**basic\_lwipbuf**

**basic\_lwipbuf** – это класс-буфер для передачи сообщений через Ethernet, использующий в качестве основной сетевой библиотеки lwip. Класс наследуется от шаблонного класса **basic\_streambuf**, что означает, что lwipbuf может работать с потоками вывода.

Сообщения, передаваемые с помощью данного класса, перекодируются в **UTF-8**. Написаны соответствующие функции, которые могут переводить **UTF-16 (UTF-32)** и **CP1251** в UTF-8. Перекодировка происходит автоматически в зависимости от типа данных, которые используются для буферизации. Так, для типа данных **char** данный класс будет переводить из CP1251; для **wchar\_t** введена проверка на размерность: 2 байта – перевод из UTF-16, 4 байта – из UTF-32. Другие кодировки и типы данных не рассматривались, и перегрузка соответствующих функций для них реализована.

Работа с русскими символами посредству использования типа данных **wchat\_t** возможна только на версии **IAR > 7**, так как IAR меньшей версии некорректно считывает русские символы в буфер. Для того, чтобы в IAR > 7 была возможна передача русских символов необходимо пройти по пути «Свойства проекта»/«C/C++ Compiler»/«Encodings» и выставить в «Default source file encoding», «Default input file encoding» и «Text output file encoding» значение «System locale».

Для работы со **статическим IP адресом** необходимо передать в поле **ap\_ip** конструктора ваш статический адрес, а также определить **#define LWIP\_DHCP 0**. Если вы хотите использовать **DHCP**, то достаточно указать **#define LWIP\_DHCP 1** (если в этом случае вы передадите статический адрес в поле ap\_ip конструктора, то он будет проигнорирован). В случае если флаг **USE\_LCD** определен как **true**, то после успешной инициализации сети на дисплее отобразится IP адрес, на который ссылается LWIP. Порт же задается программистом в конструкторе класса и также отображается на дисплее.

Работа данного класса проверена в **PuTTY** для типов подключения **Raw** и **Telnet**.

Ниже представлено описание методов класса, их параметры и логика работы.

Публичные методы:

1. basic\_lwipbuf(size\_t a\_sizebuf = IRSLIB\_LWIPBUF\_SIZE, u16\_t a\_port = IRSLIB\_LOG\_PORT, ip\_addr\_t\* ap\_ip = nullptr) – конструктор по умолчанию. На вход принимает три аргумента: a\_sizebuf – размерность буфера (по умолчанию установленное значение IRSLIB\_LWIPBUF\_SIZE = 128), a\_port – порт, по которому производится отправка сообщений (по умолчанию установленное значение IRSLIB\_LOG\_PORT = 5008), ap\_ip – IP адрес сервера (по умолчанию – nullptr).

В конструкторе производится выделение памяти как для главного буфера, так и для вспомогательных, которые используются при перекодировании сообщений. Помимо выделения памяти, происходит инициализация TCP/IP (посредству вызова приватной функции tcp\_init()). В случае неудачной инициализации соответствующий ассерт выдаст сообщение и ошибку.

1. ~basic\_lwipbuf() – деструктор, который высвобождает память, выделенную под *вспомогательные* буферы. Также, если существуют активные пользователи (сессии), то они закрываются.
2. void tick (struct netif\* ap\_netif) – функция «Тик», которую необходимо вызывать в каждой итерации цикла. На вход принимает один аргумент – указатель на структуру, которая используется для всех LWIP интерфейсов.

Данная функция обрабатывает пакеты, которые поступают от клиентов, ни низком уровне. Соответственно, если данная функция не будет вызываться, то обработку подключения/отключения клиентов к серверу не будет обрабатываться, отправка сообщений также не будет осуществляться.

1. void send(const void\* ap\_msg, size\_t a\_size\_msg) – функции отправки сообщения клиентам. На вход принимает два аргумента: ap\_msg – указатель на сообщение, a\_size\_msg – размер сообщения.

Функция осуществляет цикличную отправку сообщения всем активным клиентам сервера. Сообщения клиенты получают по логике «Первый подключился – Первый получил». Если при отправке сообщения клиенту возникла ошибка, не связанная с переполнением внутреннего буфера lwip, то сервер закрывает соединение с данным клиентом.

Стоит отметить, что здесь не происходит перекодировки сообщения, вам придется самостоятельно предварительно осуществить перекодировку сообщения.

1. u16\_t get\_port() – функция, возвращающая порт, по которому передаются сообщения.
2. ip\_addr\_t get\_ip() – функция, возвращающая IP адрес сервера. Если адрес задается с помощью DHCP, то функция вернет nullptr.
3. bool is\_any\_connected() – функция, возвращающая логический тип данных есть ли активные клиенты.
4. size\_t get\_count\_connected() – функция, возвращающая количество активных пользователей.
5. int\_type overflow(int\_type c = traits\_type::eof()) – функция, которая автоматически вызывается при переполнении главного буфера. Функция принимает на вход единственный аргумент – символ, который не влез в буфер (по умолчанию равняется traits\_type::eof(), который, что запись данных в буфер окончена).

Всегда возвращает 0.

В данной функции осуществляется очистка главного буфера посредству отправки сообщения (с помощью функции send(ap\_msg, a\_size\_msg)), хранящимся в буфере, и последующей очисткой. После того символ, который не влез в буфер добавляет в его начало, затем происходит запись других оставшихся данных в буфер.

Также здесь осуществляет перекодировка считанных данных в нужную кодировку (функции, которую осуществляют перекодировки, определены в **irsencode.h**).

Важно подметить, реализация данной функции есть только для двух типов: **char** и **wchar\_t** – для других типов данных вызовется шаблонная реализация, которая вызывает исключение (если исключения разрешены IAR’ом).

1. int sync() – функция «синхронизации данных». На самом деле данная функция просто вызывает функцию overflow() преждевременно. Всегда возвращает 0.

Приватные методы:

1. int tcp\_init() – приватная функция инициализации TCP/IP. В случае некорректной инициализации возвращает -1, в успешном – 1.
2. size\_t copy\_str\_to\_buffer\_with\_correct\_endls(const char\* ap\_str) – приватная функция обработки передаваемого сообщения. На вход принимает ссылку на обрабатываемую строку. Возвращает размерность переработанной строки.

Данная функция осуществляет копирование переданной строки в главный буфер с учетом добавления перед каждым символом ‘\n’ символ ‘\r’. Эта манипуляция позволяет скорректировать отображение перехода на новую строку у пользователя.

1. err\_t c\_tcp\_accept(void\* arg, struct tcp\_pcb\* newpcb, err\_t err) – приватная callback функция для lwip, которая вызывается во время подключения нового клиента к серверу. Принимает на вход три параметра:
   1. arg – указатель на данные, которые определяются пользователем во время инициализации TCP/IP и передаются во все callback-lwip функции. В данном случае в качестве этих данных используется указатель на массив всех активных пользователей (сессий). Во время вызова данной функции в массив добавляется дескриптор данной сессии для того, чтобы в будущем можно было отправить сообщение данному клиенту.
   2. newpcb – дескриптор на новую сессию.
   3. err – не используется.

Возвращает всегда ERR\_OK.

1. err\_t c\_tcp\_recv(void\* arg, struct tcp\_pcb\* tpcb, struct pbuf\* p, err\_t err) – приватная callback функция, которая вызывается после получения данных от пользователя. Принимает на вход четыре параметра:
   1. arg – указатель на данные, которые определяются пользователем. Как было сказано ранее, это указатель на массив всех активных пользователей. В данной функции, если пакеты не были получены от пользователя, то разрывается соединение с ним.
   2. tpcb – дескриптор соединения.
   3. p – указатель на полученные пакеты от пользователя (могут равняться нулю).
   4. err – не используется.

Всегда возвращает ERR\_OK.

1. err\_t c\_tcp\_poll(void\* arg, struct tcp\_pcb\* tpcb) – приватная callback функция, которая вызывается с заданным интервалом. Данная функция пока не используется.
2. void c\_tcp\_connection\_close(struct tcp\_pcb\* tpcb) – приватная callback функция, которая вызывается для закрытия соединения с пользователем. На вход принимает один аргумент – дескриптор соединения.

Также для данного класса определены соответствующие typedefs. Ниже перечислен их перечень:

1. typedef basic\_lwipbuf<char, char\_traits<char>> lwipbuf;
2. typedef basic\_lwipbuf<wchar\_t, char\_traits<wchar\_t>> wlwipbuf;