RCSL No.: 42-i2125

Edition: 83.12.16

Author: Lars Bone Jørgensen

Title:

RC3900 SDLC Driver
Reference Manual

TABLE OF CONTENTS PAGE

APPENDIX

1. INTRODUCTION

The driver described in this manual is the RC3900 user interface to the SDLC-function on the COM601 communication controller.

The RC3900 SDLC agrees with IBM SDLC (ref. 1) with limitations in the command repertoire and the number of data link configurations.

The intention is to serve programs written in Real-Time Pascal. Therefore the message interface is based on the PI-3 machine formats and mechanisms. When programs in other languages, such as PL/M-86, want to communicate with the driver they must obey the PI-3 machine conventions.

2. SPECIFICATION OF SDLC

This chapter describes the limitations of the SDLC driver compared to IBM SDLC.

The RC3900 SDLC can only act as a secondary station in the following data link configurations:

- Half-duplex, point-to-point, nonswitched
- Duplex, point-to-point, nonswitched
- Half-duplex, multipoint, nonswitched
- Duplex, multipoint, nonswitched
- Half-duplex, point-to-point, switched

The RC3900 SDLC is able to receive the following repertoire of commands from a primary:

I - Numbered information frame.

RR - Receive Ready.

RNR - Receive Not Ready.

SNRM - Set Normal Response Mode.

DISC - Disconnect.

TEST - Test command.
The primary solicits a TEST response from the secondary with the same information field as the TEST command.

XID - Exchange station identification.
The primary solicits on XID response with the station identification as information field.

The RC3900 SDLC can send the following responses to a primary:

I - Numbered information frame.

RR - Receive Ready.

RNR - Receive Not Ready.

FRMR - Frame Reject.
The response is sent when the secondary receives an invalid frame.

UA - Unnumbered acknowledge.

DM - Disconnect Mode.
The secondary is in the disconnected mode.

TEST - Test response.
Response to a TEST command with the same information field as the TEST command.

XID - Exchange station identification.
Response to an XID command from the primary.

3. USER INTERFACE

A request is passed to the SDLC driver by sending a message to a mailbox. The mailbox is a process parameter.

The requests are divided in three categories:

- Control and supervisor requests.
- Read request.
- Write request.

Read and write requests are always returned in the same order as received, either when data is transferred or caused by a Return Buffers request. Write requests are also returned when SNRM command is received from the primary, and when the state of the driver changes to disconnect mode or connecting mode.

Control requests are returned immediately, except the event request which is queued and returned when the state of the driver changes.

Buffers containing request messages, must have the format used by the PI-3 machine (ref. 3), and obey the rules for inter task communication.

Buffers are used according to the driver conventions for RTP, i.e. actual data is placed in the data area, requests in ul and results returned in u2.

ul : Function modification * 4 + function.
ul is used to specify the function to be performed by the driver.

u2 : Return value from the driver containing the result and status of the the request.

In all requests except event requests the field has the following format:

result modification * 8 + result.

result = 0, ok
= 1, not processed
= 2, time out
= 4, unintelligible
= 5, receiver buffer overflow
= 6, transmission attempted
= 7, already connected

result modification, bit 0 = DSR is off.
bit 1 = A CD failure has occurred.
bit 2 = A CTS failure has occurred.
bit 3 = A FRMR frame is sent
bit 4 = A DISC frame is received

The driver does not support facilities for testoutput generation, but the user is referred to the general testoutput driver, described in ref 2.

4. CONTROL REQUESTS

The control requests are used to change the mode of the SDLC driver. The driver has three modes as described in the next paragraphs.

Disconnect mode: Initially the driver is in the disconnect mode, and enters connecting mode after the reception of a connect request. The SDLC driver will enter disconnect mode in the following circumstances:

- The reception of a disconnect request.
- The reception of a DISC command from the primary.
- After the transmission of a FRMR response.
- After an unsuccesful connect request.

Connecting mode: The driver has received a connect request and is waiting for a SNRM (Set Normal Response Mode) command from the primary, which makes the channel change to connected mode.
If the SNRM command is not received within the period defined in the connect request, an event request is returned with result timeout.
If the autoconnecting parameter in the connect request was true, the driver will try to connect again otherwise it will change to disconnect mode.

Connected mode: The driver is connected and responds to requests as described in the following chapters.

Connect and disconnect requests are always returned immediately, and the actual change in state is reported as an answer to an event request.

4.1 Sense

The sense request will return the current modem signals.

```
    request:
            function = 0, function modification = 0

    answer:
            result = ok
            result modification = any combination of status bits.
```

4.2 Connect

The connect request, which is answered at once, causes the driver to enter connecting mode and wait for an SNRM command from the primary. When it is received, the driver changes to connected mode, and an event request is answered with result=connected.

If no SNRM command is received within the period defined by the setup timer, an event request is answered with result=timeout. If the autoconnecting parameter is true the driver stays in the connecting mode and try to connect again, otherwise it changes to disconnect mode.

```
    request:
            function = 0, function modification = 1
            data area: Connect description
```

The data area in a connect request is a record which describes the SDLC link and the configuration.

Data area:	Address field
+1:	DSR timer
+2:	Setup timer
+3:	Line activity timer
+4:	Flags
+5:	Configuration
+6:	Autoconnect
+7:	Station identification
.	
.	
.	
+26:	

Address field: The address field of the SDLC frames transmitted and received by the secondary.

DSR timer: The maximum time in seconds from the secondary sets the DTR signal to modem answer with DSR signal. DSR timer equal zero means no timeout.

Setup timer: The maximum time in seconds from the secondary initiates connecting to the reception of a SNRM frame. Setup timer equal zero means no timeout.

Line activity timer: The maximum time in seconds between reception of frames from the primary. Zero means no timeout.

Autoconnect: If autoconnect is not zero the driver will enter connecting mode after a fatal error and try to connect again.

Flags If flags is not zero the driver will send between 10 and 15 flags between frames.

Configuration: bit 0 - duplex/half duplex (1/0)
bit 1 - point-to-point/multipoint (1/0)
bit 2 - switched/nonswitched (1/0)
bit 3 - positiv logic/NRZI (1/0)
If the combination of bits disagrees with the possible data link configurations in chapter 2, the request is returned with result = unintelligible.

Station id: The information field sent in an XID response. The size of the field is defined by top of connect description, with a maximum of 20 bytes.

The connect request controls the modem signals DTR (Data Terminal Ready) according to the value of CI (Calling Indicator) and the configuration parameter.

If the configuration is nonswitched DTR is set immediately by the connect request.

If the configuration is switched DTR is set if DSR (Data Set Ready) is on. If DSR is off, the driver awaits the detection of CI, and then set the DTR.

Answer:

result = ok, unintelligible or already connected.
result modification = 0

4.3 Disconnect

The disconnect request initiates disconnection of the link and any transmission in progress is aborted.

The disconnect request is answered at once and when the disconnection is terminated an event is returned with result=disconnected.

After a disconnection the DTR signal remains high, and a clear_DTR request has to be sent to remove it.

```
    request:
            function = 0, function modification = 2

    answer :
            result = ok.
            result modification = 0
```

4.4 Clear DTR

The modem signal DTR is cleared. The request is used to explicit clear DTR after the link is disconnected for some reason.

If the link is not disconnected, at the time the request is received, the driver performs a disconnection before the DTR is cleared.

```
    request:
            function = 0, function modification = 3

    answer:
            result = ok, result modification = 0
```

4.5 Event Request

The request is queued at the driver until a change in the line state occurs. At that time the request is returned with a result according to the change.

The driver is able to save one event if a change occurs when the user has no event request at the driver. When the driver receives a event request it is returned at once and if more than one event had occurred since last event request the result modification equals one.

```
        request:
                function = 0, function modification = 4

        answer:
                result = 0, no events lost
                       = 1, events are lost

                result modification
                       = 0, line activity after timeout
                       = 1, line activity timeout
                       = 2, disconnect by user
                       = 3, SNRM frame received
                       = 4, DISC frame received
                       = 5, FRMR frame transmitted
                       = 6, transmitter malfunction
                       = 7, DRS timeout
                       = 8, disc by SNRM
                       = 9, CD failure
                       = 10, setup timeout
```

4.6 Return Buffers

The driver will return all read and write requests with result=returned by user or result=transmission attempted. The state of the driver changes to disconnect and finally the Return Buffer request is returned with result=ok.

```
          request:
                  function = 0, function modification = 5

          answer:
                  result = ok, result modification = 0
```

5. I/O REQUESTS

The SDLC driver supports two I/O requests, Read data and Write data. Both requests operate on one frame, i.e. the request is to receive or transmit one frame. The maximum length of a frame is a configuration parameter, which is called 'max_frame' in the following chapters.

5.1 Read Data.

The request will be answered when a frame from the link has been received or the driver receives a Return Buffers request. In case the input data area is too small to hold the received frame, an event with FRMR frame transmitted is returned, and the driver changes to connecting mode, in this case the request is answered with result=not processed and byte_count = 0.

If no line activity is observed in a period defined by the parameter line_activity in the connect request, an event is returned with timeout result.

```
request:
        function = 1, function modification = 0
        data area: Input area

answer :
        result = ok, receiver buffer overflow or not processed
        result modification = Any combination of status bit
        bytecount = Number of bytes read.
```

5.2 Write Data

This is a request to send a block of data on the link as one frame. As the standard frame size supported by the driver is 'max_frame', this is the maximum data area size too.

In case the data area is greater than 'max_frame', an event with disconnect result is returned and the driver changes to disconnect mode.

```
request:
        function = 2, function modification = 0
        data area: Write data

answer :
        result = ok, not processed or transmission attempted
        result modification = All combinations of status bits
```

A. REFERENCES

(ref. 1) IBM GA27-3093-2
IBM Synchronous Data Link Control General Information.

(ref. 2) COM601 Testdriver, Reference Manual.

(ref. 3) The PI-3 Machine