Network Working Group
Request for Comments: 1549
Category: Standards Track

W. Simpson, Editor
Daydreamer
December 1993

PPP in HDLC Framing

Status of this Memo

Abstract

The Point-to-Point Protocol (PPP) [1] provides a standard method for transporting multi-protocol datagrams over point-to-point links.

This document describes the use of HDLC for framing PPP encapsulated packets. This document is the product of the Point-to-Point Protocol Working Group of the Internet Engineering Task Force (IETF). Comments should be submitted to the ietf-ppp@ucdavis.edu mailing list.

Table of Contents

1. Introduction

This specification provides for framing over both bit-oriented and octet-oriented synchronous links, and asynchronous links with 8 bits of data and no parity. These links MUST be full-duplex, but MAY be either dedicated or circuit-switched. PPP uses HDLC as a basis for the framing.

An escape mechanism is specified to allow control data such as XON/XOFF to be transmitted transparently over the link, and to remove spurious control data which may be injected into the link by intervening hardware and software.

Some protocols expect error free transmission, and either provide error detection only on a conditional basis, or do not provide it at all. PPP uses the HDLC Frame Check Sequence for error detection. This is commonly available in hardware implementations, and a software implementation is provided.

1.1 Specification of Requirements

In this document, several words are used to signify the requirements of the specification. These words are often capitalized.

MUST

This word, or the adjective "required", means that the definition is an absolute requirement of the specification.

MUST NOT

This phrase means that the definition is an absolute prohibition of the specification.

SHOULD

This word, or the adjective "recommended", means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications must be understood and carefully weighed before choosing a different course.

MAY

This word, or the adjective "optional", means that this item is one of an allowed set of alternatives. An implementation which does not include this option MUST be prepared to interoperate with another implementation which does include the option.

1.2 Terminology

This document frequently uses the following terms:

datagram

The unit of transmission in the network layer (such as IP). A datagram may be encapsulated in one or more packets passed to the data link layer.

frame

The unit of transmission at the data link layer. A frame may include a header and/or a trailer, along with some number of units of data.

packet

The basic unit of encapsulation, which is passed across the interface between the network layer and the data link layer. A packet is usually mapped to a frame; the exceptions are when data link layer fragmentation is being performed, or when multiple packets are incorporated into a single frame.

peer

The other end of the point-to-point link.

silently discard

This means the implementation discards the packet without further processing. The implementation SHOULD provide the capability of logging the error, including the contents of the silently discarded packet, and SHOULD record the event in a statistics counter.

2. Physical Layer Requirements

PPP is capable of operating across most DTE/DCE interfaces (such as, EIA RS-232-C, EIA RS-422, EIA RS-423 and CCITT V.35). The only absolute requirement imposed by PPP is the provision of a full-duplex circuit, either dedicated or circuit-switched, which can operate in either an asynchronous (start/stop), bit-synchronous, or octet-synchronous mode, transparent to PPP Data Link Layer frames.

Interface Format

PPP presents an octet interface to the physical layer. There is

no provision for sub-octets to be supplied or accepted.

PPP does not impose any restrictions regarding transmission rate, other than that of the particular DTE/DCE interface.

Control Signals

PPP does not require the use of control signals, such as Request To Send (RTS), Clear To Send (CTS), Data Carrier Detect (DCD), and Data Terminal Ready (DTR).

When available, using such signals can allow greater functionality and performance. In particular, such signals SHOULD be used to signal the Up and Down events in the LCP Option Negotiation Automaton [1]. When such signals are not available, the implementation MUST signal the Up event to LCP upon initialization, and SHOULD NOT signal the Down event.

Because signalling is not required, the physical layer MAY be decoupled from the data link layer, hiding the transient details of the physical transport. This has implications for mobility in cellular radio networks, and other rapidly switching links.

When moving from cell to cell within the same zone, an implementation MAY choose to treat the entire zone as a single link, even though transmission is switched among several frequencies. The link is considered to be with the central control unit for the zone, rather than the individual cell transceivers. However, the link SHOULD re-establish its configuration whenever the link is switched to a different administration.

Due to the bursty nature of data traffic, some implementations have choosen to disconnect the physical layer during periods of inactivity, and reconnect when traffic resumes, without informing the data link layer. Robust implementations should avoid using this trick over-zealously, since the price for decreased setup latency is decreased security. Implementations SHOULD signal the Down event whenever "significant time" has elapsed since the link was disconnected. The value for "significant time" is a matter of considerable debate, and is based on the tariffs, call setup times, and security concerns of the installation.

3. The Data Link Layer

PPP uses the principles, terminology, and frame structure of the International Organization For Standardization’s (ISO) 3309-1979

High-level Data Link Control (HDLC) frame structure [2], as modified by "Addendum 1: Start/stop transmission" [3], which specifies modifications to allow HDLC use in asynchronous environments.

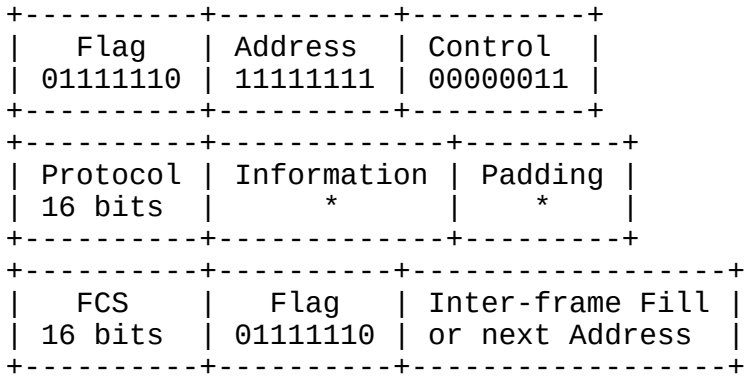
The PPP control procedures use the definitions and Control field encodings standardized in ISO 4335-1979 [4] and ISO 4335-1979/Addendum 1-1979 [5]. PPP framing is also consistent with CCITT Recommendation X.25 LAPB [6], and CCITT Recommendation Q.922 [7], since those are also based on HDLC.

The purpose of this specification is not to document what is already standardized in ISO 3309. It is assumed that the reader is already familiar with HDLC, or has access to a copy of [2] or [6]. Instead, this document attempts to give a concise summary and point out specific options and features used by PPP.

To remain consistent with standard Internet practice, and avoid confusion for people used to reading RFCs, all binary numbers in the following descriptions are in Most Significant Bit to Least Significant Bit order, reading from left to right, unless otherwise indicated. Note that this is contrary to standard ISO and CCITT practice which orders bits as transmitted (network bit order). Keep this in mind when comparing this document with the international standards documents.

3.1 Frame Format

A summary of the PPP HDLC frame structure is shown below. This figure does not include start/stop bits (for asynchronous links), nor any bits or octets inserted for transparency. The fields are transmitted from left to right.



The Protocol, Information and Padding fields are described in the Point-to-Point Protocol Encapsulation [1].

Flag Sequence

The Flag Sequence indicates the beginning or end of a frame, and always consists of the binary sequence 01111110 (hexadecimal 0x7e).

The Flag Sequence is a frame separator. Only one Flag Sequence is required between two frames. Two consecutive Flag Sequences constitute an empty frame, which is ignored, and not counted as a FCS error.

Address Field

The Address field is a single octet and contains the binary sequence 11111111 (hexadecimal 0xff), the All-Stations address. PPP does not assign individual station addresses. The All-Stations address MUST always be recognized and received. The use of other address lengths and values may be defined at a later time, or by prior agreement. Frames with unrecognized Addresses SHOULD be silently discarded.

Control Field

The Control field is a single octet and contains the binary sequence 00000011 (hexadecimal 0x03), the Unnumbered Information (UI) command with the P/F bit set to zero. The use of other Control field values may be defined at a later time, or by prior agreement. Frames with unrecognized Control field values SHOULD be silently discarded.

Frame Check Sequence (FCS) Field

The Frame Check Sequence field is normally 16 bits (two octets). The use of other FCS lengths may be defined at a later time, or by prior agreement. The FCS is transmitted with the coefficient of the highest term first.

The FCS field is calculated over all bits of the Address, Control, Protocol, Information and Padding fields, not including any start and stop bits (asynchronous) nor any bits (synchronous) or octets (asynchronous or synchronous) inserted for transparency. This also does not include the Flag Sequences nor the FCS field itself.

> Note: When octets are received which are flagged in the Async-Control-Character-Map, they are discarded before calculating the FCS.
>
> For more information on the specification of the FCS, see ISO

3309 [2] or CCITT X.25 [6].

The end of the Information and Padding fields is found by locating the closing Flag Sequence and removing the Frame Check Sequence field.

3.2. Modification of the Basic Frame

The Link Control Protocol can negotiate modifications to the basic HDLC frame structure. However, modified frames will always be clearly distinguishable from standard frames.

Address-and-Control-Field-Compression

When using the default HDLC framing, the Address and Control fields contain the hexadecimal values 0xff and 0x03 respectively.

On transmission, compressed Address and Control fields are formed by simply omitting them.

On reception, the Address and Control fields are decompressed by examining the first two octets. If they contain the values 0xff and 0x03, they are assumed to be the Address and Control fields. If not, it is assumed that the fields were compressed and were not transmitted.

By definition, the first octet of a two octet Protocol field will never be 0xff (since it is not even). The Protocol field value 0x00ff is not allowed (reserved) to avoid ambiguity when Protocol-Field-Compression is enabled and the first Information field octet is 0x03.

When other Address or Control field values are in use, Address-and-Control-Field-Compression MUST NOT be negotiated.

4. Asynchronous HDLC

This section summarizes the use of HDLC with 8-bit asynchronous links.

Flag Sequence

The Flag Sequence indicates the beginning or end of a frame. The octet stream is examined on an octet-by-octet basis for the value 01111110 (hexadecimal 0x7e).

Transparency

An octet stuffing procedure is used. The Control Escape octet is defined as binary 01111101 (hexadecimal 0x7d) where the bit positions are numbered 87654321 (not 76543210, BEWARE).

Each end of the link maintains two Async-Control-Character-Maps. The receiving ACCM is 32 bits, but the sending ACCM may be up to 256 bits. This results in four distinct ACCMs, two in each direction of the link.

The default receiving ACCM is 0xffffffff. The default sending ACCM is 0xffffffff, plus the Control Escape and Flag Sequence characters themselves, plus whatever other outgoing characters are known to be intercepted.

After FCS computation, the transmitter examines the entire frame between the two Flag Sequences. Each Flag Sequence, Control Escape octet, and octet with value less than hexadecimal 0x20 which is flagged in the sending Async-Control-Character-Map, is replaced by a two octet sequence consisting of the Control Escape octet and the original octet with bit 6 complemented (exclusive-or'd with hexadecimal 0x20).

Prior to FCS computation, the receiver examines the entire frame between the two Flag Sequences. Each octet with value less than hexadecimal 0x20 is checked. If it is flagged in the receiving Async-Control-Character-Map, it is simply removed (it may have been inserted by intervening data communications equipment). For each Control Escape octet, that octet is also removed, but bit 6 of the following octet is complemented, unless it is the Flag Sequence.

> Note: The inclusion of all octets less than hexadecimal 0x20 allows all ASCII control characters [8] excluding DEL (Delete) to be transparently communicated through all known data communications equipment.

The transmitter may also send octets with value in the range 0x40 through 0xff (except 0x5e) in Control Escape format. Since these octet values are not negotiable, this does not solve the problem of receivers which cannot handle all non-control characters. Also, since the technique does not affect the 8th bit, this does not solve problems for communications links that can send only 7-bit characters.

A few examples may make this more clear. Packet data is transmitted on the link as follows:

0x7e is encoded as 0x7d, 0x5e. 0x7d is encoded as 0x7d, 0x5d. 0x01 is encoded as 0x7d, 0x21.

Some modems with software flow control may intercept outgoing DC1 and DC3 ignoring the 8th (parity) bit. This data would be transmitted on the link as follows:

0x11 is encoded as 0x7d, 0x31. 0x13 is encoded as 0x7d, 0x33. 0x91 is encoded as 0x7d, 0xb1. 0x93 is encoded as 0x7d, 0xb3.

Aborting a Transmission

On asynchronous links, frames may be aborted by transmitting a "0" stop bit where a "1" bit is expected (framing error) or by transmitting a Control Escape octet followed immediately by a closing Flag Sequence.

Time Fill

For asynchronous links, inter-octet and inter-frame time fill MUST be accomplished by transmitting continuous "1" bits (mark-hold state).

Inter-frame time fill can be viewed as extended inter-octet time fill. Doing so can save one octet for every frame, decreasing delay and increasing bandwidth. This is possible since a Flag Sequence may serve as both a frame close and a frame begin. After having received any frame, an idle receiver will always be in a frame begin state.

Robust transmitters should avoid using this trick over-zealously, since the price for decreased delay is decreased reliability. Noisy links may cause the receiver to receive garbage characters and interpret them as part of an incoming frame. If the transmitter does not send a new opening Flag Sequence before sending the next frame, then that frame will be appended to the noise characters causing an invalid frame (with high reliability). It is suggested that implementations will achieve the best results by always sending an opening Flag Sequence if the new frame is not back-to-back with the last. Transmitters SHOULD send an open Flag Sequence whenever "appreciable time" has elapsed after the prior closing Flag Sequence. The maximum value for "appreciable time" is likely to be no greater than the typing rate of a slow typist, say 1 second.

Encoding

All octets are transmitted with one start bit, eight bits of data,

and one stop bit. There is no provision for seven bit asynchronous links.

5. Bit-synchronous HDLC

This section summarizes the use of HDLC with bit-synchronous links.

Flag Sequence

The Flag Sequence indicates the beginning or end of a frame, and is used for frame synchronization. The bit stream is examined on a bit-by-bit basis for the binary sequence 01111110 (hexadecimal 0x7e).

The "shared zero mode" Flag Sequence "011111101111110" SHOULD NOT be used. When not avoidable, such an implementation MUST ensure that the first Flag Sequence detected (the end of the frame) is promptly communicated to the link layer. Use of the shared zero mode hinders interoperability with synchronous-to-asynchronous converters.

Transparency

The transmitter examines the entire frame between the two Flag Sequences. A "0" bit is inserted after all sequences of five contiguous "1" bits (including the last 5 bits of the FCS) to ensure that a Flag Sequence is not simulated.

When receiving, any "0" bit that directly follows five contiguous "1" bits is discarded.

Since the Control Escape octet-stuffing method is not used, the default receiving and sending Async-Control-Character-Maps are 0.

There may be some use of synchronous-to-asynchronous converters (some built into modems) in point-to-point links resulting in a synchronous PPP implementation on one end of a link and an asynchronous implementation on the other. It is the responsibility of the converter to do all mapping conversions during operation.

To enable this functionality, bit-synchronous PPP implementations MUST always respond to the Async-Control-Character-Map Configuration Option with an LCP Configure-Ack. However, acceptance of the Configuration Option does not imply that the bit-synchronous implementation will do any octet mapping. Instead, all such octet mapping will be performed by the asynchronous-to-synchronous converter.

Aborting a Transmission

A sequence of more than six "1" bits indicates an invalid frame, which is ignored, and not counted as a FCS error.

Inter-frame Time Fill

For bit-synchronous links, the Flag Sequence SHOULD be transmitted during inter-frame time fill. There is no provision for inter-octet time fill.

Mark idle (continuous ones) SHOULD NOT be used for inter-frame ill. However, certain types of circuit-switched links require the use of mark idle, particularly those that calculate accounting based on periods of bit activity. When mark idle is used on a bit-synchronous link, the implementation MUST ensure at least 15 consecutive "1" bits between Flags during the idle period, and that the Flag Sequence is always generated at the beginning of a frame after an idle period.

Encoding

The definition of various encodings and scrambling is the responsibility of the DTE/DCE equipment in use, and is outside the scope of this specification.

While PPP will operate without regard to the underlying representation of the bit stream, lack of standards for transmission will hinder interoperability as surely as lack of data link standards. At speeds of 56 Kbps through 2.0 Mbps, NRZ is currently most widely available, and on that basis is recommended as a default.

When configuration of the encoding is allowed, NRZI is recommended as an alternative, because of its relative immunity to signal inversion configuration errors, and instances when it MAY allow connection without an expensive DSU/CSU. Unfortunately, NRZI encoding obviates the (1 + x) factor of the 16-bit FCS, so that one error in 2**15 goes undetected (instead of one in 2**16), and triple errors are not detected. Therefore, when NRZI is in use, it is recommended that the 32-bit FCS be negotiated, which does not include the (1 + x) factor.

At higher speeds of up to 45 Mbps, some implementors have chosen the ANSI High Speed Synchronous Interface [HSSI]. While this experience is currently limited, implementors are encouraged to cooperate in choosing transmission encoding.

6. Octet-synchronous HDLC

This section summarizes the use of HDLC with octet-synchronous links, such as SONET and optionally ISDN B or H channels.

Although the bit rate is synchronous, there is no bit-stuffing. Instead, the octet-stuffing feature of 8-bit asynchronous HDLC is used.

Flag Sequence

The Flag Sequence indicates the beginning or end of a frame. The octet stream is examined on an octet-by-octet basis for the value 01111110 (hexadecimal 0x7e).

Transparency

An octet stuffing procedure is used. The Control Escape octet is defined as binary 01111101 (hexadecimal 0x7d).

The octet stuffing procedure is described in "Asynchronous HDLC" above.

The sending and receiving implementations need escape only the Flag Sequence and Control Escape octets.

Considerations concerning the use of converters are described in "Bit-synchronous HDLC" above.

Aborting a Transmission

Frames may be aborted by transmitting a Control Escape octet followed immediately by a closing Flag Sequence. The preceding frame is ignored, and not counted as a FCS error.

Inter-frame Time Fill

The Flag Sequence MUST be transmitted during inter-frame time fill. There is no provision for inter-octet time fill.

Encoding

The definition of various encodings and scrambling is the responsibility of the DTE/DCE equipment in use, and is outside the scope of this specification.

A. Fast Frame Check Sequence (FCS) Implementation

The FCS was originally designed with hardware implementations in mind. A serial bit stream is transmitted on the wire, the FCS is calculated over the serial data as it goes out, and the complement of the resulting FCS is appended to the serial stream, followed by the Flag Sequence.

The receiver has no way of determining that it has finished calculating the received FCS until it detects the Flag Sequence. Therefore, the FCS was designed so that a particular pattern results when the FCS operation passes over the complemented FCS. A good frame is indicated by this "good FCS" value.

A.1 FCS Computation Method

The following code provides a table lookup computation for calculating the Frame Check Sequence as data arrives at the interface. This implementation is based on [9], [10], and [11]. The table is created by the code in section B.2.

```
/*
 * u16 represents an unsigned 16-bit number.  Adjust the typedef for
 * your hardware.
 */
typedef unsigned short u16;

/*
 * FCS lookup table as calculated by the table generator in section B.2
 */
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  /* Initial FCS value */
#define PPPGOODFCS16    0xf0b8  /* Good final FCS value */

/*
```

```
 * Calculate a new fcs given the current fcs and the new data.
 */
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);
}

/*
 * How to use the fcs
 */
tryfcs16(cp, len)
    register unsigned char *cp;
    register int len;
{
    u16 trialfcs;

    /* add on output */
    trialfcs = pppfcs16( PPPINITFCS16, cp, len );
    trialfcs ^= 0xffff;             /* complement */
    cp[len] = (trialfcs & 0x00ff);  /* least significant byte first */
    cp[len+1] = ((trialfcs >> 8) & 0x00ff);

    /* check on input */
    trialfcs = pppfcs16( PPPINITFCS16, cp, len + 2 );
    if ( trialfcs == PPPGOODFCS16 )
        printf("Good FCS0);
}
```

A.2. Fast FCS table generator

The following code creates the lookup table used to calculate the FCS.

```
/*
 * Generate a FCS table for the HDLC FCS.
 *
 * Drew D. Perkins at Carnegie Mellon University.
 *
 * Code liberally borrowed from Mohsen Banan and D. Hugh Redelmeier.
 */

/*
 * The HDLC polynomial: x**0 + x**5 + x**12 + x**16 (0x8408).
 */
#define P       0x8408


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

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

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

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

Security Considerations

As noted in the Physical Layer Requirements section, the link layer might not be informed when the connected state of physical layer is changed. This results in possible security lapses due to over-reliance on the integrity and security of switching systems and administrations. An insertion attack might be undetected. An attacker which is able to spoof the same calling identity might be able to avoid link authentication.

References

[1] Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", RFC 1548, December 1993

[2] International Organization For Standardization, ISO Standard 3309-1979, "Data communication - High-level data link control procedures - Frame structure", 1979.

[3] International Organization For Standardization, Proposed Draft International Standard ISO 3309-1991/PDAD1, "Information processing systems - Data communication - High-level data link control procedures - Frame structure - Addendum 1: Start/stop transmission", 1991.

[4] International Organization For Standardization, ISO Standard 4335-1979, "Data communication - High-level data link control procedures - Elements of procedures", 1979.

[5] International Organization For Standardization, ISO Standard 4335-1979/Addendum 1, "Data communication - High-level data link control procedures - Elements of procedures - Addendum 1", 1979.

[6] International Telecommunication Union, CCITT Recommendation X.25, "Interface Between Data Terminal Equipment (DTE) and Data Circuit Terminating Equipment (DCE) for Terminals Operating in the Packet Mode on Public Data Networks", CCITT Red Book, Volume VIII, Fascicle VIII.3, Rec. X.25., October 1984.

[7] International Telegraph and Telephone Consultative Committee, CCITT Recommendation Q.922, "ISDN Data Link Layer Specification for Frame Mode Bearer Services", April 1991.

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

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

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

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

Acknowledgments

This specification is based on previous RFCs, where many

contributions have been acknowleged.

Additional implementation detail for this version was provided by Fred Baker (ACC), Craig Fox (NSC), and Phil Karn (Qualcomm).

Special thanks to Morning Star Technologies for providing computing resources and network access support for writing this specification.

Chair's Address

The working group can be contacted via the current chair:

Fred Baker
Advanced Computer Communications
315 Bollay Drive
Santa Barbara, California, 93111

EMail: fbaker@acc.com

Editor's Address

Questions about this memo can also be directed to:

William Allen Simpson
Daydreamer
Computer Systems Consulting Services
1384 Fontaine
Madison Heights, Michigan 48071

EMail: Bill.Simpson@um.cc.umich.edu