POS SDLC EXPLAINED

SYNCHRONOUS DATA LINK CONTROL ON PAYMENT TERMINAL NETWORKS

In some markets, dial-up payment terminals use a synchronous communications protocol based on IBM's SDLC. While more reliable and efficient than its asynchronous counterpart, SDLC requires more effort to implement on a terminal. It is also more difficult to troubleshoot, especially without dedicated tools. This report delves into the topic.

WHAT IS SDLC?

SDLC is an acronym for Synchronous Data Link Control, a protocol developed by IBM in the early 1970s [refer to ib86]. It defines a way for two nodes to communicate in an efficient and reliable manner, with various options to satisfy different application requirements. It was the precursor to the HDLC protocol, standardised in ISO 13239.

SDLC on POS terminals

SDLC implementations on POS terminals consist of a restricted subset of the available functionality in IBM SDLC, leading to relatively simple and compact implementations.

Specifically, the link is set up in Normal Response Mode, where the host NAC acts as the master, while the POS terminal acts as a slave. The link operates in half-duplex mode, where the host periodically polls the terminal, and in turn the terminal only responds to a poll from the host. Furthermore, the window size is restricted to one, implying a stop-and-wait ARQ behaviour.

Benefits of SDLC

SDLC provides three key benefits:

(1) As a data-link layer, SDLC ensures reliable node-to-node delivery. The protocol can detect errors and request retransmissions automatically.

(2) The bit-oriented nature of the protocol has about 20% less overhead than an asynchronous equivalent. With an asynchronous protocol, each 8-bit character will have a start bit and a stop bit added to it.

(3) The protocol is transparent to the application, greatly simplifying application development.

Issues with POS SDLC

The SDLC stack is normally provided by the terminal vendor as part of the onboard OS, and the data-link layer is typically implemented as a separate process. If you are the vendor, then you have some work to do.

Although SDLC does support message fragmentation and reassembly, traditional POS implementations are usually expected to handle the largest message *without* fragmentation. This limit is usually around 700 bytes.

While the effect on typical transaction request/response exchanges is minimal, the stop-and-wait ARQ severely impacts throughput of sustained transfers, as is the case with downloads[1].

Another issue concerns what happens in the idle period. When idle, stations can send either contiguous ones or flag patterns. In a few implementations, contiguous flags (...0111111001111110 …) share the same zero bit (...011111101111110...).

[1] ITU-T V.42 is better suited for terminal software downloads because it supports selective repeat ARQ, which is more efficient. Furthermore, it is conveniently built into most modems as a standard feature.

IMPLEMENTING SDLC ON A POS TERMINAL

For a vendor implementing SDLC on a terminal, there are two aspects that need addressing: the physical layer and the data-link layer. We are assuming that the underlying modulation scheme is inherently synchronous, as is the case with the widely deployed V.22 and V.22*bis* modes.

The data-link layer processes frame headers and checksums, handles the protocol state changes, and responds to commands from the host, as needed.

The physical layer performs more mundane functions, namely adding flag delimiters and performing zero insertion and deletion. In the past, terminals were equipped with dumb modems (controllerless datapumps), plus an external HDLC controller or software-based equivalent. More recently, smart POS modems often have special *pseudo-synchronous* modes that permit the sending and receiving of frames in synchronous mode over the traditional asynchronous interface between the modem and host-processor. Furthermore, where implemented, ITU-T V.80 provides a standardized method.

When sending frames, especially large ones, care needs to be taken to ensure that no buffer overruns or underruns occur, by using appropriate flow control between host CPU and modem.

A pitfall to avoid concerns the stop-and-wait behaviour and the possible reception of a host poll while the terminal is transmitting an info-frame. While the frame is being transmitted, any received command frames should not be queued, but ignored, to avoid protocol disruption.

Implementation options for POS SDLC

Switched Point-to-Point: The connection is established as needed, and there are two link stations.

Normal Response Mode: In Normal Response Mode (NRM), the host is the Primary (master) station, while the terminal is the Secondary (slave) station. The host polls the terminal with commands, to which the terminal responds. Supervisory and Information frames follow the modulo-8 format used for POS SDLC.

Two-way alternate: This is analogous to half-duplex operation, which means at any one time, only one station can be transmitting information.

Window size: SDLC is a windowing protocol, however, for POS SDLC, the window size is limited to one. This means that at any one time, only one frame can be pending in each direction. Consequently, the P/F bit is always set.

Maximum frame size: Due to the restricted window size, the maximum frame size must accommodate the largest exchange envisaged by the application. Typical values are around 700. This might not sound like much, but keep in mind that a 1 KB packet takes around seven seconds to transit over a 1200 bps connection.

Commands/responses: SNRM/UA during link setup; RR for signalling an acknowledgement. Optionally: RNR for receiver not ready; DISC/DM for link tear-down.

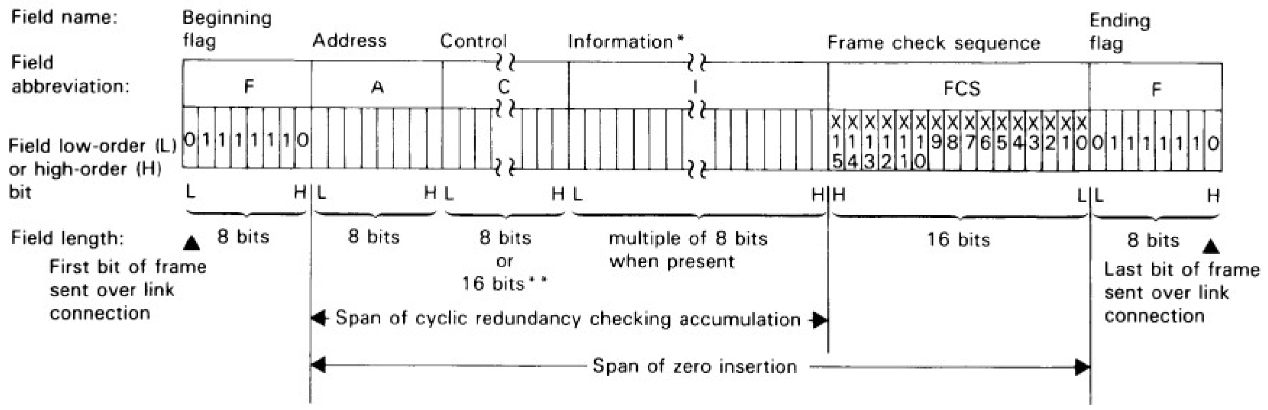
SDLC address: By convention, fixed at 0x30 on POS SDLC networks.

REFERENCE INFORMATION

SDLC overview

For detailed information, consult reference document [ib86].

SDLC frame structure



SDLC frame types

<u>**Unnumbered (U) frames:**</u> are mainly used for establishing and tearing down the link. Also used for reporting certain procedural errors, and transferring auxiliary data.

<u>**Supervisory (S) frames:**</u> assist in the transfer of information, such as acknowledging received frames, conveying ready or busy conditions, and to report frame numbering errors. S-frames always contain the receive count.

<u>**Information (I) frames:**</u> are used to transfer information; also contain both send and receive counts.

<u>**Poll/final bit (P/F):**</u> sent by a primary to a secondary station to elicit a response; sent by a secondary station to indicate the last frame of a transmission.

Typical message exchange

Below is an annotated message exchange based on a trace obtained using the 3am LineScope tool.

```
00:08.907 [   ] V.22 handshake - CONNECT 1200 bps
```

Following a successful modem handshake, the host sends the Set Normal Response Mode (SNRM) command periodically until it receives an Unnumbered Acknowledgement (UA) from the terminal.

```
00:09.762 [Ans] SDLC/HDLC Addr[30] SNRM,P
  Header[3093] Checksum[F71F]
00:09.865 [Ori] SDLC/HDLC Addr[30] UA,F
  Header[3073] Checksum[F9F8]
```

The host then polls the terminal with Receiver Ready (RR) supervisory frames, and the terminal responds. The associated number 0 is the received message counter Nr at each end.

```
00:09.955 [Ans] SDLC/HDLC Addr[30] RR0,P
  Header[3011] Checksum[EDB8]
00:10.058 [Ori] SDLC/HDLC Addr[30] RR0,F
  Header[3011] Checksum[EDB8]
00:10.142 [Ans] SDLC/HDLC Addr[30] RR0,P
  Header[3011] Checksum[EDB8]
```

Once the terminal has an application request message to send to the host, it will respond to the poll from the host with an Information frame (I-frame), containing the message. As far as the application is concerned, it sent a 28-byte message.

```
00:10.433 [Ori] SDLC/HDLC Addr[30] I0,0,F (28 bytes)
  Header[3010] Checksum[F5AB]
     0: 60 00 03 00 00 08 00 20 00 01 00 00 80 00 00 99  `...............
    16: 00 00 00 03 39 32 31 30 35 39 34 37              ....92105947
```

The host then polls the terminal with Receiver Ready (RR) with an updated received message counter Nr=1, signalling its acknowledgement.

```
00:10.515 [Ans] SDLC/HDLC Addr[30] RR1,P
  Header[3031] Checksum[EF99]
00:10.633 [Ori] SDLC/HDLC Addr[30] RR0,F
  Header[3011] Checksum[EDB8]
00:10.708 [Ans] SDLC/HDLC Addr[30] RR1,P
  Header[3031] Checksum[EF99]
00:10.800 [Ori] SDLC/HDLC Addr[30] RR0,F
  Header[3011] Checksum[EDB8]
00:10.882 [Ans] SDLC/HDLC Addr[30] RR1,P
  Header[3031] Checksum[EF99]
00:10.980 [Ori] SDLC/HDLC Addr[30] RR0,F
  Header[3011] Checksum[EDB8]
00:11.062 [Ans] SDLC/HDLC Addr[30] RR1,P
  Header[3031] Checksum[EF99]
00:11.160 [Ori] SDLC/HDLC Addr[30] RR0,F
  Header[3011] Checksum[EDB8]
```

```
00:11.242 [Ans] SDLC/HDLC Addr[30] RR1,P
  Header[3031] Checksum[EF99]
00:11.340 [Ori] SDLC/HDLC Addr[30] RR0,F
  Header[3011] Checksum[EDB8]
```

Once the host has an application response message to send, it polls the terminal with an I-frame. The numbers correspond to the message counters Ns=0 and Nr=1.

```
00:11.902 [Ans] SDLC/HDLC Addr[30] I0,1,P (72 bytes)
  Header[3030] Checksum[4BA2]
     0: 60 01 00 00 03 08 10 20 00 01 00 02 80 00 02 99  `...............
    16: 00 00 00 03 30 30 39 32 31 30 35 39 34 37 00 40  ....0092105947·@
    32: 54 45 53 54 45 20 43 4f 4d 55 4e 49 43 2e 4f 4b  TESTE·COMUNIC.OK
    48: 20 20 20 20 54 45 43 4c 45 20 41 4e 55 4c 41 20  ····TECLE·ANULA·
    64: 20 20 20 20 20 20 20 20                          ........
```

Now, the terminal responds with RR and updated Nr=1, signalling its acknowledgement. As far as the application is concerned, it received a 72-byte message.

```
00:12.013 [Ori] SDLC/HDLC Addr[30] RR1,F
  Header[3031] Checksum[EF99]
00:12.095 [Ans] SDLC/HDLC Addr[30] RR1,P
  Header[3031] Checksum[EF99]
00:12.200 [Ori] SDLC/HDLC Addr[30] RR1,F
  Header[3031] Checksum[EF99]
00:12.282 [Ans] SDLC/HDLC Addr[30] RR1,P
  Header[3031] Checksum[EF99]
00:12.399 [Ori] SDLC/HDLC Addr[30] RR1,F
  Header[3031] Checksum[EF99]
00:12.482 [Ans] SDLC/HDLC Addr[30] RR1,P
  Header[3031] Checksum[EF99]
```

Although not universally implemented, there are defined procedures to tear down the link. The host can send a DISC,P command, to which the terminal responds with a DM,F. Alternatively, the terminal can respond to a poll with the Request Disconnect (RD,F) response. But, more often than not, disconnection is initiated by one of the modems rudely hanging-up, as is the case here.

```
00:12.579 [   ] DISCONNECT
```

Given the nature of POS SDLC, dedicated tools can really help make your life easier. 3am LineScope, a diagnostic product for V.22 and V.22*bis* modems, is tailor-made for POS SDLC development and troubleshooting. It provides a unique combination of features that provide insight into the physical and data-link layers in one convenient and easy-to-use package.

We welcome queries, feedback, and problem-solving experiences from readers.

Email us at info@3amSystems.com | Product information at www.3amSystems.com/LineScope

Acronyms

ARQ	*Automatic repeat request*
bps	*bits per second*
HDLC	*High-level data link control*
IBM	*International business machines*
ISO	*International Organization for Standardization*
ITU	*International telecommunications union*
ITU-T	*ITU telecommunication standardization sector*
NAC	*Network access controller*
OS	*Operating system*
POS	*Point of sale (terminal)*
PSTN	*Public switched telephone network*
SDLC	*Synchronous data link control*

Recommendations / standards

ISO 13239	*Telecommunications and information exchange between systems: High-level data link control (HDLC) procedures*
V.22	*1200 bps duplex modem standardized for use in the PSTN*
V.22bis	*2400 bps duplex modem using the frequency division technique*
V.42	*Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion*
V.80	*In-band DCE control and synchronous data modes for asynchronous DTE*

References

[ib86] *"Synchronous data link control - Concepts"*
IBM GA27-3093-3, Jun-1986