Руководство программиста по библиотеке TCP/IP

Целевые микроконтроллеры

Данное руководство предназначено для работы со следующими микроконтроллерами (МК):
C8051F120, C8051F121, C8051F122, C8051F123, C8051F124, C8051F125, C8051F126, C8051F127, C8051F130, C8051F131, C8051F132, C8051F133

1. Введение

Стек (протоколов) TCP/IP Silicon Laboratories разработан для обеспечения МК C8051F12x и C8051F13x функциями сетевого взаимодействия. Он характеризуется большими возможностями конфигурирования и требует незначительных ресурсов памяти. Со стеком TCP/IP предлагаются мастер конфигурирования, который может генерировать каркас кода, требуемый для разработки сетевых приложений, и множество примеров, позволяющие сразу начать разработку и минимизировать время выхода на рынок готовых изделий.

Стек TCP/IP обеспечивает следующие функции:

- HTTP Web-сервер с поддержкой CGI-скриптов, e-mail SMTP-клиент, FTP-сервер, TFTP-клиент и виртуальная файловая система.
- Одновременная поддержка до 127 TCP- или UDP-сокетов. Непосредственный доступ к сокетам позволяет разрабатывать специализированные переносимые приложения.
- Поддержка протоколов PPP и SLIP с возможностью настройки параметров модема. Взаимодействие с модемом Si2457 через последовательный порт.

Стек TCP/IP свободно доступен для использования с МК Silicon Laboratories и может быть загружен с web-сайта Silicon Laboratories. Он также входит в состав набора средств разработки встраиваемых модемов (Modem-DK), который содержит:

- Демонстрационную плату C8051F12x, отладочный адаптер и блок питания.
- Плату модема Si2457FT18-EVB с платой адаптера модема AB3. **Примечание:** Для взаимодействия с модемом требуются непосредственное подключение к телефонной линии или симулятор телефонной линии.
- Оценочную версию пакета Keil C51, которая ограничена возможностью генерации объектного кода (из программного кода приложения) размером не более 4 Кбайт. Код библиотеки TCP/IP не выходит за пределы 4-Кбайтной области.
- Мастер конфигурирования TCP/IP, позволяющий генерировать специализированные библиотеки и примеры программ, которые показывают, как настроить HTTP Web-сервер, послать e-mail, отправить и принять TCP- и UDP-пакеты.

2. Обзор функций API

Стек TCP/IP предлагает набор функций, которые реализуют интерфейс прикладного программирования (API) в МК C8051F12x и C8051F13x. Эти функции предоставляют микроконтроллеру интерфейс доступа к коммутационной сети посредством модема Si2457, подключаемого к последовательному порту. Все низкоуровневые аппаратные и протокольные функции реализуются API и не требуют управления со стороны программного кода приложения. Этот API предлагается в виде библиотечного файла, откомпилированного с помощью пакета Keil C51. (Программное обеспечение устройства должно быть разработано с помощью пакета Keil C51.) Некоторые наиболее часто используемые функции API перечислены ниже:

`mn_init()`	Инициализирует все сокеты и переменные стека.
`mn_modem_connect()`	Устанавливает соединение между МК и модемом.
`mn_ppp_add_pap_user()`	Добавляет имя пользователя и пароль для использования в процессе аутентификации.
`mn_ppp_open()`	Устанавливает PPP-соединение.
`mn_server()`	Запускает HTTP- или FTP-сервер.
`mn_smtp_send_mail()`	Отправляет e-mail на почтовый SMTP-сервер.

3. Начало работы

Начать новый проект, использующий стек TCP/IP, можно одним из пяти способов:

- Модификация примера HTTP Web-сервера.
- Модификация примера почтового клиента SMTP.
- Модификация примера TCP-сокета.
- Модификация примера UDP-сокета.
- Использование мастера конфигурирования TCP/IP для генерации специализированной библиотеки и каркаса кода.

3.1. Структура каталога проекта.

Типичный каталог проекта TCP/IP состоит из следующих файлов и подкаталогов:

Группа 1:

`mn_userconst.h`	Заголовочный файл, содержащий пользовательские настройки.
`TCPIP_Project.wsp`	Файл проекта, который может быть открыт из интегрированной среды разработки Silicon Labs IDE.
`main.c`	Содержит основную программу и функцию обратного вызова.
`mn_stack000.lib`	Библиотека стека TCP/IP. Стоит обратить внимание на трехзначный номер библиотеки.

Группа 2:

`mn_defs.h`	Содержит определения типов, используемые стеком TCP/IP.
`mn_errs.h`	Содержит определения кодов ошибок.
`mn_funcs.h`	Содержит информацию о прототипах функций.
`mn_stackconst.h`	Содержит константы, требуемые стеком TCP/IP.
`mn_vars.c`	Содержит глобальные переменные, используемые стеком TCP/IP.

Группа 3:

`VFILE_DIR`	Необязательный подкаталог, содержащий HTML-файлы, рисунки и другое содержимое.
`VFILE_DIR\html2c.exe`	Необязательная утилита html2c.exe, которая преобразует содержимое в файловые массивы.
`VFILE_DIR\update.bat`	Необязательный пакетный файл, предназначенный для автоматизации процесса преобразования в файловые массивы.
`VFILE_DIR\mn_defs.h`	Необязательный заголовочный файл, необходимый только при использовании файловых массивов.
`VFILE_DIR\index.html`	Необязательная главная HTML-страница для web-сервера.
`VFILE_DIR\index.h`	Необязательная главная HTML-страница, преобразованная в файловый массив с помощью утилиты *html2c.exe.*
`VFILE_DIR\index.c`	Необязательная главная HTML-страница, преобразованная в файловый массив с помощью утилиты *html2c.exe.*

3.2. Выходные данные мастера конфигурирования TCP/IP.

Мастер конфигурирования TCP/IP генерирует специализированную библиотеку с уникальным трехзначным номером, которая описывает выбранную конфигурацию протокола. Мастер также генерирует вспомогательные структуру каталогов и каркас кода, необходимые для того, чтобы начать новый проект TCP/IP. Следует иметь ввиду, что генерируемый каркас кода будет зависеть от номера библиотеки, и библиотеки с различными трехзначными номерами нельзя брать из одного проекта и вставлять в другой без регенерации вспомогательного кода. Чтобы начать использовать код, генерируемый мастером, следует открыть файл *TCPIP_Project.wsp* с помощью команды *Project→Open Project…* из интегрированной среды разработки Silicon Laboratories IDE.

3.3. Использование примеров TCP/IP

Примеры кода, предлагаемые в данном руководстве, являются хорошим подспорьем для начала нового проекта. Если протоколы, используемые в примере, соответствуют требованиям конечного приложения (например, HTTP Web-сервер), то этот пример можно легко модифицировать, включив в него дополнительный программный код, а HTML-файлы можно заменить подходящими для данного приложения. Если требуются различные комбинации протоколов, то с помощью мастера конфигурирования TCP/IP можно сгенерировать новые библиотеку и вспомогательный код. Подробные (пошаговые) инструкции по настройке аппаратного обеспечения и запуску примеров кода приведены в руководстве пользователя, прилагаемого к набору средств разработки встраиваемых модемов (Modem-DK).

3.4. Техническая поддержка.

Если при использовании стека TCP/IP или мастера конфигурирования TCP/IP возникают какие-либо вопросы или появляются какие-либо проблемы, то можно связаться со службой технической поддержки, посетив сайт www.silabs.com и кликнув по ссылке “SUPPORT”. Если Вы разрабатываете приложение, которому требуются функции и протоколы, не поддерживаемые в настоящее время библиотекой TCP/IP, то, пожалуйста, свяжитесь с нами, и мы будем рады помочь Вам найти нужное решение.

4. Справочник по API стека TCP/IP

4.1. Группы функций

Функции стека TCP/IP разделены на следующие группы:

`Socket Functions`	Функции, осуществляющие открытие, закрытие и управление сокетами.
`Modem Functions`	Функции, осуществляющие управление соединением между МК и модемом.
`PPP Functions`	Функции, осуществляющие открытие, закрытие и управление PPP-соединением.
`Application Functions`	Функции, осуществляющие запуск и использование служб прикладного уровня (HTTP Web-сервер, FTP-сервер, TFTP-клиент, e-mail SMTP-клиент).
`Callback Functions`	Обработчики событий, вызываемые стеком для уведомления служб прикладного уровня.
`VFILE Functions`	Функции, осуществляющие управление виртуальной файловой системой.
`Support Functions`	Функции, осуществляющие преобразование типов данных.

4.2. Типы данных

Стек TCP/IP использует следующие типы данных («Приложение B – Структуры данных стека TCP/IP» на стр.30 содержит подробную информацию о структурах данных):

`byte`	8-разрядное число без знака
`SCHAR`	8-разрядное число со знаком
`word16`	16-разрядное целое число без знака
`word32`	32-разрядное целое число без знака
`PCONST_BYTE`	Указатель на байт константы (*const unsigned char**)

Типы данных сокетов:

`PSOCKET_INFO`	Указатель на структуру *SOCKET_INFO_T*

Типы данных виртуальной файловой системы:

`VF_PTR`	Указатель на структуру *VF*
`POST_FP`	Указатель на функцию создания CGI-содержимого
`PF_PTR`	Указатель на структуру *POST_FUNCS*

Типы данных почтового SMTP-клиента:

`PSMTP_INFO`	Указатель на структуру *SMTP_INFO_T*

4.3. Важные замечания

- Функция *mn_init()* должна вызываться до вызова любой другой функции стека.
- Стек TCP/IP использует Timer0, Timer1, UART1 и PCA0. SFR-регистры, связанные с этими периферийными модулями, нельзя модифицировать после вызова функции *mn_init()*.
- Стек TCP/IP разрешает прерывания от PCA0 с нормальным уровнем приоритета и прерывания от UART1 с высоким уровнем приоритета. Эти прерывания недоступны для прикладной программы.
- Мастер конфигурирования TCP/IP автоматически настраивает скорость передачи данных UART и системный тактовый генератор в зависимости от выбранной системной тактовой частоты (SYSCLK). Если процедура инициализации системного генератора модифицируется с целью изменения системной тактовой частоты, то необходимо «вручную» изменить следующие константы в файле *mn_userconst.h*:
  - **th0_flash:tl0_flash** – Значение перезагрузки Таймера 0 в режиме 1 (16-разрядный таймер) с частотой тактирования таймера, равной SYSCLK/48. Это 16-разрядное значение должно быть установлено таким образом, чтобы таймер переполнялся через 10 мс (т.е. с частотой 100 Гц). Например, если SYSCLK = 98 МГц, то это значение следует установить равным (-98000000/48/100) = -20416 = 0xB040.
  - **uart_reload** – 8-разрядное значение перезагрузки UART1, получаемое от Таймера 1 в 8-разрядном режиме с автоматической перезагрузкой. Тактовым сигналом для UART1 является SYSCLK. В табл.1 приведены рекомендуемые скорости передачи данных и значения перезагрузки для выбранных значений SYSCLK.

Таблица 1. Выбор скорости передачи данных UART1

SYSCLK	Скорость передачи данных	Значение перезагрузки
3.0625 МГц	245760 бит/с	0xFA
24.5 МГц	307200 бит/с	0xD9
49 МГц	307200 бит/с	0xB1
98 МГц	307200 бит/с	0x61

Примечание: Мастер конфигурирования TCP/IP может генерировать соответствующие значения перезагрузки для любой конфигурации системного генератора. Чтобы предотвратить перезапись существующего проекта, следует направить вывод результатов работы мастера во временную папку.

- Если каркас кода, генерируемый мастером конфигурирования TCP/IP, встраивается без добавления дополнительного кода приложения, то следует ожидать предупреждений о наличии невызываемых функций обратного вызова. Эти предупреждения не будут появляться, если код приложения содержит вызовы функций стека прикладного уровня, например *mn_server()*.
- Если полный размер программного кода превышает 64 Кб, то проект должен использовать кодовые банки, при этом библиотека TCP/IP должна располагаться в обычном разделе памяти (банк 0). Полный объем кода в обычном разделе памяти не может превышать 32 Кб. Более подробная информация об использовании кодовых банков приведена в указаниях по применению “AN130: Code Banking Using the Keil 8051 Tools”. Если необходимая библиотека не умещается в обычном разделе памяти, то для решения этой проблемы следует связаться со службой технической поддержки Silicon Laboratories.

4.4. Функции для работы с сокетами

Стек TCP/IP обеспечивает непосредственный доступ к TCP- и UDP-сокетам с помощью следующих функций:

Примечание: Для всех проектов, использующих стек TCP/IP, единственной обязательной функцией является функция *`mn_init()`*. При использовании служб прикладного уровня (например, HTTP Web-сервер, FTP-сервер или TFTP-клиент) сокеты автоматически открываются и закрываются, когда это необходимо, без управления со стороны программного кода приложения.

4.4.1. mn_init

Описание: Выполняет все процедуры инициализации, требуемые стеком TCP/IP.

Примечание: Эта функция должна вызываться до вызова любой другой функции стека.

Прототип: *`int mn_init (void);`*

Возвращаемое значение: Возвращает *TRUE*, если инициализация прошла успешно, или отрицательное число в случае ошибки.

4.4.2. mn_open

Описание: Выделяет и (при необходимости) открывает TCP- или UDP-сокет.

Примечание: Модемное соединение и PPP-соединение должны быть установлены до открытия TCP-сокета.

Прототип: `SCHAR mn_open(byte[],word16, word16, byte, byte, byte, byte *, word16);`

Пример вызова:
```
socket_no =
mn_open(dest_ip,src_port,dest_port,open_mode,proto,type,recv_buff,
buff_len);
```

Параметры:

- *dest_ip* – IP-адрес назначения, по которому отправляются пакеты.
- *src_port* – Номер порта, используемый приложением. Это должен быть либо хорошо известный (стандартный) номер порта (см. RFC 1700), либо номер, превышающий значение 1024.
- *dest_port* – Номер порта, используемый на удаленной стороне (если он известен). Если номер порта на удаленной стороне не известен, то параметр *dest_port* следует установить равным нулю. Если порт назначения установлен равным нулю, то он будет определяться автоматически стеком TCP/IP.
- *open_mode* – Используется только TCP-сокетами и может принимать одно из следующих значений:
  - *ACTIVE_OPEN* – TCP-соединение инициируется стеком TCP/IP.
  - *PASSIVE_OPEN* – Стек TCP/IP ожидает, когда удаленная сторона инициирует TCP-соединение.
  - *NO_OPEN* – Стек TCP/IP помечает сокет, как прослушивающий, но не ожидает TCP-соединения.
- *proto* – Определяет тип сокета и может принимать одно из следующих значений:
  - *PROTO_TCP* – Выделяется TCP-сокет.
  - *PROTO_UDP* – Выделяется UDP-сокет.
- *type* – Должен быть установлен как *STD_TYPE.*
- *recv_buff* – Адрес буфера, используемого для сохранения принимаемых данных.
- *buff_len* – Размер буфера, используемого для сохранения принимаемых данных.

Возвращаемое значение: В случае успеха возвращает корректный номер сокета из интервала 0 … 126. Для получения указателя на заново выделенный сокет можно использовать макрос `MK_SOCKET_PTR()`. В случае неудачи возвращает один из следующих кодов ошибок:

- `NOT_SUPPORTED` – Был запрошен сокет для неподдерживаемого протокола.
- `NOT_ENOUGH_SOCKETS` – Сокеты недоступны.
- `TCP_OPEN_FAILED` – Попытка открыть TCP-сокет закончилась неудачно.

4.4.3. mn_send

Описание: Посылает данные ранее открытому сокету.

Примечание: Если этот сокет является TCP-сокетом, то вызов этой функции может привести к приему каких-либо данных. После каждого вызова этой функции следует проверять поле *recv_len* сокета на наличие превышающего ноль значения.

Прототип: `int mn_send(SCHAR, byte *, word16);`

Пример вызова: `status = mn_send(socket_no, msg_ptr, msg_len);`

Параметры:
- *socket_no* – Номер сокета, возвращаемый функцией `mn_open()`.
- *msg_ptr* – Адрес буфера, содержащего данные для передачи.
- *msg_len* – Количество передаваемых байт.

Возвращаемое значение: В случае успеха возвращает количество переданных байт. Если количество переданных байт равно нулю, то пакет необходимо послать повторно. В остальных случаях возвращает один из следующих кодов ошибок (все значения отрицательные):
- `BAD_SOCKET_DATA` – Функции был передан некорректный номер сокета.
- `SOCKET_NOT_FOUND` – Переданный номер сокета принадлежит неактивному сокету.
- `TCP_ERROR` – Пакет был послан более `TCP_RESEND_TRYS` раз без получения ACK (только TCP-сокеты).
- `TCP_TO_LONG` – Была предпринята попытка послать пакет, длина которого превышает максимальный размер TCP-сегмента (только TCP-сокеты).
- `TCP_NO_CONNECT` - Передача данных невозможна, т.к. TCP-соединение не установлено.

4.4.4. mn_recv

Описание: Принимает данные от ранее открытого сокета с фиксированным временем ожидания, равным *SOCKET_WAIT_TICKS*. Единицами установки времени ожидания являются системные отсчеты времени длительностью 10 мс.

Примечание: Эта функция будет выполняться до тех пор, пока не будет получен пакет от передающего сокета или пока не истечет таймаут. В процессе ожидания поступления пакета будет непрерывно вызываться функция обратного вызова `callback_app_recv_idle()`. Функция `mn_recv()` немедленно остановит процесс ожидания и завершит свою работу, если `callback_app_recv_idle()` возвратит значение `NEED_TO_EXIT`.

Примечание: В случае TCP-сокетов будут автоматически отправлены ответы на управляющие TCP-пакеты, такие как SYN и FIN. Эта функция может возвратить значение `NEED_TO_LISTEN`, означающее, что сокет должен прослушивать соединение на наличие отклика с удаленной стороны прежде чем передавать другой пакет.

Прототип: `int mn_recv(SCHAR, byte *, word16);`

Пример вызова: `status = mn_recv(socket_no, buff_ptr, buff_len);`

Параметры:
- *socket_no* – Номер сокета, возвращаемый функцией `mn_open()`.
- *buff_ptr* – Адрес буфера приема.
- *buff_len* – Размер буфера приема.

Возвращаемое значение: В случае успеха возвращает количество принятых байт. В остальных случаях возвращает один из следующих кодов ошибок (все значения отрицательные):
- `BAD_SOCKET_DATA` – Функции был передан некорректный номер сокета.
- `SOCKET_NOT_FOUND` – Переданный номер сокета принадлежит неактивному сокету.
- `SOCKET_TIMED_OUT` – Таймаут сокета истек до получения пакета.
- `NEED_TO_LISTEN` – Отклик на полученный пакет был послан автоматически и сокет должен ожидать ответа (только TCP-сокеты).
- `NEED_TO_EXIT` – Функция обратного вызова `callback_app_recv_idle()` возвратила значение `NEED_TO_EXIT`.
- *Любое другое отрицательное значение* - Произошла либо ошибка контрольной суммы, либо ошибка FCS, либо TCP-соединение закрыто.

4.4.5. mn_recv_wait

Описание: Отличается от функции `mn_recv()` лишь тем, что использует время ожидания, передаваемое в виде четвертого параметра.

Прототип: `int mn_recv_wait(SCHAR, byte *, word16, word16);`

Пример вызова: `status = mn_recv_wait(socket_no, buff_ptr, buff_len, wait_ticks);`

Параметры:
- *socket_no* – Номер сокета, возвращаемый функцией `mn_open()`.
- *buff_ptr* – Адрес буфера приема.
- *buff_len* – Размер буфера приема.
- *wait_ticks* – Длительность таймаута в отсчетах системного времени.

Возвращаемое значение: См. описание функции `mn_recv()`.

4.4.6. mn_close

Описание: Закрывает ранее открытый сокет.

Прототип: `int mn_close(SCHAR);`

Пример вызова: `status = mn_close(socket_no);`

Параметры: • *socket_no* – Номер сокета, возвращаемый функцией `mn_open()`.

Возвращаемое значение: В случае успеха возвращает `FALSE`. В остальных случаях возвращает один из следующих кодов ошибок (все значения отрицательные):
- `BAD_SOCKET_DATA` – Функции был передан некорректный номер сокета.
- `SOCKET_NOT_FOUND` – Переданный номер сокета принадлежит неактивному сокету.

4.4.7. mn_abort

Описание: Немедленно закрывает ранее открытый сокет без осуществления корректного закрытия или посылки FIN (только TCP). Для UDP-сокетов функции `mn_close()` и `mn_abort()` идентичны.

Прототип: `int mn_abort(SCHAR);`

Пример вызова: `status = mn_abort(socket_no);`

Параметры: • *socket_no* – Номер сокета, возвращаемый функцией `mn_open()`.

Возвращаемое значение: В случае успеха возвращает `FALSE`. В остальных случаях возвращает один из следующих кодов ошибок (все значения отрицательные):
- `BAD_SOCKET_DATA` – Функции был передан некорректный номер сокета.
- `SOCKET_NOT_FOUND` – Переданный номер сокета принадлежит неактивному сокету.

4.4.8. mn_find_socket

Описание: Используется для получения указателя на сокет по заданным порту источника, порту назначения, IP-адресу назначения и типу сокета.

Прототип: `PSOCKET_INFO mn_find_socket(word16, word16, byte*, byte);`

Пример вызова:
```
socket_ptr = mn_find_socket(src_port, dest_port, dest_ip,
socket_type);
```

Параметры: • *src_port* – Локальный номер порта.
• *dest_port* – Номер порта на удаленной стороне.
• *dest_ip* – IP-адрес удаленного устройства.
• *socket_type* – Тип сокета, заданный при открытии сокета. Может принимать одно из следующих значений:
- `PROTO_TCP` – TCP-сокет.
- `PROTO_UDP` – UDP-сокет.

Возвращаемое значение: В случае успеха возвращает указатель на соответствующий сокет. В остальных случаях возвращает `PTR_NULL`.

4.5. Функции для работы с модемом.

При использовании модема в качестве устройства физического уровня стек TCP/IP требует установления соединения с модемом до установления соединения с помощью протоколов более высокого уровня, таких как PPP и TCP. Установление соединения с модемом заставляет его набрать номер и подключиться к удаленной сети или принять входящий вызов. Для управления соединением с модемом на физическом уровне стек TCP/IP предлагает следующие функции:

4.5.1. mn_modem_connect

Описание: Устанавливает соединение между МК и модемом и осуществляет инициализацию модема.

Порядок инициализации модема для режима ответа на вызов:
1. Инициализировать код страны и протокол; затем послать строку *MODEM_INIT_ANSWER*.
2. Ожидать строки “OK”, “RING” и “CONNECT”.
3. Если *USE_PASSWORD = 1* и *USE_PAP = 0*, то будут посланы *ANS_LOGIN_PROMPT* и *ANS_PASSWORD_PROMPT*, а возвращенные строки будут сравниваться с именем пользователя и паролем, которые хранятся в *LOGIN_NAME* и *PASSWORD* соответственно.

Порядок инициализации модема для режима вызова:
1. Инициализировать код страны и протокол; затем послать строку *MODEM_INIT_DIAL*.
2. Ожидать “OK”, затем послать *MODEM_DIAL*.
3. Ожидать “CONNECT”.
4. Если *USE_PASSWORD = 1* и *USE_PAP = 0*, то имя пользователя и пароль, хранящиеся в *LOGIN_NAME* и *PASSWORD* соответственно, будут использоваться для подключения к удаленному серверу.

Примечание: Эта функция инициализирует модем с помощью строк, задаваемых в файле *mn_userconst.h*. По умолчания максимальная длина строки составляет 10 символов.

Прототип:
```
int mn_modem_connect(byte);
```

Пример вызова:
```
status = mn_modem_connect(connect_mode);
```

Параметры:
- *connect_mode* – Определяет, будет ли модем отвечать на входящие вызовы, или будет инициировать исходящие вызовы. Может принимать одно из следующих значений:
  - *ANSWER_MODE* – Модем настраивается таким образом, чтобы отвечать на входящие вызовы.
  - *DAIL_MODE* – Модем настраивается таким образом, чтобы устанавливать связь с удаленным сервером или Internet-провайдером.

Возвращаемое значение: В случае успеха возвращает *TRUE*. В остальных случаях возвращает отрицательное число, что означает невозможность установления соединения.

4.5.2. mn_modem_disconnect

Описание: Закрывает соединение между МК и модемом и заставляет модем отключиться от телефонной линии.

Примечание: Все TCP-сокеты и PPP-соединения необходимо закрыть до вызова этой функции.

Прототип: `void mn_modem_disconnect(void);`

Пример вызова: `mn_modem_disconnect();`

4.5.3. mn_modem_send_string

Описание: Посылает модему строку инициализации.

Примечание: Эта строка должна заканчиваться символом возврата каретки (‘\r’).

Прототип: `void mn_modem_send_string(PCONST_BYTE, word16);`

Пример вызова: `mn_modem_send_string(str, len);`

Параметры:
- *str* – Адрес первого символа в массиве символьных констант (например, “ATM1L1\r”).
- *len* – Количество байт в *str*, включая символ возврата каретки (‘\r’).

4.5.4. mn_modem_wait_reply

Описание: Ожидает определенного отклика от модема в течение заданного таймаута.

Примечание: Единицами установки таймаута являются системные отсчеты времени длительностью 10 мс.

Прототип: `int mn_modem_wait_reply(PCONST_BYTE, word16, word16);`

Пример вызова: `status = mn_modem_wait_reply(str, len, timeout);`

Параметры:
- *str* – Адрес первого символа в массиве символьных констант (в строке). Ответ, полученный от модема, сравнивается с этой строкой, чтобы определить, не произошла ли ошибка.
- *len* – Количество байт в *str*, включая символ возврата каретки (‘\r’).
- *timeout* – Максимальное количество системных отсчетов времени длительностью 10 мс, задающее таймаут ожидания при неполучении ответа от модема.

Возвращаемое значение: В случае успеха возвращает *TRUE*. В остальных случаях возвращает отрицательное число, означающее, что либо истек таймаут, либо ответ модема не соответствует содержимому *str*.

4.6. Функции управления PPP-соединением.

Стек TCP/IP позволяет осуществлять выбор между протоколами канального уровня PPP и SLIP. Если выбран протокол SLIP, то все управление на канальном уровне автоматически осуществляется стеком. Если выбран протокол PPP, то прежде, чем открыть любой TCP-сокет, прикладная программа должна использовать следующие функции для установления соединения с удаленным клиентом/сервером:

Примечание: Если разрешен протокол аутентификации пароля (Password Authentication Protocol – PAP), то прикладная программа должна добавить имя пользователя и пароль в таблицу PAP до открытия PPP-соединения. Если PAP отключен, то аутентификация должна быть разрешена на уровне модема. Более подробная информация об аутентификации на физическом уровне приведена в разделе 4.5.1 на стр.11.

4.6.1. mn_ppp_open

Описание: Устанавливает PPP-соединение с удаленным PPP-клиентом/сервером.

Примечание: Все модемные соединения и процедура инициализации стека должны быть завершены до вызова этой функции.

Примечание: Константа *USE_PAP* определяет, будет ли использоваться протокол аутентификации пароля. Если *USE_PAP = TRUE*, то перед вызовом этой функции необходимо вызвать функцию *mn_ppp_add_pap_user()*. Если *USE_PAP = FALSE*, то регистрационные данные обрабатываются функцией *mn_modem_connect()*, используя при этом информацию, задаваемую в файле *mn_userconsts.h*.

Прототип: `int mn_ppp_open(byte);`

Пример вызова: `status = = mn_ppp_open(open_mode);`

Параметры:
- *open_mode* – Определяет, будет ли локальное PPP-устройство клиентом или сервером. Может принимать одно из следующих значений:
  - *ACTIVE_OPEN* – Локальный PPP-клиент пытается установить соединение с удаленным PPP-сервером. Начальная комбинация имени пользователя и пароля, добавленная с помощью функции *mn_ppp_add_pap_user()*, будет использоваться для подключения к удаленному серверу.
  - *PASSIVE_OPEN* – Локальный PPP-сервер ожидает, когда удаленный PPP-клиент и инициирует соединение. Будут проверяться все комбинации имени пользователя и пароля, добавленные с помощью функции *mn_ppp_add_pap_user()*.

Возвращаемое значение: В случае успеха возвращает *TRUE*. В остальных случаях возвращает *FALSE*.

4.6.2. mn_ppp_close

Описание: Закрывает PPP-соединение без ожидания ответа и сбрасывает PPP-устройство.

Примечание: Эту функцию следует вызывать только в том случае, если функция *mn_ppp_open()* была завершена успешно.

Прототип: `void mn_ppp_close(void);`

Пример вызова: `mn_ppp_close();`

4.6.3. mn_ppp_reset

Описание: Сбрасывает PPP-устройство.

Примечание: Эту функцию следует вызывать только в случае ошибки, препятствующей вызову функции *mn_ppp_close*.

Прототип: `void mn_ppp_reset(void);`

Пример вызова: `mn_ppp_reset();`

4.6.4. mn_ppp_add_pap_user

Описание: Добавляет имя пользователя и пароль в таблицу протокола аутентификации пароля (PAP).

Примечание: По умолчанию максимальная длина строки составляет двадцать символов, включая символ конца строки (‘0’).

Прототип: `byte mn_ppp_add_pap_user(char*, char*);`

Пример вызова: `status = mn_ppp_add_pap_user(username, password);`

Параметры:
- *username* – Символьная строка, содержащая имя пользователя и заканчивающаяся символом конца строки (‘/0’).
- *password* – Символьная строка, содержащая пароль и заканчивающаяся символом конца строки (‘/0’).

Возвращаемое значение: Если добавление имени пользователя и пароля завершилось успешно, то возвращает *TRUE*. В остальных случаях возвращает *FALSE*.

4.6.5. mn_ppp_del_pap_user

Описание: Удаляет имя пользователя и пароль из таблицы протокола аутентификации пароля (PAP).

Примечание: По умолчанию максимальная длина строки составляет двадцать символов, включая символ конца строки (‘0’).

Прототип: `byte mn_ppp_del_pap_user(char*);`

Пример вызова: `status = mn_ppp_del_pap_user(username);`

Параметры:
- *username* – Символьная строка, содержащая имя пользователя для комбинации имя пользователя/пароль и заканчивающаяся символом конца строки (‘/0’).

Возвращаемое значение: Если удаление имени пользователя и пароля завершилось успешно, то возвращает *TRUE*. В остальных случаях возвращает *FALSE*, означающее, что данное имя пользователя не найдено.

4.7. Функции прикладного уровня

TCP/IP обеспечивает управление службами прикладного уровня, такими как HTTP Web-сервер, FTP-сервер, TFTP-клиент и почтовый SMTP-клиент, с помощью следующих функций:

Примечание: Эти функции, вместе с функциями обратного вызова и функциями виртуальной файловой системы, описанными в следующих двух разделах, обеспечивают полное управление службами прикладного уровня.

4.7.1. mn_server

Описание: Используется для запуска служб прикладного уровня. При вызове этой функции будут запущены все разрешенные серверные приложения, такие как HTTP Web-сервер и FTP-сервер. Клиентские приложения, такие как TFTP-клиент и почтовый SMTP-клиент, запускаются с помощью функций, описанных в этом разделе. Выполнение этих функций не завершается до тех пор, пока не произойдет ошибка PPP-протокола или пока функция обратного вызова (*`callback_app_server_idle()`* или *`callback_app_server_process_packet()`*) не возвратит значение *NEED_TO_EXIT*.

Примечание: Эта функция при необходимости автоматически открывает и закрывает сокеты для обработки входящих запросов. Любые дополнительные сокеты (например, UDP-сокеты), которые используются прикладной программой во время простоя HTTP- или FTP-сервера, должны быть открыты до вызова этой функции.

Важное примечание для FTP-сервера:

- FTP-сервер разработан для работы с графическим интерфейсом пользователя ОС Windows и с FTP-клиентами, использующими интерфейс командной строки. FTP-сервер возвращает списки каталогов в стандартном формате ОС Unix. Если FTP-клиент поддерживает несколько форматов, то должен быть выбран стандартный формат ОС Unix.
- Поддерживаются следующие FTP-команды: USER, QUIT, RETR, STOR, DELE, PORT, TYPE, MODE, STRU, NOOP, PWD, LIST и (не обязательно) PASS. FTP-сервер всегда будет проверять имя пользователя и пароль, задаваемые массивом ftp_user в исходном файле *mn_vars.c*. Этот массив должен быть инициализирован на этапе компиляции всеми допустимыми именами пользователей и паролями.
- Виртуальная файловая система не использует подкаталоги; поэтому PWD всегда возвращает “/” и CWD запрещена.
- Для временного хранения данных FTP-сервер использует буфер, размер которого устанавливается константой *ftp_buffer_len* в файле *userconst.h*. Этот буфер должен быть достаточно большим, чтобы сохранить максимальный по объему файл из тех, которые Вы собираетесь принять. После приема файла для него выделяется память с помощью функции *malloc()* и создается элемент виртуальной файловой системы с сегментом памяти *VF_PTYPE_DYNAMIC*. Удаление файла из виртуальной файловой системы приведет к освобождению всей динамически выделенной памяти, связанной с этим файлом.

Прототип: `int mn_server(void);`

Пример вызова: `status = mn_server();`

Возвращаемое значение: Корректными возвращаемыми значениями являются следующие значения:

- *FALSE*- Либо функция *`callback_app_server_idle()`*, либо функция *`callback_app_server_process_packet()`* возвратила значение *NEED_TO_EXIT*.
- *PPP_LINK_DOWN* – PPP-соединение было разорвано.

4.7.2. mn_http_find_value

Описание: Ищет пары типа «*имя_поля* = *значение_поля*»; подбирает пару для заданного имени поля и копирует декодированное значение поля в заданный буфер. Функция создания CGI-содержимого использует эту функцию для определения значений переменных, переданных с web-сайта.

Прототип: `int mn_http_find_value(byte*, byte*, byte*);`

Пример вызова: `status = mn_http_find_value(source_ptr, field_name, field_value);`

Параметры:
- *source_ptr* – Адрес буфера, содержащего тело сообщения, в котором будет осуществляться поиск.
- *field_name* – Символьная строка, содержащая *имя_поля* и заканчивающаяся символом конца строки (‘/0’).
- *field_value* – Строковый буфер, в который будет копироваться *значение_поля*.

Возвращаемое значение: Если *имя_поля* обнаружено, то возвращает *TRUE*. В остальных случаях возвращает *FALSE*.

4.7.3. mn_tftp_get_file

Описание: Получает файл от удаленного TFTP-сервера и сохраняет его в заданном буфере.

Прототип: `long mn_tftp_get_file(byte*, byte*, byte*, long);`

Пример вызова: `num_bytes = mn_tftp_get_file(ip_addr, filename, buffer, buff_len);`

Параметры:
- *ip_addr* – Указатель на 4-байтный символьный массив, содержащий IP-адрес TFTP-сервера.
- *filename* – Строка поиска, содержащая имя файла и заканчивающаяся символом конца строки (‘/0’).
- *buffer* – Указатель на буфер в ОЗУ, предназначенный для сохранения файла.
- *buff_len* – Количество байт в буфере.

Возвращаемое значение: Возвращает количество полученных байт. В остальных случаях возвращает отрицательное число.

4.7.4. mn_smtp_start_session

Описание: Открывает TCP-соединение с SMTP-сервером, заданным в файле *mn_userconst.h*.

Примечание: Модемное соединение и PPP-соединение должны быть установлены до вызова этой функции.

Прототип: `SCHAR mn_smtp_start_session(word16);`

Пример вызова: `socket_num = mn_smtp_start_session(port);`

Параметры:
- *port* – Номер порта, используемый SMTP-сокетом. Может принимать значение от 1025 до 65535.

Возвращаемое значение: Возвращает номер сокета в случае успеха или отрицательное число в случае ошибки.

4.7.5. mn_ smtp_end_session

Описание: Закрывает соединение с SMTP-сервером, открытое с помощью функции *mn_smtp_start_session()*.

Прототип: `void mn_smtp_end_session(SCHAR);`

Пример вызова: `socket_num = mn_smtp_end_session(socket_num);`

Параметры: ● *socket_num* – Номер сокета, возвращаемый функцией *mn_smtp_start_session()*.

4.7.6. mn_smtp_send_mail

Описание: Отправляет e-mail-сообщение с необязательным вложением почтовому SMTP-серверу.

Примечание: Перед отправкой e-mail должна быть вызвана и успешно завершена функция *mn_smtp_start_session()*.

Прототип: `int mn_smtp_send_mail(SCHAR, PSMTP_INFO);`

Пример вызова: `status = mn_smtp_send_mail(socket_num, mail_info_ptr);`

Параметры: ● *socket_num* – Номер сокета, возвращаемый функцией *mn_smtp_start_session()*.
● *mail_info_ptr* – Адрес структуры *SMTP_INFO_T*, которая должна быть проинициализирована.

Возвращаемое значение: Возвращает ноль или положительное число в случае успеха или отрицательное число в случае ошибки.

4.8. Функции обратного вызова

Стек TCP/IP использует функции обратного вызова для уведомления прикладной программы о различных событиях. На рис.1 приведена схема выполнения кода функций обратного вызова. В любом проекте, который использует службы прикладного уровня, обеспечиваемые стеком TCP/IP, должны быть определены перечисленные ниже четыре функции обратного вызова. Эти функции обратного вызова должны содержать код обработки соответствующего события.

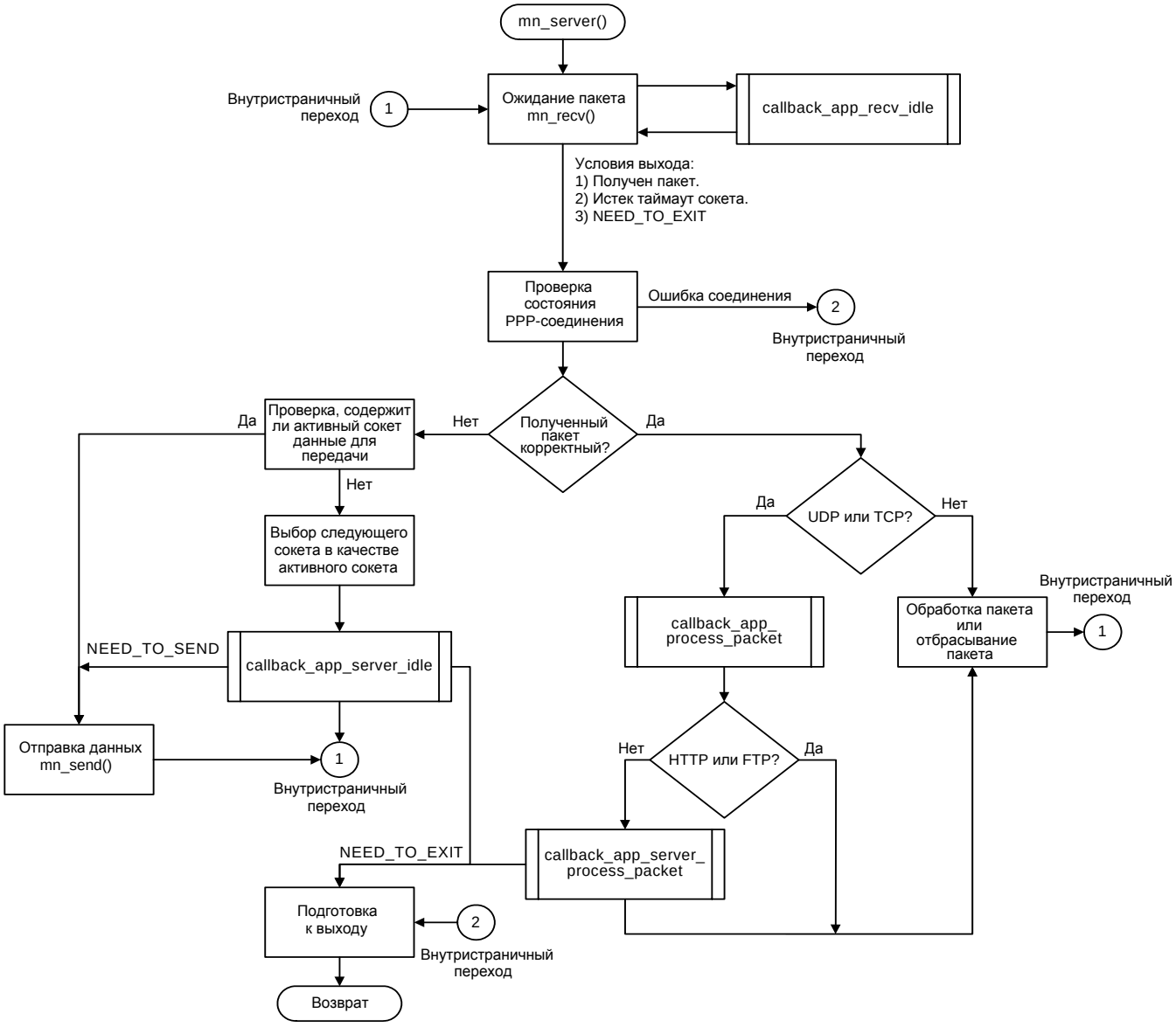

Рисунок 1. Блок-схема выполнения функций обратного вызова

4.8.1. callback_app_process_packet

Описание: Вызывается стеком TCP/IP после получения любого TCP- или UDP-пакета.

Примечание: Возвращаемое значение игнорируется для UDP-пакетов.

Прототип: `byte callback_app_process_packet(PSOCKET_INFO);`

Пример вызова: `status = callback_app_process_packet(socket_ptr);`

Параметры: ● *socket_ptr* – Указатель на сокет, который содержит данные.

Возвращаемое значение: Корректными являются следующие возвращаемые значения:

- `NEED_IGNORE_PACKET` – Стек TCP/IP не будет отсылать ACK на TCP-пакет.
- `Любое Другое Значение` – Стек TCP/IP будет отсылать ACK на TCP-пакет.

4.8.2. callback_app_server_process_packet

Описание: Вызывается стеком TCP/IP после получения любого TCP- или UDP-пакета, если они не являются HTTP- или FTP-пакетами. HTTP- или FTP-пакеты обрабатываются сервером автоматически.

Прототип: `SCHAR callback_app_server_process_packet(PSOCKET_INFO);`

Пример вызова: `status = callback_app_server_process_packet(socket_ptr);`

Параметры: ● *socket_ptr* – Указатель на сокет, который содержит данные.

Возвращаемое значение: Корректными являются следующие возвращаемые значения:

- `NEED_TO_EXIT` – Функция *mn_server()* будет завершена немедленно, возвратив управление функции *main()*.
- `Любое Другое Значение` – Сервер отбросит пакет.

4.8.3. callback_app_recv_idle

Описание: Вызывается повторно в то время, пока функция *mn_recv()* ожидает данные. Эта функция должна использоваться только для низкоприоритетных задач. Любые высокоприоритетные задачи должны быть размещены в процедуре обработки прерывания.

Прототип: `SCHAR callback_app_recv_idle(void);`

Пример вызова: `status = callback_app_recv_idle();`

Возвращаемое значение: Корректными являются следующие возвращаемые значения:

- `NEED_TO_EXIT` – Функция *mn_recv()* будет завершена немедленно. Если сервер запущен, то прекратит ожидание данных и перейдет к следующему состоянию.
- `Любое Другое Значение` – Функция *mn_recv()* будет и далее ожидать данные.

4.8.4. callback_app_server_idle

Описание: Вызывается периодически из функции *mn_server()* тогда, когда она не передает/принимает данные. Эта функция должна использоваться только для низкоприоритетных задач. Любые высокоприоритетные задачи должны быть размещены в процедуре обработки прерывания.

Прототип: `SCHAR callback_app_server_idle(PSOCKET_INFO*);`

Пример вызова: `status = callback_app_server_idle(psocket_ptr);`

Параметры: • *psocket_ptr* – Указатель на указатель на сокет, который может использоваться для передачи данных.

Примечание: Дескриптор сокета можно переназначить на другой сокет (например, **psocket_ptr = new_socket_ptr*;).

Возвращаемое значение: Корректными являются следующие возвращаемые значения:

- `NEED_TO_SEND` – Стек TCP/IP немедленно отправит в сокет накопленные данные.
- `NEED_TO_EXIT` – Функция *mn_server()* будет завершена немедленно, возвратив управление функции *main()*.
- `Любое Другое Значение` – Функция *mn_server()* будет продолжать нормальное функционирование.

4.9. Функции виртуальной файловой системы (VFILE)

Стек TCP/IP включает виртуальную файловую систему, доступную прикладной программе и службам прикладного уровня, таким как HTTP Web-сервер и FTP-сервер. Файлы, добавленные в эту файловую систему, хранятся в виде двоичных массивов во Flash-памяти или в ОЗУ и могут запрашиваться web-браузером или FTP-клиентом. Это позволяет встраивать рисунки, апплеты и другое содержимое в статичные или динамичные HTML-страницы.

Чтобы добавить статичное содержимое в виртуальную файловую систему, оно прежде всего должно быть преобразовано в файловый массив с помощью утилиты HTML2C. Утилита HTML2C читает файл содержимого (например, *image.gif*) и генерирует два файла (например, *image.c* и *image.h*), которые могут быть добавлены в проект. Для добавления в виртуальную файловую систему файла со статичным содержимым необходимо выполнить следующие действия:

1. Добавить в проект исходный C-файл (например, *image.c*).
2. Включить заголовочный файл (например, *image.h*) в начало файла *main.c* с помощью директивы *#include*.
3. Во время рабочего цикла добавить файл в файловую систему, используя *mn_vf_set_entry()* или *mn_vf_set_ram_entry()*. Эти функции преобразуют начальный адрес и длину файлового массива в имя файла, которое доступно из web-браузера или FTP-клиента.

Для добавления статичных файлов в виртуальную файловую систему, удаления их из виртуальной файловой системы и для доступа к этим файлам можно использовать следующие функции:

Функции файловой системы HTTP/FTP-сервера:

Виртуальная файловая система позволяет создавать содержимое динамичных Web-страниц с помощью CGI-скриптов. Если HTTP Web-сервер принимает распознаваемое имя скрипта, то он вызывает функцию «создания содержимого» для генерации запрашиваемого содержимого. Запросы к HTTP Web-серверу можно отправлять либо в виде части HTML-формы, либо непосредственно в виде URL. Ниже приведен пример web-браузера, запрашивающего динамичные данные из скрипта под именем «*get_data*».

http://10.10.10.163/get_data?type=temperature

Если запрос CGI-скрипта передается в виде URL, то весь текст после знака вопроса интерпретируется как аргументы, передаваемые этому скрипту. В приведенном выше примере «type» представляет собой имя поля, а «temperature» - значение поля. Различные аргументы имя_поля/значение_поля отделяются друг от друга точкой с запятой. Имя скрипта «*get_data*» распознается HTTP-сервером, т.к. оно было добавлено в файловую систему прикладной программой. CGI-скрипт можно добавить в файловую систему и удалить из файловой системы с помощью следующих функций:

Функции для работы с CGI-скриптами:

Функция *mn_pf_set_entry()* преобразует имя скрипта в указатель на функцию «создания содержимого», которая вызывается всякий раз, когда имя скрипта появляется в URL или в поле ACTION HTML-формы. Функция «создания содержимого» использует аргументы, следующие за знаком вопроса, для генерации запрашиваемых данных. Как только эта функция заканчивает генерацию данных, она определяет начальный адрес и длину данных, которые требуется отправить обратно браузеру. Стек TCP/IP управляет всем дальнейшим взаимодействием с браузером до тех пор, пока не будет получен новый запрос.

4.9.1. mn_vf_get_entry

Описание: Используется для получения указателя на структуру *VF*, соответствующую файлу в виртуальной файловой системе. Структура *VF* содержит информацию о файле, такую как начальный адрес, размер файла и сегмент памяти. «Приложение A – Пользовательские константы стека TCP/IP» на стр.27 содержит описание структуры *VF*.

Примечание: Эту функцию нельзя вызывать из процедуры обработки прерывания.

Прототип: `VF_PTR mn_vf_get_entry(byte*);`

Пример вызова: `pVF = mn_vf_get_entry(filename);`

Параметры: ● *filename* – Строка, содержащая имя требуемого файла (например, *index.html*) и заканчивающаяся символом конца строки (‘/0’).

Возвращаемое значение: Возвращает корректный указатель на структуру *VF* или *PTR_NULL* в случае, если строка поиска не совпадает ни с одним из имен файлов, добавленных в файловую систему.

4.9.2. mn_vf_set_entry

Описание: Используется для добавления в виртуальную файловую систему файла, хранящегося во встроенной Flash-памяти.

Примечание: Эту функцию нельзя вызывать из процедуры обработки прерывания.

Прототип: `VF_PTR mn_vf_set_entry(byte*, word16, PCONST_BYTE, byte);`

Пример вызова: `pVF = mn_vf_set_entry(filename, file_size, file_ptr, mem_seg);`

Параметры: ● *filename* – Строка, содержащая имя требуемого файла (например, *index.html*) и заканчивающаяся символом конца строки (‘/0’).
● *file_size* – Количество байт в файле.
● *file_ptr* – Указатель на начало файла.
● *mem_seg* – Тип памяти, в которой хранится файл. Должно быть установлено *VF_PTYPE_FLASH*.

Возвращаемое значение: Возвращает корректный указатель на структуру *VF* или *PTR_NULL* в случае, если в файловую систему уже добавлено максимально возможное число файлов.

4.9.3. mn_vf_set_ram_entry

Описание: Используется для добавления в виртуальную файловую систему файла, хранящегося в ОЗУ.

Примечание: Эту функцию нельзя вызывать из процедуры обработки прерывания.

Прототип: `VF_PTR mn_vf_set_ram_entry(byte*, word16, byte*, byte);`

Пример вызова: `pVF = mn_vf_set_ram_entry(filename, file_size, file_ptr, mem_seg);`

Параметры:
- *filename* – Строка, содержащая имя требуемого файла (например, *index.html*) и заканчивающаяся символом конца строки (‘/0’).
- *file_size* – Количество байт в файле.
- *file_ptr* – Указатель на начало файла.
- *mem_seg* – Тип памяти, в которой хранится файл. Должно быть установлено в ноль.

Возвращаемое значение: Возвращает корректный указатель на структуру *VF* или *PTR_NULL* в случае, если в файловую систему уже добавлено максимально возможное число файлов.

4.9.4. mn_vf_del_entry

Описание: Используется для удаления файла из виртуальной файловой системы. Файлы, удаленные из виртуальной файловой системы, будут невидимы для HTTP- или FTP-сервера. FTP-сервер сохраняет принятые файлы в динамически выделяемой памяти ОЗУ. Если удаляемый файл хранится в динамически выделенной памяти ОЗУ, то этот буфер памяти будет освобожден.

Примечание: Эту функцию нельзя вызывать из процедуры обработки прерывания.

Прототип: `SCHAR mn_vf_del_entry(byte*);`

Пример вызова: `status = mn_vf_del_entry(filename);`

Параметры:
- *filename* – Строка, содержащая имя требуемого файла (например, *index.html*) и заканчивающаяся символом конца строки (‘/0’).

Возвращаемое значение: Возвращает одно из следующих значений:
- `TRUE` – Файл успешно удален.
- `FALSE` – Файл не найден.
- `VFILE_ENTRY_IN_USE` – Файл используется и его нельзя удалять.

4.9.5. mn_pf_get_entry

Описание: Используется для получения указателя на функцию создания CGI-содержимого.

Примечание: Эту функцию нельзя вызывать из процедуры обработки прерывания.

Прототип: `POST_FP mn_pf_get_entry(byte*);`

Пример вызова: `function_ptr = mn_pf_get_entry(function_name);`

Параметры: ● *function_name* – Строка, содержащая имя требуемой функции и заканчивающаяся символом конца строки (‘/0’).

Возвращаемое значение: Возвращает корректный указатель на функцию создания CGI-содержимого или *PTR_NULL* в случае, если строка поиска не совпадает ни с одним из имен функций, добавленных в файловую систему.

4.9.6. mn_pf_set_entry

Описание: Используется для добавления функции создания CGI-содержимого в виртуальную файловую систему.

Примечание: Эту функцию нельзя вызывать из процедуры обработки прерывания.

Прототип: `PF_PTR mn_pf_set_entry(byte*, POST_FP);`

Пример вызова: `pPFStruct = mn_pf_set_entry(name, function_ptr);`

Параметры: ● *function_name* – Строка, содержащая имя требуемой функции и заканчивающаяся символом конца строки (‘/0’).
● *function_ptr* – Указатель на начало функции.

Возвращаемое значение: Возвращает корректный указатель на структуру *POST_FUNCS* или *PTR_NULL* в случае, если в файловую систему уже добавлено максимально возможное число функций. «Приложение A – Пользовательские константы стека TCP/IP» на стр.27 содержит описание структуры *PF*.

4.9.7. mn_pf_del_entry

Описание: Используется для удаления функции создания CGI-содержимого из виртуальной файловой системы.

Примечание: Эту функцию нельзя вызывать из процедуры обработки прерывания.

Прототип: `byte mn_pf_del_entry(byte*);`

Пример вызова: `status = mn_pf_del_entry(function_name);`

Параметры: ● *function_name* – Строка, содержащая имя требуемой функции и заканчивающаяся символом конца строки (‘/0’).

Возвращаемое значение: Возвращает *TRUE*, если функция удалена, или *FALSE*, если имя функции не обнаружено.

4.10. Вспомогательные функции

Стек TCP/IP предлагает следующие вспомогательные функции, используемые для преобразования строк:

4.10.1. mn_ustoa – unsigned int to ascii

Описание: Преобразует целое число без знака в строку ASCII-символов.

Примечание: Эту функцию нельзя вызывать из процедуры обработки прерывания.

Прототип: `byte mn_ustoa(byte*, word16);`

Пример вызова: `num_bytes = mn_ustoa(dest_buff, source);`

Параметры:
- *dest_buff* – Адрес символьного массива, в котором будет сохранена полученная строка, заканчивающаяся символом конца строки (‘/0’).
- *source* – Целое число без знака, которое будет преобразовываться в строку.

Возвращаемое значение: Возвращает количество байт, добавленных в *dest_buff*.

4.10.2. mn_uctoa – unsigned char to ascii

Описание: Преобразует беззнаковый символ в строку ASCII-символов.

Примечание: Эту функцию нельзя вызывать из процедуры обработки прерывания.

Прототип: `byte mn_uctoa(byte*, word16);`

Пример вызова: `num_bytes = mn_uctoa(dest_buff, source);`

Параметры:
- *dest_buff* – Адрес символьного массива, в котором будет сохранена полученная строка, заканчивающаяся символом конца строки (‘/0’).
- *source* – Беззнаковый символ, который будет преобразовываться в строку.

Возвращаемое значение: Возвращает количество байт, добавленных в *dest_buff*.

4.10.3. mn_getMyIPAddr_func

Описание: Формирует строку с текущим IP-адресом в следующем формате: “255.255.255.255”.

Примечание: Эту функцию нельзя вызывать из процедуры обработки прерывания.

Прототип: `word16 mn_getMyIPAddr_func(byte**);`

Пример вызова: `num_bytes = mn_getMyIPAddr_func(dest_buff);`

Параметры: ● *dest_buff* – Указатель на указатель на символьный массив, в котором будет сохранена строка с IP-адресом, заканчивающаяся символом конца строки (‘/0’).

Возвращаемое значение: Возвращает количество байт, добавленных в *dest_buff*.

4.10.4. mn_atous – ascii to unsigned int

Описание: Преобразует строку ASCII-символов в целое число без знака.

Примечание: Эту функцию нельзя вызывать из процедуры обработки прерывания.

Прототип: `word16 mn_atous(byte*);`

Пример вызова: `result = mn_atous(src_buff);`

Параметры: ● *src_buff* – Адрес строки, заканчивающейся символом конца строки (‘/0’), которая будет преобразовываться.

Возвращаемое значение: Возвращает целое число без знака, представляющее собой значение, указанное в строке.

Приложение A – Пользовательские константы стека TCP/IP

Пользовательские константы стека TCP/IP, содержащиеся в заголовочном файле *mn_userconst.h*, позволяют пользователю настраивать стек в соответствии с требованиями конкретного приложения. Большинство констант из заголовочного файла *mn_userconst.h* также можно устанавливать, используя мастер конфигурирования TCP/IP.

Количество пользовательских констант в файле *mn_userconst.h* будет изменяться в зависимости от генерируемой библиотеки. В приведенных ниже таблицах перечислены все возможные константы, однако сгенерированный заголовочный файл будет содержать только те константы, которые используются генерируемой библиотекой.

Таблица 2. Константы установки IP-адресов

Имя константы	Описание
IP_SRC_ADDR	IP-адрес МК
IP_DEST_ADDR	IP-адрес устройства назначения (если известен)
IP_SMTP_ADDR	IP-адрес SMTP-сервера

Таблица 3. Настройки модема по умолчанию

Имя константы	Описание
MODEM_COUNTRY_CODE	Строка инициализации модема, устанавливающая код страны. Обычно содержит AT-команды и должна заканчиваться символом возврата каретки (‘\r’).
MODEM_PROTOCOL	Строка инициализации модема, устанавливающая протокол. Обычно содержит AT-команды и должна заканчиваться символом возврата каретки (‘\r’).
MODEM_INIT_DIAL	Строка инициализации модема, используемая при осуществлении исходящего вызова. Обычно содержит AT-команды и должна заканчиваться символом возврата каретки (‘\r’).
MODEM_INIT_ANSWER	Строка инициализации модема, используемая при настройке модема для приема вызовов. Обычно содержит AT-команды и должна заканчиваться символом возврата каретки (‘\r’).
MODEM_DIAL	Строка инициализации модема, содержащая телефонный номер для исходящего вызова. Обычно содержит AT-команды и должна заканчиваться символом возврата каретки (‘\r’).
LOGIN_NAME	Регистрационное имя (имя пользователя), используемое при подключении к удаленному модему, или регистрационное имя, проверяемое при подключении удаленного модема к локальному модему. Эта константа используется только в том случае, если протокол аутентификации пароля PAP отключен.
PASSWORD	Пароль, используемый при подключении к удаленному модему, или пароль, проверяемый при подключении удаленного модема к локальному модему. Эта константа используется только в том случае, если протокол аутентификации пароля PAP отключен.
DIAL_LOGIN_PROMPT	Приглашение на ввод имени пользователя, ожидаемое от удаленного модема при подключении. Эта константа используется только в том случае, если протокол аутентификации пароля PAP отключен.
DIAL_PASSWORD_PROMPT	Приглашение на ввод пароля, ожидаемое от удаленного модема при подключении. Эта константа используется только в том случае, если протокол аутентификации пароля PAP отключен.
ANS_LOGIN_PROMPT	Приглашение на ввод имени пользователя, предлагаемое удаленному модему при ответе на вызов. Эта константа используется только в том случае, если протокол аутентификации пароля PAP отключен.
ANS_PASSWORD_PROMPT	Приглашение на ввод пароля, предлагаемое удаленному модему при ответе на вызов. Эта константа используется только в том случае, если протокол аутентификации пароля PAP отключен.

Таблица 4. Настройки стека TCP/IP

Имя константы	Описание
num_sockets	Устанавливает максимальное количество используемых сокетов. Это значение должно быть от 1 до 127. Каждый сокет использует приблизительно 46 байт памяти XRAM.
xmit_buff_size	Устанавливает размер буфера, используемого для передачи.
recv_buff_size	Устанавливает размер буфера, используемого для приема.
socket_wait_ticks	Количество системных отсчетов длительностью 10 мс, определяющее время ожидания пакета.
ip_time_to_live	Устанавливает значение поля «время жизни» в IP-пакете.
multicast_ttl	Устанавливает значение поля «время жизни» в IP-пакете для многоадресных (широковещательных) пакетов.
tl0_flash th0_flash	Значения перезагрузки Таймера 0, задаваемые таким образом, чтобы Таймер 0 переполнялся через 10 мс. Эти значения определяют длительность системного отсчета времени.
ppp_resend_ticks	Количество системных отсчетов, определяющее время ожидания перед повторной передачей PPP-пакета.
ppp_resend_trys	Число, определяющее, сколько раз передавать PPP-пакет перед разрывом соединения.
ppp_terminate_trys	Число, определяющее, сколько раз передавать запрос на разрыв PPP-соединения перед перенастройкой соединения.
pap_num_users	Количество записей в таблице PAP.
use_password	Если use_password = 1, то аутентификация пользователя осуществляется на уровне модема. Если для аутентификации используется PAP, то следует установить use_password = 0.
uart_reload	Значение перезагрузки для UART. Максимальная стандартная скорость передачи данных UART выбирается автоматически мастером конфигурирования TCP/IP.
arp_keep_ticks	Количество системных отсчетов, определяющее, как долго хранить записи в ARP-кэше.
arp_resend_trys	Число, определяющее, сколько раз повторно передается ARP-пакет.
arp_wait_ticks	Количество системных отсчетов, определяющее время ожидания ARP-пакета.
arp_cache_size	Количество записей в ARP-кэше.
arp_auto_update	Если arp_auto_update = 1, то ARP-кэш обновляется после приема каждого корректного пакета. По PING-запросам ARP-кэш обновляется всегда.
ftp_max_param	Размер буфера для хранения принятых параметров командной строки. Это значение не должно быть меньше 23.
ftp_buffer_len	Размер буфера приема FTP. Этот буфер должен быть достаточно большим, чтобы принять любой из ожидаемых файлов.
ftp_num_users	Количество сохраняемых комбинаций имени пользователя и пароля. Если ftp_num_users = 0, то аутентификация не будет осуществляться.
mem_pool_size	Пул памяти ОЗУ, доступный для функции *malloc()*.
http_buffer_len	Буфер, используемый для обработки HTTP-включений. Размер этого буфера должен быть равен размеру TCP-сегмента.
tftp_resend_trys	Число, определяющее, сколько раз передается TFTP-пакет перед разрывом соединения.
ping_buff_size	Если PING разрешен, то это значение определяет размер данных из PING-запроса, которые могут быть сохранены. К этому значению также добавляются 9 байт, чтобы сохранить часть заголовка PING-запроса. Если PING-запрос содержит данных больше, чем определено этим значением, то пакет будет отброшен и ответ не будет отправлен. Значение по умолчанию равно 32.

Таблица 4. Настройки стека TCP/IP (продолжение)

Имя константы	Описание
tcp_window	Это значение представляет собой размер TCP-сегмента, т.е. как количество принимаемых от удаленного соединения данных, так и количество данных, отправляемых в одном пакете. Это значение должно быть больше нуля и не должно превышать 1460. Чем больше это значение, тем выше производительность, однако при этом требуются буферы большего размера. **Примечание:** При приеме стек TCP/IP использует сегменты фиксированного размера, а не сегменты переменной длины, как указано в RFC 793. Если используется PPP, то для поддержки escape-символов значения *RECV_BUFF_SIZE* и *XMIT_BUFF_SIZE* должны как минимум вдвое превышать размер TCP-сегмента. В сетях Ethernet значения *RECV_BUFF_SIZE* и *XMIT_BUFF_SIZE* должны быть не менее чем TCP_WINDOW+58.
tcp_resend_ticks	Количество системных отсчетов, определяющее время ожидания перед повторной передачей TCP-пакета.
tcp_resend_trys	Число, определяющее, сколько раз передается TCP-пакет перед разрывом соединения.
smtp_buffer_len	Это значение представляет собой размер временного буфера для SMTP-команд. Это значение должно быть не менее 46. Рекомендуемым значением является TCP_WINDOW.
num_vf_pages	Количество элементов таблицы каталогов в виртуальной файловой системе. Может быть от 1 до 255.
num_post_funcs	Количество элементов таблицы POST-функций в виртуальной файловой системе. Может быть от 1 до 255.

Приложение B – Структуры данных стека TCP/IP

Стек TCP/IP определяет следующие структуры данных:

Структура: SOCKET_INFO_T

```
typedef struct socket_info_s {
      word16 src_port;
      word16 dest_port;
      byte ip_dest_addr[IP_ADDR_LEN];
      byte *send_ptr;
      word16 send_len;
      byte *recv_ptr;
      byte *recv_end;
      word16 recv_len;
      byte ip_proto;
      byte socket_no;
      byte socket_type;
      byte socket_state;
#if TCP
      byte tcp_state;
      byte tcp_resends;
      byte tcp_flag;
      byte recv_tcp_flag;
      byte data_offset;
      word16 tcp_unacked_bytes;
      word16 recv_tcp_window;
      SEQNUM_U RCV_NXT;
      SEQNUM_U SEG_SEQ;
      SEQNUM_U SEG_ACK;
      SEQNUM_U SND_UNA;
      TIMER_INFO_T tcp_timer;
#endif
} SOCKET_INFO_T;
```

Структура: VF

```
typedef struct vf {
      byte filename[VF_NAME_LEN];
      word16 page_size;
      PCONST_BYTE page_ptr;
      byte * ram_page_ptr;
      byte page_type;
      byte in_use_flag;
} VF;
```

Структура: POST_FUNCS

```
typedef struct post_funcs {
      byte func_name[FUNC_NAME_LEN];
      POST_FP func_ptr;
} POST_FUNCS;
```

Структура: SMTP_INFO_T

```
typedef struct smtp_info_s {
      byte *from;
      byte *to;
      byte *subject;
      byte *message;
      byte *attachment;
      byte *filename;
} SMTP_INFO_T;
```

Приложение C – Настройки компилятора, определяющие модель памяти

Фирменная API-библиотека разработана с использованием модели памяти LARGE. Использование этой библиотеки в проекте, в котором по умолчанию задана модель памяти SMALL или COMPACT, может привести к появлению предупреждений (в зависимости от установленного уровня предупреждений). Чтобы предотвратить это, следует установить по умолчанию модель памяти LARGE, и заменить этот параметр путем определения каждой функции с ключевым словом компилятора “small”.

Приложение D – Подключение встроенного модема к PC

Стек TCP/IP позволяет настраивать встроенный модем как клиент или как сервер. Встроенный модем может взаимодействовать с любым другим модемом через стандартную телефонную линию или с телефонным симулятором. Любой PC с ОС Windows 2000 или Windows XP, который имеет модем, можно настроить таким образом, чтобы принимать вызовы или звонить на встроенный модем.

Настройка PC для приема вызовов (режим сервера)

1. Откройте папку **Сетевые подключения** в **Панели управления**.
2. Дважды щелкните значок **Создание нового подключения**. Должен появиться мастер создания нового подключения.
3. Если используется ОС Windows XP, то выберите переключатель **Установить прямое подключение к другому компьютеру** и нажмите кнопку **Далее**.
4. Выберите переключатель **Принимать входящие подключения** и нажмите кнопку **Далее**.
5. Установите галочку напротив имени модема.
6. Выберите **Запретить виртуальные частные подключения.**
7. Установите галочки напротив всех пользователей, которым будет разрешено использовать модем.
8. Выберите **Протокол Интернета (TCP/IP)** и нажмите кнопку **Свойства**. В этом диалоговом окне можно настроить способ назначения IP-адресов и разрешить или ограничить доступ к локальной сети (LAN). Если встроенному модему разрешен доступ к LAN, то встроенная система может отправлять e-mail, используя почтовый SMTP-сервер в корпоративной сети.
9. Нажмите кнопку **Готово** для завершения подключения.

Настройка PC для удаленного доступа к встроенному модему (режим клиента)

1. Откройте папку **Сетевые подключения** в **Панели управления**.
2. Дважды щелкните значок **Создание нового подключения**. Должен появиться мастер создания нового подключения.
3. Выберите переключатель **Подключение к Интернету** и нажмите кнопку **Далее**.
4. Выберите переключатель **Установить подключение вручную** и нажмите кнопку **Далее**.
5. Выберите переключатель **Через обычный модем** и нажмите кнопку **Далее**.
6. Введите имя для создаваемого подключения и нажмите кнопку **Далее**.
7. Введите телефонный номер встроенного модема.
8. Выберите пользователей, которым разрешено устанавливать удаленное соединение.
9. Введите имя пользователя и пароль, используемые для удаленного подключения к встроенному модему.
10. Нажмите кнопку **Готово** для завершения подключения.

Информация для контактов

Silicon Laboratories Inc.
4635 Boston Lane
Austin, TX 78735
Tel: 1+(512) 416-8500
Fax: 1+(512) 416-9669
Toll Free: 1+(877) 444-3032
Email: MCUinfo@silabs.com
Internet: www.silabs.com