
RCSL No: 31-D616

Edition: September 1980

Author: Ejvind Lynning

Title:

RC3502 VDC201 CIRCUIT Master Driver
Reference Manual

Keywords:

RC3502, VDC201, CIRCUIT, PASCAL80.

Abstract:

This manual gives a complete description of the CIRCUIT master driver for one VDC201 controller on the RC3502. Up to eight secondary stations, typically RC850 terminals, may be multidropped on the CIRCUIT. The communication takes place according to the CIRCUIT protocol.

(20 printed pages)

Copyright © 1982, A/S Regnecentralen af 1979
RC Computer A/S

Printed by A/S Regnecentralen af 1979, Copenhagen

Users of this manual are cautioned that the specifications contained herein are subject to change by RC at any time without prior notice. RC is not responsible for typographical or arithmetic errors which may appear in this manual and shall not be responsible for any damages caused by reliance on any of the materials presented.

<u>TABLE OF CONTENTS</u>	<u>PAGE</u>
1. INTRODUCTION	1
2. FUNCTIONS SUPPORTED BY THE VDC DRIVER	4
2.1 Connection States of a Secondary Station.....	4
2.2 Status Information.....	5
2.3 Data Transfer Functions	6
2.4 Order of Request Processing.....	6
3. DRIVER INTERFACE	8
3.1 Driver Messages	8
3.1.1 Get Status Request	8
3.1.2 Connect Request	8
3.1.3 Disconnect Request	9
3.1.4 Receive Frame Request	9
3.1.5 Transmit Frame Request	10
3.2 Answers	10
4. PARENT PROCESS RESPONSIBILITIES	11
 <u>APPENDIX:</u>	
A. REFERENCES	13

INTRODUCTION

The driver described in this manual runs on an RC3502 (LAMBDA) computer which is master on a CIRCUIT with up to eight multi-dropped secondary stations, cf. Fig. 1. The description is based on concepts and terminology which is introduced in the CIRCUIT protocol [1]. The driver is supported by a VDC201 controller [2]. Typically the secondaries will be PI-1 machines in RC850 terminals.

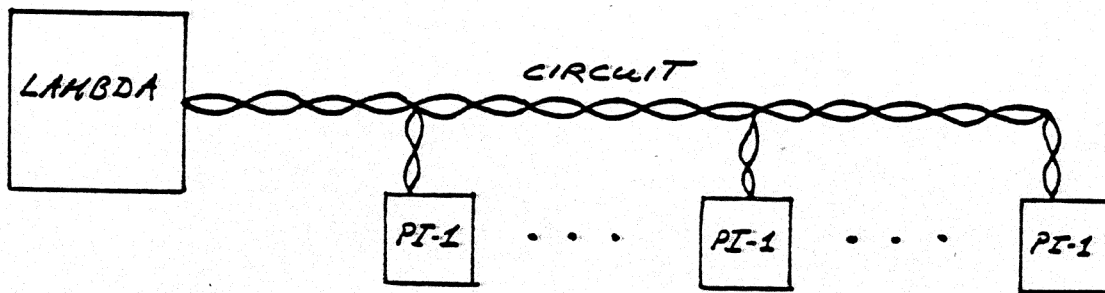


Figure 1.

For each of the secondary stations multidropped on a CIRCUIT, the CIRCUIT wire pair, the VDC201, the Z80-SIO, the RC850 CIRCUIT driver, and the VDC driver in combination form a transport facility which can be used to provide communication between two PASCAL80 process incarnations, one in the PI-1 machine, and one in the LAMBDA. These process incarnations are called users of the transport facility, cf. Fig. 2.

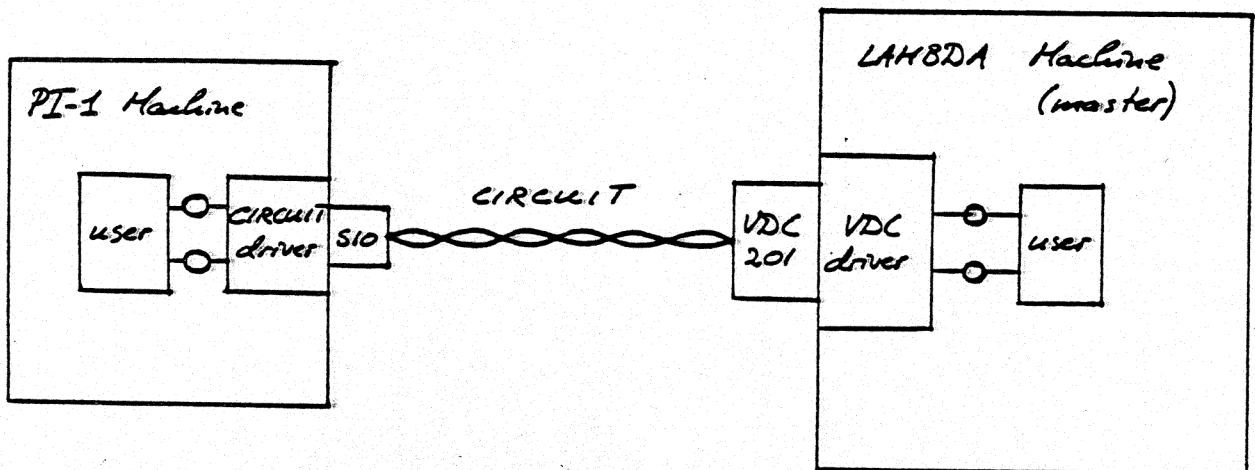


Figure 2.

The transport facility has the following properties:

- Data is transported in frames (blocks). The maximum frame size is agreed between the users and reflected in the size of data buffers passed in requests to the drivers. Every block transport involves a transmission overhead of 6 bytes.
- The transport facility is transparent, i.e. the bit pattern within each frame is delivered precisely as received, and no bit pattern or sub-pattern within a frame causes any special action within the transport facility. In particular the transport facility is completely unaware of any devices, such as display etc., which may be part of the RC850.
- The transport is correct. Any transmission errors recognised by means of FCS [1] are automatically corrected by retransmission. The users are not informed about retransmissions.
- The order (sequence) of frames is maintained through the transport facility.
- The CIRCUIT is a half duplex communication line and there is a master-slave (poll-response) relationship between the two machines [1]. Nevertheless the transport facility appears at the driver interface to provide a balanced full duplex service.
- Along with each actual frame the transport facility carries a 'user bit', which may be used for any purpose. The suggested use of this bit is to indicate when several frames transmitted in sequence comprise a logical unit ("more data" bit).

Communication on the CIRCUIT takes place according to the CIRCUIT protocol [1]. The term 'frame' as used above actually refers to the I-field of a frame in the protocol sense. The responsibility for maintaining correct communication rests with the drivers, assisted by the VDC201 and the Z80-SIO.

The RC850 CIRCUIT driver and the RC3502 VDC driver both behave according to the general conventions for Pascal80 drivers as described in [3]. Detailed information about the RC850 CIRCUIT secondary station driver may be found in [4]. The remainder of this manual describes in detail the functions provided by the RC3502 VDC CIRCUIT master driver and its interface to a user process incarnation.

2. FUNCTIONS SUPPORTED BY THE VDC DRIVER

2.

In general the secondary stations multidropped on a CIRCUIT may be considered separately from the point of view of one or more process incarnations using the VDC driver, as each request to the driver concerns precisely one secondary station.

2.1 Connection State of a Secondary Station

2.1

A secondary station is always in one of three connection states:

- disconnected: the driver makes no attempt to communicate with the station,
- connecting: the driver attempts to establish communication with the station by polling it occasionally (at least once every second),
- connected: communication with the station takes place at short intervals (at least 5 times every second); data blocks may be exchanged.

Initially when the driver is started up all its secondary stations are disconnected. The state of a station is changed from disconnected to connecting following a 'Connect' request. When the station responds to a 'Reset' with 'Answer Reset' while in the connecting state, its state is changed to connected, and the 'Connect' request message is returned. There is no limit to the amount of time a station may spend in the connecting state.

A station may be disconnected (state changed to disconnected) either because a 'Disconnect' request is received or because an unrecoverable error occurs during communication with the station.

The state of a secondary station can only change in the ways described above.

Status information about a secondary station may be obtained by issuing a 'Get Status' request. If such a request is answered with result 'successful' the data buffer contains a status record according to the type definition:

```
status type=RECORD
    state:  disconnected,connecting,connected;
    error:  0..15;
    unsol_SR:BOOLEAN
END;
```

where state is the connection state of the secondary. If state is disconnected, error indicates the reason why the station was disconnected, as follows:

no error:

0 'Disconnect' request or station has not yet been connected.

protocol error:

- 1 The station sent 'Answer Reset' during normal communication.
- 2 The station sent 'Reset' indicating it wants to be reset.
- 8 Time-out, i.e. no response from the station within 10 ms.
- 9 Framing or FCS error.
- 12 I-field received when master's RR was 0.
- 13 Received I-field was too long for buffer (of 'Receive Frame' request).
- 14 Sequence number error.

other errors:

- 10,11 Underrun/overrun, controller cannot get access to bus in time.
- 15 Controller cannot get access to line for 1 sec., i.e. carrier signal persists.

In case of the errors 8, 9, 10, and 11, five (5) retries are made before the driver (controller) gives up and disconnects the station.

For a connected station `unsol_SR`, when `TRUE`, indicates that the secondary has an (unsolicited) transmit buffer (`SR=1`), while the driver has no receive buffer (no 'Receive Frame' request).

2.3 Data Transfer Functions

2.3

Two kinds of data transfer requests are serviced:

- 'Transmit Frame', and
- 'Receive Frame'.

A data transfer request must always contain a data buffer formatted in the standard way [3]. A formatting error causes an exception in the driver. The data is transmitted or received as the I-field of a frame, and each request is answered with result 'successful' as soon as the data has been transmitted or received.

See section 3.1 for the coding of data transfer requests and answers.

2.4 Order of Request Processing

2.4

Requests are answered in the order processing is completed. This is not necessarily the order in which the driver receives them, indeed most likely not so.

A 'Get Status' or 'Disconnect' request will be processed and answered as soon as it is received, whereas a data transfer request is merely inserted in an internal send or receive queue for the concerned secondary, where it will wait until communication progresses to the point where it is completed. There is no upper limit to the amount of time this may take.

For each secondary 'Transmit Frame' requests are always processed exactly in the order they are received by the driver. The same is true for 'Receive Frame' requests; more importantly, the I-fields received on the CIRCUIT are passed as answers to 'Receive Frame' requests in the order they arrive.

The driver attempts to handle transmit requests as fast as possible, thus if there is a transmit buffer available and the relevant secondary is willing to receive it, the driver will transmit the buffer rather than poll other secondaries. When transmissions are ready for several secondaries simultaneously scheduling is cyclic.

3. DRIVER INTERFACE

3.

The following description is based on the driver interface conventions [3].

In all driver messages to the VDC driver u3 holds the number of the secondary station which the request concerns. The value of u3 must be in the range 0-7.

In data transfer requests and answers the function and result modification fields of u1 and u2 are only used to hold the 'user bit' (in 'Transmit Frame' requests and 'Receive Frame' request answers).

3.1 Driver Messages

3.1

3.1.1 Get Status Request

3.1.1

coding of u1:

```
basic function=      0
function modification= 0
```

The request is processed in all connection states. The result will be 'successful' and the data buffer will contain status information as described in section 2.2.

3.1.2 Connect Request

3.1.2

coding of u1:

```
basic function=      0
function modification= 1
```

The request is only legal in the disconnected state. It is answered with result 'successful' when the station has been reset, in which case its state changes to connected. The request may be

cancelled by a 'Disconnect' request, in which case it is answered with result 'not processed'.

3.1.3 Disconnect Request

3.1.3

coding of ul:

```

basic function=          0
function modification=  2

```

The request is illegal in the disconnected state. Otherwise it may be used to disconnect a connected station or to cancel a pending 'Connect' request. In either case it is answered with result 'successful'.

3.1.4 Receive Frame Request

3.1.4

coding of ul:

```

basic function=          1
function modification=  0

```

The data buffer of the driver message is used as a receive buffer for the I-field of a frame. If a frame with an I-field longer than the buffer (more than last-first+1 characters) is received, the station is disconnected (error 13, see section 2.2). When a frame with non-empty I-field has been successfully received in the buffer the request is answered with result 'successful' and the next-field indicating the size of the received frame (I-field). Bit 4 of the result byte (result modification) is equal to the U-bit of the C-field of the received frame [1].

Any number of 'Receive Frame' requests (up to 127) may be queued (pending) within the driver.

3.1.5 Transmit Frame Request

3.1.5

coding:

basic function= 2
 function modification= 'user bit' (bit 3 of u1)

The data buffer of the driver message is transmitted as the I-field of a frame as soon as possible (see section 2.4). The U-bit of the C-field of the frame will be equal to the specified 'user bit'. When the frame has been transmitted the request is answered with result 'successful'.

Any number of 'Transmit Frame' requests (up to 127) may be queued (pending) within the driver.

3.2 Answers

3.2

Answers with result 'successful' (0) are described in section 3.1.

If any data transfer requests are pending when a station is disconnected because of an error, one of these is answered with result 'persistent error' (3). The remaining pending requests and all data transfer requests received by the driver while the station is not connected are answered with result 'not processed' (1).

'Connect' requests for already connected stations and unintelligible driver messages (unknown function specification) are answered with result 'illegal' (4).

The result 'transient error' (2) is not used.

PARENT PROCESS RESPONSIBILITIES

4.

An application running on the RC3502 which uses one or more VDC201 controllers must include a parent process for the VDC driver. The parent process must contain a process declaration for the driver, as follows:

```
PROCESS VDC_driver(VAR VDC_sem: SEMAPHORE; level: INTEGER);  
EXTERNAL;
```

The semaphore parameters is the driver's request semaphore and level is the lower of the two interrupt levels to which the controller is attached.

One incarnation of the driver must be created for each VDC controller. The driver should run with a very high priority.

REFERENCES

A.

- [1]. CIRCUIT Protocol, Reference Manual;
RCSL 31-D 598.
- [2]. VDC201 Controller, Reference Manual;
RCSL 31-D 599.
- [3]. PASCAL80 Driver Conventions;
RCSL 31-D 617.
- [4]. RC850 CIRCUIT Secondary Station Driver, Reference Manual;
RCSL 31-D 603.

LÆSERBEMÆRKNINGER

Titel: RC3502 VDC201 CIRCUIT Master Driver RCSL Nr.: 31-D616
Reference Manual

A/S Regnecentralen af 1979 bestræber sig på at forbedre kvalitet og brugbarhed af sine publikationer. For at opnå dette ønskes læserens kritiske vurdering af denne publikation.

Kommenter venligst manualens fuldstændighed, nøjagtighed, disposition, anvendelighed og læsbarhed:

Angiv fundne fejl (reference til sidenummer):

Hvordan kan manualen forbedres:

Andre kommentarer:

Navn: _____ Stilling: _____

Firma: _____

Adresse: _____


Dato: _____

På forhånd tak!

..... Fold her

..... Riv ikke - Fold her og hæft

Frankeres
som
brev

 **REGNECENTRALEN**
af 1979
Informationsafdelingen
Lautrupbjerg 1
2750 Ballerup