Network Working Group
Request for Comments: 1662
STD: 51
Obsoletes: 1549
Category: Standards Track

W. Simpson, Editor
Daydreamer
July 1994

PPP in HDLC-like Framing

PPP с кадрированием в стиле HDLC

Статус документа

Этот документ содержит проект стандартного протокола Internet для сообщества Internet и служит запросом для комментариев и предложений с целью совершенствования протокола. Текущее состояние стандартизации и статус протокола можно узнать из документа «Internet Official Protocol Standards» (STD 1). Документ может распространяться без ограничений.

Тезисы

Протокол PPP[1] [1] обеспечивает стандартный способ передачи дейтаграмм разных протоколов по каналам «точка-точка».

Этот документ описывает кадрирование в стиле HDLC для пакетов, инкапсулированных в PPP.

Оглавление

1. Введение

Данная спецификация описывает кадрирование для битовых и октетных синхронных каналов, а также для асинхронных каналов с 8 битами данных без бита четности. Эти каналы **должны** быть полнодуплексными и **могут** быть выделенными или коммутируемыми.

[1]Point-to-Point Protocol - протокол соединений «точка-точка».

Задан механизм экранирования (escape), который обеспечивает передачу через канал сигналов XON/XOFF и удаление ложных сигналов управления, которые могут помещаться в канал программами и оборудованием.

Некоторые протоколы ожидают безошибочной передачи, а другие могут обеспечивать лишь условное детектирование ошибок или совсем не поддерживать его. Протокол PPP использует контрольные суммы HDLC FCS[1] для обнаружения ошибок. Этот механизм обычно доступен в аппаратных реализациях, предложена также его программная версия.

1.1. Уровни требований

В этом документе некоторые слова служат для задания уровня требований спецификации. Такие слова часто выделяются **шрифтом**.

MUST - должно

Это слово, а также термин **требуется** (REQUIRED) используется для требований, которые являются абсолютно необходимыми в данной спецификации.

MUST NOT - недопустимо

Эта фраза означает абсолютный запрет в рамках спецификации.

SHOULD - следует

Это слово, а также глагол **рекомендуется** (RECOMMENDED) используется для обозначения требований, от выполнения которых можно отказаться при наличии разумных причин. Однако при таком отказе следует помнить о возможных проблемах в результате отказа и принимать взвешенное решение.

MAY - можно, возможно

Это слово, а также прилагательное **необязательный** (OPTIONAL) обозначают элементы, реализация которых является необязательной. Реализация, не включающая ту или иную опцию, **должна** быть готова к работе с реализациями, которые используют эту опцию (возможно совместная работа будет обеспечиваться за счет некоторого ущерба функциональности).

1.2. Термины

Ниже приведены определения терминов, часто используемых в этом документе.

Datagram - дейтаграмма

Единица передачи на сетевом уровне (например, IP). Дейтаграмма может инкапсулироваться в один или несколько пакетов, передаваемых канальному уровню.

Frame - кадр

Единица передачи на уровне логического канала данных. Кадр может включать заголовок и/или трейлер вместе с некоторым числом блоков данных.

Packet - пакет

Базовый блок инкапсуляции, который передается через интерфейс между сетевым и канальным уровнем. Пакет обычно отображается в кадр, за исключением случаев фрагментации на канальном уровне или встраивания в один кадр множества пакетов.

Peer - партнер

Другая сторона соединения «точка-точка».

Silently discard – отбрасывание без уведомления

Реализация отбрасывает пакет без дальнейшей обработки. Реализации **следует** поддерживать возможность записи в системный журнал ошибок, включая содержимое отброшенных без уведомления кадров, а также **следует** учитывать такие события в счетчике статистики.

2. Требования физического уровня

PPP может работать через большинство интерфейсов DTE/DCE (таких, как EIA RS-232-E, EIA RS-422, CCITT V.35). Единственным абсолютным требованием PPP является поддержка полнодуплексного режима (выделенное или коммутируемое соединение), который может обеспечиваться с помощью асинхронных (старт-стоп), синхронных битовых или октетных потоках, прозрачных для кадров канального уровня PPP.

Формат интерфейса

PPP предоставляет октетный интерфейс с физическим уровнем. Субоктетная передача не поддерживается и не принимается.

Скорость передачи

PPP не вносит ограничений на скорость передачи в дополнение к ограничениям конкретного интерфейса DTE/DCE.

Сигналы управления

PPP не требует использования сигналов управления, таких как запрос передачи (RTS[2]), готовность к передаче (CTS[3]), детектирование сигнала несущей (DCD[4]), готовность терминала (DTR[5]).

Поддержка таких сигналов обеспечивает расширение функциональности и рост производительности. В частности, такие сигналы **следует** применять для указания событий Up и Down в при автоматическом согласовании опций LCP [1]. если такие сигналы не поддерживаются, реализация **должна** подавать сигнал Up для LCP при инициализации, а в случае событий Down **не следует** подавать сигнала.

Поскольку сигнализация не требуется, физический уровень **может** быть отвязан от канального уровня, не показывая детали физического транспорта. Это позволяет реализовать мобильность пользователей в собтовых сетях и на других каналах с быстрой коммутацией.

При переходе из ячейки в ячейку в рамках одной зоны реализация **может** считать всю зону одним каналом, хотя передача может происходить с переключением частоты. Канал рассматривается как соединение с центральным узлом зоны, а не с приемопередатчиками отдельных ячеек. Однако канал **следует** создавать заново при переключении на другой административный домен (зону).

[1]Frame Check Sequence - последовательность проверки кадра.
[2]Request To Send.
[3]Clear To Send.
[4]Data Carrier Detect.
[5]Data Terminal Ready.

Из-за непостоянства трафика данных некоторые реализации отключают физический уровень в периоды бездействия и заново организуют соединение при восстановлении трафика, не информируя об этом канальный уровень. Надежным реализациям следует избегать таких трюков, поскольку ускоренное восстановление соединений происходит за счет снижения уровня безопасности. Реализациям **следует** передавать сигнал Down всякий раз по истечение «значимого» времени после разрыва соединения. Продолжительность «значимого» времени является предметом дискуссий и связана с тарифами, временем организации соединения и вопросами безопасности.

3. Уровень канала данных

PPP использует принципы, описанные в структуре кадров ISO 3309-1979 HDLC четвертого издания стандарта 3309:1991 [2], которая была изменена для поддержки HDLC в асинхронных средах.

Процедуры управления PPP используют поле Control, описанное в элементах процедур ISO 4335-1979 HDLC, четвертого издания стандарта 4335:1991 [4].

Не следует считать, что каждая из упомянутых рекомендация включена в PPP. Все включения явно описаны в последующих параграфах.

Для соответствия принятой в Internet практике и предотвращения путаницы среди людей, привыкших читать RFC, все двоичные значения в последующих описаниях приводятся в порядке от старшего бита к младшему, слева направо, если явно не указано иное. Отметим, что стандарты ISO и практика CCITT указывают биты в порядке передачи (сетевой порядок). Следует помнить об этом при работе с международными стандартами.

3.1. Формат кадра

Структура кадра PPP в стиле HDLC показана ниже. На рисунке не указаны биты, служащие для синхронизации (такие, как start и stop для асинхронных каналов), а также биты и октеты, обеспечивающие прозрачность. Поля передаются слева направо.

```
+----------+----------+----------+
|   Flag   | Address  | Control  |
| 01111110 | 11111111 | 00000011 |
+----------+----------+----------+
+----------+-------------+---------+
| Protocol | Information | Padding |
| 8/16 bits|      *      |    *    |
+----------+-------------+---------+
+----------+----------+-----------------
|   FCS    |   Flag   | Inter-frame Fill
|16/32 bits| 01111110 | or next Address
+----------+----------+-----------------
```

Поля Protocol, Information и Padding описаны в разделе «Инкапсуляция PPP» документа [1].

Flag

Каждый кадр начинается и заканчивается последовательностью флагов Flag Sequence в виде 01111110 (0x7e). Все реализации постоянно проверяют эти флаги, которые служат для синхронизации кадров.

Между двумя кадрами требуется лишь одна последовательность Flag Sequence. Две таких последовательности подряд указывают пустой кадр, который отбрасывается без уведомления и не считается ошибкой FCS.

Address

Однооктетное поле Address содержит последовательность битов 11111111 (0xff) - адрес All-Stations. Индивидуальные адреса не назначаются. Адрес All-Stations **должен** поддерживаться и приниматься.

Применение других размеров и значений адресов может определено позднее или по предварительному согласованию. Кадры с нераспознанными адресами **следует** отбрасывать без уведомления.

Control

Однооктетное поле Control содержит последовательность битов 00000011 (0x03) - команда (UI[1]) со сброшенным битом P/F[2].

Использование других значений в поле Control может определено позднее или по предварительному согласованию. Кадры с нераспознанным значением поля Control **следует** отбрасывать без уведомления.

FCS

Поле FCS[3] по умолчанию имеет размер 16 битов (2 октета). FCS передается с младшего октета, который содержит коэффициент старшего порядка.

Определены также 32-битовые (4 октета) поля FCS. Они могут быть согласованы, как описано в документе «PPP LCP Extensions» [5].

Использование других размеров FCS может определено позднее или по предварительному согласованию.

Значение FCS рассчитывается для всех битов полей Address, Control, Protocol, Information и Padding без учета битов start и stop (асинхронная передача), а также всех битов или октетов, добавляемых при синхронной передаче для прозрачности. Не включаются в расчет также последовательности флагов (Flag Sequences) и само поле FCS.

При получении октетов, помеченных в Async-Control-Character-Map, они отбрасываются до расчета FCS.

Дополнительная информация о расчете FCS приведена в приложениях к этому документу.

Завершение полей Information и Padding определяется положением закрывающей последовательности флагов и удалением поля FCS.

3.2. Изменение базового формата

Протокол LCP может согласовать изменение стандартной процедуры кадрирования в стиле HDLC. Однако измененные кадры всегда будут четко отличаться от стандартных.

[1]Unnumbered Information.
[2]Poll/Final.
[3]Frame Check Sequence - последовательность проверки кадра.

Address-and-Control-Field-Compression

При использовании стандартного кадрирования в стиле HDLC поля Address и Control имеют шестнадцатеричные значения 0xff и 0x03, соответственно. Если используются другие значения Address или Control, согласование сжатия Address-and-Control-Field-Compression **недопустимо**.

На передающей стороне сжатые поля Address и Control просто опускаются.

На приеме поля Address и Control восстанавливаются путем проверки двух первых октетов. Если они содержат значения 0xff и 0x03, предполагается, что это поля Address и Control. В остальных случаях предполагается, что эти поля были сжаты и не передавались.

По определению первый октет двухоктетного поля Protocol не может иметь значения 0xff (должен быть четным). Значение 0x00ff в поле Protocol не разрешено (резерв) для предотвращения неоднозначности при включенном сжатии Protocol-Field-Compression и первом октете поля Information со значением 0x03.

4. Октетное кадрирование

В этом разделе описано кадрирование в стиле HDLC для 8-битовых асинхронных и октетных синхронных каналов.

4.1. Последовательность флагов

Flag Sequence указывает начало и конец кадра. Поток октетов просматривается пооктетно в поисках последовательности битов 01111110 (0x7e).

4.2. Прозрачность

Октет Control Escape определяется как последовательность битов 01111101 (0x7d), старший бит передается первым.

Передающая реализация **должна выделять** по меньшей мере октеты Flag Sequence и Control Escape.

После расчета FCS передатчик проверяет весь кадр между двумя Flag Sequence. Каждый октет Flag Sequence, Control Escape и любой другой октет, помеченный в отображении ACCM[1] передающей стороны, заменяется последовательностью из октета Control Escape и исходного октета, к которому применена операция XOR с 0x20.

Это дополнение бита 5, где битовые позиции нумеруются 76543210 (шестой бит в ISO с нумерацией 87654321 — обращайте на это внимание при сравнении документов).

Принимающие реализации **должны** корректно обрабатывать последовательности Control Escape.

На приемной стороне до расчета FCS проверяется каждый октет со значением меньше 0x20. Если октет помечен в ACCM принимающей стороны, он просто удаляется (он мог быть добавлен промежуточным коммуникационным оборудованием). Каждый октет Control Escape также удаляется, а к следующему за ним октету применяется операция XOE с 0x20, если это не Flag Sequence (завершение кадра).

Ниже приведено несколько примеров. Данные передаются в канал в указанном порядке.

```
0x7e передается как 0x7d, 0x5e.     (Flag Sequence)
0x7d передается как 0x7d, 0x5d.     (Control Escape)
0x03 передается как 0x7d, 0x23.     (ETX)
```

Некоторые модемы с программным управлением потоком данных могут перехватывать исходящие DC1 и DC3, игнорируя восьмой бит (четность). Такие данные будут передаваться как показано ниже .

```
0x11 передается как 0x7d, 0x31.     (XON)
0x13 передается как 0x7d, 0x33.     (XOFF)
0x91 передается как 0x7d, 0xb1.     (XON с установленной четностью)
0x93 передается как 0x7d, 0xb3.     (XOFF с установленной четностью)
```

4.3. Недействительные кадры

Слишком короткие кадры (меньше 4 октетов при использовании 16-битовых FCS), кадры, где после Control Escape сразу следует закрывающая последовательность Flag Sequence и кадры с нарушенным октетным кадрированием (передача стоп-бита 0 вместо 1) отбрасываются без уведомления и не учитываются как ошибка FCS.

4.4. Заполнение временных интервалов

4.4.1. Синхронный поток октетов

Заполнения между октетами не предусмотрено.

Для заполнения времени между кадрами **должна** передаваться последовательность Flag Sequence.

4.4.2. Асинхронный поток

Межоктетное заполнение **должно** выполняться путем передачи непрерывной последовательности битов 1 (состояние mark-hold).

Межкадровое заполнение можно рассматривать как расширение межоктетного. Это позволяет сэкономить 1 октет на каждый кадр, снижая задержку и повышая эффективность использования полосы. Такое решение возможно, благодаря тому, что Flag Sequence может указывать начало и конец кадра. После приема любого кадра приемник всегда будет находиться в состоянии начала кадра.

Отказоустойчивым передатчикам следует избегать чрезмерного использования этого трюка, поскольку за снижение задержки приходиться расплачиваться снижением надежности. На зашумленных каналах приемник может получить искаженные символы и счесть их частью входящего кадра. Если передатчик не будет отправлять новую открывающую последовательность Flag Sequence перед следующим кадром, к текущему кадру будут добавлены символы шума и кадр станет непригодным (с высокой вероятностью).

[1]Async-Control-Character-Map — отображение асинхронных символов управления.

Предполагается, что реализации будут достигать наилучших результатов, всегда передавая открывающую последовательность Flag Sequence, если новый кадр не передается сразу же за предыдущим. Передатчикам **следует** отправлять открывающую последовательность Flag Sequence всякий раз, когда после передачи предыдущей закрывающей последовательности Flag Sequence прошло «заметное время». Максимальный порог «заметности» будет не больше интервала между символами при медленном наборе с клавиатуры (около 1 секунды).

4.5. Вопросы передачи

4.5.1. Синхронная передача октетов

За определение различных вариантов кодирования и скремблирования отвечает используемое оборудование DTE/DCE и это выходит за рамки данной спецификации.

4.5.2. Асинхронная передача

Все октеты передаются с младшего бита с одним стартовым битом, 8 битами данных и одним стоп-битом. 7-битовые асинхронные каналы не предусматриваются.

5. Битовое кадрирование

В этом разделе описано использование кадрирования в стиле HDLC для битовых синхронных каналов.

5.1. Последовательность флагов

Flag Sequence указывает начало и конец кадра, а также служит для синхронизации. Поток просматривается побитово в поиске последовательностей 01111110 (0x7e).

Последовательность флагов «shared zero mode[1]» (011111101111110) применять **не следует**. Когда этого не удается избежать, реализация **должна** обеспечивать незамедлительную передачу на канальный уровень первой найденной последовательности Flag Sequence (конец кадра). Использование режима «общего нуля» препятствует взаимодействию с преобразователями синхронных битовых потоков в асинхронные и синхронные октетные потоки.

5.2. Прозрачность

После расчета FCS передатчик проверяет весь кадр между двумя последовательностями Flag Sequence. После каждой последовательности из 5 битов со значением 1 (включая последние 5 битов FCS) вставляется бит 0 для того, чтобы не возникало имитации Flag Sequence.

На приемной стороне перед расчетом FCS биты 0 после пяти последовательных битов 1 отбрасываются.

5.3. Недействительные кадры

Слишком короткие кадры (меньше 4 октетов при использовании 16-битовых FCS) и кадры, завершающиеся последовательностью из 6 и более битов 1, отбрасываются без уведомления и не учитываются как ошибка FCS.

5.4. Заполнение временных интервалов

Заполнения между октетами не предусмотрено.

Для заполнения межкадровых интервалов **следует** передавать Flag Sequence. Однако для некоторых каналов с коммутацией устройств требуется использовать маркировку бездействия (mark idle) в форме последовательности единиц (в частности, это кана, где учет ведется на основе периодов битовой активности). При использовании маркеров бездействия на битовых синхронных каналах реализация **должна** обеспечить не менее 15 последовательных битов 1 между флагами в течение периода бездействия и генерацию Flag Sequence в начале кадра после периода бездействия.

Это отличается от ISO 3309, где можно использовать маркеры бездействия от 7 до 14 битов.

5.5. Кодирование при передаче

Все октеты передаются начиная с младшего бита.

За определение различных вариантов кодирования и скремблирования отвечает используемое оборудование DTE/DCE и это выходит за рамки данной спецификации.

Хотя PPP может работать, не принимая во внимание базовое представление битовых потоков, отсутствие стандартов передачи и стандартов канального уровня будет препятствовать функциональной совместимости. При скоростях от 56 Кбит/с до 2 Мбит/с сейчас наиболее широко распространено кодирование NRZ, которое рекомендуется использовать по умолчанию.

Если возможна настройка кодирования, рекомендуется использовать NRZI, поскольку этот вариант сравнительно устойчив к ошибкам инверсии сигнала и его применение **может** обеспечить возможность организации соединений без дорогого оборудования DSU/CSU. К сожалению кодирование NRZI усугубляет отсутствие фактора x1 в 16-битовых FCS и существует вероятность пропустить 1 ошибку из 2^{15} (вместо одной из 2^{16}), а тройные ошибки не детектируются. Поэтому при использовании NRZI рекомендуется согласовывать 32-битовые FCS, где имеется фактор x1.

При более высоких скоростях вплоть до 45 Мбит/с некоторые реализаци используют ANSI HSSI[2]. Хотя такой опыт пока ограничен, разработчикам рекомендуется сотрудничество для выбора кодирования при передаче.

[1]Режим «общего нуля».
[2]High Speed Synchronous Interface — высокоскоростной синхронный интерфейс.

6. Преобразование асинхронных потоков в синхронные

Могут применяться преобразователи асинхронных потоков в синхронные (иногда их встраивают в модемы и сотовые интерфейсы), когда асинхронная реализации PPP на одной стороне канала взаимодействует с синхронной на другой стороне. В таких случаях за преобразование кадрирования в процессе работы отвечает конвертер.

Для поддержки такой функциональности синхронные реализации PPP **должны** всегда отвечать на опцию Async-Control-Character-Map пакетом LCP Configure-Ack. Однако восприятие конфигурационной опции не предполагает выполнение синхронной реализацией отображения ACCM. Это отображение будет выполняться преобразоователем.

7. Дополнительная конфигурационная опция LCP

Формат конфигурационных опций и основные опции уже определены для LCP [1].

Актуальные значения поля LCP Option Type указаны в свежей версии документа «Assigned Numbers» RFC [10]. Этот документ определяет одну опцию

```
2       Async-Control-Character-Map
```

7.1. Async-Control-Character-Map (ACCM)

Эта конфигурационная опция позволяет согласовать прозрачное использование символов управления на асинхронных каналах.

Каждая сторона синхронного канала поддерживает два отображения для асинхронных символов управления (ACCM). Принимаемое отображение ACCM имеет размер 32 бита, но передать можно до 256 битов ACCM. В результате возникает 4 различных ACCM, по 2 для каждого направления.

Для асинхронных каналов приемное отображение ACCM имеет знасение 0xffffffff. Для передаваемого ACCM по умолчанию используется 0xffffffff плюя символы Control Escape и Flag Sequence, а также другие помеченные (предшествующей настройкой) исходящие символы, которые нужно перехватывать.

Для других типов каналов по умолчанию применяется 0, поскольку отображения не требуется.

> Включение по умолчанию всех октетов меньше 0x20 позволяет использовать управляющие символы ASCII [6], за исключением DEL (Delete - удаление) прозрачно передавать через известное коммуникационное оборудорвание.

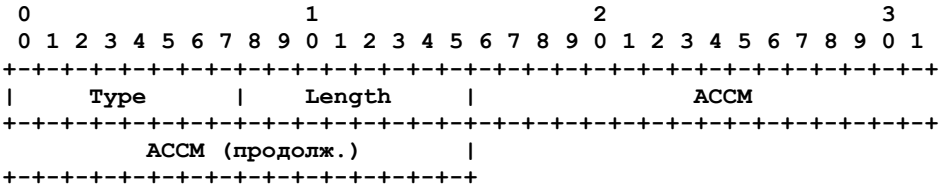
Передатчик **может** также отправлять октеты со значениями от 0x40 до 0xff (кроме 0x5e) в формате Control Escape. Поскольку эти значения октетов не согласуются, такой подход не решает проблемы приемников, которые не способны обрабатывать символы, не относящиеся к управлению. Кроме того, этот метод не решает проблему каналов, способных передавать лишь 7-битовые сиволы, поскольку он не влияет на восьмой бит.

> Следует отметить, что данная спецификация отличается в деталях от недавних дополнений, таких как 3309:1991/Amendment 2 [3]. Однако такая «расширенная прозрачность» применима лишь по предварительному соласованию. Использование методов обеспечения прозрачности в этом документе служит таким предварительным согласованием для PPP.
>
> Для совместимости с 3309:1991/Amendment 2 передатчик **может** экранировать (escape) DEL и ACCM=эквиваленты с установленным старшим битом (бит 8). Приемный алгоритм менять не требуется.
>
> В соответствии с согласованем ACCM передатчику **следует** прекратить экранирование символа DEL.

Однако отображение для всех символов управления требуется редко. Описываемая конфигурационная опция служит для информирования партнера о символах управления, для которых **должно** сохраняться отображение при передаче от партнера.

> Партнер **может** продолжать использовать отображение и для других символов, если этого требуют известные партнеру ограничения. Партнеру **следует** передать Configure-Nak с логическим объединением набора отображаемых символов, чтобы при появлении таких символов в произвольном месте приемная сторона могла игнорировать их.

Формат опции Async-Control-Character-Map показан ниже. Поля передаются слева направо.



Type

2

Length

6

ACCM

Четырехоктетное поле ACCM указывает набор отображаемых символов управления. Отображение передается начиная со старшего октета.

Каждый пронумерованный бит соответствует символу с тем же кодом. Если бит сброшен, отображение символа не требуется, а символы, соответствующие установленным битам, **должны** отображаться. Например, установка для бита 19 значения указывает, сто символ управления с кодом ASCII 19 (DC3, Control-S) **можно** передать напрямую.

> Примечание. Младший бит младшего октета (передается последним) имеет номер 0 и соответствует символу управления ASCII NUL.

A. Рекомендуемые опции LCP

Рекомендуемые опции конфигурации перечислены ниже.

Высокоскоростные каналы

```
Magic Number
Link Quality Monitoring
Без Address and Control Field Compression
Без Protocol Field Compression
```

Низкоскоростные каналы

```
Async Control Character Map
Magic Number
Address and Control Field Compression
Protocol Field Compression
```

B. Автоматическое распознавание кадров PPP

Иногда желательно детектировать кадры PPP (например, при входе в систему - login). Приведенные ниже последовательности показывают начало действительных кадров PPP LCP.

```
7e ff 03 c0 21
7e ff 7d 23 c0 21
7e 7d df 7d 23 c0 21
```

Отметим, что первые две строки не являются допустимыми именами пользователей Unix. Однако лишь третья строка дает кадр PPP с корректной контрольной суммой, когда 03 и ff берутся в качестве управляющих символов ETX и DEL без учета четности (это корректно для линий с контролем на четность) и отбрасываются.

Многие реализации решают эту задачу путем перевода интерфейса в пакетный режим, когда в процессе регистрации пользователя обнаруживается одна из приведенных выше последовательностей, без проверки исходной контрольной суммы PPP. Исходный входящий кадр PPP и незамедлительно передается Configure-Request.

C. Реализация FCS

Контрольные суммы FCS были разработаны для аппаратной реализации. Поток битов передается в линию (провод), FCS рассчитывается по мере поступления последовательных данных и полученное в результате значение FCS помещается в поток данных перед завершающей последовательностью Flag Sequence.

Получатель не может определить, где нужно заканчивать расчет FCS, пока не будет получена последовательность Flag Sequence. Поэтому при разработке FCS было предусмотрено появление определенной последовательности при выполнении операции FCS для добавленной в кадр FCS. Корректный кадр в этом случае указывается значением «good FCS» (верная контрольная сумма).

C.1. Генератор таблицы FCS

Приведенный ниже код создает таблицу поиска (просмотра) для расчета FCS-16.

```
/*
 * Генератор таблицы FCS-16.
 *
 * Drew D. Perkins из университета Carnegie Mellon.
 *
 * Код заимствован у Mohsen Banan и D. Hugh Redelmeier.
 */

/*
 * Полином для генерации FCS-16: x**0 + x**5 + x**12 + x**16.
 */
#define P       0x8408

main()
{
    register unsigned int b, v;
    register int i;

    printf("typedef unsigned short u16;\n");
    printf("static u16 fcstab[256] = {");
    for (b = 0; ; ) {
        if (b % 8 == 0)
            printf("\n");

        v = b;
        for (i = 8; i--; )
            v = v & 1 ? (v >> 1) ^ P : v >> 1;

        printf("\t0x%04x", v & 0xFFFF);
        if (++b == 256)
            break;
        printf(",");
    }
    printf("\n};\n");
}
```

C.2. Метод расчета 16-битовых FCS

Приведенный ниже код обеспечивает табличные операции (поиск) для расчета значений FCS при получении данных интерфейсом. Реализация основана на работах [7], [8], [9].

```
/*
 * u16 представляет 16-битовое целое число без знака. typedef зависит от
 * аппаратной архитектуры.
 */
typedef unsigned short u16;

/*
 * Таблица поиска FCS, созданная генератором таблиц.
 */
static u16 fcstab[256] = {
   0x0000, 0x1189, 0x2312, 0x329b, 0x4624, 0x57ad, 0x6536, 0x74bf,
   0x8c48, 0x9dc1, 0xaf5a, 0xbed3, 0xca6c, 0xdbe5, 0xe97e, 0xf8f7,
   0x1081, 0x0108, 0x3393, 0x221a, 0x56a5, 0x472c, 0x75b7, 0x643e,
   0x9cc9, 0x8d40, 0xbfdb, 0xae52, 0xdaed, 0xcb64, 0xf9ff, 0xe876,
   0x2102, 0x308b, 0x0210, 0x1399, 0x6726, 0x76af, 0x4434, 0x55bd,
   0xad4a, 0xbcc3, 0x8e58, 0x9fd1, 0xeb6e, 0xfae7, 0xc87c, 0xd9f5,
   0x3183, 0x200a, 0x1291, 0x0318, 0x77a7, 0x662e, 0x54b5, 0x453c,
   0xbdcb, 0xac42, 0x9ed9, 0x8f50, 0xfbef, 0xea66, 0xd8fd, 0xc974,
   0x4204, 0x538d, 0x6116, 0x709f, 0x0420, 0x15a9, 0x2732, 0x36bb,
   0xce4c, 0xdfc5, 0xed5e, 0xfcd7, 0x8868, 0x99e1, 0xab7a, 0xbaf3,
   0x5285, 0x430c, 0x7197, 0x601e, 0x14a1, 0x0528, 0x37b3, 0x263a,
   0xdecd, 0xcf44, 0xfddf, 0xec56, 0x98e9, 0x8960, 0xbbfb, 0xaa72,
   0x6306, 0x728f, 0x4014, 0x519d, 0x2522, 0x34ab, 0x0630, 0x17b9,
   0xef4e, 0xfec7, 0xcc5c, 0xddd5, 0xa96a, 0xb8e3, 0x8a78, 0x9bf1,
   0x7387, 0x620e, 0x5095, 0x411c, 0x35a3, 0x242a, 0x16b1, 0x0738,
   0xffcf, 0xee46, 0xdcdd, 0xcd54, 0xb9eb, 0xa862, 0x9af9, 0x8b70,
   0x8408, 0x9581, 0xa71a, 0xb693, 0xc22c, 0xd3a5, 0xe13e, 0xf0b7,
   0x0840, 0x19c9, 0x2b52, 0x3adb, 0x4e64, 0x5fed, 0x6d76, 0x7cff,
   0x9489, 0x8500, 0xb79b, 0xa612, 0xd2ad, 0xc324, 0xf1bf, 0xe036,
   0x18c1, 0x0948, 0x3bd3, 0x2a5a, 0x5ee5, 0x4f6c, 0x7df7, 0x6c7e,
   0xa50a, 0xb483, 0x8618, 0x9791, 0xe32e, 0xf2a7, 0xc03c, 0xd1b5,
   0x2942, 0x38cb, 0x0a50, 0x1bd9, 0x6f66, 0x7eef, 0x4c74, 0x5dfd,
   0xb58b, 0xa402, 0x9699, 0x8710, 0xf3af, 0xe226, 0xd0bd, 0xc134,
   0x39c3, 0x284a, 0x1ad1, 0x0b58, 0x7fe7, 0x6e6e, 0x5cf5, 0x4d7c,
   0xc60c, 0xd785, 0xe51e, 0xf497, 0x8028, 0x91a1, 0xa33a, 0xb2b3,
   0x4a44, 0x5bcd, 0x6956, 0x78df, 0x0c60, 0x1de9, 0x2f72, 0x3efb,
   0xd68d, 0xc704, 0xf59f, 0xe416, 0x90a9, 0x8120, 0xb3bb, 0xa232,
   0x5ac5, 0x4b4c, 0x79d7, 0x685e, 0x1ce1, 0x0d68, 0x3ff3, 0x2e7a,
   0xe70e, 0xf687, 0xc41c, 0xd595, 0xa12a, 0xb0a3, 0x8238, 0x93b1,
   0x6b46, 0x7acf, 0x4854, 0x59dd, 0x2d62, 0x3ceb, 0x0e70, 0x1ff9,
   0xf78f, 0xe606, 0xd49d, 0xc514, 0xb1ab, 0xa022, 0x92b9, 0x8330,
   0x7bc7, 0x6a4e, 0x58d5, 0x495c, 0x3de3, 0x2c6a, 0x1ef1, 0x0f78
};

#define PPPINITFCS16  0xffff   /* Исходное значение FCS */
#define PPPGOODFCS16  0xf0b8   /* Финальное значение FCS */

/*
 * Расчет нового значения FCS на основе имеющегося и новых данных.
 */
u16 pppfcs16(fcs, cp, len)
    register u16 fcs;
    register unsigned char *cp;
    register int len;
{
    ASSERT(sizeof (u16) == 2);
    ASSERT(((u16) -1) > 0);
    while (len--)
        fcs = (fcs >> 8) ^ fcstab[(fcs ^ *cp++) & 0xff];

    return (fcs);
}

/*
 * Как использовать fcs
 */
tryfcs16(cp, len)
    register unsigned char *cp;
    register int len;
{
    u16 trialfcs;
```

```
    /* добавление на выходе */
    trialfcs = pppfcs16( PPPINITFCS16, cp, len );
    trialfcs ^= 0xffff;                /* дополнение */
    cp[len] = (trialfcs & 0x00ff);     /* сначала младший байт */
    cp[len+1] = ((trialfcs >> 8) & 0x00ff);

    /* проверка ввода */
    trialfcs = pppfcs16( PPPINITFCS16, cp, len + 2 );
    if ( trialfcs == PPPGOODFCS16 )
        printf("Good FCS\n");
}
```

C.3. Метод расчета 32-битовых FCS

Приведенный ниже код обеспечивает табличные операции (поиск) для расчета 32-битовых значений FCS при получении данных интерфейсом.

```
/*
 * Полином для генерации FCS-32: x**0 + x**1 + x**2 + x**4 + x**5
 *                       + x**7 + x**8 + x**10 + x**11 + x**12 + x**16
 *                       + x**22 + x**23 + x**26 + x**32.
 */

/*
 * u32 представляет 32-битовое целое число без знака. typedef зависит от
 * аппаратной архитектуры.
 */
typedef unsigned long u32;

static u32 fcstab_32[256] =
   {
   0x00000000, 0x77073096, 0xee0e612c, 0x990951ba,
   0x076dc419, 0x706af48f, 0xe963a535, 0x9e6495a3,
   0x0edb8832, 0x79dcb8a4, 0xe0d5e91e, 0x97d2d988,
   0x09b64c2b, 0x7eb17cbd, 0xe7b82d07, 0x90bf1d91,
   0x1db71064, 0x6ab020f2, 0xf3b97148, 0x84be41de,
   0x1adad47d, 0x6ddde4eb, 0xf4d4b551, 0x83d385c7,
   0x136c9856, 0x646ba8c0, 0xfd62f97a, 0x8a65c9ec,
   0x14015c4f, 0x63066cd9, 0xfa0f3d63, 0x8d080df5,
   0x3b6e20c8, 0x4c69105e, 0xd56041e4, 0xa2677172,
   0x3c03e4d1, 0x4b04d447, 0xd20d85fd, 0xa50ab56b,
   0x35b5a8fa, 0x42b2986c, 0xdbbbc9d6, 0xacbcf940,
   0x32d86ce3, 0x45df5c75, 0xdcd60dcf, 0xabd13d59,
   0x26d930ac, 0x51de003a, 0xc8d75180, 0xbfd06116,
   0x21b4f4b5, 0x56b3c423, 0xcfba9599, 0xb8bda50f,
   0x2802b89e, 0x5f058808, 0xc60cd9b2, 0xb10be924,
   0x2f6f7c87, 0x58684c11, 0xc1611dab, 0xb6662d3d,
   0x76dc4190, 0x01db7106, 0x98d220bc, 0xefd5102a,
   0x71b18589, 0x06b6b51f, 0x9fbfe4a5, 0xe8b8d433,
   0x7807c9a2, 0x0f00f934, 0x9609a88e, 0xe10e9818,
   0x7f6a0dbb, 0x086d3d2d, 0x91646c97, 0xe6635c01,
   0x6b6b51f4, 0x1c6c6162, 0x856530d8, 0xf262004e,
   0x6c0695ed, 0x1b01a57b, 0x8208f4c1, 0xf50fc457,
   0x65b0d9c6, 0x12b7e950, 0x8bbeb8ea, 0xfcb9887c,
   0x62dd1ddf, 0x15da2d49, 0x8cd37cf3, 0xfbd44c65,
   0x4db26158, 0x3ab551ce, 0xa3bc0074, 0xd4bb30e2,
   0x4adfa541, 0x3dd895d7, 0xa4d1c46d, 0xd3d6f4fb,
   0x4369e96a, 0x346ed9fc, 0xad678846, 0xda60b8d0,
   0x44042d73, 0x33031de5, 0xaa0a4c5f, 0xdd0d7cc9,
   0x5005713c, 0x270241aa, 0xbe0b1010, 0xc90c2086,
   0x5768b525, 0x206f85b3, 0xb966d409, 0xce61e49f,
   0x5edef90e, 0x29d9c998, 0xb0d09822, 0xc7d7a8b4,
   0x59b33d17, 0x2eb40d81, 0xb7bd5c3b, 0xc0ba6cad,
   0xedb88320, 0x9abfb3b6, 0x03b6e20c, 0x74b1d29a,
   0xead54739, 0x9dd277af, 0x04db2615, 0x73dc1683,
   0xe3630b12, 0x94643b84, 0x0d6d6a3e, 0x7a6a5aa8,
   0xe40ecf0b, 0x9309ff9d, 0x0a00ae27, 0x7d079eb1,
   0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe,
   0xf762575d, 0x806567cb, 0x196c3671, 0x6e6b06e7,
   0xfed41b76, 0x89d32be0, 0x10da7a5a, 0x67dd4acc,
   0xf9b9df6f, 0x8ebeeff9, 0x17b7be43, 0x60b08ed5,
   0xd6d6a3e8, 0xa1d1937e, 0x38d8c2c4, 0x4fdff252,
   0xd1bb67f1, 0xa6bc5767, 0x3fb506dd, 0x48b2364b,
   0xd80d2bda, 0xaf0a1b4c, 0x36034af6, 0x41047a60,
   0xdf60efc3, 0xa867df55, 0x316e8eef, 0x4669be79,
   0xcb61b38c, 0xbc66831a, 0x256fd2a0, 0x5268e236,
   0xcc0c7795, 0xbb0b4703, 0x220216b9, 0x5505262f,
   0xc5ba3bbe, 0xb2bd0b28, 0x2bb45a92, 0x5cb36a04,
```

```
      0xc2d7ffa7, 0xb5d0cf31, 0x2cd99e8b, 0x5bdeae1d,
      0x9b64c2b0, 0xec63f226, 0x756aa39c, 0x026d930a,
      0x9c0906a9, 0xeb0e363f, 0x72076785, 0x05005713,
      0x95bf4a82, 0xe2b87a14, 0x7bb12bae, 0x0cb61b38,
      0x92d28e9b, 0xe5d5be0d, 0x7cdcefb7, 0x0bdbdf21,
      0x86d3d2d4, 0xf1d4e242, 0x68ddb3f8, 0x1fda836e,
      0x81be16cd, 0xf6b9265b, 0x6fb077e1, 0x18b74777,
      0x88085ae6, 0xff0f6a70, 0x66063bca, 0x11010b5c,
      0x8f659eff, 0xf862ae69, 0x616bffd3, 0x166ccf45,
      0xa00ae278, 0xd70dd2ee, 0x4e048354, 0x3903b3c2,
      0xa7672661, 0xd06016f7, 0x4969474d, 0x3e6e77db,
      0xaed16a4a, 0xd9d65adc, 0x40df0b66, 0x37d83bf0,
      0xa9bcae53, 0xdebb9ec5, 0x47b2cf7f, 0x30b5ffe9,
      0xbdbdf21c, 0xcabac28a, 0x53b39330, 0x24b4a3a6,
      0xbad03605, 0xcdd70693, 0x54de5729, 0x23d967bf,
      0xb3667a2e, 0xc4614ab8, 0x5d681b02, 0x2a6f2b94,
      0xb40bbe37, 0xc30c8ea1, 0x5a05df1b, 0x2d02ef8d
      };

#define PPPINITFCS32  0xffffffff   /* Исходное значение FCS */
#define PPPGOODFCS32  0xdebb20e3   /* Финальное значение FCS */

/*
 * Расчет нового значения FCS на основе имеющегося и новых данных.
 */
u32 pppfcs32(fcs, cp, len)
    register u32 fcs;
    register unsigned char *cp;
    register int len;
    {
    ASSERT(sizeof (u32) == 4);
    ASSERT(((u32) -1) > 0);
    while (len--)
        fcs = (((fcs) >> 8) ^ fcstab_32[((fcs) ^ (*cp++)) & 0xff]);

    return (fcs);
    }

/*
 * Как использовать fcs
 */
tryfcs32(cp, len)
    register unsigned char *cp;
    register int len;
{
    u32 trialfcs;

    /* добавление на выходе */
    trialfcs = pppfcs32( PPPINITFCS32, cp, len );
    trialfcs ^= 0xffffffff;              /* дополнение */
    cp[len] = (trialfcs & 0x00ff);       /* сначала младший байт */
    cp[len+1] = ((trialfcs >>= 8) & 0x00ff);
    cp[len+2] = ((trialfcs >>= 8) & 0x00ff);
    cp[len+3] = ((trialfcs >> 8) & 0x00ff);

    /* проверка ввода */
    trialfcs = pppfcs32( PPPINITFCS32, cp, len + 4 );
    if ( trialfcs == PPPGOODFCS32 )
        printf("Good FCS\n");
}
```

Вопросы безопасности

Как отмечено в разделе 2. Требования физического уровня, канальный уровень может не получать информации об изменениях на физическом уровне. Это может приводить к потере защиты из-за чрезмерной зависимости от целостности и защищенности систем коммутации и их администрирования. Атаки со вставкой данных могут остаться незамеченными. Атакующий, способный подделать отождествление вызывающей стороны может обойти аутентификацию канала.

Литература

[1] Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", STD 51[1], RFC 1661, Daydreamer, July 1994.

[2] ISO/IEC 3309:1991(E), "Information Technology - Telecommunications and information exchange between systems - High-level data link control (HDLC) procedures - Frame structure", International Organization For Standardization, Fourth edition 1991-06-01.

[1]В оригинале ошибочно указано STD 50. См. https://www.rfc-editor.org/errata/eid542. *Прим. перев.*

[3] ISO/IEC 3309:1991/Amd.2:1992(E), "Information Technology - Telecommunications and information exchange between systems - High-level data link control (HDLC) procedures - Frame structure - Amendment 2: Extended transparency options for start/stop transmission", International Organization For Standardization, 1992-01-15.

[4] ISO/IEC 4335:1991(E), "Information Technology - Telecommunications and information exchange between systems - High-level data link control (HDLC) procedures - Elements of procedures", International Organization For Standardization, Fourth edition 1991-09-15.

[5] Simpson, W., Editor, "PPP LCP Extensions", RFC 1570, Daydreamer, January 1994.

[6] ANSI X3.4-1977, "American National Standard Code for Information Interchange", American National Standards Institute, 1977.

[7] Perez, "Byte-wise CRC Calculations", IEEE Micro, June 1983.

[8] Morse, G., "Calculating CRC's by Bits and Bytes", Byte, September 1986.

[9] LeVan, J., "A Fast CRC", Byte, November 1987.

[10] Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1340[1], USC/Information Sciences Institute, July 1992.

Благодарности

Этот документ является результатом работы группы Point-to-Point Protocol под эгидой IETF[2]. Комментарии к документу следует направлять по адресу списка рассылок ietf-ppp@merit.edu.

Данная спецификация основана на предшествующих RFC, в подготовке которых участвовало множество людей.

Пример кода для 32-битовых FCS предоставлен Karl Fox (Morning Star Technologies).

Отдельная благодарность Morning Star Technologies за предоставление компьютерных ресурсов и доступ в сеть при подготовке этой спецификации.

Адрес руководителя

С рабочей группой можно связаться через ее руководителя.

Fred Baker

Advanced Computer Communications

315 Bollay Drive

Santa Barbara, California 93117

fbaker@acc.com

Адреса авторов

Вопросы, связанные с этим документом можно направлять по приведенному ниже адресу.

William Allen Simpson

Daydreamer

Computer Systems Consulting Services

1384 Fontaine

Madison Heights, Michigan 48071

Bill.Simpson@um.cc.umich.edu

bsimpson@MorningStar.com

Перевод на русский язык

Николай Малых

nmalykh@gmail.com

[1]В соответствии с RFC 3232 документ Assigned Numders утратил силу и данные публикуются на сайте. *Прим. перев.*
[2]Internet Engineering Task Force - Инженерный совет Internet.