RCSL No: 52-AA1074

Edition: November, 1981

Author: Per Mondrup

Title:

RC3502 COM201 HDLC-Driver Reference Manual

RCSL 42-i1592

A/S REGNECENTRALEN
af 1979

Keywords:

RC3502, Real Time PASCAL, HDLC-driver, X.25 COM201.

Abstract:

The RC3502 HDLC-driver implements a packet switching data transmission in accordance with CCITT revised Recommendation X.25 level 2 lap B.

(30 printed pages).

Printed by A/S Regnecentralen af 1979, Copenhagen

RCSL 42-i1592

INDHOLDSFORTEGNELSE SIDE

1. Functional Description. 1.

1.1 HDLC Protocol. 1.1

The RC3502 HDLC driver implements a packet switching data transmission in accordance with CCITT revised Recommendation X.25 level 2 lapB. (1) The driver supports both the DCE and the DTE selectable by the user process, see section 1.3 .
An HDLC driver incarnation supports one DTE or DCE, hereafter called one line consisting of a receive and a transmit link channel .

1.2 Driver Structure. 1.2

The HDLC driver process consists of three processes:

- An HDLC driver process, which is the main process created by the user and which executes on level 0.
- A receiver interrupt process, created by the main driver process, which executes on the receiver interrupt level.
- A transmitter interrupt process, created by the main driver process, which executes in the transmitter interrupt level.

The HDLC driver is identified to the user by the HDLC request semaphore, which is a process parameter supplied by the user when the driver process incarnation is created. All messages from the user to the HDLC driver is sent to this semaphore.

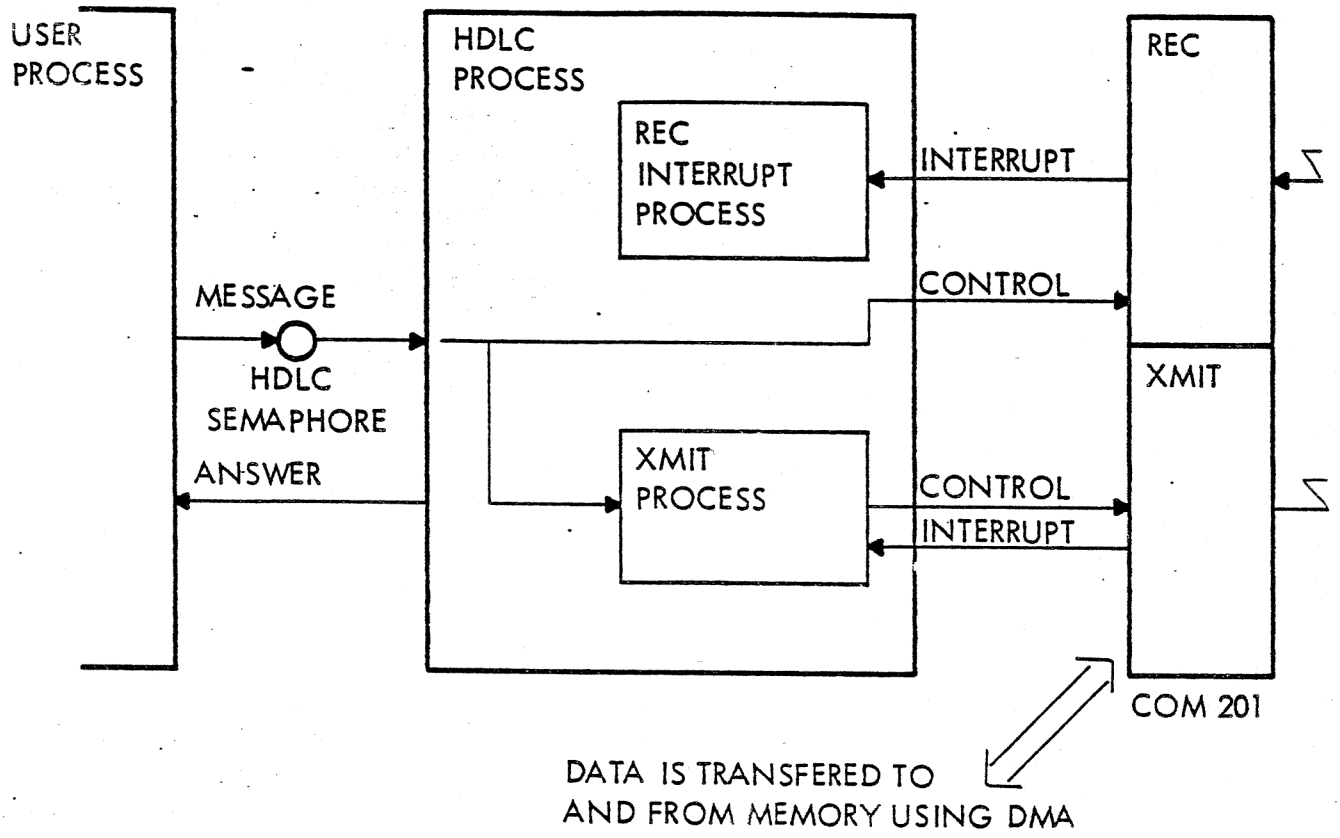

Fig. 1. HDLC Driver Structure.

1.3 Line States.

1.3

At any time a line will be in one of the following states:

- Disconnected, there is no connection with the other end and no attempt is made to obtain a connection.

- Connecting, there is no connection with the other end but attempts are made to obtain a connection.

- Connected, there is a physical and logical connection with the other end.

Initially the line state is disconnected. The line state is changed to connecting (and, when possible, connected) by means of a connect message. A disconnect message will change the line state to disconnected. Detection of a fatal error on the transmission line or protocol logic will also change the line state to disconnected or connecting depending on whether autoconnect has been specified or not.
A connect message may specify autoconnect, which causes the line to enter connecting state as result of a fatal error instead of disconnected state.
A connect message also specifies which role the driver should play in the communication:

- DTE,

- DCE, or

- alternating DTE / DCE until the connection has been completed successfully and the line state connected entered.

1.4 General Description of Driver Functions. 1.4

The HDLC driver maintains a binary transparent point-to-point transport facility. This implies that the driver will not return answers to transput messages until the data transportation to / from the other end has been acknowledged properly or until the user explicitly requests the driver to return unacknowledged message buffers.

A number of control message types enables the user to supervise and control driver functions:

- Line control messages are used to control the line state.

- Buffer control messages are used to control premature returnal of transput messages queued at the driver.

- Modem control messages are used to control the modem signals generated by the COM201 line adaptor.

- Sense messages are used to obtain status information, i.e. incoming modem signals, statistic information. A special sense line speed causes the driver to measure the modem clock frequency by transmitting a number of illegal data packages while measuring the transmission time.

- Event messages are used to report changes in the line state. Event messages are, as the only control messages, queued at the driver until a line state change occurs.

- Testoutput messages are used to control collection of testoutput and to obtain a copy of the collected driver testoutput.

All control messages, except event messages and some testoutput messages, are executed and returned immediately.

Transput messages are executed with a maximum data stack depth of 3.
The first databuffer of an inputmessage must have a capasity of at least 3 bytes, i.e. last-first>=2.

Output messages are executed according to their priority. The priority, however, has significance only until the data has been transmitted once. Input messages are executed sequentially.

The user should make no assumptions on the order of returnal of transput messages.

1.5 Modem Signal Considerations. 1.5

Although The CCITT har recommended certain standards for the utilization of modem signals in X.25, these standards are not always followed at the various installations using X.25 . Therefore the driver considers incoming modem signals during normal traffic (line state connected) as informative only, e.g. Data Carrier Detect going low does not cause the driver to initiate error recovery.
The modem signals are examined at termination of transmission and reception and the statistics updated

accordingly.

Outgoing modem signals are controlled using modem control messages.

In line state connecting, however, the following modem signal protocol is maintained by the driver prior to the logical connection action:

a. Data Terminal Ready and Request To Send is set.

b. Data Set Ready and Clear To Send = high is awaited.

1.6 Testoutput. 1.6

The driver enables collection of testoutput, which consists of a trace of protocol level events for the line.
testoutput is collected in a local area in the driver process. The testoutput can be read using a testoutput message, which may also be used to set a testmask whitch control the collection.
Note however that collection of testoutput will increase the PU-load of the driver slightly.

Each testoutput record contains information about:

- Time (relative in 100 millisec.)
- Address and control byte transmitted or received.
- u1 and u3 of received messages.
- Hardware status information.

2. Driver Interface. 2.

This section describes the interface between the user process and the HDLC driver.

2.1 Driver Process Creation. 2.1

The HDCL driver name (used in LINK call) is

hdlc

The HDLC driver incarnation should be created with following parameters:

(req_sem: semaphore; rec_level: integer)

req_sem is the semaphore to which all messages to the driver should be signalled.

rec_level is the interrupt level for the COM201 receiver; the transmitter level is hardware defined as receiver level + 1.

See section 3 for required stack size.

The driver may be started with any priority, but in order to minimize retransmissions caused by receiver overrun a high coroutine priority is recommended.

2.2 Driver Messages. 2.2

Messages signalled to the driver semaphore (req_sem) should observe the standard for driver interfaces (2,3,4), especially the result byte, u2, should always be set to 7 enabling the driver to distinguish user messages from answers to own messages.

Note that messages signalled to the driver with wrong parameters may cause an exception inside the driver process complex.

When messages are returned from the driver, u3 is always set to level, which for all messages except write data messages is equal to the driver process parameter rec_level. For write data messages u3 is set to rec_level + 1.

The result byte, u2, in messages returned from the driver will have the result set according to standard (2). Where nothing else is mentioned, the result

modification will be zero.

In the description of user field contents for each message to and from means user field contents when the message is signalled to / returned from the driver. A dash, - , means unused contents, and unch. means unchanged.

2.2.1 Control Messages. 2.2.1

2.2.1.1 Sense Status message. 2.2.1

User Fields:

	to	from
u1	0+0	0+0
u2	7	result
u3	-	level
u4	-	unch.

Data buffer: Not used.

Function:

Returns the line status in result (u2) in below format.

Result:

result:= (line_state*8) + (modem_state*16)

line_state: 0: disconnected,
1: connected.

modem_state: +1: DSR is on,
+2: DCD is on,
+4: SQD is on,
+8: CI is on.

2.2.1.2 Connect Message. 2.2.1.

* * * See also the Delay Message (2.2.1.12).

User Fields:

	to	from
u1	0+4	0+4
u2	7	result
u3	-	level
u4	-	unch.

Data buffer to driver:

```
record
  na1, na2, na3: integer;
  mode: packed record
          na4: 0..3;
          no_s_commands,
          no_poll,
          delay_i_frames,
          delay_rr_frames,
          final_alarm,
          auto_connect: boolean;
        end;
  connect_ident,
  t1,
  n2,
  k: integer;
end;
```

Data buffer from driver: Unchanged.

Function:

System parameters for the line are set to the values in the data buffer.
If the current line state is disconnected, it is set to connecting; otherwise the line state is unchanged.

Parameters:

na1, na2, na3, na4: not used.

no_s_commands: s_commands is not transmittet in case of timeout.

no_poll: poll_bit is not transmittet in case of timeout, nither is s-commands.

delay_inf: no i_frame is transmittet after a rnr_frame is received.

delay_rr: rr_frame is not transmittet before 2 input_buffers are present and rnr_frame is transmittet when only 1 input_buffer is left.

final_alarm: cmdr is transmittet if an unsolicited responce with f_bit set to 1 is recieved.

autoconnect: true means that the line will enter connecting state after a fatal error.
false means that it will enter disconnected state after having tried to reset n1 times.

connect_ident: 0 means that the line will play the DTE-role in the X.25 protocol.
1 means the DCE-role.
n>2 means alternating DTE / DCE role with alternation each n*100 mSec.
n<0 is invalid.

t1: System timer t1 (retransmission timer), unit = 100 mSec. See (1) section 2.4.11.1 . t1<=0 is invalid.

n2: Maximum number of transmissions of a package, see (1) section 2.4.11.1 . n2<1 is invalid.

k: Maximum number of outstanding frames, see (1) section 2.4.11.4 . Valid vhen 0<k<8 .

Result:
0: ok.
12: connection inpossible becourse linespeed-

measurement is going on.

2.2.1.3 Disconnect Message. 2.2.1.

User Fields:

	to	from
u1	0+8	0+8
u2	7	result
u3	-	level
u4	-	unch.

Data Buffer: Not used.

Function:

The line state is set to disconnected. The internal variable: autoconnect is cleared.

2.2.1.4 Return All Buffers Message. 2.2.1.

User Fields:

	to	from
u1	0+12	0+12
u2	7	result
u3	-	level
u4	-	unch.

Data Buffer: Not used.

Function:

The line state is set to disconnected. All transput messages and event messages queued at the driver are returned with result: not processed and finally the Return-All-Buffers message is returned with result: ok.

2.2.1.5 Return Unused Buffers Message. 2.2.1.

User Fields:

	to	from
u1	0+16	0+16
u2	7	result
u3	-	level
u4	-	unch.

Data buffer: Not used.

Function:

All transput messages queued at the driver, which have not yet been used (i.e. not yet transmitted / set up for reception) are returned with result: not processed. Finally the Return-Unused-Buffers message is returned with result: ok.

2.2.1.6 Modem Control Message. 2.2.1

User Fields:

	to	from
u1	0+24	0+24
u2	7	result
u3	-	level
u4	-	unch.

Data buffer to driver:

```
record
  na1, na2, na3: integer;
  update_RTS,
  RTS,
  update_DTR,
  DTR:              boolean
end;
```

Data buffer from driver: Unchanged.

Function:

The modem signals are set as follows:

```
if update_RTS then set_RTS(RTS);
if update_DTR then set_DTR(DTR);
```

2.2.1.7 Read Statistics Message. 2.2.1

User Fields:

	to	from
u1	0+28	0+28
u2	7	result
u3	-	level
u4	-	unch.

Data buffer to driver: Not used.

Data buffer from buffer:

```
dpinteger= record
             msb,
             lsb:  integer
           end;

record
  na1, na2, na3: integer;
  received,                    (* D.P. no of rec. infos *)
  transmitted,                 (* D.P. no of xmit. infos *)
  skipped,                     (* D.P. no of err. rec. packages *)
  retransmitted:               (* D.P. no of re-xmit. infos *)
                 dpinteger;
  rec_RNR,                     (* no of rec. RNR *)
  xmit_RNR,                    (* no of xmit. RNR *)
  rec_REJ,                     (* no of rec. REJ *)
  xmit_REJ,                    (* no of xmit. REJ *)
  ack_timeouts,                (* no of timeout retrans. *)
  DSR_off,                     (* no of detected DSR off *)
  DCD_off,                     (* no of detected DCD off *)
  SQD_off,                     (* no of detected SQD off *)
  CI_on:                       (* no of detected CI on *)
                 integer;
  last_FRMR_data: packed record
                    rej_fc_field:    byte;
                    VR:              0..7;
                    responce:        boolean;
                    VS:              0..7;
                    na1:             bit;
                    na2,na3,na4,na5: bit
                    Z, Y, X, W:      boolean;
                  end;
  receiver_overrun,            (* no of receiver overrun *)
  transmit_underrun,           (* no of xmit. underrun *)
  receiver_abort,              (* no of receiver abort *)
  time,                        (* time in units of .1 sec *)
  rate:                        (* bytes/sec in sense linespeed *)
                 integer;
  future_use:    array (1..3) of integer
end; (* statistic record *)
```

Function:

Read statistic information without resetting the counters to zero.

2.2.1.8 Read and Clear Statistics Message. 2.2.1

User Fields:

	to	from
u1	0+32	0+32
u2	7	result
u3	-	level
u4	-	unch.

Data Buffer: As Read Statistics Message above.

Function:

Read statistic information and reset the counters to zero.

2.2.1.9 Sense Line Speed Message. 2.2.1

User Fields:

	to	from
u1	0+36	0+36
u2	7	result
u3	tolerance	level
u4	-	unch.

Data buffer: Not used.

Function:

The modem clock frequency is measured by transmission of a number of dummy blocks and the result (u2) is set accordingly. This message is valid only when the line state is disconnected.
The dummy blocks are transmitted with illegal address byte and terminated with abort.
The execution of this message will take one second.
Note: A high PU-load may cause an inaccurate measurement.

tolerance=rel*16+abs gives the accepted deviation from nominal rate
if cnt= no of byte transmittet in one sec. then a deviation of
$\pm$ (cnt * rel / 100 + abs)
from the nominal byte-rate is accepted.

Result (u2):

2+0	Line state not connected.
0+8	110 bps.
0+16	300 bps.
0+24	600 bps.
0+32	1200 bps.
0+40	2400 bps.
0+48	4800 bps.
0+56	9600 bps.
0+64	19.2 kbps.
0+72	48 kbps.
0+80	64 kbps.
0+88	Not measurable.

2.2.1.10 Event Message. 2.2.1

User Fields:

	to	from
u1	0+40	0+40
u2	7	result
u3	-	level
u4	-	unch.

Data buffer: Not used.

Function:

Event messages are queued at the driver until a line state change occurs. Then it is returned with result (u2) set accordingly.
If no event message is present at the driver when a line state change occurs, the last state change is saved and returned when an event message arrives at the driver.
Note: Only one line state change (i.e. the last) is saved for future returnal.

Result:
result:= cause*8+event_lost*128

```
cause:
 0:  connected.
 1:  disconnected by user.
 2:  DISC_frame received.
 3:  SABM_frame received.
 4:  UA_frame received.
 5:  DM_frame received.
 6:  CMDR_frame received.
 7:  controlfield uninteligible.
 8:  unsolicided response with f-bit.
 9:  size-error.
10:  sequense error.
11:  timeout, driver try to reset.
12:  timeout, driver gives up.
13:  receiver malfunction.
14:  transmitter malfunction.
15:  exception.
```

2.2.1.11 Testoutput Message. 2.2.1

User Fields:

	to	from
u1	0+44	0+44
u2	7	result
u3	tp_control	level
u4	-	unch.

Data buffer to driver: Not used.

Data buffer from driver:

```
record
  first, last:    integer;
  next_cyclic:    integer;
  testbuffer:     array (0..30) of testellement
end;
```

a testelement is 4 words (8 words in case of extended testoutput)

Function:

The function of a testoutput message is controlled by

tp_control = u3 = func+mask*4

as follows: first the mask is saved and used to control whitch ellements will be collected from now

then the action controled by func is performed.

	mask	func
u3:	!5 4 3 2 1 0!	x x!

mask:

bit0: on/off bit: if=0 then no collecton takes place

bit1: collect answers from controler

bit2: collect messages from the aplication

bit3: collect messages to the transmitter

bit4: not used

bit5: collect during fifo sense

func	action
0:	the mask is saved only.
1:	the testoutput buffer is copied into the message buffer.
2:	the message is queued internal. Eatch time the testoutput buffer is filled and this queu is not empty the testoutput buffer is copied to the message buffer from the queue and returned.
3:	all messages in the internal queu is returned with result=1. Then action 1 is performed.

The format of the collected testoutput is described in section 2.3 .
Since the testoutput is collected cyclic, the testoutput buffer will always be filled with 31 testellements and the returned value of last points to the last stored ellement in the cyclic buffer.

2.2.1.12 Delay Message. 2.2.1.

User Fields:

	to	from
u1	0+4	0+4
u2	7	result
u3	-	level
u4	-	unch.

Data buffer to driver:

```
record
  na1, na2: integer; delay: integer;
  mode: packed record
          na4: 0..3;
          no_s_commands,
          no_poll,
          delay_i_frames,
          delay_rr_frames,
          final_alarm,
          auto_connect: boolean;
        end;
  connect_ident,
  t1,
  n2,
  k: integer;
end;
```

Data buffer from driver: Unchanged.

Function:

System parameters for the line are set to the values in the data buffer.
If the current line state is disconnected, it is set to connecting; otherwise the line state is unchanged.

Parameters:

na1, na2, na4: not used.

no_s_commands: s_commands is not transmittet in case of timeout.

no_poll: poll_bit is not transmittet in case of timeout, nither is s-commands.

delay_inf: no i_frame is transmittet after a rnr_frame is received.

delay_rr: rr_frame is not transmittet before 2 input_buffers are present and rnr_frame is transmittet when only 1 input_buffer is left.

final_alarm: cmdr is transmittet if an unsolicited responce with f_bit set to 1 is recieved.

autoconnect: true means that the line will enter connecting state after a fatal error.
false means that it will enter disconnected state after having tried to reset n1 times.

connect_ident: 0 means that the line will play the DTE-role in the X.25 protocol.
1 means the DCE-role.
n>2 means alternating DTE / DCE role with alternation each n*100 mSec.
n<0 is invalid.

t1: System timer t1 (retransmission timer), unit = 100 mSec. See (1) section 2.4.11.1 . t1<=0 is invalid.

n2: Maximum number of transmissions of a package, see (1) section 2.4.11.1 . n2<1 is invalid.

k: Maximum number of outstanding frames, see (1) section 2.4.11.4 . Valid vhen 0<k<8 .

delay: Minimum time in millisec. in witch the transmitter should transmit flags between two frames. Valid if $0 \leq \text{delay} \leq 255$.

Result:
0: ok.
12: connection inpossible becourse linespeed-measurement is going on.

2.2.2 Transput Messages.

2.2.2.1 Receive Message.

User Fields:

	to	from
u1	1	1
u2	7	result
u3	-	level
u4	-	unch.

Data buffer: Used according to standard.

Function:

The receive message is queued at the driver until an errorfree data package has been received and acknoledged or until a Return Buffers control message is signalled to the driver.

Result:

0	The data buffer contains a legal data package.
1	The message returnal is caused by a Return Buffers control message.

2.2.2.2 Transmit Message.

User Fields:

	to	from
u1	2	2
u2	7	result
u3	priority	level
u4	-	unch.

Data buffer: Used according to standard.

Function:

The transmit message is queued at the driver according to the priority (u3) . Priority is a value between 0 and 7 both incl. with increasing value giving increasing priority. The message is returned,

when the data package has been transmitted and acknoledged, or, when a Return Buffers control message has been signalled to the driver.

Result:

0+0	The data has been transmitted and acknoledged succesfully.
1+0	The message returnal is caused by a Return Buffers control message. The data has not been transmitted.
1+8	As result 1 except that the data has been transmitted one or more times.

2.3 Testoutput Formats. 2.3

```
testoutputellement= packed record
  aux,
  kind:     byte;
  time:     integer;
  adr,
  command: byte;
  status:  integer;
  (* extended testoutput *)
    bstate:          0..4;
    rstate,
    xstate:          0..15;
    ystate:          0..2;
    j:               0..7;

    vi,
    tn:              0..7;
*   t:               0..31;
*   mstate:          0..2;
    send,
    sending_i_frame,
    aborting:        boolean;
    rec_pointer,
    xmt_pointer:     array(0..3) of 0..15;
  (* end extended testoutput *)
 end;
```

2.3.1 Normal Testoutput. 2.3.1

Time is relative time in units of 0.1 sec. (modulo 10000).

status is the latest read status from the controler (hexadecimal).

The meaning of aux, adr, command depend of kind as follow´s:

kind	adr	command	aux
0 receiver answer	adr	cmd	alternate 1 2
1 transmitter answer	adr	cmd	0
2 message	u1	RR(0)	u3
3 send to transmitter	adr	cmd	0
4 message(u2<>7)	u2	RR(0)	0
5 cmdr	cause	I(VS,VR)	0
6 cmdr	cause	cmd	0
7 read fifo	flags	0	0
8 exception	excause	DISC*	0

2.3.2 Extended Testoutput. 2.3.2

bstate: inputbuffer state. 0 means no buffers, 4 means many buffers.

rstate: Receiver state. 0..2 means connected, 3,4,6 means connecting, 7,8 means alternating connect, 5,9,10 means disconnectet.

xstate: xmt state. reflects next frame to be send.

ystate: your state: ystate>0 means rnr has been received.

* mstate: my state: 0 means rr is transmittet, 1
* rej is transmittet and 2 rnr is
* transmittet.

j: number of blocks in latest received or transmitted frame.

vi: number of i_frames transmitted but not acknoleged yet.

tn: number of timeouts.

t: timer in units of .2 sec.

send: transmitter busy.

sending_i_frame: transmitter is sending iframe.

aborting: current frame is aborted.

rec_pointer and
xmt_pointer: fifo pointers in controler.

3. Performance and Ressource Requirements. 3.

3.1 performance 3.1

To be supplied later.

3.2 Program and Stacksize. 3.2

Programsize:	8524
Stacksize:	1024
Pools and Children:	724

4. References. 4.

(1) CCITT Draft Revised Recommendation X.25 .
Computer Communication Review, vol. 10, no. 1&2.

(2) Konventioner for drivere til NY og LAMBDA.
UDV-NYSKÆRM.EL.68 (danish).

(3) Konventioner for PASCAL80 driver interfacer.
UDV-PASCAL80.EL.1 (danish).

(4) Addendum til konvensioner for PASCAL80 Driver Interfaces.
UDV-PASCAL80.HLV.12 (danish).

RETURN LETTER

Title: RC3502 COM201 HDLC-Driver Reference Manual

RCSL No.: 52-AA1074

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback, your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability:

_____

_____

_____

_____

Do you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

How can this manual be improved?

_____

_____

_____

_____

Other comments?

_____

_____

_____

_____

_____

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

42-i 1288

..... Fold here .....

..... Do not tear - Fold here and staple .....

Affix
postage
here

REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark