**();

**Side Effects**
None

**Returns**
**TRUE**: Вы успешно приобрели собственность на модуль клиента ICMP и может теперь использовать ICMPSendPing функций ( смотри страницу 265)() и ICMPGetReply ( смотри страницу 266)().

**FALSE**: : Некоторое другое приложение использует модуль клиента ICMP. Разговор ICMPSendPing ( смотри страницу 265)(), исказит другой прикладной результат звона.

**Description**
Собственность Претензий на модуль ICMP.

**Remarks**
None

**Preconditions**
None

//*-*-*-*-*-----------------------------------------

#### 10.9.1.2 ICMPSendPing Function

**File**
ICMP.h

**C**
**void ICMPSendPing**(
DWORD **dwRemoteIP**
);

**Side Effects**
None

**Returns**
Начинается процесс передачи эхо -запроса ICMP . Как правило, это включает в себя сначала процедуру урегулирования ARP.

**Description**
None

**Remarks**
None

**Preconditions**
ICMPBeginUsage ( СМОТРИ страницу 264)() возвращавшее ИСТИНУ

**Parameters**
**Parameters Description**
**dwRemoteIP** IP-адрес ( см. стр. 147) для пинг . Должны храниться Big Endian . Экс . 192.168.0.1 должен быть передан как 0x0100A8C0 .

//*-*-*-*-*----------

### 10.9.1.3 ICMPSendPingToHost Function

**File**
ICMP.h

**C**
```
void ICMPSendPingToHost(
BYTE * szRemoteHost
);
```

**Side Effects**
None

**Returns**
Начинается процесс передачи эхо -запроса ICMP . Как правило, это включает в себя сначала процедуру урегулирования ARP.

**Description**
None

**Remarks**
None

**Preconditions**
ICMPBeginUsage ( see page 264)() returned TRUE
ICMPBeginUsage ( СМОТРИ страницу 264)() возвращавшее TRUE

**Parameters**
**Parameters Description**
szRemoteHost Имя хоста для пинг. Должны храниться в оперативной памяти , если вызывается на PIC18 . Экс .

//*-*-*-*-*-----------------------------------------

### 10.9.1.4 ICMPSendPingToHostROM Function

**File**
ICMP.h

**C**
```
void ICMPSendPingToHostROM(
ROM BYTE * szRemoteHost
);
```

**Side Effects**
None

**Returns**
None

**Description**
Начинается процесс передачи эхо -запроса ICMP . Как правило, это включает в себя сначала процедуру урегулирования ARP.

**Remarks**
None

**Preconditions**
ICMPBeginUsage ( СМОТРИ страницу 264)() возвращавшее TRUE

**Parameters**
**Parameters Description**
szRemoteHost Имя хоста для пинг. Должны храниться в ПЗУ. Должно только вызвано PIC18. www.microchip.com Ex.

//*-*-*-*-*-----------------------------------------

### 10.9.1.5 ICMPGetReply Function

**File**
ICMP.h

**C**
```
LONG ICMPGetReply();
```

**Side Effects**
None

**Returns**
-3 : Не удалось разрешить имя хоста ( тайм-аута DNS или имя хоста недействительный )

-2 : Ответа не получил еще

-1 : операции истекло

( больше , чем ICMP_TIMEOUT (см. стр. 270) ) истекло.

> = 0 : количество тиков , что прошло между начальной ICMP

**Description**
None

**Remarks**
None

**Preconditions**
ICMPBeginUsage (см. стр. 264 ) () вернула TRUE, и ICMPSendPing (см. стр. 265 ) () была вызвана

//*-*-*-*-*----------

### 10.9.1.6 ICMPEndUsage Function

**File**
ICMP.h

**C**
```
void ICMPEndUsage();
```

**Side Effects**
None

**Returns**
Ваш владение модуля ICMP будет отпущена. Вы больше не можете использовать ICMPSendPing (см.стр. 265 ) ( ) .

**Description**
Оставляет собственность на модуль ICMP.

**Remarks**
None

**Preconditions**
ICMPBeginUsage ( СМОТРИ страницу 264)(), был вызван Вами и возвращано TRUE.

//*-*-*-*-*----------

#### 10.9.1.7 ICMPSendPingToHostROM Macro

**File**
ICMP.h

**C**
```
#define ICMPSendPingToHostROM(a) ICMPSendPingToHost((BYTE*)(a))
```

**Description**
This is macro ICMPSendPingToHostROM.

//*-*-*-*-*----------

### 10.9.2 ICMP Internal Members

**Functions**
**Name Description**
ICMPProcess ( see page 268)
None

**Macros**
**Name Description**
ICMP_TIMEOUT ( seepage 270)
ICMP Timeout Value

**Module**
ICMP ( see page 263)

**Structures**
**Name Description**
ICMP_PACKET ( see page269)
ICMP Packet Structure

**Variables**
**Name Description**
ICMPFlags ( see page 269) ICMP Flag structure
ICMPState ( see page 269) ICMP State Machine Enumeration
ICMPTimer ( see page 270) ICMP tick timer variable
StaticVars ( see page 270) ICMP Static Variables
wICMPSequenceNumber (see page 271)ICMP Sequence Number

ICMPFlags ( СМОТРИ страницу 269) Структура ICMP Флаг
ICMP ICMPState ( смотри страницу 269) ICMP государственной машины перечисления
ICMPTimer ( смотри страницу 270) ICMP переменная таймера тик
ICMP StaticVars ( смотри страницу 270) ICMP статические переменные
ICMP wICMPSequenceNumber ( смотри страницу 271) ICMP Порядковый номер

**Description**
Следующее функционирует и переменные определены как внутренний в модуль ICMP.

//*-*-*-*-*----------

### 10.9.2.1 ICMPProcess Function

**File**
ICMP.h

**C**
```
void ICMPProcess(
NODE_INFO * remote,
WORD len
);
```

**Side Effects**
None

**Returns**
Генерирует ответ эха, если запрошено Подтверждает и устанавливает ICMPFlags.bReplyValid если правильный ответ на пинг одно из наших получен.

**Description**
None

**Remarks**
None

**Preconditions**
БУФЕР MAC содержит пакет типа ICMP.

**Parameters**
**Parameters Description**
*дистанционный Указатель в структуру NODE_INFO len Счета запрашивающей стороны пинг того , сколько байт заголовка пинг и полезной нагрузки в этом пакете IP

//*-*-*-*-*----------

### 10.9.2.2 ICMPFlags Variable

**File**
ICMP.c

**C**
```
struct {
unsigned char bICMPInUse : 1;
unsigned char bReplyValid : 1;
```

```
unsigned char bRemoteHostIsROM : 1;
} ICMPFlags;
```

**Members**
**Members Description**
unsigned char bICMPInUse : 1; Указывает, что Клиент ICMP используется без знака

unsigned char bReplyValid : 1; 1 ; Указывает, что правильный ответ Пинг на один из наших пингов было получено

unsigned char bRemoteHostIsROM : 1; Указывает, что имя удаленного хоста был принят в качестве аргумента указателя ROM

**Description**
ICMP Flag structure

//*-*-*-*-*----------

### 10.9.2.3 ICMP_PACKET Structure

**File**
ICMP.c

**C**

```
typedef struct {
BYTE vType;
BYTE vCode;
WORD wChecksum;
WORD wIdentifier;
WORD wSequenceNumber;
WORD wData;
} ICMP_PACKET;
```

**Description**
ICMP Packet Structure

//*-*-*-*-*----------

### 10.9.2.4 ICMPState Variable

**File**
ICMP.c

**C**

```
enum {
SM_IDLE = 0,
SM_DNS_SEND_QUERY,
SM_DNS_GET_RESPONSE,
SM_ARP_SEND_QUERY,
SM_ARP_GET_RESPONSE,
```

**SM_ICMP_SEND_ECHO_REQUEST**,
**SM_ICMP_GET_ECHO_RESPONSE**
} ICMPState;

**Description**
ICMP State Machine Enumeration

//*-*-*-*-*----------

### 10.9.2.5 ICMP_TIMEOUT Macro

**File**
ICMP.c

**C**
```
#define ICMP_TIMEOUT (4ul*TICK_SECOND)
```

**Description**
ICMP Timeout Value
ВРЕМЯ Простоя ICMP

//*-*-*-*-*----------

### 10.9.2.6 ICMPTimer Variable

**File**
ICMP.c

**C**
DWORD **ICMPTimer**;

**Description**
ICMP переменная тик таймера

//*-*-*-*-*----------

### 10.9.2.7 StaticVars Variable

**File**
ICMP.c

**C**
```
union {
union {
ROM BYTE * szROM;
BYTE * szRAM;
} RemoteHost;
NODE_INFO ICMPRemote;
} StaticVars;
```

**Description**
ICMP Static Variables

//*-*-*-*-*----------

### 10.9.2.8 wICMPSequenceNumber Variable

**File**
ICMP.c

**C**
WORD **wICMPSequenceNumber**;

**Description**
НОМЕР Последовательности ICMP

//*-*-*-*-*----------

## 10.10 MPFS2

Модуль MPFS2 файловая система обеспечивает облегченную только для чтения файловой системы , которые могут храниться во внешнем EEPROM , внешний серийный вспышки , или внутренняя флэш-память программа . Эта файловая система служит в качестве основы для веб-сервера http2 Модуль , но также используется модулем SNMP и доступен для других приложений, требующих базового хранения только для чтения возможности.

Модуль MPFS2 включает MPFS2 утилита, которая работает на вашем компьютере. Эта программа строит MPFS2 фотографий в различных форматах для использования в поддерживаемых носителей информации . Более подробная информация доступна в MPFS2 Utility ( см. стр. 63) раздел .

**Использование Внешние накопители**

Для внешних накопителей ,MPFS2 файловая система поддерживает Microchip 25LCxxx EEPROM автозапчасти для плотностей до 1Мбит . SST 25VFxxxB серийные детали Флэш также поддерживаются для плотностей до 32 Мбит .

Для использования внешних накопителей EEPROM , убедитесь, что макроконфигурация MPFS_USE_EEPROM определяется в TCPIPConfig.h . Если вы используете часть 1 Мбит ( 25LC1024 ) , также не забудьте определить USE_EEPROM_25LC1024 для того, чтобы 24-битный устройство адресации используется этой части. Для внешнего последовательного Flash, определить MFPS_USE_SPI_FLASH вместо EEPROM макросы .

Образы внешне выгружаются через HTTP . Это может быть достигнуто с помощью MPFS2 Utility , или могут быть доступны прямо из браузера . Загрузка файлов напрямую ( см. стр.

64) описан в документации MPFS2 Utility. Загрузка Изображения с помощью HTTP может быть достигнуто , как описано вПриступая к работе ( см. стр. 78) раздел .

При хранении фотографий внешне, пространство может быть зарезервированы для отдельного использования приложений. Макроконфигурация MPFS_RESERVE_BLOCK контролирует размер этого пространства . Указанное число байтов будут защищены в начале адрес устройства хранения (0x000000 ). При использовании последовательной флэш-памяти , этот адрес должен быть кратен размеру флэш сектора ( 4096 байт) .

**Использование встроенной флеш- хранилища**

При хранении фотографий в внутренней памяти флэш программы , новые образы не могут быть загружены во время выполнения. Вместо этого изображение составлен в рамках вашего проекта в MPLAB IDE . Для выбора этого варианта хранения комментарий из конфигурации макроса MPFS_USE_EEPROM в TCPIPConfig.h , то убедитесь, что файл изображения , порожденный MPFS2 утилита входит в проект MPLAB .

**Другие вопросы**

MPFS2 задает фиксированное число файлов, которые могут быть открыты одновременно. Параметр конфигурации MAX_MPFS_HANDLES контролирует , сколько файлов может быть открыт сразу. Если этот ресурс будет исчерпан, никакие новые файлы не могут быть открыта до MPFSClose (см. стр. 274 ) не будет вызван для существующего ручкой. Веб-сервер http2 ожидает , чтобы иметь возможность использовать по крайней мере, две ручки для каждого соединения , плюс один дополнительный . Дополнительные ручки должны быть выделены , если другие модули будут доступа к файловой системе , а также.

//*-*-*-*-*----------

### 10.10.1 MPFS2 Public Members

**Enumerations**
**Name Description**
MPFS_SEEK_MODE ( страница 273)
смотри Указывает метод для MPFSSeek ( смотри страницу 283)

**Functions**
**Name Description**
**MPFSClose** ( СМОТРИ страницу 274), Закрывает файл.
**MPFSFormat** ( СМОТРИ страницу274) Подготавливает образ MPFS к записи.
**MPFSGet** ( СМОТРИ страницу 275), Читает байт из файла.
**MPFSGetArray** ( СМОТРИ страницу275) Читает серию байтов из файла.
**MPFSGetBytesRem** ( СМОТРИ страницу 276)Определяет что сколько байтов остаются, чтобы прочитан.
**MPFSGetEndAddr** ( СМОТРИ страницу 276)Определяет окончание адреса файла.
**MPFSGetFilename** ( СМОТРИ страницу 277)Читает файловое имя файла, который уже открытый.
**MPFSGetFlags** ( СМОТРИ страницу277) Читает файловые флаги.

**MPFSGetID** ( СМОТРИ страницу 278), Определяет ID в ЖИРЕ для файла.
**MPFSGetLong** ( СМОТРИ страницу278) Читает DWORD или Длинную величину из MPFS.
**MPFSGetMicrotime** ( СМОТРИ страницу 279)Читает часть microtime файловой отметки времени.
**MPFSGetPosition** ( СМОТРИ страницу 279)Определяет текущую позицию в файле
**MPFSGetSize** ( смотри страницу279) Читает размер файла.
**MPFSGetStartAddr** ( СМОТРИ страницу 280)Читает стартовый адрес файла.
**MPFSGetTimestamp** ( СМОТРИ страницу 280)Читает отметку времени для определенного файла.
**MPFSOpen** ( СМОТРИ страницу 281), Открывает файл в файловой системе MPFS2.
**MPFSOpenID** ( СМОТРИ страницу281) Быстро см.- открывать файл.
**MPFSOpenROM** ( СМОТРИ страницу 282)Открывает файл в файловой системе MPFS2.
**MPFSPutArray** ( СМОТРИ страницу282) Записывает массив данных в образ MPFS.
**MPFSSeek** ( СМОТРИ страницу 283), Перемещает ток прочитавший указатель в новую позицию.
**MPFSPutEnd** ( СМОТРИ страницу283) Завершает MPFS, пишущее операцию.

**Macros**
**Name Description**
Имя Description

**MPFS_INVALID** (273) видьте как страница Указывала что указатель позиции недействителен

**MPFS_INVALID_HANDLE** ( see page 273)Указывает, что ручка будет недействительным Модулем

**Module**
MPFS2 ( see page 271)

**Types**
**Name Description**

**MPFS_HANDLE** ( see page273)смотри что страничные Ручки MPFS к настоящему времени загружены как БАЙТЫ

**Description**
Следующее функционирует и переменные доступны приложением стека.

//*-*-*-*-*----------

### 10.10.1.1 MPFS_HANDLE Type

**File**
MPFS2.h

**C**
```
typedef BYTE MPFS_HANDLE;
```

**Description**
MPFS Ручки в настоящее время хранится в виде байт

//*-*-*-*-*----------

### 10.10.1.2 MPFS_INVALID Macro

**File**
MPFS2.h

**C**
```
#define MPFS_INVALID (0xffffffffu) // Указывает что указатель позиции недействителен
```

**Description**
Указывает что указатель позиции недействителен

//*-*-*-*-*----------

### 10.10.1.3 MPFS_INVALID_HANDLE Macro

**File**
MPFS2.h

**C**
```
#define MPFS_INVALID_HANDLE (0xffu) // Указывает, что ручка недействительна
```

***Description***
Указывает, что ручка не действует

//*-*-*-*-*----------

### 10.10.1.4 MPFS_SEEK_MODE Enumeration

**File**
MPFS2.h

**C**
```
typedef enum {
MPFS_SEEK_START = 0u,
MPFS_SEEK_END,
MPFS_SEEK_FORWARD,
MPFS_SEEK_REWIND
} MPFS_SEEK_MODE;
```

**Members**
**Members Description**

MPFS_SEEK_START = 0u Seek forwards from the front of the file
MPFS_SEEK_END Seek backwards from the end of the file
MPFS_SEEK_FORWARD Seek forward from the current position
MPFS_SEEK_REWIND See backwards from the current position

MPFS_SEEK_START = 0u Ищите вперед из передней части файла
MPFS_SEEK_END Обратиться в обратном порядке от конца файла
MPFS_SEEK_FORWARD Перемотка вперед от текущей позиции
MPFS_SEEK_REWIND Смотреть назад от текущей позиции

**Description**
Указывает метод для MPFSSeek ( смотри страницу 283)

//*-*-*-*-*----------

### 10.10.1.5 MPFSClose Function

**File**
MPFS2.h

**C**
```
void MPFSClose(
MPFS_HANDLE hMPFS
);
```

**Returns**
None

**Description**
Закрывает файл и выпускает свою заглушку в пул доступных ручек.

**Preconditions**
None

**Parameters**
**Parameters Description**
hMPFS ( see page 348) дескриптор файла должен быть закрыт

//*-*-*-*-*----------

### 10.10.1.6 MPFSFormat Function

**File**
MPFS2.h

**C**
```
MPFS_HANDLE MPFSFormat();
```

**Returns**
MPFS, дескриптор, который может быть использован для **MPFSPut** команд или **MPFS_INVALID_HANDLE** (см. стр. 273 ), когда EEPROM не удалось инициализировать для записи.

**Description**
Готовит изображение MPFS для записи и блокирует изображение таким образом, что другие процессы не могут получить к нему доступ .

**Remarks**
В целях предотвращения misreads , MPFS будут недоступны , пока **MPFSClose** (см. стр. 274 ) называется. Эта функция не доступны, когда MPFS хранится во внутренней памяти флэш программы.

**Preconditions**
None

//*-*-*-*-*----------

### 10.10.1.7 MPFSGet Function

**File**
MPFS2.h

**C**
BOOL **MPFSGet**(
MPFS_HANDLE **hMPFS**,
BYTE* **c**
);

**Description**
Читает байт из файла.

**Preconditions**
Файловая ручка ссылавшаяся hMPFS уже открытая.

**Parameters**
**Parameters Description**
hMPFS ( СМОТРИ страницу 348) дескриптор файла из которого будут считываться Где, чтобы загружать байт, который был прочитан

**Return Values**
**Return Values Description**
**TRUE** байт успешно был прочитан
**FALSE** No байт был прочитан поскольку или ручка была недействительна или конец файла достигнут.

//*-*-*-*-*-----------------------------------------

### 10.10.1.8 MPFSGetArray Function

**File**
MPFS2.h

**C**
WORD **MPFSGetArray**(
MPFS_HANDLE **hMPFS**,
BYTE* **cData**,
WORD **wLen**
);

**Returns**
Количество байт, успешно читать. Если это меньше Wlen ,EOF произошло при попытке чтения .

**Description**
Читает серию байтов из файла.

**Preconditions**
Файловая ручка ссылавшаяся hMPFS уже открытая.

**Parameters**
**Parameters Description**
**hMPFS** ( СМОТРИ страницу 348) дескриптор файла из которого будут считываться
**cData** где хранить байты, которые были прочитаны
**wLen** сколько байт читать

//*-*-*-*-*-----------------------------------------

### 10.10.1.9 MPFSGetBytesRem Function

**File**
MPFS2.h

**C**
DWORD **MPFSGetBytesRem**(
MPFS_HANDLE **hMPFS**
);

**Returns**
Количество байтов, остающихся в файле как DWORD

**Description**

Определяет количество байтов, оставшихся для чтения.

**Preconditions**
Файловая ручка ссылавшаяся hMPFS уже открытая.

**Parameters**
**Parameters Description**
hMPFS ( see page 348) дескриптор файла из которого будут считываться метаданные

//*-*-*-*-*----------

### 10.10.1.10 MPFSGetEndAddr Function

**File**
MPFS2.h

**C**
MPFS_PTR **MPFSGetEndAddr**(
MPFS_HANDLE **hMPFS**
);

**Returns**
Адрес сразу после конца файла ( начальный адрес следующего файла )

**Description**
Определяет конечный адрес файла.

**Preconditions**
Файл ручка ссылается hMPFS уже открыт.

**Parameters**
**Parameters Description**
hMPFS ( see page 348) дескриптор файла из которого будут считываться метаданные

//*-*-*-*-*----------

### 10.10.1.11 MPFSGetFilename Function

**File**
MPFS2.h

**C**
BOOL **MPFSGetFilename**(
MPFS_HANDLE **hMPFS**,
BYTE* **cName**,
WORD **wLen**
);

**Description**
Читает имя файла в файл, который уже открыт.

**Preconditions**
Файл ручка ссылается hMPFS уже открыт.

**Parameters**
**Parameters Description**
**hMPFS** ( СМОТРИ страницу 348) дескриптор файла , из которого , чтобы определить имя файла
**cName** где хранить имя файла
**wLen** максимальная длина данных для хранения в **cName**

**Return Values**
**Return Values Description**
**TRUE** имя файла успешно было расположено
**FALSE** дескриптор файла при условии не открыт в настоящее время

//*-*-*-*-*----------

### 10.10.1.12 MPFSGetFlags Function

**File**
MPFS2.h

**C**
WORD **MPFSGetFlags**(
MPFS_HANDLE **hMPFS**
);

**Returns**
Флаги, которые были связаны файлом

**Description**
Читает флаги файла.

**Preconditions**
Файл ручка ссылается hMPFS уже открыт.

**Parameters**
**Parameters Description**
hMPFS ( СМОТРИ страницу 348) дескриптор файла из которого будут считываться метаданные

//*-*-*-*-*----------

### 10.10.1.13 MPFSGetID Function

**File**
MPFS2.h

**C**
WORD **MPFSGetID**(
MPFS_HANDLE **hMPFS**
);

**Returns**
ID в FAT для этого файла

**Description**
Определяет идентификатор в FAT для файла .

**Remarks**
Используйте эту функцию в сочетании с MPFSOpenID (см. стр. 281 ), чтобы быстро получить доступ к файлу без постоянного резервирования дескриптор файла .

**Preconditions**
Файл ручка ссылается hMPFS уже открыт.

**Parameters**
**Parameters Description**
hMPFS ( see page 348) дескриптор файла из которого будут считываться метаданные

//*-*-*-*-*----------

### 10.10.1.14 MPFSGetLong Function

**File**
MPFS2.h

**C**
BOOL **MPFSGetLong**(
MPFS_HANDLE **hMPFS**,
DWORD* **ul**
);

**Returns**
TRUE - байт успешно был прочитан
FALSE - No байт не был прочитан поскольку или ручка была недействительна или конец файла достигнут.

**Description**
Читает DWORD или Длинную величину из MPFS.

**Preconditions**
Файл ручка ссылается hMPFS уже открыт.

**Parameters**

**Parameters Description**
hMPFS- ( see page 348) дескриптор файла из которого будут считываться
ul - где, чтобы загружать DWORD или длинная величина, что был прочитан

//*-*-*-*-*----------

### 10.10.1.15 MPFSGetMicrotime Function

**File**
MPFS2.h

**C**
DWORD **MPFSGetMicrotime**(
MPFS_HANDLE **hMPFS**
);

**Returns**
microtime, Который был прочитан как DWORD

**Description**
Читает microtime часть временной метки файла.

**Preconditions**
Файл ручка ссылается hMPFS уже открыт.

**Parameters**
**Parameters Description**
hMPFS ( see page 348) дескриптор файла из которого будут считываться метаданные

//*-*-*-*-*----------

### 10.10.1.16 MPFSGetPosition Function

**File**
MPFS2.h

**C**
DWORD **MPFSGetPosition**(
MPFS_HANDLE **hMPFS**
);

**Returns**
Позиция в файле как DWORD (или MPFS_PTR ( смотри страницу 286))

**Description**
Определяет текущую позицию в файле

**Remarks**
Разговор MPFSSeek ( смотри страницу 283)(hMPFS, pos, MPFS_SEEK_START), возвращает указатель в эту позицию должность на более поздней Время. (Где pos - величина возвращанная этим function.)

**Preconditions**
Файл ручка ссылается hMPFS уже открыт.

**Parameters**
**Parameters Description**
hMPFS ( СМОТРИ страницу 348)
дескриптор файла , для которого необходимо определить положение

//*-*-*-*-*----------

### 10.10.1.17 MPFSGetSize Function

**File**
MPFS2.h

**C**
DWORD **MPFSGetSize**(
MPFS_HANDLE **hMPFS**
);

**Returns**
Размер, который был прочитан как DWORD

**Description**
Читает размер файла.

**Preconditions**
Файловая ручка ссылавшаяся hMPFS уже открытая.

**Parameters**
**Parameters Description**
hMPFS ( see page 348) дескриптор файла из которого будут считываться метаданные

//*-*-*-*-*----------

### 10.10.1.18 MPFSGetStartAddr Function

**File**
MPFS2.h

**C**
MPFS_PTR **MPFSGetStartAddr**(

MPFS_HANDLE **hMPFS**
);

**Returns**
Начальный адрес файла на изображении MPFS

**Description**
Читает стартовый адрес файла.

**Preconditions**
Файл ручка ссылается hMPFS уже открыт.

**Parameters**
**Parameters Description**
hMPFS ( see page 348) дескриптор файла из которого будут считываться метаданные

//*-*-*-*-*----------

### 10.10.1.19 MPFSGetTimestamp Function

**File**
MPFS2.h

**C**
DWORD **MPFSGetTimestamp**(
MPFS_HANDLE **hMPFS**
);

**Returns**
Отметка времени, которая была прочитана как DWORD

**Description**
Читает отметку времени для определенного файла.

**Preconditions**
Файловая ручка ссылавшаяся hMPFS уже открытая.

**Parameters**
**Parameters Description**
hMPFS ( see page 348) дескриптор файла из которого будут считываться метаданные

//*-*-*-*-*----------

### 10.10.1.20 MPFSOpen Function

**File**

MPFS2.h

**C**
MPFS_HANDLE **MPFSOpen**(
BYTE* **cFile**
);

**Returns**
MPFS_HANDLE ( Смотри страницу 273) в открытый файл если обнаружено, или MPFS_INVALID_HANDLE ( смотри страницу 273) если файл не мог обнаруживан или никакие свободные ручки не существуют.

**Description**
Открывает файл в файловой системе MPFS2.

**Preconditions**
None

**Parameters**
**Parameters Description**
**cFile** оканчивающихся нулем имени файла , чтобы открыть

//*-*-*-*-*----------

### 10.10.1.21 MPFSOpenID Function

**File**
MPFS2.h

**C**
MPFS_HANDLE **MPFSOpenID**(
WORD **hFatID**
);

**Returns**
MPFS_HANDLE ( Смотри страницу 273) в открытый файл если обнаружено, или MPFS_INVALID_HANDLE ( смотри страницу 273) если файл не мог обнаруживан или никакие свободные ручки не существуют.

**Description**
Быстро см.- открывать файл в файловой системе MPFS2. Используйте эту функцию вместе с MPFSGetID ( смотри страницу 278)() в быстро см.- открывать файл не tying по постоянном MPFSStub.

**Preconditions**
None

**Parameters**

**Parameters Description**
hFatID ID предшествующий открытый файл в FAT

//*-*-*-*-*-----------------------------------------

### 10.10.1.22 MPFSOpenROM Function

**File**
MPFS2.h

**C**
MPFS_HANDLE **MPFSOpenROM**(
ROM BYTE* **cFile**
);

**Returns**
MPFS_HANDLE ( Смотри страницу 273) в открытый файл если обнаружено, или MPFS_INVALID_HANDLE ( смотри страницу 273) если файл не мог обнаруживан или никакие свободные ручки не существуют.

**Description**
Открывает файл в файловой системе MPFS2.

**Remarks**
Эта функция - aliased на MPFSOpen ( смотри страницу 281) на платформах non-PIC18.

**Preconditions**
None

**Parameters**
**Parameters Description**
**cFile** оканчивающихся нулем имени файла , чтобы открыть

//*-*-*-*-*-----------------------------------------

### 10.10.1.23 MPFSPutArray Function

**File**
MPFS2.h

**C**
WORD **MPFSPutArray**(
MPFS_HANDLE **hMPFS**,
BYTE* **cData**,
WORD **wLen**
);

**Returns**
Количество успешно письменных байтов.

**Description**
Записывает массив данных в образ MPFS.

**Remarks**
Для EEPROM , реальная запись может не инициализировать , пока страница внутренний записи не будет заполнен. Чтобы убедиться, что ранее списанных данные получает хранится , **MPFSPutEnd** (см. стр. 283 ) должна быть вызвана после последнего обращения к **MPFSPutArray** .

**Preconditions**
**MPFSFormat** ( СМОТРИ страницу 274), sucessfully был вызван.

**Parameters**
**Parameters Description**
**hMPFS** ( see page 348) дескриптор файла для записи
**cData** массив байтов, чтобы записывать
**wLen** сколько байтов, чтобы записывать

//*-*-*-*-*----------

### 10.10.1.24 MPFSSeek Function

**File**
MPFS2.h

**C**
```
BOOL MPFSSeek(
MPFS_HANDLE hMPFS,
DWORD dwOffset,
MPFS_SEEK_MODE tMode
);
```

**Returns**
TRUE - поиск имел успех
FALSE - или новая позиция или сама ручка была недействительна

**Description**
Перемещает текущее указателя считывания на новое место .

**Preconditions**
Файл ручка ссылается hMPFS уже открыт.

**Parameters**
**Parameters Description**

**hMPFS** ( смотри страницу 348) файловая дескриптор файла искать с смещением
**dwOffset** смещение от заданной позиции в указанном направлении

**tMode** одно из MPFS_SEEK_MODE ( смотри страницу 273) constants

//*-*-*-*-*----------

### 10.10.1.25 MPFSPutEnd Function

**File**
MPFS2.h

**C**
```
void MPFSPutEnd(
BOOL final
);
```

**Returns**
None

**Description**
Завершает письмо MPFS операцию записи .

**Preconditions**
MPFSFormat ( СМОТРИ страницу 274) и MPFSPutArray ( смотри страницу 282), были успешно называется .

**Parameters**
**Parameters Description**
финал TRUE если приложение сделано запись, FALSE если MPFS2 был вызван эта функция локально.

//*-*-*-*-*----------

### 10.10.2 MPFS2 Stack Members

**Functions**
**Name Description**
MPFSInit ( see page 284) Initializes the MPFS module.

**Module**
MPFS2 ( see page 271)

**Description**

Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные .

//*-*-*-*-*----------

#### 10.10.2.1 MPFSInit Function

**File**
MPFS2.h

**C**
```
void MPFSInit();
```

**Returns**
None

**Description**
Устанавливает все ручки MPFS на закрытое и инициализирует доступ к EEPROM если необходимо.

**Remarks**
Эта функция вызывается только один при жизни приложения.

**Preconditions**
None

**Section**
Функциональные Определения

//*-*-*-*-*----------

### 10.10.3 MPFS2 Internal Members

**Functions**
**Name Description**
ReadProgramMemory ( see page 288) Assembly function to read all three bytes of program memory for 16-bit parts
_LoadFATRecord ( see page 288) Loads the FAT record for a specified handle.
_Validate ( see page 289) Validates the MPFS Image

ReadProgramMemory ( смотри страницу 288)Ассамблея функционировать , чтобы прочитать все три байта памяти программ для 16-битных частей

_LoadFATRecord ( смотри страницу 288)Загружает запись FAT для указанного ручкой.

_Validate ( смотри страницу 289), Проверяет MPFS изображение

**Macros**
**Name Description**
MAX_FILE_NAME_LEN (
see page 286)
Supports long file names to 64 characters
MPFS_WRITE_PAGE_SIZE
( see page 287)
Defines the size of a page in EEPROM
MPFS2_FLAG_HASINDEX
( see page 287)
Indicates a file has an associated index of dynamic variables
MPFS2_FLAG_ISZIPPED (
see page 287)
Indicates a file is compressed with GZIP compression
MPFSTell ( see page 288) Alias of MPFSGetPosition ( see page 279)
MPFS_INVALID_FAT ( see
page 290)
Indicates an invalid FAT cache

MAX_FILE_NAME_LEN ( смотри страницу 286)Поддерживает длинные имена файлов до 64 символов

MPFS_WRITE_PAGE_SIZE ( смотри страницу 287)
Определяет размер страницы в EEPROM

MPFS2_FLAG_HASINDEX ( смотри страницу 287)
Указывает файл имеет соответствующий индекс динамических переменных

MPFS2_FLAG_ISZIPPED ( смотри страницу 287)
Указываетфайл сжат со сжатием GZIP

MPFSTell Псевдонима ( смотри страницу 288) Псевдоним MPFSGetPosition ( смотри страницу 279)

MPFS_INVALID_FAT ( смотри страницу 290)
Указывает на недопустимый кэш FAT

**Module**
MPFS2 ( see page 271)

**Structures**
**Name Description**
MPFS_STUB ( see page

286)
Stores each file handle's information Handles are free when addr =
MPFS_INVALID ( see page 273)
MPFS_FAT_RECORD (
see page 289)
Stores the data for an MPFS2 FAT record

MPFS_STUB ( смотри страницу286)
Магазины информация Ручки каждого дескриптор файла бесплатны , когда Адрес =

MPFS_INVALID ( смотри страницу 273) MPFS_FAT_RECORD ( смотри страницу 289)
Загружает данные для записи MPFS2 FAT

**Types**
**Name Description**
MPFS_PTR ( смотри страницу 286) Указатели в настоящее время DWORDS

**Variables**
**Name Description**
isMPFSLocked ( see page
285)
Allows the MPFS to be locked, preventing access during updates
lastRead ( see page 286) Track the last read address to prevent unnecessary data overhead to switch locations.
MPFSStubs ( see page 287) Track the MPFS File Handles MPFSStubs[0] is reserved for internal use (FAT
access)
fatCache ( see page 289) FAT record cache
fatCacheID ( see page 290) ID of currently loaded fatCache ( see page 289)
numFiles ( see page 290) Number of files in this MPFS image

isMPFSLocked Описание Имени ( смотри страницу285)
Позволяет MPFS, быть заблокирован , предотвращая доступ во время обновления

lastRead ( смотри страницу 286), следить за последний адрес для чтения , чтобы предотвратить излишнюю нагрузку данных для переключения местах.

MPFSStubs ( СМОТРИ страницу 287), следить за MPFS Файл Ручки MPFSStubs [0] зарезервирован для внутреннего использования (FAT доступ)

fatCache (см. стр. 289 ) FAT запись кэша

fatCacheID Количества ( смотри страницу 290) загруженный идентификатор

fatCache ( смотри страницу 289)

numFiles ( смотри страницу 290) Количество файлов в этом MPFS изображения

**Description**
Следующие функции и переменные определяются как внутренние, к модулю MPFS2 .

//*-*-*-*-*----------

#### 10.10.3.1 isMPFSLocked Variable

**File**
MPFS2.c

**C**
BOOL **isMPFSLocked**;

**Description**
Позволяет MPFS, быть заблокирован , предотвращая доступ во время обновления

//*-*-*-*-*----------

#### 10.10.3.2 lastRead Variable

**File**
MPFS2.c

**C**
MPFS_PTR **lastRead**;

**Description**
Проследите последний адрес чтения, чтобы предотвратить излишнюю нагрузку данных , чтобы перейти места .

//*-*-*-*-*----------

#### 10.10.3.3 MAX_FILE_NAME_LEN Macro

**File**
MPFS2.c

**C**
**#define MAX_FILE_NAME_LEN** (64u)

**Description**
Поддерживает длинные файловые имена в 64 символов

//*-*-*-*-*----------

### 10.10.3.4 MPFS_PTR Type

**File**
MPFS2.h

**C**
```
typedef DWORD MPFS_PTR;
```

**Description**
MPFS - Указатели в настоящее время DWORDS

//*-*-*-*-*----------

### 10.10.3.5 MPFS_STUB Structure

**File**
MPFS2.h

**C**
```
typedef struct {
MPFS_PTR addr;
DWORD bytesRem;
WORD fatID;
} MPFS_STUB;
```

**Members**
**Members Description**
MPFS_PTR addr; Текущий адрес в файловой системе
DWORD bytesRem; Сколько байтов остаются в этом файле
WORD fatID; Идентификатор , какой файл в FAT был доступ

**Description**
Загружает каждые информационные дескриптор файла бесплатны , когда Адрес = MPFS_INVALID (см. стр. 273 )

//*-*-*-*-*----------

### 10.10.3.6 MPFS_WRITE_PAGE_SIZE Macro

**File**
MPFS2.h

**C**
**#define MPFS_WRITE_PAGE_SIZE** (64u) *// Определяет размер страницы в EEPROM*

**Description**
Определяет размер страницы в EEPROM

//*-*-*-*-*----------

### 10.10.3.7 MPFS2_FLAG_HASINDEX Macro

**File**
MPFS2.h

**C**
**#define MPFS2_FLAG_HASINDEX** ((WORD)0x0002*) // Указывает что файл имеет связанный индекс динамических переменных*

**Description**
Указывает что файл имеет связанный индекс динамических переменных

//*-*-*-*-*----------

### 10.10.3.8 MPFS2_FLAG_ISZIPPED Macro

**File**
MPFS2.h

**C**
**#define MPFS2_FLAG_ISZIPPED** ((WORD)0x0001*// Указывает что файл сжат сжатием GZIP*

**Description**
Указывает что файл сжат сжатием GZIP

//*-*-*-*-*----------

### 10.10.3.9 MPFSStubs Variable

**File**
MPFS2.c

**C**
MPFS_STUB **MPFSStubs**[MAX_MPFS_HANDLES+1];

**Description**
Отслеживание MPFS Файл Ручки MPFSStubs [0] зарезервирован для внутреннего использования (доступ FAT)

//*-*-*-*-*----------

### 10.10.3.10 MPFSTell Macro

**File**
MPFS2.h

**C**
**#define MPFSTell**(a) MPFSGetPosition(a)

**Description**
Псевдоним MPFSGetPosition ( смотри страницу 279)

//*-*-*-*-*----------

### 10.10.3.11 ReadProgramMemory Function

**File**
MPFS2.h

**C**
DWORD **ReadProgramMemory**(
DWORD **address**
);

**Description**
Ассамблея функционировать , чтобы прочитать все три байта памяти программ для 16-битных частей

//*-*-*-*-*----------

### 10.10.3.12 _LoadFATRecord Function

**File**
MPFS2.c

**C**

**static void _LoadFATRecord(**
WORD **fatID**
);

**Returns**
None

**Description**
Загружает запись FAT для определенной ручки.

**Remarks**
Запись FAT будет храниться в fatCache ( смотри страницу 289).

**Preconditions**
None

**Parameters**
**Parameters Description**
fatID идентификатор файла, который FAT должен быть загружен

//*-*-*-*-*----------

### 10.10.3.13 _Validate Function

**File**
MPFS2.c

**C**
**static void _Validate**();

**Returns**
None

**Description**
Проверяет , что изображение MPFS действует , и читает количество доступных файлов из заголовка изображения. Эта функция называется при загрузке, и снова после любое изображение записывается.

**Preconditions**
None

//*-*-*-*-*----------

### 10.10.3.14 MPFS_FAT_RECORD Structure

**File**

MPFS2.h

**C**
```
typedef struct {
DWORD string;
DWORD data;
DWORD len;
DWORD timestamp;
DWORD microtime;
WORD flags;
} MPFS_FAT_RECORD;
```

**Members**
**Members Description**
DWORD строка; Указатель на имя файла
DWORD; Адрес ( смотри страницу 147) данных файла
DWORD; Длина данных файла
DWORD; Отметка времени файла
DWORD microtime; Марка Microtime файлового
WORD  сигнализирует; Флаги для этого файла

**Description**
Загружает данные для записи FAT MPFS2

//*-*-*-*-*----------

### 10.10.3.15 fatCache Variable

**File**
MPFS2.c

**C**
MPFS_FAT_RECORD **fatCache**;

**Description**
FAT рекордный кеш

//*-*-*-*-*----------

### 10.10.3.16 fatCacheID Variable

**File**
MPFS2.c

**C**

WORD **fatCacheID**;

**Description**
Идентификатор загруженной fatCache ( смотри страницу 289)

//*-*-*-*-*----------

### 10.10.3.17 numFiles Variable

**File**
MPFS2.c

**C**
WORD **numFiles**;

**Description**
Количество файлов в этом образе MPFS

//*-*-*-*-*----------

### 10.10.3.18 MPFS_INVALID_FAT Macro

**File**
MPFS2.h

**C**
```
#define MPFS_INVALID_FAT (0xffffu) // Указывает неправильный кеш FAT cache
```

**Description**
Указывает на недопустимый кэш FAT

//*-*-*-*-*----------

## 10.11 NBNS

Протокол NetBIOS Name Service связывает имена хостов с IP- адресов , так же, как DNS , но на той же подсети .

Практически , это позволяет присвоение человек- имя хостов для доступа доски в той же подсети . Например . в " TCPIP Demo App" демонстрационный проект , Демонстрационная плата программируется с человеческим именем ' mchpboard " , чтобы он мог получить доступ непосредственно , а не с его IP -адрес.

//*-*-*-*-*----------

### 10.11.1 NBNS Stack Members

**Functions**
**Name Description**
NBNSGetName ( see page
291)
Reads the NetBIOS name from a UDP socket and copies it into a
user-specified buffer.
NBNSPutName ( see page
292)
Transmits the NetBIOS name across an open UDP socket.
NBNSTask ( see page 292) Sends responses to NetBIOS name requests

Имя Description NBNSGetName ( смотри страницу
291)
Читает имя NetBIOS из розетки UDP и копирует это в определенный буфер пользователя.
NBNSPutName ( СМОТРИ страницу
292)
Передает имя NetBIOS через открытую розетку UDP.
NBNSTask ( СМОТРИ страницу 292), Посылает ответы на запросы имени NetBIOS

**Macros**
**Name Description**
NBNS_PORT ( see page
293)
смотри порт Услуги страничного Имени NetBIOS

**Module**
NBNS ( see page 290)

**Structures**
**Name Description**
NBNS_HEADER ( seepage 293)
структура Заголовка NBNS

_NBNS_HEADER ( seepage 293)
структура Заголовка NBNS

**Description**
Следующее функционирует и переменные являются публикой но предполагаются, чтобы быть доступно стеком себя. Приложения должны обычно не вызывать эти функции или модифицировать эти переменные.

//*-*-*-*-*----------

#### 10.11.1.1 NBNSGetName Function

**File**
NBNS.c

**C**
```
static void NBNSGetName(
BYTE * String
);
```

**Side Effects**
None

**Returns**
None

**Description**
Читает имя NetBIOS из socket UDP и копирует это в определенный буфер пользователя.

**Remarks**
None

**Preconditions**
None

**Parameters**
**Parameters Description**
Указатель Строки в массив в который полученное имя NetBIOS должно быть скопировано.

//*-*-*-*-*----------

#### 10.11.1.2 NBNSPutName Function

**File**
NBNS.c

**C**
```
static void NBNSPutName(
BYTE * String
);
```

**Side Effects**
None

**Returns**
None
**Description**
Передает имя NetBIOS через открытую UDP socket.

**Remarks**
None

**Preconditions**
None

**Parameters**
**Parameters Description**
Имя, чтобы передаваться

//*-*-*-*-*-----------------------------------------

#### 10.11.1.3 NBNSTask Function

**File**
NBNS.h

**C**
```
void NBNSTask();
```
**Side Effects**
None

**Returns**
None

**Description**
Посылает ответы на запросы имени NetBIOS

**Remarks**
None

**Preconditions**
None

//*-*-*-*-*-----------------------------------------

#### 10.11.1.4 NBNS_HEADER Structure

**File**
NBNS.c

**C**
```
typedef struct _NBNS_HEADER {
WORD_VAL TransactionID;
WORD_VAL Flags;
WORD_VAL Questions;
```

WORD_VAL **Answers**;
WORD_VAL **AuthoritativeRecords**;
WORD_VAL **AdditionalRecords**;
} NBNS_HEADER;

**Description**
СТРУКТУРА Заголовка NBNS

//*-*-*-*-*----------

#### 10.11.1.5 NBNS_PORT Macro

**File**
NBNS.c

**C**
**#define NBNS_PORT** (137u)

**Description**
ПОРТ Услуги Имени NetBIOS

//*-*-*-*-*----------

## 10.12 Тесты производительности

Модули тестирования TCP и UDP Производительность создание способа получения метрик производительности в стека . Они может использоваться для отсчета стек на различных аппаратных платформах , и также могут быть использованы для определения , как ваш обычай Приложение повлияло производительность стека.

Стек автоматически вызывает эти модули , если они включены , так что вы никогда не должны вызывать любые функции напрямую.

Инструкция по использованию модулей можно найти в документации к UDPPerformanceTask (см. стр. 294 ) ,

TCPTXPerformanceTask (см. стр. 296 ) , и TCPRXPerformanceTask (см. стр. 295) .

//*-*-*-*-*----------

#### 10.12.1 Performance Test Stack Members

### 10.12.1 характеристик Испытания Стек Пользователи

**Функции**

**Имя Описание**

**TCPPerformanceTask** (см.страница 294 ) Тесты на производительность модуля TCP.

**UDPPerformanceTask** (см. стр. 294 ) Тесты на производительность передачи модуля UDP .

**модуль**

Тесты производительности (см. стр. 293 )

**описание**

Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные .

//*-*-*-*-*----------

#### 10.12.1.1 TCPPerformanceTask Function

**File**
TCPPerformanceTest.h

**C**
```
void TCPPerformanceTask();
```

**Returns**
None

**Description**
Эта функция вызывает как **TCPTXPerformanceTask** (см. стр. 296 ) и **TCPRXPerformanceTask** (см. стр. 295 ), чтобы выполнять функции задач производительности. Обратитесь к документации для каждого из этих функций для получения подробной информации .

**Preconditions**
TCP is initialized.

//*-*-*-*-*----------

#### 10.12.1.2 UDPPerformanceTask Function

**File**
UDPPerformanceTest.h

**C**
**void UDPPerformanceTask**();

**Returns**
None

**Description**
Эта функция тестирует производительность передачи модуля UDP . При загрузке , этот модуль будет передавать 1024 большой UDP вещание Пакеты по 1024 байт каждый . Использование анализаторов трафика , можно определить, как долго этот процесс занимает и рассчитать скорости передачи стека. Эта функция проверяет истинную производительность UDP в том, что он будет открыть сокет , передачи одного пакета , и закрыть сокет для каждого цикла. После этой начальной передачи , модуль может быть повторно включен на кнопку 3 проведение .

Эта функция особенно полезна после разработки , чтобы определить воздействие коде приложения на стека производительность . До и после сравнения укажет если ваше приложение недопустимо блокируя процессор или принимать слишком долго, чтобы выполнить .

**Preconditions**
UDP is initialized.

//*-*-*-*-*-----------------------------------------

### 10.12.2 Performance Test Internal Members
### 10.12.2 тест производительности внутренних Пользователи

**Functions**
**Name Description**
**TCPRXPerformanceTask** ( смотри страницу 295) Тесты на получение характеристик модуля TCP.

**TCPTXPerformanceTask** ( СМОТРИ страницу 296) Тесты на производительность передачи модуля TCP.

**Macros**
**Name Description**

**PERFORMANCE_PORT** ( смотри страницу 296) Какой порт UDP, чтобы транслировать из для тестов

**UDP RX_PERFORMANCE_PORT** ( смотри страницу 296) TCP порт для прослушивания (см. стр. 175 ) в течение TCP получать тесты

**TX_PERFORMANCE_PORT** ( смотри страницу 297)
Порт TCP, чтобы слушать ( смотри страницу 175) на для тестов передачи TCP

**Module**
Исполнение Тестирует ( смотри страницу 293)

**Description**
Следующие функции и переменные определяются как внутренние, к модулю.

//*-*-*-*-*----------

### 10.12.2.1 TCPRXPerformanceTask Function

**File**
TCPPerformanceTest.c

**C**
```
void TCPRXPerformanceTask();
```

**Returns**
None

**Description**
Эта функция проверяет получение характеристик модуля TCP. Для использования , открыть соединение телнет к устройству на RX_PERFORMANCE_PORT (см. стр. 296 ) ( 9763 по умолчанию). Затем с помощью телнет утилиту для загрузки больших файлов , чтобы Устройство . Каждый второй совет будет отчитываться , сколько байт было получено в предыдущем секунду.

Производительность TCP зависит от многих факторов , в том числе времени приема-передачи и размера TCP буфера. Чтобы ускорить получение результата , увеличить размер размера буфера для RX гнездо TCP_PURPOSE_TCP_PERFORMANCE_RX в TCPIPConfig.h . Время приема-передачи зависит от расстояния до устройства , так через стол будет на несколько порядков быстрее , чем через Интернет.

Эта функция особенно полезна после разработки , чтобы определить воздействие коде приложения на стека производительность . До и после сравнения укажет если ваше приложение недопустимо блокируя процессор или принимать слишком долго, чтобы выполнить .

**Preconditions**
TCP is initialized.

//*-*-*-*-*-----------------------------------------

### 10.12.2.2 TCPTXPerformanceTask Function

**File**
TCPPerformanceTest.c

**C**
**void TCPTXPerformanceTask**();

**Returns**
None

**Description**
Эта функция тестирует производительность передачи модуля TCP. Для использования , открыть соединение телнет к устройству на TX_PERFORMANCE_PORT (см. стр. 297) ( 9762 по умолчанию). Совет будет быстро передавать данные и сообщить его производительность клиенту Telnet.

Производительность TCP зависит от многих факторов , в том числе времени приема-передачи и размера TCP буфера. Чтобы ускорить получение результата , увеличить размер размера TX буфера для сокета TCP_PURPOSE_TCP_PERFORMANCE_TX в TCPIPConfig.h . Время приема-передачи зависит от расстояния до устройства , так через стол будет на несколько порядков быстрее , чем через Интернет.

Эта функция особенно полезна после разработки , чтобы определить воздействие коде приложения на стека производительность . До и после сравнения укажет если ваше приложение недопустимо блокируя процессор или принимать слишком долго, чтобы выполнить .

**Preconditions**
TCP is initialized.

//*-*-*-*-*-----------------------------------------

### 10.12.2.3 PERFORMANCE_PORT Macro

**File**
UDPPerformanceTest.c

**C**
**#define PERFORMANCE_PORT** 9

**Description**
Какой порт UDP, чтобы транслировать из для тестов UDP

//*-*-*-*-*----------

### 10.12.2.4 RX_PERFORMANCE_PORT Macro

**File**
TCPPerformanceTest.c

**C**
```
#define RX_PERFORMANCE_PORT 9763
```

**Description**
Порт TCP, чтобы слушать ( смотри страницу 175) на для тестов приемника TCP

//*-*-*-*-*----------

### 10.12.2.5 TX_PERFORMANCE_PORT Macro

**File**
TCPPerformanceTest.c

**C**
```
#define TX_PERFORMANCE_PORT 9762
```

**Description**
Порт TCP, чтобы слушать ( смотри страницу 175) на для тестов передачи TCP

//*-*-*-*-*----------

## 10.13 SMTP Client

Клиентский модуль SMTP в TCP / IP стека позволяет приложениям отправлять электронную почту любому получателю по всему миру. Эти сообщения может включать в себя информацию о состоянии или важные предупреждения. Использование электронной почты для SMS -шлюзов , предоставляемых большинство перевозчиков сотовый телефон , эти сообщения также могут быть доставлены непосредственно на мобильный телефонах .

Использование клиента SMTP требуется доступ к локальному почтовому серверу (например, mail.yourdomain.com ) для надежной работы . Ваш провайдер или сетевой администратор может обеспечить правильно адрес , но приложения для конечных пользователей будет нужно интерфейс , чтобы обеспечить эти данные .

//*-*-*-*-*-----------------------------------------

### 10.13.1 SMTP Client Examples

**Module**
SMTP Client ( see page 297)

**Description**
Следующие два примера демонстрируют использование клиента SMTP в других сценариях. Первый, и простой пример, посылает короткое сообщение, чье содержание полностью расположено в PAM сразу.

Второй пример является более сложным и демонстрирует формирования сообщения на лету в случае, когда вся сообщение не помещается в оперативную память сразу. В этом случае , сообщение , созданные в стеке, но доставка содержимого происходит на куски и обрабатывается приложением.

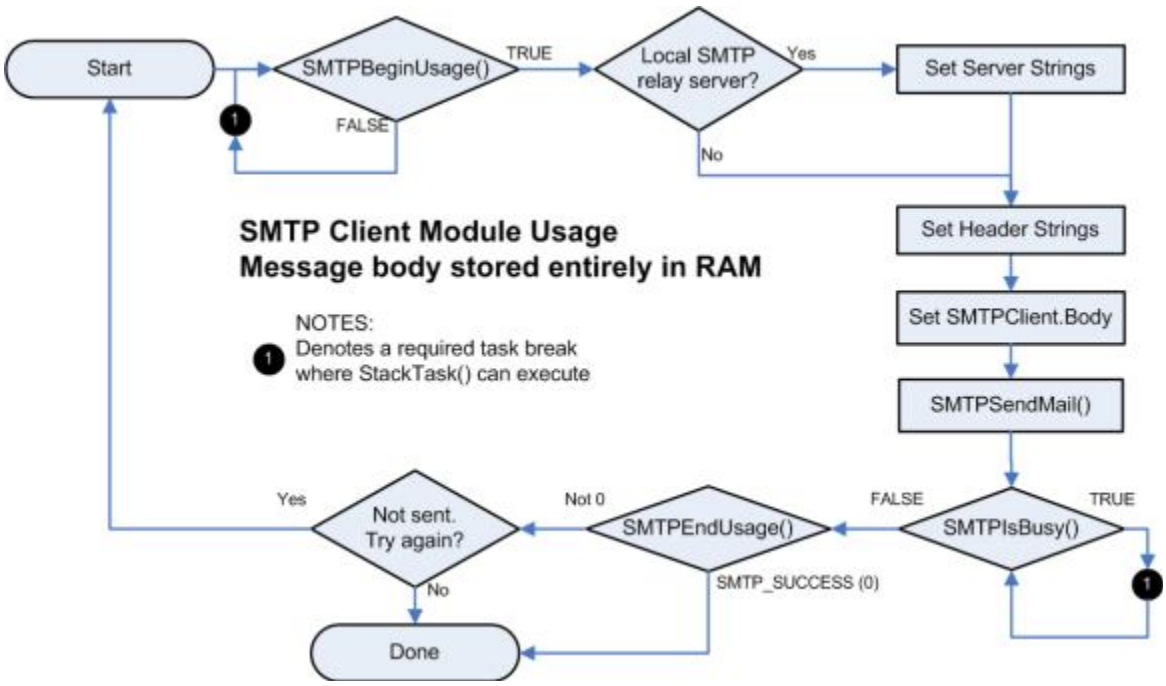
//*-*-*-*-*-----------------------------------------

#### 10.13.1.1 SMTP Client Пример коротких сообщений

Клиент SMTP API упрощается , когда сообщения могут быть помещены в буфер полностью в оперативной памяти. SMTPDemo ( см. стр. 97 ) Пример, приведенный в MainDemo.c посылает краткое сообщение электронной почты , показывающее текущее состояние кнопок в разделе . Это документ будет идти через этот пример .

Убедитесь STACK_USE_SMTP_CLIENT является раскомментировали в TCPIPConfig.h , прежде чем продолжить .

**На рисунке ниже представлен обзор процесса :**




Во-первых, позвонить SMTPBeginUsage (см. стр. 303) , чтобы убедиться, что клиент SMTP доступен и начать новое сообщение . Если ЛОЖЬ возвращается , клиент SMTP занят и приложение должно вернуться в основной цикл , чтобы StackTask выполнить снова.

Далее, установите локальный сервер реле для использования в качестве SMTPClient.Server . Если локальный сервер реле требуется имя пользователя и пароль , установите SMTPClient.Username и SMTPClient.Password в соответствующие полномочия.

Если параметры сервера не установлен, то стек будет пытаться доставить сообщение непосредственно в его хосту назначения . Это , скорее всего, не из-за спам-профилактических мер введены в действие большинства провайдеров и сетевых администраторов .

Продолжить , чтобы установить строк , сколько необходимо для сообщения заголовков. Это включает в себя строку темы , от адреса , и любая получатели , которые нужно добавить . Наконец, установите SMTPClient.Body к отправляемого сообщения .

На данный момент, убедитесь, что SMTPClient.ROMPointers настроен правильно для любых строк , которые хранятся в программе памяти . После того, как сообщение будет готово послать, позвонить SMTPSendMail (см. стр. 309 ) поручить клиент SMTP , чтобы начать передачи .

Теперьприложение должно вызвать SMTPIsBusy (см. стр. 305 ) , пока она не возвращает значение ЛОЖЬ. Каждый раз, когда истина возвращается , возвращение к основной цикл и ждать StackTask выполнить снова. Это позволяет почтовый сервер , чтобы продолжать свою работу в кооператив многозадачность образом. После того, как ЛОЖЬ , возвращается , звоните SMTPEndUsage (см. стр. 304 ), чтобы освободить клиента SMTP . Проверьте вернуться значение этой функции , чтобы определить, если сообщение было успешно отправлено.

Пример в MainDemo.c нуждается незначительные изменения использовать ваш адрес электронной почты . Сервер и полей должны быть установлены в SMTPDemo (см. стр. 97) для того, чтобы сообщения , которые будут должным образом доставлено. Как только это будет сделано, удерживая Button2 и Button3 одновременно ( самый левый две кнопки ) начнется отправки сообщения. LED1 загорается как сообщение обрабатывается , и погаснет , когда государственная машина SMTP завершает . Если передача была успешным LED2 загорается , в противном случае она останется темным.

### 10.13.1.2 SMTP Client Пример длинного сообщения

Клиент SMTP API может отправлять сообщения , которые не вписываются полностью в оперативной памяти. Для этого приложение должно управлять его состояние выхода и только написать столько байт, сколько доступные в буфере одновременно. Второй SMTPDemo (см. стр.97) Пример, приведенный в MainDemo.c посылает сообщение , которое является сброс всех содержимого оперативной памяти ПОС . Этот пример В настоящее время в комментарий . Комментарий из предыдущего короткий пример сообщение и раскомментируйте Long Пример сообщение .

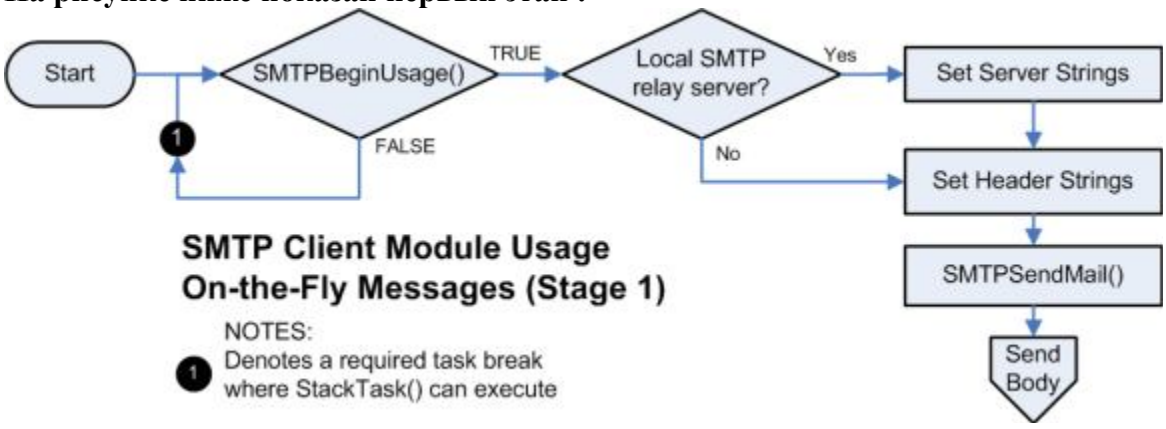
Этот документ будет идти через отправку более длинное сообщение .

Убедитесь STACK_USE_SMTP_CLIENT является раскомментировали в TCPIPConfig.h , прежде чем продолжить .

Отправка длинных сообщений разделен на три этапа. Первый этап настраивает SMTP клиент для отправки сообщения .

Второй этап отправляет сообщение в мелкие куски , как буферное пространство доступно . Заключительный этап завершает передачу и определяет, является ли или не был успешным в сообщении.

**На рисунке ниже показан первый этап :**




//*-*-*-*-*----------

Первый этап во многом похож на первые несколько шагов в отправив короткое сообщение . Во-первых, позвонить SMTPBeginUsage (см. стр.303) , чтобы убедиться, что клиент SMTP доступен и начать новое сообщение . Если ЛОЖЬ , возвращается , клиент SMTP занят и приложение должно вернуться в основной цикл , чтобы StackTask выполнить снова.

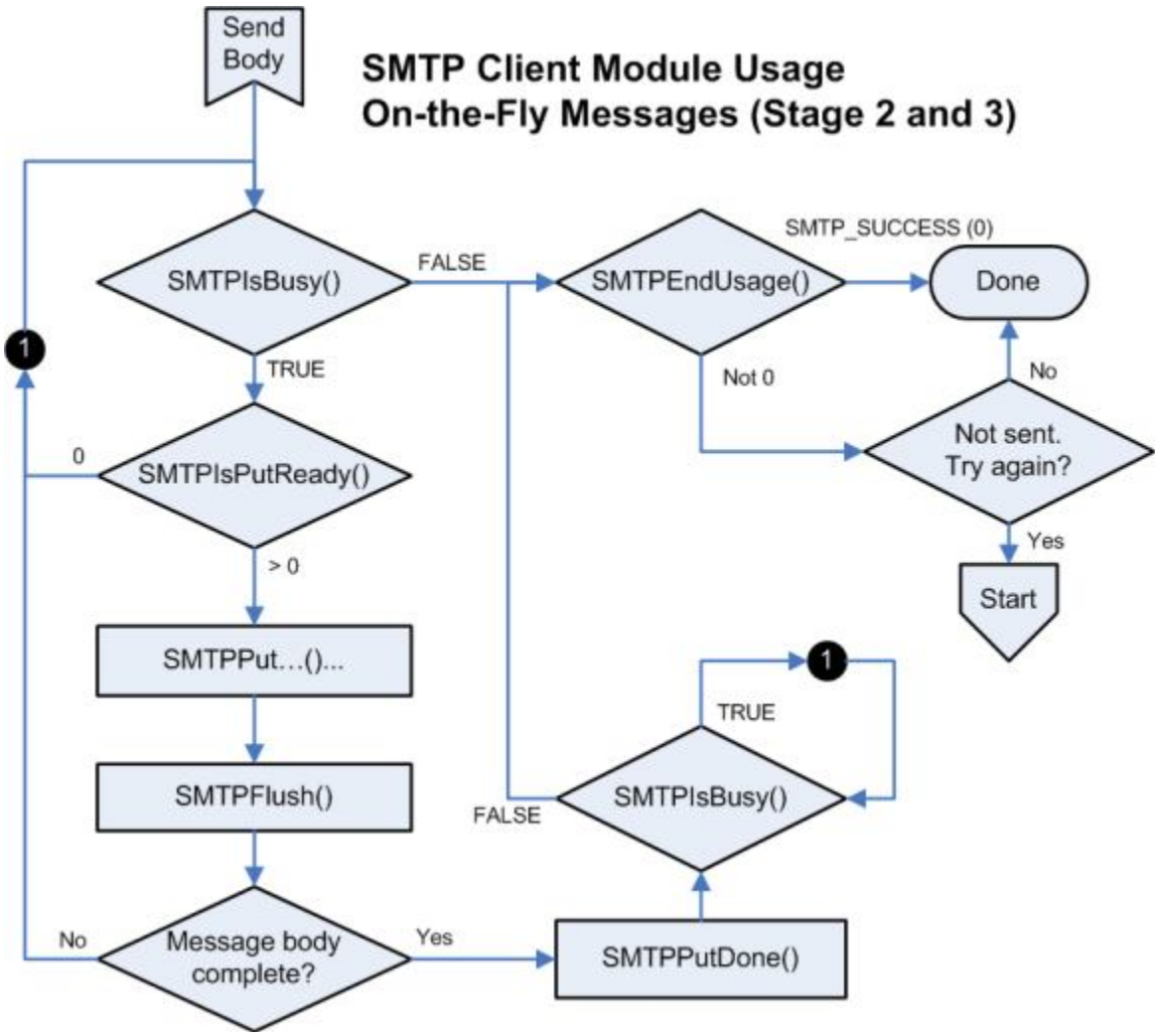
Далее, установите локальный сервер реле для использования в качестве SMTPClient.Server . Если локальный сервер реле требуется имя пользователя и пароль , установите SMTPClient.Username и SMTPClient.Password в соответствующие полномочия.

Если параметры сервера не установлен, то стек будет пытаться доставить сообщение непосредственно в его хосту назначения . Это , скорее всего, не из-за спам-профилактических мер введены в действие большинства провайдеров и сетевых администраторов .

Продолжить , чтобы установить строк , сколько необходимо для сообщения заголовков. Это включает в себя строку темы , от адреса , и любая получатели , которые нужно добавить .

Следующая часть процесса отличается . Убедитесь, что SMTPClient.Body остается готово, чтобы по умолчанию ( NULL ) . В этот момент называют SMTPSendMail (см. стр. 309) , чтобы открыть соединение с удаленным сервером и передавать заголовки . Приложение готов приступить ко второму этапу и отправить тело сообщения .

**На следующем рисунке представлен обзор стадии два и три :**




При входе во вторую стадию , приложение должно вызвать SMTPIsBusy (см. стр. 305) , чтобы убедиться, что подключение к удаленный сервер активен и не было потеряно . Если вызов выполняется успешно , звоните SMTPIsPutReady (см. стр. 305 ) , чтобы определить, сколько байт доступны в буфере TX. Если ни одного байта не доступны, вернуться в основной цикл , так что StackTask может передавать данные на удаленный узел и освободить буфер.

При наличии свободного места , любая комбинация из SMTPPut (см. стр. 306 ) , SMTPPutArray (см. стр. 306 ) ,

SMTPPutROMArray (см. стр. 307) , SMTPPutString (см. стр. 308) , и SMTPPutROMString (см. стр. 308 )

функции могут быть вызваны для передачи сообщения . Эти функции возвращают количество байт, успешно письменное . Используйте это значение , наряду со значением изначально вернулся из SMTPIsPutReady (см. стр. 305 ) , чтобы отслеживать , сколько свободного места остается в буфере TX. Как только буфер будет исчерпан , звоните SMTPFlush (см. стр. 304 ) , чтобы заставить данные, записанные для отправки.

Клиентский модуль SMTP может принять (см. стр. 169 ) столько данных, сколькоTCP Техас FIFO может иметь места. Это определяется сокет инициализации для сокетов типа TCP_PURPOSE_DEFAULT в TCPIPConfig.h , который по умолчанию 200 байт .

Если буфер TX будет исчерпан , прежде чем сообщение будет завершена, вернуться в основной цикл , так что StackTask может передавать Данные удаленного узла и освободить буфер . По возвращении , перейти к началу второго этапа для передачи на следующий часть сообщения .

Как только сообщение будет завершена, приложение будет перейти на третий этап. Позвоните SMTPPutDone (см. стр. 307) , чтобы сообщить клиент SMTP , что данных больше не остается. Тогда звоните SMTPIsBusy (см. стр. 305) неоднократно . Каждый раз, когда истина является вернулся , вернуться в основной цикл и ждать StackTask выполнить снова. После того, как ЛОЖЬ , возвращается , сообщение передачи завершил и приложение должно вызвать SMTPEndUsage (см. стр. 304 ), чтобы освободить клиента SMTP .

Проверьте возвращаемого значения этой функции , чтобы определить, является ли сообщение было успешно отправлено .

Пример в MainDemo.c нуждается незначительные изменения использовать ваш адрес электронной почты . Установите сервер и полей в SMTPDemo (см. стр. 97) , и убедитесь, что эти поля в настоящее время присвоена соответствующая SmtpClient (см. стр. 304 ) структура . Демо работает точно так же, как и предыдущий, с button2 и Button3 удерживается одновременно ( самый левый - две кнопки ) начав государственную машину . LED1 загорается каксообщение обрабатывается , и будет потушить , когда государственная машина SMTP завершает . Если передача была успешной СИД2 загорится, в противном случае он будет остаются темными .

//*-*-*-*-*----------

### 10.13.2 SMTP Client открытые члены

**Функции**

**Имя Описание**

**SMTPBeginUsage** (см.страница 303) Запросы контроль клиентского модуля SMTP .

**SMTPEndUsage** (см.страница 304) Релизы контроль клиентского модуля SMTP .

**SMTPFlush** (см. стр. 304 ) Очищает SMTP сокет и заставляет все данные, отправляемые .

**SMTPIsBusy** (см. стр.305) Определяет, является ли клиент SMTP занят.

**SMTPIsPutReady** (см.страница 305) Определяет, какая часть данных можно записать клиенту SMTP .

**SMTPPut** (см. стр. 306 ) Записывает один байт клиенту SMTP .

**SMTPPutArray** (см. стр.306) Записывает набор байт до клиента SMTP .

**SMTPPutDone** (см. стр.307) Указывает , что сообщение на лету завершена.

**SMTPPutROMArray** (см.страница 307) Записывает набор байтов из ПЗУ клиенту SMTP .

**SMTPPutROMString** (см.страница 308 ) Записывает строку из ПЗУ клиенту SMTP .

**SMTPPutString** (см. стр.308) Записывает строку клиенту SMTP .

**SMTPSendMail** (см. стр.309) Инициализацию процесса отправки сообщений.

**макрос**

**Имя Описание**

**SMTP_CONNECT_ERROR** (См. Стр. 301 ) Подключение к SMTP -сервер не смог

**SMTP_RESOLVE_ERROR** (См. Стр. 303 ) Поиск DNS для SMTP -сервера не удалось

**SMTP_SUCCESS** (см.страница 303) Сообщение было успешно отправлено

**модуль**

**SMTP Client** (см. стр. 297 )

**структуры**

**Имя Описание**

**SMTP_POINTERS** (см.страница 301) Настраивает клиент SMTP для отправки сообщения

**Переменные**

**Имя Описание**

**SmtpClient** (см. стр. 304 )

**описание**

Следующие функции и переменные доступны для применения стека.

//*-*-*-*-*----------

### 10.13.2.1 SMTP_CONNECT_ERROR Macro

**File**
SMTP.h

**C**
```
#define SMTP_CONNECT_ERROR (0x8001u) // Связь в сервер SMTP потерпевший неудачу
```
**Description**
Связь в сервер SMTP потерпела неудачу

//*-*-*-*-*-----------------------------------------

### 10.13.2.2 SMTP_POINTERS Structure

**File**
SMTP.h

**C**
```
typedef struct {
union {
BYTE * szRAM;
ROM BYTE * szROM;
} Server;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} Username;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} Password;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} To;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} CC;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} BCC;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} From;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} Subject;
```

```
union {
BYTE * szRAM;
ROM BYTE * szROM;
} OtherHeaders;
union {
BYTE * szRAM;
ROM BYTE * szROM;
} Body;
struct {
unsigned char Server : 1;
unsigned char Username : 1;
unsigned char Password : 1;
unsigned char To : 1;
unsigned char CC : 1;
unsigned char BCC : 1;
unsigned char From : 1;
unsigned char Subject : 1;
unsigned char OtherHeaders : 1;
unsigned char Body : 1;
} ROMPointers;
BOOL UseSSL;
WORD ServerPort;
} SMTP_POINTERS;
```

Эта структура указателей настраивает клиента SMTP для отправки сообщения электронной почты. Изначально все указатели будет нулевым. Набор SmtpClient (см. стр. 304) . [ Имя поля ] . SzRAM использовать строку , хранящуюся в памяти, или SmtpClient (см.страница 304) . [ имя поля ] . szROM использовать строку, хранящуюся в ПЗУ . ( Где [ имя поля ] является одним из параметров ниже).

Если диск строка определяется , SMTPClient.ROMPointers . [ Имя поля ] также должен быть установлен в 1, чтобы указать , что это поле должны быть извлечены из ПЗУ вместо ОЗУ.

**Замечания**

При форматировании адрес электронной почты , могут быть использованы стандартный формат SMTP для связывания печати имя. Этот формат помещает печати в кавычки , с адресом следующего в угловых скобках , например, "John Smith"

<john.smith@domain.com>

**Параметры**

**Параметры Описание**

**Сервер** - SMTP сервер для передачи сообщения через

**Имя пользователя** - имя пользователя, используемое при входе в SMTP -сервера , если таковые требуется

**Пароль** - пароль для питания при входе в , если таковые требуется

**TO** - Для адрес назначения для этого сообщения . Может быть через запятую список рассмотрения вопроса о , и / или отформатирована.

**CC** - CC -адреса для этого сообщения , если таковые имеются . Может быть через запятую список

рассмотрения вопроса о , и / или отформатирована.

**BCC** - BCC адресов для этого сообщения , если таковые имеются . Может быть через запятую список

рассмотрения вопроса о , и / или отформатирована.

**From** - от адреса для этого сообщения . Может быть отформатирован .

**Subject** темы в заголовке для этого сообщения .

**OtherHeaders** - Любые дополнительные заголовки для этого сообщения . Каждый дополнительный заголовок , в том числе Последний , должны быть прекращены с парой CRLF .

**Body** - При отправке сообщения из памяти , расположение тела это сообщение в памяти. Оставьте как NULL построить сообщение на лету .

**ROMPointers** - Указывает , какие параметры для чтения из ПЗУ вместо оперативной памяти.

**UseSSL** - Когда STACK_USE_SSL_CLIENT включена, этот флаг вызывает клиенту SMTP сделать подключение к SSL к серверу.

**ServerPort** - ( см. стр. 99 ) (WORD значение ) Указывает порт, который необходимо подключить (см. стр. 171 ) , чтобы дистанционного SMTP -сервер .

//*-*-*-*-*----------

### 10.13.2.3 SMTP_RESOLVE_ERROR Macro

**File**
SMTP.h

**C**
```
#define SMTP_RESOLVE_ERROR (0x8000u) // DNS поиск для SMTP -сервера не удалось
```

**Description**
Поиск DNS для SMTP -сервера не удалось

//*-*-*-*-*----------

#### 10.13.2.4 SMTP_SUCCESS Macro

**File**
SMTP.h

**C**
```
#define SMTP_SUCCESS (0x0000u) //  Сообщение было успешно отправлено
```

**Description**
Сообщение было успешно отправлено

```
//*-*-*-*-*-----------------------------------------
```

#### 10.13.2.5 SMTPBeginUsage Function

**File**
SMTP.h

**C**
```
BOOL SMTPBeginUsage();
```

**Description**
Вызывайте эту функцию перед вызовом любых других интерфейсов API клиента SMTP . Эта функция получает блокировку на SMTP Client , который может только использоваться одним приложением стека за один раз. После того, как приложение завершает работу с клиентом SMTP , он должен вызвать SMTPEndUsage (см. стр. 304) , чтобы освободить свой контроль модуля для любых других приложений , ожидающих .

Эта функция инициализирует все государственные машины и переменные SMTP обратно в состояние по умолчанию .

**Preconditions**
None

**Return Values**
**Return Values Description**
**TRUE** Приложение успешно получило управление модулем

**FALSE** модуль SMTP используется другим приложением. Вызов SMTPBeginUsage снова позже,  после возвращения в основной цикл программы

**Section**
SMTP Function Prototypes

//*-*-*-*-*----------

### 10.13.2.6 SMTPClient Variable

**File**
SMTP.c

**C**
SMTP_POINTERS **SMTPClient**;

**Section**
SMTP Client Public Variables
**********
Глобальный набор SMTP_POINTERS .

Установить эти параметры после успешного вызова SMTPBeginUsage .

//*-*-*-*-*----------

### 10.13.2.7 SMTPEndUsage Function

**File**
SMTP.h

**C**
WORD **SMTPEndUsage**();

**Description**
Вызовите эту функцию , чтобы освободить свой контроль клиентского модуля SMTP как только заявление завершения его использования . Эта функция релизы замок получен SMTPBeginUsage (см. стр. 303 ) , и освобождает клиент SMTP для использования другим приложением.

**Preconditions**
SMTPBeginUsage (см. стр. 303) вернулся ИСТИНА на предыдущем вызове .

**Return Values**
**Return Values Description**
SMTP_SUCCESS ( смотри страницу 303) сообщение успешно было послано
SMTP_RESOLVE_ERROR ( смотри страницу303) SMTP -сервер не может быть решен
SMTP_CONNECT_ERROR ( смотри страницу301)

Связь в сервер SMTP потерпевший неудачу или преждевременно была завершена 1-199 и 300-399 код ответа последнего сервера SMTP

//*-*-*-*-*----------

### 10.13.2.8 SMTPFlush Function

**File**
SMTP.h

**C**
**void SMTPFlush**();

**Returns**
None

**Description**
Сбрасывает socket SMTP и заставляет все данные, которые нужно посылать.

**Remarks**
Эта функция должна быть вызвана только внешне , когда клиент SMTP генерирует сообщение на лету . (То есть, SMTPSendMail (см. стр. 309 ) была вызвана SMTPClient.Body установлен в NULL. )

**Preconditions**
SMTPBeginUsage ( СМОТРИ страницу 303) возвращавшее ИСТИНУ на предшествующем вызове.

//*-*-*-*-*----------

### 10.13.2.9 SMTPIsBusy Function

**File**
SMTP.h

**C**
BOOL **SMTPIsBusy**();

**Description**
Вызовите эту функцию, чтобы определить, если клиент SMTP занят выполнением фоновых задач . Эта функция должна быть вызвана после любой вызов **SMTPSendMail** (см. стр. 309 ) , **SMTPPutDone** (см. стр. 307) , чтобы определить, если стек закончил выполнении своих внутренних задач . Следует также называется до любого вызова **SMTPIsPutReady** (см. стр. 305 ) , чтобы убедиться, что Клиент SMTP не преждевременно отключен . Когда эта функция возвращает значение ЛОЖЬ , то следующий вызов должен быть **SMTPEndUsage** (см. стр. 304) , чтобы освободить модуль и получить код состояния для операции.

**Preconditions**
SMTPBeginUsage ( СМОТРИ страницу 303) возвращавшее TRUE на предшествующем вызове.

**Return Values**
**Return Values Description**
**TRUE** Клиент SMTP занятый с внутренними задачами или посылкой непрерывного сообщения.

**FALSE** Client прекращается и готов к выходу .

//*-*-*-*-*----------

### 10.13.2.10 SMTPIsPutReady Function

**File**
SMTP.h

**C**
WORD **SMTPIsPutReady**();

**Returns**
Количество свободных байтов SMTP TX FIFO.

**Description**
Используйте эту функцию, чтобы определить, сколько данных можно записать клиенту SMTP при генерации непрерывного сообщения.

**Remarks**
Эта функция должна быть вызвана только внешне , когда клиент SMTP генерирует непрерывное сообщение. (То есть, **SMTPSendMail** ( смотри страницу 309), был вызван **SMTPClient.Body** установленное на NULL.)

**Preconditions**
**SMTPBeginUsage** ( СМОТРИ страницу 303) возвращавшее ИСТИНУ на предшествующем вызове, и непрерывное сообщение генерируется. Это требует, что **SMTPSendMail** ( смотри страницу 309), был вызван **SMTPClient.Body** было установлено на NULL.

//*-*-*-*-*----------

### 10.13.2.11 SMTPPut Function

**File**
SMTP.h

**C**

BOOL **SMTPPut**(
BYTE **c**
);

**Description**
Записывает один байт клиенту SMTP .

**Remarks**
Эта функция должна быть вызвана только внешне , когда клиент SMTP генерирует непрерывное сообщение. (То есть, **SMTPSendMail** ( смотри страницу 309), был вызван **SMTPClient.Body** установленное на NULL.)

**Preconditions**
**SMTPBeginUsage** ( СМОТРИ страницу 303) возвращавшее ИСТИНУ на предшествующем вызове.

**Parameters**
**Parameters Description**
c- байт, который нужно записывать

**Return Values**
**Return Values Description**
**TRUE** которое байт успешно был записан
**FALSE** байт не был записан , скорее всего, потому, что буфер переполнен

//*-*-*-*-*----------

### 10.13.2.12 SMTPPutArray Function

**File**
SMTP.h

**C**
WORD **SMTPPutArray**(
BYTE* **Data**,
WORD **Len**
);

**Returns**
Количество письменных байтов. Если менее чем Len, тогда TX FIFO становился бы полным прежде, чем все байты могли бы быть записаны.

**Description**
Записывает серию байтов клиенту SMTP.

**Remarks**
Эта функция должна быть вызвана только внешне , когда клиент SMTP генерирует непрерывное сообщение. (То есть, **SMTPSendMail** ( смотри страницу 309), был вызван **SMTPClient. Body** установленное на NULL.)

**Internal**
**SMTPPut** ( СМОТРИ страницу 306), должно быть использовано вместо **TCPPutArray** ( смотри страницу 545) поскольку "rn." должно точно быть заменено "rn..".

**Preconditions**
**SMTPBeginUsage** ( СМОТРИ страницу 303) возвращавшее ИСТИНУ на предшествующем вызове.

**Parameters**
**Parameters Description**
**Data** данные, которые нужно записывать
**Len** Сколько байтов должно быть записано

//*-*-*-*-*----------

### 10.13.2.13 SMTPPutDone Function

**File**
SMTP.h

**C**
**void SMTPPutDone**();

**Returns**
None

**Description**
Указывает, что непрерывное сообщение завершено.

**Preconditions**
**SMTPBeginUsage** ( СМОТРИ страницу 303) возвращавшее ИСТИНУ на предшествующем вызове, и клиент SMTP сгенерирован непрерывное сообщение. (То есть, **SMTPSendMail** ( смотри страницу 309), был вызван **SMTPClient**. **Body** установленное на NULL.)

//*-*-*-*-*----------

### 10.13.2.14 SMTPPutROMArray Function

**File**
SMTP.h

**C**
WORD **SMTPPutROMArray**(
ROM BYTE* **Data**,
WORD **Len**

);

**Returns**
Количество письменных байтов. Если менее чем Len, тогда TX FIFO становился бы полным прежде, чем все байты могли бы быть записаны.

**Description**
Записывает набор байтов из ПЗУ клиенту SMTP .

**Remarks**
Эта функция должна быть вызвана только внешне , когда клиент SMTP генерирует непрерывное сообщение. (То есть, **SMTPSendMail** ( смотри страницу 309), был вызван **SMTPClient. Body** установленное на NULL.)
Этой функции - aliased на **SMTPPutArray** ( смотри страницу 306) на платформах non-PIC18.

**Internal**
**SMTPPut** ( СМОТРИ страницу 306), должно быть использовано вместо **TCPPutArray** ( смотри страницу 545) поскольку "rn." должно точно быть заменено "rn..".

**Preconditions**
SMTPBeginUsage ( СМОТРИ страницу 303) возвращавшее ИСТИНУ на предшествующем вызове.

**Parameters**
**Parameters Description**
**Data** данные, которые нужно записывать
**Len** Сколько байтов должно быть записано

//*-*-*-*-*----------

### 10.13.2.15 SMTPPutROMString Function

**File**
SMTP.h

**C**
WORD **SMTPPutROMString**(
ROM BYTE* **Data**
);

**Returns**
Количество письменных байтов. Если менее чем длина Данных, тогда TX FIFO становился бы полным прежде, чем все байты могли бы быть записаны.

**Description**
Записывает строку из ПЗУ клиенту SMTP .

**Remarks**

Эта функция должна только вызываться внешне когда клиент SMTP генерирует непрерывное сообщение. (То есть, **SMTPSendMail** ( смотри страницу 309), был вызван **SMTPClient**.Тело установленное на NULL.)

**Internal**
**SMTPPut** ( СМОТРИ страницу 306), должно быть использовано вместо **TCPPutString** ( смотри страницу 547) поскольку "rn." должно точно быть заменено "rn..".

**Preconditions**
**SMTPBeginUsage** ( СМОТРИ страницу 303) возвращавшее ИСТИНУ на предшествующем вызове.

**Parameters**
**Parameters Description**
Данные, которые нужно записывать

//*-*-*-*-*----------

### 10.13.2.16 SMTPPutString Function

**File**
SMTP.h

**C**
WORD **SMTPPutString**(
BYTE* **Data**
);

**Returns**
Количество письменных байтов. Если менее чем длина Данных, тогда TX FIFO становился бы полным прежде, чем все байты могли бы быть записаны.

**Description**
Записывает строку клиенту SMTP.

**Remarks**
Эта функция должна только вызываться внешне когда клиент SMTP генерирует непрерывное сообщение. (То есть, **SMTPSendMail** ( смотри страницу 309), был вызван **SMTPClient**.Тело установленное на NULL.)

**Internal**
**SMTPPut** ( СМОТРИ страницу 306), должно быть использовано вместо **TCPPutString** ( смотри страницу 547) поскольку "rn." должно точно быть заменено "rn..".

**Preconditions**
SMTPBeginUsage ( СМОТРИ страницу 303) возвращавшее ИСТИНУ на предшествующем вызове.

**Parameters**

**Parameters Description**
Данные, которые нужно записывать

//*-*-*-*-*-----------------------------------------

### 10.13.2.17 SMTPSendMail Function

**File**
SMTP.h

**C**
**void SMTPSendMail**();

**Returns**
None

**Description**
Эта функция начинает конечный автомат , который выполняет фактическую передачу сообщения. Вызовите эту функцию в конце концов области в **SMTPClient** ( смотри страницу 304), установлены.

**Preconditions**
**SMTPBeginUsage** ( СМОТРИ страницу 303) возвращавшее ИСТИНУ на предшествующем вызове.

//*-*-*-*-*-----------------------------------------

### 10.13.3 SMTP Client Stack Members

**Functions**
**Name Description**
SMTPTask ( see page 310) Выполняет любые незаконченные задачи клиента SMTP

**Module**
SMTP Client ( see page 297)

**Description**
Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные .

//*-*-*-*-*-----------------------------------------

#### 10.13.3.1 SMTPTask Function

**File**
SMTP.h

**C**
```
void SMTPTask();
```

**Returns**
None

**Description**
Эта функция обрабатывает периодические задачи, связанные с клиентом SMTP , такие как обработка первоначальному подключению и команду последовательности.

**Замечания**

Эта функция действует как задачи (по аналогии с одним в OCPB ) . Он выполняет свою задачу в кооперативном образом , и главный Приложение должно вызвать эту функцию несколько раз, чтобы убедиться, что все открытые или новые соединения подаются своевременно .

**Preconditions**
None

//*-*-*-*-*----------

### 10.13.4 SMTP Client Internal Members

**Functions**
**Name Description**
**FindEmailAddress** ( смотри страницу 311)
Ищет строку для адреса эл.почты.

**FindROMEmailAddress** ( СМОТРИ страницу 312) Ищет строку ROM для адреса эл.почты.

**Macros**
**Name Description**
**SMTP_PORT** По умолчанию порта для неопределенного

**SMTP_SERVER_REPLY_TIMEOUT** ( смотри страницу 314) Как долго ждать, прежде чем принять чтобы связь сбросить (по умолчанию 8 секунд )

**Module**
SMTP Client ( see page 297)

**Variables**
**Name Description**
**CRPeriod** ( см. стр. 311) государственной машины для замены CR LF периода, использованные **SMTPPut** (см.страница 306) прозрачно заменить " р-н . " с " р-н .. "

**MySocket** (см. стр. 312) Разъем используется в настоящее время клиентом SMTP

**PutHeadersState** (см.страница 312) Государственная машина для написания заголовков сообщений SMTP

**ResponseCode** (см. стр.313) Код ответа от сервера , когда существует ошибка

**RXParserState** (см. стр.313) Государственная машина для разбора входящих ответов

**SMTPFlags** (см. стр. 314) Внутренние флаги , используемые SMTP Client

**SMTPServer** (см. стр.314) IP -адрес удаленного SMTP -сервера

**SMTPState** (см. стр. 315 ) Сообщение конечного автомата для SMTP Client

**TransportState** (см. стр.316) Состояние транспорта для SMTP Client

**описание**

Следующие функции и переменные предназначены для внутреннего использования модулем SMTP Client .

//*-*-*-*-*----------

### 10.13.4.1 CRPeriod Variable

**File**
SMTP.c

**C**
```
union {
BYTE * Pos;
enum {
CR_PERIOD_SEEK_CR = 0,
CR_PERIOD_SEEK_LF,
CR_PERIOD_SEEK_PERIOD,
CR_PERIOD_NEED_INSERTION
} State;
} CRPeriod;
```

**Members**
**Members Description**
**CR_PERIOD_SEEK_CR** = 0 состояние Ожидания, ждущее 'r

**CR_PERIOD_SEEK_LF** r" записан, так что контрольный следующий байт для 'n
**CR_PERIOD_SEEK_PERIOD** rn" записан, так что контрольный следующий байт для '.
**CR_PERIOD_NEED_INSERTION** "rn." записан, так что дополнительное '.' должно быть записано перед следующим байтом.

**Description**
Государственная машина для замены Периода CR LF Использованной **SMTPPut** ( смотри страницу 306), чтобы точно заменять "rn." с "rn.."

//*-*-*-*-*----------

### 10.13.4.2 FindEmailAddress Function

**File**
SMTP.c

**C**
```
static BYTE * FindEmailAddress(
BYTE * str,
WORD * wLen
);
```

**Returns**
Указатель в адрес эл.почты

**Description**
Эта функция находит адрес электронной почты в строку. Она используется внутри клиентом SMTP для разбора из фактический адрес от от и до струн , так что MAIL FROM и RCPT TO команд могут быть отправлены на SMTP -сервер.

**Preconditions**
SMTPBeginUsage ( Параметры
смотри страницу 303) возвращавшее ИСТИНУ на предшествующем вызове.

**Parameters**
**Parameters Description**
**str** строка, чтобы искать адрес эл.почты
**wLen** длину str

**Section**
SMTP Client Internal Function Prototypes

//*-*-*-*-*----------

### 10.13.4.3 FindROMEmailAddress Function

**File**
SMTP.c

**C**
**static** ROM BYTE * **FindROMEmailAddress**(
ROM BYTE * **str**,
WORD * **wLen**
);

**Returns**
Указатель в адрес эл.почты

**Description**
Эта функция находит адрес электронной почты в строку. Она используется внутри клиентом SMTP для разбора из фактический адрес от от и до струн , так что MAIL FROM и RCPT TO команд могут быть отправлены на SMTP -сервер.

**Preconditions**
SMTPBeginUsage ( СМОТРИ страницу 303) возвращавшее ИСТИНУ на предшествующем вызове.

**Parameters**
**Parameters Description**
**str** строка ROM, чтобы искать адрес эл.почты wLen длину str wLen the length of str

//*-*-*-*-*----------

### 10.13.4.4 MySocket Variable

**File**
SMTP.c

**C**
TCP_SOCKET **MySocket** = INVALID_SOCKET;

**Description**
Розетка к настоящему времени в использовании клиентом SMTP

//*-*-*-*-*----------

### 10.13.4.5 PutHeadersState Variable

**File**
SMTP.c

**C**
```
enum {
PUTHEADERS_FROM_INIT = 0,
PUTHEADERS_FROM,
PUTHEADERS_TO_INIT,
PUTHEADERS_TO,
PUTHEADERS_CC_INIT,
PUTHEADERS_CC,
PUTHEADERS_SUBJECT_INIT,
PUTHEADERS_SUBJECT,
PUTHEADERS_OTHER_INIT,
PUTHEADERS_OTHER,
PUTHEADERS_DONE
} PutHeadersState;
```

**Members**
**Members Description**

**PUTHEADERS_FROM_INIT** = 0 Подготовка отправить из заголовка

**PUTHEADERS_FROM** Отправка из заголовка

**PUTHEADERS_TO_INIT** Подготовка отправить в заголовок

**PUTHEADERS_TO** Отправка Для заголовка

**PUTHEADERS_CC_INIT** Подготовка отправить заголовок CC

**PUTHEADERS_CC** Отправка заголовок CC

**PUTHEADERS_SUBJECT_INIT** Подготовка отправить заголовок темы

**PUTHEADERS_SUBJECT** Отправка заголовок темы

**PUTHEADERS_OTHER_INIT** Подготовка отправить дополнительные заголовки

**PUTHEADERS_OTHER** Отправка дополнительные заголовки

**PUTHEADERS_DONE** урон писать все заголовки

**Description**
Государственная машина для написания заголовков сообщений SMTP

```
//*-*-*-*-*-------------------------------------------
```

### 10.13.4.6 ResponseCode Variable

**File**
SMTP.c

**C**

WORD **ResponseCode**;

**Description**
Код Ответа из сервера когда ошибка существует

//*-*-*-*-*-----------------------------------------

### 10.13.4.7 RXParserState Variable

**File**
SMTP.c

**C**
```
enum {
RX_BYTE_0 = 0,
RX_BYTE_1,
RX_BYTE_2,
RX_BYTE_3,
RX_SEEK_CR,
RX_SEEK_LF
} RXParserState;
```

**Description**
Государственная машина для разбора входящих ответов

//*-*-*-*-*-----------------------------------------

### 10.13.4.8 SMTP_PORT Macro

**File**
SMTP.c

**C**
```
#define SMTP_PORT 25 //  чтобы использоваться когда неопределенно
```

**Description**
По умолчанию порт для использования при неопределенный

//*-*-*-*-*-----------------------------------------

### 10.13.4.9 SMTP_SERVER_REPLY_TIMEOUT Macro

**File**
SMTP.c

**C**
```
#define SMTP_SERVER_REPLY_TIMEOUT (TICK_SECOND*8) // Как долго ждать , прежде чем предполагая, что соединение было закрыто ( по умолчанию 8 секунд )
```

**Description**
Как долго ждать , прежде чем приступить подключение уронили ( по умолчанию 8 секунд )

//*-*-*-*-*----------

### 10.13.4.10 SMTPFlags Variable

**File**
SMTP.c

**C**
```
union {
BYTE Val;
struct {
unsigned char RXSkipResponse : 1;
unsigned char SMTPInUse : 1;
unsigned char SentSuccessfully : 1;
unsigned char ReadyToStart : 1;
unsigned char ReadyToFinish : 1;
unsigned char ConnectedOnce : 1;
unsigned char filler : 2;
} bits;
} SMTPFlags;
```

**Description**
Внутренние флаги , используемые SMTP Client

//*-*-*-*-*----------

### 10.13.4.11 SMTPServer Variable

**File**
SMTP.c

**C**
```
IP_ADDR SMTPServer;
```

**Description**
АДРЕС IP дистанционного сервера SMTP

//*-*-*-*-*----------

### 10.13.4.12 SMTPState Variable

**File**
SMTP.c

**C**

```
enum {
SMTP_HOME = 0,
SMTP_HELO,
SMTP_HELO_ACK,
SMTP_AUTH_LOGIN,
SMTP_AUTH_LOGIN_ACK,
SMTP_AUTH_USERNAME,
SMTP_AUTH_USERNAME_ACK,
SMTP_AUTH_PASSWORD,
SMTP_AUTH_PASSWORD_ACK,
SMTP_MAILFROM,
SMTP_MAILFROM_ACK,
SMTP_RCPTTO_INIT,
SMTP_RCPTTO,
SMTP_RCPTTO_ACK,
SMTP_RCPTTO_ISDONE,
SMTP_RCPTTOCC_INIT,
SMTP_RCPTTOCC,
SMTP_RCPTTOCC_ACK,
SMTP_RCPTTOCC_ISDONE,
SMTP_RCPTTOBCC_INIT,
SMTP_RCPTTOBCC,
SMTP_RCPTTOBCC_ACK,
SMTP_RCPTTOBCC_ISDONE,
SMTP_DATA,
SMTP_DATA_ACK,
SMTP_DATA_HEADER,
SMTP_DATA_BODY_INIT,
SMTP_DATA_BODY,
SMTP_DATA_BODY_ACK,
SMTP_QUIT_INIT,
SMTP_QUIT
} SMTPState;
```

Пользователи

Пользователи Описание

**SMTP_HOME** = 0 Idle начальное состояние для клиента SMTP ( приложение готовится сообщение )

**SMTP_HELO HELO** направляется на сервер

**SMTP_HELO_ACK** поступила на ACK для HELO

**SMTP_AUTH_LOGIN** Запрос войти в

**SMTP_AUTH_LOGIN_ACK** Вход запросу принятой

**SMTP_AUTH_USERNAME** Отправка имя пользователя

**SMTP_AUTH_USERNAME_ACK** Имя пользователя принимаются

**SMTP_AUTH_PASSWORD** Отправка пароль

**SMTP_AUTH_PASSWORD_ACK** Пароль было принято

**SMTP_MAILFROM** Отправка Первичная MAIL FROM команды

**SMTP_MAILFROM_ACK MAIL FROM** был принят

**SMTP_RCPTTO_INIT** Подготовка отправить RCPT TO

**SMTP_RCPTTO** Отправка RCPT TO команды

**SMTP_RCPTTO_ACK** RCPT TO был принят

**SMTP_RCPTTO_ISDONE** урон отправки RCPT TO команд

**SMTP_RCPTTOCC_INIT** Подготовка отправить RCPT TO команд CC

**SMTP_RCPTTOCC** Отправка RCPT TO команд CC

**SMTP_RCPTTOCC_ACK** RCPT TO CC был принят

**SMTP_RCPTTOCC_ISDONE** урон отправки RCPT TO CC

**SMTP_RCPTTOBCC_INIT** Подготовка отправить RCPT TO команд BCC

**SMTP_RCPTTOBCC** Отправка RCPT TO команд BCC

**SMTP_RCPTTOBCC_ACK** RCPT TO BCC был принят

**SMTP_RCPTTOBCC_ISDONE** урон отправки RCPT TO BCC

**SMTP_DATA** Отправка команды DATA

**SMTP_DATA_ACK** ДАННЫЕ Команда accpted

**SMTP_DATA_HEADER** Отправка заголовков сообщений

**SMTP_DATA_BODY_INIT** Подготовка к теле сообщения

**SMTP_DATA_BODY** Отправка тело сообщения

**SMTP_DATA_BODY_ACK** Тело Сообщение принимаются

**SMTP_QUIT_INIT** Отправка QUIT команду

**SMTP_QUIT QUIT принято, закрытие соединения**

**Description**
Государственная машина Сообщения для Клиента SMTP

//*-*-*-*-*----------

### 10.13.4.13 TransportState Variable

**File**
SMTP.c

**C**
```
enum {
TRANSPORT_HOME = 0,
TRANSPORT_BEGIN,
TRANSPORT_NAME_RESOLVE,
TRANSPORT_OBTAIN_SOCKET,
TRANSPORT_SECURING_SOCKET,
TRANSPORT_SOCKET_OBTAINED,
TRANSPORT_CLOSE
} TransportState;
```

**Пользователи**

**Пользователи Описание**

**TRANSPORT_HOME** = 0 Состояние покоя

**TRANSPORT_BEGIN** Подготовка сделать связь

**TRANSPORT_NAME_RESOLVE** Решениеадрес SMTP -сервер

**TRANSPORT_OBTAIN_SOCKET** Получение разъем для подключения SMTP

**TRANSPORT_SECURING_SOCKET** Обеспечение сокет для SMTP через соединение SSL

Подключение **TRANSPORT_SOCKET_OBTAINED** SMTP успешным

**TRANSPORT_CLOSE STMP сокет закрыт**

**Description**
Укажите транспорт для Клиента SMTP

## 10.14 Reboot

Модуль Перезагрузка позволит пользователю удаленно перезагрузить ПИК микроконтроллера , на котором выполняется стек TCP / IP. Эта функция в основном используется для загрузчика приложений , которые необходимо сбросить микроконтроллер , чтобы войти в раздел загрузчик кода. Это Модуль будет выполнить задачу , которая прослушивает указанный порт UDP для пакета , а затем перезагружает , если он получает один . Порт может быть сконфигурированы в **Reboot**.c со следующим макроса :

#define REBOOT_PORT 69

Для повышения безопасности можно ограничить возможности перезагрузки для пользователей в той же подсети , указав следующий макрос в

//*-*-*-*-*----------

### 10.14.1 Reboot Stack Members

**Функции**

**Имя Описание**

**RebootTask** (см. стр. 317 ) Проверяет входящей трафик через порт 69. Сброс ПИК , если принятый 'R' .

**макрос**

**Имя Описание**

**REBOOT_PORT** (см. стр. 318 ) UDP TFTP порт

**REBOOT_SAME_SUBNET_ONLY**(См. стр. 318 ) Для повышения безопасности , вы можете ограничить возможности перезагрузки только пользователи на том же IP- подсети . Определить **REBOOT_SAME_SUBNET_ONLY** для того, чтобы это ограничение доступа.

**Модуль**

Перезагрузка ( смотри страницу 317)

**Описание**

Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные .

//*-*-*-*-*----------

#### 10.14.1.1 RebootTask Function

**File**
Reboot.h

**C**
```
void RebootTask();
```

**Side Effects**
None

**Returns**
None

**Описание**

Проверяет входящей трафик через порт 69. Сброс ПИК , если принятый 'R' .

**Замечания**

Этот модуль в основном для использования с загрузчиком Ethernet. При сбросе , загрузчик Ethernet может взять под свой контроль для  Второй и пусть обновление прошивки состоится .

**Предпосылки**

Стек инициализирован()

//*-*-*-*-*----------

### 10.14.1.2 REBOOT_PORT Macro

**File**
Reboot.c

**C**
```
#define REBOOT_PORT 69 // UDP TFTP port
```

**Description**
UDP TFTP port

//*-*-*-*-*----------

### 10.14.1.3 REBOOT_SAME_SUBNET_ONLY Macro

**File**
Reboot.c

**C**

**#define REBOOT_SAME_SUBNET_ONLY**

**Описание**

Для улучшенной безопасности, Вы могли захотеть ограничить возможности перезагрузки только пользователям в той же подсети IP. Определите REBOOT_SAME_SUBNET_ONLY, для того, чтобы это ограничение доступа.

//*-*-*-*-*----------

**10.15 SNMP**

**Файлы**

**Имя Описание**

| |
|---|
| SNMPv3.c (см. стр. 440 )<br>• простой протокол управления сетью (SNMP ) Версия 3 Агент<br>• Модуль для Microchip Стек TCP / IP<br>• - Обеспечивает SNMPv3 API для делать вещи<br>• - Ссылка: RFC, 3410 , 3411, 3412 , 3413 , 3414 |

| |
|---|
| SNMPv3.h (см. стр. 443 )<br>• простой протокол управления сетью (SNMP ) Версия 3 Агент<br>• Модуль для Microchip Стек TCP / IP<br>• - Обеспечивает SNMPv3 API для делать вещи<br>• - Ссылка: RFC, 3410 , 3411, 3412 , 3413 , 3414 |

| |
|---|
| SNMPv3USM.c (см. стр. 447 )<br>• простой протокол управления сетью (SNMP ) Версия 3 Агент<br>• Модуль для Microchip Стек TCP / IP |

http://www.microchip.com/

- - Обеспечивает SNMPv3 API для делать вещи
- - Ссылка: RFC, 3410 , 3411, 3412 , 3413 , 3414

SNMP.c (см. стр. 449 )

- простой протокол управления сетью (SNMP ) Версия 1 Агент
- простой протокол управления сетью (SNMP ) версии 2 сообщество Агентное
- Модуль для Microchip Стек TCP / IP
- - Обеспечивает SNMP API для делать вещи
- - Ссылка: RFC 1157 (для SNMP V1 )
- RFC 3416 (для SNMPv2C )

SNMP.h (см. стр. 451 )

- SNMP DEFS для Microchip Стек TCP / IP

**Функции**

**Имя Описание**

**_IsSNMPv3ValidStructure** (См. стр. 374 ) Decode переменную структуру длины.

**_Snmpv3IsValidAuthStructure** (См. стр. 374 ) Decode переменную структуру длины.

**_Snmpv3IsValidInt** (См. стр. 375 ) Проверяет переменную типа данных как INT и извлекает его значение.

**FindOIDsFromSnmpV3Request** (см. стр. 376 ) Находит количество varbinds в списке VarBind , полученной в PDU SNMPv3 .

**getSnmpV2GenTrapOid** (см. стр. 376 ) постановляет общего кода ловушки для общего ловушки OID.

**IsSnmpV3ASNNull** (см. стр. 377 ) Проверяет тип значения , как ASN_NULL (см. стр. 344 ) .

**IsSnmpv3ValidOID** (см. стр. 377 ) Заполняет OID тип , длину и подъязычная строку из получил PDU .

**IsSNMPv3ValidStructure** (см. стр. 378 ) Decode переменную структуру длины.

**ProcessGetBulkVar** (см. стр. 378 ) Этот рутинный процессSNMPv2c Получить общий запрос .

**ProcessGetNextVar** (см. стр. 379 ) Получает следующий узел из базы данных MIB .

**ProcessGetVar** (см. стр. 380 ) Процессы SNMP Получить запроса PDU .

**ProcessSnmpv3MsgData** (см. стр. 380 ) Эта процедура обрабатывает запрос SNMPv3 и параллельно создает PDU ответа .

**SNMPGetExactIndex** (см. стр. 381 ) для поиска точного индекса узла в случае последовательности переменная .

**SNMPIdRecrdValidation** (см. стр. 381 ) используется для ограничения доступа динамичный и не динамическая OID струны для perticular SNMP версии .

**SNMPIsValidSetLen** (см. стр. 382 ) Проверяет набор переменную длину данных в тип данных.

**Snmpv3AESDecryptRxedScopedPdu** (см. стр.383) Входящий SNMPv3 областью действия расшифровку PDU с использованием AES расшифровка протокола .

**Snmpv3AESEncryptResponseScopedPdu** (см.страница 383) Исходящие SNMPv3 областью действия PDU шифрования с использованием AES Протокол шифрования .

**Snmpv3AuthenticateRxedPduForDataIntegrity** (см.страница 384 ) Аутентификация на входящий SNMPv3 USM PDU с помощью MD5 или SHA

**Snmpv3AuthenticateTxPduForDataIntegrity** (см.страница 384 ) Подлинность исходящего SNMPv3 USM PDU с помощью MD5 или SHA

**Snmpv3AuthKeyZeroing2HmacBufLen64** (см. страница 385 ) Коврик нуля до ключа аутентификации локализованной буфера.

**Snmpv3BufferPut** (см. стр. 385)  данных Копии BYTE с динамической памятью буфера .

**Snmpv3CmprTrapSecNameAndSecLvlWithUSMDb**(См. стр. 386 ) Регулярное найти индекс имени пользователя в пользовательском база данных таблицы .

**Snmpv3ComputeHMACIpadOpadForAuthLoclzedKey** (См. стр. 386 ) Вычислить HMAC внутренняя и внешняя площадка для авторизации локализованы ключ .

**Snmpv3ComputeHmacMD5Digest** (см. стр. 387 ) Вычислить HMAC - MD5 код аутентификации

**Snmpv3ComputeHmacShaDigest** (см. стр. 388 ) Вычислить HMAC - Ша код аутентификации

**Snmpv3ComputeMd5HmacCode** (см. стр. 388 ) Вычислить HMAC - MD5 код аутентификации

**Snmpv3ComputeShaHmacCode** (см. стр. 389 ) Вычислить HMAC - Ша код аутентификации

**Snmpv3FormulateEngineID** (см. стр. 389 ) формулирует snmpEngineID (см. стр. 428) для SNMPv3 двигатель .

**Snmpv3FreeDynAllocMem** (см. стр. 390 ) Выделенные освобождение динамической памяти осуществляется это рутина.

**Snmpv3GetAuthEngineTime** (см. стр. 390 ) Обновляет переменной времени двигатель SNMP ' **snmpEngineTime** (см. стр. 429) " для SNMPv3 двигатель .

**Snmpv3GetBufferData** (см. стр. 391 ) Читает данные байт из динамически выделяемой памяти буфера .

**Snmpv3GetSecurityLevel** (см. стр. 391) Получить уровень безопасности от аутентификации и конфиденциальности типа.

**Snmpv3GetTrapSecurityLevel** ( см. стр. 392 ) обычные найти в докладе , Auth и приватности флаги Настройки в ловушке.

**Snmpv3Init** ( см. стр. 392) инициализации SNMPv3 .

**Snmpv3InitializeUserDataBase** (см. стр. 393) SNMPv3 базы данных глобальной пользователь Intialize умолчанию .

**Snmpv3IsValidAuthStructure** (см. стр. 393 ) Decode переменную структуру длины.

**Snmpv3IsValidInt** (см. стр. 394 ) Проверяет переменную типа данных как INT и извлекает его значение.

**Snmpv3MsgProcessingModelProcessPDU** (см.страница 394 ) Эта процедура собирает или заполняет сообщение обработка модель детали от полученных Запрос PDU SNMPv3 или к PDU ответа соответственно.

**Snmpv3Notify** (см. стр. 395 ) создает и отправляет SNMPv3 перехваты PDU .

**Snmpv3Pswd2LocalizedAuthKeyMD5Hashing** (см. страница 395) Преобразование MD5 Auth пароль для локализованной Key помощью SNMPEngineID .

**Snmpv3Pswd2LocalizedAuthKeySHAHashing** (см.страница 396 ) Преобразование SHA Auth пароль для локализованной Key помощью SNMPEngineID .

**Snmpv3ReportPdu** (см. стр. 396 ) Создает доклада PDU infomration для отчета PDU.

**Snmpv3ScopedPduProcessing** (см. стр. 397 ) Эта процедура собирает областью действия информацию заголовка PDU из принятого запроса PDU или заполняет SNMPv3 к PDU ответа соответственно.

**Snmpv3SetErrorStatus** (см. стр. 397 ) указан SNMPv3 статус ошибки в PDU ответа.

**Snmpv3TrapScopedpdu** (см. стр. 398 ) перехваты PDU области видимости строительство заголовка PDU.

**Snmpv3UserSecurityModelProcessPDU** (см. стр.399) Эта процедура собирает или заполняет модель безопасности ПАРАМЕТРЫ детали от полученных SNMPv3 запросить PDU или в PDU ответа соответственно.

**Snmpv3UsmAesEncryptDecryptInitVector** (см.страница 399 ) Шифрование AES и дешифрование инициализации вектор. (RFC 3826)

**Snmpv3UsmOutMsgAuthenticationParam** (см.страница 400 ) Оба MD5 и SHA1 используется для исходящего сообщения аутентификации .

**Snmpv3USMOutMsgPrivParam** (см. стр. 400 ) SNMP USM из сообщение использует конфиденциальности протокол (RFC 3826 )

**Snmpv3UsmSnmpEngnAuthPrivPswdLocalization** (см. стр. 400) Преобразование Auth и Priv пароль для локализованной Key используя SNMPEngineID .

**Snmpv3ValidateEngineId** (см. стр. 401) Проверка двигателя ID .

**Snmpv3ValidateSecNameAndSecLvl** (см. стр.401) Vaidate имя безопасности с уровнем безопасности.

**Snmpv3ValidateSecurityName** (см. стр. 402 ) Проверка SNMPv3 имя пользователя или имя безопасности .

**макрос**

**Имя Описание**

**AUTH_LOCALIZED_PASSWORD_KEY_LEN** (см. стр. 434 ) SNMPv3 аутентификации ( см. стр. 89 ) Локализованная passwed размер ключа lenegth

**INVALID_INDEX** (см. стр. 434 ) Этот макрос INVALID_INDEX .

**IS_SNMPV3_AUTH_STRUCTURE** (см. стр. 435 ) Этот макрос IS_SNMPV3_AUTH_STRUCTURE .

**MSG_AUTHORITATIVE_HEADER_LEN** ( см. стр. 435 ) Длина SNMPv3 authoratative MSG длина заголовка = длина заголовка (2 + 2 байт) + engineID ( **snmpEngnIDLength** ( См. стр. 429) байт)

• загрузки двигателя (4 байта) + двигатель время (4 байта) + имя безопасности ( **securityPrimitivesOfIncomingPdu** (см. стр. 427 ) значение) + аутентификацию

**параметры**

( **snmpOutMsgAuthParamLen** (см.страница 431 ) значение ) + конфиденциальности параметры

( **snmpOutMsgAuthParamLen** (см.431 ) Значение страница )

**MSGGLOBAL_HEADER_LEN** ( см. стр. 435 ) Длина ГЗС заголовке SNMPv3 (x) = Длина заголовка ( 2 байта )

• Размер MSGID (тип (1 байт) + длина стоимости (1 байт) значение 4 байта)

• msgMAXSIZE (тип + длина значение значение 4 байта)

• флаг MSG (тип + длина значения 1 Значение байта )

• Тип модели безопасности (тип + длина значение +1 значение байт)

**PRIV_LOCALIZED_PASSWORD_KEY_LEN** (см. стр. 436 )

| # define PRIV_LOCALIZED_PASSWORD_KEY_LEN<br><br>16 SNMPv3 конфиденциальности длина Pasword ключ размер |
|---|

**REPORT_RESPONSE** (см. стр. 436 ) Этот макрос REPORT_RESPONSE .

**SNMP_ENGINE_MAX_MSG_SIZE** (см. стр. 436 ) SNMP_ENGINE_MAX_MSG_SIZE является определяется как минимум по максимуму значения размера MSG поддерживается среди всех транспорты доступны и поддерживается движком .

**SNMP_MAX_MSG_SIZE** (см. стр. 436 ) SNMP MIN и MAX сообщение 484 байт Согласно RFC 3411

**snmpEngineMaxMessageSize** (см.страница 428 ) и RFC 1157 (раздел 4 - Спецификация протокола ) и реализация поддерживает более 484 всякий раз, когда это возможно .

**SNMP_MAX_OID_LEN_MEM_USE** (см. стр. 437 ) Этот макрос будет использоваться , чтобы избежать SNMP Коррупция буфер OID памяти

**SNMP_TRAP_COMMUNITY_MAX_LEN_MEM_USE** (см. стр. 437) Этот макрос будет использоваться , чтобы избежать SNMP Коррупция буфер OID памяти

**SNMP_V3** (см. стр. 437 ) Этот макрос SNMP_V3 .

**SNMPV3_AUTH_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE** (См. стр. 437 ) Аутентификации SNMPv3 локализованы ключ Длина для проверки памяти

**SNMPV3_H** (см. стр. 438 ) Этот макрос SNMPV3_H .

**SNMPV3_PRIV_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE** (См. стр. 438 ) SNMPv3 конфиденциальности размер ключа длина проверка памяти

**SNMPV3_USER_SECURITY_NAME_LEN_MEM_USE** (см. стр.438) Защита пользователей длина имени для памяти утверждение

**SNMPV3_USM_MAX_USER** (см. стр. 438 ) Этот макрос SNMPV3_USM_MAX_USER .

**SNMPV3MSG_AUTHENTICATION_FAIL** (см. стр. 439 ) Этот макрос SNMPV3MSG_AUTHENTICATION_FAIL.

**SNMPV3MSG_AUTHENTICATION_SUCCESS** (см. стр. 439 ) Этот макрос SNMPV3MSG_AUTHENTICATION_ SUCCESS.

**USER_SECURITY_NAME_LEN** (см. стр. 439 ) SNMPv3 длина безопасности имя пользователя

**Структуры , запись , Перечисления**

**Имя Описание**

**AccessCtrlSubSysIsAccessAllowed** (см. стр. 403) Приложение является типичным клиентом услуги доступа к подсистеме управления . В следующем примитиве обеспечивается контроль доступа Подсистеме для проверки разрешения доступа: statusInformation (см. стр. 417 ) = - успех или **errorIndication**

**dispatcherProcessPdu** (см. стр. 404) процесс поступления запроса или уведомления PDU, Диспетчер предоставляет следующие примитивы, чтобы пройти на входящий SNMP PDU в приложение.

**dispatcherStatusInfo** (см. стр. 404 ) Создание исходящего запроса или уведомления statusInformation (см. стр. 417 ) = - sendPduHandle если успех - errorIndication если отказ

**dispathcerReturnResponsePdu** (см.страница 405 ) Генерация исходящих ответов PDU Диспетчера предоставляет следующие примитивы для Приложения вернуть SNMP Response PDU в PDU

**Диспетчер**:

результат = успех или неудача

**MsgProcModPrepareDataElements** (см. стр. 406) Подготовка элементов данных из входящего сообщения SNMP Сообщение Обработка Подсистема предоставляет эту услугу примитивный для подготовки абстрактные элементы данных из входящего SNMP сообщение : результат = - успех или errorIndication

**MsgProcModPrepareOutgoingMessage** (См. стр. 407 ) Подготовка исходящей SNMP запроса или уведомления сообщение Сообщение Обработка Подсистема предоставляет эту услугу примитивный для подготовки исходящего SNMP запроса или уведомления сообщение

**MsgProcModPrepareResponseMessage** (См. стр. 407 ) Подготовка исходящего SNMP ответное сообщение Сообщение Обработка Подсистема предоставляет эту услугу примитивный для подготовки исходящего SNMP ответное сообщение : результат = - Успех или неудача

**processResponsePdu** (см. стр. 408) обрабатывать входящие PDU отклика PDU Диспетчер предоставляет следующие примитивны, чтобы пройти входящие SNMP отклика PDU к приложению :

**registerContextEngineID** (см. стр.417)Успех или errorIndication

**SecuritySysGenerateRequestMsg** (см. стр. 409)Это запись SecuritySysGenerateRequestMsg .

**SecuritySysGenerateResponseMsg** (см. стр. 410)Генерирует ответное сообщение, Подсистема защиты предоставляет следующие примитивны, чтобы генерировать Ответное сообщение :

**StateRelease** (см. стр. 410 ) Государственный выпуска справочная информация Все подсистемы , которые проходят stateReference информацию также обеспечивают примитивный для освобождения памяти , которая содержит ссылается состояние информация

**unregisterContextEngineID** (см. стр. 411 ) Это запись unregisterContextEngineID .

**Типы**

**Имя Описание**

**INOUT_SNMP_PDU** (см. стр. 412 ) Это типа INOUT_SNMP_PDU .

**REPORT_FLAG_AND_SECURITY_LEVEL_FLAGS** (см. стр. 412 ) Это тип REPORT_FLAG_AND_SECURITY_LEVEL_FLAGS.

**SecuritySysProcessIncomingMsg** (см. стр. 413 ) Это типа SecuritySysProcessIncomingMsg .

**SNMP_ENGNID_OCTET_IDENTIFIER_VAL** (см. стр.414) Пятый октет указывает, как остальные ( 6-й и Следующие октета) отформатированы . См. RFC3411 section5 Страница № 41

**SNMPNONMIBRECDINFO** (см. стр. 414 ) Это типа SNMPNONMIBRECDINFO .

**SNMPV3_HMAC_HASH_TYPE** (см. стр. 414 ) Тип хэша рассчитывается

**SNMPV3_MSG_AUTH_SEC_PARAM_RESULT** (см.страница 415) Это тип SNMPV3_MSG_AUTH_SEC_PARAM_RESULT .

**SNMPV3_MSG_PRIV_SEC_PARAM_RESULT** (см.страница 415) Это тип SNMPV3_MSG_PRIV_SEC_PARAM_RESULT .

**SNMPV3_PRIV_PROT_TYPE** (см. стр. 415 ) Это тип SNMPV3_PRIV_PROT_TYPE .

**SNMPV3_REQUEST_WHOLEMSG** (см. стр. 416 ) Это типа SNMPV3_REQUEST_WHOLEMSG .

**SNMPV3_RESPONSE_WHOLEMSG** (см. стр. 416 ) Это типа SNMPV3_RESPONSE_WHOLEMSG .

**snmpV3EngnUserDataBase** (см. стр. 416 ) Это типа snmpV3EngnUserDataBase .

**SNMPV3MSGDATA** (см. стр. 417 ) SNMPv3

**snmpV3TrapConfigDataBase** ( см. стр. 417 ) целевой конфигурации SNMPv3 по отношению к ловушке.

**statusInformation** ( см. стр. 417), успех или errorIndication

**STD_BASED_SNMP_MESSAGE_PROCESSING_MODEL** (См. стр. 418 ) Это тип STD_BASED_SNMP_MESSAGE_PROCESSING _ MODEL.

**STD_BASED_SNMP_SECURITY_MODEL** (см. стр.418) Snmp Сообщение Модель обработки

**STD_BASED_SNMPV3_SECURITY_LEVEL** (см. стр.418) Это тип STD_BASED_SNMPV3_SECURITY_LEVEL .

**USM_SECURITY_LEVEL** (см. стр. 419 ) Это типа USM_SECURITY_LEVEL .

**Переменные**

**Имя Описание**

**authKey_iPad** (см. стр. 421 ) Это переменная authKey_iPad .

**authKey_oPad** (см. стр. 421 ) Это переменная authKey_oPad .

**authoritativeSnmpEngineBoots** (см.страница 421 ) Количество раз , что авторитетный двигатель SNMP имеет ( пере-) инициализации себя с момента ее **snmpEngineID** (см. стр. 428 ) было последней конфигурации .

**authoritativeSnmpEngineTime** (см. страница 422 ) Так как значение Количество секунд

**authoritativeSnmpEngineBoots** ( см. стр. 421 ) объекта последнего изменения

**cipher_text** (см. стр. 422 ) Это переменная cipher_text .

**deciphered_text** (см. стр. 422 ) Это переменная deciphered_text .

**getZeroInstance** (см. стр. 422 ) Эта переменная используется для gext следующего запроса для нулевой например

**gSnmpV3InPduWholeMsgBuf** (см.страница 422 ) Заглушки Динамическая память и PDU данные входящей хранится PDU

**gSnmpV3OUTPduWholeMsgBuf** (см.страница 423) Подробнее заглушки динамической памяти и построена исходящих сохраняются PDU детали

**gSNMPv3PduHeaderBuf** (см. стр.423) Строительство PDU отклика смещение детали

**gSNMPv3ScopedPduDataPos** (см.страница 423) Смещение читать Scoped байтов данных PDU для обработки от динамического заглушки памяти

**gSNMPv3ScopedPduRequestBuf** ( см. стр. 423 ) Хранится запрос областью видимости детали PDU

**gSNMPv3ScopedPduResponseBuf** (см. стр. 424) Обработанные ответ областью видимости детали PDU

**gSnmpv3TrapConfigData** (см. стр.424) Глобальная база данных конфигурации SNMPv3 , который будет использоваться для уведомления ловушки

**gSNMPv3TrapMsgHeaderBuf** (см.страница 424 ) TRAP сообщение строительство заголовка PDU смещение детали

**gSnmpV3TrapOUTPduWholeMsgBuf**(См. стр. 424 ) Подробнее заглушки динамической памяти и построенные ловушки подробнее PDU

**gSNMPv3TrapScopedPduResponseBuf**(См. стр. 424 ) TRAP области видимости строительство PDU смещение детали

**gSNMPV3TrapSecurityLevel** (см.страница 425) Это переменная gSNMPV3TrapSecurityLevel .

**gSnmpv3UserDBIndex** (см. стр.425) Индекс к конкретной ссылкой настроенного в модели безопасности пользователя база данных snmpV3UserDataBase (см. стр. 432 ) .

**gUsmStatsEngineID** (см. стр. 425) глобальной переменной , чтобы узнать, сколько раз SNMPv3 двигатель ID имеет была подтверждена

**hmacAuthKeyBuf** (см. стр. 425 ) Это переменная hmacAuthKeyBuf .

**HmacMd5Digest** (см. стр. 426 ) Это переменная HmacMd5Digest .

**HmacSHADigest** (см. стр. 426 ) Это переменная HmacSHADigest .

**incomingPdu** (см. стр. 426 ) Входящие данные PDU

**incomingSnmpPDUmsgID** (см. стр. 426) Retrived Входящий значение MSG ID от PDU

**ivEncrptKeyOut** (см. стр. 426 ) Это переменная ivEncrptKeyOut .

**md5LocalizedAuthKey** (см. стр. 427 ) Это переменная md5LocalizedAuthKey .

**msgSecrtyParamLenOffset** (см. стр.427) Это переменная msgSecrtyParamLenOffset .

**securityPrimitivesOfIncomingPdu** (см. стр. 427 ) Входящий PDU безопасности примитивные детали.

**SESSION_KEY** (см. стр. 427 ) Это переменная SESSION_KEY .

**sha1LocalizedAuthKey** (см. стр.428) Это переменная sha1LocalizedAuthKey .

**snmpEngineBoots** (см. стр. 428 ) раз , что двигатель SNMP имеет ( пере-) инициализации себя число snmpEngineID (см. стр. 428 ) был последней конфигурации .

**snmpEngineID** (см. стр. 428) Резервирование 32 байт для snmpEngineID как длина строки октетов может варьироваться форму 5 до 32

**snmpEngineMaxMessageSize** (см. страница 428) Максимальный размер сообщения двигатель SNMP может справиться.

**snmpEngineMsgProcessModel** (см.страница 428) Тип сообщения модели обработки используется . Значение Максимальная дальность (2 ^ 31-1 ) , RFC3411

**snmpEngineSecurityModel** (см. стр.429) Тип модели безопасности используется. Значение Максимальная дальность (2 ^ 31-1 ) ,RFC3411

**snmpEngineTime** (см. стр. 429 ), так как стоимости snmpEngineBoots Количество секунд ( см. стр. 428 ) объекта последнего изменения

**snmpEngineTimeOffset** (см. стр.429) Хранит значение времени в секундах после сброса SNMP двигателя

**snmpEngnIDLength** (см. стр. 429) Двигатель ID длина Engine SNMP

**snmpInMsgAuthParamLen** (см. стр.430) Входящий SNMPv3 параметры аутентификации MSG строка 12 байт долго.

**snmpInMsgAuthParamStrng** (см.страница 430) Резервирование 12 байт для входящего аутентификации MSG SNMPv3 параметры .

**snmpInMsgPrivParamLen** (см. стр.430) Параметры Входящий SNMPv3 MSG конфиденциальности строка 8 байт.

**snmpInMsgPrvParamStrng** (см. стр.430) Резервирование 8 байт для входящих SNMPv3 MSG параметров конфиденциальности.

**snmpMsgBufSeekPos** ( см. стр. 430 ) Смещение читать байты данных PDU для обработки от динамической памяти огрызок

**snmpOutMsgAuthParamLen** (см. страница 431 ) Исходящие параметры аутентификации MSG SNMPv3 строка 12 байт долго.

**snmpOutMsgAuthParamStrng** (см.страница 431 ) Резервирование 12 байт для исходящей проверки подлинности MSG SNMPv3 параметры .

**snmpOutMsgPrivParamLen** (см.страница 431 ) Параметры исходящей SNMPv3 MSG конфиденциальности строка 8 байт.

**snmpOutMsgPrvParamStrng** (см.страница 431 ) Резервирование 8 байт для исходящих SNMPv3 MSG параметров конфиденциальности.

**snmpResponseSecurityFlag** (см.страница 432) Тип безопасности для исходящего сообщения в ответ на входящий сообщение .

**snmpSecurityLevel** ( см. стр. 432) Вид ценной бумаги . noAuthNoPriv (0), AuthNoPriv (1), AuthPriv (3)

**snmpV3AesDecryptInitVector** (см.страница 432) 128 Bit

**snmpV3AesEncryptInitVector** (см. страница 432) 128 Bit

**snmpV3UserDataBase** (см. стр.432) Это переменная snmpV3UserDataBase .

**описание**

Простой протокол сетевого управления V2c (община ) осуществление агентом RFC 3416 .

### 10.15.1 SNMP Общественная Пользователи

**Перечисления**

**Имя Описание**

**GENERIC_TRAP_NOTIFICATION_TYPE** (см. страница 327) Это тип GENERIC_TRAP_NOTIFICATION_TYPE.

**VENDOR_SPECIFIC_TRAP_NOTIFICATION_TYPE**(См. стр. 328 ) Это тип VENDOR_SPECIFIC_TRAP_NOTIFICATION_TYPE .

**SNMP_ACTION** (см. стр. 328 ) Этот список SNMP действия через удаленные NMS может выполнять . Это Inforamtion передается приложения через обратного вызова SNMPValidateCommunity (см. стр. 336 ) ( ) . Заявка должна проверить действие при заданных Строка сообщества .

**COMMUNITY_TYPE** (см. стр. 329 ) Это типа COMMUNITY_TYPE .

**Функции**

**Имя Описание**

**SNMPSendTrap** (см. стр.331) Подготовка , проверка удаленного узла , который будет получать ловушку и отправить перехваты PDU .

**SNMPNotify** (см. стр.332) Создает и отправляет перехваты PDU .

**SNMPSetVar** (см. стр.332) Эта процедура Установите переменную MIB с запрашиваемым стоимости.

**SNMPGetVar** (см. стр.333) Используется для получения / сбора OID переменной информации .

**SNMPIsNotifyReady** (см.страница 334 ) Постановляет, учитывая RemoteHost IP адрес в MAC-адрес.

**SNMPNotifyPrepare** (см.страница 334 ) Собирает данные уведомления ловушку и отправить ARP для удаленного хоста.

**SNMPGetNextIndex** (см.страница 335 ) Для поиска следующего индекса узла в случае переменной последовательности .

**SNMPValidateCommunity** (см. стр. 336 ) Проверяет имя сообщества для контроля доступа .

**макрос**

**Имя Описание**

**SNMP_COMMUNITY_MAX_LEN** (См. стр. 337 ) Это максимальная длина сообщества строки. Заявка должна обеспечить что эта длина не наблюдается. Модуль SNMP добавляет один байт дополнительный AFTER SNMP_COMMUNITY_MAX_LEN для добавления пустой символ '0 ' .

**OID_MAX_LEN** (см. стр. 337) Измените это , чтобы соответствовать вашему OID длину строки .

**SNMP_START_OF_VAR** (см. страница 337) Это макро SNMP_START_OF_VAR .

**SNMP_END_OF_VAR** (см. страница 337) Это макро SNMP_END_OF_VAR .

**SNMP_INDEX_INVALID** (см.страница 338 ) Это макро SNMP_INDEX_INVALID .

**TRAP_TABLE_SIZE** (см.страница 338 ) Это макро TRAP_TABLE_SIZE .

**TRAP_COMMUNITY_MAX_LEN** (См. стр. 338 ) Это макро TRAP_COMMUNITY_MAX_LEN .

**NOTIFY_COMMUNITY_LEN** (см. стр. 338 ) Это макро NOTIFY_COMMUNITY_LEN .

**модуль**

SNMP (см. стр. 318 )

**структуры**

**Имя Описание**

**TRAP_INFO** (см. стр.330) Это тип TRAP_INFO .

**Виды**

**Имя Описание**

**SNMP_ID** (см. стр. 336 ) Это переменная ID SNMP OID . Это идентификатор присваивается через MIB файл . только динамическое и AgentID переменные могут contian ID . MIB2BIB утилита обеспечивает это правила, когда BIB был сгенерирован .

**SNMP_INDEX** (см. стр.336) Это тип SNMP_INDEX .

**Unions**

**Имя Описание**

**SNMP_VAL** (см. стр. 329 ) Это типа SNMP_VAL .

**Переменные**

**Имя Описание**

**gSendTrapFlag** (см. стр.330) глобальный флаг , чтобы отправить ловушки

**gSetTrapSendFlag** (см.страница 330 ) # если определена ( SNMP_STACK_USE_V2_TRAP ) || определена ( SNMP_V1_V2_TRAP_WITH_SNMPV3 ) / / если gSetTrapSendFlag == ЛОЖЬ , то последняя переменная VarBind для / / нескольких переменных PDU VarBind структуры или если есть только VarBind переменная отправить . / / Если gSetTrapSendFlag == ИСТИНА , то v2 перехваты PDU ожидает более VarBind переменную . BYTE gSetTrapSendFlag = false; # ENDIF

**gGenericTrapNotification** ( см. стр. 331) Глобальный флаг для Generic уведомления ловушки

**gSpecificTrapNotification** (см. стр. 331) Производитель специфический код ловушка

**gOIDCorrespondingSnmpMibID**(См. стр. 331 ) Gloabal переменная для хранения SNMP ID Bap для OID получил в SNMP запросу.

**описание**

Следующие функции и переменные доступны для применения стека.

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

**10.15.1.1 GENERIC_TRAP_NOTIFICATION_TYPE Перечисление**

**файл**

SNMP.h (см. стр. 451 )

http://www.microchip.com/

**C**

```
typedef enum {
COLD_START = 0x0,
WARM_START = 0x1,
LINK_DOWN = 0x2,
LINK_UP = 0x3,
AUTH_FAILURE = 0x4,
EGP_NEBOR_LOSS = 0x5,
ENTERPRISE_SPECIFIC = 0x6
} GENERIC_TRAP_NOTIFICATION_TYPE;
```

**описание**

Это тип GENERIC_TRAP_NOTIFICATION_TYPE .

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

### 10.15.1.2 VENDOR_SPECIFIC_TRAP_NOTIFICATION_TYPE

**перечисление**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef enum {
VENDOR_TRAP_DEFAULT = 0x0,
BUTTON_PUSH_EVENT = 0x1,
POT_READING_MORE_512 = 0x2
} VENDOR_SPECIFIC_TRAP_NOTIFICATION_TYPE;
```

**описание**

Это тип VENDOR_SPECIFIC_TRAP_NOTIFICATION_TYPE .

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

### 10.15.1.3 SNMP_ACTION Перечисление

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef enum {
SNMP_GET = 0xa0,
```

**SNMP_GET_NEXT** = 0xa1,
**SNMP_GET_RESPONSE** = 0xa2,
**SNMP_SET** = 0xa3,
**SNMP_TRAP** = 0xa4,
**SNMP_V2C_GET_BULK** = 0xa5,
**SNMP_V2_TRAP** = 0xa7,
**SNMPV3_ENCRYPTION** = 0x04,
**SNMP_ACTION_UNKNOWN** = 0
} SNMP_ACTION;

**Пользователи**

**Пользователи Описание**

**SNMP_GET = 0xa0** SNMP GET идентификатор

**SNMP_GET_NEXT = 0xa1** идентификатор Snmp get_next

**SNMP_GET_RESPONSE = 0xA2** Snmp GET_RESPONSE ( см. стр. 347) идентификатор

**SNMP_SET = 0xA3** установки SNMP идентификатор

**SNMP_TRAP = 0xA4** Snmp идентификатор TRAP

**SNMP_V2C_GET_BULK = 0xa5** идентификатор Snmp GET_BULK

**SNMP_V2_TRAP = 0xa7** SNMP v2 Ловушка идентификатор

**SNMP_ACTION_UNKNOWN = 0** Snmp требуемое действие неизвестный

**описание**

Это список SNMP действия через удаленные NMS может выполнять . Это информационные веб передается приложения через обратный вызов **SNMPValidateCommunity** (см. стр. 336) ( ) . Заявка должна проверить действие для данного сообщества строки.

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

**10.15.1.4 COMMUNITY_TYPE Перечисление**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef enum {
READ_COMMUNITY = 1,
WRITE_COMMUNITY = 2,
INVALID_COMMUNITY = 3
} COMMUNITY_TYPE;
```

http://www.microchip.com/

**Пользователи**

**Пользователи Описание**

**READ_COMMUNITY = 1** только для чтения сообщество

**WRITE_COMMUNITY = 2** сообщения прочитанными записи

**INVALID_COMMUNITY = 3** Сообщество недействительным

**описание**

Это тип COMMUNITY_TYPE .

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

**10.15.1.5 SNMP_VAL союз**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef union {
DWORD dword;
WORD word;
BYTE byte;
BYTE v[sizeof(DWORD)];
} SNMP_VAL;
```

**Пользователи**

**Пользователи Описание**

DWORD DWORD ; Двойное значение слова

СЛОВО слово ; значение слова

BYTE байт ; значение байта

BYTE V [ SizeOf (DWORD) ] ; массив

**описание**

Это тип SNMP_VAL .

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

**10.15.1.6 TRAP_INFO Структура**

http://www.microchip.com/

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef struct {
BYTE Size;
struct {
BYTE communityLen;
char community[TRAP_COMMUNITY_MAX_LEN];
IP_ADDR IPAddress;
struct {
unsigned int bEnabled : 1;
} Flags;
} table[TRAP_TABLE_SIZE];
} TRAP_INFO;
```

**Пользователи**

**Пользователи Описание**

**BYTE communityLen** ; Сообщество длина имени

голец  сообщество [ TRAP_COMMUNITY_MAX_LEN ] ;  Название сообщества массив

IP_ADDR IP-адрес ;  IP адрес, по которому ловушка для отправки

неподписанных Int bEnabled : 1 ;  Ловушка включен флаг

**описание**

Это тип TRAP_INFO .

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

**10.15.1.7 gSendTrapFlag Переменная**

**файл**

CustomSNMPApp.c

**C**

BYTE **gSendTrapFlag** = FALSE;

**описание**

глобальный флаг , чтобы отправить ловушки

http://www.microchip.com/

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

**10.15.1.8 gSetTrapSendFlag Переменная**

**файл**

CustomSNMPApp.c

**C**

BYTE **gSetTrapSendFlag** = FALSE;

**описание**

#if defined(SNMP_STACK_USE_V2_TRAP) || defined(SNMP_V1_V2_TRAP_WITH_SNMPV3) //if gSetTrapSendFlag ==
FALSE, то последняя переменная VarBind для / / множественной переменной структурой PDU VarBind или если есть только VarBind переменная отправить .

// if gSetTrapSendFlag == TRUE, то v2 перехваты PDU ожидает более VarBind переменную . BYTE gSetTrapSendFlag = false;
# ENDIF

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

**10.15.1.9 gGenericTrapNotification Переменная**

**файл**

CustomSNMPApp.c

**C**

BYTE **gGenericTrapNotification** = ENTERPRISE_SPECIFIC;

**описание**

Глобальный флаг для Generic уведомления ловушки

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

**10.15.1.10 gSpecificTrapNotification Переменная**

**файл**

CustomSNMPApp.c

**C**

http://www.microchip.com/

BYTE **gSpecificTrapNotification** = VENDOR_TRAP_DEFAULT;

**описание**

Производитель специфический код ловушка

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

**10.15.1.11 gOIDCorrespondingSnmpMibID Переменная**

**файл**

CustomSNMPApp.c

**C**

BYTE **gOIDCorrespondingSnmpMibID** = MICROCHIP;

**описание**

Gloabal переменная для хранения SNMP ID Вар для OID получил в SNMP запросу.

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

**10.15.1.12 SNMPSendTrap Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

**void SNMPSendTrap**();

**Возвращает**

Ни один .

**описание**

Эта функция используется для отправки уведомления ловушки , чтобы предварительно настроенной IP-адрес , если уведомление ловушка включена . Есть отличается ловушка код уведомления . Текущая реализация посылает ловушку ошибки идентификации (4).

**Замечания**

Это функция обратного вызова вызывается приложения на некоторых предопределенных событий. Эта процедура реализована только для отправки Ошибка аутентификации Уведомление типа макрос с **PUSH_BUTTON OID** хранится в MPFS . Не Если ARP является не решен т.е. если

**SNMPIsNotifyReady** (см. стр. 334 ) () возвращает FALSE , это обычные раз в 5 секунд. Эта процедура должна быть изменена в соответствии с событием произошло и следует обновить corrsponding OID и тип уведомления на перехваты PDU .

**Предпосылки**

Если приложение определяется событие происходит отправить ловушку .

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

**10.15.1.13 SNMPNotify Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BOOL SNMPNotify(
SNMP_ID var,
SNMP_VAL val,
SNMP_INDEX index
);
```

**описание**

Эта функция создает SNMP ловушки PDU и отправляет его в ранее указанном RemoteHost . SNMPv1 перехваты PDU : | PDU типа |предприятие | агент -адр | родовое - ловушка | конкретных - ловушка | | временная метка | VarBind -лист |

V1 предприятие переходит непосредственно к SNMPv2TrapOID.0

Для ASCII STR ловушку VAL (аргумент) содержит адрес указателя переменной строки .

**Замечания**

Это будет работать, если бы не было UDP сокет открыть.

**Предпосылки**

SNMPIsNotifyReady (см. стр. 334 ) ( ) уже называют и вернулся ИСТИНА .

**Параметры**

**Параметры Описание**

**Var** - SNMP вар идентификатор, который должен быть использован в уведомлении

**Val** - Значение var. Только стоимость BYTE , WORD или DWORD может быть отправлено.

**index** - индекс var. Если это переменная является единым , индекс будет 0 , либо , если это **var** последовательность , индекс может принимать любое значение от 0 до 127

http://www.microchip.com/

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** - если уведомление SNMP успешно отправляются. Это не гарантирует, что RemoteHost получил его .

**ЛОЖЬ** Уведомление отправлено не удалось.

Это не удастся при следующих contions

1) Учитывая SNMP_BIB_FILE не существует в MPFS

2 ) Учитывая переменная не существует.

3) Ранее учитывая agentID не существует

4) Тип данных данной **var** неизвестно возможно, только если сам MPFS был поврежден .

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

**10.15.1.14 SNMPSetVar Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BOOL SNMPSetVar(
SNMP_ID var,
SNMP_INDEX index,
BYTE ref,
SNMP_VAL val
);
```

**описание**

Это функция обратного вызова вызывается модуль для request.user применения установки SNMP должны изменить эту функцию для нового переменные решения .

**Замечания**

Эта функция может вызываться несколько раз в зависимости от количества байтов в определенный набор запроса на данной переменной . только динамических переменных для чтения и записи должно быть обработано .

**Предпосылки**

**ProcessVariables** ( см. стр. 369 ) ( ) называется .

**Параметры**

**Параметры Описание**

**var**- Переменная идентификатор , значение которого должно быть установлено

**ref** - ссылка на переменную используется для передачи нескольких байт данных 0 , если первый байт установлен в противном случае ненулевое значение , чтобы указать соответствующий байт создаются .

**val** - до 4 значения данных байтов. Если тип **var** данные BYTE , значение переменной находится в Val-> байт Если Тип данных переменная является WORD, значение переменной находится в Валь -> слово Если тип **var** данные DWORD , значение переменной находится в **val** - > DWORD . Если тип **var** данные IP_ADDRESS , COUNTER32 или GAUGE32 , значение находится в Val- > DWORD Если тип **var** данные OCTET_STRING (см. стр. 352 ) , ASCII_STRING значение находится в Валь -> байт ;

передача нескольких байт будет выполнена передача остальные байты данных.

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА -** если все в порядке , чтобы установить больше байт (ы).

**ЛОЖЬ** - в противном случае.

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

**10.15.1.15 SNMPGetVar Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BOOL SNMPGetVar(
SNMP_ID var,
SNMP_INDEX index,
BYTE* ref,
SNMP_VAL* val
);
```

**описание**

Это функция обратного вызова вызывается модуль SNMP. Пользователь SNMP должен реализовать эту функцию в приложении пользователя и обеспечить соответствующие данные при вызове .

**Замечания**

Ни один .

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**var** - Переменная идентификатор , значение которого должно быть возвращено

**index** - индекс переменной, должны быть переданы

**ref** - ссылка на переменную используется для передачи данных многобайтную Это всегда SNMP_START_OF_VAR (см. стр. 337) , когда самый первый байт просил .

В противном случае , использовать это в качестве справки , чтобы отслеживать многобайтовых переводов.

**val** - Указатель до 4 буфера байт. Если тип **var** данных является байт , передача данных в **val** -> **byte**

Если тип **var** данные WORD, передавать данные в val -> word.

Если тип **var** данные DWORD ,передавать данные в val - > DWORD

Если тип **var** данные IP_ADDRESS , передачи данных в Вал - > V [] или вал - > DWORD

Если тип **var** данные COUNTER32 , TIME_TICKS или GAUGE32 , передавать данные в val - > DWORD

Если тип **var** данные ASCII_STRING или OCTET_STRING (см. стр. 352 ) передачи данных в Val- > байт с использованием многобайтных Передача механизм .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** - Если значение существует для данной переменной в данном индексе .

**FALSE** -В противном случае.

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

**10.15.1.16 SNMPIsNotifyReady Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BOOL SNMPIsNotifyReady(
IP_ADDR* remoteHost
);
```

http://www.microchip.com/

**описание**

Эта функция устраняет учитывая RemoteHost IP адрес в MAC-адрес , используя модуль ARP . Если RemoteHost не aviailable , эта функция никогда не вернется TRUE. Заявка должна реализовать логику ожидания обрабатывать " RemoteHost не Avialable " ситуация .

**Замечания**

Это будет работать, если бы не было UDP сокет открыть.

**Предпосылки**

**SNMPNotifyPrepare** (см. стр. 334 ) ( ) уже называют .

**Параметры**

**Параметры Описание**

**RemoteHost** Указатель на удаленном Host IP адрес

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если **RemoteHost** IP адрес будет решена и **SNMPNotify** (см. стр. 332 ) может назвать .

**ЛОЖЬ** Если RemoteHost IP адрес не будет решена .

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

**10.15.1.17 SNMPNotifyPrepare Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
void SNMPNotifyPrepare(
IP_ADDR* remoteHost,
char* community,
BYTE communityLen,
SNMP_ID agentIDVar,
BYTE notificationCode,
DWORD timestamp
);
```

**Возвращает**

Ни один

http://www.microchip.com/

**описание**

Эта функция подготавливает модуль SNMP для отправки уведомления SNMP ловушки для удаленного хоста. Он посылает запрос ARP для удаленного узла, чтобы узнать удаленного хоста MAC-адрес.

**Замечания**

Это, прежде ряда функций для завершения уведомления SNMP.

**Предпосылки**

**SNMPInit** (см. стр. 371 ) ( ) уже называют .

**Параметры**

**Параметры Описание**

**RemoteHost** -указатель на удаленный Host IP адрес

**community -**Строка сообщество использовать уведомить

**communityLen -** Сообщество длина строки

**agentIDVar-** System ID , чтобы использовать определить этот агент

**notificaitonCode** Код Уведомление использовать

**timestamp** -Отметка Уведомление отметка в 100- от второго .

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

**10.15.1.18 SNMPGetNextIndex Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BOOL SNMPGetNextIndex(
SNMP_ID var,
SNMP_INDEX* index
);
```

**описание**

Это функция обратного вызова вызывается модуль SNMP. Пользователь SNMP должен реализовать эту функцию в приложении пользователя и обеспечить соответствующие данные при вызове . Эта функция будет вызываться только для OID переменной последовательности типа .

http://www.microchip.com/

**Замечания**

Индекс только последовательность должно быть обработано в этой функции .

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**var** - Переменная идентификатор , значение которого должно быть возвращено

**index** - Следующая индекс переменной, должны быть переданы

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА -** Если следующий значение индекса существует для данной переменной в данном индексе и индекса параметр содержит следующий действительный индекс .

**FALSE** - В противном случае.

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

**10.15.1.19 SNMPValidateCommunity Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BYTE SNMPValidateCommunity(
BYTE * community
);
```

**Возвращает**

Эта процедура возвращает результат проверки сообщество **READ_COMMUNITY** или **WRITE_COMMUNITY** или **INVALID_COMMUNITY**

**описание**

Эта функция проверяет имя сообщества для доступа MIB к NMS. Название сообщества SNMP получил в запросе PDU проверяется имен чтения и записи сообщества. Агент дает доступ к MIB переменных только если сообщество матчи с заранее заданным значениям . Эта процедура также устанавливает gloabal флаг , чтобы отправить ловушки, если происходит сбой аутентификации .

http://www.microchip.com/

**Замечания**

Это функция обратного вызова вызывается модуль. Приложение пользователя должен реализовать эту функцию и проверить, что сообщество матчи с заранее заданного значения . Эта проверка происходит при каждом запросе NMS .

**Предпосылки**

**SNMPInit** (см. стр. 371 ) уже называют .

**Параметры**

**Параметры Описание**

**community** - Указатель на строку сообщества , полученных с NMS .

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

**10.15.1.20 SNMP_ID Тип**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef int SNMP_ID;
```

**описание**

Это переменная ID SNMP OID . Это идентификатор присваивается через MIB файл . Только динамические и AgentID переменные могут contian ID .

MIB2BIB утилита обеспечивает это правила, когда BIB был сгенерирован .

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

**10.15.1.21 SNMP_INDEX Тип**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef BYTE SNMP_INDEX;
```

**описание**

Это тип SNMP_INDEX .

http://www.microchip.com/

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

**10.15.1.22 SNMP_COMMUNITY_MAX_LEN Макро**

**файл**

TCPIP MRF24W.h

**C**

**#define SNMP_COMMUNITY_MAX_LEN** (8u)

**описание**

Это максимальная длина сообщества строки. Применение должны убедиться, что наблюдается эта длина . Модуль SNMP добавляет один байт дополнительное после **SNMP_COMMUNITY_MAX_LEN** для добавления пустой символ '0'.

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

**10.15.1.23 OID_MAX_LEN Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

**#define OID_MAX_LEN** (SNMP_MAX_OID_LEN_MEM_USE+1)

**описание**

Измените это , чтобы соответствовать вашему OID длину строки .

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

**10.15.1.24 SNMP_START_OF_VAR Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

**#define SNMP_START_OF_VAR** (0)

**описание**

http://www.microchip.com/

Это макро SNMP_START_OF_VAR .

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

**10.15.1.25 SNMP_END_OF_VAR Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_END_OF_VAR (0xff)
```

**описание**

Это макро SNMP_END_OF_VAR .

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

**10.15.1.26 SNMP_INDEX_INVALID Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_INDEX_INVALID (0xff)
```

**описание**

Это макро SNMP_INDEX_INVALID .

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

**10.15.1.27 TRAP_TABLE_SIZE Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define TRAP_TABLE_SIZE (2)
```

**описание**

http://www.microchip.com/

Это макро TRAP_TABLE_SIZE .

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

**10.15.1.28 TRAP_COMMUNITY_MAX_LEN Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define TRAP_COMMUNITY_MAX_LEN (SNMP_TRAP_COMMUNITY_MAX_LEN_MEM_USE+1)
```

**описание**

Это макро TRAP_COMMUNITY_MAX_LEN .

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

**10.15.1.29 NOTIFY_COMMUNITY_LEN Макро**

**файл**

TCPIP MRF24W.h

**C**

```
#define NOTIFY_COMMUNITY_LEN (SNMP_COMMUNITY_MAX_LEN)
```

**описание**

Это макро NOTIFY_COMMUNITY_LEN .

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

## 10.15.2 SNMP Внутренний Пользователи

**Перечисления**

**Имя Описание**

DATA_TYPE (см. стр.345)

SNMP_ERR_STATUS (см. стр. 356 )

http://www.microchip.com/

**Функции**

**Имя Описание**

**_SNMPDuplexInit** (См. стр. 342) Подготовка к полнодуплексном передачи .

**_SNMPGet** ( См. стр. 342) Read байт из SNMP UDP буфера сокета RX.

**_SNMPPut** (См. стр. 343 ) Скопируйте байт TX буфера.

**FindOIDsInRequest** (см. стр.346) Находит количество varbinds в списке VarBind полученной в PDU.

**GetDataTypeInfo** (см. стр. 350 ) Получить ASN тип данных информацию.

**IsASNNull** (см. стр. 351 ) Проверяет тип значения , как ASN_NULL (см. стр. 344 ) .

**SetErrorStatus** (см. стр. 355 ) указан SNMP статус ошибки в PDU ответа.

**IsValidLength** (см. стр. 358 ) Получает длину пакета и фактическую длину PDU.

**GetNextLeaf** (см. стр. 362 ) Ищет следующего листа в MIP дерева.

**GetOIDStringByAddr** (см. стр.363) Получить OID строки из MPFS используя адрес узла .

**GetOIDStringByID** (см. стр.363) Получить уведомление о завершении переменная OID строки из MPFS помощью уаг идентификатор .

**IsValidCommunity** (см. стр.364) Проверяет для сообщества строки типа данных и имени макс сообщества и длина , этот агент может обрабатывать .

**IsValidInt** (см. стр. 364 ) Проверяет переменную типа данных как INT и извлекает его значение.

**IsValidOID** (см. стр. 365 ) Заполняет OID тип , длину и подъязычная строку из принятого PDU.

**IsValidPDU** (см. стр. 366 ) Проверяет для типа запроса SNMP .

**IsValidStructure** (см. стр. 366 ) Decode переменную структуру длины.

**OIDLookup** (см. стр. 367 ) для поиска и проверки , является ли просьба OID находится в базе данных MIB .

**ProcessGetSetHeader** (см.страница 367) Проверяетполучил UDP-пакетов Получить / заголовок запроса Set.

**ProcessHeader** (см. стр. 368 ) Проверяет получил UDP-пакетов заголовок SNMP .

**ProcessSetVar** (см. стр. 369 ) Процессы установки SNMP запроса PDU .

**ProcessVariables** ( см. стр. 369 ) Эта процедура обрабатывает запрос SNMP и параллельно создает ответ PDU.

**ReadMIBRecord** (см. стр. 370 ) Получить OID строки из MPFS используя адрес узла .

**SNMPCheckIfPvtMibObjRequested**(См. стр. 370 ) Чтобы найти ли просил OID только для частного доступа .

**макрос**

**Имя Описание**

**_SNMPGetTxOffset** (См.страница 343) Это макро _SNMPGetTxOffset .

**_SNMPSetTxOffset** (См.страница 343)

**AGENT_NOTIFY_PORT** ( см. стр. 343) Это макро AGENT_NOTIFY_PORT .

**ASN_INT** (см. стр. 344 ) Этот макрос ASN_INT .

**ASN_NUL**L (см. стр. 344 ) Этот макрос ASN_NULL .

**ASN_OID** (см. стр. 344 ) Этот макрос ASN_OID .

**DATA_TYPE_TABLE_SIZE**(См. стр. 346 )

**GET_BULK_REQUEST** ( см. стр. 347) Это макро GET_BULK_REQUEST .

**GET_NEXT_REQUEST** (см. стр. 347) Это макро GET_NEXT_REQUEST .

**GET_REQUEST** (см.страница 347)

**GET_RESPONSE** (см.страница 347) Это макро GET_RESPONSE .

**IS_AGENT_PDU** (см.страница 348) Это макро IS_AGENT_PDU .

**IS_ASN_INT** (см. стр.348) Это макро IS_ASN_INT .

**IS_ASN_NULL** (см. стр.349) Это макро IS_ASN_NULL .

**IS_GET_NEXT_REQUEST**(См. стр. 349 ) Это макро IS_GET_NEXT_REQUEST .

**IS_GET_REQUEST** (см.страница 349 ) Это макро IS_GET_REQUEST .

**IS_GET_RESPONSE** (см.страница 349 ) Это макро IS_GET_RESPONSE .

**IS_OCTET_STRING** (см.страница 350 ) Это макро IS_OCTET_STRING .

**IS_OID** (см. стр. 350 ) Этот макрос IS_OID .

**IS_SET_REQUEST** (см.страница 351) Это макро IS_SET_REQUEST .

**IS_STRUCTURE** (см.страница 351) IS_TRAP (см. стр. 351 ) Этот макрос IS_TRAP .

**OCTET_STRING** (см. страница 352) Это макро OCTET_STRING .

**SET_REQUEST** (см. стр.354) Это макро SET_REQUEST .

**SNMP_AGENT_PORT** (см. стр. 355)

**SNMP_BIB_FILE_NAME** ( см. стр. 355) Это файл , который содержит SNMP нагрудник файл . Имя файла должно содержать все верхнее буква и должен соответствовать тому, что был включен в MPFS2 изображения.

**SNMP_COUNTER32** (см.страница 356 ) Это макро SNMP_COUNTER32 .

**SNMP_GAUGE32** (см.страница 357 ) Это макро SNMP_GAUGE32 .

**SNMP_IP_ADDR** (см. страница 357 ) Это макро SNMP_IP_ADDR .

**SNMP_NMS_PORT** (см.страница 357 ) Это макро SNMP_NMS_PORT .

**SNMP_NSAP_ADDR** (см. страница 358 ) Это макро SNMP_NSAP_ADDR .

**SNMP_OPAQUE** (см.страница 359) Это макро SNMP_OPAQUE .

**SNMP_TIME_TICKS** (см.страница 359) Это макро SNMP_TIME_TICKS .

**SNMP_V1** (см. стр. 360 )

**SNMP_V2C** (см. стр. 360 ) Этот макрос SNMP_V2C .

**STRUCTURE** (см. стр.361)

**TRAP** (см. стр. 362 ) Этот макрос TRAP.

**модуль**

**SNMP** (см. стр. 318 )

**структуры**

**Имя Описание**

**DATA_TYPE_INFO** (см.страница 345 )

**OID_INFO** (см. стр. 353 )

**PDU_INFO** (см. стр. 353 )

**reqVarErrStatus** (см. стр.354)

**SNMP_NOTIFY_INFO** (см. стр. 358)

**Unions**

**Имя Описание**

**INDEX_INFO** (см. стр.348)

http://www.microchip.com/

**MIB_INFO** (см. стр. 352 )

**SNMP_STATUS** (см. стр.359)

**Переменные**

**Имя Описание**

**appendZeroToOID** (см.страница 344) глобальный флаг , чтобы изменить OID путем добавления нуля

**dataTypeTable** (см. стр.346)ASN формат тип данных для SNMP v1 и v2c

**hMPFS** (см. стр. 348 ) ГПСЛ обработчик файлов

**SNMPAgentSocket** (см.страница 360) Snmp сокета UDP

**SNMPNotifyInfo** (см. стр.360) уведомить информацию для ловушку

**snmpReqVarErrStatus** (см. стр. 361) **vars** из списка REQ статус обработки ошибаться

**SNMPRxOffset** (см. стр.361) Snmp UDP буфер RX смещение

**SNMPStatus** (см. стр.361) Статус доступа MIB файл

**SNMPTxOffset** (см. стр. 361) Snmp UDP буфер TX смещение

**trapInfo** (см. стр. 362) Инициализация ловушки стол без каких-либо записей.

**описание**

Следующие функции и переменные обозначены как внутренний модулю SNMP.

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

**10.15.2.1 _SNMPDuplexInit Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
void _SNMPDuplexInit(
UDP_SOCKET socket
);
```

**Возвращает**

Ни один .

**описание**

По мере обработки SNMP переменные , мы подготовим ответ на лету , создавая полный дуплекс передачи . Текущий уровень УДС делает не поддерживает дуплексную передачу , так SNMP должен управлять собственной дуплексный режим . Подготовка к полнодуплексном передачи .

Установите Tx и Rx Смещение начала буфера.

**Замечания**

Эта процедура должна быть вызвана для каждого нового SNMP пакет, принятый .

**Предпосылки**

SNMPTask (см. стр. 372 ) () вызывается .

**Параметры**

**Параметры Описание**

**socket** ( см. стр. 180 ) Активная сокета UDP , для которых ПРД и ПРМ смещения должен быть установлен.

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

**10.15.2.2 _SNMPGet Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

BYTE **_SNMPGet**();

**Возвращает**

Ни один .

**описание**

Эта функция читает один байт из активного сокета UDP , приемного буфера , в то время увеличивая смещения буфера откуда следующий байт будет прочитан .

**Замечания**

Ни один .

**Предпосылки**

**SNMPTask** (см. стр. 372 ) () вызывается . Активное сокета UDP , доступных для чтения от .

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

#### 10.15.2.3 _SNMPGetTxOffset Макро

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define _SNMPGetTxOffset SNMPTxOffset
```

**описание**

Это макро _SNMPGetTxOffset .

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

#### 10.15.2.4 _SNMPPut Функция

**файл**

SNMP.h (см. стр. 451 )

**C**

```
void _SNMPPut(
BYTE v
);
```

**Возвращает**

Ни один .

**описание**

Эта функция записывает один байт на текущий активный UDP сокет , буфер передачи , в то время увеличивая буфер смещение для следующая операция записи .

**Замечания**

Ни один .

**Предпосылки**

**SNMPTask** (см. стр. 372 ) () вызывается . Активное сокета UDP работает только в в Техас с .

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

#### 10.15.2.5 _SNMPSetTxOffset Макро

http://www.microchip.com/

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define _SNMPSetTxOffset(o) (SNMPTxOffset = o)
```

**раздел**

SNMP Tx PDU смещение настройки

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

### 10.15.2.6 AGENT_NOTIFY_PORT Макро

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define AGENT_NOTIFY_PORT (0xfffe)
```

**описание**

Это макро AGENT_NOTIFY_PORT .

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

### 10.15.2.7 appendZeroToOID Переменная

**файл**

SNMP.c (см. стр. 449 )

**C**

```
BYTE appendZeroToOID = FALSE;
```

**описание**

глобальный флаг , чтобы изменить OID путем добавления нуля

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

### 10.15.2.8 ASN_INT Макро

http://www.microchip.com/

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define ASN_INT (0x02u)
```

**описание**

Это макро ASN_INT .

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

**10.15.2.9 ASN_NULL Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define ASN_NULL (0x05u)
```

**описание**

Это макро ASN_NULL .

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

**10.15.2.10 ASN_OID Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define ASN_OID (0x06u)
```

**описание**

Это макро ASN_OID .

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

**10.15.2.11 DATA_TYPE Перечисление**

**файл**

http://www.microchip.com/

SNMP.h (см. стр. 451 )

**C**

```
typedef enum {
INT8_VAL = 0x00,
INT16_VAL = 0x01,
INT32_VAL = 0x02,
BYTE_ARRAY = 0x03,
ASCII_STRING = 0x04,
IP_ADDRESS = 0x05,
COUNTER32 = 0x06,
TIME_TICKS_VAL = 0x07,
GAUGE32 = 0x08,
OID_VAL = 0x09,
DATA_TYPE_UNKNOWN
} DATA_TYPE;
```

**Пользователи**

**Пользователи Описание**

**INT8_VAL = 0x00** 8-бит целое значение

**INT16_VAL = 0x01** 16 бит целое значение

**INT32_VAL = 0x02** 32-битное целое значение

**BYTE_ARRAY = 0x03** массив байтов

**ASCII_STRING = Тип 0x04** строка ASCII

**IP_ADDRESS = 0x05** переменная IP адрес

**COUNTER32 = 0x06** 32 бит переменная-счетчик

**TIME_TICKS_VAL = 0x07** Таймер vakue переменная-счетчик

**GAUGE32 = 0x08** 32 бит переменная Guage

**OID_VAL = 0x09** Идентификатор объекта значение var

**DATA_TYPE_UNKNOWN** Неизвестный тип данных

**раздел**

**Структуры и перечисления данных**

********** **********

SNMP конкретные tyes данных

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

http://www.microchip.com/

**10.15.2.12 DATA_TYPE_INFO Структура**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef struct {
BYTE asnType;
BYTE asnLen;
} DATA_TYPE_INFO;
```

**Пользователи**

**Пользователи Описание**

BYTE **asnType** ; Тип данных ACH

BYTE **asnLen** ; Длина данных ACH

**раздел**

ASN тип данных информации

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

**10.15.2.13 DATA_TYPE_TABLE_SIZE Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define DATA_TYPE_TABLE_SIZE (sizeof(dataTypeTable)/sizeof(dataTypeTable[0]))
```

**раздел**

Макросы и определения

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

**10.15.2.14 dataTypeTable Переменная**

**файл**

SNMP.c (см. стр. 449 )

**C**

ROM DATA_TYPE_INFO **dataTypeTable**[] = { { ASN_INT, 1 }, { ASN_INT, 2 }, { ASN_INT, 4 }, { OCTET_STRING, 0xff }, { OCTET_STRING, 0xff }, { SNMP_IP_ADDR, 4 }, { SNMP_COUNTER32, 4 }, { SNMP_TIME_TICKS, 4 }, { SNMP_GAUGE32, 4 }, { ASN_OID, 0xff } };

**описание**

ASN формат тип данных для SNMP v1 и v2c

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

**10.15.2.15 FindOIDsInRequest Функция**

**файл**

SNMP.c (см. стр. 449 )

**C**

```
static BYTE FindOIDsInRequest(
WORD pdulen
);
```

**описание**

Эта процедура используется для поиска количество OIDs запрашиваемых в принятом SNMP PDU.

**Замечания**

Ни один .

**Предпосылки**

ProcessVariables ( см. стр. 369 ) ( ) называется .

**Параметры**

**Параметры Описание**

**pdulen-** Длина запросу SNMP PDU получил .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**varCount** -Количество OIDs найдено в запросе PDU.

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

**10.15.2.16 GET_BULK_REQUEST Макро**

http://www.microchip.com/

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define GET_BULK_REQUEST (0xa5)
```

**описание**

Это макро GET_BULK_REQUEST .

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

**10.15.2.17 GET_NEXT_REQUEST Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define GET_NEXT_REQUEST (0xa1)
```

**описание**

Это макро GET_NEXT_REQUEST .

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

**10.15.2.18 GET_REQUEST Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define GET_REQUEST (0xa0)
```

**раздел**

SNMP v1 и v2c типы PDU

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

**10.15.2.19 GET_RESPONSE Макро**

http://www.microchip.com/

**файл**

SNMP.h (см. стр. 451 )

C

```
#define GET_RESPONSE (0xa2)
```

**описание**

Это макро GET_RESPONSE .

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

**10.15.2.20 hMPFS Переменная**

**файл**

SNMP.c (см. стр. 449 )

**C**

```
MPFS_HANDLE hMPFS;
```

**описание**

MPFS обработчик файлов

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

**10.15.2.21 INDEX_INFO союз**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef union {
struct {
unsigned int bIsOID : 1;
} Flags;
BYTE Val;
} INDEX_INFO;
```

**Пользователи**

**Пользователи Описание**

http://www.microchip.com/

**unsigned Int bIsOID : 1** ; значение OID / индекс внутр флаг

**BYTE Val** ; значение флага OID / индекс байта

**раздел**

Информация индекс SNMP OID

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

**10.15.2.22 IS_AGENT_PDU Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define IS_AGENT_PDU(a) (a==GET_REQUEST || \
a==GET_NEXT_REQUEST || \
a==SET_REQUEST || \
a==SNMP_V2C_GET_BULK)
```

**описание**

Это макро IS_AGENT_PDU .

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

**10.15.2.23 IS_ASN_INT Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define IS_ASN_INT(a) (a==ASN_INT)
```

**описание**

Это макро IS_ASN_INT .

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

**10.15.2.24 IS_ASN_NULL Макро**

**файл**

http://www.microchip.com/

SNMP.h (см. стр. 451 )

**C**

```
#define IS_ASN_NULL(a) (a==ASN_NULL)
```

**описание**

Это макро IS_ASN_NULL .

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

**10.15.2.25 IS_GET_NEXT_REQUEST Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define IS_GET_NEXT_REQUEST(a) (a==GET_NEXT_REQUEST)
```

**описание**

Это макро IS_GET_NEXT_REQUEST .

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

**10.15.2.26 IS_GET_REQUEST Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define IS_GET_REQUEST(a) (a==GET_REQUEST)
```

**описание**

Это макро IS_GET_REQUEST .

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

**10.15.2.27 IS_GET_RESPONSE Макро**

**файл**

http://www.microchip.com/

SNMP.h (см. стр. 451 )

**C**

```
#define IS_GET_RESPONSE(a) (a==GET_RESPONSE)
```

**описание**

Это макро IS_GET_RESPONSE .

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

**10.15.2.28 IS_OCTET_STRING Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define IS_OCTET_STRING(a) (a==OCTET_STRING)
```

**описание**

Это макро IS_OCTET_STRING .

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

**10.15.2.29 IS_OID Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define IS_OID(a) (a==ASN_OID)
```

**описание**

Это макро IS_OID .

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

**10.15.2.30 GetDataTypeInfo Функция**

**файл**

http://www.microchip.com/

SNMP.h (см. стр. 451 )

**C**

```
BOOL GetDataTypeInfo(
DATA_TYPE dataType,
DATA_TYPE_INFO * info
);
```

**Возвращает**

**ИСТИНА** - успех

**ЛОЖЬ** - отказ

**описание**

Эта процедура проверяет SNMP GET dtat тип и собирает значение типа данных ACH к информации.

**Замечания**

Ни один .

**Предпосылки**

**ProcessHeader** (см. стр. 368 ) () вызывается .

**Параметры**

**Параметры Описание**

Тип данных SNMP Тип PDU данных информации .

Информация соответствия типов данных информации относительно **dataTypeTable** (см. стр. 346 )

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

**10.15.2.31 IS_SET_REQUEST Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define IS_SET_REQUEST(a) (a==SET_REQUEST)
```

**описание**

Это макро IS_SET_REQUEST .

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

**10.15.2.32 IS_STRUCTURE Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define IS_STRUCTURE(a) (a==STRUCTURE)
```

**раздел**

Конкретных проверка данных SNMP

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

**10.15.2.33 IS_TRAP Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define IS_TRAP(a) (a==TRAP)
```

**описание**

Это макро IS_TRAP .

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

**10.15.2.34 IsASNNull Функция**

**файл**

SNMP.c (см. стр. 449 )

**C**

```
static BOOL IsASNNull();
```

**описание**

Для **Get , Get_next** , **Get_Bulk SNMP** reuest ,var связывают тип значения данных должны быть **ASN_NULL** (см. стр. 344 ) и Значение поля должно быть NULL и . Эта процедура проверяет поля типа

данных и значения в принятых запросов . SET Запрос , тип данных значение не может быть **ASN_NULL** (см. стр. 344) , в противном случае запрос SNMP не обрабатывается.

**Замечания**

Ни один .

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**ИСТИНА-** Если тип значения является ASN_NULL (см. стр. 344 ) и значение NULL .

**ЛОЖЬ** -Если тип данных и значение кроме ASN_NULL (см. стр. 344 ) и NULL соответственно

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

**10.15.2.35 MIB_INFO союз**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef union {
struct {
unsigned int bIsDistantSibling : 1;
unsigned int bIsConstant : 1;
unsigned int bIsSequence : 1;
unsigned int bIsSibling : 1;
unsigned int bIsParent : 1;
unsigned int bIsEditable : 1;
unsigned int bIsAgentID : 1;
unsigned int bIsIDPresent : 1;
} Flags;
BYTE Val;
} MIB_INFO;
```

**Пользователи**

**Пользователи Описание**

**неподписанных Int bIsDistantSibling : 1** ; Объект есть дальний узел-брат

**неподписанных Int bIsConstant : 1** ; Объект является постоянным

**неподписанных Int bIsSequence : 1** ; Объект является последовательность

http://www.microchip.com/

**неподписанных Int bIsSibling : 1** ; Родного брата флаг узел

**неподписанных Int bIsParent : 1** ; Узел является материнской флаг

**неподписанных Int bIsEditable : 1** ; Узел доступен для редактирования флаг

**неподписанных Int bIsAgentID : 1** ; Узел есть риэлтор флаг документа

**неподписанных Int bIsIDPresent : 1** ; Id присутствует флаг

**BYTE Val** ; Информация MIB Obj как значение байта

**раздел**

Информация объект SNMP

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

**10.15.2.36 OCTET_STRING Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define OCTET_STRING (0x04u)
```

**описание**

Это макро OCTET_STRING .

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

**10.15.2.37 OID_INFO Структура**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef struct {
DWORD hNode;
BYTE oid;
MIB_INFO nodeInfo;
DATA_TYPE dataType;
SNMP_ID id;
WORD_VAL dataLen;
DWORD hData;
DWORD hSibling;
```

DWORD **hChild**;
BYTE **index**;
BYTE **indexLen**;
} OID_INFO;

**Пользователи**

**Пользователи Описание**

**DWORD hNode** ; Узел расположение в MIB

**BYTE oid** ; ID объекта

**MIB_INFO nodeInfo** ; Узел информации

**DATA_TYPE тип данных** ; тип данных

**SNMP_ID ID** ; Snmp Id

**WORD_VAL DataLen** ; длина данных

**DWORD hData** ; данные

**DWORD hSibling** ; Родного брата информация

**DWORD hChild** ; информация по уходу за детьми

**BYTE index** ; Индекс объекта

**BYTE indexLen** ; длина Индекс

**раздел**

SNMP MIB информация переменная объекта

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

**10.15.2.38 PDU_INFO Структура**

**файл**

SNMP.h (см. стр. 451 )

**C**

**typedef struct** {
DWORD_VAL **requestID**;
BYTE **nonRepeators**;
BYTE **maxRepetitions**;
BYTE **pduType**;
BYTE **errorStatus**;

BYTE **erroIndex**;
BYTE **snmpVersion**;
WORD **pduLength**;
} PDU_INFO;

**Пользователи**

**Пользователи Описание**

**DWORD_VAL RequestID** ; Snmp запрос ID

**BYTE nonRepeators** ; не являющиеся повторители в просьбе

**BYTE maxRepetitions** ; макс повторители в просьбе

**BYTE pduType** ; Snmp тип PDU

**BYTE ErrorStatus** ; Статус ошибки Pdu

**BYTE erroIndex** ; Индекс ошибка Pdu

**BYTE snmpVersion** ; версия SNMP

**WORD pduLength** ; длина Pdu

**раздел**

SNMP информации PDU базы данных

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

**10.15.2.39 reqVarErrStatus Структура**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef struct {
WORD noSuchObjectErr;
WORD noSuchNameErr;
WORD noSuchInstanceErr;
WORD endOfMibViewErr;
} reqVarErrStatus;
```

**Пользователи**

**Пользователи Описание**

**WORD noSuchObjectErr** ; Не список Var нет таких OBJ ошибки флаги

http://www.microchip.com/

**WORD noSuchNameErr** ; Список Var нет такой ошибки имя

**WORD noSuchInstanceErr** ; Список Var нет такой ошибки экземпляр

**WORD endOfMibViewErr** ; Список Var конец просмотра MIB ошибки

**раздел**

SNMP reuested информацию переменную состояния ошибки список .

Максимальное значение переменной в запросе поддерживается 15

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

**10.15.2.40 SET_REQUEST Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SET_REQUEST (0xa3)
```

**описание**

Это макро SET_REQUEST .

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

**10.15.2.41 SetErrorStatus Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
void SetErrorStatus(
WORD errorStatusOffset,
WORD errorIndexOffset,
SNMP_ERR_STATUS errorStatus,
BYTE errorIndex
);
```

**Возвращает**

Ни один .

**описание**

http://www.microchip.com/

Эта процедура обрабатывает принятый SNMP Получить запроса PDU для связывания переменной в запросе , а также создает ответ PDU.

**Замечания**

Ни один .

**Предпосылки**

**ProcessVariables** ( см. стр. 369 ) ( ) называется .

**Параметры**

**Параметры Описание**

**errorStatusOffset** Смещение обновить статус ошибки в ответ Tx PDU

**errorIndexOffset** Смещение обновить индекс ошибке

**ErrorStatus** Snmp технологическая ошибка в обновлении в ответ .

**ErrorIndex** Индекс запросу VarBind в списке переменная привязки , для которых статус ошибки должен быть обновляется.

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

**10.15.2.42 SNMP_AGENT_PORT Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_AGENT_PORT (161)
```

**раздел**

Порты SNMP UDP

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

**10.15.2.43 SNMP_BIB_FILE_NAME Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_BIB_FILE_NAME "snmp.bib"
```

http://www.microchip.com/

**описание**

Это файл , который содержит SNMP нагрудник файл . Имя файла должно содержать все верхнего регистра и должны соответствовать тем, что было включены в MPFS2 изображения.

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

**10.15.2.44 SNMP_COUNTER32 Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_COUNTER32 (0x41)
```

**описание**

Это макро SNMP_COUNTER32 .

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

**10.15.2.45 SNMP_ERR_STATUS Перечисление**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef enum {
SNMP_NO_ERR = 0,
SNMP_TOO_BIG,
SNMP_NO_SUCH_NAME,
SNMP_BAD_VALUE,
SNMP_READ_ONLY,
SNMP_GEN_ERR,
SNMP_NO_ACCESS,
SNMP_WRONG_TYPE,
SNMP_WRONG_LENGTH,
SNMP_WRONG_ENCODING,
SNMP_WRONG_VALUE,
SNMP_NO_CREATION,
SNMP_INCONSISTENT_VAL,
SNMP_RESOURCE_UNAVAILABE,
SNMP_COMMIT_FAILED,
SNMP_UNDO_FAILED,
```

http://www.microchip.com/

**SNMP_AUTH_ERROR**,
**SNMP_NOT_WRITABLE**,
**SNMP_INCONSISTENT_NAME**,
**SNMP_NO_SUCH_OBJ** = 128,
**SNMP_NO_SUCH_INSTANCE** = 129,
**SNMP_END_OF_MIB_VIEW** = 130
} SNMP_ERR_STATUS;

**Пользователи**

**Пользователи Описание**

**SNMP_NO_ERR = 0** Snmp нет ошибки

**SNMP_TOO_BIG** Значение слишком большая ошибка

**SNMP_NO_SUCH_NAME** Нет такого имени в MIB ошибки

**SNMP_BAD_VALUE** Не назначается значение для yar ошибки

**SNMP_READ_ONLY** только для чтения переменной , напишите не допускается Err

**SNMP_GEN_ERR** Snmp поколения ошибке

**SNMP_NO_ACCESS** Доступ для изменения или читать не предоставляется ошибаться

**SNMP_WRONG_TYPE** Тип переменной данные неправильно ошибке

**SNMP_WRONG_LENGTH** Неправильный Ошибка длины данных

**SNMP_WRONG_ENCODING** Неправильный ошибки кодирования

**SNMP_WRONG_VALUE** Неверное значение для типа var

**SNMP_NO_CREATION** Нет creationg ошибку

**SNMP_INCONSISTENT_VAL** Непоследовательность ошибки значение

**SNMP_RESOURCE_UNAVAILABE** Resource unavailbe ошибки

**SNMP_COMMIT_FAILED** Модификация не удалась ошибка

**SNMP_UNDO_FAILED** Модификация отменить неудачная

**SNMP_AUTH_ERROR** Авторизация удалось ошибку

**SNMP_NOT_WRITABLE** Переменная только для чтения

**SNMP_INCONSISTENT_NAME** Непоследовательность Имя

**SNMP_NO_SUCH_OBJ = 128** Нет такой ошибки объект

**SNMP_NO_SUCH_INSTANCE = 129** Нет такой ошибки экземпляр

**SNMP_END_OF_MIB_VIEW = 130** Достигнута конце ошибки MIB

http://www.microchip.com/

**раздел**

**SNMP** конкретные ошибки

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

**10.15.2.46 SNMP_GAUGE32 Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_GAUGE32 (0x42)
```

**описание**

Это макро SNMP_GAUGE32 .

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

**10.15.2.47 SNMP_IP_ADDR Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_IP_ADDR (0x40)
```

**описание**

Это макро SNMP_IP_ADDR .

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

**10.15.2.48 SNMP_NMS_PORT Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_NMS_PORT (162)
```

http://www.microchip.com/

**описание**

Это макро SNMP_NMS_PORT .

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

**10.15.2.49 SNMP_NOTIFY_INFO Структура**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
typedef struct {
char community[NOTIFY_COMMUNITY_LEN];
BYTE communityLen;
SNMP_ID agentIDVar;
BYTE notificationCode;
UDP_SOCKET socket;
DWORD_VAL timestamp;
SNMP_ID trapIDVar;
} SNMP_NOTIFY_INFO;
```

**Пользователи**

**Пользователи Описание**

**char community** [ NOTIFY_COMMUNITY_LEN ] ; Название **community**массив

**BYTE communityLen** ; Сообщество длина имени

**SNMP_ID agentIDVar** ; Агент ID для идентификации ловушки

**BYTE notificationCode** ; Ловушка код уведомления

**UDP_SOCKET разъем**; Udp Номер сокета

**DWORD_VAL метка** ; Отметка времени ловушку

**SNMP_ID trapIDVar** ; SNMPv2 конкретных ловушка

**раздел**

SNMP ловушка информация о заявлении для агента

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

**10.15.2.50 SNMP_NSAP_ADDR Макро**

http://www.microchip.com/

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_NSAP_ADDR (0x45)
```

**описание**

Это макро SNMP_NSAP_ADDR .

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

**10.15.2.51 IsValidLength Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BYTE IsValidLength(
WORD * len
);
```

**описание**

Проверяет текущий пакет и возвращается общая длина значения , а также фактическая длина bytes.We не поддерживают любой длины количество байтов более 2 т.е. общая стоимость длина не должна быть больше, чем 16-битная .

**Замечания**

Ни один .

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**len** -Указатель на память , где фактическая длина хранится.

**Возвращаемые значения**

**Возвращаемые значения Описание**

**lengthBytes** -Всего байт длина являются 0x80 сама плюс TempData .

http://www.microchip.com/

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

**10.15.2.52 SNMP_OPAQUE Макро**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_OPAQUE (0x44)
```

**описание**

Это макро SNMP_OPAQUE .

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

**10.15.2.53 SNMP_STATUS союз**

**файл**

SNMP.h (см. стр. 451 )

C

```
typedef union {
struct {
unsigned int bIsFileOpen : 1;
} Flags;
BYTE Val;
} SNMP_STATUS;
```

**Пользователи**

**Пользователи Описание**

**unsiged Int bIsFileOpen : 1** ;  Внутр доступ MIB файл флаг

**BYTE Val ;**  Флаг байт доступ MIB файл

**раздел**

Доступ к информации SNMP MIB конкретных файл

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

**10.15.2.54 SNMP_TIME_TICKS Макро**

http://www.microchip.com/

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_TIME_TICKS (0x43)
```

**описание**

Это макро SNMP_TIME_TICKS .

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

### 10.15.2.55 SNMP_V1 Макро

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_V1 (0)
```

**раздел**

SNMP типы агент версия

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

### 10.15.2.56 SNMP_V2C Макро

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define SNMP_V2C (1)
```

**описание**

Это макро SNMP_V2C .

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

### 10.15.2.57 SNMPAgentSocket Переменная

http://www.microchip.com/

**файл**

SNMP.c (см. стр. 449 )

**C**

UDP_SOCKET **SNMPAgentSocket** = INVALID_UDP_SOCKET;

**описание**

Snmp сокета UDP

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

**10.15.2.58 SNMPNotifyInfo Переменная**

**файл**

SNMP.c (см. стр. 449 )

**C**

SNMP_NOTIFY_INFO **SNMPNotifyInfo**;

**описание**

уведомить информацию для ловушку

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

**10.15.2.59 snmpReqVarErrStatus Переменная**

**файл**

SNMP.c (см. стр. 449 )

**C**

reqVarErrStatus **snmpReqVarErrStatus**;

**описание**

**vars** из списка REQ статус обработки ошибаться

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

**10.15.2.60 SNMPRxOffset Переменная**

http://www.microchip.com/

**файл**

SNMP.c (см. стр. 449 )

C

WORD **SNMPRxOffset** = 0;

**описание**

Snmp UDP буфер RX смещение

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

**10.15.2.61 SNMPStatus Переменная**

**файл**

SNMP.c (см. стр. 449 )

**C**

SNMP_STATUS **SNMPStatus**;

**описание**

Статус доступа MIB файл

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

**10.15.2.62 SNMPTxOffset Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

**C**

WORD **SNMPTxOffset**;

**описание**

Snmp UDP буфер TX смещение

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

**10.15.2.63 СТРУКТУРА Макро**

http://www.microchip.com/

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define STRUCTURE (0x30u)
```

**раздел**

SNMP конкретные переменные

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

### 10.15.2.64 TRAP Макро

**файл**

SNMP.h (см. стр. 451 )

**C**

```
#define TRAP (0xa4)
```

**описание**

Этот макрос TRAP.

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

### 10.15.2.65 trapInfo Переменная

**файл**

CustomSNMPApp.c

**C**

```
TRAP_INFO trapInfo = { TRAP_TABLE_SIZE };
```

**описание**

Инициализация ловушки стол без каких-либо записей.

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

### 10.15.2.66 GetNextLeaf Функция

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BOOL GetNextLeaf(
OID_INFO* rec
);
```

**описание**

Эта процедура ищет следующую листовым узлом от текущего узла . На вход этой функции является узел , откуда в следующем лист узел должен быть расположен . На следующий листовой узел будет silbing еще далекой родной брат или листовой узел из соседней ветки , если таковые присутствуют .

входной параметр **var** указатель будет обновляться с вновь обретенной конечного узла OID информации.

**Замечания**

Ни один .

**Предпосылки**

**ProcessGetBulkVar** (см. стр. 378 ) ( ) еще **ProcessGetNextVar** (см. стр. 379 ) () вызывается .

**Параметры**

**Параметры Описание**

**REC** Указатель на SNMP MIB переменной информации об объекте

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если рядом листовой узел находится .

**ЛОЖЬ** Там нет рядом листовой узел .

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

**10.15.2.67 GetOIDStringByAddr Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BOOL GetOIDStringByAddr(
OID_INFO* rec,
```

BYTE* **oidString**,
BYTE* **len**
);

**описание**

Эта процедура вызывается , когда OID строка требуется для поиска от MPFS используя адрес узла .

**Замечания**

Ни один .

**Предпосылки**

Ни один .

**Параметры**

**Параметры Описание**

**REC** Указатель на SNMP MIB переменной информации об объекте

**oidString** Указатель для хранения строки из OID искали

**len** Длина Oid

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если oid строка найдена .

**Flase** противном случае .

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

**10.15.2.68 GetOIDStringByID Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

BOOL **GetOIDStringByID**(
SNMP_ID **id**,
OID_INFO* **info**,
BYTE* **oidString**,
BYTE* **len**
);

**описание**

Эта процедура вызывается , когда OID строка требуется для поиска от MPFS использованием идентификатора агента . Строка сохраняется с агентом .

Перехваты PDU это отправить с этим OID соответствующий SNMP_ID (см. стр. 336 ) , используемый приложением агента для отправки PDU.

**Замечания**

Эта функция используется только тогда, когда TRAP включена .

**Предпосылки**

**SNMPNotify** (см. стр. 332 ) () вызывается .

**Параметры**

**Параметры Описание**

**ID** System ID , чтобы использовать определить этот агент .

**info** Указатель на SNMP MIB переменной информации об объекте

**oidString** Указатель для хранения строки из OID serached

**len** Длина Oid

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если подъязычная строка найдена для переменной идентификатор в MPFS .

**Flase** противном случае .

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

**10.15.2.69 IsValidCommunity Функция**

**файл**

SNMP.c (см. стр. 449 )

**C**

```
static BOOL IsValidCommunity(
char* community,
BYTE* len
);
```

**описание**

Эта процедура заполняет и проверяет сообщества тип данных , имя сообщества и длину от принятый запрос SNMP PDU. Название сообщества используется для доступа к государственным и частным memebrs на MIB.

**Замечания**

Ни один .

**Предпосылки**

**ProcessHeader** (см. стр. 368 ) () вызывается .

**Параметры**

**Параметры Описание**

**Community** Указатель на память , где строка сообщества будут храниться .

**len** Указатель на память , где длина распространяемого получает хранится.

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если действует община получила .

**ЛОЖЬ** сообщество Если не действует .

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

**10.15.2.70 IsValidInt Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BOOL IsValidInt(
DWORD* val
);
```

**описание**

Эта процедура заполняет и проверяет полученную переменную для типа данных как " ASN_INT (см. стр. 344 ) " и данных длина макс 4 байт.

**Замечания**

Ни один .

**Предпосылки**

**ProcessHeader** (см. стр. 368 ) ( ) или **ProcessGetSetHeader** (см. стр. 367 ) () вызывается .

**Параметры**

**Параметры Описание**

**Val** Указатель на память , где внутр значение переменной будет храниться .

**ReturnValues** **ИСТИНА** - Если действует целый тип и значение будет получена.

**ЛОЖЬ** Кроме целое типа данных и значения получили.

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

**10.15.2.71 IsValidOID Функция**

**файл**

SNMP.c (см. стр. 449 )

**C**

```
static BOOL IsValidOID(
BYTE* oid,
BYTE* len
);
```

**описание**

В этой подпрограмме OID тип данных " ASN_OID (см. стр. 344 ) " проверяется в принятом PDU. Если тип данных сравнивается , а затем только переменная привязки обрабатывается . OID длина и OID заполняется . Максимальная длина OID может быть 15 .

**Замечания**

Ни один .

**Предпосылки**

ProcessVariabels ( ) называется .

**Параметры**

**Параметры Описание**

**oid** Указатель на память , чтобы сохранить полученную OID строку

**len** Указатель памяти для хранения OID длину

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если тип значения является ASN_OID (см. стр. 344 ) и oid длина не более 15 .

**ЛОЖЬ** В противном случае.

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

**10.15.2.72 IsValidPDU Функция**

**файл**

SNMP.c (см. стр. 449 )

**C**

```
static BOOL IsValidPDU(
SNMP_ACTION* pdu
);
```

**описание**

Эта процедура заполняет и проверяет на полученной SNMP запроса типа PDU.

**Замечания**

Ни один .

**Предпосылки**

**ProcessHeader** (см. стр. 368 ) () вызывается .

**Параметры**

**Параметры Описание**

**val** Указатель на память , где хранится получил SNMP тип запроса .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА**, если этот запрос SNMP может быть обработан агентом.

**ЛОЖЬ** Если запрос не может быть обработан .

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

**10.15.2.73 IsValidStructure Функция**

**файл**

SNMP.h (см. стр. 451 )

http://www.microchip.com/

**C**

```
BYTE IsValidStructure(
WORD* dataLen
);
```

**описание**

Эта процедура используется для проверки , является ли принятый VarBind имеет тип структуры (см. стр. 361 ) и выяснить, связывание переменных длина структуры .

**Замечания**

Ни один .

**Предпосылки**

**ProcessHeader** (см. стр. 368 ) () вызывается .

**Параметры**

**Параметры Описание**

**DataLen** Указатель памяти для хранения OID длину структуры .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**headrbytes** связывания переменной длины.

**ЛОЖЬ** Если переменная структура данных не тип структуры (см. стр. 361 ) .

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

**10.15.2.74 OIDLookup Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BYTE OIDLookup(
PDU_INFO* pduDbPtr,
BYTE* oid,
BYTE oidLen,
OID_INFO* rec
);
```

**описание**

База данных MIB хранятся с агентом в двоичном формате MIB . Это двоичный формат MIB : , , ... > ] } , переменная ChildNode связывают зовут пунктирная строка OID. Каждый подъязычная является узлом в MIB дерева и менялись информацию . Эта процедура на прием запроса SNMP , будет искать для каждого OID во имя вар. Эта процедура будет возвращать информацию ли запрашиваемая переменная Название является частью MIB дерева данных structre этого агента или нет.

**Замечания**

Эта процедура работает для формата хранения MPFS2 MIB. Он использует MPFS2 API, читать, искать и собирать информацию от MIB базы данных .

**Предпосылки**

Действительно запрос SNMP с действительным формате OID будет получена.

**Параметры**

**Параметры Описание**

**pduDbPtr** Указатель получил информацию SNMP PDU элементы

**Oid** Указатель на строку OID для поиска

**oidLen** Длина Oid

**REC** Указатель на SNMP MIB переменной информации об объекте

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если полный OID строка найдена в MIB

**ЛОЖЬ** Если полный OID не совпадают. Кроме того, различные Erros вернулся являются

SNMP_END_OF_MIB_VIEW SNMP_NO_SUCH_NAME

SNMP_NO_SUCH_OBJ SNMP_NO_SUCH_INSTANCE

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

**10.15.2.75 ProcessGetSetHeader Функция**

**файл**

SNMP.c (см. стр. 449 )

**C**

```
static BOOL ProcessGetSetHeader(
PDU_INFO* pduDbPtr
);
```

http://www.microchip.com/

**описание**

Все переменные SNMP заголовка запроса PDU проверяются на их типы данных . Собирает **request_id** для запроса PDU SNMP .

**Fetch** , проверяет состояние ошибки , индекс ошибки и отказаться , поскольку они не должны быть обработаны как получил в поиске PDU.

Собирает письменный ретрансляторов и максимальные значения повторители в случае **Get_Bulk** запросу.

**Замечания**

Запрос PDU будет обработан только если программа возвращает TRUE,

**Предпосылки**

**ProcessHeader** (см. стр. 368 ) () вызывается и возвращает тип PDU и не возвращается SNMP_ACTION_UNKNOWN

**Параметры**

**Параметры Описание**

**pduDbPtr** Указатель получил PDU информационной базы данных .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если полученный заголовок запроса проверяется и передается .

**ЛОЖЬ** Если rxed заголовке запроса не является допустимым.

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

**10.15.2.76 ProcessHeader Функция**

**файл**

SNMP.c (см. стр. 449 )

**C**

```
static SNMP_ACTION ProcessHeader(
PDU_INFO* pduDbPtr,
char* community,
BYTE* len
);
```

**описание**

Собирает **PDU_INFO** (см. стр. 353 ) (SNMP PDU информация базы данных ) , имя сообщества , длина сообщество и длительность полезные данные . Эта функция проверяет полученную UDP-пакетов для этих различных переменных SNMP PDU. Последовательность, в которые получили эти элементы очень важно. Проверка делается для технологических возможностей агент и по максимуму Длина пакета UDP в виде пакетов UDP не может быть фрагментирован .

**Замечания**

Полученные PDU будет обработан только если программа возвращает тип PDU ещеPDU отбрасывается как не SNMP PDU.

**Предпосылки**

**UDPIsGetReady** (см. стр. 614 ) ( **SNMPAgentSocket** (см. стр. 360) ) называется в **SNMPTask** (см. стр. 372 ) (), он проверить , есть ли пакет на гнездо агента SNMP , должны вернуться ИСТИНА .

**Параметры**

**Параметры Описание**

**pduDbPtr** Указатель получил PDU информационной базы данных

**community** Указатель на уаг хранения , сообщества строки в rxed PDU

**len** Указатель на вар хранения , сообщества длины строки rxed в PDU

**Возвращаемые значения**

**Возвращаемые значения Описание**

**SNMP_ACTION** (см. стр. 328) Snmp тип запроса PDU.

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

**10.15.2.77 ProcessSetVar Функция**

**файл**

SNMP.h (см. стр. 451 )

**C**

```
BYTE ProcessSetVar(
PDU_INFO* pduDbPtr,
OID_INFO* rec,
SNMP_ERR_STATUS* errorStatus
);
```

**описание**

Создает Эта процедура обрабатывает принятый по протоколу SNMP запрос PDU для связывания переменной в запросе , а также ответ PDU.

http://www.microchip.com/

**Замечания**

Ни один .

**Предпосылки**

**ProcessVariables** ( см. стр. 369 ) ( ) называется .

**Параметры**

**Параметры Описание**

**pduDbPtr** Поступила PDU Информация указатель базы данных

**REC** Указатель на SNMP MIB переменной информации об объекте

**ErrorStatus** Указатель обновить данные о состоянии ошибки

**Возвращаемые значения**

**Возвращаемые значения Описание**

**copiedBytes** Количество скопированных байтов этой программой к SNMP PDU TX буфера.

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

**10.15.2.78 ProcessVariables Функция**

**файл**

SNMP.c (см. стр. 449 )

**C**

```
static BOOL ProcessVariables(
PDU_INFO* pduDbPtr,
char* community,
BYTE len
);
```

**описание**

После того, как получил PDU проверяется как SNMP PDU , оно передается для обработки в этой рутины . Это rotuine ручки Получить , Get_next , Get_Bulk , указан запрос и создает соответствующий ответ как Get_Response . Эта процедура будет принять решение о том запрос PDU могут быть обработаны или выбросить.

**Замечания**

Ни один

**Предпосылки**

Получил UDP пакет varified как SNMP запросу. ProcessHeader (см. стр. 368 ) () и ProcessGetSetHeader (см.страница 367) ( ) возвращается , но ЛОЖЬ.

**Параметры**

**Параметры Описание**

**pduDbPtr** Указатель получил PDU информационной базы данных

**community** Указатель на вар , хранения строка сообщества в rxed PDU

**len** Указатель Вар, хранения сообщества длину строки rxed в PDU

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если обработка запроса SNMP успешно.

**FALSE** Если обработка не удалось еще обработка не завершена.

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

**10.15.2.79 ReadMIBRecord Функция**

**файл**

SNMP.c (см. стр. 449 )

**C**

```
static void ReadMIBRecord(
DWORD h,
OID_INFO* rec
);
```

**Возвращает**

Ни один .

**описание**

Эта процедура вызывается , когдаOID строка требуется для поиска от MPFS используя адрес узла .

**Замечания**

Ни один .

**Предпосылки**

**GetOIDStringByID** (см. стр. 363) () или **GetOIDStringByAddr** (см. стр. 363 ) () вызывается .

**Параметры**

http://www.microchip.com/

**Параметры Описание**

**h** Узел adderess которого подъязычная следует читать .

**REC** Указатель для хранения SNMP MIB информацию переменная объекта

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

**10.15.2.80 SNMPCheckIfPvtMibObjRequested Функция**

**файл**

SNMP.c (см. стр. 449 )

**C**

```
static BOOL SNMPCheckIfPvtMibObjRequested(
BYTE* OIDValuePtr
);
```

**описание**

Эта процедура используется для поиска , принадлежит ли запрашиваемый объект частному объекта группы MIB агента . Если да, то , что MIB объект может быть доступна только с частным сообщества (поддерживается в SNMPv2c ) .

**Замечания**

Ни один .

**Предпосылки**

**ProcessVariables** ( см. стр. 369 ) ( ) называется .

**Параметры**

**Параметры Описание**

**OIDValuePtr** Указатель на память хранится с полученной OID.

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если запрашиваемый объект является частной ветви MIB.

**Flase** Если запрашиваемый объект является публично доступным.

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

**10.15.3 SNMP Стек Пользователи**

**Функции**

**Имя Описание**

**SNMPInit** (см. стр. 371) инициализации SNMP модуль внутренних органов.

**SNMPTask** (см. стр. 372) Опросы на каждые SNMP PDU , полученных .

**модуль**

SNMP (см. стр. 318 )

**описание**

Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные .

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

### 10.15.3.1 SNMPInit Функция

**файл**

SNMP.h (см. стр. 451 )

**C**

```
void SNMPInit();
```

**Возвращает**

Ни один

**описание**

Эта функция инициализирует агента SNMP . Один UDP сокет инициализация и открыл на порт 161 . Агент для приема и передачи все SNMP PDU, на этом сокета UDP .

**Замечания**

Эта функция вызывается только один раз в течение всего времени работы приложения . Один UDP сокет будет использоваться .

**Предпосылки**

По крайней мере один UDP сокет должен быть доступен. UDPInit (см. стр. 621 ) ( ) уже называют .

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

### 10.15.3.2 SNMPTask Функция

http://www.microchip.com/

**файл**

SNMP.h (см. стр. 451 )

**C**

BOOL **SNMPTask**();

**описание**

Ручка входящие запросы SNMP , а также любые исходящие ответы SNMP и условия тайм-аута .

**Замечания**

Ни один

**Предпосылки**

**SNMPInit** (см. стр. 371 ) ( ) уже называют .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если модуль SNMP закончил с государством

**ЛОЖЬ** Если государство не была завершена .

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

## 10.15.4 Функции

**Функции**

**Имя Описание**

**_IsSNMPv3ValidStructure** (См. стр. 374 ) Decode переменную структуру длины.

**_Snmpv3IsValidAuthStructure** (См. стр. 374 ) Decode переменную структуру длины.

**_Snmpv3IsValidInt** (См. стр. 375 ) Проверяет переменную типа данных как INT и извлекает его значение.

**FindOIDsFromSnmpV3Request** (см. стр. 376 ) Находит количество varbinds в списке VarBind , полученной в PDU SNMPv3 .

**getSnmpV2GenTrapOid** (см. стр. 376 ) постановляет общего кода ловушки для общего ловушки OID.

**IsSnmpV3ASNNull** (см. стр. 377 ) Проверяет тип значения , как ASN_NULL (см. стр. 344 ) .

**IsSnmpv3ValidOID** (см. стр. 377 ) Заполняет OID тип , длину и подъязычная строку из получил PDU .

**IsSNMPv3ValidStructure** (см. стр. 378 ) Decode переменную структуру длины.

**ProcessGetBulkVar** (см. стр. 378 ) Этот рутинный процессSNMPv2c Получить общий запрос .

**ProcessGetNextVar** (см. стр. 379 ) Получает следующий узел из базы данных MIB .

**ProcessGetVar** (см. стр. 380 ) Процессы SNMP Получить запроса PDU .

**ProcessSnmpv3MsgData** (см. стр. 380 ) Эта процедура обрабатывает запрос SNMPv3 и параллельно создает PDU ответа .

**SNMPGetExactIndex** (см. стр. 381 ) для поиска точного индекса узла в случае последовательности переменная .

**SNMPIdRecrdValidation** (см. стр. 381 ) используется для ограничения доступа динамичный и не динамическая OID струны для perticular SNMP версии .

**SNMPIsValidSetLen** (см. стр. 382 ) Проверяет набор переменную длину данных в тип данных.

**Snmpv3AESDecryptRxedScopedPdu** (см. стр.383) Входящий SNMPv3 областью действия расшифровку PDU с использованием AES расшифровка протокола .

**Snmpv3AESEncryptResponseScopedPdu** (см.страница 383) Исходящие SNMPv3 областью действия PDU шифрования с использованием AES Протокол шифрования .

**Snmpv3AuthenticateRxedPduForDataIntegrity** (см.страница 384 ) Аутентификация на входящий SNMPv3 USM PDU с помощью MD5 или SHA

**Snmpv3AuthenticateTxPduForDataIntegrity** (см.страница 384 ) Подлинность исходящего SNMPv3 USM PDU с помощью MD5 или SHA

**Snmpv3AuthKeyZeroing2HmacBufLen64** (см.страница 385 ) Коврик нуля до ключа аутентификации локализованной буфера.

**Snmpv3BufferPut** (см. стр. 385) данных Копии BYTE с динамической памятью буфера .

**Snmpv3CmprTrapSecNameAndSecLvlWithUSMDb** (См. стр. 386 ) Регулярное найти индекс имени пользователя в пользовательском база данных таблицы .

**Snmpv3ComputeHMACIpadOpadForAuthLoclzedKey** (См. стр. 386 ) Вычислить HMAC внутренняя и внешняя площадка для авторизации локализованы ключ .

**Snmpv3ComputeHmacMD5Digest** (см. стр. 387 ) Вычислить HMAC - MD5 код аутентификации

**Snmpv3ComputeHmacShaDigest** (см. стр. 388 ) Вычислить HMAC - Ша код аутентификации

**Snmpv3ComputeMd5HmacCode** (см. стр. 388 ) Вычислить HMAC - MD5 код аутентификации

**Snmpv3ComputeShaHmacCode** (см. стр. 389 ) Вычислить HMAC - Ша код аутентификации

**Snmpv3FormulateEngineID** (см. стр. 389 ) формулирует snmpEngineID (см. стр. 428) для SNMPv3 двигатель .

**Snmpv3FreeDynAllocMem** (см. стр. 390 ) Выделенные освобождение динамической памяти осуществляется это рутина.

**Snmpv3GetAuthEngineTime** (см. стр. 390 ) Обновляет переменной времени двигатель SNMP ' snmpEngineTime (см. стр. 429) " для SNMPv3 двигатель .

**Snmpv3GetBufferData** (см. стр. 391 ) Читает данные байт из динамически выделяемой памяти буфера .

**Snmpv3GetSecurityLevel** (см. стр. 391) Получить уровень безопасности от аутентификации и конфиденциальности типа.

**Snmpv3GetTrapSecurityLevel** ( см. стр. 392 ) обычные найти в докладе , Auth и приватности флаги Настройки в ловушке.

**Snmpv3Init** ( см. стр. 392) инициализации SNMPv3 .

**Snmpv3InitializeUserDataBase** (см. стр. 393) SNMPv3 базы данных глобальной пользователь Intialize умолчанию .

**Snmpv3IsValidAuthStructure** (см. стр. 393 ) Decode переменную структуру длины.

**Snmpv3IsValidInt** (см. стр. 394 ) Проверяет переменную типа данных как INT и извлекает его значение.

**Snmpv3MsgProcessingModelProcessPDU** (см.страница 394 ) Эта процедура собирает или заполняет сообщение обработка модель детали от полученных Запрос PDU SNMPv3 или к PDU ответа соответственно.

**Snmpv3Notify** (см. стр. 395 ) создает и отправляет SNMPv3 перехваты PDU .

**Snmpv3Pswd2LocalizedAuthKeyMD5Hashing** (см. страница 395) Преобразование MD5 Auth пароль для локализованной Key помощью SNMPEngineID .

**Snmpv3Pswd2LocalizedAuthKeySHAHashing** (см.страница 396 )Преобразование SHA Auth пароль для локализованной Key помощью SNMPEngineID .

**Snmpv3ReportPdu** (см. стр. 396 ) Создает доклада PDU infomration для отчета PDU.

**Snmpv3ScopedPduProcessing** (см. стр. 397 ) Эта процедура собирает областью действия информацию заголовка PDU из принятого запроса PDU или заполняет SNMPv3 к PDU ответа соответственно.

**Snmpv3SetErrorStatus** (см. стр. 397 ) указан SNMPv3 статус ошибки в PDU ответа.

**Snmpv3TrapScopedpdu** (см. стр. 398 ) перехваты PDU области видимости строительство заголовка PDU.

**Snmpv3UserSecurityModelProcessPDU** (см. стр.399) Эта процедура собирает или заполняет модель безопасности ПАРАМЕТРЫ детали от полученных SNMPv3 запросить PDU или в PDU ответа соответственно.

**Snmpv3UsmAesEncryptDecryptInitVector** (см.страница 399 ) Шифрование AES и дешифрование инициализации вектор. (RFC 3826)

**Snmpv3UsmOutMsgAuthenticationParam** (см.страница 400 ) Оба MD5 и SHA1 используется для исходящего сообщения аутентификации .

**Snmpv3USMOutMsgPrivParam** (см. стр. 400 ) SNMP USM из сообщение использует конфиденциальности протокол (RFC 3826 )

**Snmpv3UsmSnmpEngnAuthPrivPswdLocalization** ( см. стр. 400) Преобразование Auth и Priv пароль для локализованной Key используя SNMPEngineID .

**Snmpv3ValidateEngineId** (см. стр. 401) Проверка двигателя ID .

**Snmpv3ValidateSecNameAndSecLvl** (см. стр.401) Vaidate имя безопасности с уровнем безопасности.

**Snmpv3ValidateSecurityName** (см. стр. 402 ) Проверка SNMPv3 имя пользователя или имя безопасности .

**модуль**

SNMP (см. стр. 318 )

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

**10.15.4.1 _IsSNMPv3ValidStructure Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
BOOL _IsSNMPv3ValidStructure(
UINT8* wholeMsgPtr,
WORD* pos,
WORD* dataLen
);
```

**описание**

Эта процедура заполняет и проверяет полученную переменную для типа данных , как " СТРУКТУРЫ (см. стр. 361 ) " и длина данных для макс 4 байт.

**Замечания**

Ни один .

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**wholeMsgPtr** Указатель на память , где храниться внутр значение переменной .

Положение пос в буфере памяти , где тауре данные, которые будут varified хранится

**Val** Указатель на память , где внутр значение переменной будет храниться .

**ReturnValues** верны - Если действует целый тип и значение будет получена.

**ЛОЖЬ** Кроме целое типа данных и значения получили.

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

**10.15.4.2 _Snmpv3IsValidAuthStructure Функция**

**файл**

SNMPv3.c (см. стр. 440 )

**C**

```
BYTE _Snmpv3IsValidAuthStructure(
WORD* dataLen
);
```

описание

Эта процедура используется для проверки , является ли принятый VarBind имеет тип структуры (см. стр. 361 ) и выяснить, связывание переменных длина структуры . Это rotuine относится только к входящего запроса в SNMPv3 динамически выделяемой памяти буфера ' gSnmpV3InPduWholeMsgBuf (см. стр. 422 ) .

**Замечания**

Ни один .

**Предпосылки**

**ProcessHeader** (см. стр. 368 ) () вызывается .

**Параметры**

**Параметры Описание**

**DataLen** Указатель памяти для хранения OID длину структуры .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**headrbytes** связывания переменной длины.

**ЛОЖЬ** Если переменная структура данных не тип структуры (см. стр. 361 ) .

http://www.microchip.com/

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

**10.15.4.3 _Snmpv3IsValidInt Функция**

**файл**

SNMPv3.h (см. стр. 443 )

**C**

```
BOOL _Snmpv3IsValidInt(
UINT8 * wholeMsgPtr,
WORD* pos,
DWORD* val
);
```

**описание**

Эта процедура заполняет и проверяет полученную переменную для типа данных как " ASN_INT (см. стр. 344 ) " и данных длина макс 4 байт.

**Замечания**

Ни один .

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**wholeMsgPtr** Указатель на память , где храниться внутр значение переменной .

Положение пос в буфере памяти , где тауре данные, которые будут varified хранится

**Val** Указатель на память , где внутр значение переменной будет храниться .

**ReturnValues** верны - Если действует целый тип и значение будет получена.

**ЛОЖЬ** Кроме целое типа данных и значения получили.

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

**10.15.4.4 FindOIDsFromSnmpV3Request Функция**

**файл**

SNMPv3.c (см. стр. 440 )

http://www.microchip.com/

C

```
static BYTE FindOIDsFromSnmpV3Request(
WORD pdulen
);
```

**описание**

Эта процедура используется для поиска количество OIDs запрашиваемых в принятом SNMP PDU.

**Замечания**

Ни один .

**Предпосылки**

ProcessVariables ( см. стр. 369 ) ( ) называется .

**Параметры**

**Параметры Описание**

**pdulen** Длина запросу SNMP PDU получил .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**varCount** Количество OIDs найдено в запросе PDU.

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

**10.15.4.5 getSnmpV2GenTrapOid Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
BYTE * getSnmpV2GenTrapOid(
BYTE generic_trap_code,
BYTE * len
);
```

**описание**

Эта функция решает, учитывая общий код ловушки для общего ловушки OID.

**Замечания**

Это будет работать, если generic_trap_code не подпадающих под GENERIC_TRAP_NOTIFICATION_TYPE (см. стр. 327 )

**Предпосылки**

**SNMPNotifyPrepare** (см. стр. 334 ) ( ) уже называют .

**Параметры**

**Параметры Описание**

generic_trap_code GENERIC_TRAP_NOTIFICATION_TYPE (см. стр. 327 )

**len** общий ловушка длиной OID

**Возвращаемые значения**

**Возвращаемые значения Описание**

BYTE * TRAP OID

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

**10.15.4.6 IsSnmpV3ASNNull Функция**

**файл**

SNMPv3.c (см. стр. 440 )

C

**static** BOOL **IsSnmpV3ASNNull**();

**описание**

Для Получить , get_next , Get_Bulk SNMP reuest ,вар связывают тип значения данных должны быть ASN_NULL (см. стр. 344 ) и Значение поля должно быть NULL и . Эта процедура проверяет поля типа данных и значения в принятых запросов . SET Запрос , тип данных значение не может быть ASN_NULL (см. стр. 344) , в противном случае запрос SNMP не обрабатывается. это rotuine только относится к входящему SNMPv3 запросу динамически выделенный буфер памяти ' **gSNMPv3ScopedPduRequestBuf**

(См. стр. 423 ) .

**Замечания**

Ни один .

**Предпосылки**

Ни один

http://www.microchip.com/

**Параметры**

**Параметры Описание**

**ИСТИНА** Если тип значения является ASN_NULL (см. стр. 344 ) и значение NULL .

**ЛОЖЬ** Если тип данных и значение кроме ASN_NULL (см. стр. 344 ) и NULL соответственно

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

**10.15.4.7 IsSnmpv3ValidOID Функция**

**файл**

SNMPv3.c (см. стр. 440 )

C

```
static BOOL IsSnmpv3ValidOID(
BYTE* oid,
BYTE* len
);
```

**описание**

В этой подпрограмме OID тип данных " ASN_OID (см. стр. 344 ) " проверяется в принятом PDU. Если тип данных сравнивается , а затем только переменная привязки обрабатывается . OID длина и OID заполняется . Максимальная длина OID может быть 15 .

**Замечания**

Ни один .

**Предпосылки**

**ProcessVariabels** ( ) называется .

**Параметры**

**Параметры Описание**

подъязычная Указатель на память , чтобы сохранить полученную OID строку

**len** Указатель памяти для хранения OID длину

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если тип значения является ASN_OID (см. стр. 344 ) и подъязычная длина не более 15 .

В противном случае.

http://www.microchip.com/

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

**10.15.4.8 IsSNMPv3ValidStructure Функция**

**файл**

SNMPv3.c (см. стр. 440 )

C

BOOL **IsSNMPv3ValidStructure**(
WORD* **dataLen**
);

**описание**

Эта процедура используется для проверки , является ли принятый VarBind имеет тип структуры (см. стр. 361 ) и выяснить, связывание переменных длина структуры . Это rotuine относится только к входящего запроса в SNMPv3 динамически выделяемой памяти буфера ' gSNMPv3ScopedPduRequestBuf (см. стр. 423 ) .

**Замечания**

Ни один .

**Предпосылки**

ProcessHeader (см. стр. 368 ) () вызывается .

**Параметры**

**Параметры Описание**

DataLen Указатель памяти для хранения OID длину структуры .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если тип и значение достоверности данных Структура будет получена.

**ЛОЖЬ** Если переменная структура данных не тип структуры (см. стр. 361 ) .

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

**10.15.4.9 ProcessGetBulkVar Функция**

**файл**

SNMP.h (см. стр. 451 )

http://www.microchip.com/

C

```
BYTE ProcessGetBulkVar(
OID_INFO* rec,
BYTE* oidValuePtr,
BYTE* oidLenPtr,
BYTE* successor,
PDU_INFO* pduDbPtr
);
```

**описание**

ProcessVariables (см. стр. 369 ) () обрабатывает принятый PDU запроса SNMP для каждого из переменной связывания в связывание переменных список для получения PDU ответа . В зависимости от количества Max_repetitions для каждой переменной в списке для которых Массовая информация ожидается , ProcessGetBulkVar ( ) выполняется . Это поиск следующего лексикографическом порядке преемником из OID получил в запросе. Для каждого из итераций до макс - повторов , следующий лист узел искать в MIB тому из листового узла , найденного в последней итерации , для связываниясоответствующей переменной .

**Замечания**

Ни один .

**Предпосылки**

**ProcessVariables** ( см. стр. 369 ) ( ) называется .

**Параметры**

**Параметры Описание**

**REC** Указатель на SNMP MIB информации переменная объекта OID

**oidValuePtr** Указатель на новый узел OID найти в MIB

Длина oidLenPtr Oid

значение преемник ' i-го лексикографического наследника можно найти

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ЛОЖЬ** Если нет лексикографический наследник найдено

temp.v [0] Общее количество байтов , скопированных в ответном пакете

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

**10.15.4.10 ProcessGetNextVar Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
BYTE ProcessGetNextVar(
OID_INFO* rec,
PDU_INFO* pduDbPtr
);
```

**описание**

Эта процедура читает в MIB хранится с агентом в формате MPFS2 . Он будет искать для первого лексикографического наследника Имя переменная связывания находится в поступающей GetNextRequest -PDU . Если найдено , соответствующая переменная связывания имя и значение поля в PDU ответа устанавливаются в имя и стоимости расположенного переменной. Если лексикографическое succesor не найдено ,vlaue подал установлен на " endofMibView " и имя поля сохраняется как в запросе .

**Замечания**

Ни один .

**Предпосылки**

ProcessVariables ( см. стр. 369 ) называется .

**Параметры**

**Параметры Описание**

**REC** Указатель на информации об объекте SNMP MIB , для которых следующий узел можно найти

**pduDbPtr** Указатель получил PDU информационной базы данных

**Возвращаемые значения**

**Возвращаемые значения Описание**

Значение Общее число байтов , скопированных в ответном пакете , если успешно.

**ЛОЖЬ** Если Конец MIB достигается или обработка провал.

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

**10.15.4.11 ProcessGetVar Функция**

**файл**

SNMP.h (см. стр. 451 )

http://www.microchip.com/

C

```
BYTE ProcessGetVar(
OID_INFO* rec,
BOOL bAsOID,
PDU_INFO* pduDbPtr
);
```

**описание**

Эта процедура обрабатывает принятый SNMP Получить запроса PDU для связывания переменной в запросе , а также создает ответ PDU.

**Замечания**

Ни один .

**Предпосылки**

**ProcessVariables** ( см. стр. 369 ) ( ) называется .

**Параметры**

**Параметры Описание**

**REC** Указатель на SNMP MIB переменной информации об объекте

**bAsOID** Oid флаг .

**Возвращаемые значения**

**Возвращаемые значения Описание**

Варлен Количество байт положить в ответ TX PDU

**ЛОЖЬ** Если какой-либо из элементов проверки запроса PDU не удается.

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

**10.15.4.12 ProcessSnmpv3MsgData Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
BOOL ProcessSnmpv3MsgData(
PDU_INFO* pduDbPtr
);
```

**описание**

После того, как получил PDU проверяется как SNMPv3 PDU , оно передается для обработки в этой рутины . Это rotuine ручки Получить , Get_next , Get_Bulk , указан запрос и создает соответствующий ответ как Get_Response . Эта процедура будет принять решение о том запрос PDU могут быть обработаны или выбросить.

**Замечания**

Ни один

**Предпосылки**

Получил UDP пакет varified как правильный запрос SNMPv3 .

**Параметры**

**Параметры Описание**

**pduDbPtr** Указатель получил PDU информационной базы данных

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если обработка запроса SNMP успешно.

**FALSE** Если обработка не удалось еще обработка не завершена.

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

**10.15.4.13 SNMPGetExactIndex Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
BOOL SNMPGetExactIndex(
SNMP_ID var,
SNMP_INDEX index
);
```

**описание**

Это функция обратного вызова вызывается модуль SNMP. Пользователь SNMP должен реализовать эту функцию в приложении пользователя и обеспечить соответствующие данные при вызове . Эта функция будет вызываться только для OID переменной последовательности типа .

**Замечания**

Индекс только последовательность должно быть обработано в этой функции .

http://www.microchip.com/

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**Var** Переменная идентификатор согласно mib.h (вход)

**index** индекс переменной (вход)

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Если точное значение индекса существует для данной переменной в данном индексе .

В противном случае.

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

**10.15.4.14 SNMPIdRecrdValidation Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
BOOL SNMPIdRecrdValidation(
PDU_INFO * pduPtr,
OID_INFO * var,
BYTE * oidValuePtr,
BYTE oidLen
);
```

**описание**

Это функция обратного вызова вызывается модуль SNMP. Пользователь SNMP должен реализовать эту функцию в соответствии с SNMP версии . один нужно добавить новые SNMP MIB идентификаторы hereas на SNMP версии . e.g - SYS_UP_TIME (250) является общей для V1/V2/V3

Engine_id - это часть V3 , так поставить все Вар идентификаторы SNMPv3 в Макро STACK_USE_SNMPV3_SERVER .

**Замечания**

Ни один .

**Предпосылки**

Ни один

http://www.microchip.com/

**Параметры**

**Параметры Описание**

**Var** Переменная REC чей рекорд ID должны быть проверены

**oidValuePtr** OID Значение

Длина oidLen oidValuePtr

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** ЕслиID Var существует.

В противном случае.

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

**10.15.4.15 SNMPIsValidSetLen Функция**

**файл**

SNMP.h (см. стр. 451 )

C

BOOL **SNMPIsValidSetLen**(
SNMP_ID **var**,
BYTE **len**,
BYTE **index**
);

**описание**

Эта процедура используется для проверки dyanmic переменную длину данных для переменной типа данных. Он используется , когда запрос SET является обрабатываются . Это функция обратного вызова вызывается модуль. Приложение пользователя должен реализовать эту функцию .

**Замечания**

Эта функция будет называться только для динамических переменных , которые определены как ASCII_STRING и OCTET_STRING (см. стр.352) ( то есть длина данных превышает 4 байта )

**Предпосылки**

**ProcessSetVar** (см. стр. 369 ) () вызывается .

**Параметры**

**Параметры Описание**

http://www.microchip.com/

**Var** Переменная идентификатор , значение которого должно быть установлено

**Len** значение Длина то есть быть подтверждено .

Экземпляр индекс OID

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА**, если дано переменная может быть установлена на данном Len

**ЛОЖЬ** в противном случае.

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

**10.15.4.16 Snmpv3AESDecryptRxedScopedPdu Функция**

**файл**

SNMP.h (см. стр. 451 )

C

BYTE **Snmpv3AESDecryptRxedScopedPdu**();

**описание**

Эта процедура расшифровывает SNMPv3 входящий PDU , используя протокол AES , но до этого зашифрованном длины данных является verified. Если длина зашифрованного октет - строка не кратно 8 , то dryption будут прекращены . RFC - . 3414 (раздел 8)

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Возвращаемые значения**

**Возвращаемые значения Описание**

SNMPV3_MSG_PRIV_FAIL Невыполнение

SNMPV3_MSG_PRIV_PASS Успех

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

**10.15.4.17 Snmpv3AESEncryptResponseScopedPdu Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

BYTE **Snmpv3AESEncryptResponseScopedPdu**(
SNMPV3_RESPONSE_WHOLEMSG* **plain_text**
);

**описание**

Эта процедура шифрует SNMPv3 исходящего PDU , используя протокол AES поддерживать конфиденциальность данных. Данные шифруются в режиме Cipher Block Chaining . Длина зашифрованных данных должно быть кратно 8 , и это не то тогда данные проложенный в конце концов , если это необходимо . RFC - . 3414 (раздел 8)

**Замечания**

Ни один

**Предпосылки**

**SNMPv3Init** () и **ProcessVariabels** ( ) называются .

**Параметры**

**Параметры Описание**

**Возвращаемые значения**

**Возвращаемые значения Описание**

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

**10.15.4.18 Snmpv3AuthenticateRxedPduForDataIntegrity Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

BYTE **Snmpv3AuthenticateRxedPduForDataIntegrity**(
SNMPV3_REQUEST_WHOLEMSG* **rxDataPtr**
);

**описание**

Эта процедура проверяет подлинность SNMPv3 входящего отчета сообщение PDU , а также для различных типов запросов GET как с MD5 и SHA protocol.Ifполучил PDU имя пользователя аналогична «начальной» , то есть стоит заключать отчет PDU . RFC - 3414 .

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Параметры**

**Параметры Описание**

**rxDataPtr** входящие PDU

**Возвращаемые значения**

**Возвращаемые значения Описание**

SNMPV3_MSG_AUTH_PASS аутентификации ( см. стр. 89 ) успех

SNMPV3_MSG_AUTH_FAIL аутентификации ( см. стр. 89 ) отказ

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

**10.15.4.19 Snmpv3AuthenticateTxPduForDataIntegrity Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
BYTE Snmpv3AuthenticateTxPduForDataIntegrity(
SNMPV3_RESPONSE_WHOLEMSG* txDataPtr
);
```

**описание**

Эта процедура проверяет подлинность SNMPv3 исходящих отчета сообщение PDU , а также для получите ответ PDU для всего сообщения . RFC - 3414 .

**Замечания**

Ни один

**Предпосылки**

**SNMPv3Init** () и **ProcessVariabels** ( ) называются .

http://www.microchip.com/

**Параметры**

**Параметры Описание**

**txDataPtr** исходящих PDU

**Возвращаемые значения**

**Возвращаемые значения Описание**

SNMPV3_MSG_AUTH_PASS аутентификации ( см. стр. 89 ) успех

SNMPV3_MSG_AUTH_FAIL аутентификации ( см. стр. 89 ) отказ

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

**10.15.4.20 Snmpv3AuthKeyZeroing2HmacBufLen64 Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
void Snmpv3AuthKeyZeroing2HmacBufLen64(
UINT8* authKey,
UINT8 authKeyLen,
UINT8 hashType
);
```

**описание**

эта процедура дополнит ( 64- authKeyLen ) количество нулей в конце авт ключевой локализованной буфера.

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Параметры**

**Параметры Описание**

аутентификации authkey ключ буфер

**authKeylen** аутентификации длина ключа

тип аутентификации hashType

http://www.microchip.com/

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

**10.15.4.21 Snmpv3BufferPut Функция**

**файл**

SNMP.h (см. стр. 451 )

C

BOOL **Snmpv3BufferPut**(
BYTE **val**,
SNMPV3MSGDATA * **putbuf**
);

**описание**

Реализация стека SNMPv3 использует динамически выделенный буфер памяти для обработки запроса и ответа Пакеты . Эта процедура копирует данные байт в выделенный буфер и обновляет смещение длины Счетчики .

**Замечания**

Эта процедура используется в стеке SNMPv3 . При необходимости должны использоваться кодом приложения , действительные указатели должны быть переданы эта процедура .

**Предпосылки**

Стек SNMPv3 с большим успехом выделено буфер динамической памяти из кучи

**Параметры**

**Параметры Описание**

**Val** значение записываемого байта в буфер

**putbuf** указатель на динамически выделенный буфер , к которому ' вал ' , чтобы записать

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА**, если успешно написать в буфер

**ЛОЖЬ** отказ в письменной форме в буфер

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

**10.15.4.22 Snmpv3CmprTrapSecNameAndSecLvlWithUSMDb функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

BOOL **Snmpv3CmprTrapSecNameAndSecLvlWithUSMDb**(
BYTE **tragetIndex**,
BYTE **userTrapSecLen**,
UINT8 * **userTrapSecurityName**,
STD_BASED_SNMPV3_SECURITY_LEVEL **securityLevel**
);

**описание**

Есть два различных база данных таблицы , определенные с SNMPv3 стек , как ' snmpV3UserDataBase (см. стр. 432 ) " и ' gSnmpv3TrapConfigData (см. стр. 424) . Эта процедура используется для validte настройку уровня безопасности ловушка пользователь с SET просьба.

**Замечания**

Ни один .

**Предпосылки**

Операция SET будет разрешено , если условия безопасности USM и имя пользователя безопасности в запросе сравнивается с одним из имя безопасности пользователя хранятся в базе данных USM пользователя.

**Параметры**

**Параметры Описание**

**userTrapSecLen** сек пользователя Длина имени в запросе SET

**userTrapSecurityName** указатель на имя сек пользователя в запросе SET

**securityLevel** уровень ловушка безопасности должен быть установлен на агента

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** , если установка ловушки целевой уровень пользователя сек успешно

**Flase** Если множество не удалось из-за отсутствия согласования параметров безопасности

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

**10.15.4.23 Snmpv3ComputeHMACIpadOpadForAuthLoclzedKey**

**функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
void Snmpv3ComputeHMACIpadOpadForAuthLoclzedKey(
UINT8 userDBIndex
);
```

**описание**

Эта процедура вычисляет HMAC внутренней и внешней строки площадкой для авторизации локализованы ключа. RFC - 2104 .

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Параметры**

**Параметры Описание**

**userDBIndex** хранения паролей poniter

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

**10.15.4.24 Snmpv3ComputeHmacMD5Digest Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
UINT8* Snmpv3ComputeHmacMD5Digest(
UINT8 * inData,
UINT32 dataLen,
UINT8* userExtendedLclzdKeyIpad,
UINT8* userExtendedLclzdKeyOpad
);
```

**описание**

Эта процедура поддерживает проверку подлинности источника данных и аутентификацию MD5 целостности данных . Оба IPad и Opad используется для рассчитать authencate переварить строку . RFC - 3414 (раздел 6)

**Замечания**

http://www.microchip.com/

Ни один

**Предпосылки**

**SNMPv3Init** () и ProcessVariabels ( ) называются .

**Параметры**

**Параметры Описание**

**digestptr** выходная строка

ввод данных входных данных,

**DataLen** ввода длины данных

userExtendedLclzdKeyIpad IPAD

userExtendedLclzdKeyOpad Opad

**Возвращаемые значения**

**Возвращаемые значения Описание**

UINT8 * HMAC MD5, строку

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

**10.15.4.25 Snmpv3ComputeHmacShaDigest Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

UINT8* **Snmpv3ComputeHmacShaDigest**(
UINT8 * **inData**,
UINT32 **dataLen**,
UINT8* **userExtendedLclzdKeyIpad**,
UINT8* **userExtendedLclzdKeyOpad**
);

**описание**

Эта процедура поддерживает проверку подлинности источника данных и аутентификации SHA целостности данных . Оба IPad и Opad используется для рассчитатьauthencate переварить строку . RFC - 3414 (раздел 6)

**Замечания**

Ни один

http://www.microchip.com/

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Параметры**

**Параметры Описание**

**digestptr** выходная строка

ввод данных входных данных,

**DataLen** ввода длины данных

userExtendedLclzdKeyIpad IPAD

userExtendedLclzdKeyOpad Opad

**Возвращаемые значения**

**Возвращаемые значения Описание**

UINT8 * HMAC SHA переварить строку

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

**10.15.4.26 Snmpv3ComputeMd5HmacCode Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
void Snmpv3ComputeMd5HmacCode(
UINT8 xx_bits,
UINT8* digestptr,
UINT8 * indata,
UINT32 dataLen,
UINT8* userExtendedLclzdKeyIpad,
UINT8* userExtendedLclzdKeyOpad
);
```

**описание**

Эта процедура поддерживает проверку подлинности источника данных и аутентификацию MD5 целостности данных . Оба IPad и Opad используется для рассчитатьauthencate переварить строку . RFC - 3414 (раздел 6)

**Замечания**

Ни один

http://www.microchip.com/

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Параметры**

**Параметры Описание**

xx_bits 96 бит

digestptr выходная строка

ввод данных входных данных,

**DataLen** ввода длины данных

userExtendedLclzdKeyIpad IPAD

userExtendedLclzdKeyOpad Opad

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

**10.15.4.27 Snmpv3ComputeShaHmacCode Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
void Snmpv3ComputeShaHmacCode(
UINT8 xx_bits,
UINT8* digestptr,
UINT8 * indata,
UINT32 dataLen,
UINT8* userExtendedLclzdKeyIpad,
UINT8* userExtendedLclzdKeyOpad
);
```

**описание**

Эта процедура поддерживает проверку подлинности источника данных и аутентификации SHA целостности данных . Оба IPad и Opad используется для рассчитатьauthencate переварить строку . RFC - 3414 (раздел 6)

**Замечания**

Ни один

**Предпосылки**

**SNMPv3Init** () и **ProcessVariabels** ( ) называются .

http://www.microchip.com/

**Параметры**

**Параметры Описание**

xx_bits 96 бит

**digestptr** выходная строка

ввод данных входных данных,

**DataLen** ввода длины данных

userExtendedLclzdKeyIpad IPAD

userExtendedLclzdKeyOpad Opad

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

**10.15.4.28 Snmpv3FormulateEngineID Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
void Snmpv3FormulateEngineID(
UINT8 fifthOctectIdentifier
);
```

**описание**

Формулирует snmpEngineID (см. стр. 428 ) в зависимости от стоимости " fifthOctectIdentifier ' . как MAC_ADDR_ENGN_ID

С помощью приложения MAC-адрес. Defualt ' fifthOctectIdentifier ' установлен в MAC_ADDR_ENGN_ID в виде следующих октетов , используемых для snmpEngineID (см. стр. 428 ) имеют MAC-адрес.

Пользователь может установить этот октет по своему выбору , чтобы fomulate новый snmpEngineID (см. стр. 428 ) .

fifthOctectIdentifier = IPV4_ADDR_ENGN_ID ;

Если fifthOctectIdentifier = ADMIN_ASSIGNED_TEXT ; или fifthOctectIdentifier = ADMIN_ASSIGNED_OCTETS ; то следующий октета должны предоставляться администратором через некоторый механизм интерфейса пользовательских приложений . параметр API ' fifthOctectIdentifier ' должен быть upated в intefrace API перед прохождением через **Snmpv3FormulateEngineID** ( ) .

**Замечания**

Аутентификации ( см. стр. 89 ) и ключи шифрования генерируются с использованием соответствующих паролей и snmpEngineID (см. стр. 428) . Если snmpEngineID (см. стр. 428 ) недавно

настроен , то аутентификации и конфиденциальности ключи бы также изменить . Поэтому при использовании этого API , чтобы изменить snmpEngineID (см. стр. 428) динамически , следует позаботиться , чтобы обновить новые локализованные ключи в агента , а также у менеджера .

**Предпосылки**

**InitAppConfig** (); называется .

**Параметры**

**Параметры Описание**

**fifthOctectIdentifier** Значение 5-го октета в snmpEngineID (см. стр. 428 ), который указывает, как сложится ( 6-я и следующие октета) отформатированы .

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

**10.15.4.29 Snmpv3FreeDynAllocMem Функция**

**файл**

SNMPv3.c (см. стр. 440 )

C

**void Snmpv3FreeDynAllocMem**();

**описание**

На успешном завершении обработки запроса SNMPv3 , или отказа в обработке из-за неправильного Форматы PDU ,выделено динамической памяти требуется , чтобы освободиться . Эта процедура вызывает бесплатно ( ) , чтобы освободить память.

**Замечания**

Стек SNMPv3 же использует динамическую память широко для различных нужд обработки , следовательно, входящие и исходящие буферы PDU памяти создаются . Эта процедура проверяет памяти выделяется , прежде чем он пытается для освобождение .

**Предпосылки**

Выделяется динамический буфер памяти .

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

**10.15.4.30 Snmpv3GetAuthEngineTime Функция**

**файл**

SNMP.h (см. стр. 451 )

http://www.microchip.com/

C

**void Snmpv3GetAuthEngineTime**();

**описание**

' snmpEngineTime (см. стр. 429 ) »используется для Своевременность проверки безопасности на уровне сообщений . Snmp двигатель Держать обновления '' snmpEngineTime (см. стр. 429 ) "переменная для проверки временное окно для requrest и ответы / сообщить т.д. Эта процедура также обновляет snmpEngineBoots (см. стр. 428 ) в сценариях внутреннего сброса таймера или ' snmpEngineTime (см. стр. 429 ) " CNTR ovrflowed на (2 ^ 31 -1) значения, указанного в RFC3411 .

**Замечания**

Эта процедура вызывается каждый раз, когда RX / TX обработки PDU обрабатывается агентом SNMP. Обновление ' snmpEngineTime (см. стр. 429) " и требует frequnet доступ к внутренним регистрам таймера.

**Предпосылки**

**SNMPInit** (см. стр. 371 ) ( ) ; называется .

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

**10.15.4.31 Snmpv3GetBufferData Функция**

**файл**

SNMP.h (см. стр. 451 )

C

BYTE **Snmpv3GetBufferData**(
SNMPV3MSGDATA **getbuf**,
UINT16 **pos**
);

**описание**

Реализация стека SNMPv3 использует динамически выделенный буфер памяти для обработки запроса и ответа Пакеты . Эта процедура считывает данные байт из выделенного буфера в положениях ( смещение) , предоставленной .

**Замечания**

Читать смещения позиции должны быть предоставлены каждый раз процедура называется . Этот API не увеличивают буфер чтения смещение автоматически , каждый раз она называется .

**Предпосылки**

Стек SNMPv3 с большим успехом выделено буфер динамической памяти из кучи

**Параметры**

**Параметры Описание**

**getbuf** Структура откуда читать байт данных .

Положение пос в буфер, из которого данные, которые будут читать

**Возвращаемые значения**

**Возвращаемые значения Описание**

BYTE 1 Значение прочитанный байт

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

**10.15.4.32 Snmpv3GetSecurityLevel Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
BYTE Snmpv3GetSecurityLevel(
BYTE userIndex
);
```

**описание**

Эта процедура использует authenticationa тип дп конфиденциальности, чтобы узнать точный тип безопасности перечисления .

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Возвращаемые значения**

**Возвращаемые значения Описание**

NO_REPORT_PRIVACY_AND_AUTH_PROVIDED Только конфиденциальности и аутентификации ( см. стр. 89 )

NO_REPORT_NO_PRIVACY_BUT_AUTH_PROVIDED всего Auth , никакая частная жизнь и не отчет

NO_REPORT_NO_PRIVACY_NO_AUTH Нет отчет , нет конфиденциальности и без аутентификации ( см. стр. 89)

http://www.microchip.com/

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

**10.15.4.33 Snmpv3GetTrapSecurityLevel Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

BYTE **Snmpv3GetTrapSecurityLevel**(
STD_BASED_SNMPV3_SECURITY_LEVEL **securityLevel**
);

**описание**

Эта процедура , чтобы найти отчет , Auth и приватности флажков, задающих в ловушку еще только предстоит сформировать . Октет сообщение флаги 'S мере значительные три бита : Отчетные , PrivFlag , AuthFlag формирует различные комбинации уровня Secuity .

**Замечания**

Ни один .

**Предпосылки**

Ни один

**Возвращаемые значения**

**Возвращаемые значения Описание**

Не NO_REPORT_NO_PRIVACY_NO_AUTH Нет аутентификации , без шифрования

NO_REPORT_NO_PRIVACY_BUT_AUTH_PROVIDED аутентификации , но без шифрования

NO_REPORT_PRIVACY_AND_AUTH_PROVIDED аутентификации и шифрования

INVALID_MSG если уровень безопасности не соответствует ни одному из вышеперечисленных

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

**10.15.4.34 Snmpv3Init Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

**void Snmpv3Init**();

**описание**

Эта процедура будет intialize глобальную базу данных SNMPv3 , двигатель ID и USM

**Замечания**

Ни один

**Предпосылки**

По крайней мере один UDP сокет должен быть доступен. UDPInit (см. стр. 621 ) ( ) уже называют .

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

**10.15.4.35 Snmpv3InitializeUserDataBase Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

**void Snmpv3InitializeUserDataBase**();

**описание**

Есть три Имя пользователя по умолчанию , аутентификация, authenticaton пароль и имя конфиденциальности и пароль конфиденциальности инициализация с SNMPv3 глобальной Databse .

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () вызывается .

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

**10.15.4.36 Snmpv3IsValidAuthStructure Функция**

**файл**

SNMPv3.c (см. стр. 440 )

C

BYTE **Snmpv3IsValidAuthStructure**(
WORD* **dataLen**

http://www.microchip.com/

);

**описание**

Эта процедура используется для проверки , является ли принятый VarBind имеет тип структуры (см. стр. 361 ) и выяснить, связывание переменных длина структуры .

**Замечания**

Ни один .

**Предпосылки**

ProcessHeader (см. стр. 368 ) () вызывается .

**Параметры**

**Параметры Описание**

**DataLen** Указатель памяти для хранения OID длину структуры .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**headrbytes** связывания переменной длины.

**ЛОЖЬ** Если переменная структура данных не тип структуры (см. стр. 361 ) .

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

**10.15.4.37 Snmpv3IsValidInt Функция**

**файл**

SNMPv3.c (см. стр. 440 )

C

BOOL **Snmpv3IsValidInt**(
DWORD* **val**
);

**описание**

Эта процедура заполняет и проверяет полученную переменную для типа данных как " ASN_INT (см. стр. 344 ) " и данных длина макс 4 байт. Это rotuine относится только к входящего запроса в SNMPv3 динамически выделенный буфер памяти ' gSNMPv3ScopedPduRequestBuf (см. стр. 423 ) .

**Замечания**

Ни один .

http://www.microchip.com/

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**Val** Указатель на память , где внутр значение переменной будет храниться .

**ReturnValues** верны - Если действует целый тип и значение будет получена.

**ЛОЖЬ** Кроме целое типа данных и значения получили.

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

**10.15.4.38 Snmpv3MsgProcessingModelProcessPDU Функция**

**файл**

SNMP.h (см. стр. 451 )

C

SNMP_ERR_STATUS **Snmpv3MsgProcessingModelProcessPDU**(
BYTE **inOutPdu**
);

**описание**

Recievd SNMPv3 PDU или заголовок передачи PDU имеет данные обработки сообщений байт infomration . Эта процедура retrievs обработка messgae модель infomration из хранимой PDU или написать соответствующий MSG Proc информация к repsonse MSG буфера .

**Замечания**

Параметры обработки messgae модели , как " MsgID ',' msgMaxSize ',' msgFlags 'и' msgSecurityModel " решает SNMPv3 условия обработки двигателя относительно запроса или ответа PDU

**Предпосылки**

Действительно запрос MSG SNMPv3 получен.

**Параметры**

**Параметры Описание**

inOutPdu указывает, является ли входящее PDU должен рассматриваться для значений Proc MSG быть получены илиPDU ответ должен быть заполнен с этими ценностями

**Возвращаемые значения**

**Возвращаемые значения Описание**

SNMP_NO_CREATION Отказ из-за неправильного формата информационной обработки MSG в принятом PDU или неудача в построении PDU ответа .

SNMP_NO_ERRобработки сообщений infomration поиска или ответ PDU fomration является успешный

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

**10.15.4.39 Snmpv3Notify Функция**

**файл**

SNMP.h (см. стр. 451 )

C

BOOL **Snmpv3Notify**(
SNMP_ID **var**,
SNMP_VAL **val**,
SNMP_INDEX **index**,
UINT8 **targetIndex**
);

**описание**

Эта функция создает SNMPv3 ловушки PDU и отправляет его в ранее указанном RemoteHost .

**Замечания**

Ни один

**Предпосылки**

TRAP события.

**Параметры**

**Параметры Описание**

**Var** SNMP вар идентификатор, который должен быть использован в уведомлении

**Val** Значение вар. Только стоимость BYTE , WORD или DWORD может быть отправлено.

**index** индекс вар. Если это переменная является единым , индекс будет 0 , либо , если это УАК последовательность , индекс может принимать любое значение от 0 до 127 targetIndex -индекса

Название « gSnmpv3TrapConfigData (см. стр. 424) " таблицы пользователь безопасности для который заголовок сообщения перехваты PDU для построено .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА**, если уведомление SNMP успешно отправляются. Это не гарантирует, что RemoteHost получил его .

**ЛОЖЬ** Уведомление отправлено не удалось.

Это не удастся при следующих contions 1) Учитывая SNMP_BIB_FILE не существует в MPFS 2 ) Учитывая переменная не существует.

3) Ранее учитывая agentID не существует

4) Тип данных данной вар неизвестно возможно, только если сам ГПСЛ был поврежден . SNMPV3_MSG_PRIV_FAIL

- шифрование ловушки MSG не удалось

SNMPV3_MSG_AUTH_FAIL HAMC ловушки MSG не удалось

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

**10.15.4.40 Snmpv3Pswd2LocalizedAuthKeyMD5Hashing Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
void Snmpv3Pswd2LocalizedAuthKeyMD5Hashing(
UINT8* pswdToLocalized,
UINT8 pswdLen
);
```

**описание**

Эта процедура преобразует HMAC- MD5 ключ пароль аутентификации для локализованной ключа, используя snmpSngineID ( RFC- 3414 ) .

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Параметры**

**Параметры Описание**

**pswdToLocalized** poniter хранения паролей

**pswdLen** длина пароля .

http://www.microchip.com/

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

**10.15.4.41 Snmpv3Pswd2LocalizedAuthKeySHAHashing Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
void Snmpv3Pswd2LocalizedAuthKeySHAHashing(
UINT8* pswdToLocalized,
UINT8 pswdLen
);
```

**описание**

Эта процедура преобразует HMAC- SHA ключ пароль аутентификации для локализованной ключа, используя snmpSngineID ( RFC- 3414 ) .

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Параметры**

**Параметры Описание**

**pswdToLocalized** poniter хранения паролей

**pswdLen** длина пароля .

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

**10.15.4.42 Snmpv3ReportPdu Функция**

**файл**

SNMPv3.c (см. стр. 440 )

C

```
void Snmpv3ReportPdu(
SNMPV3MSGDATA * dynScopedBufPtr
);
```

**описание**

Обмен SNMPv3 PDU начинается с агентом отправки отчета PDU на получение любой Get_Request PDU для SNMPv3 просьба. Эта процедура Froms отчета PDU для ответа на запрашивающего объекта .

**Замечания**

Ни один .

**Предпосылки**

ProcessVariables ( см. стр. 369 ) () вызывается и действительный запрос SNMPv3 получен.

**Параметры**

**Параметры Описание**

**dynScopedBufPtr** указатель в буферную память реагирования , где ' отчет ' ответ на быть savced для передачи.

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

**10.15.4.43 Snmpv3ScopedPduProcessing Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
SNMP_ERR_STATUS Snmpv3ScopedPduProcessing(
BYTE inOutPdu
);
```

**описание**

Recievd SNMPv3 PDU или заголовок передачи PDU имеет область видимости параметров PDU , как " contextEngineID ' ' имя контекста ' и т.д. Эта процедура retrievs эти параметры infomration из хранимой входящего PDU или написать соответствующий динамически выделенная память для PDU ответа передачи .

**Замечания**

Областью действия PDU параметры MSG данные : - [ ==

**Предпосылки**

Действительно запрос MSG SNMPv3 получен.

**Параметры**

**Параметры Описание**

http://www.microchip.com/

**inOutPdu** указывает, является ли входящее PDU должен рассматриваться для Scoped PDU paraemters к быть извлечены илиPDU ответ на заполняться с этими ценностями

**Возвращаемые значения**

**Возвращаемые значения Описание**

SNMP_NO_CREATION Отказ из-за неправильного Scoped PDU формате информации в PDU или неудачи в построения PDU ответа .

SNMP_NO_ERRобласти видимости параметры извлечения или ответ PDU fomration успешно

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

**10.15.4.44 Snmpv3SetErrorStatus Функция**

**файл**

SNMPv3.c (см. стр. 440 )

C

```
void Snmpv3SetErrorStatus(
WORD errorStatusOffset,
WORD errorIndexOffset,
SNMP_ERR_STATUS errorStatus,
BYTE errorIndex,
SNMPV3MSGDATA * dynScopedPduPutBuf
);
```

**Возвращает**

Ни один .

**описание**

Эта процедура обрабатывает принятый SNMP Получить запроса PDU для связывания переменной в запросе , а также создает ответ PDU.

**Замечания**

Ни один .

**Предпосылки**

**ProcessVariables** ( см. стр. 369 ) ( ) называется .

**Параметры**

**Параметры Описание**

**errorStatusOffset** Смещение обновить статус ошибки в ответ Tx PDU

**errorIndexOffset** Смещение обновить индекс ошибке

**ErrorStatus** Snmp технологическая ошибка в обновлении в ответ .

**ErrorIndex** Индекс запросу VarBind в списке переменная привязки , для которых статус ошибки должен быть обновляется.

**dynScopedPduPutBuf** динамичный SNMPv3 областью действия PDU буфер

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

**10.15.4.45 Snmpv3TrapScopedpdu Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
UINT8 Snmpv3TrapScopedpdu(
SNMP_ID var,
SNMP_VAL val,
SNMP_INDEX index,
UINT8 targetIndex
);
```

**описание**

Эта процедура формы ловушка области видимости заголовок PDU для ловушки PDU SNMPv3 быть возникла из этого агента . Scoped PDU состоит из MSG данных : - [ == Для ASCII STR ловушки VAL ( аргумента ) содержит адрес указателя переменной строки .

**Замечания**

Ни один .

**Предпосылки**

TRAP события.

**Параметры**

**Параметры Описание**

**Var** вар идентификатор переменной, значение для передачи в ловушки PDU

Значение вал переменной

индекс индекс переменной в множественной переменной сценария связывают targetIndex -индекса

Имя таблицы ' gSnmpv3TrapConfigData (см. стр. 424) ' Пользовательский безопасности для который заголовок сообщения перехваты PDU для построено .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** областью действия поколение заголовка PDU Ловушка успешно. ЛОЖЬ -ловушка области видимости поколение заголовка PDU не удалось.

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

**10.15.4.46 Snmpv3UserSecurityModelProcessPDU Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
SNMP_ERR_STATUS Snmpv3UserSecurityModelProcessPDU(
BYTE inOutPdu
);
```

**описание**

Recievd SNMPv3 PDU или заголовок передачи PDU имеет сообщение безопасности байт данных infomration . Эта процедура retrievs

Параметры сообщения безопасности infomration из хранимой входящего PDU или написать соответствующую модель безопасности информации к repsonse MSG буфера .

**Замечания**

Параметр безопасности пользователя составляют жизненно важную информацию для аутентификации сообщений и конфиденциальность сообщения.

Параметры модели безопасности пользователя заголовок структуру MsgAuthEngnID + MsgAuthEngnBoots + MsgAuthEngnTime + MsgUserName + MsgAuthParam + MsgPrivParam

**Предпосылки**

**Действительно запрос MSG SNMPv3 получен.**

**Параметры**

**Параметры Описание**

**inOutPdu** указывает, является ли входящее PDU должен рассматриваться для модели безопасности , чтобы пользователь получены илиPDU ответ на заполняться с этими ценностями

**Возвращаемые значения**

**Возвращаемые значения Описание**

SNMP_NO_CREATION Отказ из-за неправильного формата модель безопасности Обработка информации в получил PDU или неудачу в построении PDU ответа .

SNMP_NO_ERR поиска модель безопасности пользователя или ответ PDU fomration успешно

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

**10.15.4.47 Snmpv3UsmAesEncryptDecryptInitVector Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
void Snmpv3UsmAesEncryptDecryptInitVector(
BYTE inOutPdu
);
```

**описание**

IV объединяется как: 32-разрядных snmpEngineBoots (см. стр. 428 ) преобразуется в первые 4 Octects и **snmpEngineTime** (см. стр. 429 ) преобразуется в последующих четырех байтов .

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Параметры**

**Параметры Описание**

**inOutPdu** SNMP запрос PDU

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

**10.15.4.48 Snmpv3UsmOutMsgAuthenticationParam Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
void Snmpv3UsmOutMsgAuthenticationParam(
UINT8 hashType
);
```

**описание**

Эта процедура готовит из сообщение с HMAC- MD5 или протокола аутентификации HMAC- SHA1 . (RFC -3414 - раздел 6)

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Параметры**

**Параметры Описание**

Тип протокола аутентификации hashType

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

**10.15.4.49 Snmpv3USMOutMsgPrivParam Функция**

**файл**

SNMPv3.h (см. стр. 443 )

C

**void Snmpv3USMOutMsgPrivParam**();

**описание**

**SNMPEngineTime** используется для шифрования исходящего сообщения со случайным значением .

**Замечания**

Ни один

**Предпосылки**

**SNMPv3Init** () и **ProcessVariabels** ( ) называются .

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

**10.15.4.50 Snmpv3UsmSnmpEngnAuthPrivPswdLocalization**

**функция**

http://www.microchip.com/

**файл**

SNMPv3.h (см. стр. 443 )

C

```
void Snmpv3UsmSnmpEngnAuthPrivPswdLocalization(
UINT8 userDBIndex
);
```

**описание**

Эта процедура преобразует MD5 или SHA1 и AES конфиденциальности ключ пароль к локализованной ключа, используя snmpSngineID ( RFC- 3414 -раздел 6) .

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Параметры**

**Параметры Описание**

**userDBIndex** тип протокола аутентификации

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

**10.15.4.51 Snmpv3ValidateEngineId Функция**

**файл**

SNMP.h (см. стр. 451 )

C

```
BOOL Snmpv3ValidateEngineId();
```

**описание**

Эта процедура проверяет двигателя ID .

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

http://www.microchip.com/

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ЛОЖЬ** Невыполнение

**TRUE** Истинный успех

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

**10.15.4.52 Snmpv3ValidateSecNameAndSecLvl Функция**

**файл**

SNMP.h (см. стр. 451 )

C

BOOL **Snmpv3ValidateSecNameAndSecLvl**();

**описание**

Эта процедура проверяет имя secuirity и уровень Secuirty с SNMP глобальной базы данных для входящего PDU.

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**TRUE** Истинный успех

**ЛОЖЬ** провал

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

**10.15.4.53 Snmpv3ValidateSecurityName Функция**

**файл**

SNMP.h (см. стр. 451 )

C

http://www.microchip.com/

BOOL **Snmpv3ValidateSecurityName**();

**описание**

Эта процедура проверяет имя пользователя.

**Замечания**

Ни один

**Предпосылки**

SNMPv3Init () и ProcessVariabels ( ) называются .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ЛОЖЬ** Невыполнение

Истинный успех

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

**10.15.5 Структуры , Отчеты , Перечисления**

**модуль**

SNMP (см. стр. 318 )

структуры

**Имя Описание**

**AccessCtrlSubSysIsAccessAllowed** (см. стр. 403) Приложения являются типичными клиентами услуги (услуг) о доступе Подсистема управления . В следующем примитив обеспечивается Контроль доступа Подсистема для проверки доступ разрешен :

statusInformation (см. стр. 417 ) = - успех или errorIndication

**dispatcherProcessPdu** (см. стр. 404) Process поступления запроса или уведомления PDU Диспетчер предоставляет следующие примитивны, чтобы пройти на входящий SNMP PDU в приложение.

**dispatcherStatusInfo** (см. стр. 404 ) Создание исходящий запрос или уведомление

**statusInformation** (см. стр. 417 ) = - sendPduHandle если успех - errorIndication если отказ

**dispathcerReturnResponsePdu** (см. страница 405 ) Генерация исходящих ответов PDU Диспетчер предоставляет следующие примитив для Приложение вернуть SNMP Response PDU в PDU Диспетчер:

результат = успех или неудача

**MsgProcModPrepareDataElements** ( см. стр. 406) Подготовка элементов данных из входящего сообщения SNMP Сообщение Обработка Подсистема предоставляет эту услугу примитивный для подготовки абстрактные элементы данных из входящего SNMP сообщение : результат = - успех или errorIndication

**MsgProcModPrepareOutgoingMessage**(См. стр. 407 ) Подготовка исходящей SNMP запроса или уведомления сообщение Сообщение Обработка Подсистема предоставляет эту услугу примитивный для подготовки исходящего SNMP запроса или уведомления сообщение

**MsgProcModPrepareResponseMessage** (См. стр. 407 ) Подготовка исходящего SNMP ответное сообщение Сообщение Обработка Подсистема предоставляет эту услугу примитивный для подготовки исходящего SNMP ответное сообщение : результат = - Успех или неудача

**processResponsePdu** (см. стр. 408) обрабатывать входящие PDU отклика PDU Диспетчер предоставляет следующие примитивны, чтобы пройти входящие SNMP отклика PDU к приложению :

**registerContextEngineID** (см. стр. 417) Успех или errorIndication

**SecuritySysGenerateRequestMsg** (см. стр. 409) Это запись SecuritySysGenerateRequestMsg .

**SecuritySysGenerateResponseMsg** (см. стр. 410) Генерирует ответное сообщение, Подсистема защиты предоставляет следующие примитивны, чтобы генерировать Ответное сообщение :

**StateRelease** (см. стр. 410 ) Государственный выпуска справочная информация Все подсистемы , которые проходят stateReference информацию также обеспечивают примитивный для освобождения памяти , которая содержит ссылается состояние информация

**unregisterContextEngineID** (см. стр.411 ) Это запись unregisterContextEngineID .

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

**10.15.5.1 AccessCtrlSubSysIsAccessAllowed Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
struct AccessCtrlSubSysIsAccessAllowed {
BYTE securityModel;
BYTE* securityName;
BYTE securityLevel;
BYTE viewType;
BYTE* contextName;
BYTE* variableName;
};
```

**Пользователи**

**Пользователи Описание**

BYTE **securityModel** ; Модель безопасности в использовании

BYTE * **SecurityName** ; Основным кто хочет получить доступ к

BYTE **securityLevel** ; Уровень безопасности

BYTE **ViewType** ; читать, писать , или уведомить вид

BYTE * **contextName** ; контекст , содержащий VariableName

BYTE * имяПеременной ; OID для управляемого объекта

**описание**

Приложения являются типичными клиентами услуги (услуг) на подсистема контроля доступа . В следующем примитив обеспечивается

Контроль доступа Подсистема для проверки доступ разрешен :

**statusInformation** (см. стр. 417 ) = - успех или errorIndication

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

**10.15.5.2 dispatcherProcessPdu Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
struct dispatcherProcessPdu {
BYTE messageProcessingModel;
BYTE securityModel;
BYTE* securityName;
BYTE securityLevel;
BYTE* contextEngineID;
BYTE* contextName;
BYTE pduVersion;
BYTE* PDU;
DWORD_VAL maxSizeResponseScopedPDU;
UINT32 stateReference;
};
```

**Пользователи**

**Пользователи Описание**

BYTE **messageProcessingModel** ; обычно , SNMP версии

BYTE **securityModel** ; Модель безопасности в использовании

BYTE * **SecurityName** ; от имени принципала

BYTE **securityLevel** ; Уровень безопасности

BYTE * **contextEngineID** ; данных из / в этом SNMP лица

BYTE * **contextName** ; данные из / в этом контексте

BYTE **pduVersion** ; версия PDU

BYTE * **PDU** ; Протокол SNMP блок данных DWORD_VAL

**maxSizeResponseScopedPDU** ; Максимальный размер PDU реагирования

UINT32 **stateReference** ; ссылка на информацию о состоянии необходимы при отправке ответа

**описание**

Процесс входящий запрос или уведомление PDU Диспетчер предоставляет следующие примитивны, чтобы пройти на входящий SNMP PDU в приложение.

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

**10.15.5.3 dispatcherStatusInfo Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
struct dispatcherStatusInfo {
BYTE transportDomain;
UINT32 transportAddress;
BYTE messageProcessingModel;
BYTE securityModel;
BYTE* securityName;
BYTE securityLevel;
BYTE* contextEngineID;
BYTE* contextName;
BYTE pduVersion;
BYTE* PDU;
BOOL expectResponse;
};
```

**Пользователи**

**Пользователи Описание**

BYTE **transportDomain** ; домена транспорта , который будет использоваться

UINT32 **transportAddress** ; транспортный адрес , который будет использоваться

BYTE **messageProcessingModel** ; обычно , SNMP версии

BYTE **securityModel** ; Модель безопасности в использовании

BYTE * **SecurityName** ; от имени принципала

BYTE **securityLevel** ; Уровень безопасности запрашиваемого

BYTE * **contextEngineID** ; данных из / в этом объекте

BYTE * **contextName** ; данные из / в этом контексте

BYTE **pduVersion** ; версия PDU

BYTE * **PDU** ; Протокол SNMP блок данных

BOOL **expectResponse** ; ИСТИНА или ЛОЖЬ

**описание**

Создать исходящий запрос или уведомление

**statusInformation** (см. стр. 417 ) = - sendPduHandle если успех - errorIndication если отказ

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

**10.15.5.4 dispathcerReturnResponsePdu Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
struct dispathcerReturnResponsePdu {
BYTE messageProcessingModel;
BYTE securityModel;
BYTE* securityName;
BYTE securityLevel;
BYTE* contextEngineID;
BYTE* contextName;
BYTE pduVersion;
BYTE* PDU;
UINT32 maxSizeResponseScopedPDU;
UINT32 stateReference;
statusInformation statInfo;
};
```

**Пользователи**

**Пользователи Описание**

BYTE **messageProcessingModel** ; обычно , SNMP версии

BYTE **securityModel** ; Модель безопасности в использовании

BYTE * **SecurityName** ; от имени принципала

BYTE **securityLevel** ; же, как на входящем запросе

BYTE * **contextEngineID** ; данных из / в этом SNMP лица

BYTE * **contextName** ; данные из / в этом контексте

BYTE **pduVersion** ; версия PDU

BYTE * **PDU** ; Протокол SNMP блок данных

UINT32 **maxSizeResponseScopedPDU** ; Максимальная отправитель размер может принять (см. стр. 169 )

UINT32 **stateReference** ; ссылка на информацию о состоянии , как представлено с просьбой

**statusInformation** statInfo ; Успех или errorIndication , счетчик ошибок OID / значение, если ошибка

**описание**

Генерация исходящих ответов PDU Диспетчер предоставляет следующие примитивно для приложения , чтобы вернуть SNMP Response PDU в PDU Диспетчер: результат = успех или неудача

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

**10.15.5.5 MsgProcModPrepareDataElements Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
struct MsgProcModPrepareDataElements {
BYTE transportDomain;
UINT32 transportAddress;
BYTE* wholeMsg;
UINT32 wholeMsgLength;
BYTE messageProcessingModel;
BYTE securityModel;
BYTE* securityName;
BYTE securityLevel;
BYTE* contextEngineID;
BYTE* contextName;
BYTE pduVersion;
BYTE* PDU;
BYTE pduType;
UINT32 maxSizeResponseScopedPDU;
statusInformation statInfo;
UINT32 stateReference;
};
```

http://www.microchip.com/

**Пользователи**

**Пользователи Описание**

BYTE **transportDomain** ; Происхождение транспорт домена

UINT32 **transportAddress** ; Происхождение транспортный адрес

BYTE * **wholeMsg** ; как принято от сети

UINT32 **wholeMsgLength** ; как принято от сети

BYTE **messageProcessingModel** ; обычно , SNMP версии

BYTE **securityModel** ; Модель безопасности в использовании

BYTE * **SecurityName** ; от имени принципала

BYTE **securityLevel** ; Уровень безопасности запрашиваемого

BYTE * **contextEngineID** ; данных из / в этом объекте

BYTE * **contextName** ; данные из / в этом контексте

BYTE **pduVersion** ; версия PDU

BYTE * **PDU** ; Протокол SNMP блок данных

BYTE **pduType** ; SNMP PDU введите OUT sendPduHandle ; / / Дескриптор для согласованного запросу

UINT32 **maxSizeResponseScopedPDU** ; Максимальная отправитель размер может принять (см. стр. 169 )

**statusInformation** statInfo ; Успех или счетчик ошибок errorIndication OID / значение, если ошибка

UINT32 **stateReference** ; ссылаться на информацию о состоянии , которые будут использоваться для возможного реагирования

**описание**

Подготовка элементов данных из входящего сообщения SNMP Сообщение Обработка Подсистема предоставляет эту услугу примитив для подготовки абстрактные элементы данных из входящее сообщение SNMP : результат = - успех или errorIndication

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

**10.15.5.6 MsgProcModPrepareOutgoingMessage Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

http://www.microchip.com/

```
struct MsgProcModPrepareOutgoingMessage {
BYTE transportDomain;
UINT32 transportAddress;
BYTE messageProcessingModel;
BYTE securityModel;
BYTE* securityName;
BYTE securityLevel;
BYTE* contextEngineID;
BYTE* contextName;
BYTE pduVersion;
BYTE* PDU;
BOOL expectResponse;
BYTE destTransportDomain;
UINT32 destTransportAddress;
BYTE* outgoingMessage;
UINT32 outgoingMessageLength;
};
```

**Пользователи**

**Пользователи Описание**

BYTE **transportDomain** ; домена транспорта , который будет использоваться

UINT32 **transportAddress** ; транспортный адрес , который будет использоваться

BYTE **messageProcessingModel** ; обычно , SNMP версии

BYTE **securityModel** ; Модель безопасности в использовании

BYTE * **SecurityName** ; от имени принципала

BYTE **securityLevel** ; Уровень безопасности запрашиваемого

BYTE * **contextEngineID** ; данных из / в этом объекте

BYTE * **contextName** ; данные из / в этом контексте

BYTE **pduVersion** ; версия PDU

BYTE * **PDU** ; Протокол SNMP блок данных

BOOL **expectResponse** ; ИСТИНА или ЛОЖЬ В sendPduHandle ; / / ручка для согласования входящий звонок ответы

BYTE **destTransportDomain** ; назначения транспортного домена

UINT32 **destTransportAddress** ; назначения транспортный адрес

BYTE * **outgoingMessage** ; сообщение отправить

UINT32 **outgoingMessageLength** ; его длина

**описание**

http://www.microchip.com/

Подготовка исходящей SNMP запроса или уведомления сообщение Сообщение Обработка Подсистема предоставляет эту услугу примитивный для подготовки исходящего SNMP запрос или уведомление Сообщение

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

**10.15.5.7 MsgProcModPrepareResponseMessage Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
struct MsgProcModPrepareResponseMessage {
BYTE messageProcessingModel;
BYTE securityModel;
BYTE* securityName;
BYTE securityLevel;
BYTE* contextEngineID;
BYTE* contextName;
BYTE pduVersion;
BYTE* PDU;
UINT32 maxSizeResponseScopedPDU;
UINT32 stateReference;
statusInformation statInfo;
BYTE destTransportDomain;
UINT32 destTransportAddress;
BYTE* outgoingMessage;
UINT32 outgoingMessageLength;
};
```

**Пользователи**

**Пользователи Описание**

BYTE **messageProcessingModel** ; обычно , SNMP версии

BYTE **securityModel** ; же, как на входящем запросе

BYTE * **SecurityName** ; же, как на входящем запросе

BYTE **securityLevel** ; же, как на входящем запросе

BYTE * **contextEngineID** ; данных из / в этом SNMP лица

BYTE * **contextName** ; данные из / в этом контексте

BYTE **pduVersion** ; версия PDU

BYTE * **PDU** ; Протокол SNMP блок данных

UINT32 **maxSizeResponseScopedPDU** ; Максимальный размер в состоянии принять (см. стр. 169 )

UINT32 **stateReference** ; ссылка на информацию о состоянии , как представлено с просьбой

**statusInformation** statInfo ; Успех или errorIndication , счетчик ошибок OID / значение, если ошибка

BYTE **destTransportDomain** ; назначения транспортного домена

UINT32 **destTransportAddress** ; назначения транспортный адрес

BYTE * **outgoingMessage** ; сообщение отправить

UINT32 **outgoingMessageLength** ; его длина

**описание**

Подготовка исходящего SNMP ответное сообщение

Сообщение Обработка Подсистема предоставляет эту услугу примитивный для подготовки исходящего сообщения SNMP отклика: Результат = - успех или неудача

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

**10.15.5.8 processResponsePdu Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
struct processResponsePdu {
BYTE messageProcessingModel;
BYTE securityModel;
BYTE* securityName;
BYTE securityLevel;
BYTE* contextEngineID;
BYTE* contextName;
BYTE pduVersion;
BYTE* PDU;
statusInformation statInfo;
};
```

**Пользователи**

**Пользователи Описание**

BYTE **messageProcessingModel** ; обычно , SNMP версии

BYTE **securityModel** ; Модель безопасности в использовании

BYTE * **SecurityName** ; от имени принципала

BYTE **securityLevel** ; Уровень безопасности

BYTE * **contextEngineID** ; данных из / в этом SNMP лица

BYTE * **contextName** ; данные из / в этом контексте

BYTE **pduVersion** ; версия PDU

BYTE * **PDU** ; Протокол SNMP блок данных

**statusInformation** statInfo ; Успех или errorIndication В sendPduHandle ; / / дескриптор от sendPdu

**описание**

Процесс поступления PDU отклика PDU Диспетчер предоставляет следующие примитивны, чтобы пройти на входящий SNMP Response PDU к приложению :

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

**10.15.5.9 SecuritySysGenerateRequestMsg Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
struct SecuritySysGenerateRequestMsg {
BYTE messageProcessingModel;
BYTE* globalData;
UINT32 maxMessageSize;
BYTE securityModel;
BYTE* securityEngineID;
BYTE* securityName;
BYTE securityLevel;
BYTE* scopedPDU;
BYTE* wholeMsg;
UINT32 wholeMsgLength;
};
```

**Пользователи**

**Пользователи Описание**

BYTE **messageProcessingModel** ; обычно , SNMP версии

BYTE * ОБЩАЯ ; Заголовок сообщения , администратор данных

UINT32 **MaxMessageSize** ; представляемого SNMP лица

BYTE **securityModel** ; для исходящего сообщения

BYTE * **securityEngineID** ; авторитетным SNMP лицо

BYTE * **SecurityName** ; от имени принципала

BYTE **securityLevel** ; Уровень безопасности запрашиваемого

BYTE * **scopedPDU** ; сообщение ( открытого текста ) полезной нагрузки OUT securityParameters ; / / заполняется безопасности модуль

BYTE * **wholeMsg** ; завершить генерируется сообщение

UINT32 **wholeMsgLength** ; длина сгенерированных сообщений

**описание**

Это запись **SecuritySysGenerateRequestMsg** .

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

**10.15.5.10 SecuritySysGenerateResponseMsg Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
struct SecuritySysGenerateResponseMsg {
BYTE messageProcessingModel;
BYTE* globalData;
UINT32 maxMessageSize;
BYTE securityModel;
BYTE* securityEngineID;
BYTE* securityName;
BYTE securityLevel;
BYTE* scopedPDU;
BYTE* wholeMsg;
UINT32 wholeMsgLength;
};
```

**Пользователи**

**Пользователи Описание**

BYTE **messageProcessingModel** ; обычно , SNMP версии

BYTE * ОБЩАЯ ; Заголовок сообщения , администратор данных

UINT32 **MaxMessageSize** ; представляемого SNMP лица

BYTE **securityModel** ; для исходящего сообщения

BYTE * **securityEngineID** ; авторитетным SNMP лицо

BYTE * **SecurityName** ; от имени принципала

BYTE **securityLevel** ; для исходящего сообщения

BYTE * **scopedPDU** ; сообщение ( открытый текст ) полезная нагрузка В securityStateReference ; / / ссылка на безопасности Информация о состоянии от оригинала запросу OUT securityParameters ; / / заполняется Модуль безопасности

BYTE * **wholeMsg** ; завершить генерируется сообщение

UINT32 **wholeMsgLength** ; длина сгенерированных сообщений

**описание**

Генерирует ответное сообщение, Подсистема защиты предоставляет следующие примитивны, чтобы генерировать ответное сообщение :

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

**10.15.5.11 StateRelease Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
struct StateRelease {
UINT32 stateReference;
};
```

**Пользователи**

**Пользователи Описание**

UINT32 **stateReference** ; Ручка ведения, который будет выпущен

**описание**

Государственный выпуска Справочная информация Все подсистемы , которые проходят stateReference информацию также обеспечивают примитивный , чтобы освободить память , которая содержит ссылается информация о состоянии

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

**10.15.5.12 unregisterContextEngineID Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
struct unregisterContextEngineID {
BYTE* contextEngineID;
BYTE pduType;
};
```

**Пользователи**

**Пользователи Описание**

BYTE * **contextEngineID** ; отказаться от ответственности за этот

BYTE **pduType** ; pduType (ы) , чтобы быть незарегистрированные

**описание**

Это запись **unregisterContextEngineID** .

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

## 10.15.6 Типы

**Перечисления**

**Имя Описание**

**INOUT_SNMP_PDU** (см. стр. 412 ) Это типа INOUT_SNMP_PDU .

**REPORT_FLAG_AND_SECURITY_LEVEL_FLAGS** (см. стр. 412 ) Это тип

REPORT_FLAG_AND_SECURITY_LEVEL_FLAGS.

**SNMP_ENGNID_OCTET_IDENTIFIER_VAL** (см. стр. 414) Пятый октет указывает, как остальные ( 6-й и Следующие октета) отформатированы . См. RFC3411 section5 Страница № 41

**SNMPV3_HMAC_HASH_TYPE** (см. стр. 414 ) Тип хэша рассчитывается

**SNMPV3_MSG_AUTH_SEC_PARAM_RESULT** (см.страница 415) Это тип

SNMPV3_MSG_AUTH_SEC_PARAM_RESULT .

**SNMPV3_MSG_PRIV_SEC_PARAM_RESULT** (см.страница 415) Это тип

SNMPV3_MSG_PRIV_SEC_PARAM_RESULT .

**SNMPV3_PRIV_PROT_TYPE** (см. стр. 415 ) Это тип SNMPV3_PRIV_PROT_TYPE .

**STD_BASED_SNMP_MESSAGE_PROCESSING_MODEL** (См. стр. 418 ) Это тип

STD_BASED_SNMP_MESSAGE_PROCESSING_model .

**STD_BASED_SNMP_SECURITY_MODEL** (см. стр.418) Snmp Сообщение Модель обработки

**STD_BASED_SNMPV3_SECURITY_LEVEL** (см. стр. 418) Это тип

STD_BASED_SNMPV3_SECURITY_LEVEL .

**USM_SECURITY_LEVEL** (см. стр. 419 ) Это типа USM_SECURITY_LEVEL .

модуль **SNMP** (см. стр. 318 )

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

**структуры**

**Имя Описание**

**SecuritySysProcessIncomingMsg** (см. стр. 413) Это тип SecuritySysProcessIncomingMsg .

**SNMPNONMIBRECDINFO** (см.страница 414 ) Это тип SNMPNONMIBRECDINFO .

**SNMPV3_REQUEST_WHOLEMSG**(См. стр. 416 ) Это тип SNMPV3_REQUEST_WHOLEMSG .

**SNMPV3_RESPONSE_WHOLEMSG**(См. стр. 416 ) Это тип SNMPV3_RESPONSE_WHOLEMSG .

**snmpV3EngnUserDataBase** (см.страница 416) Это тип snmpV3EngnUserDataBase .

**SNMPV3MSGDATA** (см. стр.417) SNMPv3

**snmpV3TrapConfigDataBase** (см.страница 417 ) целевой конфигурации SNMPv3 по отношению к ловушке.

**statusInformation** ( см. стр. 417), успех или errorIndication

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

**10.15.6.1 INOUT_SNMP_PDU Перечисление**

**файл**

SNMP.h (см. стр. 451 )

C

```
typedef enum {
SNMP_RESPONSE_PDU = 0x01,
SNMP_REQUEST_PDU = 0x02
} INOUT_SNMP_PDU;
```

**описание**

Это тип INOUT_SNMP_PDU .

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

**10.15.6.2 REPORT_FLAG_AND_SECURITY_LEVEL_FLAGS**

**перечисление**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef enum {
NO_REPORT_NO_PRIVACY_NO_AUTH = 0x00,
NO_REPORT_NO_PRIVACY_BUT_AUTH_PROVIDED = 0x01,
NO_REPORT_PRIVACY_PROVIDED_BUT_NO_AUTH = 0x02,
NO_REPORT_PRIVACY_AND_AUTH_PROVIDED = 0x03,
REPORT2REQ_NO_PRIVACY_NO_AUTH = 0x04,
REPORT2REQ_NO_PRIVACY_BUT_AUTH_PROVIDED = 0x05,
REPORT2REQ_PRIVACY_PROVIDED_BUT_NO_AUTH = 0x06,
REPORT2REQ_PRIVACY_AND_AUTH_PROVIDED = 0x07,
INVALID_MSG = 0xFF
} REPORT_FLAG_AND_SECURITY_LEVEL_FLAGS;
```

**Пользователи**

**Пользователи Описание**

NO_REPORT_NO_PRIVACY_NO_AUTH = 0x00 00000000b

NO_REPORT_NO_PRIVACY_BUT_AUTH_PROVIDED= 0x01 00000001b

NO_REPORT_PRIVACY_PROVIDED_BUT_NO_AUTH = 0x02 00000010b Приват без Auth не допускается

NO_REPORT_PRIVACY_AND_AUTH_PROVIDED =0x03 00000011b

REPORT2REQ_NO_PRIVACY_NO_AUTH = 0x04 00000100b

REPORT2REQ_NO_PRIVACY_BUT_AUTH_PROVIDED = 0x05 00000101b

REPORT2REQ_PRIVACY_PROVIDED_BUT_NO_AUTH = 0x06 00000110b Приват без Auth не допускается

REPORT2REQ_PRIVACY_AND_AUTH_PROVIDED = 0x07 00000111b

**описание**

Это тип REPORT_FLAG_AND_SECURITY_LEVEL_FLAGS .

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

**10.15.6.3 SecuritySysProcessIncomingMsg Структура**

**файл**

http://www.microchip.com/

SNMPv3.h (см. стр. 443 )

C

```
typedef struct {
UINT32 maxMessageSize;
UINT32 wholeMsgLength;
BYTE* wholeMsg;
BYTE* securityEngineID;
BYTE* securityName;
BYTE* scopedPDU;
UINT32 maxSizeResponseScopedPDU;
BYTE messageProcessingModel;
BYTE securityModel;
BYTE securityLevel;
UINT8 securityEngineIDLen;
BYTE securityNameLength;
} SecuritySysProcessIncomingMsg;
```

**Пользователи**

**Пользователи Описание**

UINT32 **MaxMessageSize** ; представляемого SNMP лица в securityParameters ; / / для принятого сообщения

UINT32 **wholeMsgLength** ; длина в полученном на проводе

BYTE * **wholeMsg** ; в полученном на проводе

BYTE * **securityEngineID** ; авторитетным SNMP лицо

BYTE * **SecurityName** ; выявление основных

BYTE * **scopedPDU** ; сообщение ( открытый текст ) полезная нагрузка OUT securityStateReference ; / / ссылка на госбезопасность

UINT32 **maxSizeResponseScopedPDU** ; Максимальная отправитель размер может обрабатывать

BYTE **messageProcessingModel** ; обычно , SNMP версии

BYTE **securityModel** ; для принятого сообщения

BYTE **securityLevel** ; Уровень безопасности

UINT8 **securityEngineIDLen** ; авторитетным SNMP лицо

**описание**

Это тип SecuritySysProcessIncomingMsg .

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

http://www.microchip.com/

**10.15.6.4 SNMP_ENGNID_OCTET_IDENTIFIER_VAL Перечисление**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef enum {
RESERVED = 0x0,
IPV4_ADDR_ENGN_ID = 0x01,
IPV6_ADDR_ENGN_ID = 0x02,
MAC_ADDR_ENGN_ID = 0x03,
ADMIN_ASSIGNED_TEXT = 0x04,
ADMIN_ASSIGNED_OCTETS = 0x05,
RESERVED_UNUSED = 0x06,
ENTERPRISE_DEFINED = 128
} SNMP_ENGNID_OCTET_IDENTIFIER_VAL;
```

**Пользователи**

**Пользователи Описание**

IPV4_ADDR_ENGN_ID = 0x01 4octets

IPV6_ADDR_ENGN_ID = 0x02 16 октета

MAC_ADDR_ENGN_ID = 0x03 6 октетов

RESERVED_UNUSED = 0x06 6 до 127 зарезервированы и не используется

ENTERPRISE_DEFINED = 128 128 до 255 , как defiend максимальным предприятие оставшейся длины

**описание**

Пятый октет указывает, как остальные ( 6-я и следующие октета) отформатированы . См. RFC3411 section5 странице # 41

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

**10.15.6.5 SNMPNONMIBRECDINFO Структура**

**файл**

SNMP.h (см. стр. 451 )

C

```
typedef struct {
UINT8 oidstr[16];
UINT8 version;
} SNMPNONMIBRECDINFO;
```

http://www.microchip.com/

**описание**

Это тип SNMPNONMIBRECDINFO .

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

**10.15.6.6 SNMPV3_HMAC_HASH_TYPE Перечисление**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef enum {
SNMPV3_HAMC_MD5 = 0u,
SNMPV3_HMAC_SHA1,
SNMPV3_NO_HMAC_AUTH
} SNMPV3_HMAC_HASH_TYPE;
```

**Пользователи**

**Пользователи Описание**

SNMPV3_HAMC_MD5 = 0u MD5 вычисляется

SNMPV3_HMAC_SHA1 SHA-1 в процессе расчета

**описание**

Тип хэш вычисляется

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

**10.15.6.7 SNMPV3_MSG_AUTH_SEC_PARAM_RESULT**

**перечисление**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef enum {
SNMPV3_MSG_AUTH_FAIL = 0x00,
SNMPV3_MSG_AUTH_PASS = 0x01
} SNMPV3_MSG_AUTH_SEC_PARAM_RESULT;
```

**описание**

http://www.microchip.com/

Это тип SNMPV3_MSG_AUTH_SEC_PARAM_RESULT .

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

**10.15.6.8 SNMPV3_MSG_PRIV_SEC_PARAM_RESULT**

**перечисление**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef enum {
SNMPV3_MSG_PRIV_FAIL = 0x00,
SNMPV3_MSG_PRIV_PASS = 0x01
} SNMPV3_MSG_PRIV_SEC_PARAM_RESULT;
```

**описание**

Это тип SNMPV3_MSG_PRIV_SEC_PARAM_RESULT .

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

**10.15.6.9 SNMPV3_PRIV_PROT_TYPE Перечисление**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef enum {
SNMPV3_DES_PRIV = 0x0,
SNMPV3_AES_PRIV,
SNMPV3_NO_PRIV
} SNMPV3_PRIV_PROT_TYPE;
```

**описание**

Это тип SNMPV3_PRIV_PROT_TYPE .

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

**10.15.6.10 SNMPV3_REQUEST_WHOLEMSG Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef struct {
UINT8* wholeMsgHead;
UINT8* snmpMsgHead;
WORD_VAL wholeMsgLen;
WORD_VAL snmpMsgLen;
WORD msgAuthParamOffsetInWholeMsg;
WORD scopedPduOffset;
BYTE scopedPduAuthStructVal;
WORD scopedPduStructLen;
} SNMPV3_REQUEST_WHOLEMSG;
```

**описание**

Это тип SNMPV3_REQUEST_WHOLEMSG .

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

### 10.15.6.11 SNMPV3_RESPONSE_WHOLEMSG Структура

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef struct {
UINT8* wholeMsgHead;
UINT8* snmpMsgHead;
WORD_VAL wholeMsgLen;
WORD_VAL snmpMsgLen;
UINT8* msgAuthParamOffsetOutWholeMsg;
UINT8* scopedPduOffset;
WORD scopedPduStructLen;
BYTE scopedPduAuthStructVal;
} SNMPV3_RESPONSE_WHOLEMSG;
```

**описание**

Это тип SNMPV3_RESPONSE_WHOLEMSG .

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

### 10.15.6.12 snmpV3EngnUserDataBase Структура

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef struct {
UINT8 userName[SNMPV3_USER_SECURITY_NAME_LEN_MEM_USE];
UINT8 userAuthPswd[SNMPV3_AUTH_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE];
UINT8 userPrivPswd[SNMPV3_PRIV_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE];
UINT8 userAuthPswdLoclizdKey[SNMPV3_AUTH_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE];
UINT8 userPrivPswdLoclizdKey[SNMPV3_PRIV_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE];
UINT8 userAuthLocalKeyHmacIpad[64];
UINT8 userAuthLocalKeyHmacOpad[64];
UINT8 userDBIndex;
UINT8 userHashType;
UINT8 userNameLength;
UINT8 userAuthPswdLen;
UINT8 userPrivPswdLen;
UINT8 userPrivType;
} snmpV3EngnUserDataBase;
```

**Пользователи**

**Пользователи Описание**

UINT8

**userAuthPswd** [ SNMPV3_AUTH_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE ] ;

RFC определяет не экономить пароль с управляемыми узлами вместо магазин PSWD IPad и значения Opad .

UINT8 **userAuthLocalKeyHmacIpad** [64] ; UINT8 userPrivPswdLoclizdKey [20] ;

**описание**

Это тип **snmpV3EngnUserDataBase** .

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

**10.15.6.13 SNMPV3MSGDATA Структура**

**файл**

SNMP.h (см. стр. 451 )

C

```
typedef struct {
UINT8 * head;
WORD length;
WORD maxlength;
WORD msgAuthParamOffset;
```

http://www.microchip.com/

} SNMPV3MSGDATA;

**описание**

SNMPv3

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

**10.15.6.14 snmpV3TrapConfigDataBase Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef struct {
UINT8 userSecurityName[USER_SECURITY_NAME_LEN];
STD_BASED_SNMP_MESSAGE_PROCESSING_MODEL messageProcessingModelType;
STD_BASED_SNMP_SECURITY_MODEL securityModelType;
STD_BASED_SNMPV3_SECURITY_LEVEL securityLevelType;
} snmpV3TrapConfigDataBase;
```

**описание**

целевой конфигурации SNMPv3 по отношению к ловушке.

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

**10.15.6.15 statusInformation Структура**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef struct registerContextEngineID {
BYTE* contextEngineID;
BYTE pduType;
} statusInformation;
```

**Пользователи**

**Пользователи Описание**

BYTE * **contextEngineID** ; взять ответственность за этот

BYTE **pduType** ; pduType (ы) должны быть зарегистрированы

http://www.microchip.com/

**описание**

Успех или errorIndication

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

**10.15.6.16 STD_BASED_SNMP_MESSAGE_PROCESSING_MODEL**

**перечисление**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef enum {
SNMPV1_MSG_PROCESSING_MODEL = 0X00,
SNMPV2C_MSG_PROCESSING_MODEL = 0X01,
SNMPV2U_V2_MSG_PROCESSING_MODEL = 0X02,
SNMPV3_MSG_PROCESSING_MODEL = 0X03
} STD_BASED_SNMP_MESSAGE_PROCESSING_MODEL;
```

**описание**

Это тип STD_BASED_SNMP_MESSAGE_PROCESSING_MODEL .

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

**10.15.6.17 STD_BASED_SNMP_SECURITY_MODEL Перечисление**

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef enum {
ANY_SECUTIRY_MODEL = 0x00,
SNMPV1_SECURITY_MODEL = 0X01,
SNMPV2C_SECURITY_MODEL = 0X02,
SNMPV3_USM_SECURITY_MODEL = 0X03
} STD_BASED_SNMP_SECURITY_MODEL;
```

**описание**

Snmp Сообщение Модель обработки

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

#### 10.15.6.18 STD_BASED_SNMPV3_SECURITY_LEVEL Перечисление

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef enum {
NO_AUTH_NO_PRIV = 1,
AUTH_NO_PRIV,
AUTH_PRIV
} STD_BASED_SNMPV3_SECURITY_LEVEL;
```

**описание**

Это тип STD_BASED_SNMPV3_SECURITY_LEVEL .

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

#### 10.15.6.19 USM_SECURITY_LEVEL Перечисление

**файл**

SNMPv3.h (см. стр. 443 )

C

```
typedef enum {
noAuthProtocol = 0x1,
hmacMD5Auth,
hmacSHAAuth,
noPrivProtocol,
desPrivProtocol = 0x5,
aesPrivProtocol = 0x6
} USM_SECURITY_LEVEL;
```

**описание**

Это тип USM_SECURITY_LEVEL .

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

### 10.15.7 Переменные

**модуль**

SNMP (см. стр. 318 )

**Переменные**

**Имя Описание**

**authKey_iPad** (см. стр. 421 ) Это переменная authKey_iPad .

**authKey_oPad** (см. стр. 421 ) Это переменная authKey_oPad .

**authoritativeSnmpEngineBoots** (см.страница 421 ) Количество раз , что авторитетный двигатель SNMP имеет ( пере-) инициализации себя с момента ее snmpEngineID (см. стр. 428 ) было последней конфигурации .

**authoritativeSnmpEngineTime** (см.страница 422 ) Так как значениеКоличество секунд

**authoritativeSnmpEngineBoots** ( см. стр. 421 ) объекта последнего изменения

**cipher_text** (см. стр. 422 ) Это переменная cipher_text .

**deciphered_text** (см. стр. 422 ) Это переменная deciphered_text .

**getZeroInstance** (см. стр. 422 ) Эта переменная используется для gext следующего запроса для нулевой например

**gSnmpV3InPduWholeMsgBuf** (см.страница 422 ) Заглушки Динамическая память и PDU данные входящей хранится PDU

**gSnmpV3OUTPduWholeMsgBuf** (см.страница 423) Подробнее заглушки динамической памяти и построена исходящих сохраняются PDU

детали

**gSNMPv3PduHeaderBuf** (см. стр.423) Строительство PDU отклика смещение детали

**gSNMPv3ScopedPduDataPos** (см.страница 423) Смещение читать Scoped байтов данных PDU для обработки от динамического заглушки памяти

**gSNMPv3ScopedPduRequestBuf** (см. стр. 423 ) Хранится запрос областью видимости детали PDU

**gSNMPv3ScopedPduResponseBuf** (см. стр. 424) Обработанные ответ областью видимости детали PDU

**gSnmpv3TrapConfigData** (см. стр.424) Глобальная база данных конфигурации SNMPv3 , который будет использоваться для уведомления ловушки

**gSNMPv3TrapMsgHeaderBuf** (см.страница 424 ) TRAP сообщение строительство заголовка PDU смещение детали

**gSnmpV3TrapOUTPduWholeMsgBuf** (См. стр. 424 ) Подробнее заглушки динамической памяти и построенные ловушки подробнее PDU

**gSNMPv3TrapScopedPduResponseBuf** (См. стр. 424 ) TRAP области видимости строительство PDU смещение детали

**gSNMPV3TrapSecurityLevel** (см.страница 425) Это переменная gSNMPV3TrapSecurityLevel .

**gSnmpv3UserDBIndex** (см. стр.425) Индекс к конкретной ссылкой настроенного в модели безопасности пользователя база данных snmpV3UserDataBase (см. стр. 432 ) .

**gUsmStatsEngineID** (см. стр. 425) глобальной переменной , чтобы узнать, сколько раз SNMPv3 двигатель ID имеет была подтверждена

**hmacAuthKeyBuf** (см. стр. 425 ) Это переменная hmacAuthKeyBuf .

**HmacMd5Digest** (см. стр. 426 ) Это переменная HmacMd5Digest .

**HmacSHADigest** (см. стр. 426 ) Это переменная HmacSHADigest .

**incomingPdu** (см. стр. 426 ) Входящие данные PDU

**incomingSnmpPDUmsgID** (см. стр.426) Retrived Входящий значение MSG ID от PDU

**ivEncrptKeyOut** (см. стр. 426 ) Это переменная ivEncrptKeyOut .

**md5LocalizedAuthKey** (см. стр. 427 ) Это переменная md5LocalizedAuthKey .

**msgSecrtyParamLenOffset** (см. стр. 427) Это переменная msgSecrtyParamLenOffset .

**securityPrimitivesOfIncomingPdu** (см. стр. 427 ) Входящий PDU безопасности примитивные детали.

**SESSION_KEY** (см. стр. 427 ) Это переменная SESSION_KEY .

**sha1LocalizedAuthKey** (см. стр.428) Это переменная sha1LocalizedAuthKey .

**snmpEngineBoots** (см. стр. 428 ) раз , что двигатель SNMP имеет ( пере-) инициализации себя число

**snmpEngineID** (см. стр. 428 ) был последней конфигурации .

**snmpEngineID** (см. стр. 428) Резервирование 32 байт для snmpEngineID как длина строки октетов может варьироваться форму 5 до 32

**snmpEngineMaxMessageSize** (см.страница 428) Максимальный размер сообщения двигатель SNMP может справиться.

**snmpEngineMsgProcessModel** (см. страница 428) Тип сообщения модели обработки используется . Значение Максимальная дальность (2 ^ 31-1 ) , RFC3411

**snmpEngineSecurityModel** (см. стр.429) Тип модели безопасности используется. Значение Максимальная дальность (2 ^ 31-1 ) , RFC3411

**snmpEngineTime** (см. стр. 429 ), так как стоимости snmpEngineBoots Количество секунд ( см. стр. 428 ) объекта последнего изменения

**snmpEngineTimeOffset** (см. стр.429) Хранит значение времени в секундах после сброса SNMP двигателя

**snmpEngnIDLength** (см. стр. 429) Двигатель ID длина Engine SNMP

**snmpInMsgAuthParamLen** (см. стр.430) Входящий SNMPv3 параметры аутентификации MSG строка 12 байт долго.

**snmpInMsgAuthParamStrng** (см.страница 430) Резервирование 12 байт для входящего аутентификации MSG SNMPv3 параметры .

**snmpInMsgPrivParamLen** (см. стр. 430) Параметры Входящий SNMPv3 MSG конфиденциальности строка 8 байт.

**snmpInMsgPrvParamStrng** (см. стр.430) Резервирование 8 байт для входящих SNMPv3 MSG параметров конфиденциальности.

**snmpMsgBufSeekPos** ( см. стр. 430 ) Смещение читать байты данных PDU для обработки от динамической памяти огрызок

**snmpOutMsgAuthParamLen** (см. страница 431 ) Исходящие параметры аутентификации MSG SNMPv3 строка 12 байт долго.

**snmpOutMsgAuthParamStrng** (см.страница 431 ) Резервирование 12 байт для исходящей проверки подлинности MSG SNMPv3 параметры .

**snmpOutMsgPrivParamLen** (см. страница 431 ) Параметры исходящей SNMPv3 MSG конфиденциальности строка 8 байт.

**snmpOutMsgPrvParamStrng** (см.страница 431 ) Резервирование 8 байт для исходящих SNMPv3 MSG параметров конфиденциальности.

**snmpResponseSecurityFlag** (см. страница 432) Тип безопасности для исходящего сообщения в ответ на входящий сообщение .

**snmpSecurityLevel** ( см. стр. 432) Вид ценной бумаги . noAuthNoPriv (0), AuthNoPriv (1), AuthPriv (3)

**snmpV3AesDecryptInitVector** (см. страница 432) 128 Bit

**snmpV3AesEncryptInitVector** (см.страница 432) 128 Bit

**snmpV3UserDataBase** (см. стр.432) Это переменная snmpV3UserDataBase .

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

#### 10.15.7.1 authKey_iPad Переменная

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **authKey_iPad**[64];

**описание**

Это переменная authKey_iPad .

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

#### 10.15.7.2 authKey_oPad Переменная

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **authKey_oPad**[64];

**описание**

Это переменная authKey_oPad .

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

#### 10.15.7.3 authoritativeSnmpEngineBoots Переменная

**файл**

SNMPv3USM.c (см. стр. 447 )

C

DWORD_VAL **authoritativeSnmpEngineBoots**;

**описание**

Количество раз , что авторитетный двигатель SNMP имеет ( пере-) инициализации себя с момента ее snmpEngineID (см. стр. 428 ) последний был настроен .

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

#### 10.15.7.4 authoritativeSnmpEngineTime Переменная

**файл**

SNMPv3USM.c (см. стр. 447 )

C

DWORD_VAL **authoritativeSnmpEngineTime**;

**описание**

Так как значение из authoritativeSnmpEngineBoots Количество секунд (см. стр. 421 ) объекта последнего изменения

http://www.microchip.com/

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

**10.15.7.5 cipher_text Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **cipher_text**[16];

**описание**

Это переменная cipher_text .

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

**10.15.7.6 deciphered_text Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **deciphered_text**[16];

**описание**

Это переменная deciphered_text .

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

**10.15.7.7 getZeroInstance Переменная**

**файл**

SNMP.c (см. стр. 449 )

C

BOOL **getZeroInstance** = FALSE;

**описание**

Эта переменная используется для gext следующего запроса для нулевой например

http://www.microchip.com/

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

**10.15.7.8 gSnmpV3InPduWholeMsgBuf Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

SNMPV3_REQUEST_WHOLEMSG **gSnmpV3InPduWholeMsgBuf** = {NULL,NULL,{0},{0},0,0,0,0};

**описание**

Заглушки Динамическая память и PDU данные входящей хранится PDU

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

**10.15.7.9 gSnmpV3OUTPduWholeMsgBuf Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

SNMPV3_RESPONSE_WHOLEMSG **gSnmpV3OUTPduWholeMsgBuf** = {NULL,NULL,{0},{0},0,0,0,0};

**описание**

Подробнее заглушки динамической памяти и построенные исходящие хранящиеся подробнее PDU

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

**10.15.7.10 gSNMPv3PduHeaderBuf Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

SNMPV3MSGDATA **gSNMPv3PduHeaderBuf** = {NULL,0,0,0};

**описание**

Строительство PDU отклика смещение детали

http://www.microchip.com/

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

**10.15.7.11 gSNMPv3ScopedPduDataPos Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT16 **gSNMPv3ScopedPduDataPos** = 0;

**описание**

Смещение читать Scoped байтов данных PDU для обработки от окурка динамической памяти

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

**10.15.7.12 gSNMPv3ScopedPduRequestBuf Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

SNMPV3MSGDATA **gSNMPv3ScopedPduRequestBuf** = {NULL,0,0,0};

**описание**

Хранится запрос областью видимости детали PDU

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

**10.15.7.13 gSNMPv3ScopedPduResponseBuf Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

SNMPV3MSGDATA **gSNMPv3ScopedPduResponseBuf** = {NULL,0,0,0};

**описание**

Обработанные ответ областью видимости детали PDU

http://www.microchip.com/

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

**10.15.7.14 gSnmpv3TrapConfigData Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

snmpV3TrapConfigDataBase **gSnmpv3TrapConfigData**[SNMPV3_USM_MAX_USER] = { \
{"microchip",SNMPV3_MSG_PROCESSING_MODEL,SNMPV3_USM_SECURITY_MODEL,AUTH_PRIV}, \
{"SnmpAdmin",SNMPV3_MSG_PROCESSING_MODEL,SNMPV3_USM_SECURITY_MODEL,AUTH_NO_PRIV}, \
{"root",SNMPV3_MSG_PROCESSING_MODEL,SNMPV3_USM_SECURITY_MODEL,NO_AUTH_NO_PRIV} \ };

**описание**

Глобальная база данных конфигурации SNMPv3 , который будет использоваться для уведомления ловушки

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

**10.15.7.15 gSNMPv3TrapMsgHeaderBuf Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

SNMPV3MSGDATA **gSNMPv3TrapMsgHeaderBuf**;

**описание**

TRAP сообщение строительство заголовка PDU смещение детали

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

**10.15.7.16 gSnmpV3TrapOUTPduWholeMsgBuf Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

SNMPV3_RESPONSE_WHOLEMSG **gSnmpV3TrapOUTPduWholeMsgBuf** = {NULL,NULL,{0},{0},0,0,0,0};

http://www.microchip.com/

**описание**

Подробнее заглушки динамической памяти и построенные ловушки подробнее PDU

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

**10.15.7.17 gSNMPv3TrapScopedPduResponseBuf Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

SNMPV3MSGDATA **gSNMPv3TrapScopedPduResponseBuf** = {NULL,0,0,0};

**описание**

TRAP области видимости строительство PDU смещение детали

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

**10.15.7.18 gSNMPV3TrapSecurityLevel Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT8 **gSNMPV3TrapSecurityLevel** = NO_REPORT_NO_PRIVACY_NO_AUTH;

**описание**

Это переменная gSNMPV3TrapSecurityLevel .

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

**10.15.7.19 gSnmpv3UserDBIndex Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

WORD **gSnmpv3UserDBIndex**;

http://www.microchip.com/

**описание**

Индекс к конкретной ссылкой настроенного в модели безопасности пользователя базы данных snmpV3UserDataBase (см. стр. 432) .

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

**10.15.7.20 gUsmStatsEngineID Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

WORD_VAL **gUsmStatsEngineID** = {0};

**описание**

Глобальная переменная выяснить , сколько раз SNMPv3 двигатель ID была подтверждена

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

**10.15.7.21 hmacAuthKeyBuf Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **hmacAuthKeyBuf**[64];

**описание**

Это переменная **hmacAuthKeyBuf** .

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

**10.15.7.22 HmacMd5Digest Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **HmacMd5Digest**[16];

http://www.microchip.com/

**описание**

Это переменная **HmacMd5Digest** .

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

**10.15.7.23 HmacSHADigest Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **HmacSHADigest**[20];

**описание**

Это переменная HmacSHADigest .

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

**10.15.7.24 incomingPdu Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

**struct** dispatcherProcessPdu **incomingPdu**;

**описание**

Входящие подробнее PDU

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

**10.15.7.25 incomingSnmpPDUmsgID Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

DWORD_VAL **incomingSnmpPDUmsgID**;

http://www.microchip.com/

**описание**

Retrived Входящий значение MSG ID от PDU

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

**10.15.7.26 ivEncrptKeyOut Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **ivEncrptKeyOut**[16];

**описание**

Это переменная ivEncrptKeyOut .

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

**10.15.7.27 md5LocalizedAuthKey Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **md5LocalizedAuthKey**[16];

**описание**

Это переменная md5LocalizedAuthKey .

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

**10.15.7.28 msgSecrtyParamLenOffset Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

WORD **msgSecrtyParamLenOffset**;

http://www.microchip.com/

**описание**

Это переменная msgSecrtyParamLenOffset .

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

**10.15.7.29 securityPrimitivesOfIncomingPdu Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

SecuritySysProcessIncomingMsg **securityPrimitivesOfIncomingPdu**;

**описание**

Входящий PDU безопасности примитивные детали.

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

**10.15.7.30 SESSION_KEY Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

AES_SESSION_KEY_128_BIT **session_key**;

**описание**

Это переменная SESSION_KEY .

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

**10.15.7.31 sha1LocalizedAuthKey Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **sha1LocalizedAuthKey**[20];

http://www.microchip.com/

**описание**

Это переменная sha1LocalizedAuthKey .

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

**10.15.7.32 snmpEngineBoots Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT32 **snmpEngineBoots** = 0;

**описание**

Количество раз , что двигатель SNMP имеет ( пере-) инициализированными себя так snmpEngineID (см. стр. 428 ) в прошлом был настроен.

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

**10.15.7.33 snmpEngineID Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT8 **snmpEngineID**[32+1];

**описание**

Резервирование 32 байт для snmpEngineID как длина строки октета может варьироваться форму 5 до 32

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

**10.15.7.34 snmpEngineMaxMessageSize Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

DWORD_VAL **snmpEngineMaxMessageSize**;

**описание**

Максимальный размер сообщения двигатель SNMP может справиться.

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

**10.15.7.35 snmpEngineMsgProcessModel Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT32 **snmpEngineMsgProcessModel** = 0;

**описание**

Тип сообщения модели обработки используется . Значение Максимальная дальность (2 ^ 31-1 ) , RFC3411

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

**10.15.7.36 snmpEngineSecurityModel Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT32 **snmpEngineSecurityModel** = 0;

**описание**

Тип модели безопасности используется. Значение Максимальная дальность (2 ^ 31-1 ) , RFC3411

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

**10.15.7.37 snmpEngineTime Переменная**

**файл**

http://www.microchip.com/

SNMPv3.c (см. стр. 440 )

C

DWORD_VAL **snmpEngineTime**;

**описание**

Так как значение из snmpEngineBoots Количество секунд (см. стр. 428 ) объекта последнего изменения

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

**10.15.7.38 snmpEngineTimeOffset Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

DWORD **snmpEngineTimeOffset** = 0;

**описание**

Хранит значение времени в секундах после сброса SNMP двигателя

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

**10.15.7.39 snmpEngnIDLength Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT8 **snmpEngnIDLength** = 0;

**описание**

Двигатель длина ID к двигателю SNMP

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

**10.15.7.40 snmpInMsgAuthParamLen Переменная**

http://www.microchip.com/

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **snmpInMsgAuthParamLen**;

**описание**

Входящий SNMPv3 параметры аутентификации MSG строка 12 байт.

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

**10.15.7.41 snmpInMsgAuthParamStrng Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **snmpInMsgAuthParamStrng**[];

**описание**

Резервирование 12 байт для входящих SNMPv3 MSG параметров аутентификации .

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

**10.15.7.42 snmpInMsgPrivParamLen Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT8 **snmpInMsgPrivParamLen** = 8;

**описание**

Параметры Входящий SNMPv3 MSG конфиденциальности строка 8 байт.

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

**10.15.7.43 snmpInMsgPrvParamStrng Переменная**

http://www.microchip.com/

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **snmpInMsgPrvParamStrng**[];

**описание**

Резервирование 8 байт для входящих SNMPv3 MSG параметров конфиденциальности.

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

**10.15.7.44 snmpMsgBufSeekPos Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT16 **snmpMsgBufSeekPos** = 0;

**описание**

Смещение читать байты данных PDU для обработки от окурка динамической памяти

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

**10.15.7.45 snmpOutMsgAuthParamLen Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **snmpOutMsgAuthParamLen**;

**описание**

Исходящие параметры аутентификации SNMPv3 MSG строка 12 байт.

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

**10.15.7.46 snmpOutMsgAuthParamStrng Переменная**

http://www.microchip.com/

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **snmpOutMsgAuthParamStrng**[];

**описание**

Резервирование 12 байт для исходящих SNMPv3 MSG параметров аутентификации .

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

**10.15.7.47 snmpOutMsgPrivParamLen Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT8 **snmpOutMsgPrivParamLen** = 8;

**описание**

Параметры исходящей SNMPv3 MSG конфиденциальности строка 8 байт.

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

**10.15.7.48 snmpOutMsgPrvParamStrng Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **snmpOutMsgPrvParamStrng**[];

**описание**

Резервирование 8 байт для исходящих SNMPv3 MSG параметров конфиденциальности.

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

**10.15.7.49 snmpResponseSecurityFlag Переменная**

http://www.microchip.com/

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT8 **snmpResponseSecurityFlag** = 0;

**описание**

Тип безопасности для исходящего сообщения в ответ на входящее сообщение .

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

**10.15.7.50 snmpSecurityLevel Переменная**

**файл**

SNMPv3.c (см. стр. 440 )

C

UINT8 **snmpSecurityLevel** = 0;

**описание**

Вид ценной бумаги . noAuthNoPriv (0), AuthNoPriv (1), AuthPriv (3)

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

**10.15.7.51 snmpV3AesDecryptInitVector Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **snmpV3AesDecryptInitVector**[16+1];

**описание**

128 Bit

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

**10.15.7.52 snmpV3AesEncryptInitVector Переменная**

http://www.microchip.com/

**файл**

SNMPv3USM.c (см. стр. 447 )

C

UINT8 **snmpV3AesEncryptInitVector**[16+1];

**описание**

128 Bit

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

**10.15.7.53 snmpV3UserDataBase Переменная**

**файл**

SNMPv3USM.c (см. стр. 447 )

C

snmpV3EngnUserDataBase **snmpV3UserDataBase**[SNMPV3_USM_MAX_USER];

**описание**

Это переменная snmpV3UserDataBase .

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

## 10.15.8 макросы

**макрос**

**Имя Описание**

**AUTH_LOCALIZED_PASSWORD_KEY_LEN** (см. стр. 434 ) SNMPv3 аутентификации ( см. стр. 89 ) Локализованная passwed размер ключа lenegth

**INVALID_INDEX** (см. стр. 434 ) Этот макрос INVALID_INDEX .

**IS_SNMPV3_AUTH_STRUCTURE** (см. стр. 435 ) Этот макрос IS_SNMPV3_AUTH_STRUCTURE .

**MSG_AUTHORITATIVE_HEADER_LEN** ( см. стр. 435 ) Длина SNMPv3 authoratative MSG длина заголовка = длина заголовка (2 + 2 байт) + engineID ( snmpEngnIDLength ( См. стр. 429) байт)

• загрузки двигателя (4 байта) + двигатель время (4 байта) + имя безопасности

( securityPrimitivesOfIncomingPdu (см. стр. 427 ) значение) + аутентификацию

**параметры**

( snmpOutMsgAuthParamLen (см.страница 431 ) значение ) + конфиденциальности параметры

( snmpOutMsgAuthParamLen (см.431 ) Значение страница )

**MSGGLOBAL_HEADER_LEN** ( см. стр. 435 ) Длина ГЗС заголовке SNMPv3 (x) = Длина заголовка ( 2 байта )

• Размер MSGID (тип (1 байт) + длина стоимости (1 байт) значение 4 байта)

• msgMAXSIZE (тип + длина значение значение 4 байта)

• флаг MSG (тип + длина значения 1 Значение байта )

• Тип модели безопасности (тип + длина значение +1 значение байт)

**PRIV_LOCALIZED_PASSWORD_KEY_LEN** (см. стр. 436 ) # определить

PRIV_LOCALIZED_PASSWORD_KEY_LEN

16 SNMPv3 конфиденциальности длина Pasword ключ размер

**REPORT_RESPONSE** (см. стр. 436 ) Этот макрос REPORT_RESPONSE .

**SNMP_ENGINE_MAX_MSG_SIZE** (см. стр. 436 ) SNMP_ENGINE_MAX_MSG_SIZE является определяется как минимум по максимуму значения размера MSG поддерживается среди всех

транспорты доступны и поддерживается движком .

**SNMP_MAX_MSG_SIZE** (см. стр. 436 ) SNMP MIN и MAX сообщение 484 байт Согласно RFC 3411

**snmpEngineMaxMessageSize** (см.страница 428 ) и RFC 1157 (раздел 4 - Спецификация протокола ) и реализация поддерживает более 484 всякий раз, когда это возможно .

**SNMP_MAX_OID_LEN_MEM_USE** (см. стр. 437 ) Этот макрос будет использоваться , чтобы избежать SNMP Коррупция буфер OID памяти

**SNMP_TRAP_COMMUNITY_MAX_LEN_MEM_USE** (см. стр.437) Этот макрос будет использоваться , чтобы избежать SNMP Коррупция буфер OID памяти

**SNMP_V3** (см. стр. 437 ) Этот макрос SNMP_V3 .

**SNMPV3_AUTH_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE** (См. стр. 437 ) Аутентификации SNMPv3 локализованы ключ  Длина для проверки памяти

**SNMPV3_H** (см. стр. 438 ) Этот макрос SNMPV3_H .

**SNMPV3_PRIV_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE** (См. стр. 438 ) SNMPv3 конфиденциальности размер ключа длина проверка памяти

http://www.microchip.com/

**SNMPV3_USER_SECURITY_NAME_LEN_MEM_USE** (см. стр.438) Защита пользователей длина имени для памяти утверждение

**SNMPV3_USM_MAX_USER** (см. стр. 438 ) Этот макрос SNMPV3_USM_MAX_USER .

**SNMPV3MSG_AUTHENTICATION_FAIL** (см. стр. 439 ) Этот макрос SNMPV3MSG_AUTHENTICATION_FAIL.

**SNMPV3MSG_AUTHENTICATION_SUCCESS** (см. стр. 439 ) Этот макрос SNMPV3MSG_AUTHENTICATION_ SUCCESS .

**USER_SECURITY_NAME_LEN** (см. стр. 439 ) SNMPv3 длина безопасности имя пользователя

**модуль** SNMP (см. стр. 318 )

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

**10.15.8.1 AUTH_LOCALIZED_PASSWORD_KEY_LEN Макро**

**файл**

SNMPv3.h (см. стр. 443 )

C

**#define AUTH_LOCALIZED_PASSWORD_KEY_LEN** (20)

**описание**

SNMPv3 аутентификации ( см. стр. 89 ) Локализованная passwed размер ключа lenegth

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

**10.15.8.2 INVALID_INDEX Макро**

**файл**

SNMPv3.c (см. стр. 440 )

C

**#define INVALID_INDEX** 0xFF

**описание**

Это макро INVALID_INDEX .

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

http://www.microchip.com/

**10.15.8.3 IS_SNMPV3_AUTH_STRUCTURE Макро**

**файл**

SNMP.h (см. стр. 451 )

C

```
#define IS_SNMPV3_AUTH_STRUCTURE(a) (a==SNMPV3_ENCRYPTION)
```

**описание**

Это макро IS_SNMPV3_AUTH_STRUCTURE .

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

**10.15.8.4 MSG_AUTHORITATIVE_HEADER_LEN Макро**

**файл**

SNMPv3.c (см. стр. 440 )

C

```
#define MSG_AUTHORITATIVE_HEADER_LEN(x) ( x=(2+2 \
+1+1+snmpEngnIDLength \
+1+1+4 \
+1+1+4 \
+1+1+securityPrimitivesOfIncomingPdu.securityNameLength
\
+1+1+snmpOutMsgAuthParamLen \
+1+1+snmpOutMsgPrivParamLen) \
)
```

**описание**

Длина SNMPv3 authoratative MSG заголовка длиной = длина заголовка ( 2 + 2 байта) + engineID ( snmpEngnIDLength (см.страница 429 ) байт )

• загрузки двигателя (4 байта) + время двигатель ( 4 байта )+ имя безопасности ( securityPrimitivesOfIncomingPdu (см. стр. 427 ) значение ) + параметры аутентификации ( snmpOutMsgAuthParamLen (см. 431 ) номера страницы ) + параметры конфиденциальности ( snmpOutMsgAuthParamLen ( см. стр. 431 ) значение )

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

**10.15.8.5 MSGGLOBAL_HEADER_LEN Макро**

**файл**

SNMPv3.c (см. стр. 440 )

C

```
#define MSGGLOBAL_HEADER_LEN(x) ( x= (2 \
+1+1+4 \
+1+1+4 \
+1+1+1 \
+1+1+1)\
)
```

**описание**

Длина MSG заголовке SNMPv3 (x) = длина заголовка ( 2 байта )

• Размер MSGID (тип (1 байт) + длина значения (1 байт) значение 4 байта)

• msgMAXSIZE (тип + длина значения стоимости 4 байта )

• флаг MSG (тип + длина значения 1 значения байта )

• Тип модели безопасности (тип + длина значения 1 значения байта )

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

### 10.15.8.6 PRIV_LOCALIZED_PASSWORD_KEY_LEN Макро

**файл**

SNMPv3.h (см. стр. 443 )

C

```
#define PRIV_LOCALIZED_PASSWORD_KEY_LEN (20)
```

**описание**

# определить PRIV_LOCALIZED_PASSWORD_KEY_LEN 16 SNMPv3 конфиденциальности Pasword размер ключа длины

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

### 10.15.8.7 REPORT_RESPONSE Макро

**файл**

SNMP.h (см. стр. 451 )

http://www.microchip.com/

C

**#define REPORT_RESPONSE** (0xa8)

**описание**

Это макро REPORT_RESPONSE .

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

**10.15.8.8 SNMP_ENGINE_MAX_MSG_SIZE Макро**

**файл**

SNMPv3.c (см. стр. 440 )

C

**#define SNMP_ENGINE_MAX_MSG_SIZE** 1024

**описание**

SNMP_ENGINE_MAX_MSG_SIZE определяется как минимум значений максимальный размер MSG поддерживаемых среди всех транспортирует доступен и поддерживается движком .

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

**10.15.8.9 SNMP_MAX_MSG_SIZE Макро**

**файл**

SNMP.h (см. стр. 451 )

C

**#define SNMP_MAX_MSG_SIZE** 484

**описание**

SNMP MIN и MAX сообщение 484 байт Как за RFC 3411 snmpEngineMaxMessageSize (см. стр. 428) и RFC 1157 (раздел 4 - спецификация протокола ) и реализация поддерживает более 484 , где это возможно .

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

**10.15.8.10 SNMP_MAX_OID_LEN_MEM_USE Макро**

http://www.microchip.com/

**файл**

SNMP.h (см. стр. 451 )

C

**#define SNMP_MAX_OID_LEN_MEM_USE** (18)

**описание**

Этот макрос будет использоваться , чтобы избежать SNMP OID с коррупцией буфера памяти

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

**10.15.8.11 SNMP_TRAP_COMMUNITY_MAX_LEN_MEM_USE**

**макрос**

**файл**

SNMP.h (см. стр. 451 )

C

**#define SNMP_TRAP_COMMUNITY_MAX_LEN_MEM_USE** (8)

**описание**

Этот макрос будет использоваться , чтобы избежать SNMP OID с коррупцией буфера памяти

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

**10.15.8.12 SNMP_V3 Макро**

**файл**

SNMP.h (см. стр. 451 )

C

**#define SNMP_V3** (3)

**описание**

Это макро SNMP_V3 .

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

http://www.microchip.com/

**10.15.8.13**

**SNMPV3_AUTH_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE**

**макрос**

**файл**

SNMPv3.h (см. стр. 443 )

C

**#define SNMPV3_AUTH_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE**
(AUTH_LOCALIZED_PASSWORD_KEY_LEN+1)

**описание**

Аутентификации SNMPv3 локализованы Длина ключа для проверки памяти

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

**10.15.8.14 SNMPV3_H Макро**

**файл**

SNMPv3.h (см. стр. 443 )

C

**#define SNMPV3_H**

**описание**

Это макро SNMPV3_H .

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

**10.15.8.15**

**SNMPV3_PRIV_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE**

**макрос**

**файл**

SNMPv3.h (см. стр. 443 )

C

http://www.microchip.com/

**#define SNMPV3_PRIV_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE**
(PRIV_LOCALIZED_PASSWORD_KEY_LEN+1)

**описание**

SNMPv3 конфиденциальности размер ключа длиной для проверки памяти

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

**10.15.8.16 SNMPV3_USER_SECURITY_NAME_LEN_MEM_USE**

**макрос**

**файл**

SNMPv3.h (см. стр. 443 )

C

**#define SNMPV3_USER_SECURITY_NAME_LEN_MEM_USE** (USER_SECURITY_NAME_LEN+1)

**описание**

Защита пользователей длина имени для проверки памяти

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

**10.15.8.17 SNMPV3_USM_MAX_USER Макро**

**файл**

SNMPv3.h (см. стр. 443 )

C

**#define SNMPV3_USM_MAX_USER** 3

**описание**

Это макро SNMPV3_USM_MAX_USER .

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

**10.15.8.18 SNMPV3MSG_AUTHENTICATION_FAIL Макро**

**файл**

http://www.microchip.com/

SNMPv3.h (см. стр. 443 )

C

**#define SNMPV3MSG_AUTHENTICATION_FAIL** 0

**описание**

Это макро SNMPV3MSG_AUTHENTICATION_FAIL .

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

**10.15.8.19 SNMPV3MSG_AUTHENTICATION_SUCCESS Макро**

**файл**

SNMPv3.h (см. стр. 443 )

C

**#define SNMPV3MSG_AUTHENTICATION_SUCCESS** 1

**описание**

Это макро SNMPV3MSG_AUTHENTICATION_SUCCESS .

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

**10.15.8.20 USER_SECURITY_NAME_LEN Макро**

**файл**

SNMPv3.h (см. стр. 443 )

C

**#define USER_SECURITY_NAME_LEN** (16)

**описание**

SNMPv3 Пользователь длина безопасности Имя

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

## 10.15.9 Файлы

http://www.microchip.com/

**Файлы**

**Имя Описание**

SNMPv3.c (см. стр. 440 )

• простой протокол управления сетью (SNMP ) Версия 3 Агент

*

• Модуль для Microchip Стек TCP / IP

• - Обеспечивает SNMPv3 API для делать вещи

*

• - Ссылка: RFC, 3410 , 3411, 3412 , 3413 , 3414

SNMPv3.h (см. стр. 443 )

• простой протокол управления сетью (SNMP ) Версия 3 Агент

*

• Модуль для Microchip Стек TCP / IP

• - Обеспечивает SNMPv3 API для делать вещи

*

• - Ссылка: RFC, 3410 , 3411, 3412 , 3413 , 3414

SNMPv3USM.c (см. стр. 447 )

• простой протокол управления сетью (SNMP ) Версия 3 Агент

*

• Модуль для Microchip Стек TCP / IP

• - Обеспечивает SNMPv3 API для делать вещи

*

• - Ссылка: RFC, 3410 , 3411, 3412 , 3413 , 3414

SNMP.c (см. стр. 449 )

• простой протокол управления сетью (SNMP ) Версия 1 Агент

• простой протокол управления сетью (SNMP ) версии 2 сообщество

Агентное

• Модуль для Microchip Стек TCP / IP

• - Обеспечивает SNMP API для делать вещи

http://www.microchip.com/

*

• - Ссылка: RFC 1157 (для SNMP V1 )

• RFC 3416 (для SNMPv2C )

SNMP.h (см. стр. 451 )

• SNMP DEFS для Microchip Стек TCP / IP

*

**модуль** SNMP (см. стр. 318 )

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

## 10.15.9.1 SNMPv3.c

**Функции**

**Имя Описание**

**_Snmpv3IsValidAuthStructure**(См. стр. 374 ) Расшифруйте переменную структуру длины.

**FindOIDsFromSnmpV3Request** (См. стр. 376 ) Находит количество varbinds в списке VarBind полученной в PDU SNMPv3 .

**IsSnmpV3ASNNull** (см.страница 377) Проверяет тип значения , как ASN_NULL (см. стр. 344 ) .

**IsSnmpv3ValidOID** (см.страница 377) Заполняет OID тип , длину и подъязычная строку из принятого PDU.

**IsSNMPv3ValidStructure** (см. стр. 378 ) Расшифруйте переменную структуру длины.

**Snmpv3FreeDynAllocMem** (см. стр. 390 ) Выделенные освобождение динамической памяти осуществляется этой рутины .

**Snmpv3IsValidAuthStructure**(См. стр. 393 ) Расшифруйте переменную структуру длины.

**Snmpv3IsValidInt** (см. стр.394) Проверяет переменную типа данных как INT и извлекает его значение.

**Snmpv3ReportPdu** (см.страница 396 ) Создает отчета PDU infomration для отчета PDU.

**Snmpv3SetErrorStatus** (см.страница 397) Установите SNMPv3 статус ошибки в PDU ответа.

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

**макрос**

**Имя Описание**

**INVALID_INDEX** (см. стр. 434 ) Этот макрос INVALID_INDEX .

**MSG_AUTHORITATIVE_HEADER_LEN** (См. стр. 435 ) Длина SNMPv3 authoratative MSG заголовка длиной = длина заголовка (2 + 2 байта ) + engineID ( snmpEngnIDLength (см. стр. 429 ) байта)

• загрузки двигателя (4 байта) + время двигатель ( 4 байта ) + имя безопасности ( securityPrimitivesOfIncomingPdu (см. стр.427 ) значения) + параметры аутентификации ( snmpOutMsgAuthParamLen (см. 431 ) номера страницы ) + конфиденциальности параметры ( snmpOutMsgAuthParamLen (см. 431 ) значение стр. )

**MSGGLOBAL_HEADER_LEN** (см.страница 435 ) Длина MSG заголовке SNMPv3 (x) = длина заголовка ( 2 байта )

• Размер MSGID (тип (1 байт) + длина значения (1 байт) 4 байт значение )

• msgMAXSIZE (тип + длина значения стоимости 4 байта )

• флаг MSG (тип + длина значения 1 значения байта )

• Тип модели безопасности (тип + длина значения 1 значения байта )

**SNMP_ENGINE_MAX_MSG_SIZE** (см. стр. 436) SNMP_ENGINE_MAX_MSG_SIZE определяется как минимум что максимальные значения размера MSG поддерживаемые среди всех транспортов доступен и поддерживается движком .

**Переменные**

**Имя Описание**

**gSnmpV3InPduWholeMsgBuf** (см.страница 422 ) Заглушки Динамическая память и PDU данные входящей хранится PDU

**gSnmpV3OUTPduWholeMsgBuf** (см. страница 423) Подробнее заглушки динамической памяти и построена исходящих сохраняются PDU детали

**gSNMPv3PduHeaderBuf** (см. стр.423) Строительство PDU отклика смещение детали

**gSNMPv3ScopedPduDataPos** (см.страница 423) Смещение читать Scoped байтов данных PDU для обработки от динамического заглушки памяти

**gSNMPv3ScopedPduRequestBuf** (см. стр. 423 ) Хранится запрос областью видимости детали PDU

**gSNMPv3ScopedPduResponseBuf** (см. стр. 424) Обработанные ответ областью видимости детали PDU

**gSnmpv3TrapConfigData** (см. стр.424) Глобальная база данных конфигурации SNMPv3 , который будет использоваться для уведомления ловушки

**gSnmpV3TrapOUTPduWholeMsgBuf** (См. стр. 424 ) Подробнее заглушки динамической памяти и построенные ловушки подробнее PDU

**gSNMPv3TrapScopedPduResponseBuf** (См. стр. 424 ) TRAP области видимости строительство PDU смещение детали

**gSNMPV3TrapSecurityLevel** (см.страница 425) Это переменная gSNMPV3TrapSecurityLevel .

**gUsmStatsEngineID** (см. стр. 425) глобальной переменной , чтобы узнать, сколько раз SNMPv3 двигатель ID имеет была подтверждена

**incomingPdu** (см. стр. 426 ) Входящие данные PDU

**incomingSnmpPDUmsgID** (см. стр. 426) Retrived Входящий значение MSG ID от PDU

**msgSecrtyParamLenOffset** (см. стр.427) Это переменная msgSecrtyParamLenOffset .

**securityPrimitivesOfIncomingPdu** (см. стр. 427 ) Входящий PDU безопасности примитивные детали.

**snmpEngineBoots** (см. стр. 428 ) раз , что двигатель SNMP имеет ( пере-) инициализации себя число

**snmpEngineID** (см. стр. 428 ) был последней конфигурации .

**snmpEngineID** (см. стр. 428) Резервирование 32 байт для snmpEngineID как длина строки октетов может варьироваться форму 5 до 32

**snmpEngineMaxMessageSize** (см. страница 428) Максимальный размер сообщения двигатель SNMP может справиться.

**snmpEngineMsgProcessModel** (см.страница 428) Тип сообщения модели обработки используется . Значение Максимальная дальность (2 ^ 31-1 ) , RFC3411

**snmpEngineSecurityModel** (см. стр.429) Тип модели безопасности используется. Значение Максимальная дальность (2 ^ 31-1 ) , RFC3411

**snmpEngineTime** (см. стр. 429 ), так как стоимости snmpEngineBoots Количество секунд ( см. стр. 428 ) объекта последнего изменения

**snmpEngineTimeOffset** (см. стр.429) Хранит значение времени в секундах после сброса SNMP двигателя

**snmpEngnIDLength** (см. стр. 429) Двигатель ID длина Engine SNMP

**snmpInMsgPrivParamLen** (см. стр.430) Параметры Входящий SNMPv3 MSG конфиденциальности строка 8 байт.

**snmpMsgBufSeekPos** ( см. стр. 430 ) Смещение читать байты данных PDU для обработки от динамической памяти огрызок

**snmpOutMsgPrivParamLen** (см. страница 431 ) Параметры исходящей SNMPv3 MSG конфиденциальности строка 8 байт.

**snmpResponseSecurityFlag** (см.страница 432) Тип безопасности для исходящего сообщения в ответ на входящий сообщение .

**snmpSecurityLevel** ( см. стр. 432) Вид ценной бумаги . noAuthNoPriv (0), AuthNoPriv (1), AuthPriv (3)

http://www.microchip.com/

**SNMPTxOffset** (см. стр. 361 ) SNMP UDP буфер TX смещение

**описание**

• Простой протокол сетевого управления (SNMP) Версия 3 Агент

*

• Модуль для Microchip Стек TCP / IP

• - Обеспечивает SNMPv3 API для делать вещи

*

• - Ссылка: RFC, 3410 , 3411, 3412 , 3413 , 3414

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

**10.15.9.2 SNMPv3.h**

**Перечисления**

**Имя Описание**

**REPORT_FLAG_AND_SECURITY_LEVEL_FLAGS** (см. стр. 412 ) Это тип REPORT_FLAG_AND_SECURITY_LEVEL_FLAGS.

**SNMP_ENGNID_OCTET_IDENTIFIER_VAL** (см. стр.414) Пятый октет указывает, как остальные ( 6-й и Следующие октета) отформатированы . См. RFC3411 section5 Страница № 41

**SNMPV3_HMAC_HASH_TYPE** (см. стр. 414 ) Тип хэша рассчитывается

**SNMPV3_MSG_AUTH_SEC_PARAM_RESULT** (см.страница 415) Это тип SNMPV3_MSG_AUTH_SEC_PARAM_RESULT .

**SNMPV3_MSG_PRIV_SEC_PARAM_RESULT** (см.страница 415) Это тип SNMPV3_MSG_PRIV_SEC_PARAM_RESULT .

**SNMPV3_PRIV_PROT_TYPE** (см. стр. 415 ) Это тип SNMPV3_PRIV_PROT_TYPE .

**STD_BASED_SNMP_MESSAGE_PROCESSING_MODEL** (См. стр. 418 ) Это тип STD_BASED_SNMP_MESSAGE_PROCESSING_ MODEL.

**STD_BASED_SNMP_SECURITY_MODEL** (см. стр. 418) Snmp Сообщение Модель обработки

**STD_BASED_SNMPV3_SECURITY_LEVEL** (см. стр.418) Это тип STD_BASED_SNMPV3_SECURITY_LEVEL .

**USM_SECURITY_LEVEL** (см. стр. 419 ) Это типа USM_SECURITY_LEVEL .

http://www.microchip.com/

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

**Функции**

**Имя Описание**

**_IsSNMPv3ValidStructure** (См. стр. 374 ) Decode переменную структуру длины.

**_Snmpv3IsValidInt** (См. стр. 375 ) Проверяет переменную типа данных как INT и извлекает его значение.

**Snmpv3AESEncryptResponseScopedPdu** (см.страница 383) Исходящие SNMPv3 областью действия PDU шифрования с использованием AES  Протокол шифрования .

**Snmpv3AuthenticateRxedPduForDataIntegrity** (см.страница 384 ) Аутентификация на входящий SNMPv3 USM PDU с помощью MD5 или SHA

**Snmpv3AuthenticateTxPduForDataIntegrity** (см. страница 384 ) Подлинность исходящего SNMPv3 USM PDU с помощью MD5 или SHA

**Snmpv3AuthKeyZeroing2HmacBufLen64** (см.страница 385 ) Коврик нуля до ключа аутентификации локализованной буфера.

**Snmpv3CmprTrapSecNameAndSecLvlWithUSMDb**(См. стр. 386 ) Регулярное найти индекс имени пользователя в пользовательском база данных таблицы .

**Snmpv3ComputeHMACIpadOpadForAuthLoclzedKey** (См. стр. 386 ) Вычислить HMAC внутренняя и внешняя площадка для авторизации локализованы ключ .

**Snmpv3ComputeHmacMD5Digest** (см. стр. 387 ) Вычислить HMAC - MD5 код аутентификации

**Snmpv3ComputeHmacShaDigest** (см. стр. 388 ) Вычислить HMAC - Ша код аутентификации

**Snmpv3ComputeMd5HmacCode** (см. стр. 388 ) Вычислить HMAC - MD5 код аутентификации

**Snmpv3ComputeShaHmacCode** (см. стр. 389 ) Вычислить HMAC - Ша код аутентификации

**Snmpv3GetSecurityLevel** (см. стр. 391) Получить уровень безопасности от аутентификации и конфиденциальности типа.

**Snmpv3GetTrapSecurityLevel** ( см. стр. 392 ) обычные найти в докладе , Auth и приватности флаги Настройки в ловушке.

**Snmpv3Init** ( см. стр. 392) инициализации SNMPv3 .

**Snmpv3InitializeUserDataBase** (см. стр. 393) SNMPv3 базы данных глобальной пользователь Intialize умолчанию .

**Snmpv3Pswd2LocalizedAuthKeyMD5Hashing** (см. страница 395) Преобразование MD5 Auth пароль для локализованной Key помощью SNMPEngineID .

**Snmpv3Pswd2LocalizedAuthKeySHAHashing** (см. страница 396 ) Преобразование SHA Auth пароль для локализованной Key помощью SNMPEngineID .

**Snmpv3USMOutMsgPrivParam** (см. стр. 400 ) SNMP USM из сообщение использует конфиденциальности протокол (RFC 3826 )

**Snmpv3UsmSnmpEngnAuthPrivPswdLocalization** ( см. стр. 400) Преобразование Auth и Priv пароль для локализованной Key используя SNMPEngineID .

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

**макрос**

**Имя Описание**

**AUTH_LOCALIZED_PASSWORD_KEY_LEN** (см. стр. 434 ) SNMPv3 аутентификации ( см. стр. 89 ) Локализованная passwed размер ключа lenegth

**PRIV_LOCALIZED_PASSWORD_KEY_LEN** (см. стр. 436 ) # определить PRIV_LOCALIZED_PASSWORD_KEY_LEN 16 SNMPv3 конфиденциальности длина Pasword ключ размер

**SNMPV3_AUTH_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE** (См. стр. 437 ) Аутентификации SNMPv3 локализованы ключ Длина для проверки памяти

**SNMPV3_H** (см. стр. 438 ) Этот макрос SNMPV3_H .

**SNMPV3_PRIV_LOCALIZED_PASSWORD_KEY_LEN_MEM_USE** (См. стр. 438 ) SNMPv3 конфиденциальности размер ключа длина проверка памяти

**SNMPV3_USER_SECURITY_NAME_LEN_MEM_USE** (см. стр. 438) Защита пользователей длина имени для памяти утверждение

**SNMPV3_USM_MAX_USER** (см. стр. 438 ) Этот макрос SNMPV3_USM_MAX_USER .

**SNMPV3MSG_AUTHENTICATION_FAIL** (см. стр. 439 ) Этот макрос SNMPV3MSG_AUTHENTICATION_FAIL.

**SNMPV3MSG_AUTHENTICATION_SUCCESS** (см. стр. 439 ) Этот макрос SNMPV3MSG_AUTHENTICATION_ SUCCESS

**USER_SECURITY_NAME_LEN** (см. стр. 439 ) SNMPv3 длина безопасности имя пользователя

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

**структуры**

**Имя Описание**

**AccessCtrlSubSysIsAccessAllowed** ( см. стр. 403) Приложения являются типичными клиентами услуги (услуг) о доступе Подсистема управления . В следующем примитив обеспечивается Контроль доступа Подсистема для проверки доступ разрешен : **statusInformation** (см. стр. 417 ) = - успех или errorIndication

**dispatcherProcessPdu** (см. стр. 404) Process поступления запроса или уведомления PDU Диспетчер предоставляет следующие примитивны, чтобы пройти на входящий SNMP PDU в приложение.

**dispatcherStatusInfo** (см. стр. 404 ) Создание исходящий запрос или уведомление

**statusInformation** (см. стр. 417 ) = - sendPduHandle если успех - errorIndication если отказ

**dispathcerReturnResponsePdu** (см.страница 405 ) Генерация исходящих ответов PDU Диспетчер предоставляет следующие примитив для Приложение вернуть SNMP Response PDU в PDU Диспетчер:

результат = успех или неудача

**MsgProcModPrepareDataElements** (см. стр. 406) Подготовка элементов данных из входящего сообщения SNMP Сообщение Обработка Подсистема предоставляет эту услугу примитивный для подготовки абстрактные элементы данных из входящего SNMP сообщение : результат = - успех или errorIndication

**MsgProcModPrepareOutgoingMessage** (См. стр. 407 ) Подготовка исходящей SNMP запроса или уведомления сообщение Сообщение Обработка Подсистема предоставляет эту услугу примитивный для подготовки исходящего SNMP запроса или уведомления сообщение

**MsgProcModPrepareResponseMessage** (См. стр. 407 ) Подготовка исходящего SNMP ответное сообщение Сообщение Обработка Подсистема предоставляет эту услугу примитивный для подготовки исходящего SNMP ответное сообщение : результат = - Успех или неудача

**processResponsePdu** (см. стр. 408) обрабатывать входящие PDU отклика PDU Диспетчер предоставляет следующие примитивны, чтобы пройти входящие SNMP отклика PDU к приложению :

**registerContextEngineID** (см. стр. 417) Успех или errorIndication

**SecuritySysGenerateRequestMsg** (см. стр. 409) Это запись SecuritySysGenerateRequestMsg .

**SecuritySysGenerateResponseMsg** (см. стр. 410) Генерирует ответное сообщение, Подсистема защиты предоставляет следующие примитивны, чтобы генерировать Ответное сообщение :

**StateRelease** (см. стр. 410 ) Государственный выпуска справочная информация Все подсистемы , которые проходят stateReference информацию также обеспечивают примитивный для освобождения памяти , которая содержит ссылается состояние информация

**unregisterContextEngineID** (см. стр.411 ) Это запись unregisterContextEngineID .

**SecuritySysProcessIncomingMsg** (см. стр. 413) Это тип SecuritySysProcessIncomingMsg .

**SNMPV3_REQUEST_WHOLEMSG** ( см. стр. 416) Это тип SNMPV3_REQUEST_WHOLEMSG .

**SNMPV3_RESPONSE_WHOLEMSG**(См. стр. 416 ) Это тип SNMPV3_RESPONSE_WHOLEMSG .

**snmpV3EngnUserDataBase** (см.страница 416) Это тип snmpV3EngnUserDataBase .

**snmpV3TrapConfigDataBase** (см.страница 417 ) целевой конфигурации SNMPv3 по отношению к ловушке.

**statusInformation** ( см. стр. 417), успех или errorIndication

http://www.microchip.com/

**описание**

• Простой протокол сетевого управления (SNMP) Версия 3 Агент

*

• Модуль для Microchip Стек TCP / IP

• - Обеспечивает SNMPv3 API для делать вещи

*

• - Ссылка: RFC, 3410 , 3411, 3412 , 3413 , 3414

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

**10.15.9.3 SNMPv3USM.c**

**Переменные**

**Имя Описание**

**authKey_iPad** (см. стр.421) Это переменная authKey_iPad .

**authKey_oPad** (см. стр.421) Это переменная authKey_oPad .

**authoritativeSnmpEngineBoots**(См. стр. 421 ) Что авторитетный двигатель SNMP имеет ( пере-) инициализации количество раз Сам с прошлого его snmpEngineID (см. стр. 428 ) был настроен.

**authoritativeSnmpEngineTime** (См. стр. 422 ) Так как значение из authoritativeSnmpEngineBoots Количество секунд (См. стр. 421 ) объекта последнего изменения

**cipher_text** (см. стр. 422 ) Это переменная cipher_text .

**deciphered_text** (см. стр.422) Это переменная deciphered_text .

**gSNMPv3TrapMsgHeaderBuf** (См. стр. 424 )TRAP сообщение строительство заголовка PDU смещение детали

**gSnmpv3UserDBIndex** (см.страница 425) Индекс к конкретной ссылкой настроенного в модели безопасности базы данных пользователей

**snmpV3UserDataBase** (см. стр. 432) .

**hmacAuthKeyBuf** (см. стр.425) Это переменная hmacAuthKeyBuf .

**HmacMd5Digest** (см. стр.426) Это переменная HmacMd5Digest .

**HmacSHADigest** (см. стр.426) Это переменная HmacSHADigest .

**ivEncrptKeyOut** (см. стр.426) Это переменная ivEncrptKeyOut .

http://www.microchip.com/

**md5LocalizedAuthKey** (см.страница 427 ) Это переменная md5LocalizedAuthKey .

**SESSION_KEY** (см. стр. 427 ) Это переменная SESSION_KEY .

**sha1LocalizedAuthKey** (см.страница 428) Это переменная sha1LocalizedAuthKey .

**snmpInMsgAuthParamLen** (см. стр. 430 ) Входящий SNMPv3 параметры аутентификации MSG строка 12 байт.

**snmpInMsgAuthParamStrng** (См. стр. 430 ) Резервирование 12 байт для входящих SNMPv3 MSG параметров аутентификации .

**snmpInMsgPrvParamStrng** (см. стр. 430 ) Резервирование 8 байт для входящих SNMPv3 MSG параметров конфиденциальности.

**snmpOutMsgAuthParamLen**(См. стр. 431 ) Исходящие параметры аутентификации SNMPv3 MSG строка 12 байт.

**snmpOutMsgAuthParamStrng** (См. стр. 431 ) Резервирование 12 байт для исходящих SNMPv3 MSG параметров аутентификации .

**snmpOutMsgPrvParamStrng** (См. стр. 431 ) Резервирование 8 байт для исходящих SNMPv3 MSG параметров конфиденциальности.

**snmpV3AesDecryptInitVector** (См. стр. 432 ) 128 Bit

**snmpV3AesEncryptInitVector**(См. стр. 432 ) 128 Bit

**snmpV3UserDataBase** (см.страница 432) Это переменная snmpV3UserDataBase .

**описание**

• Простой протокол сетевого управления (SNMP) Версия 3 Агент

*

• Модуль для Microchip Стек TCP / IP

• - Обеспечивает SNMPv3 API для делать вещи

*

• - Ссылка: RFC, 3410 , 3411, 3412 , 3413 , 3414

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

## 10.15.9.4 SNMP.c

**Функции**

http://www.microchip.com/

**Имя Описание**

**FindOIDsInRequest** (см. стр.346) Находит количество varbinds в списке VarBind полученной в PDU.

**IsASNNull** (см. стр. 351 ) Проверяет тип значения , как ASN_NULL (см. стр. 344 ) .

**IsValidCommunity** (см. стр.364) Проверяет для сообщества строки типа данных и имени макс сообщества и длина , этот агент может обрабатывать .

**IsValidOID** (см. стр. 365 ) Заполняет OID тип , длину и подъязычная строку из принятого PDU.

**IsValidPDU** (см. стр. 366 ) Проверяет для типа запроса SNMP .

**ProcessGetSetHeader** (см.страница 367) Проверяетполучил UDP-пакетов Получить / заголовок запроса Set.

**ProcessHeader** (см. стр. 368 ) Проверяет получил UDP-пакетов заголовок SNMP .

**ProcessVariables** ( см. стр. 369 ) Эта процедура обрабатывает запрос SNMP и параллельно создает ответ PDU.

**ReadMIBRecord** (см. стр. 370 ) Получить OID строки из MPFS используя адрес узла .

**SNMPCheckIfPvtMibObjRequested** (См. стр. 370 ) Чтобы найти ли просил OID только для частного доступа .

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

**Переменные**

**Имя Описание**

**appendZeroToOID** (см.страница 344) глобальный флаг , чтобы изменить OID путем добавления нуля

**dataTypeTable** (см. стр. 346) ASN формат тип данных для SNMP v1 и v2c

**getZeroInstance** (см. стр.422) Эта переменная используется для gext следующего запроса для нулевой например

**hMPFS** (см. стр. 348 ) ГПСЛ обработчик файлов

**SNMPAgentSocket** (см. страница 360) Snmp сокета UDP

**SNMPNotifyInfo** (см. стр.360) уведомить информацию для ловушку

**snmpReqVarErrStatus** ( см. стр. 361) вары из списка REQ статус обработки ошибаться

**SNMPRxOffset** (см. стр.361) Snmp UDP буфер RX смещение

**SNMPStatus** (см. стр.361) Статус доступа MIB файл

**описание**

http://www.microchip.com/

• Простой протокол управления сетью (SNMP ) Версия 1 Агент

• простой протокол управления сетью (SNMP ) версии 2 сообщество Агентное

• Модуль для Microchip Стек TCP / IP

• - Обеспечивает SNMP API для делать вещи

*

• - Ссылка: RFC 1157 (для SNMP V1 )

• RFC 3416 (для SNMPv2C )

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

**10.15.9.5 SNMP.h**

**Перечисления**

**Имя Описание**

**COMMUNITY_TYPE** (см. стр. 329 ) Это типа COMMUNITY_TYPE .

**DATA_TYPE** (см. стр. 345 )

**GENERIC_TRAP_NOTIFICATION_TYPE** (см. страница 327) Это тип GENERIC_TRAP_NOTIFICATION_TYPE.

**INOUT_SNMP_PDU** (см. стр. 412 ) Это типа INOUT_SNMP_PDU .

**SNMP_ACTION** (см. стр. 328 ) Этот список SNMP действия через удаленные NMS может выполнять . Это Inforamtion передается приложения через обратного вызова SNMPValidateCommunity (см. стр. 336 ) ( ) .

Заявка должна проверить действие при заданных Строка сообщества .

**SNMP_ERR_STATUS** (см. стр. 356 )

**VENDOR_SPECIFIC_TRAP_NOTIFICATION_TYPE**(См. стр. 328 ) Это тип VENDOR_SPECIFIC_TRAP_NOTIFICATION_TYPE .

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

Функции

Имя Описание

**_SNMPDuplexInit** (См. стр. 342) Подготовка к полнодуплексном передачи .

**_SNMPGet** ( См. стр. 342) Read байт из SNMP UDP буфера сокета RX.

**_SNMPPut** (См. стр. 343 ) Скопируйте байт TX буфера.

**GetDataTypeInfo** (см. стр. 350 ) Получить ASN тип данных информацию.

**GetNextLeaf** (см. стр. 362 ) Ищет следующего листа в MIP дерева.

**GetOIDStringByAddr** (см. стр. 363 ) Получить OID строки из MPFS используя адрес узла .

**GetOIDStringByID** (см. стр. 363 ) Получить уведомление о завершении переменная OID строки из MPFS используя вар ID .

**getSnmpV2GenTrapOid** (см. стр. 376 ) постановляет общего кода ловушки для общего ловушки OID.

**IsValidInt** (см. стр. 364 ) Проверяет переменную типа данных как INT и извлекает его значение.

**IsValidLength** (см. стр. 358 ) Получает длину пакета и фактическую длину PDU.

**IsValidStructure** (см. стр. 366 ) Decode переменную структуру длины.

**OIDLookup** (см. стр. 367) Для поиска и проверки , является ли просьба OID находится в MIB B базе .

**ProcessGetBulkVar** (см. стр. 378 ) Этот рутинный процессSNMPv2c Получить общий запрос .

**ProcessGetNextVar** (см. стр. 379 ) Получает следующий узел из базы данных MIB .

**ProcessGetVar** (см. стр. 380 ) Процессы SNMP Получить запроса PDU .

**ProcessSetVar** (см. стр. 369 ) Процессы установки SNMP запроса PDU .

**ProcessSnmpv3MsgData** (см. стр.380) Эта процедура обрабатывает запрос SNMPv3 и параллельно создает PDU ответа .

**SetErrorStatus** (см. стр. 355 ) указан SNMP статус ошибки в PDU ответа.

**SNMPGetExactIndex** (см. стр. 381 ) для поиска точного индекса узла в случае переменной последовательности .

**SNMPGetNextIndex** (см. стр. 335 ) для поиска следующего индекса узла в случае переменной последовательности .

**SNMPGetVar** (см. стр. 333 ) используется для получения / сбора OID переменной информации .

**SNMPIdRecrdValidation** (см. стр. 381 ) используется для ограничения доступа динамичный и не динамическая строка OID для A perticular SNMP версии .

**SNMPInit** (см. стр. 371) инициализации SNMP модуль внутренних органов.

**SNMPIsNotifyReady** (см. стр. 334 ) решает учитывая RemoteHost IP -адрес в MAC-адрес.

**SNMPIsValidSetLen** (см. стр. 382 ) Проверяет набор переменную длину данных в тип данных.

**SNMPNotify** (см. стр. 332) создает и отправляет перехваты PDU .

**SNMPNotifyPrepare** (см. стр. 334 ) Собирает информацию уведомления ловушку и отправить ARP для удаленного хоста.

**SNMPSendTrap** (см. стр. 331 ) Подготовка , проверка удаленного узла , который будет получать ловушку и отправить перехваты PDU .

**SNMPSetVar** (см. стр. 332 ) Эта процедура Установите переменную MIB с запрашиваемым стоимости.

**SNMPTask** (см. стр. 372) Опросы на каждые SNMP PDU , полученных .

**Snmpv3AESDecryptRxedScopedPdu** ( см. стр. 383) Входящий SNMPv3 областью действия расшифровку PDU , используя расшифровку AES Протокол .

**Snmpv3BufferPut** ( см. стр. 385) данных Копии BYTE динамически выделяется буфер памяти .

**Snmpv3FormulateEngineID** (см. стр.389) Формулирует **snmpEngineID** (см. стр. 428) для SNMPv3 двигатель .

**Snmpv3GetAuthEngineTime** (см. стр. 390) ОбновленияSNMP переменная времени двигатель ' snmpEngineTime (см.страница 429 ) ' для SNMPv3 двигателя.

**Snmpv3GetBufferData** (см. стр. 391 ) Читает данные байт из динамически выделенный буфер памяти.

**Snmpv3MsgProcessingModelProcessPDU** (См. стр. 394 ) Эта процедура собирает или заполняет модель обработки сообщений детали из принятого запроса PDU SNMPv3 или ответ PDU соответственно.

**Snmpv3Notify** (см. стр. 395 ) создает и отправляет SNMPv3 перехваты PDU .

**Snmpv3ScopedPduProcessing** (см.страница 397) Эта процедура собирает областью действия информацию заголовка PDU от получил запрос PDU или заполняет SNMPv3 к PDU ответа соответственно.

**Snmpv3TrapScopedpdu** (см. стр. 398 ) перехваты PDU области видимости строительство заголовка PDU.

**Snmpv3UserSecurityModelProcessPDU** (См. стр. 399 ) Эта процедура собирает или заполняет модели безопасности любые параметры детали из принятого запроса PDU SNMPv3 или ответ PDU соответственно.

**Snmpv3UsmAesEncryptDecryptInitVector** (См. стр. 399 ) Шифрование AES и дешифрование инициализации вектор. (RFC 3826)

**Snmpv3UsmOutMsgAuthenticationParam** (См. стр. 400 ) Оба MD5 и SHA1 используется для исходящего сообщения аутентификации .

**Snmpv3ValidateEngineId** (см. стр. 401) Подтвердить двигателя ID .

**Snmpv3ValidateSecNameAndSecLvl** (см. стр. 401) Vaidate имя безопасности с уровнем безопасности .

**Snmpv3ValidateSecurityName** (см.страница 402) Подтвердить SNMPv3 имя пользователя или безопасности.

**SNMPValidateCommunity** (см. стр.336) Проверяет имя сообщества для контроля доступа .

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

**макрос**

**Имя Описание**

**_SNMPGetTxOffset** (См. стр. 343 ) Этот макрос _SNMPGetTxOffset .

**_SNMPSetTxOffset** (См. стр. 343 )

**AGENT_NOTIFY_PORT** (см. стр. 343 ) Этот макрос AGENT_NOTIFY_PORT .

**ASN_INT** (см. стр. 344 ) Этот макрос ASN_INT .

**ASN_NULL** (см. стр. 344 ) Этот макрос ASN_NULL .

**ASN_OID** (см. стр. 344 ) Этот макрос ASN_OID .

**DATA_TYPE_TABLE_SIZE** (см. стр. 346 )

**GET_BULK_REQUEST** (см. стр. 347 ) Этот макрос GET_BULK_REQUEST .

**GET_NEXT_REQUEST** (см. стр. 347 ) Этот макрос GET_NEXT_REQUEST .

**GET_REQUEST** (см. стр. 347 )

**GET_RESPONSE** (см. стр. 347 ) Этот макрос GET_RESPONSE .

**IS_AGENT_PDU** (см. стр. 348 ) Этот макрос IS_AGENT_PDU .

**IS_ASN_INT** (см. стр. 348 ) Этот макрос IS_ASN_INT .

**IS_ASN_NULL** (см. стр. 349 ) Этот макрос IS_ASN_NULL .

**IS_GET_NEXT_REQUEST** (см. стр. 349 ) Этот макрос IS_GET_NEXT_REQUEST .

**IS_GET_REQUEST** (см. стр. 349 ) Этот макрос IS_GET_REQUEST .

**IS_GET_RESPONSE** (см. стр. 349 ) Этот макрос IS_GET_RESPONSE .

**IS_OCTET_STRING** (см. стр. 350 ) Этот макрос IS_OCTET_STRING .

**IS_OID** (см. стр. 350 ) Этот макрос IS_OID .

**IS_SET_REQUEST** (см. стр. 351 ) Этот макрос IS_SET_REQUEST .

**IS_SNMPV3_AUTH_STRUCTURE** (см. стр. 435 ) Этот макрос IS_SNMPV3_AUTH_STRUCTURE .

**IS_STRUCTURE** (см. стр. 351 )

**IS_TRAP** (см. стр. 351 ) Этот макрос IS_TRAP .

**OCTET_STRING** (см. стр. 352 ) Этот макрос OCTET_STRING .

**OID_MAX_LEN** (см. стр. 337) Измените это , чтобы соответствовать вашему OID длину строки .

**REPORT_RESPONSE** (см. стр. 436 ) Этот макрос REPORT_RESPONSE .

**SET_REQUEST** (см. стр. 354 ) Этот макрос SET_REQUEST .

**SNMP_AGENT_PORT** (см. стр. 355 )

**SNMP_BIB_FILE_NAME** (см. стр. 355 ) Это файл , который содержит SNMP нагрудник файл . имя файла должен содержать всю верхнего регистра и должен соответствовать, что было включено в MPFS2 изображения.

**SNMP_COUNTER32** (см. стр. 356 ) Этот макрос SNMP_COUNTER32 .

**SNMP_END_OF_VAR** (см. стр. 337 ) Этот макрос SNMP_END_OF_VAR .

**SNMP_GAUGE32** (см. стр. 357 ) Этот макрос SNMP_GAUGE32 .

**SNMP_INDEX_INVALID** (см. стр. 338 ) Этот макрос SNMP_INDEX_INVALID .

**SNMP_IP_ADDR** (см. стр. 357 ) Этот макрос SNMP_IP_ADDR .

**SNMP_MAX_MSG_SIZE** (см. стр. 436 ) SNMP MIN и MAX сообщение 484 байт Как за RFC 3411 snmpEngineMaxMessageSize (см. стр. 428 ) и RFC 1157 (раздел 4 - спецификация протокола ) и реализация поддерживает более 484 всегда, когда это возможно.

**SNMP_MAX_OID_LEN_MEM_USE** (см. стр.437) Этот макрос будет использоваться , чтобы избежать SNMP OID память буфера с коррупцией

**SNMP_NMS_PORT** (см. стр. 357 ) Этот макрос SNMP_NMS_PORT .

**SNMP_NSAP_ADDR** (см. стр. 358 ) Этот макрос SNMP_NSAP_ADDR .

**SNMP_OPAQUE** (см. стр. 359 ) Этот макрос SNMP_OPAQUE .

**SNMP_START_OF_VAR** (см. стр. 337 ) Этот макрос SNMP_START_OF_VAR .

**SNMP_TIME_TICKS** ( см. стр. 359 ) Этот макрос SNMP_TIME_TICKS .

**SNMP_TRAP_COMMUNITY_MAX_LEN_MEM_USE**(См. стр. 437 ) Этот макрос будет использоваться , чтобы избежать SNMP OID память буфера с коррупцией

**SNMP_V1** (см. стр. 360 )

**SNMP_V2C** (см. стр. 360 ) Этот макрос SNMP_V2C .

**SNMP_V3** (см. стр. 437 ) Этот макрос SNMP_V3 .

**СТРУКТУРА** (см. стр. 361 )

**TRAP** (см. стр. 362 ) Этот макрос TRAP.

**TRAP_COMMUNITY_MAX_LEN** (см. стр. 338 ) Этот макрос TRAP_COMMUNITY_MAX_LEN .

**TRAP_TABLE_SIZE** (см. стр. 338 ) Этот макрос TRAP_TABLE_SIZE .

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

**структуры**

http://www.microchip.com/

**Имя Описание**

**DATA_TYPE_INFO** (см.страница 345 )

**OID_INFO** (см. стр. 353 )

**PDU_INFO** (см. стр. 353 )

**reqVarErrStatus** (см. стр.354)

**SNMP_NOTIFY_INFO** (см. стр. 358)

**SNMPNONMIBRECDINFO** (См. стр. 414 ) Это тип SNMPNONMIBRECDINFO .

**SNMPV3MSGDATA** (см.страница 417 )SNMPv3

**TRAP_INFO** (см. стр.330) Это тип TRAP_INFO .

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

**Виды**

**Имя Описание**

**SNMP_ID** (см. стр. 336 ) Это переменная ID SNMP OID . Это идентификатор присваивается через MIB файл . только динамическое и AgentID переменные могут contian ID . MIB2BIB утилита обеспечивает это правила, когда BIB был сгенерирован .

**SNMP_INDEX** (см. стр.336) Это тип SNMP_INDEX .

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

**Союзы**

**Имя Описание**

**INDEX_INFO** (см. стр.348)

**MIB_INFO** (см. стр. 352 )

**SNMP_STATUS** (см. стр.359)

**SNMP_VAL** (см. стр. 329 ) Это типа SNMP_VAL .

**описание**

• SNMP DEFS для Microchip Стек TCP / IP

http://www.microchip.com/

*

********** **********

• FileName : SNMP.h

• Зависимости: StackTsk.h

• UDP.h

• Процессор: PIC18 , PIC24F , PIC24H , dsPIC30F , dsPIC33F , PIC32

• Составитель: Microchip C32 v1.05 или выше

• Microchip C30 v3.12 или выше

• Microchip C18 v3.30 или выше

• HI-TECH PICC -18 PRO 9.63PL2 или выше

• Компания : Microchip Technology , Inc

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

### 10.16 SNTP Client

Модуль SNTP реализует Простой протокол сетевого времени . Модуль (по умолчанию) обновляет свою внутреннюю время каждый 10 минут , используя пул общественных глобальных серверов времени . Затем он вычисляет опорные раз на любой вызов **SNTPGetUTCSeconds** (см. стр. 457 ) с помощью внутреннего модуля таймера Tick .

Модуль SNTP хорош для предоставления абсолютные отметки времени . Тем не менее, это не следует полагаться для измерения времени различия (особенно небольшие различия ). Бассейн общественного серверов времени реализована с использованием циклического DNS , поэтому каждый обновление придет с другого сервера . В отличие задержки в сети и то, что эти серверы не проверено означает, что на этот раз может быть нелинейными. В то время как надежна , это не гарантирует , чтобы быть точным .

Модуль Tick обеспечивает гораздо лучшую точность ( так как это приводится в действие аппаратных часов ) и разрешение , и должны использоваться для измерения тайм-ауты и другие внутренние требования .

Разработчики могут изменить значение **NTP_SERVER** (см. стр. 461 ) , если они хотят , чтобы всегда указывают на предпочтительный сервер времени , или указать регион при доступе серверы времени . Умолчанию используется глобальный пул .

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

10.16.1 клиент SNTP открытые члены

Функции

Имя Описание

SNTPGetUTCSeconds (см. стр. 457 )Получает текущее время от модуля SNTP.

модуль

SNTP Client(см. стр. 457 )

описание

Следующие функции и переменные доступны для применения стека.

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

10.16.1.1 SNTPGetUTCSeconds Функция

файл

SNTP.h

C

DWORD **SNTPGetUTCSeconds**();

**Возвращает**

С начала века Количество секунд . (По умолчанию 01- янв -1970 00:00:00 )

**описание**

Эта функция получает текущее время как сообщает модуля SNTP. Используйте это значение для абсолютного времени штамповки . Значение Возвращается (по умолчанию ) число секунд, прошедших с 01 -Jan - 1970 00:00:00 .

**Замечания**

Не используйте эту функцию для измерения разности времени . Модуль Tick больше подходит для этих требований .

**Предпосылки**

Ни один

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

### 10.16.2 клиент SNTP Стек Пользователи

**Функции**

**Имя Описание**

**SNTPClient** (см. стр. 458 ) Периодически проверяет текущее время из пула серверов.

**модуль**

Клиент SNTP (см. стр. 457 )

**описание**

Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные .

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

### 10.16.2.1 SNTPClient Функция

**файл**

SNTP.h

C

**void SNTPClient**();

**Возвращает**

Ни один

**описание**

Эта функция периодически проверяет пул серверов времени , чтобы получить текущую дату / время .

**Замечания**

Эта функция требует только доступный UDP сокет при обработке , но освобождает этот сокет , когда модуль SNTP простаивает.

**Предпосылки**

UDP инициализации .

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

### 10.16.3 клиент SNTP Внутренние Пользователи

**макрос**

**Имя Описание**

**NTP_EPOCH** (см. стр. 461 ) Ссылка Epoch использовать . (по умолчанию: 01- янв -1970 00:00:00 )

**NTP_FAST_QUERY_INTERVAL**(См. Стр. 461 ) Определяет , как долго ждать , чтобы повторить попытку обновления после сбоя. Обновления могут принимать до 6 секунд на провал , так что это 14 -секундная задержка на самом деле только 8 -й повторите попытку.

**NTP_QUERY_INTERVAL** (см. стр. 461 ) Определяет, как часто для повторной синхронизации даты / времени (по умолчанию : 10 минут )

**NTP_REPLY_TIMEOUT** (см.страница 461 ) Определяет , как долго ждать , прежде чем приступить запрос не удалось

**NTP_SERVER** (см. стр. 461 ) Это обычно доступные серверы времени сеть . Фактический IP вернулся из бассейна будет меняться каждую минуту для того, чтобы распределить нагрузку по всему слою 1 временщики . Для лучшей точности и сетевые накладные расходы , вы должны найти сервер бассейн, самый близкий к вашему географии , но это все равно будет работать , если вы используете глобальный адрес pool.ntp.org или выбрать не тот или отправить ваш встроенное устройство на другой географии.

**NTP_SERVER_PORT** (см.страница 462) Порт для связи с серверами NTP

**модуль**

Клиент SNTP (см. стр. 457 )структуры

**Имя Описание**

**NTP_PACKET** (см. стр.459) Определяет структуру пакета NTP

**Переменные**

**Имя Описание**

**dwLastUpdateTick** (см.страница 460 ) Отметьте кол последнего обновления

**dwSNTPSeconds** (см.страница 460 ) Секунды значение, полученное последнего обновления

**описание**

Следующие функции и переменные определяются как внутренние, в модуль SNTP Client .

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

### 10.16.3.1 NTP_PACKET Структура

**файл**

SNTP.c

C

```
typedef struct {
struct {
BYTE mode : 3;
BYTE versionNumber : 3;
BYTE leapIndicator : 2;
} flags;
BYTE stratum;
CHAR poll;
CHAR precision;
DWORD root_delay;
DWORD root_dispersion;
DWORD ref_identifier;
DWORD ref_ts_secs;
DWORD ref_ts_fraq;
DWORD orig_ts_secs;
DWORD orig_ts_fraq;
DWORD recv_ts_secs;
DWORD recv_ts_fraq;
DWORD tx_ts_secs;
DWORD tx_ts_fraq;
} NTP_PACKET;
```

**Пользователи**

**Пользователи Описание**

```
struct {
BYTE mode : 3;
BYTE versionNumber : 3;
BYTE leapIndicator : 2;
} flags;
```

**Флаги для пакета**

**BYTE mode** : 3; режим NTP

**BYTE versionNumber**: 3 ; SNTP номер версии

**BYTE leapIndicator** : 2 ; Прыжок второй индикатор

**BYTE stratum** ; Уровень Stratum местного часы

**CHAR poll**; интервал Опрос

**CHAR precision**; Точность ( секунд ближайшей степени 2 )

**DWORD root_delay** ; Корневая задержка между локальной машине и сервере

**DWORD root_dispersion** ; Корневая дисперсии ( максимальная ошибка )

**DWORD ref_identifier** ; Идентификатор Опорная частота

**DWORD ref_ts_secs** ; Ссылка : время ( в секундах)

**DWORD ref_ts_fraq** ; Ссылка : время ( фракции )

**DWORD orig_ts_secs** ; Возникновение метка времени ( в секундах)

**DWORD orig_ts_fraq** ; Возникновение метка времени ( фракции )

**DWORD recv_ts_secs** ; Момент, в который запрос прибыл в отправителя (секунд)

**DWORD recv_ts_fraq** ; Момент, в который запрос прибыл в отправителя ( фракции )

**DWORD tx_ts_secs** ; Момент, в который запрос оставили отправителя (секунды )

**DWORD tx_ts_fraq** ; Момент, в который запрос оставили отправитель ( фракции )

**описание**

Определяет структуру пакета NTP

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

### 10.16.3.2 dwLastUpdateTick Переменная

**файл**

SNTP.c

C

DWORD **dwLastUpdateTick** = 0;

**описание**

Отметьте кол последнего обновления

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

### 10.16.3.3 dwSNTPSeconds Переменная

**файл**

SNTP.c

C

DWORD **dwSNTPSeconds** = 0;

**описание**

Секунды значение, полученное последнего обновления

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

### 10.16.3.4 NTP_EPOCH Макро

**файл**

SNTP.c

C

**#define NTP_EPOCH** (86400ul * (365ul * 70ul + 17ul))

**описание**

Ссылка Эпоха использовать . (по умолчанию: 01- янв -1970 00:00:00 )

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

### 10.16.3.5 NTP_FAST_QUERY_INTERVAL Макро

**файл**

SNTP.c

C

**#define NTP_FAST_QUERY_INTERVAL** (14ull * TICK_SECOND)

**описание**

Определяет , как долго ждать , чтобы повторить попытку обновления после сбоя. Обновления может занять до 6 секунд на провал , так что это 14 -секундная задержка на самом деле только 8 -й повтора .

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

**10.16.3.6 NTP_QUERY_INTERVAL Макро**

**файл**

SNTP.c

C

**#define NTP_QUERY_INTERVAL** (10ull*60ull * TICK_SECOND)

**описание**

Определяет, как часто для повторной синхронизации даты / времени (по умолчанию : 10 минут )

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

**10.16.3.7 NTP_REPLY_TIMEOUT Макро**

**файл**

SNTP.c

C

**#define NTP_REPLY_TIMEOUT** (6ul*TICK_SECOND)

**описание**

Определяет , как долго ждать , прежде чем приступить запрос не удалось

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

**10.16.3.8 NTP_SERVER Макро**

**файл**

SNTP.c

C

**#define NTP_SERVER** "pool.ntp.org"

**описание**

Они, как правило доступные серверы времени сеть . Фактическое IP возвращается из пула будут меняться каждую минуту с тем, чтобы распределить нагрузку по всему слою 1 временщиков . Для лучшей точности и сетевые накладные расходы , вы должны найти сервер бассейн ближайший к вашему географии , но это все равно будет работать , если вы используете глобальный адрес pool.ntp.org или выбрать не тот или судно Ваш встроенное устройство к другому географии.

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

**10.16.3.9 NTP_SERVER_PORT Макро**

**файл**

SNTP.c

C

**#define NTP_SERVER_PORT** (123ul)

**описание**

Порт для связи с серверами NTP

**10.17 SSL**

**Файлы**

**Имя Описание**

SSLClientSize.h (см. стр. 525 ) Это файл SSLClientSize.h .

**описание**

Модуль SSL добавлена поддержка шифрования в TCP слоя путем внедрения протокола SSLv3 . Этот протокол является стандарт для безопасной связи через Интернет , а также предотвращает слежка или фальсификации данных , как он проходит через

ненадежной сетью . Эта реализация SSL поддерживает протокол RSA асимметричного шифрования и Arcfour симметричная протокол шифрования .

Длина ключа компилятора сервера SSL Максимальная RSA ( SSL_RSA_KEY_SIZE (см. Стр. 471 ) ) и Длина ключа клиента SSL Максимальная RSA ( SSL_RSA_CLIENT_SIZE (см. Стр. 471 ) )

C18 1024 1024

C30 1024 1024

C32 2048 2048

В соответствии с ограничениями США экспортного контроля , шифрование часть модуля SSL необходимо приобрести отдельно от Microchip . Библиотека шифрования данных подпрограмм ( SW300052 ) доступен за символическую плату от http://www.microchipdirect.com/productsearch.aspx?Keywords=SW300052 .

Более новые версии TCP / IP стека включают изменения в модуль SSL для обеспечения более широкий спектр ключа RSA длины . Это позволит Стек TCP / IP для поддержки более широкого спектра клиентских SSL / серверов . Из-за этих изменений , вы можете

нужно вручную изменить вашу копию RSA.c или получить обновленную копию от MicrochipDirect . В следующей таблице перечисляет которые криптографических дополнения требуется для какой версии стека.

Обратите внимание, что SW300052 v2.6 содержит дополнения для Stack v5.31 .

TCP / IP стек версия Дополнение для Stack v5.31 Дополнение для Stack v5.31 с ручными изменениями 5.36.2 или старший

**Совместимость . Несовместимые .**

| TCP/IP Stack Version | Add-on for Stack v5.31 | Add-on for Stack v5.31 with manual modifications |
|---|---|---|
| 5.36.2 or older | Compatible. | Incompatible. |
| 5.36.4 | Incompatible. | Compatible. |
| 5.41 | Incompatible. | Compatible. **(Note)** |
| 5.42 | Incompatible. | Compatible. **(Note)** |

5.36.4 несовместимы. Совместимость .

5.41 Несовместимые . Совместимость . (Примечание)

5.42 Несовместимые . Совместимость . (Примечание)

**Примечание**: ручные изменения для стека v5.31 будет работать правильно для Stack v5.41 и v5 , 42 для SSL. Тем не менее, при использовании Модуль SNMP , после шифрования Дополнений установки вам нужно будет вручную скопировать файл AES_PIC32MX.a AES библиотеки от " <Microchip Solutions> / Microchip / Crypto " до " <Microchip Solutions> / Microchip / TCPIP стека . "

Перечень необходимых изменений ( " Требуется SSL changes.pdf " ) можно найти в микрочип приложений библиотек установки каталог , в \ Microchip \ Помощь \ Дополнительный TCPIP Помощь подкаталоге.

**Поддержка Клиента SSL**

Клиент SSL может быть начато , сначала открыв соединение TCP , то вызов TCPStartSSLSession инициировать SSL рукопожатие процесс . Рукопожатие использует открытый ключ из сертификата , предоставляемой сервером . Ключом длиной до 1024

бит поддерживаются на всех процессорах ; длина ключа до 2048 бит поддерживаются на PIC32 микроконтроллеров .

SSL_RSA_CLIENT_SIZE (см. стр. 471 ) макро в SSLClientSize.h (см. стр. 525 ) устанавливает максимальное Длина сертификат ключа , который должен обрабатывать ваш клиент .

# define SSL_RSA_CLIENT_SIZE ( 1024ul ) / / Размер Encryption буфера ( должен быть больше чем размер ключа )

После того, как рукопожатие начал , звоните TCPSSLIsHandshaking (см. стр. 468 ) , пока она не возвращает значение ЛОЖЬ. Это будет означать, что рукопожатие завершила и весь трафик теперь крепится с помощью 128-битного шифрования Arcfour . Если рукопожатие не выполняется для какой-либо причине , соединение TCP автоматически прекращается в соответствии с требованиями спецификации протокола SSL .

Для повышения производительности , модуль SSL кэширует параметры безопасности для самых недавно сделанных соединений. Это позволяет быстрые пересоединения в том же узле без вычислительной счет другого RSA рукопожатия . По умолчанию , эти два Последние соединения кэшируются , но это может быть изменено в TCPIPConfig.h .

Поддержка клиента SSL уже включена для SMTP. Когда STACK_USE_SSL_CLIENT определен, SMTP модулю автоматически добавляет поле в SmtpClient (см. стр. 304 ) называется UseSSL . Это поле управляет или нет SMTP Клиентский модуль будет пытаться установить соединение SSL перед передачей данных.

Обратите внимание, что TCP сокетов с использованием SSL может потребовало увеличения размера буфера TX / RX для поддержки SSL. Вы можете настроить размер

Ваши розетки с помощью утилиты конфигурирования TCP / IP входит в стек.

**Поддержка сервера SSL**

Чтобы начать сервер SSL , сначала открыть сокет TCP для прослушивания с помощью TCPOpen (см. стр. 542 ) . Тогда звоните TCPAddSSLListener (см. стр. 468 ) для прослушивания (см. стр. 175) для входящих соединений SSL на другой порт . это

позволяетодин разъем для совместного использования ресурсов на уровне приложений и слушать (см. стр. 175) для соединений на двух разных порты. Соединения , происходящие на первоначально открыт порт будет продолжаться необеспеченных, в то время как соединения на SSL порт будет Первый завершить SSL рукопожатия , чтобы обеспечить данные.

Если ваше приложение не будет принимать (см. стр. 169 ) необеспеченный трафик , просто открыть не обеспеченных гнездо на правах свободного порта число, то убедитесь, что каждый входящий надежности соединения ( не на этом порту ) , позвонив TCPIsSSL (см. стр. 469 ) .

Поддержка сервера SSL включается автоматически для http2 когда STACK_USE_SSL_SERVER определяется . По умолчанию Http2 модуль будет слушать (см. стр. 175) для необеспеченных трафика на порту 80 и защищенным соединением на порт 443.

Эта реализация сервер SSL поддерживает ключи длиной до 1024 бит на большинстве ПИК микроконтроллеров и 2048 бит на PIC32 микроконтроллеры . SSL_RSA_KEY_SIZE (см. стр. 471) макро в TCPIPConfig.h задает ключ сертификата сервера Длина .

/ / Биты в ключ SSL RSA . Этот параметр используется для SSL разорвать

/ / Соединения только .

# define SSL_RSA_KEY_SIZE ( 512ul )

Обратите внимание, что TCP сокетов с использованием SSL может потребовало увеличения размера буфера TX / RX для поддержки SSL. Вы можете настроить размер Ваши розетки с помощью утилиты конфигурирования TCP / IP входит в стек.

**Недостатки**

SSL был разработан для настольных ПК с более быстрыми процессорами и значительно больше ресурсов , чем доступно на встраиваемых платформ . Несколько компромиссы должны быть сделаны для того , чтобы использовать SSL в менее ресурсоемких образом.

Клиентский модуль SSL не выполняет никакой проверки или проверки сертификатов . Это потребует много корень сертификаты будут храниться локально для проверки , которая не является возможным для памяти ограниченной части. Это не идет на компромисс

настройки безопасностисоединение было установлено , но не обеспечивает полную безопасность против человека - в- середине атак. ( Такого рода атаки редко и было бы трудно выполнить . )

Ни клиент SSL , ни сервер может полностью проверить ПДК до обработки данных. SSL записи включают в себя подпись на убедиться, что сообщения не были изменены в процессе передачи. Это аутентификации сообщения ( см. стр. 89) кодекс, или MAC, вставляется

после по крайней мере каждый 16kB трафика . (Он обычно вставляется гораздо чаще , чем это. ) Без 16kB оперативной памяти в буфер пакеты для каждого гнезда , входящие данные должны быть переданы на прикладном уровне , прежде чем MAC может быть полностью проверена.

Недействительные ПДК будет по-прежнему вызывать соединение немедленно прекратить , но к тому времени это будет обнаружено некоторые плохие данные могут уже достигли

приложение. Поскольку ARCFOUR шифр используется это потоковый шифр , было бы трудно использовать это в любом значимым образом. Злоумышленник не сможет контролировать то, что данные на самом деле изменении или вставке , так как это

без знания ключа даст мусор. Тем не менее, важно понимать, что входные данные не

полностью проверяется перед передачей приложения.

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

### 10.17.1 Генерирующие Сертификаты сервера

**модуль**

SSL (см. стр. 462 )

**описание**

Сертификаты SSL , используемые модулем SSL TCP / IP стека хранятся в исходном файле CustomSSLCert.c . Следующий ряд шагов описывают, как создавать структуры в CustomSSLCert.c с помощью сертификата SSL .

1 . Загрузите и установите библиотеку OpenSSL . Есть несколько сторонних сайтов, которые предлагают SSL инсталляторов (например,

http://www.slproweb.com/products/Win32OpenSSL.html ) . Обратите внимание, что некоторые дистрибутивы не могут включать в себя все команды

указано в документации OpenSSL .

2 . Откройте консоль и изменить каталог в / папки бин OpenSSL .

3 . Если у вас нет ключа и сертификата , вы можете создать их в первую очередь . В следующем примере консольные команды будет

генерировать 512 -битный ключ в :

1 . Генерация ключа : OpenSSL genrsa выезда 512bits.key 512

2 . Создайте запроса на подпись сертификата ( CSR ) . Вам нужно будет добавить дополнительную информацию в ответ на запрос : OpenSSL

REQ- новый ключ 512bits.key выезда 512bits.csr

3 .Генерация сертификата X.509 , если самостоятельно подписание ( или отправить КСО к сертификации для подписания ) : OpenSSL x509 -REQ

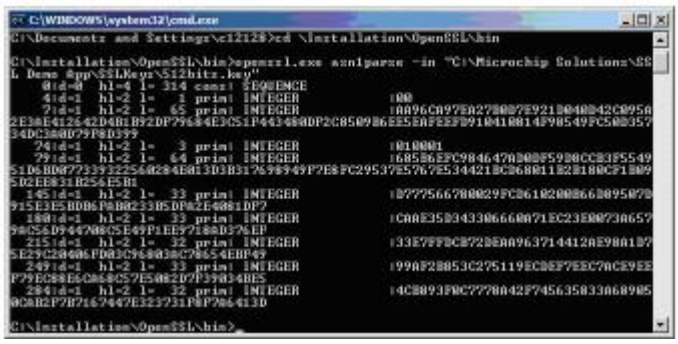
- дней 365 - в 512bits.csr - signkey 512bits.key выезда 512bits.crt (обратите внимание , что если параметр- дней не указан,

срок действия по умолчанию составляет 30 дней )

4 . Для получения дополнительной документации , обратитесь к http://www.openssl.org/docs/apps/openssl.html .

4. Разбор ключевой файл с помощью команды : openssl.exe asn1parse - в " [ каталог, содержащий ключ ] \ 512bits.key "

5. Вы должны увидеть экран:




6 . Если вы не используете ENCX24J600 семейный устройство , то последние 5 целые отображаются здесь являются SSL_P , SSL_Q , SSL_dP , SSL_dQ и SSL_qInv параметры , соответственно. Тем не менее, они выводятся здесь в формате обратного порядка , и

реализация криптографической библиотеки Microchip требуется параметры в прямой порядок байтов , так что вам придется ввести параметры в C массивов в обратном порядке . Например , целое со смещением 145 :

145:d=1 h1=2 ;= 33 prim: INTEGER
:D7775667800 29FCD610200B66D89507D
915E3E5BDB6FAB0233B5DFA2E4081DF7

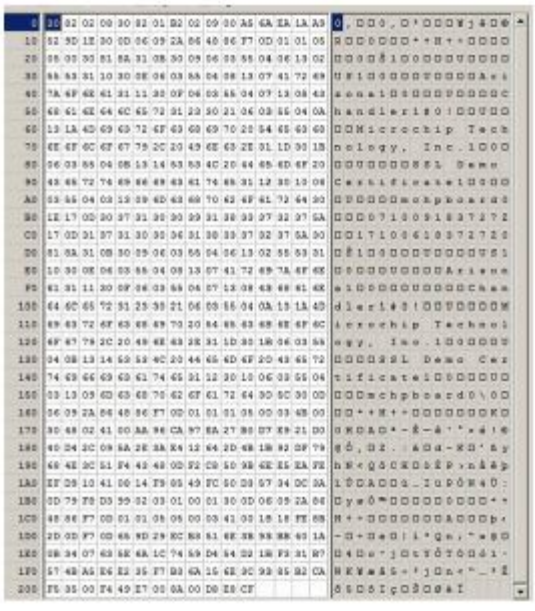
будут обменены в файле CustomSSLCert.c :

```
ROM BYTE SSL_P[] = {
0xF7, 0x1D, 0x08, 0xE4, 0xA2, 0xDF, 0xB5, 0x33,
0x02, 0xAB, 0x6F, 0xDB, 0x5B, 0x3E, 0x5E, 0x91,
0x7D, 0x50, 0x89, 0x6D, 0xB6, 0x00, 0x02, 0x61,
0xCD, 0x9F, 0x02, 0x80, 0x67, 0x56, 0x77, 0xD7
};
```

7. Если вы используете ENCX24J600 семейный устройство, затем второй и четвертый целые отображаются здесь являются SSL_N и SSL_D параметры , соответственно. Там нет необходимости , чтобы сделать порядка байтов изменение формата для этих параметров . Для примера , ожидаемые значения SSL_N и SSL_D показаны на рисунке ниже:

8. Разбор свой сертификат X.509 с помощью команды : openssl.exe asn1parse - в " [ директория, содержащая Дрессировка ] \ 512bits.crt " выезда cert.bin

9. Откройте выходной файл cert.bin в шестнадцатеричном редакторе . Например , вот информация сертификата по умолчанию генерируется из



512bits.crt приведены в стеке :

10. Эта информация должна быть копируются в SSL_CERT (см. стр. 495 ) [] массива. Заметим, что это двоичные данные ( небольшое целое число ), поэтому он не получает обратный порядок байт - местами , как частных ключевых параметров .

```
ROM BYTE SSL_CERT[524] = {
0x30, 0x82, 0x02, 0x08, 0x30, 0x82, 0x01, 0xb2, 0x02, 0x09, 0x00, 0xa5, 0x6a, 0xea, 0x1a, 0xa9,
0x52, 0x9d, 0x1e, 0x30, 0x0d, 0x06, 0x09, 0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d, 0x01, 0x01, 0x05,
0x05, 0x00, 0x30, 0x81, 0x8a, 0x31, 0x0b, 0x30, 0x09, 0x06, 0x03, 0x55, 0x04, 0x06, 0x13, 0x02,
0x55, 0x53, 0x31, 0x10, 0x30, 0x0e, 0x06, 0x03, 0x55, 0x04, 0x08, 0x13, 0x07, 0x41, 0x72, 0x69,
0x7a, 0x6f, 0x6e, 0x61, 0x31, 0x11, 0x30, 0x0f, 0x06, 0x03, 0x55, 0x04, 0x07, 0x13, 0x08, 0x43,
0x68, 0x61, 0x6e, 0x64, 0x6c, 0x65, 0x72, 0x31, 0x23, 0x30, 0x21, 0x06, 0x03, 0x55, 0x04, 0x0a,
0x13, 0x1a, 0x4d, 0x69, 0x63, 0x72, 0x6f, 0x63, 0x68, 0x69, 0x70, 0x20, 0x54, 0x65, 0x63, 0x68,
0x6e, 0x6f, 0x6c, 0x6f, 0x67, 0x79, 0x2c, 0x20, 0x49, 0x6e, 0x63, 0x2e, 0x31, 0x1d, 0x30, 0x1b,
0x06, 0x03, 0x55, 0x04, 0x0b, 0x13, 0x14, 0x53, 0x53, 0x4c, 0x20, 0x44, 0x65, 0x6d, 0x6f, 0x20,
0x43, 0x65, 0x72, 0x74, 0x69, 0x66, 0x69, 0x63, 0x61, 0x74, 0x65, 0x31, 0x12, 0x30, 0x10, 0x06,
0x03, 0x55, 0x04, 0x03, 0x13, 0x09, 0x6d, 0x63, 0x68, 0x70, 0x62, 0x6f, 0x61, 0x72, 0x64, 0x30,
0x1e, 0x17, 0x0d, 0x30, 0x37, 0x31, 0x30, 0x30, 0x39, 0x31, 0x38, 0x33, 0x37, 0x32, 0x37, 0x5a,
0x17, 0x0d, 0x31, 0x37, 0x31, 0x30, 0x30, 0x36, 0x31, 0x38, 0x33, 0x37, 0x32, 0x37, 0x5a, 0x30,
0x81, 0x8a, 0x31, 0x0b, 0x30, 0x09, 0x06, 0x03, 0x55, 0x04, 0x06, 0x13, 0x02, 0x55, 0x53, 0x31,
0x10, 0x30, 0x0e, 0x06, 0x03, 0x55, 0x04, 0x08, 0x13, 0x07, 0x41, 0x72, 0x69, 0x7a, 0x6f, 0x6e,
```

0x61, 0x31, 0x11, 0x30, 0x0f, 0x06, 0x03, 0x55, 0x04, 0x07, 0x13, 0x08, 0x43, 0x68, 0x61, 0x6e,
0x64, 0x6c, 0x65, 0x72, 0x31, 0x23, 0x30, 0x21, 0x06, 0x03, 0x55, 0x04, 0x0a, 0x13, 0x1a, 0x4d,
0x69, 0x63, 0x72, 0x6f, 0x63, 0x68, 0x69, 0x70, 0x20, 0x54, 0x65, 0x63, 0x68, 0x6e, 0x6f, 0x6c,
0x6f, 0x67, 0x79, 0x2c, 0x20, 0x49, 0x6e, 0x63, 0x2e, 0x31, 0x1d, 0x30, 0x1b, 0x06, 0x03, 0x55,
0x04, 0x0b, 0x13, 0x14, 0x53, 0x53, 0x4c, 0x20, 0x44, 0x65, 0x6d, 0x6f, 0x20, 0x43, 0x65, 0x72,
0x74, 0x69, 0x66, 0x69, 0x63, 0x61, 0x74, 0x65, 0x31, 0x12, 0x30, 0x10, 0x06, 0x03, 0x55, 0x04,
0x03, 0x13, 0x09, 0x6d, 0x63, 0x68, 0x70, 0x62, 0x6f, 0x61, 0x72, 0x64, 0x30, 0x5c, 0x30, 0x0d,
0x06, 0x09, 0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d, 0x01, 0x01, 0x01, 0x05, 0x00, 0x03, 0x4b, 0x00,
0x30, 0x48, 0x02, 0x41, 0x00, 0xaa, 0x96, 0xca, 0x97, 0xea, 0x27, 0xb0, 0xd7, 0xe9, 0x21, 0xd0,
0x40, 0xd4, 0x2c, 0x09, 0x5a, 0x2e, 0x3a, 0xe4, 0x12, 0x64, 0x2d, 0x4b, 0x1b, 0x92, 0xdf, 0x79,
0x68, 0x4e, 0x3c, 0x51, 0xf4, 0x43, 0x48, 0x0d, 0xf2, 0xc8, 0x50, 0x9b, 0x6e, 0xe5, 0xea, 0xfe,
0xef, 0xd9, 0x10, 0x41, 0x08, 0x14, 0xf9, 0x85, 0x49, 0xfc, 0x50, 0xd3, 0x57, 0x34, 0xdc, 0x3a,
0x0d, 0x79, 0xf8, 0xd3, 0x99, 0x02, 0x03, 0x01, 0x00, 0x01, 0x30, 0x0d, 0x06, 0x09, 0x2a, 0x86,
0x48, 0x86, 0xf7, 0x0d, 0x01, 0x01, 0x05, 0x05, 0x00, 0x03, 0x41, 0x00, 0x18, 0x18, 0xfe, 0x8b,
0x2d, 0x0d, 0xf7, 0x0d, 0x65, 0x9d, 0x29, 0xec, 0xb3, 0x51, 0x6e, 0x3b, 0x93, 0xbb, 0x40, 0x1a,
0x0b, 0x34, 0x07, 0x63, 0x5e, 0x6a, 0x1c, 0x74, 0x59, 0xd4, 0x54, 0xd2, 0x1b, 0xf3, 0x31, 0xb7,
0x57, 0x4b, 0xa5, 0xe6, 0xe2, 0x35, 0xf7, 0xb3, 0x6a, 0x15, 0x6e, 0x3c, 0x93, 0x85, 0xb2, 0xca,
0xf5, 0x35, 0x00, 0xf4, 0x49, 0xe7, 0x00, 0x8a, 0x00, 0xd8, 0xe8, 0xcf
10.17 SSL Microchip TCP/IP Stack Help Generating Server Certificates
466
};

11. Обновите SSL_CERT_LEN (см. стр. 495 ) переменную , чтобы содержать правильное значение.

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

**10.17.2 SSL Общественная Пользователи**

**Перечисления**

**Имя Описание**

SSL_SUPPLEMENTARY_DATA_TYPES

(См. стр. 470 )

Это тип SSL_SUPPLEMENTARY_DATA_TYPES .

**Функции**

**Имя Описание**

**TCPAddSSLListener** (см.страница 468 ) Слушает для подключения SSL на определенный порт .

**TCPSSLIsHandshaking** (см. стр. 468 ) Определяет,SSL сеанс еще рукопожатия .

**TCPStartSSLClient** (см.страница 469 ) Начинает клиента SSL сеанс .

**TCPIsSSL** (см. стр. 469 ) Определяет, является ли соединение TCP обеспечивается с SSL.

**SSLStartSession** (см.страница 470 ) Начинается новый сеанс SSL для данного соединения TCP .

**макрос**

**Имя Описание**

**SSL_INVALID_ID** (см.страница 467 )Идентификатор недействительных ассигнований SSL

**SSL_RSA_KEY_SIZE** (см.страница 471 )Биты в ключ SSL RSA . Этот параметр используется для SSL разорвать только подключения .

Единственным допустимым значением является 512 бит (768 и 1024 бит не работают в это время ) .

Однако следует отметить, что клиентские SSL операции у в настоящее время работают до 1024 бит RSA длина ключа .

**SSL_RSA_CLIENT_SIZE** (см. стр. 471) Размер Encryption буфера ( должен быть больше, чем размер ключа )

**модуль**

**SSL** (см. стр. 462 )

**структуры**

**Имя Описание**

**SSL_PKEY_INFO** (см.страница 470 ) Для хэш информации ключа , нам нужно на самом деле получить открытый ключ информация ... 1024 битный ключ на 8 бит / байт = 128 байт , необходимых для открытого ключа.

**описание**

Следующие функции и переменные доступны для применения стека.

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

**10.17.2.1 SSL_INVALID_ID Макро**

**файл**

SSL.h

C

**#define SSL_INVALID_ID** (0xFFu) / / Идентификатор недействительных ассигнований SSL

**описание**

Идентификатор недействительных ассигнований SSL

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

**10.17.2.2 TCPAddSSLListener Функция**

**файл**

TCP.h

C

BOOL **TCPAddSSLListener**(
TCP_SOCKET **hTCP**,
WORD **port**
);

**описание**

Эта функция добавляет дополнительный порт прослушивания в связи TCP . Соединения , сделанные на этом альтернативном порту будут обеспечены через SSL.

**Предпосылки**

TCP инициализации и HTCP слушает.

**Параметры**

**Параметры Описание**

Подключение HTCP TCP , чтобы обеспечить порт SSL порт для прослушивания (см. стр. 75 ) на

**Возвращаемые значения**

**Возвращаемые значения Описание**

Был добавлен

**ИСТИНА** порт SSL .

**ЛОЖЬ** сокет не был слушающий сокет .

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

**10.17.2.3 TCPSSLIsHandshaking Функция**

**файл**

TCP.h

C

BOOL **TCPSSLIsHandshaking**(
TCP_SOCKET **hTCP**
);
**описание**

Эта функция вызывается после вызова TCPStartSSLClient ( см. стр. 469 ), пока ЛОЖЬ возвращается . Ваше приложение может продолжить его нормальной передачи данных ( которая в настоящее время обеспечено ) .

**Предпосылки**

TCP инициализации и HTCP подключен .

**Параметры**

**Параметры Описание**

Подключение HTCP TCP проверить

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** SSL рукопожатия все еще прогрессирует

**ЛОЖЬ** SSL рукопожатия завершила

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

**10.17.2.4 TCPStartSSLClient Функция**

**файл**

TCP.h

C

```
BOOL TCPStartSSLClient(
TCP_SOCKET hTCP,
BYTE* host
);
```

**описание**

Эта функция возрастает текущего подключения к защищенному соединению SSL по инициирования клиентского SSL рукопожатия .

**Замечания**

Параметр хозяин в настоящее время игнорируются и не проверяется .

**Предпосылки**

TCP инициализации и HTCP уже подключен.

**Параметры**

**Параметры Описание**

Подключение HTCP TCP , чтобы обеспечить

пройдет ожидаемый имя хоста на сертификат (в настоящее время игнорируются)

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** соединениеSSL был инициирован

**ЛОЖЬ** Недостаточные ресурсы SSL ( заглушки ) были доступны

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

**10.17.2.5 TCPIsSSL Функция**

**файл**

TCP.h

C

```
BOOL TCPIsSSL(
TCP_SOCKET hTCP
);
```

**описание**

Вызовите эту функцию , чтобы определить, является ли соединение TCP обеспечены SSL.

**Предпосылки**

TCP инициализации и HTCP подключен .

**Параметры**

**Параметры Описание**

Подключение HTCP TCP проверить

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** соединения обеспечивается с помощью SSL

**ЛОЖЬ** соединения не защищен

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

**10.17.2.6 SSLStartSession Функция**

**файл**

SSL.h

C

```
BYTE SSLStartSession(
TCP_SOCKET hTCP,
void * buffer,
BYTE supDataType
);
```

**описание**

Начинается новый сеанс SSL для данного соединения TCP .

**Предпосылки**

SSL был инициализирован и HTCP подключен .

**Параметры**

**Параметры Описание**

HTCP сокет начать подключение SSL на буфер указатель на буфер данных дополнительного

supDataType тип дополнительных данных для хранения

**Возвращаемые значения**

**Возвращаемые значения Описание**

SSL_INVALID_ID (см. стр. 467 ) недостаточные ресурсы SSL , чтобы начать новое соединение другие выделено SSL заглушки ID

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

**10.17.2.7 SSL_SUPPLEMENTARY_DATA_TYPES Перечисление**

**файл**

SSL.h

C

```
typedef enum {
SSL_SUPPLEMENTARY_DATA_NONE = 0,
SSL_SUPPLEMENTARY_DATA_CERT_PUBLIC_KEY
} SSL_SUPPLEMENTARY_DATA_TYPES;
```

**описание**

Это тип SSL_SUPPLEMENTARY_DATA_TYPES .

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

**10.17.2.8 SSL_PKEY_INFO Структура**

**файл**

SSL.h

C

```
typedef struct {
WORD pub_size_bytes;
BYTE pub_key[SSL_RSA_CLIENT_SIZE/8];
BYTE pub_e[3];
BYTE pub_guid;
} SSL_PKEY_INFO;
```

**Пользователи**

**Пользователи Описание**

BYTE pub_guid ; Это используется как TCP_SOCKET (см. стр. 553 ), который является BYTE

**описание**

Для хэш информации ключа , нам нужно на самом деле получить информацию открытого ключа ... 1024 -битный ключ в 8 бит / байт = 128 байт требуется для открытого ключа.

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

**10.17.2.9 SSL_RSA_KEY_SIZE Макро**

**файл**

TCPIP MRF24W.h

C

```
#define SSL_RSA_KEY_SIZE (512ul)
```

**описание**

Биты в ключ SSL RSA . Этот параметр используется для SSL разорвать только подключения . Единственным допустимым значением является 512 бит ( 768 и 1024 бит не работают в это

время ) . Однако следует отметить, что клиентские SSL операции у в настоящее время работают до 1024 бит длиной ключа RSA .

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

**10.17.2.10 SSL_RSA_CLIENT_SIZE Макро**

**файл**

SSLClientSize.h (см. Стр. 525 )

C

**#define SSL_RSA_CLIENT_SIZE** (1024ul)/ / Размер Encryption буфера ( должен быть больше, чем размер ключа )

**описание**

Размер Encryption буфера ( должен быть больше, чем размер ключа )

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

**10.17.3 SSL Stack Пользователи**

**Перечисления**

**Имя Описание**

**SSL_STATE** (см. стр.472)Это тип SSL_STATE .

**Функции**

**Имя Описание**

**SSLInit** (см. стр. 473 ) Инициализирует SSL двигатель.

**SSLPeriodic** (см. стр.473) Выполняет любые периодические задачи для модуля SSL .

**TCPRequestSSLMessage** (см. стр. 473 ) Просит сообщение SSL должен быть передан .

**TCPSSLGetPendingTxSize**(См. Стр. 474 ) Определяет , сколько байтов в ожидании для будущей записи SSL .

**TCPSSLHandleIncoming** (см. стр. 474 ) Руки вновь приходим данные TCP к модулю SSL для обработки.

**TCPSSLHandshakeComplete**(См. Стр. 475 ) Очищает флаг SSL рукопожатия .

**TCPSSLInPlaceMACEncrypt**(См. Стр. 475 ) Шифрование и ПДК данные в месте в буфере TCP TX.

**TCPSSLPutRecordHeader** (см. стр. 476) Записывает заголовок SSL записи и отправляет запись SSL .

**TCPStartSSLServer** (см.страница 477) Начинает сеанс сервера SSL .

**макрос**

**Имя Описание**

**SSL_MIN_SESSION_LIFETIME** (см. стр. 477) Минимальный срок службы для SSL Сессии Сессии не может быть перераспределены до это много времени прошло

**SSL_RSA_LIFETIME_EXTENSION**(См. Стр. 477 ) Продление срока службы для RSA операции Sessions жизни продлевается на эта сумма , когда расчет RSA сделан

**модуль**

**SSL** (см. стр. 462 )

**описание**

Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные .

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

**10.17.3.1 SSL_STATE Перечисление**

**файл**

TCP.h

C

```
typedef enum {
SSL_NONE = 0,
SSL_HANDSHAKING,
SSL_ESTABLISHED,
SSL_CLOSED
} SSL_STATE;
```

**Пользователи**

**Пользователи Описание**

**SSL_NONE** = 0 Обеспечение не включена

**SSL_HANDSHAKING** Рукопожатие прогрессирует (данные приложения не допускается)

**SSL_ESTABLISHED** Соединение устанавливается и закрепляется

**SSL_CLOSED** соединения был закрыт ( не applicaiton данных не допускается )

**описание**

Это тип SSL_STATE .

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

**10.17.3.2 SSLInit Функция**

**файл**

SSL.h

C

**void SSLInit**();

**Возвращает**

Ни один

**описание**

Инициализация SSL двигатель.

**Замечания**

Эта функция вызывается только один при жизни приложения.

**Предпосылки**

Ни один

**раздел**

Прототипы функций

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

### 10.17.3.3 SSLPeriodic Функция

**файл**

SSL.h

C

**void SSLPeriodic**(
TCP_SOCKET **hTCP**,
BYTE **sslStubID**
);

**Возвращает**

Ни один

**описание**

Эта функция выполняет периодические задачи для модуля SSL . Это включает в себя обработку для операций RSA .

**Предпосылки**

SSL уже инициализирована .

**Параметры**

**Параметры Описание**

HTCP сокет , для которого необходимо проводить периодические функции

IDSSL заглушки , чтобы использовать

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

**10.17.3.4 TCPRequestSSLMessage Функция**

**файл**

TCP.h

C

BOOL **TCPRequestSSLMessage**(
TCP_SOCKET **hTCP**,
BYTE **msg**
);

**описание**

Эта функция вызывается для запроса , что конкретное сообщение SSL передаваться . Это сообщение должно быть вызван только Модуль SSL .

**Предпосылки**

TCP инициализации .

**Параметры**

**Параметры Описание**

Подключение HTCP TCP использовать MSG один из видов SSL_MESSAGE для передачи.

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** Сообщение было предложено .

**ЛОЖЬ** Другое сообщение уже находится в ожидании передачи .

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

**10.17.3.5 TCPSSLGetPendingTxSize Функция**

**файл**

TCP.h

C

```
WORD TCPSSLGetPendingTxSize(
TCP_SOCKET hTCP
);
```

**Возвращает**

Ни один

**описание**

Эта функция определяет , сколько байтов в ожидании для будущей записи SSL .

**Предпосылки**

TCP инициализации и HTCP подключен при активном подключении к SSL .

**Параметры**

**Параметры Описание**

Подключение HTCP TCP проверить

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

**10.17.3.6 TCPSSLHandleIncoming Функция**

**файл**

TCP.h

C

```
void TCPSSLHandleIncoming(
TCP_SOCKET hTCP
);
```

**Возвращает**

Ни один

**описание**

Эта функция обрабатывает поступающие данные TCP в качестве записи SSL и выполняет любое необходимое репозиционирование и расшифровки .

**Замечания**

Эта функция никогда не должны называться одним из приложений. Он используется только самим модулем SSL .

**Предпосылки**

TCP инициализации и HTCP связано с активного сеанса SSL.

**Параметры**

**Параметры Описание**

Подключение HTCP TCP для обработки входящих данных о

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

**10.17.3.7 TCPSSLHandshakeComplete Функция**

**файл**

TCP.h

C

```
void TCPSSLHandshakeComplete(
TCP_SOCKET hTCP
);
```

**Возвращает**

Ни один

**описание**

Эта функция сбрасывает флаг , указывающий, чтоSSL рукопожатия завершена.

**Замечания**

Эта функция никогда не должны называться одним из приложений. Он используется только самим модулем SSL .

**Предпосылки**

TCP инициализации и HTCP подключен .

**Параметры**

**Параметры Описание**

Подключение HTCP TCP установить

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

**10.17.3.8 TCPSSLInPlaceMACEncrypt Функция**

**файл**

TCP.h

C

```
void TCPSSLInPlaceMACEncrypt(
TCP_SOCKET hTCP,
ARCFOUR_CTX* ctx,
BYTE* MACSecret,
WORD len
);
```

**Возвращает**

Ни один

**описание**

Эта функция шифрует данные в буфере TCP a calcuating MAC . Когда шифрование закончена, MAC добавляется к буфер и запись будет готов к передаче .

**Замечания**

Эта функция никогда не должны называться одним из приложений. Он используется только самим модулем SSL .

**Предпосылки**

TCP инициализации HTCP подключен , и SBOX CTX автора загружается .

**Параметры**

**Параметры Описание**

Подключение HTCP TCP для шифрования в CTX ARCFOUR контекст шифрования использовать MACSecret MAC шифрования секрет в использовании

**len** - Количество байт для Crypt

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

**10.17.3.9 TCPSSLPutRecordHeader Функция**

**файл**

TCP.h

C

```
void TCPSSLPutRecordHeader(
TCP_SOCKET hTCP,
BYTE* hdr,
BOOL recDone
);
```

**Возвращает**

Ни один

**описание**

Эта функция записывает заголовок SSL записи в ожидании данных TCP SSL , то указывает, что данные готовы для отправки по перемещение указателя txHead .

Если запись будет завершена, установлен recDone ИСТИНА. Указатель sslTxHead будут перемещены вперед 5 байт , чтобы оставить пространство для Будущее Заголовок записи . Если запись лишь частично направил , использовать ЛОЖЬ и оставить указатель там, где это так, что больше данных может добавить к записи. Частичные записи могут быть использованы только для сообщения SERVER_CERTIFICATE рукопожатия .

**Замечания**

Эта функция никогда не должны называться одним из приложений. Он используется только самим модулем SSL .

**Предпосылки**

TCP инициализации и HTCP связано с активного сеанса SSL.

**Параметры**

**Параметры Описание**

Подключение HTCP TCP написать заголовок и передавать с

Заголовок HDR Запись (5 байт ), чтобы отправить или NULL , чтобы просто переместить pointerctx

**recDone** ИСТИНА , если запись делается , иначе ложь

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

**10.17.3.10 TCPStartSSLServer Функция**

**файл**

TCP.h

C

```
BOOL TCPStartSSLServer(
TCP_SOCKET hTCP
);
```

**описание**

Эта функция устанавливает сеанс сервера SSL при установлении нового соединения устанавливается на SSL порт.

**Предпосылки**

TCP инициализации и HTCP уже подключен.

**Параметры**

**Параметры Описание**

Подключение HTCP TCP , чтобы обеспечить

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** соединениеSSL был инициирован

**ЛОЖЬ** Недостаточные ресурсы SSL ( заглушки ) были доступны

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

**10.17.3.11 SSL_MIN_SESSION_LIFETIME Макро**

**файл**

SSL.h

C

```
#define SSL_MIN_SESSION_LIFETIME (1*TICK_SECOND)
```

**описание**

Минимальный срок службы для SSL Сессии Сессии не может быть перераспределены до этого много времени не истечет

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

**10.17.3.12 SSL_RSA_LIFETIME_EXTENSION Макро**

**файл**

SSL.h

C

**#define SSL_RSA_LIFETIME_EXTENSION** (8*TICK_SECOND)

**описание**

Продление срока службы для RSA операции Sessions жизни продлевается на эту сумму , когдарасчет RSA сделан

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

**10.17.4 SSL Внутренняя Пользователи**

**Перечисления**

**Имя Описание**

**SM_SSL_RX_SERVER_HELLO**(См. стр. 491 )Государственная машина для SSLRxServerHello (см. стр. 514 )

**SSL_ALERT_LEVEL** (см.страница 492 )Описывает два типа записей оповещения

**SSL_MESSAGES** (см. стр.497)Описывает типы сообщений SSL ( рукопожатие и оповещения )

**SSL_SESSION_TYPE** (см.страница 500 )Тип SSL сессии Перечисление

**Функции**

**Имя Описание**

**CalculateFinishedHash** (см. стр. 483 ) Рассчитывает рукопожатие хэш данных. Хашид может быть либо MD5 или SHA-1, и эта функция будет вычислять соответственно .

**GenerateHashRounds** (см. стр. 483 ) Создает хэш- раундов, чтобы найти или главный секрет или клавишный блок .

**GenerateSessionKeys** (см. стр. 484 ) Создает ключи записи сеанса и MAC секреты

**HSEnd** (см. стр. 484 ) Хэши (см. стр. 202) содержание сообщения в хэш Рукопожатие

структуры и начинает новую рукопожатие хэш .

**HSGet** (см. стр. 485 ) Читает данные из сокета , прозрачно хэширования в рукопожатие хэшей.

**HSGetArray** (см. стр. 485 )

• Функция: статическая СЛОВО HSGetArray ( TCP_SOCKET (см. стр. 553 ) SKT ,

BYTE * данные , СЛОВО длина)

*

• Никогда Условие: Ни

*

• Вход: SKT - разъем для считывания данных с

• данные - массив для чтения в , или NULL

• len - количество байт для чтения

*

• Выход: Количество прочитанных байтов

*

Не • Побочные эффекты : Нет

*

• Обзор : чтение данных из сокета , прозрачно хэширования

• в рукопожатие хэшей.

*

• Никогда Примечание: Ни один

**HSGetWord** (см. стр. 486 ) Читает данные из сокета , прозрачно хэширования в рукопожатие хэшей.

**HSPut** (см. стр. 486 ) Записывает данные в сокет , прозрачно хэширования в рукопожатие хэшей.

**HSPutArray** (см. стр. 487 )

• Функция: статическая СЛОВО HSPutArray ( TCP_SOCKET (см. стр. 553 ) SKT ,

BYTE * данные , BYTE длина)

*

• Никогда Условие: Ни

*

• Вход: SKT - разъем для записи данных в

• данные - данные , чтобы написать

• Лен - количество байтов для записи

*

• Выход: число записанных байтов

*

Не • Побочные эффекты : Нет

*

• Обзор : Записывает данные в сокет , прозрачно хэширования

• в рукопожатие хэшей.

*

• Никогда Примечание: Ни один

**HSPutROMArray** (см.страница 488 ) Это функция HSPutROMArray .

**HSPutWord** (см. стр. 488 ) Записывает данные в сокет , прозрачно хэширования в рукопожатие хэшей.

**HSStart** (см. стр. 488 ) Устанавливает буфер для хранения данных для отслеживания хэш рукопожатия

**LoadOffChip** (см. стр.490) Копирует данные из оперативной памяти Ethernet к локальной памяти

**SaveOffChip** (см. стр.491) Копирует данные в ПОС памяти в ОЗУ Ethernet

**SSLBufferAlloc** (см. стр.502) Выделяет буфер для использования .

**SSLBufferFree** (см. стр.503) Указано буфер освобождается

**SSLBufferSync** (см. стр.504) Указано буфер загружается в оперативную память . Только загружает в случае необходимости, и сохраняет любой текущий буфер перед переключением .

**SSLHashAlloc** (см. стр.505) Выделяет хэш для использования.

**SSLHashFree** (см. стр.505) Указано хэш освобождается

**SSLHashSync** (см. стр.506) Указано хэш загружается в оперативную память . Только загружает в случае необходимости, и сохраняет любой ток хэш перед переключением .

**SSLKeysSync** (см. стр.507) Указано ключевое множество загружается в оперативную память . Только загружает в случае необходимости, и сохраняет любой текущий ключ установлен перед переключением .

**SSLMACAdd** (см. стр.508) Это функция SSLMACAdd .

**SSLMACBegin** (см. стр.508) Это функция SSLMACBegin .

**SSLMACCalc** (см. стр.508) Это функция SSLMACCalc .

**SSLRSAOperation** (см.страница 508) Приостановка обработки соединения до расчет RSA не будет завершена.

**SSLRxAlert** (см. стр. 509 ) Принимает предупреждающее сообщение и решает, что делать

**SSLRxAntiqueClientHello** (см. стр. 510) Получает сообщение SSLv2 ClientHello , инициируя новый сеанс SSL с клиент

**SSLRxCCS** (см. стр. 510 ) Принимает ChangeCipherSpec с удаленного сервера

**SSLRxClientHello** (см.страница 511 ) Получает сообщение ClientHello , инициируя новый сеанс SSL с клиентом

**SSLRxClientKeyExchange**(См. Стр. 511 ) Получает сообщение ClientKeyExchange и начинает процесс расшифровки.

**SSLRxFinished** (см. стр.512) Получает Готовые сообщение от удаленного узла

**SSLRxHandshake** (см.страница 512) Получает сообщение рукопожатие .

**SSLRxRecord** (см. стр.513)Получает запись SSL .

**SSLRxServerCertificate** (см. стр. 513 ) Принимает ServerCertificate с удаленного сервера , находит открытый ключ информация , и выполняет операцию RSA.

**SSLRxServerHello** (см.страница 514) Получает ServerHello с удаленного сервера

**SSLSessionMatchID** (см.страница 515 ) Находит в кэше SSL сессию для повторного использования. Синхронизация найдено сеанс в ОЗУ .

**SSLSessionMatchIP** (см.страница 515 ) Находит в кэше SSL сессию для повторного использования

**SSLSessionNew** (см.страница 516) Находит место для нового сессии SSL

**SSLSessionSync** (см.страница 517 ) Указано сессия загружается в оперативную память . Только загружает в случае необходимости, и сохраняет любой Нынешняя сессия перед переключением , если он был обновлен .

**SSLStartPartialRecord** (см. стр. 518) Начинается длинный послужной SSL .

**SSLStubAlloc** (см. стр.519) Выделяет заглушку для использования.

**SSLStubFree** (см. стр.519) Указано заглушка выпущен

**SSLStubSync** (см. стр.520) Указано заглушки загружается в оперативную память . Только загружает в случае необходимости, и сохраняет любой ток заглушки перед переключением .

**SSLTerminate** (см. стр.520) Завершает соединение SSL и освобождает выделенные ресурсы.

**SSLTxCCSFin** (см. стр.521 ) Генерирует сеансовых ключей от главного секрета , то выделяет и генерирует контекст шифрования. После того, как обработка завершена , передает Изменение шифра Spec сообщение и готовой сообщение рукопожатие на сервер .

**SSLTxClientHello** (см.страница 521 ) Передает сообщение ClientHello инициировать новый сеанс SSL с сервер .

**SSLTxClientKeyExchange** (см. стр. 522 ) Передает зашифрованное предварительно главный секрет к серверу и просит Изменение Cipher Spec. Также генерирует главный секрет из предварительного мастера секрет, что был использован .

**SSLTxMessage** (см. стр.522) Передает сообщение об SSL .

**SSLTxRecord** (см. стр.523 ) Передает запись SSL .

**SSLTxServerCertificate** (см. стр. 523 ) Передает сообщение сертификат с указанным открытым ключом сервераСертификат .

**SSLTxServerHello** (см.страница 524 ) Передает сообщение ServerHello .

**SSLTxServerHelloDone** (см. стр. 525) Передает сообщение ServerHelloDone .

**макрос**

**Имя Описание**

**RESERVED_SSL_MEMORY** (см. стр. 490 ) Общая площадь необходим всем требованиям хранения SSL

**SSL_ALERT** ( см. стр. 492 ) код протокола для записей оповещения

**SSL_APPLICATION** (см. стр. 492 ) Протокол код для записи данных приложений

**SSL_BASE_BUFFER_ADDR** ( см. стр. 493 ) базовый адрес для SSL буферов

**SSL_BASE_HASH_ADDR** ( см. стр. 493 ) базовый адрес для SSL хэшей

**SSL_BASE_KEYS_ADDR** ( см. стр. 493 ) базовый адрес для ключей SSL

**SSL_BASE_SESSION_ADDR** ( см. стр. 493 ) базовый адрес для SSL сессий

**SSL_BASE_STUB_ADDR** (см. стр. 493 ) Базовый адрес для SSL окурков

**SSL_BUFFER_SIZE** (см. стр. 494 ) количество места, необходимого на один буфер SSL

**SSL_BUFFER_SPACE** (см. стр. 494 ) количество места, необходимого всем буфера SSL

**SSL_CHANGE_CIPHER_SPEC** ( см. стр. 495) код протокола для Изменить Cipher Spec записей

**SSL_HANDSHAKE** ( см. стр. 495) код протокола для рукопожатия записей

**SSL_HASH_SIZE** (см. стр. 495) количество места, необходимого одним SSL хэш

**SSL_HASH_SPACE** (см. стр. 496) количество места, необходимого всем SSL хэш

**SSL_KEYS_SIZE** (см. стр. 497 ) количество места, необходимого на одного ключа SSL

**SSL_KEYS_SPACE** (см. стр. 497 ) количество места, необходимого на всех ключевых SSL

**SSL_RSA_EXPORT_WITH_ARCFOUR_40_MD5** (См. Стр. 498 ) Это макро

**SSL_RSA_EXPORT_WITH_ARCFOUR_40_MD5 .**

**SSL_RSA_WITH_ARCFOUR_128_MD5** (см.страница 498)Это макро **SSL_RSA_WITH_ARCFOUR_128_MD5 .**

**SSL_SESSION_SIZE** (см. стр. 499) количество места, необходимого на один сеанс SSL

**SSL_SESSION_SPACE** (см. стр. 499) количество места, необходимого всем SSL сессии

**SSL_STUB_SIZE** (см. стр. 501 ) количество места, необходимого одним SSL заглушки

**SSL_STUB_SPACE** (см. стр. 501 ) количество места, необходимого всеми окурков SSL

**SSL_VERSION** (см. стр. 502) SSL номер версии

**SSL_VERSION_HI** ( см. стр. 502) SSL номер версии (старший байт )

**SSL_VERSION_LO** ( см. стр. 502) SSL номер версии (младший байт )

**SSLFinishPartialRecord** (см. стр. 504 ) Этот макрос SSLFinishPartialRecord .

**SSLFlushPartialRecord** (см. стр. 504 ) Этот макрос SSLFlushPartialRecord .

**SSLSessionUpdated** (см. стр. 517 ) Этот макрос SSLSessionUpdated .

**модуль**

**SSL** (см. стр. 462 )

**структуры**

**Имя Описание**

**SSL_KEYS** ( см. стр. 496) определение памяти для ключей SSL . Эта область делится на локальных и удаленных районах.

Во время рукопожатия, Local.random и Remote.random удерживайте ServerRandom и ClientRandom значения . После того, как ключи сессии рассчитывается ,Local.app и Remote.app содержать MAC тайну, запись Контекст порядковый номер , и шифрование для модуля Arcfour .

**SSL_SESSION** (см. стр.498)Складское помещение для идентификаторов SSL сессии. (Код_сеанса и MasterSecret )

**SSL_SESSION_STUB** (см. стр. 499)Значение заготовки для SSL_SESSION (см. стр. 498 ) . Тэг связывает это сессия с удаленного узла , либо путем сопоставления с удаленным IP - адреса, когда мы клиент или первые 3 байта идентификатора сессии , когда мы находимся хозяин .

Когда сеанс бесплатно / истек , тег 0x00000000 . Значение lastUsed является Тик рассчитывать , когда сессия была последний раз был использован так , что пожилые сессии могут быть перезаписаны в первую очередь.

**SSL_STUB** ( см. стр. 500 ) держатель памяти для общей информации , связанной cSSL соединений.

**Союзы**

**Имя Описание**

**SSL_BUFFER** (см. стр.494 )Общий объем буфера для SSL. Элемент hashRounds используется, когда это буфер необходим для квитирования вычислений хеш , а полный элемент используется как SBOX для расчетов Arcfour .

**Переменные**

**Имя Описание**

**isBufferUsed** (см. стр.489 )Указывает , которые буферы используются

**isHashUsed** (см. стр. 489 ) Указывает, какой хэши используются

**isStubUsed** (см. стр. 489 ) Указывает, какой заглушки используются

маски (см. стр. 490 ) Маски для каждого бита в * используется переменных

**ptrHS** (см. стр. 490 ), используемый в буферных результатов рукопожатие

**SSL_CERT** (см. стр. 495 ) данных сертификата открытого RSA ?

**SSL_CERT_LEN** (см.страница 495)RSA длиной открытый сертификат ?

**sslBufferID** (см. стр. 503 ) Какие буфер загружен

**sslHash** (см. стр. 505 ) Хранение Hash

**sslHashID** (см. стр. 506 ) Какие хэш загружен

**sslKeys** (см. стр. 507 ) текущей сессии SSL,

**sslKeysID** (см. стр. 507 ) SSL_KEYS (см. стр. 496 ) , которые загружаются

**sslRSAStubID** (см. стр.509)Какие заглушка с использованием RSA , если таковые имеются

**sslSession** (см. стр. 514 ) Текущие данные сессия

**sslSessionID** (см. стр.515) Какие сессия загружается

**sslSessionStubs** (см. стр.516)8 байт сессия заглушки

**sslSessionUpdated** (см.страница 517 ) Или нет, это был обновлен

**sslStub** (см. стр. 518 ) текущего SSL, заглушки

**sslStubID** (см. стр. 520) SSL_STUB (см. стр. 500 ) , который загружается

**описание**

Следующие функции и переменные определяются как внутренние, в модуль SSL .

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

### 10.17.4.1 CalculateFinishedHash Функция

**файл**

SSL.c

C

```
static void CalculateFinishedHash(
BYTE hashID,
BOOL fromClient,
BYTE * result
);
```

**Побочные** эффекты

Ни один

**Возвращает**

Ни один

**описание**

Рассчитывает рукопожатие хэш данных. Хашид может быть либо MD5 или SHA-1, и эта функция будет вычислять соответственно .

**Замечания**

Ни один

**Предпосылки**

Хашид имеет все данные рукопожатие хэшированные сих пор, и нынешняя сессия синхронизирован дюйма

**Параметры**

**Параметры Описание**

Хашид хеш суммы использовать

fromClient ИСТИНА, если клиент отправителя привести , где хранить результаты

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

**10.17.4.2 GenerateHashRounds Функция**

**файл**

SSL.c

C

```
static void GenerateHashRounds(
BYTE num,
BYTE* rand1,
BYTE* rand2
);
```

**Побочные эффекты**

Уничтожает SSL буферное пространство

**Возвращает**

Ни один

**описание**

Создает хэш- раундов, чтобы найти или главный секрет или клавишный блок .

**Замечания**

Эта функция будет переполнить буфер после 7 тура , но на практике пит = 3 или пит = 4 .

**Предпосылки**

Буфер SSL выделяется для временного использования , а также данные для запуска раундов на находится в sslSession.masterSecret

**Параметры**

**Параметры Описание**

пит сколько раундов вычислить

rand1 первое случайное блок данных , чтобы использовать

rand2 второй случайный блок данных , чтобы использовать

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

**10.17.4.3 GenerateSessionKeys Функция**

**файл**

SSL.c

C

```
static void GenerateSessionKeys();
```

**Побочные эффекты**

Уничтожает буферное пространство SSL

**Возвращает**

Ни один

**описание**

Создает ключи записи сеанса и MAC секреты

**Замечания**

Ни один

**Предпосылки**

Буфер SSL выделяется для временного использования , ключи сеанса , синхронизируются , и буферы приема и передачи выделяются на S- коробки.

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

**10.17.4.4 HSEnd Функция**

**файл**

SSL.c

C

**static void HSEnd**();

**Побочные** эффекты

Ни один

**Возвращает**

Ни один

**описание**

Хэши ( см. стр. 202) содержание сообщения в хэш- структур Рукопожатие и начинает новую рукопожатие хэш.

**Замечания**

Ни один

**Предпосылки**

Ни один

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

**10.17.4.5 HSGet Функция**

**файл**

SSL.c

C

**static** WORD **HSGet**(

TCP_SOCKET **skt**,
BYTE ***** **b**
);

**Побочные эффекты**

Ни один

**Возвращает**

Количество прочитанных байтов

**описание**

Читает данные из сокета , прозрачно хэширования в рукопожатие хэшей.

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

SKT разъем для считывания данных с

б байт для чтения в

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

**10.17.4.6 HSGetArray Функция**

**файл**

SSL.c

C

```
static WORD HSGetArray(
TCP_SOCKET skt,
BYTE * data,
WORD len
);
```

**описание**

• Функция: статическая СЛОВО HSGetArray ( TCP_SOCKET (см. стр. 553 ) SKT , BYTE * данные , СЛОВО длина)

*

• Никогда Условие: Ни

*

• Вход: SKT - разъем для считывания данных с

• данные - массив для чтения в , или NULL

• Лен - количество байт для чтения

*

• Выход: Количество прочитанных байтов

*

Не • Побочные эффекты : Нет

*

• Обзор : чтение данных из сокета , прозрачно хэширования

• в рукопожатие хэшей.

*

• Никогда Примечание: Ни один

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

**10.17.4.7 HSGetWord Функция**

**файл**

SSL.c

C

```
static WORD HSGetWord(
TCP_SOCKET skt,
WORD * w
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Количество прочитанных байтов

**описание**

Читает данные из сокета , прозрачно хэширования в рукопожатие хэшей.

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

SKT разъем для считывания данных с

ж слово читать в

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

### 10.17.4.8 HSPut Функция

**файл**

SSL.c

C

```
static WORD HSPut(
TCP_SOCKET skt,
BYTE b
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Количество записанных байт

**описание**

Записывает данные в сокет , прозрачно хэширования в рукопожатие хэшей.

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

SKT разъем для записи данных в

б байт писать

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

**10.17.4.9 HSPutArray Функция**

**файл**

SSL.c

C

```
static WORD HSPutArray(
TCP_SOCKET skt,
BYTE * data,
WORD len
);
```

**описание**

• Функция: статическая СЛОВО HSPutArray ( TCP_SOCKET (см. стр. 553 ) SKT , BYTE * данные , BYTE длина)

*

• Никогда Условие: Ни

*

• Вход: SKT - разъем для записи данных в

• данные - данные , чтобы написать

• Лен - количество байтов для записи

*

• Выход: число записанных байтов

*

Не • Побочные эффекты : Нет

*

• Обзор : Записывает данные в сокет , прозрачно хэширования

• в рукопожатие хэшей.

*

• Никогда Примечание: Ни один

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

**10.17.4.10 HSPutROMArray Функция**

**файл**

SSL.c

C

```
static WORD HSPutROMArray(
TCP_SOCKET skt,
ROM BYTE * data,
WORD len
);
```

**описание**

Это функция HSPutROMArray .

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

**10.17.4.11 HSPutWord Функция**

**файл**

SSL.c

C

```
static WORD HSPutWord(
TCP_SOCKET skt,
WORD w
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Количество записанных байт

**описание**

Записывает данные в сокет , прозрачно хэширования в рукопожатие хэшей.

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

SKT разъем для записи данных в

ж слово написать

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

**10.17.4.12 HSStart Функция**

**файл**

SSL.c

C

```
static void HSStart();
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Устанавливает буфер для хранения данных для отслеживания хэш рукопожатия

**Замечания**

Ни один

**Предпосылки**

Ни один

**раздел**

Рукопожатие Хэш и ввода / вывода Функции

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

**10.17.4.13 isBufferUsed Переменная**

**файл**

SSL.c

C

WORD **isBufferUsed**;

**описание**

Указывает , которые буферы используются

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

**10.17.4.14 isHashUsed Переменная**

**файл**

SSL.c

C

WORD **isHashUsed**;

**описание**

Указывает , которые хэши используются

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

**10.17.4.15 isStubUsed Переменная**

**файл**

SSL.c

C

WORD **isStubUsed**;

**описание**

Указывает, какие заглушки используются

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

**10.17.4.16 маски переменной**

**файл**

SSL.c

C

ROM WORD **masks**[16] = { 0x0001, 0x0002, 0x0004, 0x0008, 0x0010, 0x0020, 0x0040, 0x0080, 0x0100, 0x0200, 0x0400, 0x0800, 0x1000, 0x2000, 0x4000, 0x8000 };

**описание**

Маски для каждого бита в * используется переменных

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

**10.17.4.17 ptrHS Переменная**

**файл**

SSL.c

C

BYTE * **ptrHS**;

**описание**

Используется в буферных результатов рукопожатие

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

### 10.17.4.18 RESERVED_SSL_MEMORY Макро

**файл**

SSL.h

C

```
#define RESERVED_SSL_MEMORY ((DWORD)(SSL_STUB_SPACE + SSL_KEYS_SPACE
+ SSL_HASH_SPACE +
SSL_BUFFER_SPACE + SSL_SESSION_SPACE))
```

**описание**

Общая площадь необходим всем требованиям хранения SSL

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

### 10.17.4.19 LoadOffChip Функция

**файл**

SSL.c

C

```
static void LoadOffChip(
BYTE * ramAddr,
PTR_BASE ethAddr,
WORD len
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Копирует данные из оперативной памяти Ethernet к локальной памяти

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

ramAddr адрес назначения в оперативной памяти

ethAddr адрес источника в Ethernet памяти

лен количество байт для копирования

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

**10.17.4.20 SaveOffChip Функция**

**файл**

SSL.c

C

```
static void SaveOffChip(
BYTE * ramAddr,
PTR_BASE ethAddr,
WORD len
```

);

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Копирует данные в ПОС памяти в ОЗУ Ethernet

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

адрес источника ramAddr в оперативной памяти

ethAddr адрес назначения в Ethernet памяти

лен количество байт для копирования

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

**10.17.4.21 SM_SSL_RX_SERVER_HELLO Перечисление**

**файл**

SSL.h

C

```
typedef enum {
RX_SERVER_CERT_START = 0u,
RX_SERVER_CERT_FIND_KEY,
RX_SERVER_CERT_FIND_N,
RX_SERVER_CERT_READ_N,
RX_SERVER_CERT_READ_E,
RX_SERVER_CERT_CLEAR
} SM_SSL_RX_SERVER_HELLO;
```

**описание**

Государственная машина для SSLRxServerHello (см. стр. 514 )

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

### 10.17.4.22 SSL_ALERT Макро

**файл**

SSL.h

C

```
#define SSL_ALERT 21u/ / код протокола для записей оповещения
```

**описание**

Протокол код записей оповещения

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

### 10.17.4.23 SSL_ALERT_LEVEL Перечисление

**файл**

SSL.h

C

```
typedef enum {
SSL_ALERT_WARNING = 1u,
SSL_ALERT_FATAL = 2u
} SSL_ALERT_LEVEL;
```

**Пользователи**

**Пользователи Описание**

SSL_ALERT_WARNING = 1u Предупреждение является предупреждением ( сессия может быть возобновлена )

SSL_ALERT_FATAL = 2u Предупреждение является фатальным ( сессия не является возобновляем )

**описание**

Описывает два типа записей оповещения

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

**10.17.4.24 SSL_APPLICATION Макро**

**файл**

SSL.h

C

```
#define SSL_APPLICATION 23u  / / протокол для записи данных приложений
```

**описание**

Протокол код записей данных приложений

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

### 10.17.4.25 SSL_BASE_BUFFER_ADDR Макро

**файл**

SSL.c

C

```
#define SSL_BASE_BUFFER_ADDR (BASE_SSLB_ADDR + SSL_STUB_SPACE +
SSL_KEYS_SPACE +
SSL_HASH_SPACE)
```

**описание**

Базовый адрес для SSL буферов

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

### 10.17.4.26 SSL_BASE_HASH_ADDR Макро

**файл**

SSL.c

C

```
#define SSL_BASE_HASH_ADDR (BASE_SSLB_ADDR + SSL_STUB_SPACE +
SSL_KEYS_SPACE)
```

**описание**

Базовый адрес для SSL хэшей

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

**10.17.4.27 SSL_BASE_KEYS_ADDR Макро**

**файл**

SSL.c

C

```
#define SSL_BASE_KEYS_ADDR (BASE_SSLB_ADDR + SSL_STUB_SPACE)
```

**описание**

Базовый адрес для ключей SSL

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

**10.17.4.28 SSL_BASE_SESSION_ADDR Макро**

**файл**

SSL.c

C

```
#define SSL_BASE_SESSION_ADDR (BASE_SSLB_ADDR + SSL_STUB_SPACE +
SSL_KEYS_SPACE +
SSL_HASH_SPACE + SSL_BUFFER_SPACE)
```

**описание**

Базовый адрес для SSL сессий

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

**10.17.4.29 SSL_BASE_STUB_ADDR Макро**

**файл**

SSL.c

C

```
#define SSL_BASE_STUB_ADDR (BASE_SSLB_ADDR)
```

**описание**

Базовый адрес для SSL окурков

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

**10.17.4.30 SSL_BUFFER союз**

**файл**

SSL.h

C

```
typedef union {
struct {
HASH_SUM hash;
BYTE md5_hash[16];
BYTE sha_hash[20];
BYTE temp[256-sizeof(HASH_SUM)-16-20];
} hashRounds;
BYTE full[256];
} SSL_BUFFER;
```

**описание**

Общий объем буфера для SSL. Элемент hashRounds используется, когда этот буфер необходим для расчетов хэш рукопожатие, и полный элемент используется в качестве SBOX для расчетов Arcfour .

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

**10.17.4.31 SSL_BUFFER_SIZE Макро**

**файл**

SSL.h

C

**#define SSL_BUFFER_SIZE** ((DWORD)**sizeof**(SSL_BUFFER)) / / Количество пространстве необходимы одном буфере SSL

**описание**

Объем пространства необходимы один буфер SSL

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

**10.17.4.32 SSL_BUFFER_SPACE Макро**

**файл**

SSL.h

C

**#define SSL_BUFFER_SPACE** (SSL_BUFFER_SIZE*MAX_SSL_BUFFERS) / / количество места, необходимого всеми буфера SSL

**описание**

Количество места, необходимого всем буфера SSL

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

**10.17.4.33 SSL_CERT Переменная**

**файл**

SSL.c

C

ROM BYTE **SSL_CERT**[];

**описание**

Данные RSA общественного сертификат?

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

**10.17.4.34 SSL_CERT_LEN Переменная**

**файл**

SSL.c

C

ROM WORD **SSL_CERT_LEN**;

**описание**

RSA длиной открытый сертификат ?

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

**10.17.4.35 SSL_CHANGE_CIPHER_SPEC Макро**

**файл**

SSL.h

C

**#define SSL_CHANGE_CIPHER_SPEC** 20u / / протокол для Изменить Cipher Spec записей

**описание**

Протокол код Изменить Cipher Spec записей

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

**10.17.4.36 SSL_HANDSHAKE Макро**

**файл**

SSL.h

C

**#define SSL_HANDSHAKE** 22u / / протокол для рукопожатия записей

**описание**

Протокол код Рукопожатие записей

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

**10.17.4.37 SSL_HASH_SIZE Макро**

**файл**

SSL.h

C

**#define SSL_HASH_SIZE** ((DWORD)**sizeof**(HASH_SUM))/ / Количество

пространстве нужны один SSL хэш

**описание**

Количество места, необходимого одним SSL хэш

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

**10.17.4.38 SSL_HASH_SPACE Макро**

**файл**

SSL.h

C

**#define SSL_HASH_SPACE** ((DWORD)(SSL_HASH_SIZE*MAX_SSL_HASHES)) / / количество места, необходимого всеми SSL хэш

**описание**

Количество места, необходимого всем SSL хэш

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

**10.17.4.39 SSL_KEYS Структура**

**файл**

SSL.h

C

```
typedef struct {
union {
struct {
BYTE MACSecret[16];
DWORD sequence;
ARCFOUR_CTX cryptCtx;
BYTE reserved[6];
} app;
BYTE random[32];
} Local;
union {
struct {
BYTE MACSecret[16];
DWORD sequence;
ARCFOUR_CTX cryptCtx;
BYTE reserved[6];
} app;
BYTE random[32];
} Remote;
} SSL_KEYS;
```

**Пользователи**

**Пользователи Описание**

BYTE MACSecret [16] ; MAC секрет записи сервера

DWORD sequence ; Записи порядковый номер сервера

ARCFOUR_CTX cryptCtx ; Контекст шифрования записи сервера

BYTE reserved [ 6] ; Будущее расширение

BYTE random [32] ; Server.random значение

BYTE MACSecret [16] ; MAC секрет записи клиента

DWORD sequence ; Клиентам записи порядковый номер

ARCFOUR_CTX cryptCtx ; Контекст шифрования записи клиента

BYTE reserved [ 6] ; Будущее расширение

BYTE random [32] ; Client.random значение

**описание**

Определение памяти для ключей SSL . Эта область делится на локальных и удаленных районах. Во время рукопожатия, Local.random и Remote.random держать ServerRandom и ClientRandom значения . После того, как ключи сессии рассчитывается , то Local.app

и Remote.app содержат секрет MAC , запись порядковый номер, и контекст шифрования для модуля Arcfour .

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

**10.17.4.40 SSL_KEYS_SIZE Макро**

**файл**

SSL.h

C

**#define SSL_KEYS_SIZE** ((DWORD)**sizeof**(SSL_KEYS)) / / Количество пространстве нужны один ключ SSL

**описание**

Объем пространства необходимы одного ключа SSL

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

**10.17.4.41 SSL_KEYS_SPACE Макро**

**файл**

SSL.h

C

**#define SSL_KEYS_SPACE** (SSL_KEYS_SIZE*MAX_SSL_CONNECTIONS) / / количество места, необходимого на всех ключевых SSL

**описание**

Объем пространства необходимы всего ключа SSL

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

**10.17.4.42 SSL_MESSAGES Перечисление**

**файл**

SSL.h

C

```
typedef enum {
SSL_HELLO_REQUEST = 0u,
SSL_CLIENT_HELLO = 1u,
SSL_ANTIQUE_CLIENT_HELLO = 18u,
SSL_SERVER_HELLO = 2u,
SSL_CERTIFICATE = 11u,
SSL_SERVER_HELLO_DONE = 14u,
SSL_CLIENT_KEY_EXCHANGE = 16u,
SSL_FINISHED = 20u,
SSL_ALERT_CLOSE_NOTIFY = 0u+0x80,
SSL_ALERT_UNEXPECTED_MESSAGE = 10u+0x80,
SSL_ALERT_BAD_RECORD_MAC = 20u+0x80,
SSL_ALERT_HANDSHAKE_FAILURE = 40u+0x80,
SSL_NO_MESSAGE = 0xff
} SSL_MESSAGES;
```

**Пользователи**

**Пользователи Описание**

**SSL_HELLO_REQUEST** = 0u HelloRequest рукопожатие сообщение (в настоящее время не поддерживается)

**SSL_CLIENT_HELLO** = 1u ClientHello рукопожатие сообщение

**SSL_ANTIQUE_CLIENT_HELLO** = 18u SSLv2 ClientHello сообщение рукопожатие ( Поддерживаемые для обратной совместимости . Это внутренне определенное значение . )

**SSL_SERVER_HELLO** = 2u ServerHello рукопожатие сообщение

**SSL_CERTIFICATE** = 11u ServerCertifiate сообщение рукопожатие

**SSL_SERVER_HELLO_DONE** = 14U ServerHelloDone сообщение рукопожатие

**SSL_CLIENT_KEY_EXCHANGE** = 16u ClientKeyExchange сообщение рукопожатие

**SSL_FINISHED** = 20u Готовые сообщение рукопожатие

**SSL_ALERT_CLOSE_NOTIFY** = 0u +0 x80 CloseNotify сообщение оповещения ( фиктивное значение для внутреннего использования )

**SSL_ALERT_UNEXPECTED_MESSAGE**= 10й +0 x80 UnexpectedMessage предупреждающее сообщение ( фиктивное значение для внутреннего использования )

**SSL_ALERT_BAD_RECORD_MAC** = 20u +0 x80 Сообщение оповещения BadRecordMAC ( фиктивное значение для внутреннего использования )

**SSL_ALERT_HANDSHAKE_FAILURE** =40u +0 x80 HandshakeFailure предупреждающее сообщение ( фиктивное значение для внутреннего использования )

SSL_NO_MESSAGE = 0xff никакое сообщение в настоящее время просил ( для внутреннего использования значение )

**описание**

Описывает типы сообщений SSL ( рукопожатие и оповещения )

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

**10.17.4.43 SSL_RSA_EXPORT_WITH_ARCFOUR_40_MD5 Макро**

**файл**

SSL.c

C

```
#define SSL_RSA_EXPORT_WITH_ARCFOUR_40_MD5 0x0003u
```

**описание**

Это макро SSL_RSA_EXPORT_WITH_ARCFOUR_40_MD5 .

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

**10.17.4.44 SSL_RSA_WITH_ARCFOUR_128_MD5 Макро**

**файл**

SSL.c

C

```
#define SSL_RSA_WITH_ARCFOUR_128_MD5 0x0004u
```

**описание**

Это макро SSL_RSA_WITH_ARCFOUR_128_MD5 .

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

**10.17.4.45 SSL_SESSION Структура**

**файл**

SSL.h

C

```
typedef struct {
BYTE sessionID[32];
BYTE masterSecret[48];
} SSL_SESSION;
```

**Пользователи**

**Пользователи Описание**

BYTE sessionID [32] ; SSL ID сессии для этой сессии

BYTE masterSecret [48] ; Связанное Master Secret для этой сессии

**описание**

Складское помещение для идентификаторов SSL сессии. (Код_сеанса и MasterSecret )

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

**10.17.4.46 SSL_SESSION_SIZE Макро**

**файл**

SSL.h

C

**#define SSL_SESSION_SIZE** ((DWORD)**sizeof**(SSL_SESSION))/ / количество места, необходимого по одной сессии SSL

**описание**

Объем пространства необходимы одного сеанса SSL

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

**10.17.4.47 SSL_SESSION_SPACE Макро**

**файл**

SSL.h

C

**#define SSL_SESSION_SPACE** (SSL_SESSION_SIZE*MAX_SSL_SESSIONS) / / Количество пространстве нужен всем SSL сессии

**описание**

Количество места, необходимого всем SSL сессии

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

**10.17.4.48 SSL_SESSION_STUB Структура**

**файл**

SSL.h

C

```
typedef struct {
DWORD_VAL tag;
DWORD lastUsed;
} SSL_SESSION_STUB;
```

**Пользователи**

**Пользователи Описание**

DWORD_VAL тег ; Выявление тег для подключения Когда мы клиент , это удаленный IP При

мыхозяин, это 0x00 следуют первые 3 байта идентификатор сеанса Когда это заглушка

свободен / истек , это 0x00

DWORD lastUsed ; Отметьте количество , когда сессия была последний раз был использован

**описание**

Значение заготовки для SSL_SESSION (см. стр. 498 ) . Тэг связывает это сеанс с удаленным узлом , либо путем сопоставления на удаленный IP адрес , когда мы находимся клиент или первые 3 байта идентификатора сессии , когда мы находимся хозяин . Когда сессия бесплатно / истек , тег 0x00000000 . LastUsed значениесчетчика тактов , когда сессия была

последний раз был использован так, чтобы старше сессии могут быть перезаписаны в первую очередь.

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

**10.17.4.49 SSL_SESSION_TYPE Перечисление**

**файл**

SSL.h

C

```
typedef enum {
SSL_CLIENT,
SSL_SERVER
} SSL_SESSION_TYPE;
```

**Пользователи**

**Пользователи Описание**

SSL_CLIENT Часовой устройство клиент SSL

SSL_SERVER Часовой устройство является хозяином SSL

**описание**

Тип SSL сессии Перечисление

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

**10.17.4.50 SSL_STUB Структура**

**файл**

SSL.h

C

```
typedef struct {
WORD wRxBytesRem;
WORD wRxHsBytesRem;
BYTE rxProtocol;
BYTE rxHSType;
BYTE idSession;
BYTE idMD5, idSHA1;
BYTE idRxHash;
BYTE idRxBuffer, idTxBuffer;
DWORD_VAL dwTemp;
struct {
unsigned char bIsServer : 1;
unsigned char bClientHello : 1;
unsigned char bServerHello : 1;
unsigned char bServerCertificate : 1;
unsigned char bServerHelloDone : 1;
unsigned char bClientKeyExchange : 1;
unsigned char bRemoteChangeCipherSpec : 1;
unsigned char bRemoteFinished : 1;
unsigned char bLocalChangeCipherSpec : 1;
unsigned char bLocalFinished : 1;
unsigned char bExpectingMAC : 1;
unsigned char bNewSession : 1;
unsigned char bCloseNotify : 1;
unsigned char bDone : 1;
unsigned char bRSAInProgress : 1;
unsigned char bKeysValid : 1;
} Flags;
BYTE requestedMessage;
void * supplementaryBuffer;
BYTE supplementaryDataType;
} SSL_STUB;
```

**Пользователи**

**Пользователи Описание**

WORD **wRxBytesRem** ; Байт влево, чтобы считать в текущей записи

WORD **wRxHsBytesRem** ; Байт влево, чтобы считать в текущем Рукопожатие submessage

BYTE **rxProtocol** ; Протокол сообщение читается

BYTE **rxHSType** ; Рукопожатие сообщение поступают

BYTE **idSession** ; ID попутного сессии

BYTE **idRxHash** ; ID для MAC хэш (Техас не нуждается в настойчивости )

DWORD_VAL **dwTemp** ; Используется для государственной машины в RxCertificate

unsigned char **bIsServer** : 1 ; Мы сервер

unsigned char **bClientHello** : 1 ; ClientHello была переданных / полученных

unsigned char **bServerHello** : 1 ; ServerHello была переданных / полученных

unsigned char **bServerCertificate** : 1 ; ServerCertificate была переданных / полученных

unsigned char **bServerHelloDone** : 1 ; ServerHelloDone была переданных / полученных

unsigned char **bClientKeyExchange** : 1 ; ClientKeyExchange была переданных / полученных

unsigned char **bRemoteChangeCipherSpec** : 1 ; Удаленный узел послал сообщение ChangeCipherSpec

unsigned char **bRemoteFinished** : 1 ; Удаленного узла направил Готовые сообщение

unsigned char **bLocalChangeCipherSpec**: 1 ;Мы направили сообщение ChangeCipherSpec

unsigned char **bLocalFinished** : 1 ; Мы направили Готовые сообщение

unsigned char **bExpectingMAC** : 1 ; Мы ожидаем, что MAC в конце сообщения

unsigned char **bNewSession** : 1 ; ИСТИНА, если новая сессия , ЛОЖЬ, если возобновление

unsigned char **bCloseNotify** : 1 ; Будь или не был отправленCloseNotify / полученных

unsigned char **bDone** : 1 ; ИСТИНА, если соединение закрывается

unsigned char **bRSAInProgress** : 1 ; ИСТИНА, когда RSA оп находится в процессе

unsigned char **bKeysValid** : 1 ; ПРАВДА, если ключи сеанса были получены

BYTE **requestedMessage** ; В настоящее время предложено сообщение для отправки , или 0xff

**описание**

Держатель памяти для общей информации , связанной cSSL соединений.

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

**10.17.4.51 SSL_STUB_SIZE Макро**

**файл**

SSL.h

C

**#define SSL_STUB_SIZE** ((DWORD)**sizeof**(SSL_STUB))  / / Количество пространстве нужны один SSL заглушки

**описание**

Количество места, необходимого одним SSL заглушки

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

**10.17.4.52 SSL_STUB_SPACE Макро**

**файл**

SSL.h

C

**#define SSL_STUB_SPACE** (SSL_STUB_SIZE*MAX_SSL_CONNECTIONS)  / / количество места, необходимого всеми окурков SSL

**описание**

Объем пространства нуждаются все заглушки SSL

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

**10.17.4.53 SSL_VERSION Макро**

**файл**

SSL.h

C

**#define SSL_VERSION** (0x0300u) / / SSL номер версии

**описание**

SSL номер версии

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

**10.17.4.54 SSL_VERSION_HI Макро**

**файл**

SSL.h

C

**#define SSL_VERSION_HI** (0x03u)/ / SSL номер версии (старший байт )

**описание**

SSL номер версии (старший байт )

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

**10.17.4.55 SSL_VERSION_LO Макро**

**файл**

SSL.h

C

**#define SSL_VERSION_LO** (0x00u) / / SSL номер версии ( младший байт)

**описание**

SSL номер версии ( младший байт)

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

**10.17.4.56 SSLBufferAlloc Функция**

**файл**

SSL.c

C

```
static void SSLBufferAlloc(
BYTE * id
);
```

**Побочные эффекты**

Ни один

**Возвращает**

ID - выделенный буфер ID или SSL_INVALID_ID (см. стр. 467 ), если ни одна доступны

**описание**

Выделяет буфер для использования .

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

ID Где хранить выделенный ID

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

**10.17.4.57 SSLBufferFree Функция**

**файл**

SSL.c

C

```
static void SSLBufferFree(
BYTE * id
);
```

**Побочные эффекты**

Ни один

**описание**

Указано буфер освобождается

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

ID буфер ID для бесплатно

Вывод ID - SSL_INVALID_ID (см. Стр. 467 )

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

**10.17.4.58 sslBufferID Переменная**

**файл**

SSL.c

C

BYTE **sslBufferID**;

**описание**

Какие буфер загружен

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

**10.17.4.59 SSLBufferSync Функция**

**файл**

SSL.c

C

```
static void SSLBufferSync(
BYTE id
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Указано буфер загружается в оперативную память . Только загружает в случае необходимости, и сохраняет любой текущий буфер перед переключением .

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

ID буфер ID для синхронизации с ОЗУ

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

**10.17.4.60 SSLFinishPartialRecord Макро**

**файл**

SSL.h

C

```
#define SSLFinishPartialRecord(a) TCPSSLPutRecordHeader(a, NULL, TRUE);
```

**описание**

Это макро SSLFinishPartialRecord .

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

**10.17.4.61 SSLFlushPartialRecord Макро**

**файл**

SSL.h

C

```
#define SSLFlushPartialRecord(a) TCPSSLPutRecordHeader(a, NULL, FALSE);
```

**описание**

Это макро SSLFlushPartialRecord .

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

**10.17.4.62 sslHash Переменная**

**файл**

SSL.c

C

```
HASH_SUM sslHash;
```

**описание**

Хэш хранения

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

### 10.17.4.63 SSLHashAlloc Функция

**файл**

SSL.c

C

```
static void SSLHashAlloc(
BYTE * id
) ;
```

**Побочные эффекты**

Ни один

**описание**

Выделяет хэш для использования.

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

ID Где хранить выделенный ID

Вывод ID - Выделенные хэш ID или SSL_INVALID_ID (см. стр. 467 ), если ни одна доступны

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

### 10.17.4.64 SSLHashFree Функция

**файл**

SSL.c

C

```
static void SSLHashFree(
BYTE * id
);
```

**Побочные** эффекты

Ни один

**описание**

Указано хэш освобождается

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

ID хэш ID для бесплатно

Вывод ID - SSL_INVALID_ID (см. Стр. 467 )

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

### 10.17.4.65 sslHashID Переменная

**файл**

SSL.c

C

BYTE **sslHashID**;

**описание**

Какие хэш загружен

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

### 10.17.4.66 SSLHashSync Функция

**файл**

SSL.c

C

```
static void SSLHashSync(
BYTE id
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Указано хэш загружается в оперативную память . Только загружает в случае необходимости, и сохраняет любые текущие хэш перед переключением .

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

ID хэш ID для синхронизации с ОЗУ

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

**10.17.4.67 sslKeys Переменная**

**файл**

SSL.c

C

SSL_KEYS **sslKeys**;

**описание**

Нынешняя сессия SSL

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

### 10.17.4.68 sslKeysID Переменная

**файл**

SSL.c

C

BYTE **sslKeysID**;

**описание**

SSL_KEYS (см. стр. 496 ) , которые загружаются

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

### 10.17.4.69 SSLKeysSync Функция

**файл**

SSL.c

C

```
static void SSLKeysSync(
BYTE id
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Указано ключевое множество загружается в оперативную память . Только нагрузки в случае необходимости, и сохраняет любой текущий набор ключей , прежде чем переключится .

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

ID ключ набор ID для синхронизации с ОЗУ

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

**10.17.4.70 SSLMACAdd Функция**

**файл**

SSL.h

C

```
void SSLMACAdd(
BYTE* data,
WORD len
);
```

**описание**

Это функция SSLMACAdd .

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

### 10.17.4.71 SSLMACBegin Функция

**файл**

SSL.h

C

```
void SSLMACBegin(
BYTE* MACSecret,
DWORD seq,
BYTE protocol,
WORD len
);
```

**описание**

Это функция SSLMACBegin .

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

### 10.17.4.72 SSLMACCalc Функция

**файл**

SSL.h

C

```
void SSLMACCalc(
BYTE* MACSecret,
BYTE* result
);
```

**описание**

Это функция SSLMACCalc .

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

### 10.17.4.73 SSLRSAOperation Функция

**файл**

SSL.c

C

**static** RSA_STATUS **SSLRSAOperation**();

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Приостановка обработки соединения до расчет RSA не будет завершена.

**Замечания**

Эта функция существует вне функций квитирования так что система не берет на себя расходы по возобновлению и

приостановления рукопожатие хэшей.

**Предпосылки**

RSA модуль был обеспечен , операция RSA находится на рассмотрении , sslStub.wRxHsBytesRem является значение

sslStub.wRxBytesRem после завершения, и sslStub.wRxBytesRem есть значение sslStub.rxProtocol после завершения. Также требуется sslStub (см. стр. 518) для синхронизации .

**Раздел**

**Прототипы функций**

********** **********

**Криптографические функции расчета**

### 10.17.4.74 sslRSAStubID Переменная

**файл**

SSL.c

C

BYTE **sslRSAStubID**;

**описание**

Какие заглушка с использованием RSA , если таковые имеются

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

### 10.17.4.75 SSLRxAlert Функция

**файл**

SSL.c

C

```
static void SSLRxAlert(
TCP_SOCKET hTCP
);
```

**Побочные** эффекты

Ни один

**Возвращает**

Ни один

**описание**

Принимает предупреждающее сообщение и решает, что делать

**Замечания**

Ни один

**Предпосылки**

sslStub (см. стр. 518 ) синхронизируется

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, для чтения из

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

**10.17.4.76 SSLRxAntiqueClientHello Функция**

**файл**

SSL.c

C

```
static void SSLRxAntiqueClientHello(
TCP_SOCKET hTCP
);
```

**Побочные** эффекты

Ни один

**Возвращает**

Ни один

**описание**

Получает сообщение SSLv2 ClientHello , инициируя новый сеанс SSL с клиентом

**Замечания**

Это единственный SSLv2 сообщение мы поддерживаем , и предоставляется для браузеров , ищущих обратную совместимость . Связи должны быть обновлены до SSLv3.0 сразу после , в противном случае соединение не будет установлено .

**Предпосылки**

Рукопожатие мясорубка запускается , и SSL имеет заглушку , назначенный .

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, чтобы отправить сообщение

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

**10.17.4.77 SSLRxCCS Функция**

**файл**

SSL.c

C

```
static void SSLRxCCS(
TCP_SOCKET hTCP
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Получает ChangeCipherSpec с удаленного сервера

**Замечания**

Ни один

**Предпосылки**

sslStub (см. стр. 518 ) синхронизируется .

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, для чтения из

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

**10.17.4.78 SSLRxClientHello Функция**

**файл**

SSL.c

C

```
static void SSLRxClientHello(
TCP_SOCKET hTCP
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Получает сообщение ClientHello , инициируя новый сеанс SSL с клиентом

**Замечания**

Ни один

**Предпосылки**

Рукопожатие мясорубка запускается , и SSL имеет заглушку , назначенный .

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, чтобы отправить сообщение

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

### 10.17.4.79 SSLRxClientKeyExchange Функция

**файл**

SSL.c

C

```
static void SSLRxClientKeyExchange(
TCP_SOCKET hTCP
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Получает сообщение ClientKeyExchange и начинает процесс расшифровки.

**Замечания**

Ни один

**Предпосылки**

sslStub (см. стр. 518 ) синхронизируется и HSStart (см. стр. 488 ) ( ) был вызван .

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, для чтения из

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

### 10.17.4.80 SSLRxFinished Функция

**файл**

SSL.c

C

```
static void SSLRxFinished(
TCP_SOCKET hTCP
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Получает Готовые сообщение от удаленного узла

**Замечания**

Ни один

**Предпосылки**

sslStub (см. стр. 518 ) синхронизируется и HSStart (см. стр. 488 ) ( ) был вызван .

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, для чтения из

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

### 10.17.4.81 SSLRxHandshake Функция

**файл**

SSL.h

C

```
void SSLRxHandshake(
TCP_SOCKET hTCP
);
```

**Возвращает**

Ни один

**описание**

Эта функция принимает сообщения квитирования , считывает заголовок квитирования и передает данные с к соответствующей рукопожатие парсер .

**Предпосылки**

Указанный SSL заглушки инициализируется и сокет TCP подключен . Также требуется, чтобы rxBytesRem была заселена и текущий SSL заглушки была синхронизирована в память.

**Параметры**

**Параметры Описание**

HTCP Сокет TCP прочитать сообщение рукопожатие от

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

### 10.17.4.82 SSLRxRecord Функция

**файл**

SSL.h

C

```
WORD SSLRxRecord(
TCP_SOCKET hTCP,
BYTE sslStubID
);
```

**Возвращает**

WORD , указывающее количество байтов данных были расшифрованы , но оставили в потоке.

**описание**

Читает не более одного SSL заголовка Record из потока TCP и определяет , что делать с остальными данными . Если не все данные доступны для записи, то функция возвращает и будущее вызов (ы) в SSLRxRecord ( ) будет обрабатывать остальные Данные до конца записи достигается . Если данные процесса это звонок от прошлого записи, следующая запись не будет запущен до тех пор, следующий вызов .

**Замечания**

SSL запись заголовки , MAC нижние колонтитулы и асимметричная блока шифрования обивка (если таковые имеются ) будут извлечены из потока TCP этим

функция. Данные будут расшифрованы , но оставили в потоке.

**Предпосылки**

Указанный SSL заглушки инициализируется и сокет TCP подключен .

**Параметры**

**Параметры Описание**

HTCP The TCP сокет из которого будут считываться

ID Активный SSL заглушки ID

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

### 10.17.4.83 SSLRxServerCertificate Функция

**файл**

SSL.c

C

```
static void SSLRxServerCertificate(
TCP_SOCKET hTCP
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Принимает ServerCertificate с удаленного сервера , находит информацию открытого ключа , и выполняет операцию RSA.

**Замечания**

Это ярлыки полный разбор сертификата, просто найти идентификатор алгоритма с открытым ключом для RSA. Оттуда, Следующий ASN.1 структуру является открытым ключом . Это структура состоит из значение для N , после чего значение для E.

**Предпосылки**

sslStub (см. стр. 518 ) синхронизируется и HSStart (см. стр. 488 ) ( ) был вызван .

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, для чтения из

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

**10.17.4.84 SSLRxServerHello Функция**

**файл**

SSL.c

C

```
static void SSLRxServerHello(
TCP_SOCKET hTCP
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Получает ServerHello с удаленного сервера

**Замечания**

Ни один

**Предпосылки**

sslStub (см. стр. 518 ) синхронизируется и HSStart (см. стр. 488 ) ( ) был вызван .

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, для чтения из

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

**10.17.4.85 sslSession Переменная**

**файл**

SSL.c

C

SSL_SESSION **sslSession**;

**описание**

Текущие данные сессия

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

**10.17.4.86 sslSessionID Переменная**

**файл**

SSL.c

C

BYTE **sslSessionID**;
**описание**

Какие сессия загружается

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

### 10.17.4.87 SSLSessionMatchID Функция

**файл**

SSL.c

C

```
static BYTE SSLSessionMatchID(
BYTE* SessionID
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Согласованный ID сессии или SSL_INVALID_ID (см. стр. 467 ), если не найдено

**описание**

Находит в кэше SSL сессию для повторного использования. Синхронизация найдено сеанс в ОЗУ .

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

Код_сеанса идентификатор сессии в соответствии

раздел сообщения сервера

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

**10.17.4.88 SSLSessionMatchIP Функция**

**файл**

SSL.c

C

```
static BYTE SSLSessionMatchIP(
IP_ADDR ip
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Согласованный ID сессии или SSL_INVALID_ID (см. стр. 467 ), если не найдено

**описание**

Находит в кэше SSL сессию для повторного использования

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

И.П. пройдет заседание , чтобы соответствовать раздел  сообщения Клиент

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

**10.17.4.89 SSLSessionNew Функция**

**файл**

SSL.c

C

**static** BYTE **SSLSessionNew**();

**Побочные эффекты**

Ни один

**Возвращает**

Выделенные ID сессии или SSL_INVALID_ID (см. стр. 467 ), если ни одна доступны

**описание**

Находит место для нового сессии SSL

**Замечания**

Ни один

**Предпосылки**

Ни один

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

### 10.17.4.90 sslSessionStubs Переменная

**файл**

SSL.c

C

SSL_SESSION_STUB **sslSessionStubs**[MAX_SSL_SESSIONS];
**описание**

8 байт сессия заглушки

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

### 10.17.4.91 SSLSessionSync Функция

**файл**

SSL.c

C

```
static void SSLSessionSync(
BYTE id
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Указано сессия загружается в оперативную память . Только загружает в случае необходимости, и сохраняет любой текущий сеанс перед переключением , если он был

обновляется.

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

ID идентификатор сеанса для синхронизации с ОЗУ

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

**10.17.4.92 SSLSessionUpdated Макро**

**файл**

SSL.c

C

```
#define SSLSessionUpdated sslSessionUpdated = TRUE;
```

**описание**

Это макро SSLSessionUpdated .

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

**10.17.4.93 sslSessionUpdated Переменная**

**файл**

SSL.c

C

```
BOOL sslSessionUpdated;
```

**описание**

Или нет, это был обновлен

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

### 10.17.4.94 SSLStartPartialRecord Функция

**файл**

SSL.h

C

```
void SSLStartPartialRecord(
TCP_SOCKET hTCP,
BYTE sslStubID,
BYTE txProtocol,
WORD wLen
);
```

**Возвращает**

Ни один

**описание**

Эта функция позволяет сообщения больше , чем буфер TCP для отправки , что это часто бывает на сертификат рукопожатие сообщение . Окончательный длина сообщения должны быть известны , чтобы передавать заголовок . После того, как называется ,

SSLFlushPartialRecord (см. стр. 504 ) и SSLFinishPartialRecord (см. стр. 504 ) должна быть вызвана написать остальные Данные , завершить , и подготовиться к новой записи.

**Замечания**

Частичные сообщения не поддерживают нынешний шифра спецификации , так что это можно использовать только во время процедуры рукопожатия .

**Предпосылки**

Указанный SSL заглушки инициализируется и сокет TCP подключен .

**Параметры**

**Параметры Описание**

HTCP The TCP сокет с данных и ждут, чтобы быть переданы

ID Активный SSL заглушки ID

txPortocolномер протокола SSL для подключения к этой записи

Wlen Длина всех данных для отправки

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

**10.17.4.95 sslStub Переменная**

**файл**

SSL.c

C

SSL_STUB **sslStub**;
**описание**

Нынешний SSL заглушки

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

**10.17.4.96 SSLStubAlloc Функция**

**файл**

SSL.c

C

**static** BOOL **SSLStubAlloc**();

**Побочные эффекты**

Ни один

**Возвращает**

ИСТИНА, если шейка было выделено , иначе ложь

**описание**

Выделяет заглушку для использования.

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

Выходные данные Нет

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

**10.17.4.97 SSLStubFree Функция**

**файл**

SSL.c

C

```
static void SSLStubFree(
BYTE id
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Указано заглушка выпущен

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

ID заглушка ID для бесплатно

Выходные данные Нет

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

**10.17.4.98 sslStubID Переменная**

**файл**

SSL.c

C

BYTE **sslStubID**;
**описание**

SSL_STUB (см. стр. 500 ) , который загружается

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

**10.17.4.99 SSLStubSync Функция**

**файл**

SSL.c

C

```
static void SSLStubSync(
BYTE id
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Указано заглушки загружается в оперативную память . Только загружает в случае необходимости, и сохраняет любые текущие заглушку перед переключением .

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

ID заглушка ID для синхронизации с ОЗУ

**раздел**

Ethernet Буфер ОЗУ управления

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

### 10.17.4.100 SSLTerminate Функция

**файл**

SSL.h

C

```
void SSLTerminate(
BYTE sslStubId
);
```

**Возвращает**

Ни один

**описание**

Завершает соединение SSL и освобождает выделенные ресурсы.

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

IDSSL заглушки ID прекратить

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

### 10.17.4.101 SSLTxCCSFin Функция

**файл**

SSL.c

C

```
static void SSLTxCCSFin(
TCP_SOCKET hTCP
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Генерирует сеансовых ключей от главного секрета , то выделяет и формирует контекст шифрования. После обработки завершена, передает сообщение Изменить Cipher Spec и готовой рукопожатие сообщение на сервер.

**Замечания**

Ни один

**Предпосылки**

sslStub (см. стр. 518 ) синхронизируется , и нынешняя сессия имеет действительный предварительно главный секрет в использовании.

**Параметры**

**Параметры Описание**

HTCPсокета TCP написать сообщение для раздел

Клиентские и серверные сообщения

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

**10.17.4.102 SSLTxClientHello Функция**

**файл**

SSL.c

C

```
static void SSLTxClientHello(
TCP_SOCKET hTCP
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Передает сообщение ClientHello инициировать новый сеанс SSL с сервером.

**Замечания**

Ни один

**Предпосылки**

Достаточно места доступен в HTCP написать сообщение целиком .

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, чтобы отправить сообщение

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

**10.17.4.103 SSLTxClientKeyExchange Функция**

**файл**

SSL.c

C

```
static void SSLTxClientKeyExchange(
TCP_SOCKET hTCP
);
```

**Побочные** эффекты

Ни один

**Возвращает**

Ни один

**описание**

Передает зашифрованное предварительно главный секрет к серверу и просит сменить Cipher Spec. Также генерируетМастер

Секрет от предварительно главного секрета , который был использован .

**Замечания**

Ни один

**Предпосылки**

sslStub (см. стр. 518 ) синхронизируется , sslStub.dwTemp.v [1] содержит длину открытого ключа , и RxBuffer содержит зашифрованную предварительно главный секрет в адрес 0x80.

**Параметры**

**Параметры Описание**

HTCPсокета TCP написать сообщение для

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

### 10.17.4.104 SSLTxMessage Функция

**файл**

SSL.h

C

```
void SSLTxMessage(
TCP_SOCKET hTCP,
BYTE sslStubID,
BYTE msg
);
```

**Возвращает**

Ни один

**описание**

Эта функция передает определенное сообщение SSL для рукопожатий и предупреждений . Поддерживаемые Сообщения перечислены в SSL_MESSAGES (см. стр. 497 ) .

**Предпосылки**

Указанный SSL заглушки инициализируется и сокет TCP подключен .

**Параметры**

**Параметры Описание**

HTCP The TCP сокет с данных и ждут, чтобы быть переданы

ID Активный SSL заглушки ID

MSG Один из SSL_MESSAGES (см. стр. 497 ) Типы для отправки

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

### 10.17.4.105 SSLTxRecord Функция

**файл**

SSL.h

C

```
void SSLTxRecord(
TCP_SOCKET hTCP,
BYTE sslStubID,
BYTE txProtocol
);
```

**Возвращает**

Ни один

**описание**

Передает все отложенные данные в буфере TCP TX как запись SSL , используя указанный протокол . Эта функция прозрачно шифрует и ПДК данные, если есть активная шифр спецификации .

**Предпосылки**

Указанный SSL заглушки инициализируется и сокет TCP подключен .

**Параметры**

**Параметры Описание**

HTCP The TCP сокет с данных и ждут, чтобы быть переданы ID Активный SSL заглушки ID

txPortocolномер протокола SSL для подключения к этой записи

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

### 10.17.4.106 SSLTxServerCertificate Функция

**файл**

SSL.c

C

```
static void SSLTxServerCertificate(
TCP_SOCKET hTCP
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Передает сообщение сертификат с указанным сертификатом открытого ключа сервера.

**Замечания**

Сертификат определяется в CustomSSLCert.c . Эта функция требует специальной обработки для частичного записей , так как сертификат, вероятно, будет больше, чем буфер TCP и SSL сообщения рукопожатие ограничены , чтобы поместиться в одной SSL рукопожатия запись

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, чтобы отправить сообщение

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

### 10.17.4.107 SSLTxServerHello Функция

**файл**

SSL.c

C

```
static void SSLTxServerHello(
TCP_SOCKET hTCP
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Передает сообщение ServerHello .

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, чтобы отправить сообщение

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

**10.17.4.108 SSLTxServerHelloDone Функция**

**файл**

SSL.c

C

```
static void SSLTxServerHelloDone(
TCP_SOCKET hTCP
);
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Передает сообщение ServerHelloDone .

**Замечания**

Ни один

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

HTCP в TCP Socket, чтобы отправить сообщение

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

**10.17.5 Файлы**

**Файлы**

**Имя Описание**

SSLClientSize.h (см. стр. 525 ) Это файл SSLClientSize.h .

модульSSL (см. стр. 462 )

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

**10.17.5.1 SSLClientSize.h**

**макрос**

**Имя Описание**

**SSL_RSA_CLIENT_SIZE** (см. стр. 471) Размер Encryption буфера ( должен быть больше, чем размер ключа )

**описание**

Это файл SSLClientSize.h .

**10.18 TCP**

**Variables**

**Name Description**

NextPort (см. стр. 571 ) переменную отслеживания в течение следующего числа локальный порт клиента

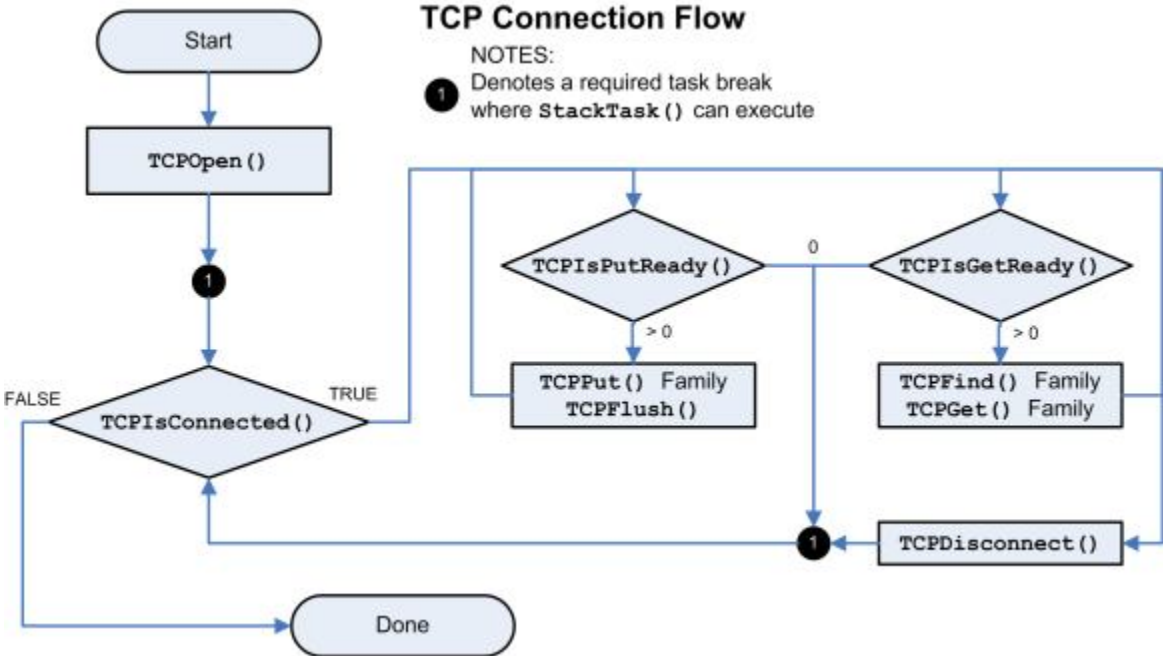
**Description**

TCP является стандартным протоколом транспортного уровня описано в RFC 793. Он обеспечивает надежный поток на основе соединений через ненадежные сети и формирует основу для HTTP, SMTP, и многих других стандартов протоколов .

Для соединений, производимых над передачи данных гарантийного TCP за счет пропускной способности. Подключения осуществляются через трехсторонний процесс рукопожатие , обеспечивая связь один-к -одному . Удаленные узлы рекламы , сколько данных они готовы получать , и все данные, передаваемые следует признать . Если удаленный узел не в состоянии подтвердить о получение данных , это автоматически повторяется. Это гарантирует, что ошибки сети , такие как потерянные или испорченные , автоматически корректируется .

Чтобы достичь этого, TCP должен работать в буфере . Как только буфер передачи заполнен, нет больше данных не могут быть отправлены до пульте дистанционного управления узел подтвердил получение . Для Microchip Стек TCP / IP , приложение должно вернуться в основной цикл стека в порядке чтобы это произошло . Кроме того, удаленный узел не может передавать больше данные, пока локальное устройство не подтвердил о получение и что пространство в буфере . Когда локальное приложение необходимо прочитать больше данных , он должен вернуться в основной цикл стека и ждать нового пакета приехать.

Диаграмма TCP поток ниже приводится обзор для использования модуля TCP :



Розетки ( см. стр. 152 ) открываются с помощью TCPOpen (см. стр. 542 ) . Эта функция может либо открыть сокет для ждать клиентских подключений , или может сделать подключение клиента к удаленному узлу . Удаленный узел может быть указан хозяина назвать строку, которая будет решена в DNS , в IP -адрес или NODE_INFO структура, содержащая ранее разрешенный IP и MAC адресную информацию .

После подключения приложения могут считывать и записывать данные . На каждой записи, приложение должно проверить, что сокет по-прежнему подключен . Для большинства приложений вызов TCPIsConnected (см. стр. 540 ) будет достаточно , но TCPWasReset (см. страница 549 ) также могут быть использованы для прослушивания сокета, которые могут перевернуть быстро.

Для записи данных , звоните TCPIsPutReady (см. стр. 541 ), чтобы проверить , сколько места доступно . Затем вызвать любой из TCPPut (см. стр. 545 ) семейство функций для записи данных , как свободного места . После завершения вызова TCPFlush (см. стр. 537 ), чтобы немедленно передавать данные . С другой стороны, вернуться в основной цикл стека. Данные будут передаваться , когда либо а) половина передавать переполнении буфера или б) время задержки прошло (обычно 40мс ) .

Для чтения данных , звоните TCPIsGetReady (см. стр. 541 ), чтобы определить , сколько байтов готовы быть восстановлен. Затем с помощью TCPget ( см. стр. 537 ) семейство функций для чтения данных из сокета , и / или TCPFind ( см. стр. 534) семейство функции для поиска данных в буфере . Когда не остается больше данных , вернуться в основной цикл стека ждать больше данных в приехать.

Если приложение должно закрыть соединение , звоните TCPDisconnect (см. стр. 533 ) , а затем вернуться в основной цикл стека и ждать удаленного узла признать отключение . Сокеты клиента вернется в исходное состояние , во время прослушивания сокет будет ждать нового соединения .

Для получения дополнительной информации , обратитесь к GenericTCPClient (см. стр. 98 ) или GenericTCPServer (см. стр. 101 ) примеры , или читать соответствующую RFC .

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

### 10.18.1 TCP Public Members

**Functions**
**Name Description**

**TCPAdjustFIFOSize** (см.страница 531 )

Регулирует относительные размеры буферов приема и передачи .

**TCPClose** (см. стр. 532 ) Отключает открытый сокет и разрушает дескриптор сокета , в том числе сервер Режим розетка ручки .

**TCPDiscard** (см. стр. 533 ) Удаляет все отложенные данные в TCP RX FIFO.

**TCPDisconnect** (см. стр.533)Отключает открытый сокет .

**TCPFindArrayEx** (см.страница 534)Ищет строку в буфере TCP RX.

**TCPFindEx** (см. стр. 535 ) Ищет байт в буфере TCP RX.

**TCPFindROMArrayEx** (см.страница 536) Ищет ROM строки в буфере TCP RX .

**TCPFlush** (см. стр. 537 ) Сразу передает все ожидающие данные TX .

**TCPget** (см. стр. 537 ) Получает один байт в сокет TCP .

**TCPGetArray** (см. стр.538) Читает массив байтов данных из TCP сокета получения FIFO. Данные удаляются из FIFO в процессе.

**TCPGetRemoteInfo** (см.страница 538 ) Получает сведения о текущей открытой розетки.

**TCPGetRxFIFOFree** (см.страница 539 ) Определяет , сколько байт свободны в RX FIFO.

**TCPGetTxFIFOFull** (см. страница 540 ) Определяет , сколько байтов на рассмотрении в TCP TX FIFO.

**TCPIsConnected** (см.страница 540 ) Определяет, является ли сокет установил соединение .

**TCPIsGetReady** (см. стр.541)Определяет, сколько байт можно прочитать из буфера TCP RX.

**TCPIsPutReady** (см. стр.541) Определяет , сколько свободного места доступно в буфере TCP TX.

**TCPOpen** (см. стр. 542) Открывает TCP сокет для прослушивания или в качестве клиента .

**TCPPeek** (см. стр. 544 ) Peaks на одном байта в TCP RX FIFO , не удаляя его из буфера.

**TCPPeekArray** (см. стр. 544) Считывает заданное число байт данных из TCP RX FIFO без удаляя их из буфера.

**TCPPut** (см. стр. 545 ) Записывает один байт в сокет TCP .

**TCPPutArray** (см. стр.545)Записывает массив из оперативной памяти на сокет TCP .

**TCPPutROMArray** (см.страница 546)Записывает массив из ПЗУ в сокет TCP .

**TCPPutROMString** (см.страница 546)Записывает строку с завершающим нулем из ПЗУ в сокет TCP . Нуль- терминатор не копируется в сокет.

**TCPPutString** (см. стр.547 ) Записывает строку с завершающим нулем из оперативной памяти на сокет TCP . Нуль- терминатор не копируется в сокет.

**TCPRAMCopy** (см. стр. 547 ) Данные копии в / из различных средах памяти.

**TCPRAMCopyROM** (см. страница 548 ) Данные копии в / из различных средах памяти.

**TCPWasReset** (см. стр. 549) Self- очистка семафор inidicating сброс сокета .

**Macros**
**Name Description**
**INVALID_SOCKET** (см. стр. 529 ) Розетка является недействительным или не может быть открыт

**UNKNOWN_SOCKET** (см. стр.529)Гнездо не известно

**TCP_ADJUST_GIVE_REST_TO_RX**(См. стр. 529 )Изменение размера флаг : дополнительные байты перейти к RX

**TCP_ADJUST_GIVE_REST_TO_TX**(См. стр. 529 )Изменение размера флаг : дополнительные байты идти в TX

**TCP_ADJUST_PRESERVE_RX** (см. стр. 529 )Изменение размера флаг : попытаться сохранить RX буфер

**TCP_ADJUST_PRESERVE_TX** (см. стр. 530)Изменение размера флаг : попытаться сохранить TX буфер

**TCP_OPEN_IP_ADDRESS** (см.страница 530 )Выделяют необъявленную идентификатор диагностическую если код пытается использовать TCP_OPEN_IP_ADDRESS а функция **STACK_CLIENT_MODE** является не включен .

**TCP_OPEN_NODE_INFO** (см.страница 530 ) Выделяют необъявленную идентификатор диагностическую если код пытается использовать

**TCP_OPEN_NODE_INFO** а функция STACK_CLIENT_MODE не включен .

**TCP_OPEN_RAM_HOST** (см. страница 530 ) Выделяют необъявленную идентификатор диагностическую если код пытается использовать

**TCP_OPEN_RAM_HOST** а функция STACK_CLIENT_MODE не включен .

**TCP_OPEN_ROM_HOST** (см.страница 531 )Выделяют необъявленную идентификатор диагностическую если код пытается использовать

**TCP_OPEN_ROM_HOST** а функция STACK_CLIENT_MODE не включен .

**TCP_OPEN_SERVER** (см. стр. 531) Создание серверного сокета и игнорировать dwRemoteHost .

**TCPConnect** ( см. стр. 532) псевдоним для TCPOpen (см. стр. 542 ) в качестве клиента.

**TCPFind** ( см. стр. 534) псевдоним для TCPFindEx (см. стр. 535 ) без параметров длины .

**TCPFindArray** ( см. стр. 534) псевдоним для TCPFindArrayEx (см. стр. 534 ) без параметров длины .

**TCPFindROMArray** (см. стр. 536) Псевдоним для TCPFindROMArrayEx (см. стр. 536) с не длина

параметр .

**TCPGetRxFIFOFull** (см. стр. 539 ) Псевдоним для TCPIsGetReady (см. стр. 541 ) предусмотрено API

полнота

**TCPGetTxFIFOFree** (см. стр. 540 ) Псевдоним в TCPIsPutReady ( см. стр. 541 ) предусмотрено API

полнота

**TCPListen** ( см. стр. 542) псевдоним для TCPOpen (см. стр. 542 ) в качестве сервера .

**Module**
TCP ( see page 526)

**Description**
Следующие функции и переменные доступны для применения стека.

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

### 10.18.1.1 INVALID_SOCKET Macro

**File**
TCP.h
**C**
```
#define INVALID_SOCKET (0xFE) // сокет является недействительным или не может быть открыт
```
**Description**
The socket является недействительным или не может быть открыт

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

### 10.18.1.2 UNKNOWN_SOCKET Macro

**File**
TCP.h
**C**
```
#define UNKNOWN_SOCKET (0xFF) // сокет не известно
```

*описание*

**Description**
The socket не известно

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

### 10.18.1.3 TCP_ADJUST_GIVE_REST_TO_RX Macro

**File**
TCP.h
**C**
```
#define TCP_ADJUST_GIVE_REST_TO_RX 0x01u // Изменить размер флага : дополнительные байты перейти к RX
```

**Description**
Изменение размера флаг : дополнительные байты перейти к RX

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

### 10.18.1.4 TCP_ADJUST_GIVE_REST_TO_TX Macro

**File**
TCP.h
**C**
```
#define TCP_ADJUST_GIVE_REST_TO_TX 0x02u / / Изменить размер флага : дополнительные байты идти в Техас
```

**Description**
Изменение размера флаг : дополнительные байты идти в TX

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

### 10.18.1.5 TCP_ADJUST_PRESERVE_RX Macro

**File**
TCP.h
**C**
```
#define TCP_ADJUST_PRESERVE_RX 0x04u / / Изменить размер флаг : попытка сохранить RX буфер
```

**Description**
Изменение размера флаг : попытаться сохранить RX буфер

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

### 10.18.1.6 TCP_ADJUST_PRESERVE_TX Macro

**File**

TCP.h

**C**

```
#define TCP_ADJUST_PRESERVE_TX 0x08u / / Изменить размер флаг : попытка сохранить TX буфер
```

**Description**

Изменение размера флаг : попытаться сохранить TX буфер

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

### 10.18.1.7 TCP_OPEN_IP_ADDRESS Macro

**File**

TCP.h

**C**

```
#define TCP_OPEN_IP_ADDRESS
You_need_to_enable_STACK_CLIENT_MODE_to_use_TCP_OPEN_IP_ADDRESS
```

**Description**

Выделяют необъявленную идентификатор диагностическую если код пытается использовать TCP_OPEN_IP_ADDRESS while STACK_CLIENT_MODE
функция не включена .

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

### 10.18.1.8 TCP_OPEN_NODE_INFO Macro

**File**

TCP.h

**C**

```
#define TCP_OPEN_NODE_INFO
You_need_to_enable_STACK_CLIENT_MODE_to_use_TCP_OPEN_NODE_INFO
```

**Description**

Выделяют необъявленную идентификатор диагностическую если код пытается использовать TCP_OPEN_NODE_INFO while STACK_CLIENT_MODE не включен .

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

#### 10.18.1.9 TCP_OPEN_RAM_HOST Macro

**File**
TCP.h
**C**
**#define TCP_OPEN_RAM_HOST**
You_need_to_enable_STACK_CLIENT_MODE_to_use_TCP_OPEN_RAM_HOST
**Description**
Выделяют необъявленную идентификатор диагностическую если код пытается использовать TCP_OPEN_RAM_HOST while STACK_CLIENT_MODE не включен .

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

#### 10.18.1.10 TCP_OPEN_ROM_HOST Macro

**File**
TCP.h
**C**
**#define TCP_OPEN_ROM_HOST**
You_need_to_enable_STACK_CLIENT_MODE_to_use_TCP_OPEN_ROM_HOST
**Description**
Выделяют необъявленную идентификатор диагностическую если код пытается использовать TCP_OPEN_ROM_HOST while STACK_CLIENT_MODE не включен .

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

#### 10.18.1.11 TCP_OPEN_SERVER Macro

**File**
TCP.h
**C**
**#define TCP_OPEN_SERVER** 0u
**Description**
Создание серверного сокета и игнорировать dwRemoteHost .

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

#### 10.18.1.12 TCPAdjustFIFOSize Function

**File**
TCP.h

**C**
BOOL **TCPAdjustFIFOSize**(

TCP_SOCKET **hTCP**,
WORD **wMinRXSize**,
WORD **wMinTXSize**,
BYTE **vFlags**
);

**Side Effects**
Любые неподтвержденные или не переданные данные в TX FIFO всегда удаляется .

**Description**
Эта функция может быть использована для регулировки относительные размеры в RX и TX FIFO в зависимости от насущных потребностей

Приложение . Поскольку больше FIFO может позволить больше данных для передачи в данном пакете , регулируя относительные размеры на лету может позволяют оптимальной скорости передачи для односторонних прикладных протоколов . Например, HTTP , как правило, начинается с приема большие объемы данных от клиента , затем переключается на обслуживание больших объемов данных обратно . Регулировка FIFO на них точки могут увеличить производительность существенно . После того, как FIFO корректируется ,обновление окна посылается .

Если ни один или оба TCP_ADJUST_GIVE_REST_TO_TX (см. стр. 529 ) и TCP_ADJUST_GIVE_REST_TO_RX (см.страница 529 ) установлены , функция распределяет оставшееся пространство в равной степени.

Полученные данные могут быть сохранены тех пор, пока буфер расширяется и не обернуты .

**Remarks**
По крайней мере один байт всегда должны быть выделены в буфер RX так, чтобыFIN могут быть получены. Функция автоматически корректирует для этого.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
hTCP The socket to be adjusted
wMinRXSize Minimum number of byte for the RX FIFO
wMinTXSize Minimum number of bytes for the RX FIFO
vFlags Any combination of TCP_ADJUST_GIVE_REST_TO_RX ( see page 529),
TCP_ADJUST_GIVE_REST_TO_TX ( see page 529),
TCP_ADJUST_PRESERVE_RX ( see page 529).
TCP_ADJUST_PRESERVE_TX ( see page 530) в настоящее время не поддерживается.

**Return Values**
**Return Values Description**
**TRUE** The FIFOs были успешно регулировать
**FALSE** Минимальная RX , Минимальная TX , или флаги не могут быть размещены и, следовательно, сокет был оставлен без изменения .

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

### 10.18.1.13 TCPConnect Macro

**File**
TCP.h

**C**
```
#define TCPConnect(remote,port) TCPOpen((DWORD)remote, TCP_OPEN_NODE_INFO, port, TCP_PURPOSE_DEFAULT)
```

**Description**
Эта функция является псевдонимом для **TCPOpen** (см. стр. 542 ) для клиентских сокетов. Она предназначена для обеспечения обратной совместимости с более старыми версии стека . Новые приложения должны использовать **TCPOpen** (см. стр. 542 ) API вместо этого.

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

### 10.18.1.14 TCPClose Function

**File**
TCP.h

**C**
```
void TCPClose(
TCP_SOCKET hTCP
);
```

**Returns**
None

**Description**
Отключает открытый сокет и разрушает дескриптор сокета , в том числе сокетов ручками режиме сервера . Эта функция выполняет идентично **TCPDisconnect** (см. стр. 533 ) функция ( ) , за исключением того, как клиент и режим сервера гнездо ручки отказался от участия в TCP / IP стека по возвращении .

**Preconditions**
None
**Parameters**
**Parameters Description**
hTCP Handle to the socket to disconnect and close.

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

### 10.18.1.15 TCPDiscard Function

**File**
TCP.h

**C**
```
void TCPDiscard(
TCP_SOCKET hTCP
);
```

**Returns**
None

**Description**
Отменяет все отложенные данные в TCP RX FIFO.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
HTCP Сокет которого RX FIFO должен быть очищен.

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

### 10.18.1.16 TCPDisconnect Function

**File**
TCP.h

**C**
```
void TCPDisconnect(
TCP_SOCKET hTCP
);
```

**Returns**
None

**Description**
Эта функция закрывает подключение к удаленному узлу , отправив FIN (если в данный момент он подключен) .

Функция может быть вызвана во второй раз , чтобы заставить сокет закрыт , отправив RST пакет . Это полезно, когда приложение знает , что удаленный узел не будет посылать ACK (если он вышел из строя или потерял связь ) , или когда потребности приложений повторно использовать гнездо немедленно, независимо от того, удаленный узел хотели бы передавать больше данных перед закрытия .

Для сокетов в режиме клиент , по возвращении , ручка HTCP освобождается в IP стек TCP / и не должны больше использоваться приложение (к немедленному последующим вызовом **TCPDisconnect** ( ) , чтобы заставить передачу RST , в случае необходимости , за исключением) .

Для сокетов в режиме сервера , по возвращении , ручка HTCP НЕ отказался в стек TCP / IP. После закрытия , сокет возвращается в состояние прослушивания позволяет будущие запросы на подключение , которые будут обслуживаться . Это оставляет ручку HTCP в допустимом состоянии и должны быть сохранены для будущих операций на гнездо . Если вы хотите , чтобы закрыть сервер и отказаться сокет обратно IP стека TCP / , вызовите **TCPClose** (см. стр. 532 ) ( ) API вместо **TCPDisconnect** ( ) .

**Remarks**
Если сокет с использованием SSL , запись CLOSE_NOTIFY сначала будет передан , чтобы позволить SSL сессии, которая возобновляется в более позднее время .

**Preconditions**
None

**Parameters**
**Parameters Description**
hTCP Handle of the socket to disconnect.

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

### 10.18.1.17 TCPFind Macro

**File**
TCP.h

**C**
```
#define TCPFind(a,b,c,d) TCPFindEx(a,b,c,0,d)
```

**Description**
Эта функция является псевдонимом для **TCPFindEx** (см. стр. 535 ) без параметров длины . Она предназначена для обеспечения обратной совместимости с более старой API.

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

### 10.18.1.18 TCPFindArray Macro

**File**
TCP.h

**C**
**#define TCPFindArray**(a,b,c,d,e) **TCPFindArrayEx**(a,b,c,d,0,e)

**Description**
Эта функция является псевдонимом для **TCPFindArrayEx** (см. стр. 534 ) без параметров длины. Это обеспечивается для обратной совместимость с более старым API.

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

10.18.1.19 **TCPFindArrayEx Function**
**File**
TCP.h
**C**
WORD **TCPFindArrayEx**(
TCP_SOCKET **hTCP**,
BYTE* **cFindArray**,
WORD **wLen**,
WORD **wStart**,
WORD **wSearchLen**,
BOOL **bTextCompare**
);
**Description**
Эта функция находит первое occurrance массива байтов в буфере TCP RX. Он может быть использован приложением к абстрактной поиск из своего кода приложения. Для повышения эффективности , функция может ограничение объема поиска, чтобы определенный диапазон байт. Он также может выполнять поиск без учета регистра , если требуется. Например, если буфер содержит "Я люблю ПОС микроконтроллеров ! " и поиск массив "любовь" с длиной 4 , значение 2 будет вернулся .

**Remarks**
Так как эта функция как правило, должны передать данные с внешних накопителей , чтобы внутреннем ОЗУ для сравнения , его производительность значительно снижается, если буфер заполнен и массив не найден . Для повышения производительности , попробуйте поискать для персонажей что , как ожидается, существует или ограничить область поиска , насколько это возможно . Модуль http2 , например, использует этот функционировать для разбора заголовков . Тем не менее, он ищет для переноса строк , то разделения толстой кишки, затем считывает имя заголовка в ОЗУ для окончательного сравнения. Это оказалось значительно быстрее , чем при поиске полных строк имени заголовка напрямую.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
**hTCP** The Сокет для поиска внутри .
**cFindArray** Массив байтов , чтобы найти в буфере .
**wLen** Length of cFindArray.
**wStart** Zero- нулевой индексируются стартовая позиция в буфере .

**wSearchLen** Длина от wStart искать в буфере .
**bTextCompare** TRUE для учета регистра текстового поиска , FALSE для бинарного поиска

**Return Values**
**Return Values Description**
0xFFFF Search array not found
В противном случае нулевую позицию индекса первой occurrance

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

### 10.18.1.20 TCPFindEx Function

**File**
TCP.h

**C**
WORD **TCPFindEx**(
TCP_SOCKET **hTCP**,
BYTE **cFind**,
WORD **wStart**,
WORD **wSearchLen**,
BOOL **bTextCompare**
);

**Description**
Эта функция находит первое occurrance из байта в буфере TCP RX. Он может быть использован приложением к абстрактным поисках из собственного кода приложения. Для повышения эффективности , функция может ограничение объема поиск определенной диапазон байт . Он также может выполнять поиск без учета регистра , если требуется.

Например, если буфер содержит "Я люблю ПОС микроконтроллеров ! " и байт cFind является' ' , значение 1 , будут возвращены.

**Remarks**
Так как эта функция как правило, должны передать данные с внешних накопителей , чтобы внутреннем ОЗУ для сравнения , его производительность значительно снижается, если буфер заполнен и массив не найден . Для повышения производительности , попробуйте поискать для персонажей что , как ожидается, существует или ограничить область поиска , насколько это возможно . Модуль http2 , например, использует этот функцию для разбора заголовков . Тем не менее, он ищет для переноса строк , то разделения толстой кишки, затем считывает имя заголовка в ОЗУ для окончательного сравнения. Это оказалось значительно быстрее , чем при поиске полных строк имени заголовка напрямую.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**

hTCP The socket  Гнездо для поиска внутри.
cFind Байт найти в буфере .
wStart нулевой индексируются стартовая позиция в буфере .
wSearchLen Длина от wStart искать в буфере .
bTextCompare TRUE для учета регистра текстового поиска , FALSE для бинарного поиска

**Return Values**
**Return Values Description**
0xFFFF Поиск массив не найдено
В противном случае нулевую позицию индекса первой occurrance

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

### 10.18.1.21 TCPFindROMArray Macro

**File**
TCP.h

**C**
```
#define TCPFindROMArray(a,b,c,d,e) TCPFindArray(a,(BYTE*)b,c,d,e)
```

**Description**
Эта функция является псевдонимом для TCPFindROMArrayEx (см. стр. 536 ) без параметров длины . Это обеспечивается для обратной совместимость с более старым API.

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

### 10.18.1.22 TCPFindROMArrayEx Function

**File**
TCP.h
**C**
```
WORD TCPFindROMArrayEx(
TCP_SOCKET hTCP,
ROM BYTE* cFindArray,
WORD wLen,
WORD wStart,
WORD wSearchLen,
BOOL bTextCompare
);
```
**Description**
Эта функция находит первое occurrance массива байтов в буфере TCP RX. Он может быть использован приложением к абстрактной поиск из своего кода приложения. Для повышения эффективности , функция может ограничение объема поиска, чтобы определенный диапазон байт. Он также может выполнять поиск без учета регистра , если требуется.

Например, если буфер содержит "Я люблю ПОС микроконтроллеров ! " и поиск массив "любовь" с длиной 4 , значение 2 будет вернулся .

**Remarks**
Так как эта функция как правило, должны передать данные с внешних накопителей , чтобы внутреннем ОЗУ для сравнения , его производительность значительно снижается, если буфер заполнен и массив не найден . Для повышения производительности , попробуйте поискать для персонажей что , как ожидается, существует или ограничить область поиска , насколько это возможно . Модуль http2 , например, использует этот функционировать для разбора заголовков . Тем не менее, он ищет для переноса строк , то разделения толстой кишки, затем считывает имя заголовка в ОЗУ для окончательного сравнения. Это оказалось значительно быстрее , чем при поиске полных строк имени заголовка напрямую.

Эта функция является псевдонимом для TCPFindArrayEx (см. стр. 534 ) на не- PIC18 платформ.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
**hTCP** The socket to search within.
**cFindArray** Массив байтов , чтобы найти в буфере .
**wLen Length** of cFindArray.
**wStart Zero** нулевой индексируются стартовая позиция в буфере .wSearchLen Length from wStart to search in the buffer.
bTextCompare TRUE for case-insensitive text search, FALSE for binary search

**Return Values**
**Return Values Description**
0xFFFF Search array not found
В противном случае нулевую позицию индекса первой occurrance

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

### 10.18.1.23 TCPFlush Function

**File**
TCP.h

**C**
```
void TCPFlush(
TCP_SOCKET hTCP
);
```

**Returns**
None

**Description**
Эта функция немедленно передает все ожидающие данные TX с флагом PSH . Если эта функция не вызывается , данные автоматически быть отправлены , когда либо а) буфер TX наполовину полон или б)TCP_AUTO_TRANSMIT_TIMEOUT_VAL (см. стр. 565) ( по умолчанию: 40мс ) истекло.

**Remarks**
Данные приложения SSL автоматически покраснел , поэтому эта функция не имеет никакого эффекта для SSL сокетов.

**Preconditions**
TCP is initialized and the socket is connected.

**Parameters**
**Parameters Description**
hTCP The socket данные должны быть переданы .

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

### 10.18.1.24 TCPGet Function

**File**
TCP.h

**C**
BOOL **TCPGet**(
TCP_SOCKET **hTCP**,
BYTE* **byte**
);

**Description**
Получает один байт в сокет TCP .

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
hTCP Сокет , из которого , чтобы прочитать .

байт Указатель на месте , в котором байт чтения должны быть сохранены .

**Return Values**
**Return Values Description**
TRUE байт было прочитано из буфера.

FALSE Буфер был пуст или сокет не подключен.

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

### 10.18.1.25 TCPGetArray Function

**File**
TCP.h

**C**
WORD **TCPGetArray**(
TCP_SOCKET **hTCP**,
BYTE* **buffer**,
WORD **count**
);

**Returns**
Количество прочитанных байтов из сокета. Если меньше, чем длина, буфер RX FIFO опустел или сокет не проведен .

**Description**
Читает массив байтов данных из TCP сокета получения FIFO. Данные удаляется из FIFO в процессе.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
hTCP The socket из которого данные должны быть считаны .
**buffer** Указатель на массив для хранения данных, которые были прочитаны.

**len** Количество байт для чтения.

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

### 10.18.1.26 TCPGetRemoteInfo Function

**File**
TCP.h

**C**
SOCKET_INFO* **TCPGetRemoteInfo**(
TCP_SOCKET **hTCP**

);

**Returns**
Возвращает

SOCKET_INFO ( см. стр. 550) структуру , связанную с этим гнездом . Эта структура выделяется статически функционировать и действует только до в следующий раз **TCPGetRemoteInfo** () вызывается .

**Description**
Возвращает SOCKET_INFO (см. стр. 550 ) структуру , связанную с этим гнездом . Это содержит NODE_INFO структура с IP и MAC -адрес ( или шлюза MAC ) и удаленного порта .

**Preconditions**
TCP инициализации и сокет подключен .

**Parameters**
**Parameters Description**
hTCP The socket to проверить.

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

### 10.18.1.27 TCPGetRxFIFOFree Function

**File**
TCP.h

**C**
WORD **TCPGetRxFIFOFree**(
TCP_SOCKET **hTCP**
);

**Returns**
Число байтов бесплатно в TCP RX FIFO. Если ноль, никаких дополнительных данных не может быть принят до удаляемым приложений некоторые данные, используя один из TCPget ( см. стр. 537 ) семейных функций .

**Description**
Определяет , сколько байт свободны в RX FIFO.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
hTCP The socket to проверить.

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

### 10.18.1.28 TCPGetRxFIFOFull Macro

**File**
TCP.h

**C**
```
#define TCPGetRxFIFOFull(a) TCPIsGetReady(a)
```

**Description**
Alias to TCPIsGetReady ( see page 541) предусмотрено API полноты

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

### 10.18.1.29 TCPGetTxFIFOFree Macro

**File**
TCP.h

**C**
```
#define TCPGetTxFIFOFree(a) TCPIsPutReady(a)
```

**Description**
Alias to TCPIsPutReady ( see page 541) предусмотрено API полноты

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

### 10.18.1.30 TCPGetTxFIFOFull Function

**File**
TCP.h

**C**
```
WORD TCPGetTxFIFOFull(
TCP_SOCKET hTCP
);
```

**Returns**
Количество байт , ожидающих быть сброшены в TCP TX FIFO.

**Description**
Определяет , сколько байтов на рассмотрении в TCP TX FIFO.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
hTCP The socket to проверить.

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

### 10.18.1.31 TCPIsConnected Function

**File**
TCP.h
**C**
BOOL **TCPIsConnected**(
TCP_SOCKET **hTCP**
);
**Description**
Эта функция определяет, является ли сокет имеет установленную подключение к удаленному узлу. Эта функция вызывается после вызова TCPOpen (см. стр. 542 ), чтобы определить , когда соединение установлено и готово к использованию. Эта функция исторически используется для проверки отключений , но **TCPWasReset** (см. стр. 549 ) в настоящее время является более подходящим решением.

**Remarks**
Имеется разъем называется связным, только если он находится в состоянии TCP_ESTABLISHED . Розетки ( см. стр. 152 ) в процессе Открытие или закрытие вернет FALSE .

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
hTCP The socket to check.

**Return Values**
**Return Values Description**
**TRUE** The socket имеет установленное соединение с удаленным узлом.
**FALSE** The socket не подключены.

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

### 10.18.1.32 TCPIsGetReady Function

**File**
TCP.h

**C**
WORD **TCPIsGetReady**(

TCP_SOCKET **hTCP**
);

**Returns**
Имеющаяся число байтов для чтения из буфера TCP RX.

**Description**
Вызовите эту функцию, чтобы определить, сколько байт можно прочитать из буфера TCP RX. Если эта функция возвращает ноль , Заявка должна вернуться в основной цикл стека , прежде чем продолжить , чтобы ждать больше данных , чтобы прибыть.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
hTCP The socket to check. проверить

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

### 10.18.1.33 **TCPIsPutReady Function**

**File**
TCP.h

**C**
WORD **TCPIsPutReady**(
TCP_SOCKET **hTCP**
);

**Returns**
Имеющаяся число байтов для записи в буфере TCP TX.

**Description**
Вызовите эту функцию, чтобы определить, сколько байт можно записать в буфер TCP TX. Если эта функция возвращает ноль , Заявка должна вернуться в основной цикл стека , прежде чем продолжить , чтобы передать больше данных .

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
hTCP The socket to check.

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

### 10.18.1.34 TCPListen Macro

**File**
TCP.h

**C**
```
#define TCPListen(port) TCPOpen(0, TCP_OPEN_SERVER, port, TCP_PURPOSE_DEFAULT)
```

**Description**
Эта функция является псевдонимом для **TCPOpen** (см. стр. 542 ) для серверных сокетов. Она предназначена для обеспечения обратной совместимости с старые версии стека . Новые приложения должны использовать **TCPOpen** (см. стр. 542 ) API вместо этого.

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

### 10.18.1.35 TCPOpen Function

**File**
TCP.h

**C**
```
TCP_SOCKET TCPOpen(
DWORD dwRemoteHost,
BYTE vRemoteHostType,
WORD wPort,
BYTE vSocketPurpose
);
```

**Description**
Предоставляет унифицированный метод для открытия TCP сокеты. Эта функция может открыть и клиент и сервер сокеты. Для клиента розетки, он может принимать (см. стр. 169 ) имени узла строку запроса в DNS, в IP- адрес в виде строки , в IP- адреса в бинарной форме , или ранее решены структура NODE_INFO содержащий удаленный IP -адрес и соответствующему MAC -адрес.

Когда имя узла или IP- адрес только обеспечивается , модуль TCP будет внутренне выполнения необходимой DNS и / или ARP шаги разрешения , прежде чем сообщить , что сокет TCP подключен ( через вызов TCPISConnected возвращения TRUE) . сервер розетки этот параметр игнорируют назначения и слушать (см. стр. 175) только на указанном порту .

Поле vSocketPurpose позволяет розетки должен быть открыт с изменяющимися параметрами размера буфера и носителей информации памяти .

Это поле соответствует заранее определенных розеток , выделенных в **TCPSocketInitializer** [ ] массива в **TCPIPConfig**.h .

TCPIPConfig.h файл можно редактировать с помощью мастера настройки TCP / IP.

Розетки ( см. стр. 152 ) статически выделяется при загрузке, но можно утверждать, с этой функцией и освобождается с помощью **TCPDisconnect** (см. стр. 533 ) или **TCPClose** (см. стр. 532 ) ( для клиентских сокетов ) . Розетки сервера могут быть освобождены с помощью **TCPClose** (см. стр. 532 ) только ( звонки на **TCPDisconnect** (см. стр. 533 ) вернет серверных сокетов в состоянии прослушивания , позволяет повторное использование) .

**Remarks**
Эта функция заменяет старый **TCPConnect** (см. стр. 532 ) и **TCPListen** (см. стр. 542 ) функции . Если TCP_OPEN_RAM_HOST (см. стр. 530 ) или TCP_OPEN_ROM_HOST (см. стр. 531 ) используются для назначения  Тип ,клиентский модуль DNS также должен быть включен ( STACK_USE_DNS должны быть определены в **TCPIPConfig**.h ) .

**Preconditions**
TCP is initialized.

**Example**
```
// Open a server socket
skt = TCPOpen(NULL, TCP_OPEN_SERVER, HTTP_PORT,
TCP_PURPOSE_HTTP_SERVER);
// Open a client socket to www.microchip.com
// Двойной бросок здесь предотвращает предупреждения компилятора
skt = TCPOpen((DWORD)(PTR_BASE)"www.microchip.com",
TCP_OPEN_ROM_HOST, 80, TCP_PURPOSE_DEFAULT);
// Повторное открытие клиентского сокета , не повторяя DNS или ARP
SOCKET_INFO cache = TCPGetSocketInfo(skt); //Вызов со старым сокетом
skt = TCPOpen((DWORD)(PTR_BASE)&cache.remote, TCP_OPEN_NODE_INFO,
cache.remotePort.Val, TCP_PURPOSE_DEFAULT);
```

**Parameters**
**Parameters Description**
**dwRemoteHost** Только клиентских сокетов. Обеспечить указатель на завершающуюся нулем строку имя удаленного хоста (например: " www.microchip.com " или " 192.168.1.123 " ) , буквальный IP -адрес назначения (например: 0x7B01A8C0 или тип данных IP_ADDR ) , или Указатель на структуру NODE_INFO с удаленным IP -адресом и удаленным узлом или шлюз MAC -адрес, указанный . Если строка условии , обратите внимание, что это должно быть статически размещена в оперативной памяти и никогда не могут быть изменены или освобождается , пока **TCPIsConnected** (см. стр. 540 ) возвращает значение ИСТИНА .

Этот параметр игнорируется для серверных сокетов.

vRemoteHostType Любой из следующих флажков, чтобы определить смысл dwRemoteHost

параметр :

• **TCP_OPEN_SERVER** (см. стр. 531 ) - Откройте сокет сервера и

игнорировать параметр dwRemoteHost .

• **TCP_OPEN_RAM_HOST** (см. стр. 530) - Откройте клиентский сокет и

подключения (см. стр. 171 ) , что на удаленный хост чье имя хранится в виде

нулем строку в массиве памяти . Например: " www.microchip.com " или

" 192.168.0.123 " (BYTE * тип)

• **TCP_OPEN_ROM_HOST** (см. стр. 531 ) - Откройте клиентский сокет и

подключения (см. стр. 171 ) , что на удаленный хост чье имя хранится в виде

нулем строку в буквальном строки или ROM массива. Пример:

" www.microchip.com " или " 192.168.0.123 " (ПЗУ BYTE * тип)

• TCP_OPEN_IP_ADDRESS (см. стр. 530) - Откройте клиентский сокет и

подключения (см. стр. 171 ) , что на удаленный IP -адрес. Пример: 0x7B01A8C0 для

192.168.1.123 (тип DWORD) . Обратите внимание, что порядок байтов Big Endian .

• TCP_OPEN_NODE_INFO (см. стр. 530) - Откройте клиентский сокет и

Connect ( см. стр. 171 ) , что на удаленный IP и MAC- адреса пару хранится

в структуре NODE_INFO . dwRemoteHost должны быть указателем на

Структура NODE_INFO . Эта опция предоставляется для обратной совместимости

с приложениями, построенными на предыдущих версиях стека , что только реализованных

**TCPConnect** (см. стр. 532 ) () функцию . Он также может быть использован , чтобы пропустить

DNS и разрешения ARP шаги при подключении к удаленному узлу , который Вы уже подключены к и кэшируются адреса .

wPort TCP порт для прослушивания (см. стр. 175 ) или подключения (см. стр. 171) , чтобы :

• Клиент розетки - удаленный порт TCP , к которому соединение должно быть

сделал . Локальный порт для клиентских сокетов , будут приниматься автоматически по

Модуль TCP .

• розетки сервера - локальный порт TCP , на котором слушать (см. стр. 175) для

соединения.

vSocketPurpose Любой из TCP_PURPOSE_ * константы определены в TCPIPConfig.h или

TCPIPConfig утилита (см. TCPSocketInitializer [] массив ) .

**Return Values**
**Return Values Description**
INVALID_SOCKET ( see page 529) Нет SOCKET указанного типа не были доступны должен быть открыт.

В противном случаеTCP_SOCKET (см. стр. 553 ) справиться. Сохранить ручку и использовать его при называя все другие интерфейсы TCP .

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

### 10.18.1.36 TCPPeek Function

**File**
TCP.h

**C**
BYTE **TCPPeek**(
TCP_SOCKET **hTCP**,
WORD **wStart**
);

**Description**
Пики при один байт в TCP RX FIFO , не удаляя его из буфера .

**Remarks**
Используйте **TCPPeekArray** (см. стр. 544 ) () для чтения более одного байта . Это будет работать лучше , чем вызов **TCPPeek** ( ) в цикле.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
**hTCP** Сокет до пика от (читай , не снимая из потока ) .
**wStart** нулевой индексируются стартовая позиция в FIFO заглянуть от .

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

### 10.18.1.37 TCPPeekArray Function

**File**
TCP.h

**C**
WORD **TCPPeekArray**(
TCP_SOCKET **hTCP**,
BYTE * **vBuffer**,

WORD **wLen**,
WORD **wStart**
);

**Description**
Читает указанное число байтов данных из TCP RX FIFO , не удаляя их из буфера. Отсутствие контроля TCP действия предпринимаются в результате этой функции (пример: не обновление окна не отправляется на удаленный узел ) .

**Remarks**
None

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
**hTCP** The socket до пика от (читай , не снимая из потока ) .
**vBuffer** Направление писать заглянул байтов данных .
**wLen** Длина байт до пика от RX FIFO и скопировать на vBuffer .

**wStart** нулевой индексируются стартовая позиция в FIFO , чтобы начать выглядывая .

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

### 10.18.1.38 TCPPut Function

**File**
TCP.h

**C**
BOOL **TCPPut**(
TCP_SOCKET **hTCP**,
BYTE **byte**
);

**Description**
Записывает один байт в сокет TCP .

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
**hTCP** The socket какие данные должны быть записаны .
**byte** Байт писать.

**Return Values**
**Return Values Description**

**TRUE** байт записан в буфер передачи .

**FALSE** передачи был полон, или сокет не подключен.

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

### 10.18.1.39 TCPPutArray Function

**File**
TCP.h

**C**
WORD **TCPPutArray**(
TCP_SOCKET **hTCP**,
BYTE* **Data**,
WORD **Len**
);

**Returns**
Количество байтов, записанных в сокет. Если меньше, чем длина, буфер стал полностью или сокет не проведен .

**Description**
Записывает массив из оперативной памяти на сокет TCP .

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
**hTCP** The socket какие данные должны быть записаны .
**data** Указатель  в массив для записи.

**len** Количество записываемых байтов .

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

### 10.18.1.40 TCPPutROMArray Function

**File**
TCP.h

**C**
WORD **TCPPutROMArray**(
TCP_SOCKET **hTCP**,

ROM BYTE* **Data**,
WORD **Len**
);

**Returns**
Количество байтов, записанных в сокет. Если меньше, чем длина, буфер стал полностью или сокет не проведен .

**Description**
Записывает массив из ПЗУ в сокет TCP .

**Remarks**
Эта функция является псевдонимом для **TCPPutArray** (см. стр. 545 ) на не- PIC18 платформ.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
**hTCP** The socket какие данные должны быть записаны .
**data** Указатель в массив для записи.

**len** Количество записываемых байтов .

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

### 10.18.1.41 TCPPutROMString Function

**File**
TCP.h

**C**
ROM BYTE* **TCPPutROMString**( TCP_SOCKET **hTCP**, ROM BYTE* **Data**);

**Returns**
Указатель на байт, следующий последнего байта записаны в сокет . Если этот указатель не разыменовывает в байт NUL , буфера стал полностью или сокет не подключен .

**Description**
Записывает строку с завершающим нулем из ПЗУ в сокет TCP . Нуль- терминатор не копируется в сокет.

**Remarks**
Возвращаемое значение этой функции отличается от **TCPPutArray** (см. стр. 545 ) . Чтобы написать длинных строк в одном государстве , инициализировать указатель данных * на первый байт , а затем вызвать эту функцию неоднократно ( нарушение в основной цикл стека после каждого вызова ) пока возвращаемое значение не разыменовывает в байт NUL . Сохранить возвращаемого значения , что и новой отправной * указатель данных иным образом.

Эта функция является псевдонимом для **TCPPutString** (см. стр. 547 ) на не- PIC18 платформ.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
**hTCP** The socket какие данные должны быть записаны .
**data** Указатель для строки, которая будет написана .

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

### 10.18.1.42 TCPPutString Function

**File**
TCP.h

**C**
BYTE* **TCPPutString**(
TCP_SOCKET **hTCP**,
BYTE* **Data**
);

**Returns**
Указатель на байт, следующий последнего байта записаны в сокет . Если этот указатель не разыменовывает в байт NUL , буфера стал полностью или сокет не подключен .

**Description**
Записывает строку с завершающим нулем из оперативной памяти на сокет TCP . Нуль-терминатор не копируется в сокет.

**Remarks**
Возвращаемое значение этой функции отличается от **TCPPutArray** (см. стр. 545 ) . Чтобы написать длинных строк в одном государстве , инициализировать указатель данных * на первый байт , а затем вызвать эту функцию неоднократно ( нарушение в основной цикл стека после каждого вызова ) пока возвращаемое значение не разыменовывает в байт NUL . Сохранить возвращаемого значения , что и новой отправной * указатель данных иным образом.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
**hTCP** The socket какие данные должны быть записаны .
**data** Указатель данных для строки, которая будет написана .

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

### 10.18.1.43 TCPRAMCopy Function

**File**
TCP.c

**C**
```
static void TCPRAMCopy(
PTR_BASE wDest,
BYTE vDestType,
PTR_BASE wSource,
BYTE vSourceType,
WORD wLength
);
```

**Returns**
None

**Description**
Эти данные функция копирует между средами памяти (PIC RAM, SPI RAM, and Ethernet buffer RAM).

**Remarks**
Копирование в область назначения , которая перекрывает с адресом источника поддерживается, только если начальный адрес назначения находится на ниже адрес памяти ( ближе к 0x0000 ), чем указателя источника. Тем не менее, если они делают перекрываются там должно быть не менее 4 байта не- перекрытия , чтобы обеспечить правильные результаты в связи с требованиями аппаратного DMA .

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
**ptrDest** Address ( see page 147) для записи
**vDestType** назначения meidum (TCP_PIC_RAM, TCP_ETH_RAM, TCP_SPI_RAM)
**ptrSource** Address ( see page 147) to copy from
**vSourceType** Source medium (TCP_PIC_RAM, TCP_ETH_RAM, or TCP_SPI_RAM)
**wLength** Количество байт для копирования

**Section**
Function Prototypes

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

### 10.18.1.44 TCPRAMCopyROM Function

**File**
TCP.c

**C**
**static void TCPRAMCopyROM**(
PTR_BASE **wDest**,
BYTE **wDestType**,
ROM BYTE* **wSource**,
WORD **wLength**
);

**Returns**
None

**Description**
Эти данные функция копирует между средами памяти ( ПОС RAM, SPI памяти и буферного ОЗУ Ethernet ). Эта функция должна быть используется при копировании с диска.

**Remarks**
Копирование в область назначения , которая перекрывает с адресом источника поддерживается, только если начальный адрес назначения находится на ниже адрес памяти ( ближе к 0x0000 ), чем указателя источника.

Эта функция является псевдонимом для **TCPRAMCopy** (см. стр. 547 ) на не- PIC18 платформ.

.

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
wDest Address ( see page 147) для записи
wDestType назначения meidum (TCP_PIC_RAM, TCP_ETH_RAM, TCP_SPI_RAM)
wSource Address ( see page 147) скопировать с
wLength Количество байт для копирования

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

### 10.18.1.45 TCPWasReset Function

**File**
TCP.h

**C**
BOOL **TCPWasReset**(TCP_SOCKET **hTCP**);

**Description**
Эта функция является семафор самостоятельно очистка указывающее, будет ли был отключен с момента предыдущего вызова сокет .

Эта функция работает для всех возможных отключений : вызов **TCPDisconnect** (см. стр. 533 ) , в FIN от удаленного узла , или тайм-аут подтверждения вызвано потерей сетевого соединения . Она также возвращает TRUE, после первого вызова **TCPInit** (см. стр. 554) . Приложения должны использовать эту функцию , чтобы сбросить свои автоматы .

Эта функция была добавлена в связи с возможностью возникновения ошибки при опираясь на **TCPIsConnected** (см. стр. 540 ) returing

ЛОЖЬ для проверки состояния , требующего сброса машины состояний . Если сокет закрыт (из-за FIN ACK ), а затем немедленно возобновлено ( в связи с приходом нового SYN ) в том же цикле стека , вызовы **TCPIsConnected** (см.страница 540 ) от применения никогда не вернется ЛОЖЬ хотя сокет был отключен. Это может привести к ошибкам для протоколов, таких как HTTP , в котором клиент сразу же открыть новое соединение при закрытии предварительного одной . Опираясь на эта функция вместо позволяет приложениям ловушка эти условия и для нового соединения правильно сбросить свое внутреннее состояние .

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
hTCP Сокет проверить .

**Return Values**
**Return Values Description**
**TRUE** сокет разрывает соединение с другим абонентом.

**FALSE** сокет не был отключен с момента предыдущего вызова.

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

### 10.18.2 TCP Stack Members

**Enumerations**
**Name Description**
TCP_STATE ( see page553)TCP States as defined by RFC 793

**Functions**
**Name Description**
TCPInit ( see page 554) Initializes the TCP module.
TCPProcess ( see page555)Ручки входящие сегменты TCP .
TCPTick ( see page 555) Выполняет периодические задачи TCP .

TCPSSLDecryptMAC ( seepage 556) Расшифровка и данные ПДК прибываете через SSL.

TCPStartSSLClientEx ( seepage 556) Начинает клиента SSL сеанс .

**Module**
TCP ( see page 526)

**Structures**
**Name Description**

SOCKET_INFO (см. стр.550)Информация о сокете

TCB ( см. стр. 551 ) остаток от деления данных TCP блок управления . Остальная часть УТС хранится в Ethernet буферного ОЗУ или в другом месте , как это определено vMemoryMedium . Текущий размер 41 ( PIC18 ) , 42 ( PIC24/dsPIC ) , или 48 байт ( PIC32 )
TCB_STUB (см. стр. 552 ) TCP Блок управления ( УТС) для хранения данных заглушек . Заглушки хранятся в местном ПОС RAM для скорости. Текущий размер составляет 34 байт ( PIC18 ), 36 байт ( PIC24/dsPIC ) или 56 ( PIC32 )

**Types**
**Name Description**
TCP_SOCKET ( see page
553)
A TCP_SOCKET хранится в виде одного байта

**Description**
Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны, как правило, не называют эти функции или модифицировать эти переменные .

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

### 10.18.2.1 SOCKET_INFO Structure

**File**
TCP.h

**C**
```
typedef struct {
NODE_INFO remote;
WORD_VAL remotePort;
} SOCKET_INFO;
```

**Members**
**Members Description**

NODE_INFO remote; Структура NODE_INFO для удаленного узла
WORD_VAL remotePort; Номер порта связан с удаленным узлом

**Description**
Информация о сокете

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

### 10.18.2.2 TCB Structure

**File**
TCP.h

**C**
```
typedef struct {
DWORD retryInterval;
DWORD MySEQ;
DWORD RemoteSEQ;
PTR_BASE txUnackedTail;
WORD_VAL remotePort;
WORD_VAL localPort;
WORD remoteWindow;
WORD wFutureDataSize;
union {
NODE_INFO niRemoteMACIP;
DWORD dwRemoteHost;
} remote;
SHORT sHoleSize;
struct {
unsigned char bFINSent : 1;
unsigned char bSYNSent : 1;
unsigned char bRemoteHostIsROM : 1;
unsigned char bRXNoneACKed1 : 1;
unsigned char bRXNoneACKed2 : 1;
unsigned char filler : 3;
} flags;
WORD wRemoteMSS;
WORD_VAL localSSLPort;
BYTE retryCount;
BYTE vSocketPurpose;
} TCB;
```

**Members**
**Members Description**
DWORD RetryInterval ; Как долго ждать перед передачей повторной попыткой

DWORD MySEQ ; Часовой порядковый номер

DWORD RemoteSEQ ; Удаленный порядковый номер

PTR_BASE txUnackedTail ; Техас хвост указатель для данных, которые еще не acked

WORD_VAL RemotePort ; Число Удаленный порт

WORD_VAL локальный_порт ; Локальный номер порта

WORD remoteWindow ; Дистанционное размер окна

WORD wFutureDataSize ; Сколько из- того, было получено данных

NODE_INFO niRemoteMACIP ; 10 байт для MAC и IP адрес

DWORD dwRemoteHost ; ОЗУ или указатель ROM к имени хоста строки (например: " www.microchip.com " )

КОРОТКИЕ sHoleSize ; Размер отверстия , или -1 ни для кого существует. ( 0 означает отверстие только что был заполнен )

unsigned char bFINSent : 1 ; FIN был отправлен

unsigned char bSYNSent : 1 ; SYN был отправлен

unsigned char bRemoteHostIsROM : 1 ; Удаленный хост хранится в ПЗУ

unsigned char bRXNoneACKed1 : 1 ; Дубликат ACK , скорее всего получил

unsigned char bRXNoneACKed2 : 1 ; Второй дубликат ACK , скорее всего получил

unsigned char наполнитель: 3 ; использования в будущем

WORD wRemoteMSS ; Максимальная вариант размер сегмента advirtised удаленным узлом во время начальной рукопожатие

WORD_VAL localSSLPort ; Локальный номер SSL -порт ( для прослушивания розетки )

BYTE retryCount ; Счетчик попыток передачи

BYTE vSocketPurpose ; Цель розетки (как определено в TCPIPConfig.h )

**Description**
Остаток данных TCP блок управления . Остальная часть TCB хранится в буферной памяти Ethernet или в другом месте , как это определено vMemoryMedium . Текущий размер 41 ( PIC18 ), 42 ( PIC24/dsPIC ) , или 48 байт ( PIC32 )

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

### 10.18.2.3 TCB_STUB Structure

**File**

TCP.h

**C**

```
typedef struct {
PTR_BASE bufferTxStart;
PTR_BASE bufferRxStart;
PTR_BASE bufferEnd;
PTR_BASE txHead;
PTR_BASE txTail;
PTR_BASE rxHead;
PTR_BASE rxTail;
DWORD eventTime;
WORD eventTime2;
union {
WORD delayedACKTime;
WORD closeWaitTime;
} OverlappedTimers;
TCP_STATE smState;
struct {
unsigned char vUnackedKeepalives : 3;
unsigned char bServer : 1;
unsigned char bTimerEnabled : 1;
unsigned char bTimer2Enabled : 1;
unsigned char bDelayedACKTimerEnabled : 1;
unsigned char bOneSegmentReceived : 1;
unsigned char bHalfFullFlush : 1;
unsigned char bTXASAP : 1;
unsigned char bTXASAPWithoutTimerReset : 1;
unsigned char bTXFIN : 1;
unsigned char bSocketReset : 1;
unsigned char bSSLHandshaking : 1;
unsigned char filler : 2;
} Flags;
WORD_VAL remoteHash;
PTR_BASE sslTxHead;
PTR_BASE sslRxHead;
BYTE sslStubID;
BYTE sslReqMessage;
BYTE vMemoryMedium;
} TCB_STUB;
```

**Members**

**Members Description**

PTR_BASE bufferTxStart ; Первый байт TX буфера

PTR_BASE bufferRxStart ; Первый байт RX буфера. Техас буфер заканчивается 1 байт до

PTR_BASE bufferEnd ; Последний байт RX буфера

PTR_BASE txHead ; Глава указатель для TX

PTR_BASE txTail ; Хвост указатель для TX

PTR_BASE rxHead ; Глава указатель для RX

PTR_BASE rxTail ; Хвост указатель для RX

DWORD EventTime ; Повторных передач пакетов , изменения состояния

СЛОВО eventTime2 ; Обновления окон , автоматическая коробка передач

10.18 TCP Microchip TCP / IP стека Помощь TCP стека Пользователи

552

СЛОВО delayedACKTime ; Задержка таймер подтверждения приема

СЛОВО closeWaitTime ; Таймер TCP_CLOSE_WAIT тайм-аут

TCP_STATE smState ; Государство этого сокета

unsigned char vUnackedKeepalives : 3 ; Граф , как многие поддержку активности были отправлены без ответа

unsigned char BSERVER : 1 ; Разъем должен вернуться к прослушиванию состояние, когда закрыты

unsigned char bTimerEnabled : 1 ; Таймер включен

unsigned char bTimer2Enabled : 1 ; Во-вторых таймер включен

unsigned char bDelayedACKTimerEnabled : 1 ;

DelayedACK таймер включен

unsigned char bOneSegmentReceived : 1 ; Сегмент был получен

unsigned char bHalfFullFlush : 1 ; Флеш для наполовину полный

unsigned char bTXASAP : 1 ; Передача как можно скорее (для Flush )

unsigned char bTXASAPWithoutTimerReset : 1 ;

Передача как можно скорее (для Flush ) , но не обнуляют повторной передачи таймеры

unsigned char bTXFIN : 1 ; FIN должна быть передана

unsigned char bSocketReset : 1 ; Разъем были восстановлены (само- клиринговую семафор )

unsigned char bSSLHandshaking : 1 ; Разъем находится в SSL рукопожатия

unsigned char наполнитель: 2 ; Будущее расширение

WORD_VAL remoteHash ; Состоит из RemoteIP , RemotePort , локальный_порт для подключенных сокетов. Это локальный_порт количество только для прослушивания серверные сокеты .

PTR_BASE sslTxHead ; Положение данных, написанных на следующей записи приложений SSL Также служит

кэш localSSLPort когда smState = TCP_LISTENING

PTR_BASE sslRxHead ; Положение входящих данных еще не обработанных SSL

BYTE sslStubID ; Какие sslStub (см. стр. 518 ) связано с связи с этим

BYTE sslReqMessage ; В настоящее время предложено сообщение SSL

BYTE vMemoryMedium ; Какие памяти среднийУТС на самом деле хранится

**описание**

TCP Блок управления ( УТС) для хранения данных заглушек . Заглушки хранятся в местном ПОС памяти для скорости. Текущий размер составляет 34 байт ( PIC18 ) ,

36 байт ( PIC24/dsPIC ), или 56 ( PIC32 )

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

### 10.18.2.4 TCP_SOCKET Type

**File**
TCP.h

**C**
```
typedef BYTE TCP_SOCKET;
```

**Description**
TCP_SOCKET хранится в виде одного байта

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

### 10.18.2.5 TCP_STATE Enumeration

**File**
TCP.h

**C**
```
typedef enum {
```

**TCP_GET_DNS_MODULE**,
**TCP_DNS_RESOLVE**,
**TCP_GATEWAY_SEND_ARP**,
**TCP_GATEWAY_GET_ARP**,
**TCP_LISTEN**,
**TCP_SYN_SENT**,
**TCP_SYN_RECEIVED**,
**TCP_ESTABLISHED**,
**TCP_FIN_WAIT_1**,
**TCP_FIN_WAIT_2**,
**TCP_CLOSING**,
**TCP_CLOSE_WAIT**,
**TCP_LAST_ACK**,
**TCP_CLOSED**,
**TCP_CLOSED_BUT_RESERVED**
} TCP_STATE;
**Members**
**Members Description**

TCP_GET_DNS_MODULE Специальное государство для сокетов в режиме клиент TCP

TCP_DNS_RESOLVE Специальное государство для сокетов в режиме клиент TCP

TCP_GATEWAY_SEND_ARP Специальное государство для сокетов в режиме клиент TCP

TCP_GATEWAY_GET_ARP Специальное государство для сокетов в режиме клиент TCP

TCP_LISTEN гнездо слушает соединения

TCP_SYN_SENTSYN был отправлен , ожидая SYN + ACK

TCP_SYN_RECEIVEDSYN был получен , ожидая ACK

TCP_ESTABLISHED Разъем подключен и устанавливается соединение

TCP_FIN_WAIT_1 FIN состояние ожидания 1

TCP_FIN_WAIT_2 FIN состояние ожидания 2

TCP_CLOSING Разъем закрывается TCP_TIME_WAIT , государство не реализован

TCP_CLOSE_WAIT Ожидание закрыть сокет

TCP_LAST_ACK Окончательный ACK был отправлен

TCP_CLOSED гнездо находится в режиме ожидания и нераспределенные

Для сокетов в режиме клиент TCP TCP_CLOSED_BUT_RESERVED особое состояние . Разъем находится в режиме ожидания , но все же выделяется

до окончания применение ручки.

**описание**

TCP Штаты , как это определено RFC 793

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

### 10.18.2.6 TCPInit Function

**File**
TCP.h
**C**
```
void TCPInit();
```
**Returns**
None
**Description**
Инициализирует модуль TCP . Эта функция настраивает буферы TCP в памяти и инициализирует каждый сокет в состояние ЗАКРЫТО. Если было выделено недостаточно памяти для гнезда TCP , функция будет висеть здесь, чтобы быть захвачены отладчика.

**Remarks**
Эта функция вызывается только один при жизни приложения.

**Preconditions**
None

**Section**
Function Declarations

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

### 10.18.2.7 TCPProcess Function

**File**
TCP.h
**C**
```
BOOL TCPProcess(
NODE_INFO* remote,
IP_ADDR* localIP,
WORD len
);
```
**Description**
Эта функция обрабатывает входящие сегменты TCP . Когда приходит сегмент , это по сравнению с открытых сокетов с использованием хэш удаленный порт и IP . На матче ,данные передаются **HandleTCPSeg** (см. стр. 560 ) для дальнейшей обработки.

**Preconditions**
TCP инициализации и сегмент TCP готов в буфере MAC .

**Parameters**
**Parameters Description**
Структура дистанционного NODE_INFO

**localIP** IP адрес этого стека (для проверки заголовка )

**len** Общая длина ожидания сегмента TCP

**Return Values**
**Return Values Description**
**TRUE** сегмент был надлежащим образом.
**FALSE** в противном случае

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

### 10.18.2.8 TCPTick Function

**File**
TCP.h

**C**
```
void TCPTick();
```

**Returns**
None

**Description**
Эта функция выполняет все необходимые периодические задачи TCP . Государственная машина каждого сокета проверяется , и любое , прошедшее тайм-аут периоды обрабатываются .

**Preconditions**
TCP is initialized.

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

### 10.18.2.9 TCPSSLDecryptMAC Function

**File**
TCP.h
**C**
```
void TCPSSLDecryptMAC(
TCP_SOCKET hTCP,
ARCFOUR_CTX* ctx,
WORD len
);
```
**Returns**

None
**Description**
Эта функция дешифрует данные в буфере TCP и вычисляет MAC над данными . Все данные остаются в том же месте в буфере TCP. Она призвана помочь обрабатывать входящие записи SSL .

**Remarks**
Эта функция никогда не должны называться одним из приложений. Он используется только самим модулем SSL .

**Preconditions**
TCP инициализации HTCP подключен , и SBOX CTX автора загружается .

**Parameters**
**Parameters Description**
hTCP TCP Подключение HTCP TCP для расшифровки в

ctx ARCFOUR контекст шифрования использовать

len Количество байт для Crypt

inPlace **TRUE** написать на место, **ЛОЖЬ** написать в конце в настоящее время видимых данных .

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

### 10.18.2.10 TCPStartSSLClientEx Function

**File**
TCP.h

**C**
```
BOOL TCPStartSSLClientEx(
TCP_SOCKET hTCP,
BYTE* host,
void * buffer,
BYTE suppDataType
);
```

**Description**
Эта функция возрастает текущего подключения к защищенному соединению SSL по инициирования клиентского SSL рукопожатия .

**Remarks**
Параметр хозяин в настоящее время игнорируются и не проверяется .

**Preconditions**

TCP инициализации и HTCP уже подключен.

**Parameters**
**Parameters Description**
**hTCP** TCP connection to secure
**host Expected** имя хоста на сертификат (в настоящее время игнорируются)
**buffer** для дополнительного возвращения данных
**suppDataType** Тип дополнительных данных для копирования

**Return Values**
**Return Values Description**
ИСТИНА соединениеSSL был инициирован

ЛОЖЬ Недостаточные ресурсы SSL ( заглушки ) были доступны

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

### 10.18.3 TCP Internal Members

**Functions**
**Name Description**
CloseSocket (см. стр.559)Закрывает сокет TCP .

FindMatchingSocket (см.страница 560 )Находит подходящий разъем для сегмента TCP .

HandleTCPSeg (см. стр.560)Процессы на входящий сегмент TCP .

SendTCP (см. стр. 562 ) Передает сегмент TPC .

SwapTCPHeader (см.страница 563)Меняет местами порядка байтов из заголовка TCP .

SyncTCB (см. стр. 564 ) кэш Очищает MyTCB и загружает указанный УТС. Ничего не делает на кэш ударил .

**Macros**
**Name Description**
ACK (см. стр. 559) Признать Отметить как определено в RFC

FIN (см. стр. 559 ) FIN Отметить как определено в RFC

LOCAL_PORT_END_NUMBER (см.страница 561)Конец порт для клиентских сокетов

LOCAL_PORT_START_NUMBER (см.страница 561)Начиная порт для клиентских сокетов

PSH ( см. стр. 562) принудительные Отметить как определено в RFC

RST (см. стр. 562 ) Флаг сброса , как определено в RFC

SENDTCP_KEEP_ALIVE (см. стр.563)Вместо передачи обычных данных ,мусора октет передается в соответствии с RFC 1122 раздел 4.2.3.6

SENDTCP_RESET_TIMERS (см. стр.563) Указывает, что этот пакет является повторной передачей (без сброса ) или новый пакет ( сброс обязательно)

SYN ( см. стр. 564) SYN Отметить как определено в RFC

SyncTCBStub (см. стр. 564 ) Приливы MyTCBStub (см. стр. 562 ) кэш и загружает до указано TCB_STUB (см. стр. 552 ) . Ничего не делает на кэш ударил .

TCP_AUTO_TRANSMIT_TIMEOUT_VAL (См. стр. 565 ) Таймаут перед автоматической передачи непокрасневшим данные

TCP_WINDOW_UPDATE_TIMEOUT_VAL (См. стр. 565 ) Таймаут перед автоматической передачи обновление окна из-за

TCPget (см. стр. 537 ) () или TCPGetArray (см. стр. 538 ) ( ) вызов функции

TCP_CLOSE_WAIT_TIMEOUT (см. страница 565 ) Тайм-аут для государства CLOSE_WAIT

TCP_DELAYED_ACK_TIMEOUT (см.страница 565 ) Тайм-аут для алгоритма замедленного подтверждения

TCP_FIN_WAIT_2_TIMEOUT (см.страница 566 )Тайм-аут для FIN ждать 2 государства

TCP_KEEP_ALIVE_TIMEOUT (см.страница 567)Тайм-аут для Keep-Alive сообщений, когда трафик не передается

TCP_MAX_RETRIES (см. стр. 567 ) Максимальное количество попыток повторной передачи

TCP_MAX_SEG_SIZE_RX (см. стр.567)TCP максимальный размер сегмента для RX. Это значение advirtised во время установления соединения и удаленного узла следует повиноваться ему . Это должен быть установлен на 536 , чтобы избежать фрагментации IP слой от причинения потери пакетов . Тем не менее, повышение его стоимости может повысить выступление на (небольшой) риск введения несовместимость с некоторые специальные удаленные узлы (например: те, подключенные через медленного набора модем ) .

TCP_MAX_SEG_SIZE_TX (см. стр.568) TCP максимальный размер сегмента для TX. Максимальная сегмент Техас Размер фактически govered по MSS опции удаленного узла advirtised во время установления соединения. Однако, если удаленный узел определяет unhandlably большой MSS (например: > Ethernet MTU) , это определить устанавливает жесткое ограничение , так что мы не вызывают TX переполнения буфера . Если удаленный узел не advirtise с MSS вариант, все сегменты TX фиксируются на 536 байт максимум.

TCP_MAX_SYN_RETRIES (см. стр.568)Меньше, чем всех других попыток сократить SYN Flood DoS продолжительность

TCP_MAX_UNACKED_KEEP_ALIVES (см. стр. 568 )Максимальное количество Keep-Alive сообщений, которые могут быть отправлены без ответа , прежде чем автоматически закрытия подключение

TCP_OPTIMIZE_FOR_SIZE (см. стр. 568) Для маленького размера и лучшей пропускной способности

TCP_OPTIMIZE_FOR_SIZE всегда должен быть включен на Продукты PIC24/dsPIC . На PIC32 продукции существует очень мало разница и depnds на уровне оптимизации компиляторов

TCP_OPTIONS_END_OF_LIST (см.страница 569 ) Конец Список TCP Option Отметить

TCP_OPTIONS_MAX_SEG_SIZE (см.страница 569 )Максимальный размер сегмента флаг TCP

TCP_OPTIONS_NO_OP (см. стр. 569 ) Нет Op TCP Опция TCP_SOCKET_COUNT (см. стр. 570 ) Определяет количество определенных розеток TCP

TCP_START_TIMEOUT_VAL (см.страница 570 )Тайм-аут для повторной передачи Unacked данные

TCP_SYN_QUEUE_MAX_ENTRIES (см. стр. 571 )Количество TCP RX SYN -пакетов , чтобы сохранить , если они не могут быть обслуживаются сразу

TCP_SYN_QUEUE_TIMEOUT (см.страница 571 ) Тайм-аут для того, когда записи очереди SYN удаляются, если непригодным УРГ ( см. стр. 571 ) флаг срочности , как определено в RFC

**Module**
TCP ( see page 526)

**Structures**
**Name Description**
TCP_HEADER (см. стр.566) Структура TCP данные заголовка

TCP_OPTIONS (см. стр. 569) Структура данных параметры TCP

TCP_SYN_QUEUE (см. страница 570 ) Структура, которая содержит все важные элементы объекта Входящие SYN пакета в для того, чтобы установить связь в будущем времени, если все гнезда на ожидающий порт уже подключены к кому-то

**Variables**
**Name Description**
hCurrentTCP (см. стр.561)Текущий гнездо TCP

MyTCB (см. стр. 561 ) В настоящее время загружены УТС

MyTCBStub (см. стр. 562 ) Псевдоним для текущего заглушки TCP .

SYNQueue ( см. стр. 564 ) массива сохраненных входящих SYN запросов , которые должны обслуживаться позже

TCBStubs ( см. стр. 564 ) Это переменные TCBStubs .

**описание**

Следующие функции и переменные обозначены как внутренний модулю TCP.

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

### 10.18.3.1 ACK Macro

**File**
TCP.c
**C**
```
#define ACK (0x10) / /Флаг подтверждения , как определено в RFC
```

**Description**
Признать Отметить как определено в RFC

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

### 10.18.3.2 CloseSocket Function

**File**
TCP.c

**C**
```
static void CloseSocket();
```

**Returns**
None

**Description**
Эта функция закрывает сокет TCP . Вся информация состояние сокета сбрасывается , и любые буферные байты отбрасываются. Сокет

больше не доступны в приложении после этого момента .

**Description**
TCPStub соответствующее гнездо , чтобы быть закрыты синхронизирован .

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

#### 10.18.3.3 FIN Macro

**File**
TCP.c
**C**
```
#define FIN (0x01) // / FIN Отметить как определено в RFC
```

**Description**
FIN Флаг , как определено в RFC

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

#### 10.18.3.4 FindMatchingSocket Function

**File**
TCP.c
**C**
```
static BOOL FindMatchingSocket(
TCP_HEADER* h,
NODE_INFO* remote
);
```
**Description**
Эта функция просматривает розеток и попыток соответствовать один с данного заголовка TCP и структуры NODE_INFO .

Если сокет будет найден, его индекс сохраняется в hCurrentTCP (см. стр. 561 ) и связанный с ним MyTCBStub (см. стр. 562 )и MyTCB загружаются . В противном случае , INVALID_SOCKET (см. стр. 529 ) помещают в hCurrentTCP (см. стр. 561 ) .

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
h TCP чтобы быть сопоставлены с дистанционногоудаленный узел , пославший этот заголовок

**Return Values**
**Return Values Description**
ИСТИНА матч был найден и загружен в hCurrentTCP (см. стр. 561 )

ЛОЖЬ Нет подходит гнездо было найдено и hCurrentTCP (см. стр. 561 ) является

INVALID_SOCKET (см. стр. 529 )

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

### 10.18.3.5 HandleTCPSeg Function

**File**
TCP.c

**C**
```
static void HandleTCPSeg(
TCP_HEADER* h,
WORD len
);
```

**Returns**
None

**Description**
После того, как входящий сегмент был согласован с сокетом , эта функция выполняет необходимую обработку с данными.

В зависимости от сегмента и государства, это может включать копирование данных в буфер TCP , повторно монтажных вне порядка

Пакеты , продолжая инициализацию или закрытия рукопожатие, или вообще закрывает сокет .

**Preconditions**
TCP инициализации и текущий заглушки TCP уже синхронизированы .

**Parameters**
**Parameters Description**
**h** TCP для этого пакета

**len** Общая длина буфера этого сегмента

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

### 10.18.3.6 hCurrentTCP Variable

**File**
TCP.c

**C**
```
TCP_SOCKET hCurrentTCP = INVALID_SOCKET;
```

**Description**
Текущий TCP socket

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

#### 10.18.3.7 LOCAL_PORT_END_NUMBER Macro

**File**
TCP.c

**C**
```
#define LOCAL_PORT_END_NUMBER (5000u)
```

**Description**
Конец порт для клиентских сокетов

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

#### 10.18.3.8 LOCAL_PORT_START_NUMBER Macro

**File**
TCP.c

**C**
```
#define LOCAL_PORT_START_NUMBER (1024u)
```

**Description**
Начиная порт для клиентских сокетов

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

#### 10.18.3.9 MyTCB Variable

**File**
TCP.c
**C**
TCB **MyTCB**;
**Description**
В настоящее время загружены TCB

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

#### 10.18.3.10 MyTCBStub Variable

**File**

TCP.c

**C**
TCB_STUB **MyTCBStub**;

**Description**
Псевдоним для текущего заглушки TCP .

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

10.18.3.11 **PSH Macro**
**File**
TCP.c
**C**
```
#define PSH (0x08) // При нажатии кнопки Отметить как определено в RFC
```

**Description**
Нажмите Отметить как определено в RFC

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

10.18.3.12 **RST Macro**
**File**
TCP.c
**C**
```
#define RST (0x04) // Флаг сброса , как определено в RFC
```

**Description**
Флаг сброса , как определено в RFC

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

10.18.3.13 **SendTCP Function**

**File**
TCP.c

**C**
```
static void SendTCP(
BYTE vTCPFlags,
BYTE vSendFlags
);
```

**Returns**
None

**Description**
Эта функция собирает и передает сегмент TCP , включая любые необходимые данные . Он также поддерживает повторные передачи , держать - дееспособности и другие типы пакетов .

**Preconditions**
TCP is initialized.

**Parameters**
**Parameters Description**
**vTCPFlags** Дополнительные флаги TCP включить

**vSendFlags** любые комбинации SENDTCP_ * констант для изменения поведения передачи или содержание.

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

### 10.18.3.14 SENDTCP_KEEP_ALIVE Macro

**File**
TCP.c

**C**
```
#define SENDTCP_KEEP_ALIVE 0x02
```

**Description**
Вместо передачи обычных данных ,мусора октет передается в соответствии с RFC 1122 раздел 4.2.3.6

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

### 10.18.3.15 SENDTCP_RESET_TIMERS Macro
**File**
TCP.c
**C**
```
#define SENDTCP_RESET_TIMERS 0x01
```
**Description**
Указывает, что этот пакет является повторной передачей (без сброса ) или новый пакет ( сброс обязательно)

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

#### 10.18.3.16 SwapTCPHeader Function

**File**
TCP.c

**C**
```
static void SwapTCPHeader(
TCP_HEADER* header
);
```

**Returns**
None

**Description**
Эта функция меняет местами порядка байтов заданного заголовка TCP для сравнения.

**Preconditions**
None

**Parameters**
**Parameters Description**
Заголовок  TCP , который должен быть заменен

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

#### 10.18.3.17 SYN Macro
**File**
TCP.c
**C**
```
#define SYN (0x02) // SYN флага , как определено в RFC
```

**Description**
SYN флага , как определено в RFC

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

#### 10.18.3.18 SyncTCB Function
**File**
TCP.c
**C**
```
static void SyncTCB();
```
**Description**
Приливы MyTCB кэш и загружает указанный УТС. Ничего не делает на кэш хит .

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

#### 10.18.3.19 SyncTCBStub Macro

**File**
TCP.c

**C**
```
#define SyncTCBStub(a) hCurrentTCP = (a)
```

**Description**
Приливы MyTCBStub ( см. стр. 562 ) кэш и загружает указанный TCB_STUB (см. стр. 552 ) . Ничего не делает на кэшу .

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

### 10.18.3.20 SYNQueue Variable

**File**
TCP.c

**C**
```
TCP_SYN_QUEUE SYNQueue[TCP_SYN_QUEUE_MAX_ENTRIES];
```

**Description**
Массив сохраненных входящих SYN запросов , которые должны обслуживаться позже

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

### 10.18.3.21 TCBStubs Variable

**File**
TCP.c

**C**
```
TCB_STUB TCBStubs[TCP_SOCKET_COUNT];
```

**Description**
Это переменные TCBStubs .

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

### 10.18.3.22 TCP_AUTO_TRANSMIT_TIMEOUT_VAL Macro

**File**
TCP.c

**C**
**#define TCP_AUTO_TRANSMIT_TIMEOUT_VAL** (TICK_SECOND/25ull) // Время ожидания перед  автоматической передачи непокрасневшим данные

**Description**
Таймаут перед автоматической передачи непокрасневшим данные

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

### 10.18.3.23 TCP_WINDOW_UPDATE_TIMEOUT_VAL Macro

**File**
TCP.c
**C**
**#define TCP_WINDOW_UPDATE_TIMEOUT_VAL** (TICK_SECOND/5ull) // Время ожидания перед автоматическим

передачи обновление окна из-за TCPget ( ) или TCPGetArray вызов функции ( )

**Description**
Таймаут перед автоматической передачи обновление окна из-за TCPget (см. стр. 537 ) ( ) или TCPGetArray (см.страница 538 ) вызов функции ( )

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

### 10.18.3.24 TCP_CLOSE_WAIT_TIMEOUT Macro

**File**
TCP.c

**C**
**#define TCP_CLOSE_WAIT_TIMEOUT** ((DWORD)TICK_SECOND/5) // Тайм-аут для государства CLOSE_WAIT

**Description**
Тайм-аут для государства CLOSE_WAIT

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

### 10.18.3.25 TCP_DELAYED_ACK_TIMEOUT Macro

**File**
TCP.c

**C**
```
#define TCP_DELAYED_ACK_TIMEOUT ((DWORD)TICK_SECOND/10) // Время ожидания  для Алгоритм замедленного признание
```

**Description**
Тайм-аут для алгоритма замедленного подтверждения

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

### 10.18.3.26 TCP_FIN_WAIT_2_TIMEOUT Macro
**File**
TCP.c
**C**
```
#define TCP_FIN_WAIT_2_TIMEOUT ((DWORD)TICK_SECOND*5) // Тайм-аут для FIN ждать 2 государства
```

**Description**
Тайм-аут для FIN ждать 2 государства

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

### 10.18.3.27 TCP_HEADER Structure
**File**
TCP.c

**C**
```
typedef struct {
WORD SourcePort;
WORD DestPort;
DWORD SeqNumber;
DWORD AckNumber;
struct {
unsigned char Reserved3 : 4;
unsigned char Val : 4;
} DataOffset;
union {
struct {
unsigned char flagFIN : 1;
unsigned char flagSYN : 1;
unsigned char flagRST : 1;
unsigned char flagPSH : 1;
unsigned char flagACK : 1;
unsigned char flagURG : 1;
unsigned char Reserved2 : 2;
} bits;
BYTE byte;
} Flags;
WORD Window;
WORD Checksum;
WORD UrgentPointer;
```

} TCP_HEADER;
**Members**
**Members Description**
WORD SourcePort; Local port number
WORD DestPort; Remote port number
DWORD SeqNumber; Часовой порядковый номер

DWORD AckNumber; Признавая удаленного порядковый номер

struct {
unsigned char Reserved3 : 4;
unsigned char Val : 4;
} DataOffset;
Data offset flags nibble
union {
struct {
unsigned char flagFIN : 1;
unsigned char flagSYN : 1;
unsigned char flagRST : 1;
unsigned char flagPSH : 1;
unsigned char flagACK : 1;
unsigned char flagURG : 1;
unsigned char Reserved2 : 2;
} bits;
BYTE byte;
} Flags;
TCP Flags as defined in RFC
WORD Window; Часовой окна бесплатно RX буфер
WORD Checksum; Контрольная нагрузка данных

WORD UrgentPointer; Срочно указатель
**Description**
TCP Header Data Structure

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

### 10.18.3.28 TCP_KEEP_ALIVE_TIMEOUT Macro
**File**
TCP.c
**C**
```
#define TCP_KEEP_ALIVE_TIMEOUT ((DWORD)TICK_SECOND*10) //  Тайм-аут для поддержания активности Сообщения , когда трафик не отправленные
```

**Description**
Тайм-аут для Keep-Alive сообщений, когда трафик не передается

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

### 10.18.3.29 TCP_MAX_RETRIES Macro

**File**
TCP.c

**C**
```
#define TCP_MAX_RETRIES (5u) // Максимальное количество попыток повторной передачи
```

**Description**
Максимальное количество попыток повторной передачи

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

### 10.18.3.30 TCP_MAX_SEG_SIZE_RX Macro
**File**
TCP.c
**C**
```
#define TCP_MAX_SEG_SIZE_RX (536u)
```
**Description**
TCP максимальный размер сегмента для RX. Это значение advirtised во время установления соединения и удаленного узла следует повиноваться ему . Это должен быть установлен на 536 , чтобы избежать фрагментации IP слоя от причинения потери пакетов . Тем не менее, повышение ее значение может повысить производительность на (небольшой) риск введения несовместимость с некоторыми специальными удаленных узлов (пример: числе связанных через медленный до звона модем ) .

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

### 10.18.3.31 TCP_MAX_SEG_SIZE_TX Macro
**File**
TCP.c
**C**
```
#define TCP_MAX_SEG_SIZE_TX (1460u)
```
**Description**
TCP максимальный размер сегмента для TX. Максимальный размер сегмента Техас фактически **govered** по MSS опции удаленного узла

**advirtised** во время установления соединения. Тем не менее, если удаленный узел определяет **unhandlably** большой MSS (например: > Ethernet MTU) , это определить устанавливает жесткое ограничение , так что мы не вызывают никаких переполнение буфера TX . Если удаленный узел не advirtise Опция MSS , все сегменты TX фиксируются на 536 байт максимум.

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

### 10.18.3.32 TCP_MAX_SYN_RETRIES Macro

**File**
TCP.c

**C**
```
#define TCP_MAX_SYN_RETRIES (2u) // Меньше, чем всех других попыток сократить SYN Flood
```

**Description**
Меньше, чем всех других попыток сократить SYN Flood DoS продолжительность

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

### 10.18.3.33 TCP_MAX_UNACKED_KEEP_ALIVES Macro

**File**
TCP.c

**C**
```
#define TCP_MAX_UNACKED_KEEP_ALIVES (6u) // Maximum number of keep-alive messages that can be sent without receiving a response before automatically closing the connection
```

**Description**
Максимальное количество Keep-Alive сообщения, которые могут быть отправлены без ответа , прежде чем автоматически закрытия  подключение

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

### 10.18.3.34 TCP_OPTIMIZE_FOR_SIZE Macro

**File**
TCP.c

**C**
```
#define TCP_OPTIMIZE_FOR_SIZE
```

**Description**
Для маленького размера и лучшей пропускной способности , TCP_OPTIMIZE_FOR_SIZE всегда должен быть включен на продукты PIC24/dsPIC .

На PIC32 продукции существует очень мало разницы и depnds на уровне оптимизации компиляторов

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

### 10.18.3.35 TCP_OPTIONS Structure

**File**
TCP.c
**C**
```
typedef struct {
BYTE Kind;
BYTE Length;
WORD_VAL MaxSegSize;
} TCP_OPTIONS;
```
**Members**
**Members Description**
BYTE Kind; Type of option
BYTE Length; Length
WORD_VAL MaxSegSize; Максимальный размер сегмента
**Description**
Структура данных параметры TCP

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

### 10.18.3.36 TCP_OPTIONS_END_OF_LIST Macro

**File**
TCP.c
**C**
```
#define TCP_OPTIONS_END_OF_LIST (0x00u) // Конец Список TCP Option Отметить
```
**Description**
Конец Список TCP Option Отметить

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

### 10.18.3.37 TCP_OPTIONS_MAX_SEG_SIZE Macro

**File**
TCP.c

**C**
```
#define TCP_OPTIONS_MAX_SEG_SIZE (0x02u) // Максимальный размер сегмента флаг TCP
```

**Description**
Максимальный размер сегмента флаг TCP

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

### 10.18.3.38 TCP_OPTIONS_NO_OP Macro

**File**
TCP.c

**C**
```
#define TCP_OPTIONS_NO_OP (0x01u) // Вариант № Op TCP
```

**Description**
Нет Опция Op TCP

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

### 10.18.3.39 TCP_SOCKET_COUNT Macro

**File**
TCP.c

**C**
```
#define TCP_SOCKET_COUNT (sizeof(TCPSocketInitializer)/sizeof(TCPSocketInitializer[0]))
```

**Description**
Определяет количество определенных розеток TCP

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

### 10.18.3.40 TCP_START_TIMEOUT_VAL Macro

**File**
TCP.c

**C**
```
#define TCP_START_TIMEOUT_VAL ((DWORD)TICK_SECOND*1) // Тайм-аут , чтобы повторно Unacked данные
```

**Description**

Тайм-аут для повторной передачи Unacked данные

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

### 10.18.3.41 TCP_SYN_QUEUE Structure

**File**
TCP.c

**C**
```
typedef struct {
NODE_INFO niSourceAddress;
WORD wSourcePort;
DWORD dwSourceSEQ;
WORD wDestPort;
WORD wTimestamp;
} TCP_SYN_QUEUE;
```

**Members**
**Members Description**
NODE_INFO niSourceAddress ; Удаленный IP- адрес и MAC- адрес

WORD wSourcePort ; Число дистанционного TCP порт , что ответ SYN должен быть отправлен в

DWORD dwSourceSEQ ; Удаленный TCP порядковый номер , который должен быть признан , когда мы посылаем наш ответ SYN

WORD wDestPort ; Локальный порт TCP , который оригинальный SYN был предназначен для

WORD wTimestamp ; Таймер истекает старые SYN -пакеты , которые не могут быть обслужены вообще

**описание**

Структура, которая содержит все важные элементы объекта Входящие SYN пакета для того, чтобы установить связь в будущем время, если все разъемы на порт прослушивания уже подключены к кому-то

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

### 10.18.3.42 TCP_SYN_QUEUE_MAX_ENTRIES Macro

**File**
TCP.c
**C**
```
#define TCP_SYN_QUEUE_MAX_ENTRIES (3u) // Количество TCP RX SYN -пакетов спасти , если они не могут быть обслужены немедленно
```

**Description**
Количество TCP SYN пакетов RX , чтобы сохранить если они не могут быть обслужены немедленно

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

### 10.18.3.43 TCP_SYN_QUEUE_TIMEOUT Macro

**File**
TCP.c
**C**
```
#define TCP_SYN_QUEUE_TIMEOUT ((DWORD)TICK_SECOND*3) // Тайм-аут для того, когда  очереди SYN записи будут удалены , если непригодным
```

**Description**
Тайм-аут для того, когда записи очереди SYN удаляются, если непригодным

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

### 10.18.3.44 URG Macro

**File**
TCP.c

**C**
```
#define URG (0x20) // Срочно Флаг , как определено в RFC
```

**Description**
Срочно Флаг , как определено в RFC

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

## 10.18.4 Variables

**Module**
TCP ( see page 526)

**Variables**
**Name Description**
NextPort (см. стр. 571 ) переменную отслеживания в течение следующего числа локальный порт клиента

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

### 10.18.4.1 NextPort Variable

**File**
TCP.c

**C**
WORD **NextPort**;
**Description**
Отслеживание переменную для следующего числа локальный порт клиента

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

### 10.19 Telnet

Telnet обеспечивает двунаправленную интерактивную связь между двумя узлами в Интернете или на локальной вычислительной сети .

Telnet код включен с TCP / IP стека компании Microchip является демонстрацией структуры приложения Telnet. Это демо начинается с прослушивания для подключения Telnet. Когда клиент пытается сделать один, демо подскажет клиенту для имя пользователя и пароль , и если правильным обеспечивается , выведет и периодически обновлять несколько значений , полученных из Демонстрационная плата .

Есть несколько изменений , что может понадобиться , чтобы сделать на Telnet.c и / или Telnet.h , чтобы удовлетворить вашу заявку . Все Члены Telnet с общественностью может быть заново определены в разделе конкретных приложений из TCPIPConfig.h . Вы также можете изменить некоторые строки Telnet Внутренняя членов, расположенных в Telnet.c , чтобы более точно отразить вашу заявку . вы также необходимо изменить TelnetTask (см. стр. 574 ) функционировать включить функциональность вы хотели бы . Вы можете вставлять или изменять состояния в TelnetTask (см. стр. 574 ) по мере необходимости.

#### 10.19.1 Общественных Макро Участников Telnet

**макрос**

**Имя Описание**

MAX_TELNET_CONNECTIONS

(См. стр. 572 )

Количество одновременно позволило Telnet (см. стр. 572) сессий . примечание что вы должны иметь равное число типа TCP_PURPOSE_TELNET TCP розетки , объявленные в TCPSocketInitializer [ ] массива выше для многократного соединения для работы . Если меньше розетки доступны , чем этого определения , то меньшее из двух величин будет реальный предел .

**TELNET_PASSWORD** (см.страница 573) По умолчанию Telnet (см. стр. 572 ) пароля

**TELNET_PORT** (см. стр.573)По умолчанию локальный порт прослушивания для Telnet (см. стр. 572) сервера . Порт 23 является протокол по умолчанию .

**TELNETS_PORT** (см. стр.573)По умолчанию локальный порт прослушивания для Telnet (см. стр. 572) сервер , когда SSL обеспечено. Порт 992 является протокол telnets умолчанию .

**TELNET_USERNAME** (см.страница 573)По умолчанию имя пользователя и пароль, необходимый для входа в Telnet (см. стр.572) сервер .

**модуль**

Telnet (см. стр. 572 )

**описание**

Следующие функции и переменные доступны для применения стека.

### 10.19.1.1 MAX_TELNET_CONNECTIONS Макро

**файл**

TCPIP MRF24W.h

C

**#define MAX_TELNET_CONNECTIONS** (1u)

**описание**

Количество одновременно позволило Telnet (см. стр. 572) сессий . Обратите внимание, что вы должны иметь равное количество Тип **TCP_PURPOSE_TELNET TCP** розетки , объявленные в **TCPSocketInitializer** [ ] массива выше для нескольких подключений к работа . Если меньше розетки доступны , чем этого определения , то наименьшее из двух величин будет реальный предел .

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

### 10.19.1.2 TELNET_PASSWORD Макро

TCPIP MRF24W.h

C

**#define TELNET_PASSWORD** "microchip"

**Описание**

По умолчанию пароль Telnet (см. стр. 572 )

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

### 10.19.1.3 TELNET_PORT Макро

TCPIP MRF24W.h

C

```
#define TELNET_PORT 23
```

**Описание**

По умолчанию локальный порт прослушивания для Telnet (см. стр. 572) сервера . Порт 23 является протокол по умолчанию .

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

### 10.19.1.4 TELNETS_PORT Макро

TCPIP MRF24W.h

C

```
#define TELNETS_PORT 992
```

**Описание**

По умолчанию локальный порт прослушивания для Telnet ( см. стр. 572) сервера, когда SSL обеспечено. Порт 992 является протокол telnets умолчанию .

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

### 10.19.1.5 TELNET_USERNAME Макро

**файл**

TCPIP MRF24W.h

C

```
#define TELNET_USERNAME "admin"
```

**описание**

По умолчанию имя пользователя и пароль, необходимый для входа в Telnet (см. стр. 572) сервера .

### 10.19.2 Telnet Стек Пользователи

**Функции**

**Имя Описание**

**TelnetTask** (см. стр. 574 ) Выполняет Telnet (см. стр. 572 ) серверных задач , связанных с . Содержит Telnet (см. стр. 572) государственную машину и переменные отслеживания состояния .

**модуль**

Telnet (см. стр. 572 )

**описание**

Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные .

#### 10.19.2.1 TelnetTask Функция

**файл**

Telnet.h

C

```
void TelnetTask();
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Выполняет Telnet (см. стр. 572 ) серверных задач , связанных с . Содержит Telnet (см. стр. 572 ) государственная машина и государственную отслеживание переменных.

**Замечания**

Ни один

**Предпосылки**

Стек инициализируется ( )

### 10.19.3 Telnet Внутренняя Пользователи

**модуль**

**Telnet** (см. стр. 572 )

**Переменные**

**Имя Описание**

**strSpaces** (см. стр. 575 ) Строка с дополнительными пространств , для Demo

**strAuthenticated** (см. стр.575) Успешное сообщение аутентификации

**strDisplay** (см. стр. 575 ) Демо выходная строка

**strGoodBye** ( см. стр. 575 ) Демо разъединения сообщение

**strPassword** (см. стр.576) DO Подавить Локальное эхо ( остановка Telnet-клиент от печатных напечатанных символов ) Доступ запрещен сообщение

**strTitle** (см. стр. 576) Демо заголовок строки

**описание**

Следующие функции и переменные определяются как внутренние, к Telnet (см. стр. 572) модуля.

#### 10.19.3.1 strSpaces Переменная

**файл**

Telnet.c

C

ROM BYTE **strSpaces**[] = " ";

**описание**

Строка с дополнительными пространств , для Demo

#### 10.19.3.2 strAuthenticated Переменная

**файл**

Telnet.c

C

ROM BYTE **strAuthenticated**[] = "\r\nLogged in successfully\r\n\r\n" "\r\nPress 'q' to

quit\r\n";

**описание**

Успешное сообщение аутентификации

**10.19.3.3 strDisplay Переменная**

**файл**

Telnet.c

C

ROM BYTE **strDisplay**[] = "\r\nSNTP Time: (disabled)" "\r\nAnalog: 1023"
"\r\nButtons: 3 2 1 0" "\r\nLEDs: 7 6 5 4 3 2 1 0";

**описание**

Демо выходная строка

**10.19.3.4 strGoodBye Переменная**

**файл**

Telnet.c

C

ROM BYTE **strGoodBye**[] = "\r\n\r\nGoodbye!

**описание**

Демо сообщение отключение

**10.19.3.5 strPassword Переменная**

**файл**

Telnet.c

C

ROM BYTE **strPassword**[] = "Password: \xff\xfd\x2d";

**описание**

НЕ Подавить Локальное эхо (клиент остановка Telnet из печати набрали символов) Доступ запрещен сообщение

**10.19.3.6 strTitle Переменная**

**файл**

Telnet.c

C

ROM BYTE **strTitle**[] = "\x1b[2J\x1b[31m\x1b[1m" "Microchip Telnet Server 1.1\x1b[0m\r\n"
" ( для этого демо , тип " администратора "для входа в систему и " микрочип "для пароля. ) \ г \ п " " Логин:  " ;

**описание**

Демо строка заголовка

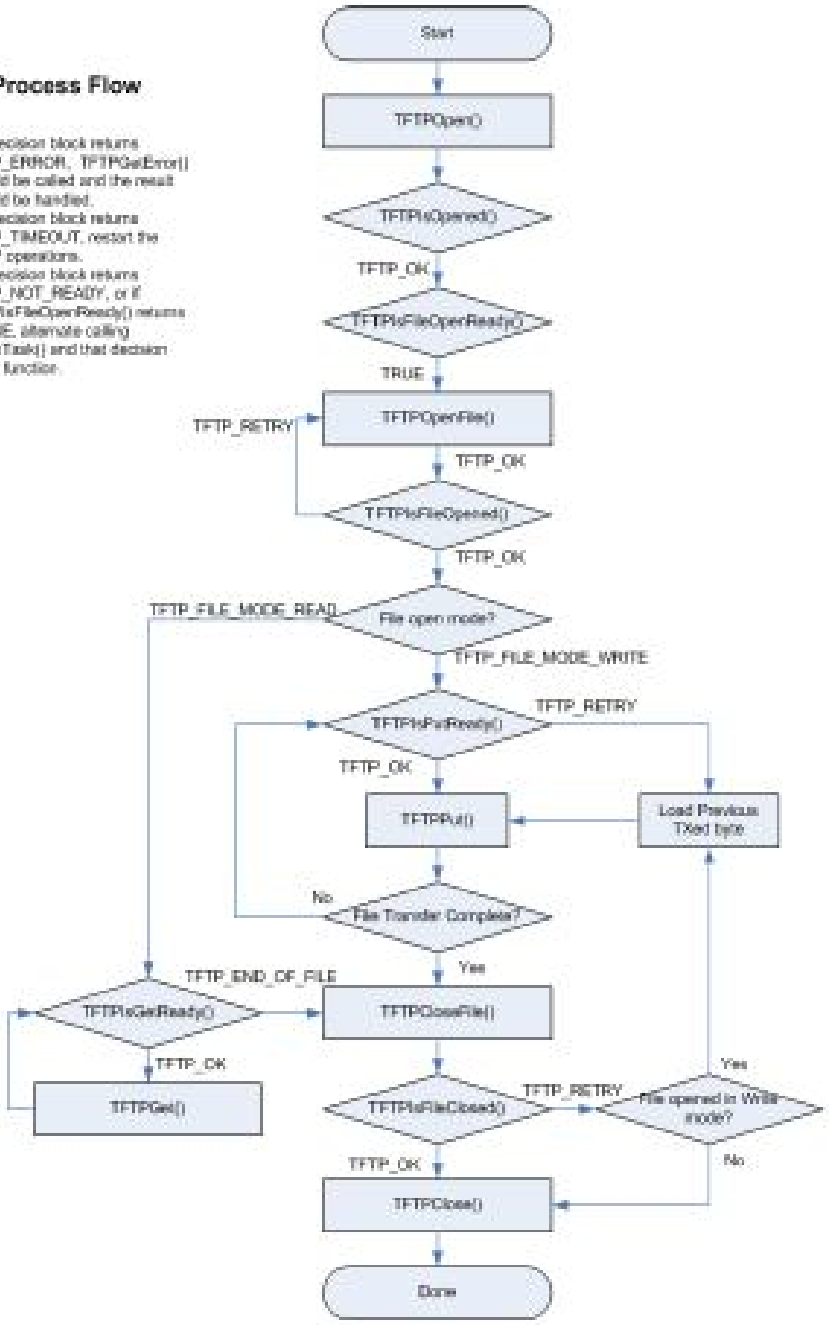
**10.20 TFTP**

Упрощенный протокол передачи файлов обеспечивает ненадежны для закачивания и скачивания сервисы для приложений , связанных с  UDP - сервер на базе TFTP .

## TFTP Process Flow

Notes

- If a decision block returns TFTP_ERROR, TFTPGetError() should be called and the result should be handled.
- If a decision block returns TFTP_TIMEOUT, restart the TFTP operations.
- If a decision block returns TFTP_NOT_READY, or if TFTPIsFileOpenReady() returns FALSE, alternate calling StackTask() and that decision block function.



### 10.20.1 TFTP Общественная Пользователи

**Перечисления**

**Имя Описание**

**TFTP_ACCESS_ERROR** (см. Стр. 586 ) Стандартные коды ошибок , как это определено TFTP спец. Используйте для декодирования значения повторно настроились по TFTPGetError (см. стр. 580 ) ( ) .

**_TFTP_ACCESS_ERROR**(См. Стр. 586 ) Стандартные коды ошибок , как это определено TFTP спец. Используйте для декодирования значения повторно настроились по TFTPGetError (см. стр. 580 ) ( ) .

**TFTP_FILE_MODE** (см.страница 586 ) Файл открытый режим , используемый по TFTPFileOpen ( ) .

**_TFTP_FILE_MODE** (См.страница 586 ) Файл открытый режим , используемый по TFTPFileOpen ( ) .

**TFTP_RESULT** (см. стр.587) Enum . результатов , возвращаемый большинство функций TFTP .

**_TFTP_RESULT** (См.страница 587 ) Enum . результатов , возвращаемый большинство функций TFTP .

**Функции**

**Имя Описание**

**TFTPCloseFile** (см. стр. 579 ) Посылает файлов сообщений закрытия.

**TFTPGet** (см. стр. 580 ) Получает байт данных от данных, которые были прочитать.

**TFTPIsFileClosed** (см. стр. 581 ) Определяет, является ли файл был закрыт .

**TFTPIsFileOpened** (см. стр. 581 ) Определяет , если файл был открыт .

**TFTPIsGetReady** (см. стр. 582 ) Определяет, является ли блок данных готова для чтения.

**TFTPIsOpened** (см. стр. 583 ) Определяет , если соединение TFTP открыт .

**TFTPIsPutReady** (см. стр. 583 ) Определяет, данные могут быть записаны в файл.

**TFTPOpen** (см. стр. 584 ) инициализацию модуля TFTP .

**TFTPOpenFile** (см. стр. 585 ) готовит и отправляет имя файла TFTP и пакетном режиме .

**TFTPOpenROMFile** (см. стр. 585 ) осуществление PIC18 ROM аргумент **TFTPOpenFile** (см.страница 585 )

**TFTPPut** (см. стр. 586 ) Написать байт в файл.

**TFTPGetUploadStatus** (см. стр. 587 ) Возвращает состояние загрузки файлов TFTP начинаются посредством вызова **TFTPUploadRAMFileToHost** (см. стр. 589 ) () или **TFTPUploadFragmentedRAMFileToHost** (см. стр. 588 ) ( )функции.

**TFTPUploadFragmentedRAMFileToHost**(См. Стр. 588 ) Передает в случайное , потенциально не смежные , массив байтов RAM в виде файла на удаленный сервер TFTP.

**TFTPUploadRAMFileToHost** (см.страница 589 ) Передает непрерывный массив байт RAM в виде файла на удаленный TFTP сервер .

**макрос**

**Имя Описание**

T**FTPClose** ( см. стр. 579 ) Макро : void **TFTPClose** (void) Закрывает сокет клиента TFTP .

**TFTPGetError** (см. стр. 580 ) Макро : WORD **TFTPGetError** (void) Возвращает ранее сохраненный код ошибки.

**TFTPIsFileOpenReady** (см. стр. 582 ) Макро : BOOL **TFTPIsFileOpenReady** (void) Проверяет , что это хорошо , чтобы отправить файл TFTP открытый запрос удаленный сервер .

**TFTP_UPLOAD_COMPLETE** (см. стр.590) Коды состояния для **TFTPGetUploadStatus** (см. стр. 587 ) ( ) функция. Ноль означает, что обновление успех , > 0 означает работать и < 0 означает фатальную ошибку.

**TFTP_UPLOAD_CONNECT** (см. стр. 590 ) Этот макрос **TFTP_UPLOAD_CONNECT** .

**TFTP_UPLOAD_CONNECT_TIMEOUT** (см. Стр. 590) Это макро **TFTP_UPLOAD_CONNECT_TIMEOUT** .

**TFTP_UPLOAD_GET_DNS** ( см. стр. 590 ) Этот макрос **TFTP_UPLOAD_GET_DNS** .

**TFTP_UPLOAD_HOST_RESOLVE_TIMEOUT** (См. Стр. 591 ) Это макро **TFTP_UPLOAD_HOST_RESOLVE_TIMEOUT** .

**TFTP_UPLOAD_RESOLVE_HOST** (см.страница 591 ) Это макро **TFTP_UPLOAD_RESOLVE_HOST** .

**TFTP_UPLOAD_SEND_DATA** (см. Стр.591) Это макро **TFTP_UPLOAD_SEND_DATA** .

**TFTP_UPLOAD_SEND_FILENAME** (см.страница 591 ) Это макро **TFTP_UPLOAD_SEND_FILENAME** .

**TFTP_UPLOAD_SERVER_ERROR** (см.страница 591 ) Это макро **TFTP_UPLOAD_SERVER_ERROR** .

**TFTP_UPLOAD_WAIT_FOR_CLOSURE** (см. Стр. 592 ) Это макро **TFTP_UPLOAD_WAIT_FOR_CLOSURE** .

**модуль**

**TFTP** (см. стр. 576 )

**структуры**

**Имя Описание**

**TFTP_CHUNK_DESCRIPTOR**(См. Стр. 589 ) Это тип **TFTP_CHUNK_DESCRIPTOR** .

**описание**

Следующие функции и переменные доступны для применения стека.

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

### 10.20.1.1 TFTPClose Макро

**файл**

TFTPc.h

**C**

```
#define TFTPClose (void) UDPClose ( _tftpSocket )
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Макро : void TFTPClose (void) UDPClose

Закрывает сокет клиента TFTP .

**Замечания**

После того, как закрывается , приложение должно сделать **TFTPOpen** (см. стр. 584 ) для выполнения любых новых операций TFTP .

Если сервер TFTP не изменяется в течение приложения срока службы , не может нужно позвонить **TFTPClose** и держать гнездо TFTP открытым.

**Предпосылки**

**TFTPOpen** (см. стр. 584 ) уже называют и **TFTPIsOpened** (см. стр. 583 ) ( ) возвращается TFTP_OK .

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

**10.20.1.2 TFTPCloseFile Функция**

**файл**

TFTPc.h

C

void TFTPCloseFile ( ) ;

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Если файл открывается в режиме чтения , это гарантирует, что в прошлом ACK посылается сервером Если файл открыт в режиме записи , он гарантирует, что Последний блок отправляется на сервер и ждет ответа сервера с ACK.

**Замечания**

**TFTPIsFileClosed** (см. стр. 581 ) ( ) должен быть вызван для подтверждения , если файл был действительно закрыт .

**Предпосылки**

**TFTPOpenFile** (см. стр. 585 ) () была вызвана и **TFTPIsFileOpened** (см. стр. 581 ) ( ) вернулся с TFTP_OK .

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

**10.20.1.3 TFTPGet Функция**

**файл**

TFTPc.h

C

BYTE TFTPGet ( ) ;

**Побочные эффекты**

Ни один

**Возвращает**

байт данных , как полученных от удаленного сервера.

**описание**

Выбирает следующий байт данных из сокета TFTP. Если конец блока данных будет достигнута , он выдает ACK на сервере, так что следующий блок данных может быть получены.

**Замечания**

Используйте эту функцию , чтобы прочитать файл с сервера .

**Предпосылки**

**TFTPOpenFile** (см. стр. 585 ) ( ) вызывается с TFTP_FILE_MODE_READ и **TFTPIsGetReady** (см. стр. 582 ) ( ) =ИСТИНА

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

### 10.20.1.4 TFTPGetError Макро

**файл**

TFTPc.h

C

#define TFTPGetError ( _tftpError )

**Побочные эффекты**

Ни один

**Возвращает**

Код ошибки , возвращаемый удаленном сервере. Заявка может использовать TFTP_ACCESS_ERROR (см. стр. 586 ) перечисления . Декодировать стандартный код ошибки .

**описание**

Макро : СЛОВО TFTPGetError (недействительными)

Возвращает ранее сохраненный код ошибки.

**Замечания**

Ни один

**Предпосылки**

Один из функции TFTP вернулся с результатом TFTP_ERROR .

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

**10.20.1.5 TFTPIsFileClosed Функция**

**файл**

TFTPc.h

C

TFTP_RESULT TFTPIsFileClosed ( ) ;

**Побочные эффекты**

Ни один

**Возвращает**

**TFTP_OK** если файл был успешно **closdd**

**TFTP_RETRY** если режим файла было писать и удаленный сервер не получит последний пакет . Заявка должна повторить с последнего блока .

**TFTP_TIMEOUT** если все попытки были исчерпаны в заключительном файла.

**TFTP_ERROR** если удаленный сервер послал сообщение об ошибке в ответ на последний блок . Фактический код ошибки может быть прочитан по телефону

**TFTPGetErro**r (см. стр. 580 ) ( )

**TFTP_NOT_READY** если файл еще не закрыто .

**описание**

Если режим файл читается , он просто делает , что последний блок признал . Если режим файла Написать , он ждет сервера ACK. Если нет ACK

был получен в течение указанного времени ожидания инструктирует Appliaction для повторной последний блок . Он отслеживает попыток и заявляет тайм-аут

все попытки были исчерпаны .

**Замечания**

Ни один

**Предпосылки**

**TFTPCloseFile** (см. стр. 579 ) ( ) уже называют .

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

**10.20.1.6 TFTPIsFileOpened Функция**

**файл**

TFTPc.h

C

TFTP_RESULT TFTPIsFileOpened ( ) ;

**Побочные эффекты**

Ни один

**Возвращает**

**TFTP_OK** если файл готов чтения или записи

**TFTP_RETRY** если предыдущая попытка была истекло необходимо повторно.

**TFTP_TIMEOUT** если все попытки были исчерпаны .

**TFTP_ERROR** если удаленный сервер ответил с ошибкой

**TFTP_NOT_READY** если файл еще не началась .

**описание**

Ожидает дистанционного ответа сервера о предыдущей попытке открыть файл . Если ответ не будет получен в течение указанного времени ожидания , fnction возвращается с **TFTP_RETRY** и логики приложения должны выдать другой **TFTPFileOpen** () .

**Замечания**

Ни один

**Предпосылки**

**TFTPOpenFile** (см. стр. 585 ) () вызывается .

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

**10.20.1.7 TFTPIsFileOpenReady Макро**

**файл**

TFTPc.h

C

```
#define  TFTPIsFileOpenReady UDPIsPutReady ( _tftpSocket )
```

**Побочные эффекты**

Ни один

**Возвращает**

Правда, если это нормально, чтобы позвонить TFTPOpenFile (см. стр. 585 ) ( ) False , если нет.

**описание**

Макро : BOOL TFTPIsFileOpenReady (недействительными)

Проверяет , что это хорошо , чтобы отправить файл TFTP открытый запрос к удаленному серверу .

**Замечания**

Ни один

**Предпосылки**

**TFTPOpen** (см. стр. 584 ) уже называют и **TFTPIsOpened** (см. стр. 583 ) ( ) возвращается **TFTP_OK** .

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

**10.20.1.8 TFTPIsGetReady Функция**

**файл**

TFTPc.h

C

TFTP_RESULT TFTPIsGetReady ( ) ;

**Побочные эффекты**

Ни один

**Возвращает**

**TFTP_OK** если он есть больше байт данных , доступных для чтения

**TFTP_TIMEOUT** если тайм-аут произошел ждет новых данных.

**TFTP_END_OF_FILE** если конец файла достиг .

**TFTP_ERROR** если удаленный сервер возвратил ошибку . Фактический код ошибки может быть прочитан по телефону TFTPGetError (см. стр.580 ) ()

**TFTP_NOT_READY** если еще ждут новых данных.

**описание**

Ожидает блока данных. Если блок данных не поступает в течение указанного времени ожидания , он автоматически посылает ACK для предыдущего блока , чтобы напомнить сервер для отправки следующий блок данных. Если все попытки будут исчерпаны , он возвращается с TFTP_TIMEOUT .

**Замечания**

По умолчанию этот Funciton использует " октет " или двоичного режима передачи файлов .

**Предпосылки**

**TFTPOpenFile** (см. стр. 585 ) ( ) вызывается с **TFTP_FILE_MODE_READ** и **TFTPIsFileOpened** (см. стр. 581 ) ( ) вернулся с TRUE.

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

### 10.20.1.9 TFTPIsOpened Функция

**файл**

TFTPc.h

C

TFTP_RESULT TFTPIsOpened ( ) ;

**Побочные эффекты**

Ни один

**Возвращает**

**TFTP_OK** если предыдущая вызов **TFTPOpen** (см. стр. 584 ) завершена

**TFTP_TIMEOUT** если удаленный хост не ответил на предыдущий ARP запроса.

**TFTP_NOT_READY** если удаленный до сих пор не ответил и тайм-аут еще не истек .

**описание**

Ожидает ARP ответ и открывает UDP сокет для выполнения дальнейших операций TFTP .

**Замечания**

После вскрытия приложение может держать гнездо TFTP открытым и будущих операций TFTP . Если TFTPClose (см. стр. 579 ) ( ) является призван закрыть соединение **TFTPOpen** (см. стр. 584 ) () должен быть вызван снова перед выполнением любой другой TFTP операций.

**Предпосылки**

**TFTPOpen** (см. стр. 584 ) ( ) уже называют .

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

**10.20.1.10 TFTPIsPutReady Функция**

**файл**

TFTPc.h

C

TFTP_RESULT TFTPIsPutReady ( ) ;

**Побочные эффекты**

Ни один

**Возвращает**

**TFTP_OK** если это нормально написать еще байт данных .

**TFTP_TIMEOUT** если Истекло ждет ACK от сервера

**TFTP_RETRY** если все сервер не послал ACK от времени и применения необходимо отправить последний блок .

**TFTP_ERROR** если удаленный сервер возвратил ошибку . Фактический код ошибки может быть прочитан по телефону **TFTPGetError** (см. стр.580 ) ()

**TFTP_NOT_READY** если еще ждут ...

**описание**

Ожидает ACK от сервера . Если ACK не поступает в течение указанного времени ожидания , то он указывает приложение , чтобы повторить последний блок на возвращение **TFTP_RETRY** .

Если все попытки будут исчерпаны , он возвращается с **TFTP_TIMEOUT .**

**Замечания**

Ни один

**Предпосылки**

**TFTPOpenFile** (см. стр. 585 ) ( ) вызывается с **TFTP_FILE_MODE_WRITE** и **TFTPIsFileOpened** (см. стр. 581 ) ( ) вернулся с TRUE.

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

**10.20.1.11 TFTPOpen Функция**

**файл**

TFTPc.h

C

```
void TFTPOpen (
  IP_ADDR * host
) ;
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Инициирует ARP для данного хоста и готовит модуль TFTP на следующий последовательности вызовов функций .

**Замечания**

Используйте **TFTPIsOpened** (см. стр. 583 ) () , чтобы проверить , если соединение было успешно открыт или нет.

**Предпосылки**

Модуль UDP уже инициализирован и по крайней мере один UDP сокет доступен .

**Параметры**

**Параметры Описание**

Host IP -адрес удаленного сервера TFTP

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

**10.20.1.12 TFTPOpenFile Функция**

**файл**

TFTPc.h

C

```
void TFTPOpenFile (
  BYTE * fileName,
  TFTP_FILE_MODE mode
) ;
```

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Готовит и передает имя файла TFTP и пакетном режиме .

**Замечания**

По умолчанию этот Funciton использует " октет " или двоичного режима передачи файлов . Используйте **TFTPIsFileOpened** (см. стр. 581 ) ( ) , чтобы проверить,

файл готов для чтения или записи.

**Предпосылки**

**TFPTIsFileOpenReady** ( ) = ИСТИНА

**Параметры**

**Параметры Описание**

имя_файла Имя файла , который должен быть открыт .

Режим Режим доступа файл должен быть **TFTP_FILE_MODE_READ** для чтения

**TFTP_FILE_MODE_WRITE** для записи

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

**10.20.1.13 TFTPOpenROMFile Функция**

**файл**

TFTPc.h

C

```
void TFTPOpenROMFile (
   ROM BYTE * fileName,
   TFTP_FILE_MODE mode
) ;
```

**описание**

Реализация PIC18 ROM аргумент TFTPOpenFile (см. стр. 585 )

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

**10.20.1.14 TFTPPut Функция**

**файл**

TFTPc.h

C

```
void TFTPPut (
BYTE c
```

) ;

**Побочные эффекты**

Ни один

**Возвращает**

Ни один

**описание**

Кладет учитывая байт данных в гнездо TFTP. Если конец блока данных будет достигнута , он передает весь блок .

**Замечания**

Используйте эту функцию , чтобы написать файл на сервер.

**Предпосылки**

**TFTPOpenFile** (см. стр. 585 ) ( ) вызывается с **TFTP_FILE_MODE_WRITE** и **TFTPIsPutReady** (см. стр. 583 ) ( ) =ИСТИНА

**Параметры**

**Параметры Описание**

с байт данных , который должен быть написан

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

**10.20.1.15 TFTP_ACCESS_ERROR Перечисление**

**файл**

TFTPc.h

C

typedef enum _TFTP_ACCESS_ERROR {

TFTP_ERROR_NOT_DEFINED = 0,

TFTP_ERROR_FILE_NOT_FOUND ,

TFTP_ERROR_ACCESS_VIOLATION ,

TFTP_ERROR_DISK_FULL ,

TFTP_ERROR_INVALID_OPERATION ,

TFTP_ERROR_UNKNOWN_TID ,

TFTP_ERROR_FILE_EXISTS ,

TFTP_ERROR_NO_SUCH_USE

} TFTP_ACCESS_ERROR ;

**описание**

Стандартные коды ошибок , как это определено TFTP спец. Используется для декодирования значения настраивали на **TFTPGetError** (см. стр. 580 ) ( ) .

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

### 10.20.1.16 TFTP_FILE_MODE Перечисление

**файл**

TFTPc.h

C

typedef enum _TFTP_FILE_MODE {

TFTP_FILE_MODE_READ = 1,

TFTP_FILE_MODE_WRITE = 2

} TFTP_FILE_MODE ;

**описание**

Файл открытый режим , используемый по **TFTPFileOpen** ( ) .

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

### 10.20.1.17 TFTP_RESULT Перечисление

**файл**

TFTPc.h

C

typedef enum _TFTP_RESULT {

TFTP_OK = 0,

TFTP_NOT_READY ,

TFTP_END_OF_FILE ,

TFTP_ERROR ,

TFTP_RETRY ,

TFTP_TIMEOUT

} TFTP_RESULT ;

**описание**

**Enum** . результатов , возвращаемый большинство функций TFTP .

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

**10.20.1.18 TFTPGetUploadStatus Функция**

**файл**

TFTPc.h

C

CHAR TFTPGetUploadStatus ( ) ;

**Возвращает**

Код состояния . Отрицательные результаты фатальные ошибки . Положительные результаты показывают операции отправки TFTP еще обрабатывается .

Нулевой результат указывает на успешное завершение загрузки файла ( API TFTP теперь сложа руки и можно использовать для дальнейших вызовов) . Удельный возврат

следующие значения :

0 ( **TFTP_UPLOAD_COMPLETE** (см. стр. 590 ) ) : Загрузить успешно завершена

1( **TFTP_UPLOAD_GET_DNS** (см. стр. 590 ) ) : Попытка получить клиентский модуль DNS

2( **TFTP_UPLOAD_RESOLVE_HOST** (см. стр. 591 ) ) : Попытка решить TFTP имя хоста

3( **TFTP_UPLOAD_CONNECT** (см. стр. 590 ) ) : Попытка ARP и связаться с сервером TFTP
4

( **TFTP_UPLOAD_SEND_FILENAM**E (см. стр. 591 ) ) : Попытка отправить имя файла и получить подтверждение .

5( **TFTP_UPLOAD_SEND_DATA** (см. стр. 591 ) ) : Попытка отправить содержимое файла и получить подтверждение .

6( **TFTP_UPLOAD_WAIT_FOR_CLOSURE** (см. стр. 592 ) ) : Попытка направить окончательный пакет содержимого файлов и получать подтверждение.

-1 ( **TFTP_UPLOAD_HOST_RESOLVE_TIMEOUT** (см. стр. 591 ) ) : Не удалось разрешить имя хоста

-2( **TFTP_UPLOAD_CONNECT_TIMEOUT** (см. стр. 590 ) ) : Не удалось завершить ARP и получить доступ к серверу

-3( **TFTP_UPLOAD_SERVER_ERROR** (см. стр. 591 ) ) : сервер TFTP возвратил ошибку ( например: отказ в доступе ) или загрузку файлов не удалось из-за тайм-аута (частичная файл может быть загружен ) .

**описание**

Возвращает состояние загрузки файлов TFTP начал , вызвав **TFTPUploadRAMFileToHost** (см. стр. 589 ) () или **TFTPUploadFragmentedRAMFileToHost** (см. стр. 588 ) ( ) функций.

**Замечания**

Клиентский модуль DNS должен быть включен для использования этой функции . т.е. **STACK_USE_DNS** должны быть определены в **TCPIPConfig**.h .

**Предпосылки**

Ни один

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

**10.20.1.19 TFTPUploadFragmentedRAMFileToHost Функция**

**файл**

TFTPc.h

C

```
void TFTPUploadFragmentedRAMFileToHost (
ROM BYTE * vRemoteHost ,
ROM BYTE * vFilename ,
TFTP_CHUNK_DESCRIPTOR * vFirstChunkDescriptor
) ;
```

**Возвращает**

Ни один

**описание**

Передает в случайное , потенциально не смежные , массив байтов RAM в виде файла на удаленный сервер TFTP.

**Замечания**

Клиентский модуль DNS должен быть включен для использования этой функции . т.е. **STACK_USE_DNS** должны быть определены в **TCPIPConfig**.h .

Вызовите **TFTPGetUploadStatus** (см. стр. 587 ) () для определения состояния файловой загрузки.

Это возможно только иметь одну операцию TFTP активного в данный момент времени . После начала операции TFTP по телефону **TFTPUploadRAMFileToHost** (см. стр. 589 ) ( ) или **TFTPUploadFragmentedRAMFileToHost** ( ) , вы не должны ждать, пока **TFTPGetUploadStatus** (см. стр. 587 ) () возвращает код состояния завершения ( <= 0) до вызова любой другой API TFTP функции.

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**vRemoteHost** ROM строка удаленного сервера TFTP для загрузки на (например: " www.myserver.com " ).

Для архитектур устройств, которые не делают различия между ОЗУ и ПЗУ указатели ( PIC24 , DSPIC и PIC32 ) , эта строка должна оставаться выделено и немодифицированные в оперативной памяти , пока процесс загрузки TFTP не завершит (как указано **TFTPGetUploadStatus** (см. стр. 587 ) ()).

**vFilename** ROM строка удаленного файла для создания / перезаписи (например: " status.txt " ) . Для устройства архитектуры, которые не делают различия между ОЗУ и ПЗУ указателей ( PIC24 , КЛУБА и PIC32 ) , эта строка должна оставаться выделено и немодифицированного в оперативной памяти пока процесс загрузки TFTP не завершит (как указано **TFTPGetUploadStatus** (см. стр. 587 ) ()).

**vFirstChunkDescriptor** Указатель на статической или глобальной ( постоянного ) массив **TFTP_CHUNK_DESCRIPTOR** ( См. стр. 589) структуры описания того, что оперативной памяти адреса файл Содержание должно быть получено от .

**TFTP_CHUNK_DESCRIPTOR.vDataPointer** поле должно быть установлено в память адрес данных для передачи , а также **TFTP_CHUNK_DESCRIPTOR.wDataLength** поле должно

быть установлено на номер байтов для передачи из данного указателя. **TFTP_CHUNK_DESCRIPTOR** (См. стр. 589 ) Массив должен заканчиваться фиктивным дескрипторов Кто есть **TFTP_CHUNK_DESCRIPTOR.vDataPointer** указатель установлен в NULL. Обратитесь к **TFTPUploadRAMFileToHost** (см. стр. 589 ) () API для примера вызова Последовательность поскольку он лишь оберткой к этому Функция **TFTPUploadFragmentedRAMFileToHost ( )** .

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

**10.20.1.20 TFTPUploadRAMFileToHost Функция**

**файл**

TFTPc.h

C

```
void TFTPUploadRAMFileToHost (
ROM BYTE * vRemoteHost ,
ROM BYTE * vFilename ,
BYTE * Vdata ,
WORD wDataLength
) ;
```

**Возвращает**

Ни один

**описание**

Передает непрерывный массив байт RAM в виде файла на удаленный сервер TFTP.

**Замечания**

Клиентский модуль DNS должен быть включен для использования этой функции . т.е. **STACK_USE_DNS** должны быть определены в **TCPIPConfig**.h .

Вызовите **TFTPGetUploadStatus** (см. стр. 587 ) () для определения состояния файловой загрузки.

Это возможно только иметь одну операцию TFTP активного в данный момент времени . После начала операции TFTP по телефону **TFTPUploadRAMFileToHost** () или **TFTPUploadFragmentedRAMFileToHost** (см. стр. 588 ) () , вы не должны ждать, пока

**TFTPGetUploadStatus** (см. стр. 587 ) () возвращает код состояния завершения ( <= 0) до вызова любой другой API TFTP функции.

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**vRemoteHost** ROM строка удаленного сервера TFTP для загрузки на (например: " www.myserver.com " ).

Для архитектур устройств, которые не делают различия между ОЗУ и ПЗУ указатели ( PIC24 , DSPIC и PIC32 ) , эта строка должна оставаться выделено и немодифицированные в оперативной памяти , пока процесс загрузки TFTP не завершит (как указано **TFTPGetUploadStatus** (см. стр. 587 ) ()).

**vFilename** ROM строка удаленного файла для создания / перезаписи (например: " status.txt " ) . Для устройства архитектуры, которые не делают различия между ОЗУ и ПЗУ указателей ( PIC24 ,КЛУБА и PIC32 ) , эта строка должна оставаться выделено и немодифицированного в оперативной памяти пока процесс загрузки TFTP не завершит (как указано **TFTPGetUploadStatus** (см. стр. 587 ) ()).

**Vdata** Указатель на RAM массив данных для записи в файл .

**wDataLength** Количество байт , на который указывает Vdata . Это будет окончательный размер файла Загруженный файл . Обратите внимание, что , поскольку он определяется как тип WORD , максимума возможно размер файла 65535 байт . Для более длинных файлов , звоните

**TFTPUploadFragmentedRAMFileToHost** (см. стр. 588 ) функция ( ) вместо этого.

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

**10.20.1.21 TFTP_CHUNK_DESCRIPTOR Структура**

**файл**

TFTPc.h

C

```
typedef struct {

BYTE * vDataPointer ;

WORD wDataLength ;
```

} TFTP_CHUNK_DESCRIPTOR ;

**описание**

Это тип TFTP_CHUNK_DESCRIPTOR

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

**10.20.1.22 TFTP_UPLOAD_COMPLETE Макро**

**файл**

TFTPc.h

C

#define TFTP_UPLOAD_COMPLETE 0

**описание**

Коды состояния для TFTPGetUploadStatus (см. стр. 587 ) () функцию . Ноль означает, что обновление успех , > 0 означает работать и

< 0 означает фатальную ошибку.

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

**10.20.1.23 TFTP_UPLOAD_CONNECT Макро**

**файл**

TFTPc.h

C

#define TFTP_UPLOAD_CONNECT 3

**описание**

Это макро TFTP_UPLOAD_CONNECT .

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

**10.20.1.24 TFTP_UPLOAD_CONNECT_TIMEOUT Макро**

**файл**

TFTPc.h

C

#define TFTP_UPLOAD_CONNECT_TIMEOUT -2

**описание**

Это макро TFTP_UPLOAD_CONNECT_TIMEOUT .

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

**10.20.1.25 TFTP_UPLOAD_GET_DNS Макро**

**файл**

TFTPc.h

C

#define TFTP_UPLOAD_GET_DNS 1

**описание**

Это макро TFTP_UPLOAD_GET_DNS .

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

**10.20.1.26 TFTP_UPLOAD_HOST_RESOLVE_TIMEOUT Макро**

**файл**

TFTPc.h

C

#define TFTP_UPLOAD_HOST_RESOLVE_TIMEOUT -1

**описание**

Это макро TFTP_UPLOAD_HOST_RESOLVE_TIMEOUT .

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

**10.20.1.27 TFTP_UPLOAD_RESOLVE_HOST Макро**

**файл**

TFTPc.h

C

```
#define TFTP_UPLOAD_RESOLVE_HOST 2
```

**описание**

Это макро TFTP_UPLOAD_RESOLVE_HOST .

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

**10.20.1.28 TFTP_UPLOAD_SEND_DATA Макро**

**файл**

TFTPc.h

C

```
#define TFTP_UPLOAD_SEND_DATA 5
```

**описание**

Это макро TFTP_UPLOAD_SEND_DATA .

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

**10.20.1.29 TFTP_UPLOAD_SEND_FILENAME Макро**

**файл**

TFTPc.h

C

```
#define TFTP_UPLOAD_SEND_FILENAME 4
```

**описание**

Это макро TFTP_UPLOAD_SEND_FILENAME .

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

**10.20.1.30 TFTP_UPLOAD_SERVER_ERROR Макро**

**файл**

TFTPc.h

C

#define TFTP_UPLOAD_SERVER_ERROR -3

**описание**

Это макро TFTP_UPLOAD_SERVER_ERROR .

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

**10.20.1.31 TFTP_UPLOAD_WAIT_FOR_CLOSURE Макро**

**файл**

TFTPc.h

C

#define TFTP_UPLOAD_WAIT_FOR_CLOSURE 6

**описание**

Это макро TFTP_UPLOAD_WAIT_FOR_CLOSURE .

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

**10.20.2 TFTP стека Пользователи**

**макрос**

**Имя Описание**

**TFTP_ARP_TIMEOUT_VAL**(См. Стр. 592 ) Количество секунд ожидания перед объявлением ошибку перерыва на Put

**TFTP_GET_TIMEOUT_VAL**(См. Стр. 593 ) Количество секунд ожидания перед объявлением ошибку перерыва на Get .

**TFTP_MAX_RETRIES (**см. Стр. 593 ) Количество попыток до объявления об ошибки тайм-аут.

**модуль**

**TFTP** (см. стр. 576 )

**описание**

Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны, как правило, не называют эти функции или модифицировать эти переменные .

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

### 10.20.2.1 TFTP_ARP_TIMEOUT_VAL Макро

**файл**

TFTPc.h

C

```
#define TFTP_ARP_TIMEOUT_VAL ( 3U * TICKS_PER_SECOND )
```

**описание**

Количество секунд ожидания перед объявлением ошибку перерыва на Put

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

### 10.20.2.2 TFTP_GET_TIMEOUT_VAL Макро

**файл**

TFTPc.h

C

```
#define TFTP_GET_TIMEOUT_VAL ( 3U * TICKS_PER_SECOND )
```

**описание**

Количество секунд ожидания перед объявлением ошибку перерыва на Get .

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

**10.20.2.3 TFTP_MAX_RETRIES Макро**

**файл**

TFTPc.h

C

group2;TFTP_MAX_RETRIES (3U)

**описание**

Количество попыток до объявления об ошибки тайм-аут.

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

**10.20.3 TFTP Внутренняя Пользователи**

**Перечисления**

**Имя Описание**

**TFTP_OPCODE** (см. стр.595 ) Перечень кодов операций TFTP

**TFTP_STATE** (см. стр.596) Государственная машина TFTP

**Функции**

**Имя Описание**

**_TFTPSendAck** (См. Стр.597) Личное вспомогательная функция

**_TFTPSendFileName** (См.страница 597) Личное вспомогательная функция

**_TFTPSendROMFileName**(См. Стр. 598 ) PIC18 ROM реализация переменной аргумент **_TFTPSendFileName** (см.страница 597)

**макрос**

**Имя Описание**

**TFTP_BLOCK_SIZE** (см.страница 595 ) Размер блока TFTP - 512 байт

**TFTP_BLOCK_SIZE_MSB**(См. Стр. 595 ) Старший бит TFTP_BLOCK_SIZE (см. стр. 595 )

**TFTP_CLIENT_PORT** (см. Стр. 595 ) Клиент TFTP порт - уникальный порт на этом устройстве

**TFTP_SERVER_PORT** (см. Стр. 596 ) TFTP-сервер Порт

**модуль**

**TFTP** (см. стр. 576 )

**Переменные**

**Имя Описание**

**MutExVar** (см. стр. 594 ) взаимоисключающими группы переменных для сохранения памяти.

**_tftpError** ( см. стр. 596 ) переменной для сохранения состояние ошибки вызывает для последующей передачи

**_tftpFlags** (см. стр. 596 ) Флаги состояния TFTP

**_tftpRetries** (см. стр. 597 ) переменная трекер для числа попыток TFTP

**_tftpSocket** ( см. стр. 598 ) TFTP Разъем для сервера TFTP ссылку

**_tftpStartTick** (см. стр. 598 ) Ремень переменная используется для обнаружения условия тайм-аута

**_tftpState** (см. стр. 598 ) TFTP конечный автомат, трекер

**smUpload** (см. стр. 598 ) Это переменная smUpload .

**uploadChunkDescriptor** (см. стр.599) Это переменная **uploadChunkDescriptor** .

**uploadChunkDescriptorForRetransmit**(См. Стр. 599 ) Это переменная **uploadChunkDescriptorForRetransmit** .

**vUploadFilename** (см. стр. 599 ) Это переменная **vUploadFilename** .

**vUploadRemoteHost** (см. стр.599)

**TFTPUploadRAMFileToHost** (см. стр. 589 ) ( ) ,

**TFTPUploadFragmentedRAMFileToHost** (см. стр. 588 ) () и **TFTPGetUploadStatus** (см. стр. 587 ) ( ) функции требуют DNS

Клиентский модуль будет включен , чтобы они работали . ОЗУ и ПЗУ ресурсы для этих функций могут быть сохранены , если клиент DNS Модуль не включен .

**wUploadChunkOffset** (см. стр.600) Это переменная **wUploadChunkOffset** .

**wUploadChunkOffsetForRetransmit**(См. Стр. 600 ) Это переменная **wUploadChunkOffsetForRetransmit** .

**описание**

Следующие функции и переменные определяются как внутренние, к модулю TFTP.

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

### 10.20.3.1 MutExVar Переменная

**файл**

TFTPc.c

C

```
union {
  struct {
NODE_INFO _hostInfo ;
}  group1;
 struct {
WORD_VAL _tftpBlockNumber ;
WORD_VAL _tftpDuplicateBlock ;
WORD_VAL _tftpBlockLength ;
} group2;
} MutExVar ;
```

**описание**

Взаимоисключающими группы переменных по сохранению памяти .

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

### 10.20.3.2 TFTP_BLOCK_SIZE Макро

**файл**

TFTPc.c

C

#define TFTP_BLOCK_SIZE ( 0x200L )

**описание**

Размер блока TFTP - 512 байт

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

**10.20.3.3 TFTP_BLOCK_SIZE_MSB Макро**

**файл**

TFTPc.c

C

#define TFTP_BLOCK_SIZE_MSB ( 0x02u )

**описание**

Старший бит TFTP_BLOCK_SIZE (см. стр. 595 )

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

**10.20.3.4 TFTP_CLIENT_PORT Макро**

**файл**

TFTPc.c

C

#define TFTP_CLIENT_PORT 65352L

**описание**

Клиент TFTP порт - уникальный порт на этом устройстве

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

### 10.20.3.5 TFTP_OPCODE Перечисление

**файл**

TFTPc.c

C

```
typedef enum {
TFTP_OPCODE_RRQ = 1,
TFTP_OPCODE_WRQ ,
TFTP_OPCODE_DATA ,
TFTP_OPCODE_ACK ,
TFTP_OPCODE_ERROR
} TFTP_OPCODE ;
```

**Пользователи**

**Пользователи Описание**

TFTP_OPCODE_RRQ = 1 Get

TFTP_OPCODE_WRQ Положите

TFTP_OPCODE_DATA фактических данных

TFTP_OPCODE_ACK Ack для GET / Put

TFTP_OPCODE_ERROR Ошибка

**описание**

Перечень кодов операций TFTP

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

### 10.20.3.6 TFTP_SERVER_PORT Макро

**файл**

TFTPc.c

C

```
#define TFTP_SERVER_PORT ( 69L )
```

**описание**

TFTP-сервер Порт

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

**10.20.3.7 TFTP_STATE Перечисление**

**файл**

TFTPc.c

C

```
typedef enum {
SM_TFTP_WAIT = 0,
SM_TFTP_READY ,
SM_TFTP_WAIT_FOR_DATA ,
SM_TFTP_WAIT_FOR_ACK ,
SM_TFTP_DUPLICATE_ACK ,
SM_TFTP_SEND_ACK ,
SM_TFTP_SEND_LAST_ACK
} TFTP_STATE ;
```

**описание**

Государственная машина TFTP

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

**10.20.3.8 _tftpError Переменная**

**файл**

TFTPc.c

C

```
СЛОВО _tftpError ;
```

**описание**

Переменная сохранить состояние ошибки причин для последующей передачи

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

### 10.20.3.9 _tftpFlags Переменная

**файл**

TFTPc.c

C

```
union {
  struct {
    unsigned int bIsFlushed : 1;
    unsigned int bIsAcked : 1;
    unsigned int bIsClosed : 1;
    unsigned int bIsClosing : 1;
    unsigned int bIsReading : 1;
  } bits;
  BYTE Val;
} _tftpFlags;
```

**описание**

Флаги состояния TFTP

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

### 10.20.3.10 _tftpRetries Переменная

**файл**

TFTPc.c

C

BYTE _tftpRetries ;

**описание**

Переменная трекер для числа попыток TFTP

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

**10.20.3.11 _TFTPSendAck Функция**

**файл**

TFTPc.c

C

```
static void _TFTPSendAck(
    WORD_VAL blockNumber
);
```

**описание**

Личное вспомогательная функция

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

**10.20.3.12 _TFTPSendFileName Функция**

**файл**

TFTPc.c

C

```
static void _TFTPSendFileName(
    TFTP_OPCODE command,
    BYTE * fileName
);
```

**описание**

Личное вспомогательная функция

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

### 10.20.3.13 _TFTPSendROMFileName Функция

**файл**

TFTPc.c

C

```
static void _TFTPSendROMFileName(
    TFTP_OPCODE opcode,
    ROM BYTE * fileName
);
```

**описание**

PIC18 ROM реализация переменной аргумент **_TFTPSendFileName** (см. стр. 597 )

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

### 10.20.3.14 _tftpSocket Переменная

**файл**

TFTPc.c

C

```
UDP_SOCKET _tftpSocket ;
```

**описание**

TFTP Разъем для сервера TFTP ссылку

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

### 10.20.3.15 _tftpStartTick Переменная

**файл**

TFTPc.c

C

DWORD _tftpStartTick ;

**описание**

Ремень переменная используется для обнаружения условия тайм-аута

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

**10.20.3.16 _tftpState Переменная**

**файл**

TFTPc.c

C

TFTP_STATE **_tftpState** ;

**описание**

Конечный автомат, трекер TFTP

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

**10.20.3.17 smUpload Переменная**

**файл**

TFTPc.c

C

CHAR smUpload = TFTP_UPLOAD_COMPLETE ;

**описание**

Это переменная **smUpload** .

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

**10.20.3.18 uploadChunkDescriptor Переменная**

**файл**

TFTPc.c

C

TFTP_CHUNK_DESCRIPTOR * **uploadChunkDescriptor** ;

**описание**

Это переменная **uploadChunkDescriptor** .

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

**10.20.3.19 uploadChunkDescriptorForRetransmit Переменная**

**файл**

TFTPc.c

C

TFTP_CHUNK_DESCRIPTOR * **uploadChunkDescriptorForRetransmit** ;

**описание**

Это переменная **uploadChunkDescriptorForRetransmit** .

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

**10.20.3.20 vUploadFilename Переменная**

**файл**

TFTPc.c

C

ROM BYTE * **vUploadFilename** ;

**описание**

Это переменная **vUploadFilename** .

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

**10.20.3.21 vUploadRemoteHost Переменная**

**файл**

TFTPc.c

C

ROM BYTE * **vUploadRemoteHost** ;

**описание**

TFTPUploadRAMFileToHost (см. стр. 589 ) ( ) , TFTPUploadFragmentedRAMFileToHost (см. стр. 588 ) () и TFTPGetUploadStatus (см. стр. 587 ) ( ) функции требуютклиентский модуль DNS должна быть включена , чтобы они работали .

RAM и ROM ресурсы для этих функций могут быть сохранены , если клиентский модуль DNS не включен .

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

**10.20.3.22 wUploadChunkOffset Переменная**

**файл**

TFTPc.c

C

WORD **wUploadChunkOffset** ;

**описание**

Это переменная **wUploadChunkOffset** .

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

**10.20.3.23 wUploadChunkOffsetForRetransmit Переменная**

**файл**

TFTPc.c

C

WORD **wUploadChunkOffsetForRetransmit** ;

**описание**

Это переменная **wUploadChunkOffsetForRetransmit** .

## 10.21 Tick Module

Модуль Tick обеспечивает точные возможности хранения данных о времени , основанные на аппаратных часов . По умолчанию , он использует таймер 0 на 8-битные части и Таймер 1 на 16 - и 32-разрядные семьи . Модуль прерываний приводом , что делает выбор времени стабильной и точны. Таким образом, он также подходит для часов реального времени .

Модуль Tick существует для оказания помощи в осуществлении задержек и не блокирующих тайм-аутов . Вместо того чтобы использовать цикл для рассчитывать на определенный номер, использовать модуль Tick и сравните предыдущий раз с текущим временем . Таким способом приложений может вернуть ее неиспользуемые циклы в стек в течение длительных задержек , что повышает общую эффективность системы .

Тик лучше всего работает в сочетании с государственной машиной . В общем , звоните TickGet (см. стр. 603 ) и сохранить результат . Вернуться на Основное применение стека , и на будущих вызовов сравнить текущее значение Tick хранимой один . константы **TICK_SECOND** (см. стр. 602 ) , **TICK_MINUTE** (см. стр. 602 ), а **TICK_HOUR** (см. стр. 602 ) может быть использован для сравнения с шагом логических времени .

В следующем примере реализуется задержку 0,5 секунды , используя модуль Tick :

```
TICK startTime;

/ / ... Государственная машина и другие государства

case SM_SET_DELAY:
startTime = TickGet();
sm = SM_DELAY_WAIT;
return;
case SM_DELAY_WAIT:
if((LONG)(TickGet() - startTime) < TICK_SECOND/2)
return;
case SM_DELAY_DONE:
/ / Это состояние вводится только после 0,5 секунды истечет .
```

Клещи хранятся в виде 48 - разрядных целых чисел. Используя различные **TickGet** (см. стр. 603 ), **TickGetDiv256** (см. стр. 603 ) , и **TickGetDiv64K** (см. стр. 604 ) функционирует Tick подходит для измерения приращения времени от нескольких микросекунд до нескольких лет .

Если абсолютные отметки времени требуется , модуль клиент SNTP может быть более подходящим .

### 10.21.1 Тик Открытый Пользователи

**Функции**

**Имя Описание**

**TickConvertToMilliseconds** (См. стр. 602 ) Преобразует значение Tick или разницу в миллисекундах.

**TickGet** (см. стр. 603 ) Получает текущее значение Tick .

**TickGetDiv256** (см. стр.603) Получает текущее значение Tick , деленное на 256.

**TickGetDiv64K** (см. стр.604) Получает текущее значение Tick , деленной на 64K.

**макрос**

**Имя Описание**

**TICK_HOUR** (см. стр.602) Представляет один час в клещи

**TICK_MINUTE** (см. стр.602) Представляет одну минуту в клещах

**TICK_SECOND** (см. стр.602) Представляет одну секунду в клещах

**модуль**

**Tick Module** ( см. стр. 600 )

**Виды**

**Имя Описание**

**TICK** (см. стр. 601 ) Все КЛЕЩИ хранятся в виде 32-разрядных целых чисел без знака . Это не рекомендуется , так как это конфликты с другими определениями галочку , используемых в других программных библиотек Microchip и, следовательно, представляет собой слияние и проблемы Maintence . Вместо использования Тип данных TICK , просто использовать тип данных базы DWORD вместо этого.

**описание**

Следующие функции и переменные доступны для применения стека.

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

#### 10.21.1.1 TICK Тип

**файл**

Tick.h

C

**typedef** DWORD **TICK**;

**описание**

Все КЛЕЩИ хранятся в виде 32-разрядных целых чисел без знака . Это не рекомендуется , так как это противоречит с другими определениями галочку , используемых в другие библиотеки  Microchip программного обеспечения и, следовательно, представляет собой слияние и проблемы Maintence . Вместо того чтобы использовать тип данных Tick, просто использовать тип данных базы DWORD вместо этого.

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

### 10.21.1.2 TICK_HOUR Макро

файл

Tick.h

C

**#define TICK_HOUR** ((QWORD)TICKS_PER_SECOND*3600ull)

описание

Представляет один час в Ticks

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

### 10.21.1.3 TICK_MINUTE Макро

файл

Tick.h

C

#define TICK_MINUTE ((QWORD)TICKS_PER_SECOND*60ull)

описание

Представляет одну минуту в Ticks

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

### 10.21.1.4 TICK_SECOND Макро

файл

Tick.h

C

```
#define TICK_SECOND ((QWORD)TICKS_PER_SECOND)
```

описание

Представляет одну секунду в Ticks

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

### 10.21.1.5 TickConvertToMilliseconds Функция

файл

Tick.h

C

```
DWORD TickConvertToMilliseconds(
DWORD dwTickValue
);
```

**Возвращает**

Входное значение выражается в миллисекундах .

**описание**

Эта функция преобразует значение Tick или разницу в миллисекундах. Например, **TickConvertToMilliseconds** ( 32768 ) возвращает 1000 , когда32.768kHz часы , не делителя управляет модуль Тик прерывание.

**Замечания**

Эта функция выполняет деление на DWORDS , которая идет медленно. Старайтесь не использовать его без крайней необходимости ( например, для отображения данных на пользователя). Для сравнения тайм-аута , сравнить текущее значение к величине, кратной или фракции TICK_SECOND (см. стр.602 ), который будет рассчитываться только один раз во время компиляции.

**Предпосылки**

Ни один

**Параметры**

**Параметры Описание**

**dwTickValue** Значение преобразовать в миллисекундах

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

### 10.21.1.6 TickGet Функция

**файл**

Tick.h

**C**

DWORD TickGet ( ) ;

**Возвращает**

Нижние 32 бита текущего значения Tick .

**описание**

Эта функция возвращает текущее значение Tick , позволяя времени и измерение код записать в неблокирующей моды.

Эта функция получает младшие 32 бита внутреннего Tick счетчика , и используется для измерения приращения времени от нескольких микросекунд до нескольких часов . Используйте **TickGetDiv256** (см. стр. 603 ) или **TickGetDiv64K** (см. стр. 604 ) в течение более длительных периодов времени.

**Предпосылки**

Ни один

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

### 10.21.1.7 TickGetDiv256 Функция

**файл**

Tick.h

**C**

DWORD TickGetDiv256 ( ) ;

**Возвращает**

Средние 32 бита текущего значения Tick .

**описание**

Эта функция возвращает текущее значение Tick , позволяя времени и измерение код записать в неблокирующей моды.

Эта функция получает средние 32 бита внутреннего клеща счетчика , и используется для измерения приращения времени , начиная от нескольких минут до нескольких недель . Используйте **TickGet** (см. стр. 603 ) на более короткие периоды или **TickGetDiv64K** (см. стр. 604 )для более длинных.

**Предпосылки**

Ни один

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

### 10.21.1.8 TickGetDiv64K Функция

**файл**

Tick.h

**C**

DWORD TickGetDiv64K ( ) ;

**Возвращает**

Верхние 32 бита текущего значения Tick .

**описание**

Эта функция возвращает текущее значение Tick , позволяя времени и измерение код записать в неблокирующей моды.

Эта функция получает наиболее значимые 32 бита внутреннего клеща счетчика , и используется для измерения приращения времени от нескольких дней до нескольких лет , или для измерения абсолютного времени . Использование **TickGet** (см. стр. 603 ) или **TickGetDiv256** (см. стр. 603 ) на более короткие периоды времени.

**Предпосылки**

Ни один

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

### 10.21.2 Тик Stack Функции

**Функции**

**Имя Описание**

**TickInit** (см. стр. 604) Инициализирует модуль менеджера Tick .

**TickUpdate** (см. стр. 605 ) Обновляет значение галочку при возникновении прерывания .

**модуль**

Тик модуль ( см. стр. 600 )

**описание**

Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны как правило, не называют эти функции или модифицировать эти переменные .

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

#### 10.21.2.1 TickInit Функция

**файл**

Tick.h

**C**

```
void TickInit();
```

**Возвращает**

Ни один

**описание**

Настраивает модуль Tick и все необходимые аппаратные ресурсы .

Замечания

Эта функция вызывается только один при жизни приложения.

**Предпосылки**

Ни один

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

**10.21.2.2 TickUpdate Функция**

**файл**

Tick.h

**C**

**void TickUpdate**();

**Возвращает**

Ни один

**описание**

Обновляет значение галочку при возникновении прерывания .

**Предпосылки**

Ни один

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

**10.21.3 Tick Внутренние Пользователи**

**Функции**

**Имя Описание**

**GetTickCopy** (см. стр.606) Считывает значение галочку .

**макрос**

**Имя Описание**

**TICKS_PER_SECOND** (см. стр. 606 ) Внутренняя тактовая частота приводит таймер с 1:256 Prescaler # определить **TICKS_PER_SECOND** ( 32768ul ) / / 32 кГц кристалл приводит таймер , не скалярного

**модуль**

Тик модуль ( см. стр. 600 )

**Переменные**

**Имя Описание**

**dwInternalTicks** (см. стр.605)

Внутренний счетчик для хранения клещей . Эта переменная увеличивается в ISR и поэтому должны быть отмечены летучих предотвратить компилятора оптимизатору переназначения код, чтобы использовать это значение в контексте главного в то время прерывания отключена.

**vTickReading** (см. стр.606)

Значение 6 байт для хранения клещей . Позволяет для использования в течение длительных периодов времени.

**описание**

Следующие функции и переменные определяются как внутренние, к модулю Tick.

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

### 10.21.3.1 dwInternalTicks Переменная

**файл**

Tick.c

**C**

```
volatile DWORD dwInternalTicks = 0;
```

**описание**

Внутренний счетчик для хранения клещей . Эта переменная увеличивается в ISR и, следовательно, должны быть отмечены летучих чтобы предотвратить компилятор оптимизатор от переназначения код, чтобы использовать это значение в контексте главного в то время прерывания отключены .

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

### 10.21.3.2 GetTickCopy Функция

**файл**

Tick.c

**C**

```
static void GetTickCopy();
```

**Возвращает**

Ни один

**описание**

Эта функция выполняет прерывания безопасного и синхронизированное чтения значения Tick 48 -битной .

**Предпосылки**

Ни один

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

**10.21.3.3 TICKS_PER_SECOND Макро**

**файл**

Tick.h

**C**

```
#define TICKS_PER_SECOND ((GetPeripheralClock()+128ull)/256ull)// Внутренний частота ядра диски таймер с 1:256 делителя
```

**описание**

Внутренняя тактовая частота приводит таймер с 1:256 предварительного делителя

#define **TICKS_PER_SECOND** ( 32768ul ) / / 32 кГц кристалл приводит таймер без каких-либо скалярного

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

**10.21.3.4 vTickReading Переменная**

**файл**

Tick.c

**C**

```
BYTE vTickReading [6] ;
```

**описание**

Значение 6 байт для хранения клещей . Позволяет для использования в течение длительных периодов времени.

## 10.22 UDP

**Виды**

**Имя Описание**

**UDP_STATE** (см. стр.627)UDP States

**описание**

UDP является стандартным протоколом транспортного уровня описано в RFC 768 . Он обеспечивает быструю, но ненадежную передачу данных, основанных по сетям и форм основой SNTP , SNMP , DNS, и многие другие стандарты протокола .

Соединения более UDP следует рассматривать как трансферты данных грамм основе . Каждый пакет является отдельным юридическим лицом , приложение следует ожидать некоторые пакеты попадают вне порядка или даже не достигают узла назначения . Это в отличие от TCP, в котором соединение мыслится как поток и сетевые ошибки автоматически корректируются . Эти компромиссы в надежности сделаны для увеличения пропускной способности. В общем, UDP передает работать 2 в 3 раза быстрее , чем те, составил более TCP.

Поскольку UDP является пакетно-ориентированным , каждый пакет должен быть рассмотрен в полном объеме вашим приложением , прежде чем вернуться к основной стек петлю . При получении пакета , ваше приложение будет называться справиться с этим. Этот пакет больше не будет доступна в следующий раз ваше приложение называется , так что вы должны либо выполнить всю необходимую обработку или скопировать данные в другое место, прежде чем возвращения . При передаче пакета , приложение должно строить и передавать полный пакет в одном цикле .

Диаграмма UDP поток ниже приводится обзор для использования UDP модуля :

Розетки ( см. стр. 152 ) открываются с помощью **UDPOpen** (см. стр. 611) . Эта функция может либо открыть сокет для ждать входящих сегментов , или может сделать подключение клиента к удаленному узлу . При выполнении соединения клиента , вы будете необходимо выполнить все требуемые DNS и / или разрешение ARP , используя эти модули непосредственно перед вызовом **UDPOpen** (см.страница 611 ) .

После открытия сокета , вы можете сразу же начать передачу данных. Для передачи сегмента, вызовите **UDPIsPutReady** (см. стр. 615 ), чтобы определить , сколько байт можно записать и для обозначения текущего активного сокета. Затем, используя любой из **UDPPut** ( см. стр. 615) семейство функций для записи данных в сокет. После того как все данные были записаны , вызовите **UDPFlush** (см. стр. 613 ) , чтобы построить и передать пакет . Эта последовательность должна быть выполнена всего за один шаг. Если ваше приложение возвращает в основной цикл стека после вызова **UDPPut** (см. стр. 615 ), но перед вызовом **UDPFlush** (см. стр. 613 ) , данные могут быть потеряны или модуль может вести себя непредсказуемо .

Для проверки полученных сегментов , вызовите **UDPIsGetReady** (см. стр. 614 ) . Если возвращаемое значение не равно нулю , приложение должны потреблять сегмент читая данные с **UDPGet** (см. стр. 613 ) семьи. После того как все данные считаны , возвращение в основной цикл стека ждать дополнительного сегмента. UDP сегменты хранятся только в течение одной итерации кооператива многозадачность цикл , поэтому приложение должно завершить свою обработку на отрезке или скопировать его в другом месте , прежде чем вернуться .

Заметим, что это поведение отличается от TCP , который буферизующей входящие данные через несколько циклов стека.

Когда сокет больше не нужен , вызовите **UDPClose** (см. стр. 612) , чтобы освободить ее обратно в бассейн для использования в будущем .

### 10.22.1 UDP Общественная Пользователи

**Функции**

**Имя Описание**

**UDPOpenEx** (см. стр.610) Открывает UDP сокет для клиента.

**UDPClose** (см. стр. 612) Закрывает UDP сокет и освобождает ручку .

**UDPDiscard** (см. стр.612) Отмена всех оставшихся RX данные из сокета UDP .

**UDPFlush** (см. стр. 613 ) Передает все отложенные данные в UDP сокет.

**UDPGet** (см. стр. 613 ) Считывает байт из текущего активного сокета.

**UDPGetArray** (см. стр.614 ) Читает массив байтов от активного в данный момент гнездо .

**UDPIsGetReady** (см. стр.614 ) Определяет, сколько байт можно прочитать из розетки UDP .

**UDPIsPutReady** (см. стр.615) Определяет, сколько байт можно записать в сокет UDP .

**UDPPut** (см. стр. 615 ) Записывает байт в активной в данный момент гнездо .

**UDPPutArray** (см. стр.616) Записывает массив байтов с активным розетки.

**UDPPutROMArray** (см.страница 616) Записывает массив байтов из ПЗУ в активной в данный момент гнездо .

**UDPPutROMString** (см.страница 617) Записывает нулем строку из ПЗУ в активной в данный момент гнездо .

**UDPPutString** (см. стр.617) Записывает нулем строку с активным розетки.

**UDPSetRxBuffer** (см. страница 618) Перемещение указателя в буфере RX.

**UDPSetTxBuffer** (см. стр.618) Перемещение указателя в буфере TX.

**UDPIsOpened** (см. стр.619) Определяет, является ли сокет установил соединение .

**макрос**

**Имя Описание**

**INVALID_UDP_PORT** (см.страница 609) Указывает на UDP порт , который недопустим

I**NVALID_UDP_SOCKET** (см. Стр. 609) Указывает на UDP сокет , который недопустим

**UDPOpen (** см. стр. 611 ) Макрос на устаревшей версии **UDPOpen** .

**UDP_OPEN_IP_ADDRESS** (См. Стр. 619 ) Создание клиентского сокета и использовать **dwRemoteHost** как буквальное IP адресу.

**UDP_OPEN_NODE_INFO** (См. Стр. 619 ) Создание клиентского сокета и использовать **dwRemoteHost** как указатель на **NODE_INFO** структура, содержащая точную удаленный IP -адрес и MAC- адрес, используемый в .

**UDP_OPEN_RAM_HOST** (см. Стр. 620) Выделяют необъявленную идентификатор диагностическую если код пытается использовать

**UDP_OPEN_RAM_HOST** в то время как клиентский модуль DNS не включен .

**UDP_OPEN_ROM_HOST** (см. Стр. 620) Выделяют не объявленную идентификатор диагностическую если код пытается использовать

**UDP_OPEN_ROM_HOST** в то время как клиентский модуль DNS не включен .

**UDP_OPEN_SERVER** (см. Стр. 620) Создание серверного сокета и игнорировать **dwRemoteHost** .

**модуль**

**UDP** (см. стр. 607 )

**Виды**

**Имя Описание**

**UDP_SOCKET** (см. стр.609)

Обеспечивает дескриптор сокета UDP

**описание**

Следующие функции и переменные доступны для применения стека.

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

### 10.22.1.1 INVALID_UDP_PORT Макро

**файл**

UDP.h

C

**#define INVALID_UDP_PORT** (0ul) / / Указывает на UDP порт , который недопустим

**описание**

Указывает на UDP порт , который недопустим

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

### 10.22.1.2 INVALID_UDP_SOCKET Макро

**файл**

UDP.h

C

**#define INVALID_UDP_SOCKET** (0xffu) / / Указывает на UDP сокет , который недопустим

**описание**

Указывает на UDP сокет , который недопустим

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

### 10.22.1.3 UDP_SOCKET Тип

**файл**

UDP.h

C

**typedef** BYTE **UDP_SOCKET**;

**описание**

Обеспечивает дескриптор сокета UDP

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

### 10.22.1.4 UDPOpenEx Функция

**файл**

UDP.h

C

```
UDP_SOCKET UDPOpenEx(
DWORD remoteHost,
BYTE remoteHostType,
UDP_PORT localPort,
UDP_PORT remotePort
);
```

**описание**

Предоставляет унифицированный метод для открытия UDP сокеты. Эта функция может открыть и клиент и сервер сокеты. Для клиента розетки, он может принимать (см. стр. 169 ) имени узла строку запроса в DNS, в IP- адрес в виде строки , в IP- адреса в бинарной форме , илиранее решены структура NODE_INFO содержащий удаленный IP -адрес и соответствующему MAC -адрес.

Когда имя узла или IP- адрес только обеспечивается , модуль UDP будет внутренне выполнения необходимой DNSResolve (см.страница 186 ) и / или разрешения ARP шаги , прежде чем сообщить , что UDP сокет подключен ( через вызов **UDPISOpen** возвращение TRUE) . Розетки сервера этот параметр игнорируют назначения и слушать (см. стр. 175) только на указанном порту .

Розетки ( см. стр. 152 ) статически выделяется при загрузке, но можно утверждать, с этой функцией и освобождается с помощью **UDPClose** (См. стр. 612 ) .

**Замечания**

При окончании использования UDP дескриптор сокета , вызовите UDPClose (см. стр. 612 ) функция ( ) , чтобы освободить разъем и удалить ручка .

**Предпосылки**

**UDPInit** (см. стр. 621 ) должна быть вызвана .

**Параметры**

**Параметры Описание**

**RemoteHost** Указатель на удаленном информации узла ( MAC и IP- адрес) для этого соединения . Если это сервер гнездо ( принимает первый пакет ) или назначения является

широковещательным адрес , то этот параметр должен быть NULL . Только для клиентов сокетов. Обеспечивать указатель на строку с нулевым символом удаленного имени хоста (например: " www.microchip.com " или " 192.168.1.123 " ) , буквальный IP адрес назначения (например: 0x7B01A8C0 илитип данных IP_ADDR ) , или указатель на NODE_INFO Структура с удаленным IP -адресом и удаленным узлом или шлюза MAC адрес указан, Если строка предоставляется.

**remoteHostType** Любой из следующих флажков, чтобы определить смысл RemoteHost

параметр :

• **UDP_OPEN_SERVER** (см. стр. 620) = открыть сокет сервера и игнорировать параметр RemoteHost . (например - SNMP агент , DHCP -сервер , Анонс ( см. стр. 155 ) )

• **UDP_OPEN_IP_ADDRESS** (см. стр. 619 ) = Откройте клиентский сокет и подключения (см. стр. 171 ) , что на удаленный IP -адрес. Пример: 0x7B01A8C0 для 192.168.1.123 (тип DWORD) . Обратите внимание, что порядок байтов Big Endian .

• **UDP_OPEN_NODE_INFO** (см. стр. 619 ) = Откройте клиентский сокет и Connect ( см. стр. 171 ) , что на удаленный IP и MAC- адреса пару хранится в структуре NODE_INFO .

• **UDP_OPEN_RAM_HOST** (см. стр. 620) = Откройте клиентский сокет и подключения (см. стр. 171 ) , что на удаленный хост чье имя хранится в виде нулем строку в массиве памяти . Например: " www.microchip.com " или " 192.168.0.123 "

• **UDP_OPEN_ROM_HOST** (см. стр. 620) = Откройте клиентский сокет и подключения (см. стр. 171 ) , что на удаленный хост чье имя хранится в виде нулем строку в буквальном строки или ROM массива.

**Пример**:

" www.microchip.com " или " 192.168.0.123 "

**localPort** UDP номер порта для прослушивания (см. стр. 175 ) на . Если 0 , стек будет динамически присвоить уникальный номер порта для использования.

**RemotePort** Для клиентских сокетов , количество удаленного порта .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**Success** UDP дескриптор сокета , который может использоваться для последующих вызовов API UDP .

**Failure** INVALID_UDP_SOCKET (см. стр. 609 ) . Эта функция не терпит неудачу, когда не более UDP ручки гнездо доступны . Увеличение MAX_UDP_SOCKETS сделать больше розетки доступны.

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

### 10.22.1.5 UDPOpen Макро

**файл**

UDP.h

C

**#define UDPOpen**(localPort,remoteNode,remotePort)
UDPOpenEx((DWORD)remoteNode,UDP_OPEN_NODE_INFO,localPort,remotePort)
**описание**

**UDPOpen** макрос замена реализации наследие **UDPOpen** . Создает UDP дескриптор сокета для transmiting или получение UDP -пакеты. Эта функция вызывается для получения дескриптора требуемую другой функции UDP .

**Замечания**

При окончании использования UDP дескриптор сокета , вызовите UDPClose (см. стр. 612 ) функция ( ) , чтобы освободить разъем и удалить ручка .

**Предпосылки**

**UDPInit** (см. стр. 621 ) ( ) должны быть предварительно называется .

**Параметры**

**Параметры Описание**

**localPort** UDP номер порта для прослушивания (см. стр. 175 ) на . Если 0 , стек будет динамически присвоить уникальный номер порта для использования.

**remoteNode** Указатель на удаленном информации узла ( MAC и IP- адрес) для этого соединения . Если это сервер гнездо ( принимает первый пакет ) или назначения является широковещательным адрес , то этот параметр должен быть NULL .

**RemotePort** Для клиентских сокетов , количество удаленного порта .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**Success** UDP дескриптор сокета , который может использоваться для последующих вызовов API UDP .

**Failure INVALID_UDP_SOCKET** (см. стр. 609 ) . Эта функция не терпит неудачу, когда не более UDP ручки гнездо доступны . Увеличение **MAX_UDP_SOCKETS** сделать больше розетки доступны.

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

### 10.22.1.6 UDPClose Функция

**файл**

UDP.h

C

```
void UDPClose(
UDP_SOCKET s
);
```

**Возвращает**

Ни один

**описание**

**UDP_SOCKET** (см. стр. 609 ) **UDPOpen** (см. стр. 611 ) ( **UDP_PORT** (см. стр. 625 ) **localPort**, **NODE_INFO**

* **remoteNode** , **UDP_PORT** ( см. стр. 625 ) **RemotePort** ) ;

Закрывает сокет UDP и освобождает ручку . Вызовите эту функцию , чтобы освободить разъем и вернуть его в бассейн для использования будущего коммуникации.

**Замечания**

Эта функция не влияет на ранее назначенный активный сокет.

**Предпосылки**

**UDPInit** (см. стр. 621 ) ( ) должны быть предварительно называется .

**Параметры**

**Параметры Описание**

**s** - дескриптор сокета должен быть освобожден. Если предоставляетсянедопустимое значение ручка, Функция безопасного ничего не делает.

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

### 10.22.1.7 UDPDiscard Функция

**файл**

UDP.h

C

```
void UDPDiscard();
```

**Возвращает**

Ни один

**описание**

Эта функция удаляет все оставшиеся данные, полученные в текущем окне UDP сокет.

**Замечания**

Это безопасно для вызова этой функции больше , чем это необходимо . Если данные не доступны , эта функция ничего не делает.

**Предпосылки**

**UDPIsGetReady** (см. стр. 614 ) ( ) был вызван ранее , чтобы выбрать текущий активный разъем.

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

### 10.22.1.8 UDPFlush Функция

**файл**

UDP.h

C

```
void UDPFlush();
```

**Возвращает**

Ни один

**описание**

Эта функция формирует пакет UDP с отложенным данных TX и помечает его для передачи по сетевому интерфейсу . С UDP является протоколом на основе кадров , эта функция должна быть вызвана , прежде чем вернуться в основной цикл стека , когда какие-либо данные написано .

**Замечания**

Обратите внимание, что в отличие от **TCPFlush** (см. стр. 537 ) , **UDPFlush** должен быть вызван , прежде чем вернуться в основной цикл стека. Существует нет авто передачи для UDP сегментов .

**Предпосылки**

**UDPIsPutReady** (см. стр. 615 ) ( ) был вызван ранее для указания текущего сокет, а данные были записаны в Гнездо с помощью **UDPPut** ( см. стр. 615) семейство функций .

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

### 10.22.1.9 UDPGet Функция

**файл**

UDP.h

C

```
BOOL UDPGet(
BYTE * v
);
```

**описание**

Эта функция читает один байт из активного в данный момент UDP сокет , а уменьшая оставшуюся длину буфера.

**UDPIsGetReady** (см. стр. 614 ) следует использовать перед вызовом этой функции , чтобы указать текущий активный разъем.

**Предпосылки**

**UDPIsGetReady** (см. стр. 614 ) ( ) был вызван ранее для указания текущего сокета.

**Параметры**

**Параметры Описание**

**V** - Буфер для получения считываемых данных .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** байт успешно читать

**ЛОЖЬ** Данные не остались в буфере чтения

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

### 10.22.1.10 UDPGetArray Функция

**файл**

UDP.h

C

WORD **UDPGetArray**(
BYTE * **cData**,
WORD **wDataLen**
);

**Возвращает**

Количество байт, успешно читаются из буфера UDP . Если это значение меньше **wDataLen** , то буфер сливают и не более того данные не доступны .

**описание**

Эта функция читает массив байтов из активного в данный момент UDP сокет , а уменьшая оставшиеся байты доступны . **UDPIsGetReady** (см. стр. 614 ) следует использовать перед вызовом этой функции , чтобы указать текущий активный разъем.

**Предпосылки**

**UDPIsGetReady** (см. стр. 614 ) ( ) был вызван ранее для указания текущего сокета.

**Параметры**

**Параметры Описание**

**cDATA-** буфер для принятия байты время чтения . Если NULL , байты просто отбрасываются без написано нигде ( эффективно пропускает байтов в RX буфера , хотя , если вам нужно , чтобы пропустить большой объем данных , ищет помощи

**UDPSetRxBuffer** (см. стр. 618 ) ( ) будет более эффективным ) .

**wDateLen-** Количество байт для чтения из розетки.

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

### 10.22.1.11 UDPIsGetReady Function

**File**
UDP.h
**C**
WORD **UDPIsGetReady**(
UDP_SOCKET **s**
);

**Возвращает**

Число байтов , которые можно прочитать с этого сокета .

**описание**

Эта функция определяет, будет ли байт можно прочитать из указанного сокета UDP . Он также готовит UDP модуль для чтения по установив указанную сокет активного в данный момент подключения.

**Предпосылки**

**UDPInit** (см. стр. 621 ) ( ) должны быть предварительно называется .

**Параметры**

**Параметры Описание**

**s** - сокета является активным ( который уже был открыт или слушает )

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

### 10.22.1.12 UDPIsPutReady Функция

**файл**

UDP.h

C

**C**
WORD **UDPIsPutReady**(
UDP_SOCKET **s**
);

**Возвращает**

Число байтов , которые могут быть записаны к этому socket .

**описание**

Эта функция определяет, байт может быть записан с указанным сокетом UDP . Он также готовит UDP модуль для записи на установив указанную сокет активного в данный момент подключения.

**Предпосылки**

**UDPInit** (см. стр. 621 ) ( ) должны быть предварительно называется .

**Параметры**

**Параметры Описание**

**s-** socket Разъем быть сделаны активными

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

**10.22.1.13 UDPPut Функция**

**файл**

UDP.h

C

BOOL **UDPPut**(
BYTE **v**
);

**описание**

Эта функция записывает один байт на текущий активный UDP сокет , а увеличивая длину буфера . **UDPIsPutReady** (См. стр. 615) следует использовать перед вызовом этой функции , чтобы указать текущий активный разъем.

**Предпосылки**

**UDPIsPutReady** (см. стр. 615 ) ( ) был вызван ранее для указания текущего сокета.

**Параметры**

**Параметры Описание**

**v -** Байт должен быть загружен в буфер передачи .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** - Байт был успешно записан в сокет.

**ЛОЖЬ -** буфер передачи уже заполнен , и поэтому запись не удалось.

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

**10.22.1.13 UDPPut Функция**

**файл**

UDP.h

C

BOOL **UDPPut**(
BYTE **v**
);

**описание**

Эта функция записывает один байт на текущий активный UDP сокет , а увеличивая длину буфера . **UDPIsPutReady**

(См. стр. 615) следует использовать перед вызовом этой функции , чтобы указать текущий активный разъем.

**Предпосылки**

UDPIsPutReady (см. стр. 615 ) ( ) был вызван ранее для указания текущего сокета.

**Параметры**

**Параметры Описание**

**v** -Байт должен быть загружен в буфер передачи .

Возвращаемые значения

Возвращаемые значения Описание

**ИСТИНА** - Байт был успешно записан в сокет.

**ЛОЖЬ -** буфер передачи уже заполнен , и поэтому запись не удалось.

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

**10.22.1.14 UDPPutArray Функция**

**файл**

UDP.h

C

WORD **UDPPutArray**(
BYTE * **cData**,
WORD **wDataLen**
);

**Возвращает**

Число байтов, успешно размещены в буфере передачи UDP . Если это значение меньше wDataLen , то буфер стала полноправным и вход был усечен .

**описание**

Эта функция записывает массив байтов с активным UDP сокет , а увеличивая длину буфера .

**UDPIsPutReady** (см. стр. 615 ) следует использовать перед вызовом этой функции , чтобы указать текущий активный разъем.

**Предпосылки**

**UDPIsPutReady** (см. стр. 615 ) ( ) был вызван ранее для указания текущего сокета.

**Параметры**

**Параметры Описание**

**cDATA** - Массив писать в сокет.

**wDateLen** Количество байтов из cDATA которые должны быть записаны .

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

**10.22.1.15 UDPPutROMArray Функция**

**файл**

UDP.h

C

WORD **UDPPutROMArray**(
ROM BYTE * **cData**,
WORD **wDataLen**
);

**Возвращает**

Число байтов, успешно размещены в буфере передачи UDP . Если это значение меньше wDataLen , то буфер стала полноправным и вход был усечен .

**описание**

Варианты функций ПЗУ для PIC18

Эта функция записывает массив байтов из ПЗУ в активной в данный момент UDP сокет , а увеличивая длину буфера .

**UDPIsPutReady** (см. стр. 615 ) следует использовать перед вызовом этой функции , чтобы указать текущий активный разъем.

**Замечания**

Эта функция является псевдонимом для **UDPPutArray** (см. стр. 616 ) на не- PIC18 платформ.

**Предпосылки**

**UDPIsPutReady** (см. стр. 615 ) ( ) был вызван ранее для указания текущего сокета.

Параметры

**Параметры Описание**

**cDATA** - Массив писать в сокет.

**wDateLen** Количество байтов из **cDATA** которые должны быть записаны .

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

**10.22.1.16 UDPPutROMString Функция**

**файл**

UDP.h

C

ROM BYTE* **UDPPutROMString**(
ROM BYTE * **strData**
**);**

**Возвращает**

Указатель байт после последнего байта, записанного . Заметим, что это отличается от **UDPPutArray** (см. стр. 616 ) функции. Если этот указатель не разыменовывает к нулевой байт , то буфер стала полноправным и входные данные были усечены .

**описание**

Эта функция записывает строку с завершающим нулем из ПЗУ в активной в данный момент UDP сокет , а увеличивая буфер Длина . **UDPIsPutReady** (см. стр. 615 ) следует использовать перед вызовом этой функции , чтобы указать текущий активный разъем.

**Замечания**

Эта функция является псевдонимом для **UDPPutString** (см. стр. 617) на не- PIC18 платформ.

**Предпосылки**

**UDPIsPutReady** (см. стр. 615 ) ( ) был вызван ранее для указания текущего сокета.

**Параметры**

**Параметры Описание**

**cDATA**- Указатель на строку , чтобы записать в сокет.

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

**10.22.1.17 UDPPutString Функция**

**файл**

UDP.h

C

BYTE* **UDPPutString**(
BYTE * **strData**
);

**Возвращает**

Указатель байт после последнего байта, записанного . Заметим, что это отличается от **UDPPutArray** (см. стр. 616 ) функции. Если этот указатель не разыменовывает к нулевой байт , то буфер стала полноправным и входные данные были усечены .

**описание**

Эта функция записывает строку с завершающим нулем на текущий активный UDP сокет , а увеличивая длину буфера .

**UDPIsPutReady** (см. стр. 615 ) следует использовать перед вызовом этой функции , чтобы указать текущий активный разъем.

**Предпосылки**

**UDPIsPutReady** (см. стр. 615 ) ( ) был вызван ранее для указания текущего сокета.

**Параметры**

**Параметры Описание**

**cDATA** - Указатель на строку , чтобы записать в сокет.

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

**10.22.1.18 UDPSetRxBuffer Функция**

**файл**

UDP.h

C

**void UDPSetRxBuffer**(
WORD **wOffset**
);

**Возвращает**

Ни один

**описание**

Эта функция позволяет расположение чтения в буфере RX должны быть указаны . Будущие вызовы к **UDPGet** (см. стр. 613 ) и **UDPGetArray** (см. стр. 614 ) будет читать данные из указанном месте вперед.

**Предпосылки**

**UDPInit** (см. стр. 621 ) ( ) должны быть предварительно позвонил и сокет в настоящий момент.

**Параметры**

**Параметры Описание**

**wOffset** Смещение от начала UDP полезной нагрузки пакета данных для установки указателя чтения.

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

### 10.22.1.19 UDPSetTxBuffer Функция

**файл**

UDP.h

C

```
void UDPSetTxBuffer(
WORD wOffset
) ;
```

**Возвращает**

Ни один

**описание**

Эта функция позволяет расположение записи в буфере TX должны быть указаны . Будущие вызовы к **UDPPut** (см. стр. 615) , **UDPPutArray** (см. стр. 616 ) , **UDPPutString** (см. стр. 617) , и т.д. будет записывать данные с указанного места.

**Предпосылки**

**UDPInit** (см. стр. 621 ) ( ) должны быть предварительно позвонил и сокет в настоящий момент.

**Параметры**

**Параметры Описание**

**wOffset** - Смещение от начала UDP полезной нагрузки пакета данных для установки указателя записи.

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

**10.22.1.20 UDPIsOpened Функция**

**файл**

UDP.h

C

BOOL **UDPIsOpened**(
UDP_SOCKET **socket**
);

**описание**

Эта функция определяет, является ли сокет имеет установленную подключение к удаленному узлу. Эта функция вызывается после вызова

**UDPOpen** (см. стр. 611 ), чтобы определить , когда соединение установлено и готово к использованию.

**Замечания**

Ни один

**Предпосылки**

UDP инициализации .

**Параметры**

**Параметры Описание**

socket ( см. стр. 180) Гнездо для проверки .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** сокет был открыт и ARP была решена.

**ЛОЖЬ** сокет не подключены .

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

### 10.22.1.21 UDP_OPEN_IP_ADDRESS Макро

**файл**

UDP.h

C

**#define UDP_OPEN_IP_ADDRESS** 3u

**описание**

Создание клиентского сокета и использовать **dwRemoteHost** как буквальное IP адресу.

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

### 10.22.1.22 UDP_OPEN_NODE_INFO Макро

**файл**

UDP.h

C

**#define UDP_OPEN_NODE_INFO** 4u

**описание**

Создание клиентского сокета и использовать **dwRemoteHost** как указатель на структуру **NODE_INFO** содержащий точную удаленный IP решения и MAC- адрес использовать .

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

### 10.22.1.23 UDP_OPEN_RAM_HOST Макро

**файл**

UDP.h

C

**#define UDP_OPEN_RAM_HOST**
You_need_to_enable_STACK_USE_DNS_to_use_UDP_OPEN_RAM_HOST

**описание**

Выделяют необъявленную идентификатор диагностическую если код пытается использовать **UDP_OPEN_RAM_HOST** в то время как клиентский модуль DNS не

включен .

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

### 10.22.1.24 UDP_OPEN_ROM_HOST Макро

**файл**

UDP.h

C

**#define UDP_OPEN_ROM_HOST**
You_need_to_enable_STACK_USE_DNS_to_use_UDP_OPEN_ROM_HOST

**описание**

Выделяют необъявленную идентификатор диагностическую если код пытается использовать **UDP_OPEN_ROM_HOST** в то время как клиентский модуль DNS не включен .

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

### 10.22.1.25 UDP_OPEN_SERVER Макро

**файл**

UDP.h

C

**#define UDP_OPEN_SERVER** 0u

**описание**

Создание серверного сокета и игнорировать **dwRemoteHost** .

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

### 10.22.2 UDP стека Пользователи

**Функции**

**Имя Описание**

**UDPInit** (см. стр. 621 ) Инициализирует UDP модуль .

**UDPProcess** (см. стр.621) Ручки на входящий сегмент UDP .

**UDPTask** (см. стр. 622 ) Выполняет государственное управление и обслуживание номеров для UDP.

**модуль**

**UDP** (см. стр. 607 )

**описание**

Следующие функции и переменные являются открытыми , но предназначены только обратиться к нему по самой стека. приложения должны, как правило, не называют эти функции или модифицировать эти переменные .

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

**10.22.2.1 UDPInit Функция**

**файл**

UDP.h

C

недействительными **UDPInit** ( ) ;

**Возвращает**

Ни один

**описание**

Инициализирует модуль UDP . Эта функция инициализирует все UDP сокетов в закрытом состоянии .

**Замечания**

Эта функция вызывается только один при жизни приложения.

**Предпосылки**

Ни один

**раздел**

Прототипы функций

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

### 10.22.2.2 UDPProcess Функция

**файл**

UDP.h

C

```
BOOL UDPProcess(
NODE_INFO * remoteNode,
IP_ADDR * localIP,
WORD len
);
```

**описание**

Эта функция обрабатывает входящий UDP сегмент , чтобы определить , если это приемлемо и должны быть переданы в один из стека приложения для обработки.

**Предпосылки**

**UDPInit** (см. стр. 621 ) ( ) была вызванаUDP сегмент готов в буфере MAC .

**Параметры**

**Параметры Описание**

**remoteNode** удаленного узла , который послал этот сегмент .

**LocalIP** IP- адрес назначения для этого сегмента .

**Len -** Общая протяженность сегмента UDP .

**Возвращаемые значения**

**Возвращаемые значения Описание**

**ИСТИНА** действует пакет ждет и приложений стека следует называть справиться с этим.

**ЛОЖЬ** пакет был отброшен .

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

### 10.22.2.3 UDPTask Функция

**файл**

UDP.h

C

**void UDPTask**();

**описание**

Выполняет государственное управление и обслуживание номеров для UDP. Это внутренняя функция предназначен для вызова на **StackTask** ( ) ( не Пользователь API) .

**Замечания**

**UDPTask** () вызывается один раз в **StackTask** ( ) итерации для того, чтобы звонки на **UDPIsPutReady** (см. стр. 615) ( ) всегда обновлять расположение Ethernet Написать указатель между **StackTask** ( ) итераций .

**Предпосылки**

Ни один

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

**10.22.3 UDP Внутренняя Пользователи**

**Функции**

**Имя Описание**

**FindMatchingSocket** (см.страница 623 ) Матчи на входящий сегмент UDP , к текущему розетки.

**макрос**

**Имя Описание**

**LOCAL_UDP_PORT_END_NUMBER** (См. Стр. 624 ) Последний номер порта для рандомизированного отбора локальный номер порта

**LOCAL_UDP_PORT_START_NUMBER** (См. Стр. 624 ) Число Первый порт для рандомизированного отбора локальный номер порта

**модуль**

**UDP** (см. стр. 607 )

**структуры**

**Имя Описание**

**UDP_HEADER** (см. стр.625 ) Магазины заголовок пакета UDP

**UDP_SOCKET_INFO** (см.страница 625 ) Хранение информации о текущем UDP сокет

**Виды**

**Имя Описание**

**UDP_PORT** (см. стр.625 ) Сохраняет ряд UDP-порта

**Переменные**

**Имя Описание**

**activeUDPSocket** (см.страница 623 ) Указывает, какой сокет UDP в настоящее время активно

**LastPutSocket** (см. стр.624) Указывает последний разъем , к которому данные были записаны

**SocketWithRxData** (см.страница 624 ) Указывает, какой сокет настоящее время полученные данные для этого цикла

**UDPRxCount** (см. стр.626) Количество прочитанных байтов из этого UDP сегменте

**UDPSocketInfo** (см. стр.626) Сохраняет массив информации, касающейся каждого сокета UDP

**UDPTxCount** (см. стр.626) Количество байтов, записанных в этом UDP сегменте

**wGetOffset** (см. стр. 626 ) Offset с начала полезной нагрузки , откуда данные будут читать.

**wPutOffset** (см. стр. 627 ) Offset с начала полезной нагрузки , где данные должны быть записаны .

**описание**

Следующие функции и переменные обозначены как внутренний модулю UDP .

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

**10.22.3.1 activeUDPSocket Переменная**

**файл**

udp.c

C

UDP_SOCKET **activeUDPSocket**;

**описание**

Указывает, какой сокет UDP в настоящее время активно

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

**10.22.3.2 FindMatchingSocket Функция**

**файл**

udp.c

C

```
static UDP_SOCKET FindMatchingSocket(
UDP_HEADER * h,
NODE_INFO * remoteNode,
IP_ADDR * localIP
);
```

**Возвращает**

**UDP_SOCKET (** см. стр. 609) ручку соответствующий разъем, или
**INVALID_UDP_SOCKET** (см. стр. 609 ) , когда нет матч можно было бы сделать .

**описание**

Эта функция пытается сопоставить входящий сегмент UDP , к текущему гнездо для обработки.

**Предпосылки**

UDP заголовка сегмента и IP- заголовок оба были извлечены .

**Параметры**

**Параметры Описание**

**h** -UDP заголовок, который был получен.

**remoteNode** IP и MAC удаленного узла , который послал этот сегмент .

**localIP** IP-адрес , что этот сегмент был предназначен для .

**раздел**

Прототипы функций

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

**10.22.3.3 LastPutSocket Переменная**

**файл**

udp.c

C

UDP_SOCKET **LastPutSocket** = INVALID_UDP_SOCKET;

**описание**

Указывает последний разъем , к которому данные были записаны

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

**10.22.3.4 LOCAL_UDP_PORT_END_NUMBER Макро**

**файл**

udp.c

C

**#define LOCAL_UDP_PORT_END_NUMBER** (8192u)

**описание**

Последний номер порта для рандомизированного отбора локальный номер порта

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

**10.22.3.5 LOCAL_UDP_PORT_START_NUMBER Макро**

**файл**

udp.c

C

#define LOCAL_UDP_PORT_START_NUMBER ( 4096u )

**описание**

Число Первый порт для рандомизированного отбора локальный номер порта

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

**10.22.3.6 SocketWithRxData Переменная**

**файл**

udp.c

C

UDP_SOCKET **SocketWithRxData** = INVALID_UDP_SOCKET;

**описание**

Указывает, какой сокет настоящее время полученные данные для этого цикла

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

**10.22.3.7 UDP_HEADER Структура**

**файл**

UDP.h

C

```
typedef struct {
UDP_PORT SourcePort;
UDP_PORT DestinationPort;
WORD Length;
WORD Checksum;
} UDP_HEADER;
```

**Пользователи**

**Пользователи Описание**

**UDP_PORT SourcePort** ; Источник UDP порт

**UDP_PORT DestinationPort** ; Направление UDP порт

**WORD  Length** ; Длина данных

**WORD Checksum** ; UDP контрольная сумма данных

**описание**

Магазины заголовок пакета UDP

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

**10.22.3.8 UDP_PORT Тип**

**файл**

UDP.h

C

```
typedef WORD UDP_PORT;
```

**описание**

Сохраняет ряд UDP-порта

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

**10.22.3.9 UDP_SOCKET_INFO Структура**

**файл**

UDP.h

C

```
typedef struct {
union {
NODE_INFO remoteNode;
DWORD remoteHost;
} remote;
UDP_PORT remotePort;
UDP_PORT localPort;
UDP_STATE smState;
DWORD retryInterval;
BYTE retryCount;
struct {
unsigned char bRemoteHostIsROM : 1;
} flags;
WORD eventTime;
} UDP_SOCKET_INFO;
```

**Пользователи**

**Пользователи Описание**

**NODE_INFO** remoteNode ; 10 байт для MAC и IP адрес

**DWORD RemoteHost** ; ОЗУ или указатель ROM к имени хоста строки (например: "www.microchip.com" )

**UDP_PORT RemotePort** ; Номер порта UDP удаленного узла

**UDP_PORT** локальный_порт ; Локальный номер порта UDP , или **INVALID_UDP_PORT** (см. стр. 609 ) , когда бесплатно

**UDP_STATE smState** ; Государство этого сокета

неподписанные символ bRemoteHostIsROM : 1 ; Удаленный хост хранится в ПЗУ

**описание**

Хранение информации о текущем UDP сокет

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

**10.22.3.10 UDPRxCount Переменная**

**файл**

udp.c

C

WORD **UDPRxCount**;

**описание**

Количество прочитанных байтов из этого UDP сегменте

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

**10.22.3.11 UDPSocketInfo Переменная**

**файл**

udp.c

C

UDP_SOCKET_INFO **UDPSocketInfo**[MAX_UDP_SOCKETS];

**описание**

Сохраняет массив информации, касающейся каждого сокета UDP

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

### 10.22.3.12 UDPTxCount Переменная

**файл**

udp.c

C

WORD **UDPTxCount**;

**описание**

Количество байтов, записанных в этом UDP сегменте

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

### 10.22.3.13 wGetOffset Переменная

**файл**

udp.c

C

WORD **wGetOffset**;

**описание**

Смещение от начала полезной нагрузки , откуда данные должны быть считаны .

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

### 10.22.3.14 wPutOffset Переменная

**файл**

udp.c

C

WORD **wPutOffset**;

**описание**

Смещение от начала полезной нагрузки , где данные должны быть записаны .

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

**10.22.4 Типы**

**Перечисления**

**Имя Описание**

**UDP_STATE** (см. стр.627) UDP States

**модуль**

**UDP** (см. стр. 607 )

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

**10.22.4.1 UDP_STATE Перечисление**

**файл**

UDP.h

C

```
typedef enum {
UDP_DNS_IS_RESOLVED,
UDP_DNS_RESOLVE,
UDP_GATEWAY_SEND_ARP,
UDP_GATEWAY_GET_ARP,
UDP_CLOSED,
UDP_OPENED
} UDP_STATE;
```

**Пользователи**

**Пользователи Описание**

**UDP_DNS_IS_RESOLVED** Специальное состояние для UDP сокетов в режиме клиент

**UDP_DNS_RESOLVE** Специальное состояние для UDP сокетов в режиме клиент

**UDP_GATEWAY_SEND_ARP** Специальное состояние для UDP сокетов в режиме клиент

**UDP_GATEWAY_GET_ARP** Специальное состояние для UDP сокетов в режиме клиент

**UDP_CLOSED** гнездо находится в режиме ожидания и нераспределенные

**описание**

**UDP** Штаты

## 11. Wi-Fi API

### Модули

Имя Описание

**Wi-Fi Connection Profile-** Wi-Fi соединения профиля (см. стр. 631 ) Функции в установке , использовании и профилей подключения демонтажа

**Wi-Fi Connection Algorithm -** Wi-Fi соединения Алгоритм (см. Стр.648) Функции , чтобы изменить поведение процесса подключения

**Wi-Fi Connection Manager -** Функции для управления процессом подключения

**Wi-Fi Scan** (см. стр. 675) Функции направить MRF24WB0M инициировать исследования площадки

**Wi-Fi Tx Power Control -**Wi-Fi Tx Управление питанием ( см. стр. 678 ) API для управления мощностью Tx о MRF24WB0M

**Wi-Fi Power Save -** Wi-Fi Экономия энергии (см. стр. 681 ) Функции , чтобы изменить энергосбережение черты MRF24WB0M

**Wi-Fi Miscellaneous** -Wi-Fi Разное (см. стр. 689) Функции для управления различными черты MRF24WB0M

### описание

В отличие от Ethernet , приложение Wi-Fi должен инициировать соединение с точкой доступа или специальную сеть ) до данных коммуникации может начаться. Для того, чтобы инициировать соединение существует последовательность шагов, которые должны быть соблюдены.

1)профиль подключения должны быть созданы (см. WF_CPCreate (см. стр. 633 ) ( ) ) . Соединительный профиль содержит информацию направляя WiFi водителя о характере связи , который будет создан . Соединительный профиль определяет :

**а**. SSID ( имя точки доступа )

**б** . Безопасности ( открытый, WEP, WPA и т.д.)

**с** . Тип сети (инфраструктура или специальной ) .

Функции соединения профиля используются для создания и определить профиль подключения . Эти все функции начинаются с **WF_CP ...**

2) Алгоритм подключения должны быть определены , и применяется ко всем профилям подключения . Для большинства приложений по умолчанию будет достаточно . Например , список алгоритм канал соединение по умолчанию для сканирования 1 , 6 и 11 . Однако, если, в вашем Приложение, которое вы знаете точка доступа всегда будет на канале 6 вы могли бы изменить эту настройку , в результате чего сканирование процесс более эффективным . Функции , относящиеся к алгоритму подключения все начинается с **WF_CA ...**

3) После того, как профиль подключения и алгоритм подключения настроены для приложения ,**WF_CMConnect** (см.Функция страница 671 ) ( ) должен быть вызван , чтобы начать процесс подключения.

4) После **WF_Connect** () вызывается основное приложение будете уведомлены, когдаMRF24WB0M удалось (или не удалось) в Установка соединения с использованием механизма событий. Файл **WF_Config.c** имеет функцию , **WF_ProcessEvent** (см. стр.698) ( ) , что является шаблоном для обработки MRF24WB0M событий . В WiFi демо это просто печатается на консоль ( если UART включена ) , что произошло событие. Этот файл может быть изменен, чтобы удовлетворить потребности приложения - например, приложение может зависят в глобальном флага, который будет установлен в **WF_ProcessEvent** (см. стр. 698 ) ( ) , когда соединение установлено .

Пожалуйста, обратитесь к **WF_ProcessEvent** (см. стр. 698 ) для получения дополнительной информации об обработке событий WiFi. В MRF2WB0M демо ( под демонстрационное приложение , Wi-Fi консоли , и **Wi-Fi EZ Config** демо каталогов ) содержать функцию , **WF_Connect ( )** , в MainDemo.c , который выполняет все дальнейшие шаги и может упоминаться как пример того, как приступить к WiFi подключение . Файл **WF_Config.h** имеет несколько времени компиляции константы, которые могут быть настроены (например, MY_DEFAULT_SSID_NAME ) по мере необходимости.

В этом файле справки книге описывается хоста API к на-чипе диспетчера соединений MRF24WB0M что создает и поддерживает Wi-Fi соединения. API разделена на этих основных разделов :

**Initialization** - Инициализации (см. стр.138) Функции для инициализации хост API и MRF24WB0M

**Connection Profile -**Функции соединения профиля для создания и поддержания одного или нескольких профилей подключения

**Connection Algorithm** - Алгоритм соединения Функции для тонкой настройки алгоритма соединения

**Connection Manager** - Менеджер соединения Функции запустить и остановить подключение к 802.11

**Scan** - Функции сканирования для поиска беспроводных сетей

T**x Power Control** -Функции Tx управления питанием для управления мощностью MRF24WB0M Tx

**Power Save**-Функции , чтобы сэкономить расход энергии на MRF24WB0M

**Multicast** - функции для создания многоадресной фильтры

**Miscellaneous** - Дополнительные функции Для установки пользовательского MAC-адрес , получить информацию об устройстве и т.д.

**MRF24WB0M Events** - События функции для обработки события из MRF24WB0M

**SPI**

Файл WF_Spi.c содержит функции, которыеWi-Fi драйвер будет использовать для инициализации , отправлять и получать сообщения SPI между центральный процессор и MRF24WB0M . Для связи с MRF24WB0M , который всегда раб SPI , центральный процессор

Контроллер SPI должен быть настроен следующим образом:

• Режим = 0

• CPOL (часы полярности ) = 0

• CPHA ( фаза синхронизации ) = 0

• хостов по ЦПУ установлен в качестве ведущего

• Часы вхолостую высокой

• 8-битный длина передачи

• Данные изменения по заднему фронту

• Данные пробы на переднем фронте

Ниже приведен список функций в WF_Spi.c , которые должны быть настроены для конкретного хоста архитектуры процессора :

**Функция Описание**

**WF_SpiInit** () Инициализирует центрального процессора SPI контроллер для использования драйвером Wi-Fi. Вызывается Wi-Fi драйвер во время инициализации .

**WF_SpiTxRx ( )** передает и / или получает данные SPI от MRF24WB0M .

Низкий **WF_SpiEnableChipSelect ( )** Набор раб выберите линию на MRF24WB0M (начало SPI передачу ) .

Если SPI автобус делится с другими устройствами , то эта функция также должна сохранить Текущий контекст SPI , а затем настроить контекст MRF24WB0M SPI .

**WF_SpiDisableChipSelect ( )** Набор раб выберите линию на MRF24WB0M высокой (конец SPI передачи ).

Если SPI автобус делится с другими устройствами , то эта функция также нуждается в восстановлении контекст SPI ( сохраняется в течение **WF_SpiEnableChipSelect ( )** ) .

**Внешнее прерывание**

Файл WF_Eint.c содержит функции, которыеWi-Fi Driver будет использовать для включения и отключения MRF24WB0M внешний прерывать , а также получить статус прерывания . Функции в этом модуле должны быть настроены для конкретной архитектуры центрального процессора.

MRF24WB0M утверждает свое EXINT линии ( внешнее прерывание ) ( активный низкий) при возникновении определенных событий , например, данных сообщение принимается. Обратите внимание, что центральный процессор имеет выбор, либо настроить EXINT линию для получения актуальной прервать , или он может быть опрошены . Ниже приведен список из Wi-Fi функций драйвера внутри WF_Eint.c , которые должны быть настроены для конкретных хостов архитектура процессора .

**Функция Описание**

**WF_EintInit ( )** Настраивает прерывание для использования и оставляет ее в отключенном состоянии . Будет называться Wi-Fi драйвер во время инициализации . Если опросEXINT контактный то эта функция не будет любая работа , чтобы сделать, кроме как покинуть прерывание в логически отключенном состоянии.

**WF_EintEnable ( )** Включает внешнее прерывание MRF24WB0M . Если с использованием реальных прерывания затем включите прервать . Если опрос EXINT контактный то эта функция позволяет опрос штифта .

**WF_EintDisable ( )** Отключает внешнее прерывание MRF24WB0M . Если с использованием реальных прерывания затем отключить прервать . Если опрос EXINT контактный то эта функция отключает опрос штифта.

**WF_EintIsr ( )** Это процедура обработки прерывания вызывается, когда EXINT линия идет на низком уровне. Он должен выполните необходимое ведение домашнего хозяйства, например, очистка прерывание. Прерывание должны оставаться отключен до тех пор, Wi-Fi Driver не называет **WF_EintEnable ( )**. Wi-Fi Функция драйвера , **WFEintHandler ( )** должен быть вызван .

**WF_EintIsDisabled ( )** Возвращает истину, если внешнее прерывание отключена , иначе возвращает ложь.

**WFEintHandler ()** Эта функция не нужно подгонять - это часть драйвера Wi-Fi. Однако он добавляется в этот список, потому что он должен быть вызван каждый раз, когда MRF24WB0M прервать Процедура обслуживания (ISR) происходит.

**WF_Config**

Модуль WF_Config ( WF_Config.h / WF_Config.c ) используется для управления несколько аспектов поведения водителя WiFi. Наиболее из настройки модуля Wi-Fi делается из контекста данного модуля .

Удаление неиспользуемых функций драйвера В WF_Customize.h есть блок из определено, что можно прокомментировать , чтобы удалить те разделы хозяина Wi-Fi Водитель, который не нужны приложения. Это позволяет сохранить код и пространстве данных.

# определить контрольные функции

**WF_USE_SCAN_FUNCTIONS-** сканирования API

**WF_USE_TX_POWER_CONTROL_FUNCTIONS-** Tx управления мощностью API

**WF_USE_POWER_SAVE_FUNCTIONS** энергосбережения API

**WF_USE_MULTICAST_FUNCTIONS**- Multicast API

**WF_USE_INDIVIDUAL_SET_GETS** -Влияет на все получения и установки функции , за исключением следующих:

WF_CPSetElements (см. стр. 640 ) ( )

WF_CPGetElements (см. стр. 635 ) ( )

WF_CASetElements (см. стр. 660 ) ( )

WF_CAGetElements (см. Стр. 652 ) ( )

**WF_USE_GROUP_SET_GETS** -влияет на следующие функции:

WF_CPSetElements (см. стр. 640 ) ( )

WF_CPGetElements (см. стр. 635 ) ( )

WF_CASetElements (см. стр. 660 ) ( )

WF_CAGetElements (см. Стр. 652 ) ( )

**WF_DEBUG**

Это определение позволяет WF_ASSERT макрос в драйвере Wi-Fi. Код клиента может свободно использовать этот макрос . WF_ASSERT макрос может быть скомпилирован в или через WF_DEBUG определить . См.выше WF_DEBUG комментарий определить в WF_Customize.h для деталей.

**WF_CONSOLE**

Водитель Wi-Fi имеет UART консольное приложение , построенный в том, что позволяет ввести в командные строки и их анализу. Если эта функциональность не требуется , чем это может быть собрано из закомментировавWF_CONSOLE определить .

**WF_ProcessEvent ( )**

Эта функция вызывается Wi-Fi Driver , когда происходит событие, центральный процессор должен быть уведомлен о . Есть несколько Wi-Fi подключение событий, связанных с , что приложение может выбрать, следует ли получать уведомления или нет. И , есть несколько событий приложение всегда будет оповещен о .

Функциия WF_ProcessEvent (см. стр. 698 ) ( ) может быть настроена для поддержки желаемой обработки событий .

## 11.1 Wi-Fi соединения профиля

**модуль**

Wi-Fi интерфейс API (см. стр. 628 )

**макрос**

**Имя Описание**

RAW_MGMT_RX_ID (см.страница 648 ) Это макро RAW_MGMT_RX_ID .

**описание**

В этом разделе описываются функции API , связанные с созданием и использованием профилей подключения . По меньшей мере один соединительный профиль должен быть создан . Соединительный профиль определяет элементы, необходимые по MRF24WB0M установить соединение с Wi-Fi сеть .

Изменение профиля подключения После установки соединения

Соединительный профиль может быть обновлена в то время как он используется для активного соединения. Однако обновления не вступили в силу пока один из следующих не происходит :

• Подключение отключена и повторно включен ведущим приложением

• Алгоритм соединения теряет подключение , исчерпывает все попытки , а затем перезагружает профиль подключения .

Чтобы гарантировать, что обновления профиля подключения состоится в известной точке времени рекомендуется , что вызов хост-приложения **WF_CMDisconnect** (см. стр. 672 ) ( ) , обновить профиль подключения , а затем вызвать **WF_CMConnect** (см. стр.671 ) () .

### 11.1.1 Подключение профиля открытые члены

**Функции**

**Имя Описание**

**WF_CPCreate** (см. стр. 633 ) Создает Профиля на MRF24W .

**WF_CPDelete** (см. стр. 633 ) Удаляет Профиля на MRF24W .

**WF_Behaviorist** (см. стр. 634 ) Получает желаемое поведение Специальную

**WF_CPGetBssid** (см. стр.634) Получает BSSID для указанного соединения Профиль ID .

**WF_CPGetDefaultWepKeyIndex**(См. Стр. 635 ) Получает значение активных ключей WEP для использования.

**WF_CPGetElements** (см.страница 635 ) Читает элементы соединения профиля для указанного ID .

**WF_CPGetIds** (см. стр. 636 ) Получает битовую маску CP ID .

**WF_CPGetNetworkType** (см.страница 636 ) Получает сеть для указанного соединения Профиль ID .

**WF_CPGetSecurity** (см. стр.637 ) Получает безопасности для указанного профиля подключения .

**WF_CPGetSsid** (см. стр. 638) Получает SSID для указанного соединения Профиль ID .

**WF_CPSetAdHocBehavior** (см. стр. 638 ) Выбор желаемого поведения Специальную

**WF_CPSetBssid** (см. стр.639) Устанавливает BSSID для указанного соединения Профиль ID .

**WF_CPSetDefaultWepKeyIndex** (См. Стр. 640 ) Выбор одной из 4 клавиш WEP для использования.

**WF_CPSetElements** (см. страница 640) Записывает данные для конкретного профиля подключения элемента.

**WF_CPSetNetworkType** (см. страница 641 ) Устанавливает сеть для указанного соединения Профиль ID .

**WF_CPSetSecurity** (см. стр. 641 ) Устанавливает безопасности для указанного профиля подключения .

**WF_CPSetSsid** (см. стр.642) Устанавливает SSID для указанного соединения Профиль ID .

**WF_CPGetSsidType** (см. страница 644 ) Получает тип SSID для указанного соединения Профиль ID .

**WF_CPSetSsidType** (см. страница 644 ) Устанавливает тип SSID для указанного соединения Профиль ID .

**WF_CPGetWepKeyType** (см. страница 645 ) Получает тип ключа WEP для указанного соединения Профиль ID .

**WF_CPGetWPSCredentials** ( см. стр. 645 ) Получает WPS учетные данные для указанного соединения Профиль ID .

**WF_CPSetWepKeyType** (см.страница 646) Задает тип ключа WEP для указанного соединения Профиль ID .

**модуль**

Wi-Fi соединения профиля (см. стр. 631 )

**структуры**

**Имя Описание**

**WFCPElementsStruct** (см.страница 643 ) Структура профиля соединения элементов

**описание**

//*-*-*-*-*----------

### 11.1.1.1 WF_CPCreate Функция

файл

WFApi.h

C

```
void WF_CPCreate (
UINT8 * p_CpId
) ;
```

**Возвращает**

Ни один .

**описание**

Подключение профиля Функции

Просит MRF24W создать профиль подключения (CP) , присвоить ему идентификатор , и установить все элементы по умолчанию. ID , возвращенный этой функцией используется в других профильных функций подключения . Максимум 2 профилей подключения может существовать на MRF24W .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**p_CpId** Указатель где соединения Анкета ID будет написано . Если функция завершается с ошибкой , CP ID будет установлен в 0xFF.

//*-*-*-*-*----------

**11.1.1.2 WF_CPDelete Функция**

**файл**

WFApi.h

**C**

```
void WF_CPDelete (

UINT8 CPID

) ;
```

**Возвращает**

Ни один .

**описание**

Удаляет указанный Профиля . Если профиля соединения был в FLASH оно будет удалено из FLASH .

**Замечания**

Первый релиз этого кода не будет поддерживать FLASH, только два CP находится в памяти .

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID профиля подключения для удаления.

//*-*-*-*-*----------

**11.1.1.3 WF_CPGetAdHocBehavior Функция**

**файл**

WFApi.h

C

void WF_CPGetAdHocBehavior (

UINT8 CPID ,

UINT8 * p_adHocBehavior

) ;

**Возвращает**

Ни один .

**описание**

Получает поведение AdHoc внутри профиля подключения . Допустимые значения:

• WF_ADHOC_CONNECT_THEN_START

• WF_ADHOC_CONNECT_ONLY

• WF_ADHOC_START_ONLY

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

**adHocBehavior** Указатель на месте от стоимости поведения AdHoc для этого профиля подключения.

//*-*-*-*-*----------

### 11.1.1.4 WF_CPGetBssid Функция

**файл**

WFApi.h

C

```
void WF_CPGetBssid (
UINT8 CPID ,
UINT8 * p_bssid
) ;
```

**Возвращает**

Ни один .

**описание**

Получает элемент BSSID в профиль подключения .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

p_bssid Указатель на BSSID

//*-*-*-*-*----------

### 11.1.1.5 WF_CPGetDefaultWepKeyIndex Функция

**файл**

WFApi.h

C

void WF_CPGetDefaultWepKeyIndex (

UINT8 CPID ,

UINT8 * p_defaultWepKeyIndex

) ;

**Возвращает**

Ни один .

**описание**

Применяется только если соединения профиля тип защиты является либо WF_SECURITY_WEP_40 или WF_SECURITY_WEP_104 .

Выбирает, какой из четырех ключей WEP для использования.

**Замечания**

Обратите внимание, что единственный ключ 0 среди производителей AP обычно используется . Использование любого из других трех клавиш может быть непредсказуемым

от бренда к бренду .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

p_defaultWepKeyIndex Указатель на индекс ключа WEP использовать (0 — 3)

//*-*-*-*-*----------

### 11.1.1.6 WF_CPGetElements Функция

**файл**

WFApi.h

C

void WF_CPGetElements (

UINT8 CPID ,

tWFCPElements * p_elements

) ;

**Возвращает**

Ни один .

**описание**

Получает все элементы соединения профиля для указанного CP ID . Если хостов по ЦПУ не хватает памяти для создания Структура такого размера затем вызвать отдельные функции получить .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID Connectino Анкета ID .

p_elements указатель на структуру элементов соединения профиля .

//*-*-*-*-*----------

**11.1.1.7 WF_CPGetIds Функция**

**файл**

WFApi.h

C

void WF_CPGetIds (

UINT8 * cpIdList

) ;

**Возвращает**

Ни один .

**описание**

Возвращает список всех соединения анкеты годов , которые были созданы на MRF24W . Это не следует путать с Подключение алгоритма connectionProfileList . Эта функция возвращает

битовую маску , соответствующую список всех Connection Профили , которые были созданы ( являются ли они в connectionProfileList или нет) . Любые Профили подключений , которые имеют был сохранен на FLASH будут включены .

**Замечания**

первый релиз будет поддерживать только два Профайлы соединений в памяти . Сохранение компартий мигать не будут поддерживаться .

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**p_cpIdList** Указатель на значение, представляющее битовую маску , где каждый бит индекс (плюс 1) соответствует соединения Профиля, который был создан. Например, если это значение 0x03 , то соединения профиля идентификаторов 1 и и 2 были созданы .

//*-*-*-*-*----------

### 11.1.1.8 WF_CPGetNetworkType Функция

**файл**

WFApi.h

C

```
void WF_CPGetNetworkType (
UINT8 CPID ,
UINT8 * p_networkType
) ;
```

**Возвращает**

Ни один .

**описание**

Получает элемент Тип сети профиля подключения . Допустимые значения:

• WF_INFRASTRUCTURE

• WF_ADHOC

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

NETWORKTYPE Тип сети для создания (инфраструктура и специальными )

//*-*-*-*-*----------

**11.1.1.9 WF_CPGetSecurity Функция**

**файл**

WFApi.h

C

```
void WF_CPGetSecurity (
UINT8 CPID ,
UINT8 * p_securityType ,
UINT8 * p_wepKeyIndex ,
UINT8 * p_securityKey ,
UINT8 * p_securityKeyLength
) ;
```

**Возвращает**

Ни один .

**описание**

Настраивает безопасность для профиля подключения .

Безопасность Длина ключа

WF_SECURITY_OPEN N / A N / A

WF_SECURITY_WEP_40 шестигранная 4 , 5 ключей байт

WF_SECURITY_WEP_104 шестигранная 4 , 13 клавиши байт

WF_SECURITY_WPA_WITH_KEY шестигранная 32 байта

WF_SECURITY_WPA_WITH_PASS_PHRASE ASCII 8-63 символов ASCII

WF_SECURITY_WPA2_WITH_KEY шестигранная 32 байта

WF_SECURITY_WPA2_WITH_PASS_PHRASE ASCII 8-63 символов ASCII

WF_SECURITY_WPA_AUTO_WITH_KEY шестигранная 32 байта

WF_SECURITY_WPA_AUTO_WITH_PASS_PHRASE ASCII 8-63 символов ASCII

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**CPID** соединения Анкета ID

**SecurityType** Значение, соответствующее типу желаемого безопасности .

**wepKeyIndex** от 0 до 3 (используется только если тип защиты является
WF_SECURITY_WEP_40 или

WF_SECURITY_WEP_104 )

**p_securityKey** двоичной ключ или фраза (не используется , если безопасность
WF_SECURITY_OPEN )

**securityKeyLength** Количество байтов в p_securityKey (не используется , если безопасность

WF_SECURITY_OPEN )

//*-*-*-*-*----------

**11.1.1.10 WF_CPGetSsid Функция**

**файл**

WFApi.h

C

void WF_CPGetSsid (

UINT8 CPID ,

UINT8 * p_ssid ,

UINT8 * p_ssidLength

) ;

**Возвращает**

Ни один .

**описание**

Получает SSID и SSID длина элементов в профиль подключения .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

p_ssid Указатель на SSID строки

ssidLength Pumber байтов в SSID

//*-*-*-*-*----------

### 11.1.1.11 WF_CPSetAdHocBehavior Функция

**файл**

WFApi.h

C

void WF_CPSetAdHocBehavior (

UINT8 CPID ,

UINT8 adHocBehavior

) ;

**Возвращает**

Ни один .

**описание**

WICOM_MODE

Устанавливает поведение AdHoc внутри профиля подключения . Допустимые значения:

• WF_ADHOC_CONNECT_THEN_START

• WF_ADHOC_CONNECT_ONLY

• WF_ADHOC_START_ONLY

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

adHocBehavior Значение поведения AdHoc для этого профиля подключения.

//*-*-*-*-*----------

**11.1.1.12 WF_CPSetBssid Функция**

**файл**

WFApi.h

C

```
void WF_CPSetBssid (
UINT8 CPID ,
UINT8 * p_bssid
) ;
```

**Возвращает**

Ни один .

**описание**

Устанавливает элемент BSSID в профиль подключения .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

p_bssid Указатель на BSSID

//*-*-*-*-*----------

**11.1.1.13 WF_CPSetDefaultWepKeyIndex Функция**

**файл**

WFApi.h

C

void WF_CPSetDefaultWepKeyIndex (

UINT8 CPID ,

UINT8 defaultWepKeyIndex

) ;

**Возвращает**

Ни один .

**описание**

Применяется только если соединения профиля тип защиты является либо WF_SECURITY_WEP_40 или WF_SECURITY_WEP_104 .

Выбирает, какой из четырех ключей WEP для использования.

**Замечания**

Обратите внимание, что единственный ключ 0 среди производителей AP обычно используется . Использование любого из других трех клавиш может быть непредсказуемым от бренда к бренду .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

defaultWepKeyIndex индекс ключа WEP использовать (0 — 3)

//*-*-*-*-*-----------------------------------------

**11.1.1.14 WF_CPSetElements Функция**

**файл**

WFApi.h

C

void WF_CPSetElements (

UINT8 CPID ,

tWFCPElements * p_elements

) ;

**Возвращает**

Ни один .

**описание**

Устанавливает все элементы соединения профиля . Если хостов по ЦПУ не хватает памяти , чтобы создать структуру этого размера, то называть отдельные установленных функций .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван .

**Параметры**

**Параметры Описание**

CPID Connectino Анкета ID .

p_elements указатель на структуру элементов соединения профиля .

//*-*-*-*-*-----------------------------------------

**11.1.1.15 WF_CPSetNetworkType Функция**

**файл**

WFApi.h

C

```
void WF_CPSetNetworkType (
UINT8 CPID ,
UINT8 NETWORKTYPE
) ;
```

**Возвращает**

Ни один .

**описание**

Задает элемент Тип сети профиля подключения . Допустимые значения:

• WF_INFRASTRUCTURE

• WF_ADHOC

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

NETWORKTYPE Тип сети для создания (инфраструктура и специальными )

```
//*-*-*-*-*-----------------------------------------
```

**11.1.1.16 WF_CPSetSecurity Функция**

**файл**

WFApi.h

C

```
void WF_CPSetSecurity (
UINT8 CPID ,
UINT8 SecurityType ,
UINT8 wepKeyIndex ,
```

UINT8 * p_securityKey ,

UINT8 securityKeyLength

) ;

**Возвращает**

Ни один .

**описание**

Настраивает безопасность для профиля подключения .

Безопасность Длина ключа

WF_SECURITY_OPEN N / A N / A

WF_SECURITY_WEP_40 шестигранная 4 , 5 ключей байт

WF_SECURITY_WEP_104 шестигранная 4 , 13 клавиши байт

WF_SECURITY_WPA_WITH_KEY шестигранная 32 байта

WF_SECURITY_WPA_WITH_PASS_PHRASE ASCII 8-63 символов ASCII

WF_SECURITY_WPA2_WITH_KEY шестигранная 32 байта

WF_SECURITY_WPA2_WITH_PASS_PHRASE ASCII 8-63 символов ASCII

WF_SECURITY_WPA_AUTO_WITH_KEY шестигранная 32 байта

WF_SECURITY_WPA_AUTO_WITH_PASS_PHRASE ASCII 8-63 символов ASCII

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**CPID** соединения Анкета ID

**SecurityType** Значение, соответствующее типу желаемого безопасности .

**wepKeyIndex** от 0 до 3 (используется только если тип защиты является
WF_SECURITY_WEP_40 или

WF_SECURITY_WEP_104 )

**p_securityKey** двоичной ключ или фраза (не используется , если безопасность WF_SECURITY_OPEN )

**securityKeyLength** Количество байтов в p_securityKey (не используется , если безопасность

WF_SECURITY_OPEN )

//*-*-*-*-*----------

**11.1.1.17 WF_CPSetSsid Функция**

**файл**

WFApi.h

C

```
void WF_CPSetSsid (
UINT8 CPID ,
UINT8 * p_ssid ,
UINT8 ssidLength
) ;
```

**Возвращает**

Ни один .

**описание**

Устанавливает SSID и SSID длина элементов в профиль подключения . Обратите внимание, чтоточка доступа может иметь либо видимым или скрытый SSID . Если точка доступа используется скрытый SSID затем активное сканирование должны использоваться (см. ScanType поле в Соединения Алгоритм ) .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**CPID** соединения Анкета ID

**p_ssid** Указатель на SSID строки

**ssidLength** Количество байтов в SSID

//*-*-*-*-*----------

### 11.1.1.18 WFCPElementsStruct Структура

**файл**

WFApi.h

C

структура WFCPElementsStruct {

UINT8 SSID [ WF_MAX_SSID_LENGTH ] ;

UINT8 BSSID [ WF_BSSID_LENGTH ] ;

UINT8 ssidLength ;

UINT8 SecurityType ;

UINT8 SecurityKey [ WF_MAX_SECURITY_KEY_LENGTH ] ;

UINT8 securityKeyLength ;

UINT8 wepDefaultKeyId ;

UINT8 NETWORKTYPE ;

UINT8 adHocBehavior ;

UINT8 hiddenSSID ;

UINT8 wepKeyType ;

} ;

**Пользователи**

**Пользователи Описание**

**UINT8 SSID [ WF_MAX_SSID_LENGTH ]** ; SSID , которое должно быть меньше или равно 32 символов. Установите все 0 если не используется . Если ssidLength 0 это поле игнорируется. Если SSID не определяется то MRF24W , при использовании этого профиля для подключения (см.страница 171 ) , будет сканировать все каналы , входящие в его региональном домене .

По умолчанию: SSID не используется.

**UINT8 BSSID [ WF_BSSID_LENGTH ]** ; Basic Service Set Identifier , всегда 6 байт . Это 48-битный MAC из SSID . Это необязательное поле, может быть использован для указания

конкретного SSID , если более одного AP существует с тем же SSID. Это поле может Также можно использовать вместо части SSID.

Установите каждый байт 0xFF , если BSSID не собирается использовать . По умолчанию:

BSSID не используется ( все раз FF )

**UINT8 ssidLength** ; Количество ASCII байты в SSID. Установите в 0 является SSID не будет

используется .

По умолчанию: 0

**UINT8 SecurityType ;** Определяет требуемый уровень безопасности для связи . Возможны следующие варианты :

**UINT8**

**SecurityKey [ WF_MAX_SECURITY_KEY_LENGTH ] ;** Установите в NULL если SecurityType является WF_SECURITY_OPEN . Если securityKeyLength 0 это поле игнорируется.

**UINT8 securityKeyLength** ; Количество байтов, используемых в SecurityKey . Установите в 0 , если SecurityType является WF_SECURITY_OPEN .

**UINT8 wepDefaultKeyId** ; Это поле используется, только если SecurityType является WF_SECURITY_WEP . Это поле обозначает , какой из четырех ключей WEP , определенных в SecurityKey к использовать при подключении к сети Wi-Fi . Диапазон составляет от 0 до 3 , с по умолчанию быть 0 .

**UINT8 NETWORKTYPE** ; WF_INFRASTRUCTURE или WF_ADHOC

По умолчанию: WF_INFRASTRUCTURE

**UINT8 adHocBehavior** ; Применяется только если NETWORKTYPE является WF_ADHOC . Настраивает ADHOC поведение . Возможны следующие варианты :

**UINT8 hiddenSSID** ; 1 - включить скрытый SSID в специальном режиме

**UINT8 wepKeyType** ; 0 - общий ключ , 1 - открытый ключ

**описание**

Структура профиля соединения элементов

//*-*-*-*-*----------

**11.1.1.19 WF_CPGetSsidType Функция**

**файл**

WFApi.h

C

```
void WF_CPGetSsidType (
UINT8 CPID ,
UINT8 * hidden
) ;
```

**Возвращает**

Ни один .

**описание**

Получает элемент типа SSID в профиль подключения .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

скрытый Указатель типа SSID

//*-*-*-*-*----------

### 11.1.1.20 WF_CPSetSsidType Функция

**файл**

WFApi.h

C

```
void WF_CPSetSsidType (
UINT8 CPID ,
Uint8 hidden
```

) ;

**Возвращает**

Ни один .

**описание**

Устанавливает элемент типа SSID в профиль подключения . Это справедливо только тогда, когда мы создаем AdHoc сети .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

//*-*-*-*-*----------

**11.1.1.21 WF_CPGetWepKeyType Функция**

**файл**

WFApi.h

C

void WF_CPGetWepKeyType (

UINT8 CPID ,

UINT8 * p_wepKeyType

) ;

**Возвращает**

Ни один .

**описание**

Получает элемент Тип сети профиля подключения . Допустимые значения:

• WF_SECURITY_WEP_SHAREDKEY

• WF_SECURITY_WEP_OPENKEY

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

//*-*-*-*-*----------

### 11.1.1.22 WF_CPGetWPSCredentials Функция

**файл**

WFApi.h

C

```
void WF_CPGetWPSCredentials (
UINT8 CPID ,
tWFWpsCred * p_cred
) ;
```

**Возвращает**

Ни один .

**описание**

Получает WPS полномочия после WPS завершена

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

p_cred Указатель на полномочиях

//*-*-*-*-*----------

### 11.1.1.23 WF_CPSetWepKeyType Функция

**файл**

WFApi.h

C

void WF_CPSetWepKeyType (

UINT8 CPID ,

UINT8 wepKeyType

) ;

**Возвращает**

Ни один .

**описание**

Задает тип ключа WEP для указанного соединения Профиль ID .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

//*-*-*-*-*----------

### 11.1.2 Подключение профиля Внутренние Пользователи

**Функции**

**Имя Описание**

**LowLevel_CPGetElement** (см. стр. 646 )Получить элемент соединительного профиля на MRF24W .

**LowLevel_CPSetElement** (

см. стр. 647 )

Установите элемент соединительного профиля на MRF24W .

**модуль**

Wi-Fi соединения профиля (см. стр. 631 )

**описание**

//*-*-*-*-*----------

#### 11.1.2.1 LowLevel_CPGetElement Функция

**файл**

WFConnectionProfile.c

C

статической силы LowLevel_CPGetElement (

UINT8 CPID ,

UINT8 elementId ,

UINT8 * p_elementData ,

UINT8 elementDataLength ,

UINT8 dataReadAction

) ;

**Возвращает**

Ни один .

**описание**

Все соединения профиля " Получить элемент» функции вызывают эту функцию для построения сообщение управления . Вызывающий должен исправить

любые вопросы прямой порядок байт до вызова этой функции .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

elementId Элемент , который был прочитан

p_elementData Указатель на котором будет написано данных элементов

elementDataLength Количество байтов данных элементов, которые будут прочитаны

dataReadAction Если TRUE , то читать данные на paramters и бесплатно Упр ответного буфера . Если ЛОЖЬ

затем вернуться после ответа получил , не читайте никаких данных , как вызывающей сделает

что , и не свободного буфера , как абонент будет сделать это также .

//*-*-*-*-*----------

**11.1.2.2 LowLevel_CPSetElement Функция**

**файл**

WFConnectionProfile.c

C

статической силы LowLevel_CPSetElement (

UINT8 CPID ,

UINT8 elementId ,

UINT8 * p_elementData ,

UINT8 elementDataLength

) ;

**Возвращает**

Ни один .

**описание**

МЕСТНЫЕ прототипы функций

Все соединения профиля функции ' набор элементов ' вызвать эту функцию для построения сообщение управления . Вызывающий должен исправить

любые вопросы прямой порядок байт до вызова этой функции .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

CPID соединения Анкета ID

elementId Элемент , что создается

p_elementData Указатель на данные элемента

elementDataLength Количество байт , на который указывает p_elementData

//*-*-*-*-*----------

### 11.1.3 Макросы

**макрос**

**Имя Описание**

RAW_MGMT_RX_ID (см.

страница 648 )

Это макро RAW_MGMT_RX_ID .

**модуль**

Wi-Fi соединения профиля (см. стр. 631 )

//*-*-*-*-*----------

#### 11.1.3.1 RAW_MGMT_RX_ID Макро

**файл**

WFConnectionProfile.c

C

# определить RAW_MGMT_RX_ID RAW_RX_ID

**описание**

Это макро RAW_MGMT_RX_ID .

//*-*-*-*-*----------

## 11.2 Wi-Fi Алгоритм соединения

**модуль**

Wi-Fi интерфейс API (см. стр. 628 )

**описание**

Алгоритм соединение используется для тонкой настройкиалгоритма MRF24WB0M используется в процессе соединения. связь

Алгоритм может быть изменен только при MRF24WB0M не подключен к сети 802.11 .

//*-*-*-*-*----------

**11.2.1 соединения Алгоритм общественного Пользователи**

**Функции**

**Имя Описание**

**WF_CAGetBeaconTimeout** (см.страница 650 ) Читает значение тайм-аута маяк .

**WF_CAGetBeaconTimeoutAction**(См. Стр. 650 ) Читает соединение алгоритм действий ожидания маяк .

**WF_CAGetChannelList** (см.страница 651 ) Получает список каналов .

**WF_CAGetConnectionProfileList**(См. Стр. 651 ) В настоящее время не поддерживается

**WF_CAGetDeauthAction** (см.страница 652) Читает соединение алгоритм deauth действие.

**WF_CAGetElements** (см. стр.652) Читает все элементы Алгоритм соединения .

**WF_CAGetEventNotificationAction**(См. Стр. 653 ) Читает соединение алгоритм событий действия уведомления .

**WF_CAGetListenInterval** (см.страница 654) Получает слушать (см. стр. 175 ) интервал .

**WF_CAGetListRetryCount** (см.страница 654) Получает список число попыток

**WF_CAGetMaxChannelTime** (см. стр. 655 ) Получает Макс Channel Время (в миллисекундах)

**WF_CAGetMinChannelTime** (см. стр. 655 ) Получает текущий соединение алгоритм минимального времени канала.

**WF_CAGetProbeDelay** (см.страница 656 ) Получает задержки пробного (в микросекундах)

**WF_CAGetRssi** (см. стр. 656 ) Получает порог RSSI

**WF_CAGetScanCount** (см.страница 657 ) Получает количество сканирования

**WF_CAGetScanType** (см. стр.657) Получает соединение алгоритм тип сканирования

**WF_CASetBeaconTimeout** (см.страница 658 ) Устанавливает значение тайм-аута маяк .

**WF_CASetBeaconTimeoutAction**(См. Стр. 658 ) Действие предпринять, еслисоединение потеряно из-за тайм-аута маяка.

**WF_CASetChannelList** (см.страница 659 ) Устанавливает список каналов .

**WF_CASetConnectionProfileList**(См. Стр. 659 ) В настоящее время не поддерживается

**WF_CASetDeauthAction** (см.страница 660 ) Устанавливает DeauthAction используемой соединением алгоритма .

**WF_CASetElements** (см. стр.660) Записывает все элементы Алгоритм соединения .

**WF_CASetEventNotificationAction**(См. Стр. 661 ) Устанавливает WiFi события , что хозяин желает получать уведомления о .

**WF_CASetListenInterval** (см.страница 662 ) Устанавливает слушать (см. стр. 175 ) интервал .

**WF_CASetListRetryCount** (см.страница 662 ) Устанавливает список число попыток

**WF_CASetMaxChannelTime** (см. стр. 663 ) Устанавливает максимальное время канала (в миллисекундах)

**WF_CASetMinChannelTime** (см. стр. 663 ) Устанавливает время минимальная канала (в миллисекундах)

**WF_CASetProbeDelay** (см.страница 664 ) Устанавливает задержку зонд (в микросекундах)

**WF_CASetRssi** (см. стр. 664 ) Установка порога RSSI

**WF_CASetScanCount** (см.страница 665 ) Устанавливает количество сканирования

**WF_CASetScanType** (см. стр.665) Устанавливает соединение алгоритмически типа сканирования

**WF_CAGetDtimInterval** (см.страница 668 ) Получает интервала DTIM .

**WF_CASetDtimInterval** (см.страница 668 ) Устанавливает интервала DTIM .

**модуль**

Wi-Fi соединения Алгоритм (см. стр. 648 )

**структуры**

**Имя Описание**

**WFCAElementsStruct** (см.страница 666) Соединения Алгоритм Элементы

**описание**

Следующие функции и переменные доступны для применения стека.

//*-*-*-*-*----------

### 11.2.1.1 WF_CAGetBeaconTimeout Функция

**файл**

WFApi.h

C

void WF_CAGetBeaconTimeout (

UINT8 * p_beaconTimeout

) ;

**Возвращает**

Ни один .

**описание**

Получает Маяк вышло используется подключение алгоритма .

**Значение Описание**

0 Нет мониторинг состояния ожидания маяк . Хозяин не будет уведомлен об этом событии .

1-255 Количество маркеров, пропущенных перед отключения случае происходит и beaconTimeoutAction происходит . Если включено, хозяин будет получать сообщение о событии , временно или постоянно теряется , показывающее связь , и если повторная попытка , связь успешное мероприятие .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_beaconTimeout Указатель где написано значение тайм-аута маяк

//*-*-*-*-*----------

### 1.2.1.2 WF_CAGetBeaconTimeoutAction Функция

файл

WFApi.h

C

void WF_CAGetBeaconTimeoutAction (

UINT8 * p_beaconTimeoutAction

) ;

**Возвращает**

Ни один .

**описание**

Возвращает действие Маяк вышло используется подключение алгоритма .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_beaconTimeoutAction Указатель где написано возвращенное значение . значение

будет либо • WF_ATTEMPT_TO_RECONNECT

• WF_DO_NOT_ATTEMPT_TO_RECONNECT

//*-*-*-*-*----------

### 11.2.1.3 WF_CAGetChannelList Функция

**файл**

WFApi.h

C

```
void WF_CAGetChannelList (
UINT8 * p_channelList ,
UINT8 * p_numChannels
) ;
```

**Возвращает**

Ни один .

**описание**

Получает список каналов , используемой соединением алгоритма .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_channelList Указатель где список каналов будут возвращены

p_numChannels Указатель где будут возвращены число каналов в списке

//*-*-*-*-*----------

### 11.2.1.4 WF_CAGetConnectionProfileList Функция

**файл**

WFApi.h

C

```
void WF_CAGetConnectionProfileList (
UINT8 cpList [ WF_CP_LIST_LENGTH ]
) ;
```

**Возвращает**

Ни один

**описание**

В настоящее время не поддерживается

**Замечания**

В настоящее время не поддерживается. Размер список всегда WF_CP_LIST_SIZE .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

cpList Массив профиль подключения идентификаторов используется для создания списка CP

//*-*-*-*-*----------

### 11.2.1.5 WF_CAGetDeauthAction Функция

файл

WFApi.h

C

```
void WF_CAGetDeauthAction (
UINT8 * p_deauthAction
) ;
```

**Возвращает**

Ни один .

**описание**

Получает DeauthAction используемой соединением алгоритма .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_deauthAction Указатель где написано возвращенное значение . Значение будет

быть либо • WF_ATTEMPT_TO_RECONNECT

• WF_DO_NOT_ATTEMPT_TO_RECONNECT

//*-*-*-*-*----------

### 11.2.1.6 WF_CAGetElements Функция

файл

WFApi.h

C

```
void WF_CAGetElements (
```

tWFCAElements * p_elements

) ;

**Возвращает**

Ни один

**описание**

Отправляет сообщение в MRF24W в котором содержится просьба все элементы Алгоритм соединения .

**Замечания**

Ни один

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_elements указатель на структуру вывода ( tWFCAElements ), где соединение

элементы алгоритм написаны .

//*-*-*-*-*----------

**11.2.1.7 WF_CAGetEventNotificationAction Функция**

файл

WFApi.h

C

void WF_CAGetEventNotificationAction (

UINT8 * p_eventNotificationAction

) ;

**Возвращает**

Ни один .

**описание**

Возвращает действие уведомление о событии , используемой соединением алгоритма . Значение считывается будет немного маска,

соответствует следующей таблице:

Бит событие

0 WF_NOTIFY_CONNECTION_ATTEMPT_SUCCESSFUL

1 WF_NOTIFY_CONNECTION_ATTEMPT_FAILED

2 WF_NOTIFY_CONNECTION_TEMPORARILY_LOST

3 WF_NOTIFY_CONNECTION_PERMANENTLY_LOST

4 WF_NOTIFY_CONNECTION_REESTABLISHED

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_eventNotificationAction Указатель где написано возвращенное значение .

**//*-*-*-*-*----------**

### 11.2.1.8 WF_CAGetListenInterval Функция

**файл**

WFApi.h

C

void WF_CAGetListenInterval (

UINT16 * p_listenInterval

) ;

**Возвращает**

Ни один .

**описание**

Получает Слушайте интервал используется подключение алгоритма . Это значение измеряется в 100 мс интервалами, маяк по умолчанию

Период точек доступа.

**Значение Описание**

1 MRF24W просыпается каждые 100 мс для получения BUFFERED сообщения .

2 MRF24W просыпается каждые 200 мс для получения BUFFERED сообщения .

.....

65535 MRF24W просыпается каждые 6535,5 секунд ( ~ 109 минут) , чтобы получить BUFFERED сообщения .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь. Используется, если выбран PS Опрос включена .

**Параметры**

**Параметры Описание**

p_listenInterval Указатель где слушать (см. стр. 175 ) интервал возвращается

//*-*-*-*-*----------

**11.2.1.9 WF_CAGetListRetryCount Функция**

файл

WFApi.h

C

```
void WF_CAGetListRetryCount (
UINT8 * p_listRetryCount
) ;
```

**Возвращает**

Ни один

**описание**

См. описание в WF_CASetListRetryCount (см. стр. 662 ) ( )

**Замечания**

Ни один

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_listRetryCount Указатель где написано количество список повтора .

//*-*-*-*-*----------

**11.2.1.10 WF_CAGetMaxChannelTime Функция**

**файл**

WFApi.h

C

```
void WF_CAGetMaxChannelTime (
UINT16 * p_minChannelTime
) ;
```

**Возвращает**

Ни один

**описание**

Получает максимальное время диспетчер соединений ожидает ответа зонда после отправки пробный запрос .

**Замечания**

По умолчанию 400 мс

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_maxChannelTime Указатель где написано максимальное время канал

//*-*-*-*-*----------

**11.2.1.11 WF_CAGetMinChannelTime Функция**

**файл**

WFApi.h

C

void WF_CAGetMinChannelTime (

UINT16 * p_minChannelTime

) ;

**Возвращает**

Ни один

**описание**

Получает минимальное время диспетчер соединений ожидает ответа зонда после отправки пробный запрос .

**Замечания**

Ни один

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_minChannelTime Указатель где минимальное время ожидания ответа зонда (в миллисекундах) будет

записать .

//*-*-*-*-*----------

### 11.2.1.12 WF_CAGetProbeDelay Функция

**файл**

WFApi.h

C

void WF_CAGetProbeDelay (

UINT16 * p_probeDelay

) ;

**Возвращает**

Ни один

**описание**

Отложить до передачи пробный запрос после события смены каналаколичество микросекунд
.

**Замечания**

По умолчанию США за 20

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_probeDelay Указатель где написано задержка зонда

//*-*-*-*-*----------

**11.2.1.13 WF_CAGetRssi Функция**

**файл**

WFApi.h

C

void WF_CAGetRssi (

UINT8 * p_rssi

) ;

**Возвращает**

Ни один

**описание**

Смотреть WF_CASetRssi (см. стр. 664 ) . Обратите внимание, что эта функция обрабатывает только извлекает порог RSSI используется во время соединения

- Это не ток ПУПС существующего соединения . Если желательно получить текущее состояние RSSI тосканирование должно быть

выполняется и результат сканирования будет содержать текущее состояние RSSI .

**Замечания**

По умолчанию 255

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_rssi Указатель где написано значение RSSI

//*-*-*-*-*----------

**11.2.1.14 WF_CAGetScanCount Функция**

**файл**

WFApi.h

C

void WF_CAGetScanCount (

UINT8 * p_scanCount

) ;

**Возвращает**

Ни один

**описание**

Количество разДиспетчер подключений будет сканировать канал , пытаясь найти определенную сеть WiFi .

**Замечания**

По умолчанию 1

Предпосылки

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_scanCount Указатель где написано количество выполненных операций сканирования

//*-*-*-*-*----------

### 11.2.1.15 WF_CAGetScanType Функция

**файл**

WFApi.h

C

```
void WF_CAGetScanType (
UINT8 * p_scanType
) ;
```

**Возвращает**

Ни один

**описание**

Считывает текущий соединение алгоритм тип сканирования .

**Замечания**

Активность сканирования вызывает MRF24W отправить пробные запросы . Пассивный сканирования подразумеваетMRF24W слушает только для маяки . По умолчанию WF_ACTIVE_SCAN .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_scanType Указатель где написано соединения Алгоритм тип сканирования .

//*-*-*-*-*----------

### 11.2.1.16 WF_CASetBeaconTimeout Функция

**файл**

WFApi.h

C

```
void WF_CASetBeaconTimeout (
UINT8 beaconTimeout
```

) ;

**Возвращает**

Ни один .

**описание**

Устанавливает Beacon Тайм-аут , используемой соединением алгоритма .

**Значение Описание**

0  Нет мониторинг состояния ожидания маяк . Хозяин не будет уведомлен об этом событии .

1-255 Количество маркеров, пропущенных перед отключения случае происходит и beaconTimeoutAction происходит . Если включено, хозяин будет получать сообщение о событии , временно или постоянно теряется , показывающее связь , и если повторная попытка , связь успешное мероприятие .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**beaconTimeout** Количество маркеров, которые можно пропустить , прежде чем действия в

beaconTimeoutAction берется .

//*-*-*-*-*----------

### 11.2.1.17 WF_CASetBeaconTimeoutAction Функция

файл

WFApi.h

C

void WF_CASetBeaconTimeoutAction (

UINT8 beaconTimeoutAction

) ;

**Возвращает**

Ни один .

**описание**

Устанавливает Action Маяк тайм-аута , используемой соединением алгоритма .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

beaconTimeoutAction Действие взять , если соединение теряется из-за

тайм-аута маяка. Выбор либо

• WF_ATTEMPT_TO_RECONNECT

• WF_DO_NOT_ATTEMPT_TO_RECONNECT

//*-*-*-*-*----------

**11.2.1.18 WF_CASetChannelList Функция**

**файл**

WFApi.h

C

```
void WF_CASetChannelList (
UINT8 * p_channelList ,
Uint8 numChannels
) ;
```

**Возвращает**

Ни один .

**описание**

Устанавливает список каналов , используемой соединением алгоритма .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**p_channelList** Указатель на канал список .

**numChannels** Число каналов в p_channelList . Если установлено в 0 ,MRF24W будет использовать все в силе каналы для текущего регионального домена .

//*-*-*-*-*----------

### 11.2.1.19 WF_CASetConnectionProfileList Функция

файл

WFApi.h

C

```
void WF_CASetConnectionProfileList (
UINT8 cpList [ WF_CP_LIST_LENGTH ]
) ;
```

**Возвращает**

Ни один

**описание**

В настоящее время не поддерживается

**Замечания**

В настоящее время не поддерживается. Размер список всегда WF_CP_LIST_SIZE . Список должен начать с индексом 0 . Неиспользуемые записи в списке

должен быть установлен на 0xff .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**cpList** Массив профиль подключения идентификаторов используется для создания списка CP

//*-*-*-*-*----------

### 11.2.1.20 WF_CASetDeauthAction Функция

файл

WFApi.h

C

void WF_CASetDeauthAction (

UINT8 deauthAction

) ;

**Возвращает**

Ни один .

**описание**

Действие взять , если соединение теряется из-за приема сообщения deauthentification от точки доступа .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

deauthAction Действие принять в случае деаутентификации .

Допустимые значения

• WF_ATTEMPT_TO_RECONNECT

• WF_DO_NOT_ATTEMPT_TO_RECONNECT

//*-*-*-*-*----------

### 11.2.1.21 WF_CASetElements Функция

**файл**

WFApi.h

C

void WF_CASetElements (

tWFCAElements * p_elements

) ;

**Возвращает**

Ни один

**описание**

Алгоритм соединения Функции

Отправляет сообщение в MRF24W которая устанавливает все элементы Алгоритм соединения .

**Замечания**

Ни один

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_**elements** Указатель на входной структуре ( tWFCAElements ), содержащей соединение

Алгоритм элементы .

//*-*-*-*-*----------

**11.2.1.22 WF_CASetEventNotificationAction Функция**

файл

WFApi.h

C

void WF_CASetEventNotificationAction (

UINT8 eventNotificationAction

) ;

**Возвращает**

Ни один .

**описание**

Устанавливает Action уведомление о событии , используемой соединением алгоритма . Битовая маска для допустимых записей выглядит следующим образом:

Бит событие

0 WF_NOTIFY_CONNECTION_ATTEMPT_SUCCESSFUL

1 WF_NOTIFY_CONNECTION_ATTEMPT_FAILED

2 WF_NOTIFY_CONNECTION_TEMPORARILY_LOST

3 WF_NOTIFY_CONNECTION_PERMANENTLY_LOST

4 WF_NOTIFY_CONNECTION_REESTABLISHED

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**eventNotificationAction** Битовая маска указывает , какие события хозяин хочет, чтобы его notifed из .

//*-*-*-*-*----------

**11.2.1.23 WF_CASetListenInterval Функция**

**файл**

WFApi.h

C

void WF_CASetListenInterval (

UINT16 listenInterval

) ;

**Возвращает**

Ни один .

**описание**

Устанавливает слушать (см. стр. 175 ) интервал , используемый Connection алгоритма . Это значение измеряется в 100 мс интервалах , по умолчанию период маяка точек доступа .

**Значение Описание**

1 MRF24W просыпается каждые 100 мс для получения BUFFERED сообщения .

2 MRF24W просыпается каждые 200 мс для получения BUFFERED сообщения .

.....

65535 MRF24W просыпается каждые 6535,5 секунд ( ~ 109 минут) , чтобы получить BUFFERED сообщения .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь. Используется, если выбран PS Опрос включена .

**Параметры**

**Параметры Описание**

**listenInterval** Количество 100 мс промежутках между случаях, когда MRF24W просыпается

получать BUFFERED сообщения из сети.

//*-*-*-*-*----------

**11.2.1.24 WF_CASetListRetryCount Функция**

**файл**

WFApi.h

C

void WF_CASetListRetryCount (

UINT8 listRetryCount

) ;

**Возвращает**

Ни один

**описание**

Количество раз для выбора профиля подключения Список до отказа от попытки подключения . Поскольку списки еще не поддерживается эта функция фактически задала несколько раздиспетчер подключений попытается подключиться (см. стр. 171 ) с текущий Профиль соединения до отказа .

**Замечания**

Ни один

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**listRetryCount** от 0 до 254 или WF_RETRY_FOREVER (255)

//*-*-*-*-*----------

**11.2.1.25 WF_CASetMaxChannelTime Функция**

**файл**

WFApi.h

C

void WF_CASetMaxChannelTime (

UINT16 MinChannelTime

) ;

**Возвращает**

Ни один

**описание**

Максимальное время (в миллисекундах) диспетчер соединений будет ждать ответа зонда после отправки пробный запрос . Если нет зонд ответы не будут получены в MaxChannelTime то диспетчер соединений будет перейти к следующему каналу , если таковые слева сканирования или бросить курить.

**Замечания**

По умолчанию 400 мс

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

MaxChannelTime Максимальное время ожидания пробного ответа (в миллисекундах)

//*-*-*-*-*----------

**11.2.1.26 WF_CASetMinChannelTime Функция**

**файл**

WFApi.h

C

```
void WF_CASetMinChannelTime (
UINT16 MinChannelTime
) ;
```

**Возвращает**

Ни один

**описание**

Минимальное время (в миллисекундах) диспетчер соединений будет ждать ответа зонда после отправки пробный запрос . Если нет зонд ответы не будут получены в MinChannelTime то диспетчер соединений будет перейти к следующему каналу , если таковые слева сканирования или бросить курить.

**Замечания**

По умолчанию составляет 200 мс

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

MinChannelTime Минимальное время ожидания пробного ответа (в миллисекундах)

//*-*-*-*-*----------

**11.2.1.27 WF_CASetProbeDelay Функция**

**файл**

WFApi.h

C

```
void WF_CASetProbeDelay (
UINT16 probeDelay
) ;
```

**Возвращает**

Ни один

**описание**

Отложить до передачи пробный запрос после события смены каналаколичество микросекунд .

**Замечания**

По умолчанию США за 20

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

probeDelay Желаемый задержка зонда

//*-*-*-*-*----------

**11.2.1.28 WF_CASetRssi Функция**

**файл**

WFApi.h

C

```
void WF_CASetRssi (
```

UINT8 RSSI

) ;

**Возвращает**

Ни один

**описание**

Определяет поведение RSSI при подключении . Это значение используется только тогда, когда 1 ) Текущий профиля соединения не определила SSID или BSSID 2)SSID определяется в текущем соединительного профиля и несколько точек доступа будут обнаружены с же SSID .

Значения: 0 : Подключите к первой сети найдено 1 - 254 ( MRF24WB ) , 1 - 128 ( MRF24WG ) : Подключайте (см. стр. 171 ) , чтобы сети , если RSSI больше или равно заданному значению 255 Подключение к высшей RSSI найдено

Обратите внимание, что RSSI является относительной величиной , без единиц - это не коррелирует с дБм.

**Замечания**

По умолчанию 255

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**scanCount** Желаемый Количество сканирования

//*-*-*-*-*----------

### 11.2.1.29 WF_CASetScanCount Функция

файл

WFApi.h

C

void WF_CASetScanCount (

UINT8 scanCount

) ;

**Возвращает**

Ни один

**описание**

Количество разДиспетчер подключений будет сканировать канал , пытаясь найти определенную сеть WiFi .

**Замечания**

По умолчанию 1

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**scanCount** Желаемый Количество сканирования

//*-*-*-*-*----------

**11.2.1.30 WF_CASetScanType Функция**

**файл**

WFApi.h

C

```
void WF_CASetScanType (
UINT8 ScanType
) ;
```

**Возвращает**

Ни один

**описание**

Настраивает алгоритм подключения для нужного типа сканирования.

**Замечания**

Активность сканирования вызывает MRF24W отправить пробные запросы . Пассивный сканирования подразумеваетMRF24W слушает только для маяки . По умолчанию WF_ACTIVE_SCAN .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**ScanType** Желаемый тип сканирования . Либо WF_ACTIVE_SCAN или WF_PASSIVE_SCAN .

//*-*-*-*-*----------

**11.2.1.31 WFCAElementsStruct Структура**

файл

WFApi.h

C

структура WFCAElementsStruct {

UINT16 listenInterval ;

UINT8 ScanType ;

UINT8 RSSI ;

UINT8 connectionProfileList [ WF_CP_LIST_LENGTH ] ;

UINT8 listRetryCount ;

UINT8 eventNotificationAction ;

UINT8 beaconTimeoutAction ;

UINT8 deauthAction ;

UINT8 Channellist [ WF_CHANNEL_LIST_LENGTH ] ;

UINT8 numChannelsInList ;

UINT8 beaconTimeout ;

UINT8 scanCount ;

UINT8 pad1 ;

UINT16 MinChannelTime ;

UINT16 MaxChannelTime ;

UINT16 probeDelay ;

UINT16 dtimInterval ;

UINT16 beaconPrd ;

} ;

**Пользователи**

**Пользователи Описание**

**UINT16 listenInterval** ; Этот параметр используется только при режиме PS Опрос включена . Посмотреть WF_PsPollEnable (см. стр. 684) ( ) . Количество 100 мс промежутках между случаи, когда MRF24W просыпается в полученных буферных сообщений из сети. Диапазон составляет от 1 ( 100 мс ) , чтобы 6553,5 сек ( ~ 109 мин).

Обратите внимание, что стандарт 802.11 определяет слушать (см. стр. 175 ) интервал с точки зрения периодов Beacon , которые обычно 100 мс . Если MRF24W является общения к сети с сетью , которая имеет периодов Beacon , что является не 100 мс он будет округлить вверх (или вниз ) по мере необходимости в соответствии с фактической Beacon Период , насколько это возможно .

Важное примечание: Если listenInterval изменяется, пока подключен к сетиMRF24W автоматически подключится к сети с Новое значение Beacon период . Это может привести к временной потере данных Пакеты .

**UINT8 ScanType** ; WF_ACTIVE_SCAN ( пробные запросы разосланы ) или WF_PASSIVE_SCAN( слушать (см. стр. 175 ) только )

По умолчанию: WF_ACTIVE_SCAN

**UINT8 RSSI** ; Определяет RSSI ограничения при подключении . Это поле используется, только если :

1 .Профиль соединения не определен SSID или BSSID или

2 .SSID определяется в профиль подключения и несколько ЗС обнаружил с тем же SSID.

**UINT8 connectionProfileList [ WF_CP_LIST_LENGTH ] ;**

Примечание: списки соединения профиля пока не поддерживаются . Этот массив должен

быть установлен на всех FF -х годов.

**UINT8 listRetryCount** ; Это поле используется для указания на количество попыток для одного соединения профиль , прежде чем принимать соединения потеряли действие.

Диапазон от 1 до 254 или WF_RETRY_FOREVER ( 255) По умолчанию 3

**UINT8 eventNotificationAction** ; Есть несколько событий подключения связанных , которые могут возникнуть . Хозяин имеет возможность получать уведомления (или нет) , когда некоторые из этих событий . Это Уведомление управления на местах событий для событий подключения связанных .

**UINT8 beaconTimeoutAction** ; Задает действиеДиспетчер подключений следует принимать , если соединение потеряли из-за Beacon тайм-аут. Если это поле установлено

WF_ATTEMPT_TO_RECONNECT то число попыток ограничено к значению в listRetryCount .

Варианты: WF_ATTEMPT_TO_RECONNECT или
WF_DO_NOT_ATTEMPT_TO_RECONNECT

По умолчанию: WF_ATTEMPT_TO_RECONNECT

**UINT8 deauthAction** ; Обозначает , что действиеДиспетчер подключений должны принять, если он получает Деаутентификации сообщение от точки доступа .

Если это поле установлено WF_ATTEMPT_TO_RECONNECT то число попытки ограничивается значением в listRetryCount .

Варианты: WF_ATTEMPT_TO_RECONNECT или
WF_DO_NOT_ATTEMPT_TO_RECONNECT

По умолчанию: WF_ATTEMPT_TO_RECONNECT

**UINT8 Channellist [ WF_CHANNEL_LIST_LENGTH ] ;**

Список из одного или более каналов, которыедолжны использовать MRF24W когда подключения или сканирования . Если numChannelsInList установлен в 0, то это параметр должен быть установлен в NULL.

По умолчанию: Все допустимые каналы для регионального домена MRF24W ( установлен на уровне производство) .

**UINT8 numChannelsInList ;** Количество каналов в Channellist . Если установлено значение 0, то MRF24W будет заполнить список со всеми действующими каналов для регионального домена .

По умолчанию:число действительных каналов для регионального области MRF24W (набор при изготовлении ) .

**UINT8 beaconTimeout ;** Определяет количество маяков , которые могут быть пропущены перед действием описано в beaconTimeoutAction берется .

**UINT8 scanCount** ; Для сканирования канал , пытаясь найти конкретныйколичество раз

Точка доступа .

По умолчанию: 1

**UINT16 MinChannelTime** ; Минимальное время (в миллисекундах) диспетчер соединений будет ждать зондировать реакцию после отправки пробный запрос . Если нет датчика ответы не являются получил в MinChannelTime то диспетчер соединений пойдет на следующий канал , если таковые остались для сканирования , или бросить курить.

По умолчанию: 200 мс

**UINT16 MaxChannelTime ;** Если ответ зонд получил в MinChannelTime то соединение Менеджер продолжит сбор дополнительных ответов зонда до MaxChannelTime прежде, чем идти на следующий канал в Channellist . Единицы находятся в миллисекундах.

По умолчанию: 400 мс

**UINT16 probeDelay** ; Отложить до передачи пробный запрос количество микросекунд после события смены канала .

По умолчанию: США за 20

**UINT16 dtimInterval ;** По умолчанию: 4

**UINT16 beaconPrd** ; По умолчанию: 100 (мс)

**описание**

Соединения Алгоритм Элементы

//*-*-*-*-*----------

**11.2.1.32 WF_CAGetDtimInterval Функция**

**файл**

WFApi.h

C

void WF_CAGetDtimInterval (

UINT16 * p_dtimInterval

) ;

**Возвращает**

Ни один .

**описание**

Получает интервала DTIM используемой соединением алгоритма .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь. Используется, если выбран PS Опрос включена .

//*-*-*-*-*----------

### 11.2.1.33 WF_CASetDtimInterval Функция

**файл**

WFApi.h

C

```
void WF_CASetDtimInterval (
UINT16 dtimInterval
) ;
```

**Возвращает**

Ни один .

**описание**

Устанавливает интервала DTIM используемой соединением алгоритма .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь. Используется, если выбран PS Опрос включена .

## 11.2.2 Подключение Алгоритм Внутренние Пользователи

**Функции**

**Имя Описание**

**LowLevel_CAGetElement** (см. стр. 669 ) Получить элемент алгоритма соединения на MRF24W .

**LowLevel_CASetElement** (см. стр. 669 ) Установите элемент алгоритма соединения на MRF24W .

**SetEventNotificationMask** (см. стр. 670 ) Устанавливает маску уведомления о событии .

**модуль**

Wi-Fi соединения Алгоритм (см. стр. 648 )

**описание**

Следующие функции и переменные определяются как внутренние, к модулю.

//*-*-*-*-*----------

### 11.2.2.1 LowLevel_CAGetElement Функция

**файл**

WFConnectionAlgorithm.c

C

статической силы LowLevel_CAGetElement (

UINT8 elementId ,

UINT8 * p_elementData ,

UINT8 elementDataLength ,

UINT8 dataReadAction

) ;

**Возвращает**

Ни один .

**описание**

Функция низкого уровня , чтобы отправить соответствующее сообщение управления к MRF24W чтобы получить соединение алгоритм элемент .

**Замечания**

Все соединения Алгоритм " Получить элемент» функции вызывают эту функцию для построения сообщение управления . Вызывающий должен исправить до любых вопросов с прямым порядком байтов после получения данных от этой функции .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**elementId** Элемент , который был прочитан

p_**elementData** Указатель на котором будет написано данных элементов

**elementDataLength** Количество байтов данных элементов, которые будут прочитаны

**dataReadAction** Если TRUE , то читать данные на paramters и бесплатно Упр ответного буфера . Если ЛОЖЬ затем вернуться после ответа получил , не читайте никаких данных , как вызывающей сделает что , и не свободного буфера , как абонент будет сделать это также .

//*-*-*-*-*----------

**11.2.2.2 LowLevel_CASetElement Функция**

**файл**

WFConnectionAlgorithm.c

C

статической силы LowLevel_CASetElement (

UINT8 elementId ,

UINT8 * p_elementData ,

UINT8 elementDataLength

) ;

**Возвращает**

Ни один .

**описание**

МЕСТНЫЕ прототипы функций

Функция низкого уровня , чтобы отправить соответствующее сообщение управления к MRF24W установить соединение алгоритм элемент .

**Замечания**

Все подключения функции алгоритма ' набор элементов ' вызвать эту функцию для построения сообщение управления . Вызывающий должен исправить

до любых вопросов с прямым порядком байтов до вызова этой функции .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**elementId** Элемент , что создается

p_**elementData** Указатель на данные элемента

elementDataLength Количество байт , на который указывает p_elementData

//*-*-*-*-*----------

**11.2.2.3 SetEventNotificationMask Функция**

**файл**

WFConnectionAlgorithm.c

C

статической силы SetEventNotificationMask (

UINT8 eventNotificationBitMask

) ;

**Возвращает**

Ни один .

**описание**

Устанавливает маску уведомления о событиях для подключения Алгоритм . Допустимые значения:

**Значение Событие**

0x01 WF_NOTIFY_CONNECTION_ATTEMPT_SUCCESSFUL

0x02 WF_NOTIFY_CONNECTION_ATTEMPT_FAILED

0x04 WF_NOTIFY_CONNECTION_TEMPORARILY_LOST

0x08 WF_NOTIFY_CONNECTION_PERMANENTLY_LOST

0x10 WF_NOTIFY_CONNECTION_REESTABLISHED

0x1f WF_NOTIFY_ALL_EVENTS

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**eventNotificationBitMask** Битовая маска определении того, какие события хозяин будет уведомлен о .

### 11.3 Wi-Fi Диспетчер подключений

**модуль**

Wi-Fi интерфейс API (см. стр. 628 )

**описание**

Диспетчер подключения использует алгоритм подключения и один или несколько профилей подключения для подключения (см. стр. 171) , чтобы сеть .

#### 11.3.1 Менеджер соединения открытые члены

**Функции**

**Имя Описание**

**WF_CMConnect** (см. стр.671 ) Дает команду MRF24W установить соединение .

**WF_CMDisconnect** (см.страница 672) Дает команду MRF24W , чтобы закрыть все открытые соединения и / или прекратить попытке подключения (см. стр. 171) .

**WF_CMGetConnectionState**(См. Стр. 673 ) Возвращает текущее состояние соединения.

**WF_CMInfoGetFSMStats** (см. стр. 673 ) СМ Инфо Функции

**WF_CMGetConnectContext** (См. Стр. 673 ) Получает контекст соединения WF

**WF_GetTxMode** (см. стр.674) Получает TX значение режима

**WF_SetStackVersion** (см.страница 674 ) Можно назвать уведомить версию стека РФ FW

**WF_SetTxMode** (см. стр.674) Настраивает 802,11 Парам.отпр

**WFEnableDebugPrint** (см. страница 675) Можно назвать , чтобы позволить printfs для WPS и P2P

**модуль**

**Wi-Fi Connection Manager** ( см. стр. 671 )

**описание**

Следующие функции и переменные доступны для применения стека.

//*-*-*-*-*----------

### 11.3.1.1 WF_CMConnect Функция

**файл**

WFApi.h

C

```
void WF_CMConnect (
UINT8 CPID
) ;
```

**Возвращает**

Ни один .

**описание**

Функции диспетчера подключений

Направляет диспетчер подключений для поиска и подключения (см. стр. 171 ) к сети Wi-Fi . Эта функция не ждет пока попытка соединения не будет успешным , но сразу же возвращается . Смотреть WF_ProcessEvent (см. стр. 698 ) для событий, которые может произойти в результате попытки подключения на успех или нет.

Заметим, что если Профиль соединения используются имеет WPA или WPA2 включен безопасность и использует фразы , Диспетчер соединений сначала вычислить ключ PSK , а затем начать процесс подключения. Ключ расчет может занять до 30 секунд.

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**CPID** Если это значение равно существующего соединительного профиля в ID , чем только что Подключение профиля будет использоваться для попытки подключения к сети Wi-Fi . Если это значение установлено в WF_CM_CONNECT_USING_LIST то connectionProfileList будет использоваться для подключения (см. стр. 171 ) , начиная с Первый профиля соединения в списке .

//*-*-*-*-*----------

### 11.3.1.2 WF_CMDisconnect Функция

**файл**

WFApi.h

C

UINT16 WF_CMDisconnect ( ) ;

**Возвращает**

Итоги работы . Успех или неудача

**описание**

Направляет диспетчер подключений , чтобы закрыть все открытые попытки подключения или соединения в прогресс. Нет дальнейшие попытки подключения (см. стр. 171 ) не будут приняты , пока WF_CMConnect (см. стр. 671 ) () вызывается . Создает событие WF_EVENT_CONNECTION_PERMANENTLY_LOST когда соединение успешно завершена .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван .

//*-*-*-*-*----------

### 11.3.1.3 WF_CMGetConnectionState Функция

**файл**

WFApi.h

C

void WF_CMGetConnectionState (

UINT8 * p_state ,

UINT8 * p_currentCpId

) ;

**Возвращает**

Ни один .

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**p_state** Указатель на месте, где состояние соединения будет написано

**p_currentCpId** Указатель на месте текущего профиля подключения ID , что происходит обращение .

//*-*-*-*-*----------

### 11.3.1.4 WF_CMInfoGetFSMStats Функция

**файл**

WFApi.h

C

```
void WF_CMInfoGetFSMStats (
tWFCMInfoFSMStats * p_info
) ;
```

**описание**

CM Инфо Функции

//*-*-*-*-*----------

### 11.3.1.5 WF_CMGetConnectContext Функция

**файл**

WFApi.h

C

```
void WF_CMGetConnectContext (
tWFConnectContext * p_ctx
```

) ;

**Возвращает**

Ни один .

**описание**

Получить контекст соединения

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

//*-*-*-*-*----------

### 11.3.1.6 WF_GetTxMode Функция

**файл**

WFApi.h

C

void WF_GetTxMode (

UINT8 * режим

) ;

**Возвращает**

Ни один .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

//*-*-*-*-*----------

### 11.3.1.7 WF_SetStackVersion Функция

**файл**

WFApi.h

C

```
void WF_SetStackVersion (
UINT8 майор,
UINT8 несовершеннолетнего
) ;
```

**Возвращает**

Ни один .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

```
//*-*-*-*-*-------------------------------------------
```

### 11.3.1.8 WF_SetTxMode Функция

**файл**

WFApi.h

C

```
void WF_SetTxMode (
режим UINT8
) ;
```

**Возвращает**

Ни один .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

Режим - Tx значение режима . Варианты выбора WF_TXMODE_G_RATES (по умолчанию) - будет использовать все 802.11g цены

WF_TXMODE_B_RATES - будет использовать только цены 802.11b

WF_TXMODE_LEGACY_RATES - будет использовать только 1 и 2 Мбит ставки

//*-*-*-*-*----------

### 11.3.1.9 WFEnableDebugPrint Функция

**файл**

WFApi.h

C

void WFEnableDebugPrint (

опция UINT8

) ;

**Возвращает**

Ни один .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

//*-*-*-*-*----------

## 11.4 Wi-Fi сканирования

**модуль**

Wi-Fi интерфейс API (см. стр. 628 )

**описание**

Если приложение уже знает сетевой SSID , что он хочет присоединиться , чем он может настроить профиль подключения с этим информация и попытка присоединиться к сети . Тем не менее, существуют приложения, в первую очередь необходимо динамически определить, что инфраструктуры и / или Adhoc сети находятся в этом районе, а затем решить, к какой

сети зарегистрироваться . Функции сканирования API являются используется для сбора этой информации.

### 11.4.1 Поиск открытые члены

**Функции**

**Имя Описание**

**WF_Scan** (см. стр. 676 ) дает команду MRF24W начать операцию сканирования . Это создаст

**WF_EVENT_SCAN_RESULTS_READY** событие.

**WF_ScanGetResult** (см.страница 677 ) Читайте результаты сканирования назад от MRF24W .

**модуль**

Wi-Fi сканирования (см. стр. 675 )

**описание**

Следующие функции и переменные доступны для применения стека.

//*-*-*-*-*----------

#### 11.4.1.1 **WF_Scan** Функция

**файл**

WFApi.h

C

```
UINT16 WF_Scan (
UINT8 CPID
) ;
```

**Возвращает**

Итоги работы . Успех или неудача

**описание**

Функции сканирования

Направляет MRF24W инициировать операцию сканирования с использованием входного соединения анкеты . Ведущее приложение будет уведомлен  что результаты проверки готовы, когда он получает событие WF_EVENT_SCAN_RESULTS_READY . Поле EventInfo для это событие будет содержать количество результатов сканирования. После того, как

результаты проверки готовы они могут быть получены с помощью WF_ScanGetResult (см. стр. 677 ) ( ) .

Результаты сканирования не сохраняются на MRF24W до :

1 . Вызов WF_Scan () еще раз (после Результаты сканирования вернулся из предыдущего вызова ) .

2 . Сбросить MRF24W .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**CPID** профиля соединения для использования. Если CPID действует то значения за что

Подключение профиля будет использоваться для фильтрации результатов сканирования . Если CPID установлен в

WF_SCAN_ALL ( 0xFF ), то фильтрпо умолчанию будет использоваться .

Действительно CPID

• Если CP имеет определенный SSID только результаты сканирования с этой SSID сохраняются.

• Если CP не имеют определенный SSID то все сканируется SSID будет сохраняется

• Только результаты сканирования от инфраструктуры или AdHoc сетей сохраняются , в зависимости от значения NETWORKTYPE в профиль подключения

• Список каналов , что сканируется будет определяться исходя из Channellist в Соединения Алгоритм (который должен быть определен перед вызовом этой функции ) .

CPID равна WF_SCAN_ALL

• Все результаты сканирования сохраняются ( как инфраструктура и Ad Hoc сети) .

• будут сканироваться все каналы в рамках регионального домена MRF24W в .

• Нет Профили подключений не должны быть определены перед вызовом этой функции .

•Подключение Алгоритм не должны быть определены до вызова этого функция.

//*-*-*-*-*----------

### 11.4.1.2 WF_ScanGetResult Функция

**файл**

WFApi.h

C

```
void WF_ScanGetResult (
UINT8 ListIndex ,
tWFScanResult * p_scanResult
) ;
```

**Возвращает**

Ни один .

**описание**

Послезавершения осмотра эта функция используется для чтения один или несколько из сканирования результатов от MRF24W . Результаты сканирования будут записаны непрерывно , начиная с p_scanResults (см. структуру tWFScanResult для формата результате сканирования ) .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь. WF_EVENT_SCAN_RESULTS_READY событий должен иметь уже occurrerd .

**Параметры**

**Параметры Описание**

Индекс ListIndex (0 на основе Список ), о въезде сканирования для получения .

**p_scanResult** Указатель на месте для хранения сканирования результата структуру

Получить RSSI MRF24WB : RSSI_MAX (200) , RSSI_MIN (106)

MRF24WG RSSI_MAX (128) , RSSI_MIN (43) p_scanResult - > RSSI

## 11.5 Wi-Fi Tx Управление питанием

**модуль**

Wi-Fi интерфейс API (см. стр. 628 )

**описание**

Функции API в этом разделе используются для настройки параметров управления мощностью MRF24WB0M Tx .

Приложение может контролировать Tx власть изменить власть макс Tx через WF_TxPowerSetMinMax (см. стр. 679 ) (), чтобы

нечто иное, чем 10 дБм .

### 11.5.1 Tx Управление питанием Общественный Пользователи

**Функции**

**Имя Описание**

**WF_TxPowerGetMinMax** (см. стр. 678 ) Получает Tx мин и максимальной мощности на MRF24W .

**WF_TxPowerSetMinMax** (см. стр. 679 ) Устанавливает Tx мин и максимальной мощности на MRF24W .

**WF_TxPowerGetFactoryMax**(См. Стр. 679 ) Получает заводское Максимальная мощность Tx от MRF24W .

**WF_TxPowerGetMax** (см.страница 680 ) Получает максимальной мощности Tx на MRF24WG0M .

**WF_TxPowerSetMax** (см.страница 680 ) Устанавливает максимальной мощности Tx на MRF24WG0M .

модуль

**Wi-Fi Tx Управление питанием** (см. стр. 678 )

**описание**

Следующие функции и переменные доступны для применения стека.

//*-*-*-*-*----------

#### 11.5.1.1 WF_TxPowerGetMinMax Функция

**файл**

WFApi.h

C

void WF_TxPowerGetMinMax (

INT8 * p_minTxPower ,

INT8 * p_maxTxPower

) ;

**Возвращает**

Ни один .

**описание**

После инициализации власть MRF24W макс Tx определяется заводе значение . Эта функция может установить другой минимум и максимальные уровни мощности Tx . Тем не менее, эта функция никогда не может установить максимальную Tx мощности, которая превышает на заводе-изготовителе значение, которое может быть считана с помощью WF_TxPowerGetFactoryMax (см. стр. 679 ) () .

**Замечания**

Никакого преобразования единиц , необходимых , вход MRF24W не в дБ .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_minTxPower Указатель на месте написатьminTxPower

p_maxTxPower Указатель на месте написать maxTxPower

//*-*-*-*-*----------

### 11.5.1.2 WF_TxPowerSetMinMax Функция

**файл**

WFApi.h

C

void WF_TxPowerSetMinMax (

INT8 minTxPower ,

INT8 maxTxPower

) ;

**Возвращает**

Ни один .

**описание**

! определена ( MRF24WG )

! определена ( MRF24WG )

********** **********

После инициализации власть MRF24W макс Tx определяется заводе значение . Эта функция может установить другой минимум и максимальные уровни мощности Tx . Тем не менее, эта функция никогда не может установить максимальную Tx мощности, которая превышает на заводе-изготовителе значение, которое может быть считана с помощью **WF_TxPowerGetFactoryMax** (см. стр. 679 ) () .

**Замечания**

Никакого преобразования единиц , необходимых , вход MRF24W не в дБ .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**minTxPower** Желаемый minTxPower ( от -10 до 10 дБ )

**maxTxPower** Желаемый maxTxPower (от -10 до 10 дБ )

//*-*-*-*-*----------

### 11.5.1.3 WF_TxPowerGetFactoryMax Функция

**файл**

WFApi.h

C

void WF_TxPowerGetFactoryMax (

INT8 * p_factoryMaxTxPower

) ;

**Возвращает**

Ни один .

**описание**

определена ( MRF24WG )

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_**factoryMaxTxPower** для MRF24WB , желаемый maxTxPower ( от -10 до 10 дБм) , шагом в 1 дБ для

MRF24WG , желаемый maxTxPower ( от 0 до 18 дБм) , шагом в 1 дБ

//*-*-*-*-*----------

### 11.5.1.4 WF_TxPowerGetMax Функция

**файл**

WFApi.h

C

```
void WF_TxPowerGetMax (
INT8 * p_maxTxPower
) ;
```

**Возвращает**

Ни один .

**описание**

После инициализации власть MRF24WG0M макс Tx определяется заводе значение . Эта функция может установить другой Максимальные уровни мощности Tx . Тем не менее, эта функция никогда не может установить максимальную Tx мощности, которая превышает значение заводской настройки ,

который может быть прочитан через WF_TxPowerGetFactoryMax (см. стр. 679 ) () .

**Замечания**

Никакого преобразования единиц , необходимых , вход MRF24WG0M не в дБм .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_**maxTxPower** Указатель на месте написать maxTxPower

//*-*-*-*-*----------

### 11.5.1.5 WF_TxPowerSetMax Функция

**файл**

WFApi.h

C

void WF_TxPowerSetMax (

INT8 maxTxPower

) ;

**Возвращает**

Ни один .

**описание**

Функции Tx управления питанием

После инициализации власть MRF24WG0M макс Tx определяется заводе значение . Эта функция может установить другой Максимальные уровни мощности Tx . Тем не менее, эта функция никогда не может установить максимальную Tx мощности, которая превышает значение заводской настройки , который может быть прочитан через **WF_TxPowerGetFactoryMax** (см. стр. 679 ) () .

**Замечания**

Никакого преобразования единиц , необходимых , вход MRF24WG0M не в дБм .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**maxTxPower** Допустимый диапазон (от 0 до 18 дБм)

### 11.6 Wi-Fi энергосбережения

**модуль**

Wi-Fi интерфейс API (см. стр. 628 )

**Функции**

**Имя Описание**

**GetAppPowerSaveMode** (см. стр. 686 ) Это функция GetAppPowerSaveMode .

**SetAppPowerSaveMode** (см. Стр. 687 ) Это функция SetAppPowerSaveMode .

**макрос**

**Имя Описание**

REG_DISABLE_LOW_POWER_MASK

(См. стр. 688 )

Это макро REG_DISABLE_LOW_POWER_MASK .

**Переменные**

**Имя Описание**

**g_AppPowerSaveModeEnabled**(См. Стр. 687 ) Это переменная g_AppPowerSaveModeEnabled .

**g_rxDtim** (см. стр. 687 ) Это переменные g_rxDtim .

**g_sleepNeeded** (см. стр.688 ) Это переменная g_sleepNeeded .

**gRFModuleVer1209orLater** (см. стр. 688 ) Это переменная gRFModuleVer1209orLater .

**описание**

MRF24WB0M поддерживает два энергосберегающих режима - сон и зимуют .

**Режим Описание**

Сна Этот режим используется , когда в режиме PS Опрос где MRF24WB0M просыпается себя вверх на периодическое интервалы отправлять запросы в сеть для получения сообщений в буфере точкой доступа . Посмотреть listenInterval в структуре tWFCAElements .

Когда в спящем режиме в MRF24WB0M передатчик схемы приемника выключены вместе с другими схема для минимизации энергопотребления .

Спящий режим периодически вступил в результате центральный процессор благоприятной режим PS Опрос .

Hibernate Этот режим эффективно превращает MRF24WB0M отправился максимального энергосбережения . MRF24WB0M состояние не сохраняется , и когда MRF24WB0M берется из состояния гибернации он выполняет перезагрузить .

Режим гибернации контролируется путем переключения булавку XCEN33 на модуле MRF24WB0M (по гибернации , низкий для выхода ) .

Этот режим следует использовать , когда приложение позволяет модуль MRF24WB0M быть от для длительные периоды времени .

Экономия энергии Функции 802.11 чипсеты имеют две хорошо известные режимы рабочая мощность . Активный режим питания определяется как радио всегда на либо передачи или приема , а это означает , что, когда он не передает то он пытается получить. Режим экономии энергии определяется как работает с радио выключено , когда нет ничего , чтобы передавать и только повернув радиоприемник , когда требуется.

Режим энергосбережения это режим, который требует взаимодействия с точкой доступа . Точка доступа получает уведомление через пакет от станции , что он вступает в режим экономии энергии . В результате точка доступа не требуется для буферизации любые пакеты , которые предназначены для станции , пока станция не объявляет, что он готов еще раз принимать пакеты . Продолжительность что

Станция может оставаться в этом режиме ограничено и обычно в 10 раз больше интервала радиомаяка от точки доступа.

Если хост ожидает пакеты из сети он должен работать в активном режиме . Если, однако, экономия энергии имеет решающее значение и пакеты не ожидается , то хозяин должен рассмотреть работает в режиме энергосбережения . Из-за характера точек доступа не все ведут себя одинаково, есть возможность того, чтоточка доступа может привести к аннулированию соединение станций , если он не слышал от станции за данный период времени. По этой причине режим энергосбережения следует использовать с осторожностью.

802.11 название для энергосберегающем режиме является PS- Poll ( Power- Сохранить Опрос ) .

//*-*-*-*-*----------

**11.6.1 Power Save открытые члены**

**Функции**

**Имя Описание**

**WF_GetPowerSaveState** (см. стр. 683 ) Возвращает текущее энергосбережения государство.

**WF_HibernateEnable** (см.страница 683) Переводит MRF24W в спящий режим .

**WF_PsPollDisable** (см.страница 684) Отключение режима PS- Poll .

**WF_PsPollEnable** (см. страница 684) Включает режим PS Опрос .

**модуль**

**Wi-Fi Power Save** (см. стр. 681 )

**описание**

Следующие функции и переменные доступны для применения стека.

//*-*-*-*-*-------------------------------------------
### 11.6.1.1 WF_GetPowerSaveState Функция

**файл**

WFApi.h

C

```
void WF_GetPowerSaveState (
UINT8 * p_powerSaveState
) ;
```

**Возвращает**

Ни один .

**описание**

Возвращает текущую власть MRF24W сохранить государство.

**Значение Определение**

---

WF_PS_HIBERNATE MRF24W в спящем состоянии

WF_PS_PS_POLL_DTIM_ENABLED MRF24W в режиме PS- Poll с DTIM включен

WF_PS_PS_POLL_DTIM_DISABLED MRF24W в режиме PS- Poll с DTIM отключена

WF_PS_POLL_OFF MRF24W не в любом энергосберегающий режим государства

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_powerSaveState Указатель где написано государственная власть

//*-*-*-*-*----------

### 11.6.1.2 WF_HibernateEnable Функция

**файл**

WFApi.h

C

аннулированию WF_HibernateEnable ( ) ;

**Возвращает**

Ни один .

**описание**

Включает режим гибернации на MRF24W , которая эффективно отключает устройство для максимального энергосбережения .

MRF24W состояние не сохраняется , когда он переходит в спящий режим . Чтобы снять MRF24W от режима гибернации вызова **WF_Init** ().

**Замечания**

Обратите внимание, что , поскольку MRF24W не спасает положение , будет разрыв между IP стека TCP / и MRF24B0M состояние . Если желательно приложением использовать спящий режим , дополнительные меры должны быть приняты , чтобы спасти приложение состояние . Тогда хозяин должен быть сброшен . Это обеспечит чистую связь между MRF24W и TCP IP стек / Будущие версии стека могут иметь возможность сохранять контекст стека , а также, обеспечение чистой проснуться для MRF24W без необходимости сброса хоста.

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

//*-*-*-*-*----------

### 11.6.1.3 WF_PsPollDisable Функция

**файл**

WFApi.h

C

аннулированию WF_PsPollDisable ( ) ;

**Возвращает**

Ни один .

**описание**

Функции управления питанием

Отключение режима PS Опрос . MRF24W будет оставаться активным , а не идти спать .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

//*-*-*-*-*----------

**11.6.1.4 WF_PsPollEnable Функция**

**файл**

WFApi.h

C

void WF_PsPollEnable (

BOOL rxDtim

) ;

**Возвращает**

Ни один .

**описание**

Включает режим PS Опрос . PS- Poll ( Power- Сохранить Опрос ) это режим позволяет длительное время автономной работы . MRF24W координирует с Точка доступа , чтобы заснуть и проснуться через определенные промежутки времени для проверки сообщений данных, которыеточка доступа буфера . ListenInterval в Connection Алгоритм определяет интервал ожидания . По умолчанию режим PS- Poll отключена .

Когда PS Опрос включена, хост Драйвер WF автоматически заставить MRF24W просыпаться каждый раз, когда хост посылает Tx данных или управляющее сообщение в MRF24W . Когда

операция Хост сообщение является полным водитель MRF24W будет автоматически повторно включить режим PS Опрос .

Когда приложение , скорее всего, испытывают большой объем трафика данных , то режим PS- Опрос должен быть отключен на двоих

причины:

1 . Нет энергосбережения не будут реализованы в присутствии интенсивного движения данных.

2 . Производительность будет неблагоприятно сказались какWi-Fi хоста Драйвер постоянно включает и выключает режим PS- Poll через Сообщения SPI .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

rxDtim ИСТИНА слушает с интервалом DTIM и ложных слушает на CASetListenInterval

//*-*-*-*-*----------

**11.6.2 Power Save Внутренние Пользователи**

**Функции**

**Имя Описание**

**SendPowerModeMsg** (см.страница 685) Отправить сообщение управления режима питания к MRF24W .

**SetPowerSaveState** (см.страница 686) Устанавливает желаемое энергосбережения состояния MRF24W .

**модуль**

**Wi-Fi Power Save** (см. стр. 681 )

**описание**

Следующие функции и переменные определяются как внутренние, к модулю.

//*-*-*-*-*----------

#### 11.6.2.1 SendPowerModeMsg Функция

**файл**

WFPowerSave.c

C

статической силы SendPowerModeMsg (

tWFPwrModeReq * p_powerMode

) ;

**Возвращает**

Ни один .

**описание**

МЕСТНЫЕ прототипы функций

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_powerMode Указатель на tWFPwrModeReq структуру отправить MRF24W .

//*-*-*-*-*----------

#### 11.6.2.2 SetPowerSaveState Функция

**файл**

WFPowerSave.c

C

void SetPowerSaveState (

UINT8 powerSaveState

) ;

**Возвращает**

Ни один .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**powerSaveState** Значение энергосбережения государство лучшего.

**11.6.3 Функции**

**Функции**

**Имя Описание**

**GetAppPowerSaveMode** (см. стр. 686 ) Это функция GetAppPowerSaveMode .

**SetAppPowerSaveMode** (см. Стр. 687 ) Это функция SetAppPowerSaveMode .

**модуль**

Wi-Fi Power Save (см. стр. 681 )

//*-*-*-*-*----------

**11.6.3.1 GetAppPowerSaveMode Функция**

**файл**

WFPowerSave.c

C

BOOL GetAppPowerSaveMode ( ) ;

**описание**

Это функция GetAppPowerSaveMode .

//*-*-*-*-*----------

**11.6.3.2 SetAppPowerSaveMode Функция**

**файл**

WFPowerSave.c

C

void SetAppPowerSaveMode (

состояние BOOL

) ;

**описание**

Это функция SetAppPowerSaveMode .

**11.6.4 Переменные**

**модуль**

Wi-Fi Power Save (см. стр. 681 )

**Переменные**

**Имя Описание**

**g_AppPowerSaveModeEnabled**(См. Стр. 687 ) Это переменная g_AppPowerSaveModeEnabled .

**g_rxDtim** (см. стр. 687 ) Это переменные g_rxDtim .

**g_sleepNeeded** (см. стр.688 ) Это переменная g_sleepNeeded .

**gRFModuleVer1209orLater** (см. стр. 688 )Это переменная gRFModuleVer1209orLater .

//*-*-*-*-*----------

**11.6.4.1 g_AppPowerSaveModeEnabled Переменная**

**файл**

WFPowerSave.c

C

BOOL g_AppPowerSaveModeEnabled = false;

**описание**

Это переменная g_AppPowerSaveModeEnabled .

//*-*-*-*-*----------

**11.6.4.2 g_rxDtim Переменная**

**файл**

WFPowerSave.c

C

BOOL g_rxDtim ;

**описание**

Это переменная g_rxDtim .

//*-*-*-*-*----------

**11.6.4.3 g_sleepNeeded Переменная**

**файл**

WFPowerSave.c

C

BOOL g_sleepNeeded = false;

**описание**

Это переменная g_sleepNeeded .

//*-*-*-*-*----------

**11.6.4.4 gRFModuleVer1209orLater Переменная**

**файл**

WFPowerSave.c

C

BOOL gRFModuleVer1209orLater ;

**описание**

**Это переменная gRFModuleVer1209orLater .**

**//*-*-*-*-*----------**

### 11.6.5 Макросы

**макрос**

**Имя Описание**

REG_DISABLE_LOW_POWER_MASK

(См. стр. 688 )

Это макро REG_DISABLE_LOW_POWER_MASK .

**модуль**

Wi-Fi Power Save (см. Стр. 681 )

**//*-*-*-*-*----------**

#### 11.6.5.1 REG_DISABLE_LOW_POWER_MASK Макро

**файл**

WFPowerSave.c

C

# определить REG_DISABLE_LOW_POWER_MASK ( ( UINT16 ) ( 0x00) )

**описание**

Это макро REG_DISABLE_LOW_POWER_MASK .

## 11.7 Wi-Fi Разное Microchip Стек TCP / IP Помощь

## 11.7 Wi-Fi Разное

**модуль**

Wi-Fi интерфейс API (см. стр. 628 )

**описание**

11.7.1 Wi-Fi Разное открытые члены

**Функции**

**Имя Описание**

**WF_GetDeviceInfo** (см.страница 690 )Получает информацию об устройстве WF

**WF_GetMacAddress** (см.страница 690 )Получает MRF24W MAC-адрес

**WF_GetMacStats** (см.страница 690 )Получает статистику MAC .

**WF_GetMultiCastFilter** (см. стр. 691 )Получает фильтр многоадресной адрес от одного из двух фильтров многоадресной .

**WF_GetRegionalDomain** (см. стр. 692 )Получает MRF24W Региональный домен

**WF_GetRtsThreshold** (см.страница 692)Получает порог РТС

**WF_SetMacAddress** (см.страница 693 )Использует различные MAC -адрес для MRF24W

**WF_SetMultiCastFilter** (см. стр. 693 )Устанавливает фильтр многоадресной адрес , используя один из двух групповых фильтров.

**WF_SetRegionalDomain** (см. стр. 694 )Включает или выключает MRF24W регионального домена .

**WF_SetRtsThreshold** (см.страница 695 )Устанавливает порога RTS .

**WF_EnableSWMultiCastFilter**(См. стр. 697 )Заставляет модуль FW использовать программный фильтр вместо аппаратного фильтра

**WF_MulticastSetConfig** (см. стр. 697 )Устанавливает фильтр многоадресной адрес , используя один из двух групповых фильтров.

**модуль**

Wi-Fi Разное ( см. стр. 689 )

**структуры**

**Имя Описание**

tWFDeviceInfoStruct (см.

страница 695 )

используется в WF_GetDeviceInfo (см. стр. 690 )

WFMacStatsStruct (см.

страница 696 )

Это запись WFMacStatsStruct .

**описание**

Следующие функции и переменные доступны для применения стека.

//*-*-*-*-*----------

**11.7.1.1 WF_GetDeviceInfo Функция**

**файл**

WFApi.h

C

void WF_GetDeviceInfo (

tWFDeviceInfo * p_deviceInfo

) ;

**Возвращает**

Ни один .

**описание**

функции Версия

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_deviceInfo Указатель где информационная устройство будет написано

//*-*-*-*-*----------

**11.7.1.2 WF_GetMacAddress Функция**

**файл**

WFApi.h

C

void WF_GetMacAddress (

UINT8 * p_mac

) ;

**Возвращает**

Ни один .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_mac Указатель где Mac будет написано ( должен указывать на буфер 6- байт)

**//*-*-*-*-*----------**

### 11.7.1.3 WF_GetMacStats Функция

**файл**

WFApi.h

C

void WF_GetMacStats (

tWFMacStats * p_macStats

) ;

**Возвращает**

Ни один .

**описание**

MAC Статистика

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

p_**macStats** Указатель где статистика MAC написаны

//*-*-*-*-*----------

**11.7.1.4 WF_GetMultiCastFilter Функция**

**файл**

WFApi.h

C

void WF_GetMultiCastFilter (

UINT8 multicastFilterId ,

UINT8 MulticastAddress [6]

) ;

**Возвращает**

Ни один .

**описание**

Получает текущее состояние указанного Multicast Filter.

Обычно назовет SendGetParamMsg , но это GetParam возвращает все 6 Адрес фильтры больше + 2 байта в общей сложности 48 байт плюс заголовок . Так , делая этот сообщ вручную, чтобы не требуют большого выделение стека для хранения всех данных.

Точный формат возвращено сообщение : [0] - всегда Упр ответ (2) [1] — всегда WF_GET_PARAM_SUBTYPE (16) [2] - результат ( 1 в случае успеха ) [3] - состояние Mac (не используется) [4] - длина блока данных ( длина ответных данных , начиная с индекса 6 ) [5] - не используется [ 6-11 ] - Сравнить адрес ( см. стр. 147 ) 0 адрес [12] - Сравнить Адрес (см. стр. 147 ) 0 группа [13] — Сравнить Адрес (см. стр. 147) 0 типа [ 14-19 ] - Сравнить Адрес (см. стр. 147 ) 1 адрес [20] - Сравнить Адрес (см.стр. 147) 1 группа [21] - Сравнить адрес ( см. стр. 147 ) 1 типа [ 22-27 ] - Сравнить Адрес (см. стр. 147 ) 2 адрес [ 28] - Сравнить адрес ( см. стр. 147 ) 2 группы [29] - Сравнить Адрес (см. стр. 147 ) 2 типа [ 30-35 ] -Сравните

адрес ( см. стр. 147 ) 3 адрес [36] - Сравнить Адрес (см. стр. 147 ) 3 группа [37] - сравните адрес(См. стр. 147) 3 типа [ 38-43 ] - Сравнить Адрес (см. стр. 147 ) 4 адрес [44] - Сравнить Адрес (см. стр.147 ) 4 группа [45] - Сравнить Адрес (см. стр. 147 ) 4 Тип [ 46-51 ] - Сравнить Адрес (см. стр. 147 ) 5 адрес[52] - Сравнить адрес ( см. стр. 147 ) 5 группы [53] - Сравнить Адрес (см. стр. 147 ) 5 Тип

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**multicastFilterId** WF_MULTICAST_FILTER_1 или WF_MULTICAST_FILTER_2

MulticastAddress адрес 6- байт

//*-*-*-*-*----------

**11.7.1.5 WF_GetRegionalDomain Функция**

**файл**

WFApi.h

C

```
void WF_GetRegionalDomain (
UINT8 * p_regionalDomain
) ;
```

**Возвращает**

Ни один .

**описание**

см. **tWFRegDomain** перечислены типы

Получает региональный домен на MRF24W . Допустимые значения:

• WF_DOMAIN_FCC

• WF_DOMAIN_IC

• WF_DOMAIN_ETSI

• WF_DOMAIN_SPAIN

• WF_DOMAIN_FRANCE

• WF_DOMAIN_JAPAN_A

• WF_DOMAIN_JAPAN_B

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**p_regionalDomain** Указатель где региональное значение домен будет написано

**//*-*-*-*-*----------**

### 11.7.1.6 WF_GetRtsThreshold Функция

**файл**

WFApi.h

C

```
void WF_GetRtsThreshold (
UINT16 * p_rtsThreshold
) ;
```

**Возвращает**

Ни один .

**описание**

Получает размер пакета порог RTS / CTS.

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**p_rtsThreshold** Указатель где написано порог РТС

//*-*-*-*-*----------

**11.7.1.7 WF_SetMacAddress Функция**

**файл**

WFApi.h

C

void WF_SetMacAddress (

UINT8 * p_mac

) ;

**Возвращает**

Ни один .

**описание**

MAC-адрес ( см. стр. 147 ) Функции Направляет MRF24W использовать входной MAC - адрес вместо со стандартными заводскими настройками MAC-адрес. Эта функция не перезаписать заводские настройки по умолчанию , который является флэш-память - это просто говорит MRF24W использовать другой MAC.

**Замечания**

Ни один .

**Предпосылки**

MACInit должен быть вызван в первую очередь. Не может быть вызвана , когда MRF24W находится в подключенном состоянии .

**Параметры**

**Параметры Описание**

p_mac Указатель на 6- байт MAC , которые будут отправлены в MRF24W

//*-*-*-*-*----------

### 11.7.1.8 WF_SetMultiCastFilter Функция

**файл**

WFApi.h

C

void WF_SetMultiCastFilter (

UINT8 multicastFilterId ,

UINT8 MulticastAddress [6]

) ;

**Возвращает**

Ни один .

**описание**

**Multicast** Функции

Эта функция позволяет приложению настроить до двух групповой адрес (см. стр. 147 ) Фильтры на MRF24WB0M .

Если два активных групповых фильтры настроены они OR'd вместе -MRF24WB0M получит и перейти к центральным процессором принятых пакетов с любой групповой адрес . Допустимые значения для многоадресной фильтра являются:

• WF_MULTICAST_FILTER_1

• WF_MULTICAST_FILTER_2

По умолчанию оба Multicast Фильтры неактивны .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**multicastFilterId** WF_MULTICAST_FILTER_1 или WF_MULTICAST_FILTER_2

**MulticastAddress** адрес 6- байт (все 0xFF инактивирует фильтр )

//*-*-*-*-*----------

### 11.7.1.9 WF_SetRegionalDomain Функция

**файл**

WFApi.h

C

void WF_SetRegionalDomain (

UINT8 regionalDomain

) ;

**Возвращает**

Ни один .

**описание**

см. tWFRegDomain перечислены типы

Устанавливает региональный домен на MRF24W . Обратите внимание, что эта функция не заменяет установленный на заводе регионального домена в FLASH . По умолчанию MRF24W будет использовать установленный на заводе региональный домен . Это является недопустимым , чтобы вызвать эту функцию в то время как в связный состояние .

Допустимые значения для регионального домена являются:

• WF_DOMAIN_FCC

• WF_DOMAIN_IC

• WF_DOMAIN_ETSI

• WF_DOMAIN_SPAIN

• WF_DOMAIN_FRANCE

• WF_DOMAIN_JAPAN_A

• WF_DOMAIN_JAPAN_B

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь. Эта функция не должна вызываться в то время как в подключенном состоянии .

**Параметры**

**Параметры Описание**

**regionalDomain** значение, чтобы установить региональный домен , чтобы

//*-*-*-*-*----------

**11.7.1.10 WF_SetRtsThreshold Функция**

**файл**

WFApi.h

C

```
void WF_SetRtsThreshold (
UINT16 rtsThreshold
) ;
```

**Возвращает**

Ни один .

**описание**

RTS Threshold Функции Устанавливает RTS / CTS размера пакета порог , когда RTS / CTS кадр будет отправлен . По умолчанию используется 2347 байт - максимальный для 802,11 . Рекомендуется, чтобы пользователь оставьте по умолчанию в 2347 , пока они не понимают производительности и энергопотребления Последствия установив его меньше . Допустимые значения от 0 до WF_RTS_THRESHOLD_MAX ( 2347 ) .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**Параметры**

**Параметры Описание**

**rtsThreshold** Значение порогового значения размера пакета

//*-*-*-*-*----------

**11.7.1.11 tWFDeviceInfoStruct Структура**

**файл**

WFApi.h

C

структура tWFDeviceInfoStruct {

UINT8 DeviceType ;

UINT8 romVersion ;

UINT8 patchVersion ;

} ;

Пользователи

Пользователи Описание

UINT8 DeviceType ; MRF24WB0M_DEVICE_TYPE

UINT8 romVersion ; ROM номер версии

UINT8 patchVersion ; Патч номер версии

**описание**

используется в WF_GetDeviceInfo (см. стр. 690 )

//*-*-*-*-*----------

**11.7.1.12 WFMacStatsStruct Структура**

**файл**

WFApi.h

C

структура WFMacStatsStruct {

UINT32 MibWEPExcludeCtr ;

UINT32 MibTxBytesCtr ;

UINT32 MibTxMulticastCtr ;

UINT32 MibTxFailedCtr ;

UINT32 MibTxRtryCtr ;

UINT32 MibTxMultRtryCtr ;

UINT32 MibTxSuccessCtr ;

UINT32 MibRxDupCtr ;

UINT32 MibRxCtsSuccCtr ;

UINT32 MibRxCtsFailCtr ;

UINT32 MibRxAckFailCtr ;

UINT32 MibRxBytesCtr ;

UINT32 MibRxFragCtr ;

UINT32 MibRxMultCtr ;

UINT32 MibRxFCSErrCtr ;

UINT32 MibRxWEPUndecryptCtr ;

UINT32 MibRxFragAgedCtr ;

UINT32 MibRxMICFailureCtr ;

} ;

**Пользователи**

**Пользователи Описание**

UINT32 MibWEPExcludeCtr ; Количество кадров, полученных с охраняемых кадров подполе раме Поле управления устанавливается в ноль , а значение dot11ExcludeUnencrypted вызывает то, что рамка для выбрасывания .

UINT32 MibTxBytesCtr ; Общее количество Tx байтов , которые были переданы

UINT32 MibTxMulticastCtr ; Количество кадров, переданных успешно , что было многоадресной немного установлен в MAC-адрес назначения .

UINT32 MibTxFailedCtr ; Количество Tx кадров , которые не из-за количества присылает превышения количество повторов .

UINT32 MibTxRtryCtr ; Количество разпереданный кадр необходимо повторно

UINT32 MibTxMultRtryCtr ; Количество раз кадр был успешно передан после более чем один ретрансляция .

UINT32 MibTxSuccessCtr ; Количество Tx кадров, переданных успешно .

UINT32 MibRxDupCtr ; Количество кадров, принятых где поле Последовательность управления указывает дублировать .

UINT32 MibRxCtsSuccCtr ; Количество CTS кадров, полученных в ответ на раме PTC.

UINT32 MibRxCtsFailCtr ; Количество разкаркасных PTC не был получен в ответ на CTS кадра.

UINT32 MibRxAckFailCtr ; Количество разAck не был получен в ответ на Tx кадра.

UINT32 MibRxBytesCtr ; Общее количество рецептурных полученных байтов .

UINT32 MibRxFragCtr ; Количество успешных принятых кадров (управление или данные )

UINT32 MibRxMultCtr ; Количество кадров, принятых с многоадресной битом в пункте назначения MAC адрес .

UINT32 MibRxFCSErrCtr ; Количество кадров, полученных с недопустимым кадров контрольной суммы ( FCS ) .

UINT32 MibRxWEPUndecryptCtr ; Количество полученных кадров , где Защищенный Рамка подполе раме Поле управления устанавливается в единицу , а значение WEPOn для ключа сопоставляется с MAC-адрес передатчика указывает кадр не должны были зашифрованы .

UINT32 MibRxFragAgedCtr ; Количество раз, фрагменты " возрасте из ' или не были получены в допустимое время .

UINT32 MibRxMICFailureCtr ; Количество MIC неудач , которые произошли .

**описание**

Это запись WFMacStatsStruct .

**//*-*-*-*-*----------**

### 11.7.1.13 WF_EnableSWMultiCastFilter Функция

**файл**

WFApi.h

C

аннулированию WF_EnableSWMultiCastFilter ( ) ;

**Возвращает**

Ни один .

**описание**

Эта функция позволяет приложению настроить до макс 16 групповой адрес (см. стр. 147 ) Фильтры на MRF24WG .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

//*-*-*-*-*----------

**11.7.1.14 WF_MulticastSetConfig Функция**

**файл**

WFApi.h

C

void WF_MulticastSetConfig (

tWFMultiCastConfig * p_config

) ;

**Возвращает**

Ни один .

**описание**

Эта функция позволяет приложению настроить до двух групповой адрес (см. стр. 147 ) Фильтры на MRF24W . Если два Активные групповые фильтры настроены они OR'd вместе - MRF24W получит и перейти к центральным процессором , полученной пакеты из любой групповой адрес . Допустимые значения в p_config являются:

FilterId - WF_MULTICAST_FILTER_1 или WF_MULTICAST_FILTER_2

Действие - WF_MULTICAST_DISABLE_ALL (по умолчанию)фильтр многоадресной отбрасывает все полученные многоадресных сообщений - они будут не направляется принимающей ПОС. Остальные поля в этой структуре игнорируются.

WF_MULTICAST_ENABLE_ALLфильтр многоадресной вперед все полученные многоадресных сообщений на хост ПОС.

Остальные поля в этой структуре игнорируются.

WF_MULTICAST_USE_FILTERSMAC фильтр будет использоваться и остальные поля в этой структуре настроить, какие  Multicast сообщений будут пересылаться на принимающей ПОС.

macBytes - Массив , содержащий MAC- адрес для фильтрации на (используя адрес назначения каждого входящего 802.11 кадра ) .

Конкретные байт с MAC-адресом может быть обозначен как " не заботится ' байт. Смотреть macBitMask . Это поле в только если

действие = WF_MULTICAST_USE_FILTERS .

macBitMask -байт , где биты 5:00 соответствуют macBytes [ 5:00 ]. Если бит равен нулю , то соответствующий байт должен MAC быть точным соответствием для кадра была перенаправлена на принимающей ПОС. Если бит является одним то соответствующий MAC-байт" не заботиться »и не используется в процессе фильтрации многоадресной рассылки. Это поле в используется, только если действие = WF_MULTICAST_USE_FILTERS .

По умолчанию оба Multicast Фильтры неактивны .

**Замечания**

Ни один .

**Предпосылки**

**MACInit** должен быть вызван в первую очередь.

**пример**

- Фильтр на Multicast Адрес 1:00:05 e: xx: xx: xx , где xx не волнует байт .

p_config - > FilterId = WF_MULTICAST_FILTER_1

[0] [1] [ 2] [3 ] [4] [5]

p_config - > macBytes [] = 01 , 00 , 5e , и далее, и далее, и далее ( 0xff являются не волнует байт )

p_config - > macBitMask = 0x38 -> биты 5:03 = 1 ( не важно на байт 3,4,5 )

-> Биты 2:00 = 0 ( Точное совпадение требуется на байт 0,1,2 )

**//*-*-*-*-*----------**

**11.8 WF_ProcessEvent**

**модуль**

Wi-Fi интерфейс API (см. стр. 628 )

**описание**

Есть несколько событий , которые могут произойти на MRF24WB0M , что центральный процессор может хотите знать о . Все MRF24WB0M события пройти через функции WF_ProcessEvent ( ) , описанной в следующем разделе.

**Обработка событий**

Функция WF_ProcessEvent ( ) , как основное приложение уведомляется о событиях . Эта функция будет вызываться в Wi-Fi пройдет драйвер , когда происходит событие . Эта функция не следует вызывать непосредственно ведущим приложением . Эта функция , расположенный в WF_Customize.c , должны быть изменены пользователем по мере необходимости . Так как эта функция вызывается из водителя WiFi есть некоторые ограничения , а именно - никто не может вызывать любые Wi-Fi функции драйвера , когда внутри WFProcessEvent . Рекомендуется, чтобы что клиент просто установить флаг для определенного события и обработать его в основном цикле . Рамки для этой функции приведен ниже.

Прототип для этой функции находится :

UINT16 WF_ProcessEvent ( UINT8 событие, UINT16 EventInfo ) ;

Есть два входа функции :

Мероприятие Событие , которое произошло.

EventInfo Дополнительная информация о мероприятии . Это применимо не для всех событий .

В приведенной ниже таблице представлены возможные значения , что параметры событий и EventInfo может иметь. Обратите внимание, что уведомление о событии из

некоторые события могут быть дополнительно отключить через :

1 . Битовая маска eventNotificationAction в структуре tWFCAElements (см. Wi-Fi соединения Алгоритм (см. стр. 648 ) ) , или

2 . Функция WF_CASetEventNotificationAction ( ) (см. стр. 661 ) .

мероприятие EventInfo

WF_EVENT_CONNECTION_SUCCESSFUL Попытка подключения была успешной.

EventInfo :

• Всегда WF_NO_ADDITIONAL_INFO

(Необязательно событие)

WF_EVENT_CONNECTION_FAILED Попытка подключения не удалась EventInfo :

• WF_JOIN_FAILURE

• WF_AUTHENTICATION_FAILURE

• WF_ASSOCIATION_FAILURE

• WF_WEP_HANDSHAKE_FAILURE

• WF_PSK_CALCULATION_FAILURE

• WF_PSK_HANDSHAKE_FAILURE

• WF_ADHOC_JOIN_FAILURE

• WF_SECURITY_MISMATCH_FAILURE

• WF_NO_SUITABLE_AP_FOUND_FAILURE

• WF_RETRY_FOREVER_NOT_SUPPORTED_FAILURE

(Необязательно событие)

WF_EVENT_CONNECTION_TEMPORARILY_LOSTустановленное соединение было временно потерял - соединение

Алгоритм пытается восстановить соединение. Поле EventInfo указывает , почему

связь была потеряна.

EventInfo :

• WF_BEACON_TIMEOUT

• WF_DEAUTH_RECEIVED

• WF_DISASSOCIATE_RECEIVED

(Необязательно событие)

WF_EVENT_CONNECTION_PERMANENTLY_LOSTустановленное соединение было окончательно потерян - соединение

Алгоритм либо кончились попыток или был настроен не повторить .

EventInfo поле показывает , почему связь была потеряна.

EventInfo :

• WF_BEACON_TIMEOUT

• WF_DEAUTH_RECEIVED

• WF_DISASSOCIATE_RECEIVED

Это событие также может образоваться, когда WF_CMDisconnect (см.

стр. 672 ) () не будет вызвана , в этом случае поле EventInfo не имеет

смысл. (Необязательно событие)

WF_EVENT_CONNECTION_REESTABLISHEDсоединение, которое было временно потеряли была restablished

Всегда WF_NO_ADDITIONAL_INFO (Необязательно событие)

WF_EVENT_SCAN_RESULTS_READY запроспроверки по телефону WF_Scan (см. стр. 676 ) ( ) завершила и результаты могут быть считаны с MRF24WB0M .

EventInfo : Количество результатов сканирования

12.2 WF_ProcessEvent ( ) Рамочная

Ниже приводится структура для WF_ProcessEvent ( ) . Каждый случай заявление должно быть изменены по мере необходимости обрабатывать события приложение заинтересовать

```
 void WF_ProcessEvent(UINT8 event, UINT16 eventInfo)

{
switch (event)
{
case WF_EVENT_CONNECTION_SUCCESSFUL:
/ * Код приложения здесь * /
break;
case WF_EVENT_CONNECTION_FAILED:
/ * Код приложения здесь * /
break;
case WF_EVENT_CONNECTION_TEMPORARILY_LOST:
/* Application code here */
break;
case WF_EVENT_CONNECTION_PERMANENTLY_LOST:
/* Application code here */
break;
case WF_EVENT_FLASH_UPDATE_SUCCESSFUL:
/* Application code here */
break;
case WF_EVENT_FLASH_UPDATE_FAILED:
/* Application code here */
break;
case WF_EVENT_KEY_CALCULATION_COMPLETE:
/* Application code here */
break;
case WF_EVENT_SCAN_RESULTS_READY:
/* Application code here */
break;
case WF_EVENT_IE_RESULTS_READY:
/* Application code here */
break;
default:
WF_ASSERT(FALSE);
break;
}
}
```

**11.9 Точка доступа Совместимость**

**модуль**

Wi-Fi интерфейс API (см. стр. 628 )

**описание**

**введение**

MRF24WB0M прошел через сертификационных испытаний Wi- Fi.org . Не все маршрутизаторы пройти через сертификации Wi- Fi.org , и некоторые из них предварительно настроен в режимах Гринфилд . Кроме того, пользователи могут установить настройки, которые серьезно ограничивают производительность или предотвратить коммуникаций. Это приложение записка предназначена для обеспечения на постоянной снимок совместимости среди немногих из самых популярные и ведущие мировые точки доступа , а также большая группа во всем мире единицы . Результаты испытаний покажут удобство из модулей Microchip Wi-Fi , работающих с последней версии стека Microchip TCPIP .

**Wi-Fi Alliance Тестирование**

Перевезти логотип Wi-Fi Alliance , продукты Wi-Fi должны успешно пройти многочисленные испытания , в том числе испытаний на совместимость . Wi-Fi тестирование на совместимость выполняется с 4 пунктов представитель доступа, с подмножеством тестов, выполненных на каждой из

точек доступа. Устройства проходят испытания в этих точек доступа для таких характеристик, как связи , безопасности, пропускная способность, и широта других спецификаций . Microchip Wi-Fi модули успешно прошли тестирование Wi-Fi Alliance .

Доклад под названием WFA7150 и доступен в http://certifications.wi-fi.org/pdf_certificate.php?cid=WFA7150

**Дополнительная Wi-Fi Тестирование совместимости**

Wi-Fi технология резко расширяется охват и приложений Интернета для встраиваемых устройств. Во многих случаях Wi-Fi является новым для рынков и приложений он идущими . В результате , Microchip чувствует важно поднять планку тестирование на совместимость , и воспитание разработчика.

Microchip , таким образом, приняла стенд Wi- Fi.org для получения дополнительной общего тестирования точки доступа . Цель этих испытаний заключается в обеспечить базовое подключение в нескольких незащищенных и защищенных сценариев с глобальным представлением верхней доступа продажи пунктов.

Pass Критерии

Следующие тесты являются частью текущего набора тестирования и должны пройти в качестве точки доступа для считаться совместимыми .

• После в условиях отсутствия безопасности , WEP40 и WEP104 , WPA-PSK (TKIP) , WPA2-PSK (AES)

• AP ассоциации , Iperf UDP загрузки / выгрузки , Iperf загрузки TCP / скачать, DHCP , ICMP пинг

Во многих случаях есть и другие режимы, которые могут быть запущены с точками доступа , а пользователь должен предпринять меры предосторожности, если режим нет в списке, то совместимость не обязательно гарантировано. Эти режимы , как правило, Гринфилд использование, режимы , являющиеся устарела по Wi- Fi.org или случаев ограничения использования точки доступа для большего целей частных сетей , а не для правда совместимость Wi-Fi .

Примеры специальных режимов не обязательно часть результатов :

• WPA-PSK (AES) безопасности : безопасность WPA-PSK определяется как используя TKIP . Это смешанный режим . Этот режим работает , если AP всего автоматически определяет и не смешивается .

• WPA2-PSK (TKIP) безопасности : безопасность WPA2-PSK определяется как использование 802.11i с AES. Это смешанный режим . Этот режим работает , если AP всего автоматически определяет и не смешивается .

• 802.11g только , только 802.11n , 802.11g / п только : это частные режимы сети ( вырезания обязательное поддержку 802.11b ) .

Эти режимы могут работать, если базовые ставки ограничены 1 и 2 Мбит в 802,11 .

Список совместимых точек доступа :

• 2Wire 1701HG

• 2Wire 2701HG -B

• 3COM 3CRWER100 -75

• 3COM WL- 524

• Actiontec GT704 -WG

• AirLink AR690W

• Белкин N1

• Белкин F5D7231 -4

• Буффало WHR- G125

• Буффало WHR- HP- G54

• Corega CG- WLAPGMN

• Corega CG- WLBARGO

• D-Link DIR -655

• D-Link WBR -1310

• Dynex DX - WGRTR

• Уровень 1 WBR- 3408

• Linksys WRT150Nv1.1

• Linksys WRT310N

• Microsoft MN- 700

• Netgear WGR614v9

• Netgear WGT624v3

• Netgear WNDR3300

• Netgear WPN824v2

• Proxim AP- 700

• SMC Networks SMCWBR14T -G

• TP-Link TL - WR541G

• Westell B90 - 327W15 -06

* Примечание испытаний, проведенных :

• Основные ассоциации с точкой доступа (без обеспечения)

• Ассоциация с WEP безопасности

• Ассоциация с безопасностью WPA/WPA2-PSK

• проверка тест пинг .

### 11.10 WiFi Советы и хитрости

**модуль**

Wi-Fi интерфейс API (см. стр. 628 )

**описание**

Советы по Установка маршрутизаторов для 802.11b пользования

Цель этого раздела состоит в описании настройки для наиболее типичных AP настраиваемых параметров для совместимости с устройствами Microchip MRF24WB0Mx :

1. Настройки DHCP - для DHCP на стороне LAN ( где Ap DHCP сервер ) , установить маршрутизатор к Enable DHCP сервер . Установить клиент Аренда время, чтобы быть больше , чем типичный нерабочее время станции для того, чтобы IP- адрес, предоставленный не меняется каждый Время станция находится под напряжением . Если опция всегда для трансляции присутствует для установки DHCP (трансляций весь каталог DHCP ответы на всех клиентов ) , она должна быть отключена .

2. Настройки Скорость передачи данных - Убедитесь , что в стоимость услуг включают 802.11b . 802.11g или 802.11n только цену ( зеленый поле) должно быть избежать, но смешанные настройки , как правило, приемлемо. Если параметр Basic Rate определен, он должен быть установлен в 1 и 2 Мбит только .

3. широковещания SSID - . Обычно Должен быть включен так, что AP посылает фреймы , содержащие SSID. Если отключена , гарантировать, что Microchip Stack установлен для активного сканирования .

4. Выбор канала - Для отладочных целей , это характерно использовать фиксированный канал вместо автоматического выбора канала . Если фиксированный канал выбран для MRF24 станции , выберите соответствующий канал для точки доступа .

5. Multicast через удаленный - При использовании функций мультивещания ( ZeroConfig например) убедиться, что маршрутизатор настроен на включить переадресацию многоадресных пакетов.

6. Beacon Interval - . Установите значение для интервала времени между AP маяков , типичным является 100msec . Для меньшей мощности , это может быть установлено на меньшее значение , например 30 мс , если интервал DTIM соответственно увеличивается .

7. RTS Threshold - . Установите значение размера кадра , выше которого RTS / CTS будет использоваться , типичная 2347.

8. Порог фрагментации - Установите значение для размера кадра , выше которого пакеты будут фрагментированы , типичным является 2346 .

9. DTIM Интервал - . Установите значение Доставка трафика сообщение индикации интервала , типичным является 3 если Beacon Interval устанавливается для 100 мс . Для более низкой мощности с MRF24WB0M , если Beacon Interval установлен на 30 мс , то DTIM должен быть установлен в 100 , чтобы позволить 300 мс DTIM Интервал .

10. WLAN Partition (или AP Isolation) - . Предотвращает клиентов AP от общения друг с другом , как правило, отключается.

11. WMM Enable - Позволяет беспроводной мультимедийный трафик , не отключить , если необходимо для других служб AP .

12. Краткое Защитный интервал (GI) - . Снижает защитный интервал между кадрами , не отключить , если необходимо для других служб AP .

13. Wi-Fi Protected Setup (WPS ) - Позволяет открытие WPS устройства , не отключить , если необходимо для других служб AP .

14. кадров серийной съемки - обеспечивает более высокую пропускную способность беспроводной сети пакета, не отключить , если необходимо для других служб AP . Это может быть называется турбо, или другие маркетинговые термины .

15. Режим защиты CTS - . Повышает надежность 802.11g трафика , не отключить , если необходимо для других служб AP .

16. Key Entry - . Безопасности могут быть введены либо с числовым ключом или текстовая строка . Убедитесь, что вы введите то, что АП ожидает . Если только начинаете , то лучше иметь другую станцию , как ноутбук для проверки того, что AP ожидает .

**11.11 Горячие темы**

**модуль**

Wi-Fi интерфейс API (см. стр. 628 )

**описание**

Принимающая контролируемых Разъяснения диспетчера подключений

Следующие разъяснения должны отметить для использования устройства с Microchip TCPIP стека версий , если не указано иное.

**1 . Null Строка ESSID**

Это можно назвать WF_CMConnect (см. стр. 671 ) ( CPID ) с CPID нуля. Если это произойдет , диспетчер соединений можете использовать ошибочные значения для SSID , сетевой режим , настройки безопасности и т.д., которые будут вызывать модуль для подключения (См. стр. 171) к неправильному AP или нет подключения (см. стр. 171) на всех. Единственные допустимые значения , которые могут быть использованы для ссылки соединительного профиля являются 1 и 2 ( при условии, что WF_CPCreate (см. стр. 633 ) ( & CPID ) удалось создать эти профильные ссылки до проведения покушении связи ) .

**Решение проблемы :**

При создании профиля подключения , проверить, что число профиль вернулся всегда либо 1, либо 2 . Если возвращенное значение равно 0, удалить профиль и создать его заново . При подключении с WF_CMConnect (см. стр. 671 ) ( CPID ) , убедитесь, что только действительный номер профиля ранее вернулся из WF_CPCreate (см. стр. 633 ) ( & CPID ) используется .

2 . Сканирования Управление конфликт сообщение

Сообщения управления всегда должны вернуться успешной, либо вызывает предположением в принимающей водителя. Неудачная Сообщение управления может произойти , когда соединение повтора включена ( MY_DEFAULT_LIST_RETRY_COUNT > 0) в результате

чего устройство для сканирования из-за разрыва связи , а затемотключить или подключить (см. стр. 171) , или сканировать команда отправляется .

**Решение проблемы :**

Если вы управляете подключения (см. стр. 171 ) / подключите от хозяина активно , затем отключить все прошивки повторную попытку , используя "нет повтора сканирования "и" никаких действий де - авторизации " .

**а**. Чтобы отключить сканирование Повторить

WF_CASetListRetryCount (см. стр. 662 ) ( MY_DEFAULT_LIST_RETRY_COUNT ) ; должен быть 0

**б**. Чтобы отключить действие Де - авторизации

WF_CASetDeauthAction (см. стр. 660 ) ( WF_DO_NOT_ATTEMPT_TO_RECONNECT );

**в**. Чтобы отключить действие Де - аутентификации

WF_CASetBeaconTimeoutAction (см. стр. 658 ) (WF НЕ попытается восстановить связь ) ;

**г.** Используйте "Connect" только на " постоянной потере " или " сбоя соединения " .

**д**. Чтобы сделать поиск , в первую очередь проверьте состояние прошивки сначала с помощью WF_CMGetConnectionState (см. стр. 673 ) ( )

**е.** Если возвращение состояние WF_CSTATE_NOT_CONNECTED (или WF_CSTATE_CONNECTION_PERMANENTLY_LOST ), то это означает, прошивки находится в режиме ожидания, так хозяин может выдать хозяина сканирование безопасно

II . Если возвращение государство WF_CSTATE_CONNECTED_INFRASTRUCTURE , то это означает, прошивка в связных . В В этом случае команда сканирования может быть выдан , но сторожевой таймер должен быть использован для времени для конфликта. Кроме того, убедитесь, что таймер управления установлен , по крайней мере 0.4 seconds на канал сканируется , чтобы предотвратить очереди буфера запросов Tx от тайм-аута .

III . Если возвращение государство WF_CSTATE_CONNECTION_IN_PROGRESS (или WF_CSTATE_RECONNECTION_IN_PROGRESS ) ,

то это означает, прошивка в середине процесса соединения исканирование не должна быть инициирована .

**е .** Если функция "Отключить " желательно , сторожевой таймер должен быть использован для решения случай, когда возникает конфликт с над воздушным дистанцироваться или де авторизовать .

**г** . Для сторожевой времени , рекомендуется синхронизация 2x таймаут управления пакетов (то есть, использование 4seconds не если менеджмент Тайм-аут был увеличен ) .

**б .** Если вы используете только повторную попытку прошивки , а не делать какие-либо управление соединениями ( сканирование, подключить (см. стр. 171 ) ,

холостом ходу и т.д. ), то вы можете использовать MY_DEFAULT_LIST_RETRY_COUNT > 0 или повторить навсегда ( MY_DEFAULT_LIST_RETRY_COUNT = 255) . Если вы потеряете соединение , вы можете восстановить с помощью страницы " подключения (см.171) " API . Не используйте "Disconnect" .

. Если функция "Отключить " желательно , сторожевой таймер должен быть использован для решения случай, когда возникает конфликт с над воздушным дистанцироваться или де авторизовать .