RCSL No: 52-AA1116

Edition: September, 1982

Author: Jens Kristian Kjærgård

Title:

RC3502 IMS Driver Reference Manual

RCSL 42-i1905

A/S REGNECENTRALEN
af 1979

Keywords:

RC3502, Real Time Pascal, Asynchronous Multiplexer, Intelligent Slave, Driver.

Abstract:

This document describes the RC3502 IMS Driver. The differenct functions and the specific formats are described as well as the necessary process environment.

(38 printed pages).

Printed by A/S Regnecentralen af 1979, Copenhagen

RCSL 42-i1905

TABLE OF CONTENTS PAGE

1. INTRODUCTION 1.

This manual describes how to communicate with the RC3542 8 channel asynchronous multiplexer from the RC3502 processor. In the following the RC3542 is designated by its technical name as the IMS (Intelligent Multiport Serial Communication Controller). The communication is performed through the IMS-driver (a RC3502 process), which is the main issue of this manual.

The manual follows the RC standard for driver manuals.

Chapter 2 describes the different functions of the IMS-driver. Chapter 3 describes the specific formats of driver messages. Chapter 4 describes how to declare and create an IMS-driver.

Appendix A describes the differences between the IMS and LAM driver (ref. 2).

1.1 Summary of IMS Functions 1.1

The RC3502 IMS driver supports:

- 8 full duplex asynchronous V.24 lines.
- Individual, software selectable line characteristics of:
  - baud rates from 110 bps to 19,200 bps
  - a datasize of 5, 6, 7, or 8 databits
  - four modes of parity
  - 1, 1 1/2, or 2 stopbits
- A general conversion mechanism capable of classification and conversion of received characters.
- Input continued.
- Timeout supervision
- Modem signal supervision

- XON/XOFF flowcontrol

- ALC protocol

1.2 Functional Overview 1.2

The IMS is operating as an intelligent slave on the RC3502 backplane bus based on the concept of a dual port RAM. Furthermore, the RC3502 and the IMS is able to interrupt each other. The hardware is further described in ref. 1 .

Upon these basic hardware facilities a protocol is built, which allows blocks of characters to be transferred to and from the eight lines independently.

The IMS driver is responsible:

- for the observance of this protocol.

- for the partitioning of input and output messages into buffers of limited size.

- for supplying the IMS with line characteristics, conversion and classification tables received through control messages from application programs.

2. FUNCTIONS SUPPLIED BY THE IMS DRIVER 2.

An application program communicates with the IMS driver by sending messages to the IMS driver semaphore and receiving the returned messages at the answer semaphore.

Characters are received/transmitted on one of the 8 IMS lines according to the characteristics of the line, specified in a line status record. (See section 3.2).

Characters in messages are always treated as 8 bit bytes. Superfluous bits (i.e. a dataformat containing less than 8 databits) are masked off when transmitted, and zeroed when received.

2.1 Transmitting 2.1

Transmitting characters work as illustrated in figure 1 and described below:

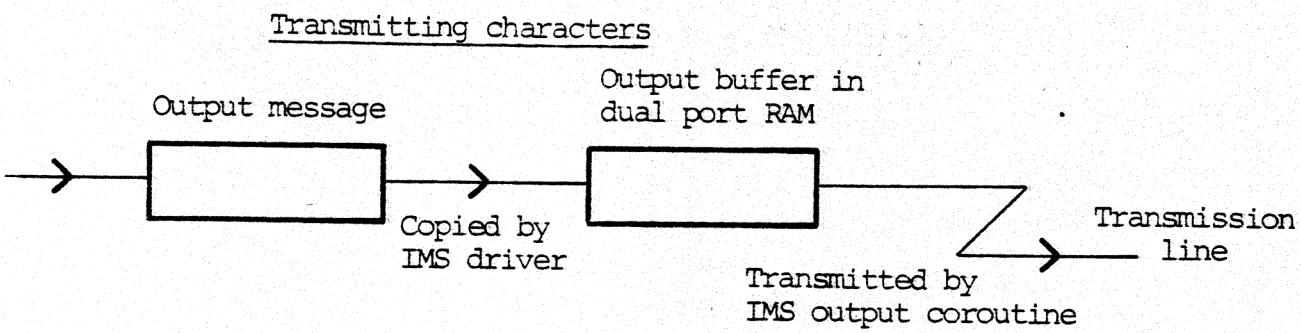

Fig. 1.

1. The output message is, if necessary, partitioned by the IMS driver and each part copied to the output buffer.

2. The IMS is notified through an interrupt and the output coroutine then transmits the characters on the line through a UART.

2.2 Receiving 2.2

Receiving characters work as illustrated in figure 2 and described below:

Receiving characters

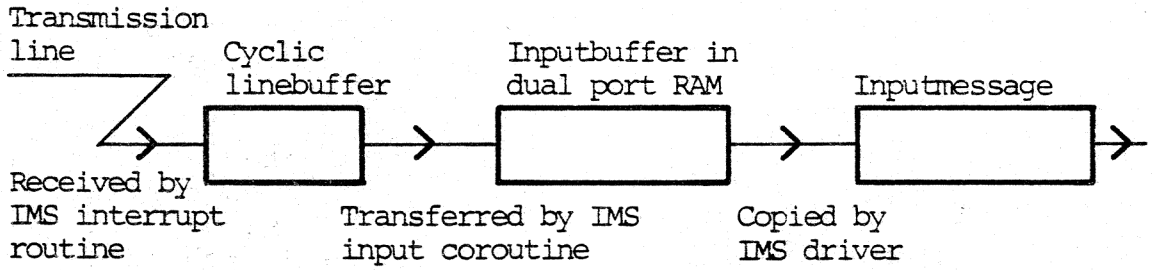

Fig. 2.

1. The character is received by an UART, which interrupts the IMS CPU.

2. The interrupt routine stores the character in a cyclic line buffer and signals on input coroutine.

3. When an input or attention operation is present the input coroutine classifies, converts, echoes, and stores the character in the input buffer for the line according to the contents of a classification and conversion table. This conversion and classification mechanism is described further in the next section.

   An input operation can be specified as continuing which means that characters assembled in the line buffer are included in the input buffer. If an operation has not been specified as continuing the line buffer is emptied and only characters received afterwards are handled.

4. If a character is classified as termination or attention the input request is terminated and the IMS driver notified.

5. The IMS driver copies the input buffer into the input message. Several input buffers can be assembled by the IMS driver into one input message.

If echoing or transmission of flow control characters (see section 2.5) are to be performed and the output line is busy executing an output operation no echoing is performed but result echo-error is set in the answer. The same applies to a timeout error during output of echo characters.

A character received with parity or stop bit error is treated as the character SUB (dec: 26) and result parity is set in the answer.

A received break signal is treated as a character with conversion index -1 when conversion is defined. If conversion is not defined, a break signal is treated as a character with parity error and a SUB-character is stored.

2.3 Classification and Conversion 2.3

When characters are moved from the line buffer to the input buffer, the character can be converted according to the contents of an application supplied conversion table.

The following actions are taken based on the class- ification:

normal_conv: If set only a single character is echoed.

If not set the character is echoed according to a special conversion record, allowing multible echoes, i.e. NL can be echoed as CR, NL.

attention: Terminates input or attention operation, and sets attention result in the answer. Furthermore, an output operation in progress on the line is terminated with result attention.

termination: Terminates input operation.

blind: The character is not stored in the input buffer.

noecho: The character is not echoed.

erase_last: If the current input_buffer is empty the character is not echoed. If the current input buffer is non-empty, the last stored character in the input buffer is deleted, and the erasing character echoed as usual.

Storing of the erasing character is in both cases determined by the blind class- ification.

erase_all: All characters in the current input_buffer are erased. Echoing and storing of the erasing character is determined as usual.

mark: Sets status mark in the result, if the character is stored (is not classified as blind).

If conversion is specified the following steps are taken, when a character is transferred between the line buffer and the input buffer.

1. The received character is converted and classified using the character as index in the conversion table.

2. The character is echoed, if both the input operation is specified as echo and the character is classified as echo. If the character is classified as normal_conv the converted character is echoed else the characters specified in special conversion record is echoed.

3. The converted character is stored in the input buffer according to the blind erase_last and erase_all classification.

4. Operations according to the attention, termination, and mark classifications are performed.

If no conversion is specified the received character is echoed according to the echo specification in the input request and stored in the input_buffer.

2.4 Time Supervision 2.4

Time supervision is performed on a per line basis.

Each line has two input timers and one output timer.

itimer1 specifies the timeout period for the first input character.

itimer2 specifies the timeout period for the following input characters in a buffer.

otimer specifies the timeout period for a character output. The otimer is used as timeout period for an echo character, too.

A timer value of x indicates that at least x-1 and at most x time_units are allowed for the operation.

All timers are decremented after the duration of a timeunit. When a timer is decremented from one to zero a timeout occurs. An initial timer value of 0 is interpreted as an infinite timeout. An initial timer value of 1 is without meaning.

The initial time values are set by the operation set_timers.

A time_unit for the IMS is approximately 100 ms = 0.1 s.

A set_timer operation does not change the timeout period for the the character being input and/or output. I.e. if an infinite timeout has been specified, and no characters are received, a set_timer operation will not give a timeout.

2.5 Flow Control 2.5

The IMS-driver includes facilities for flow control by means of XON/XOFF.

Flow control can be specified in two ways:

- As a part of the line status.

- As a specification in an input operation.

The first way is used to control transmission of characters. The second way to control the receipt of characters.

2.5.1 Flow control on output 2.5.1

As a part of the line characteristic it is specified whether the line is in flow control or not. If the line is in flow control mode receiving an XOFF (dec : 19) character will stop further transmitting on the line until an XON (dec : 17) character is received.

The current state of the transmitting line (either able to transmit or temporarily suspended because of receipt of an XOFF character) can be obtained through reading the line characteristics (see section 3.2).

Echoing of characters is not affected by the flow control state.

After an output timeout error the state of the output line is set to 'transmission enabled'.

2.5.2 Flow control on input 2.5.2

If an input message has flow control specification, an XON character is transmitted on the line, when the input operation is started.

The receipt of an XON will start transmission at the opposite end of the line (if flow control has been specified). At termination of the input message an XOFF character is sent stopping further transmission.

Transmission of XON and XOFF characters in conjunction with flow controlled input is treated as an echo, and thus the transmission is independent of the flow control state on the output line. A flow controlled input message will normally be specified as continuing to allow characters received between termination of the input message and the recognition of the XOFF character at the opposite end, to be included in the next input message.

2.6 ALC-protocol 2.6

The IMS driver supports the ALC/FD protocol as specified in ref. 3 .

The ALC/FD protocol is characterized by a block format and a character format.

The character format must be set through specifying an appropriate line_status.

The block format is supported by the IMS as follows:

Output:

- <STX> is transmitted.
- The contents of the output message (specified by first, last, next as usual) is transmitted assuming that it has the format of <OPK><BBL><INFO>. CHS is calculated.
- <ETX><CHS> is tranmitted.

Input:

- The input line is scanned for an <STX> character. If other characters than <STX> is received result mark is set (no error result), but the operation continues.
- <OPK> is received and the length is determined possibly by receiving the succeeding <BBL> character.
- <INFO> is received, and the checksum calculated.
- The next character is received, and unless it is an <ETX>, result format is set.
- <CHS> is received and if it does not correspond to the calculated checksum result format is set.
- <OPK><BBL><INFO> is returned in the input message with first, last, and next updated as usual.

No conversion or classification is performed on ALC messages.

Only specification of continuing input is allowed in conjunction with ALC input.

2.7 Caution Due to Partitioning of Messages 2.7

Although the partitioning of messages into fixed sized buffers should be transparent it may be observed in the following situations:

- If both an input and an attention operation is present. Partitioning of the input request can result in skipping characters between each part due to the presence of the attention operation.

- If an attention operation is terminated between two parts of an output operation the output operation will not be terminated.

- If a flow controlled input message is partitioned, flow control characters will be transmitted for each part of the input message.

- If a character has classification erase_all only those characters assembled in the current part of the input message are erased.

3. IMS DRIVER INTERFACE 3.

3.1 Messages and Answers 3.1

All messages are signalled to the IMS driver general semaphore and all answers are returned to the answer semaphore except when the driver is removed or enters an exception.

The general format for messages is:

	message	answer
u1	function	function
u2	not used	result
u3	lineno	lineno
u4	not used	unchanged

All message data types starts with first, last, next: integer in accordance with the standard driver conventions (ref. 4).

The driver supports message data stacks of a maximal depth of 1, i.e. unstacked (chained) buffers.

The 8 IMS lines are numbered 0 - 7, and lineno must be within this range.

The following results are defined:

Basic Results:

0 request executed successfully.

1 request not processed but returned by a reset operation.

2 error described by result modification.

4 unintelligible request.

Result modification:

+ 0 timeout	The request is terminated, because of a timeout.
+ 8 echo_error	An echo_error has occured, operation continues.
+16 attention	An attention character is received.
+32 format	Parity- stopbit- or ALC protocol error. The operation continues.
+64 overrun	Characters lost because of hardware overrun or internal buffer overrun. The operation termintes.
+128 mark	The operation continues.

In case of a fatal error in the IMS all messages are returned with the result field set to 255, and in this particular case the lineno field is changed to contain a value of an internal IMS error. This value will be an illegal line number.

The IMS driver will try to recover from a fatal error.

3.2 Line status 3.2

The characteristics and status of a transmission line is communicated through a line_status record:

Line status record

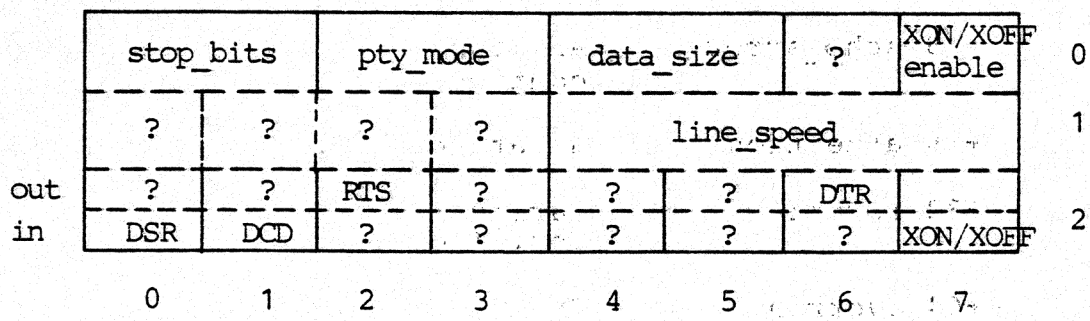

Fig. 3.

```
line_status = packed record

  (* first byte *)

  stop_bits:    (in-
                valid,stop_bit_1,stop_bit_1_5,stop_bit_2);

  pty_mode:     (ig-
                nore_parity,odd_parity,no_parity,even_parity
                );
                (* ignore parity means:
                Parity bits received are masked off and
                ignored. Characters transmitted has even
                parity supplied.
                No parity means: Characters are received
                and transmitted without parity bit *).

  data_size:    (bits_5,bits_6,bits_7,bits_8);

  ?:            boolean;

  XON_enable:   boolean;
                (* enable flow control *)

                (* second byte *)
```

```
    ?,?,?,?:        boolean;

    line_speed:     (b_50,b_75,b_110,b_134__5,b_150,b_300,
                     b_600,b_1200,b_1800,b_2000,b_2400,b_3600,
                     b_4800,b_7200,b_9600,b_19200);

    (* third byte *)

    DSR,            (* DataSetReady incoming modem signal *)

    DCD,            (* DataCarrierDetect incoming modem  signal
                    *)

    RTS,            (* ReadyToSend outgoing modem signal *)

    ?,?,?,

    DTR,            (* DataTerminalReady  outgoing modem signal
                    *)

    XON_state:      boolean; (* Current output state *)

end; (* line_status *)
```

3.3 Conversion Table 3.3

A conversion table is formatted as below:

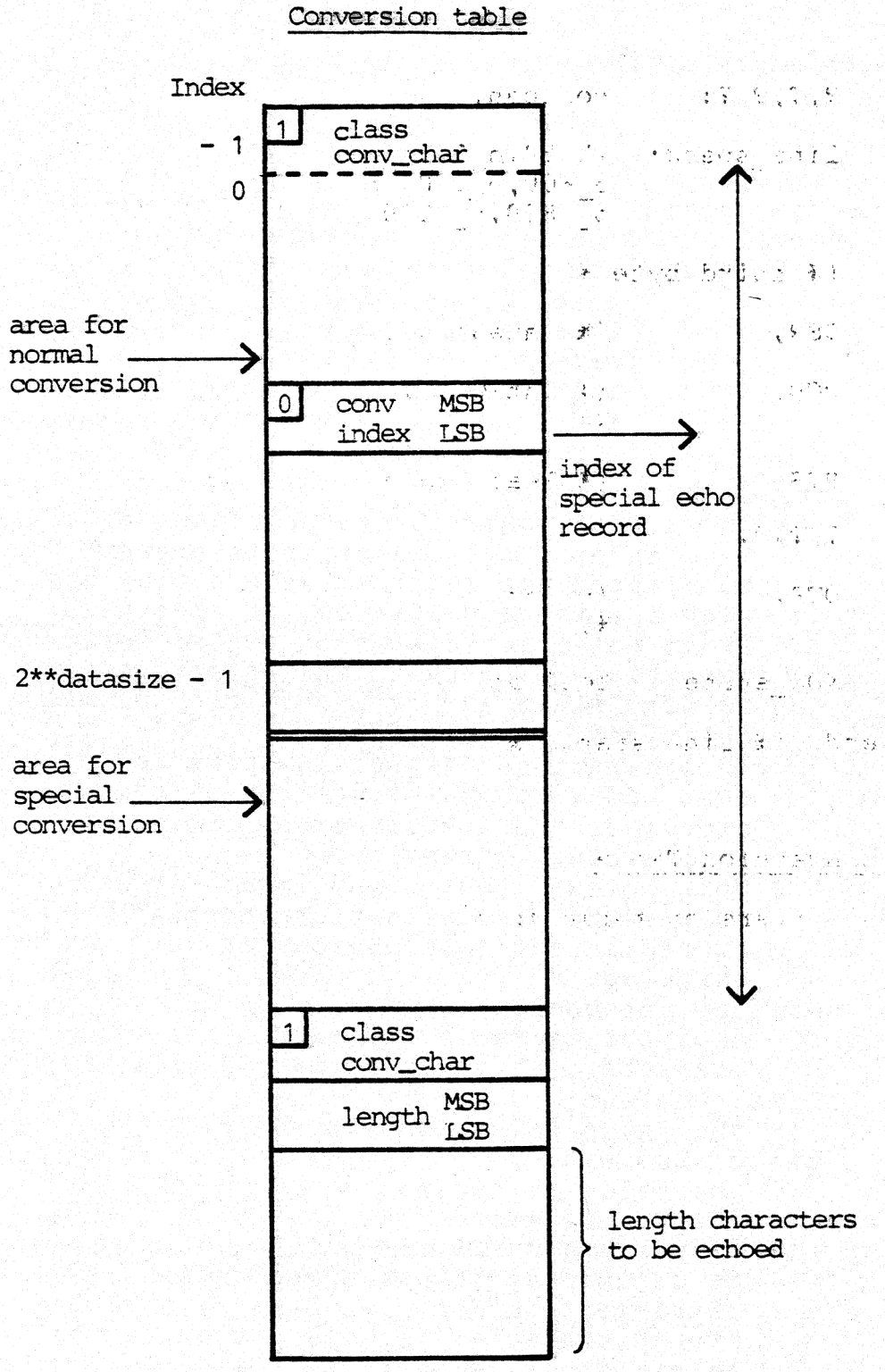

Fig. 4. Conversion Table

3.4.2 Line Control 3.4.2

Format:

```
      message   answer
u1    4         4
u2    -         result
u3    lineno    lineno
u4    -         unchanged
buf   line_control
line_status

buffer_type =
  record
    f, l, n: integer; (* not used *)
    new_line_state,
    actual_changes: line_status;
```

Function:

Fields with a non-zero ordinary value in actual_changes are updated with the value supplied in new_line_state. The resulting current line_status is returned in new_line_state.

3.4.3 Set Conversion 3.4.3

Format:

```
      message   answer
u1    8         8
u2    -         result
u3    lineno    lineno
u4    -         unchanged
buf   conv_spec conv_spec

conv_spec =
  record
    f, l,  n: integer; (* not used *)
    conv_control: integer;
    conv_tab: conversion_table;
  end;
```

Sets conversion and classification table for the specified line.

The value of conv_control is interpreted as follows:

-2: Clear current conversion table, i.e. incoming characters are neither classified nor converted. Conv_tab is not used and can be omitted.

-1: Use conv_tab as new conversion table.

0..7: Use the same conversion table as conv_control. Conv_tab is not used and can be omitted.

Conversion will be temporarily cleared during the setconversion operating and subecho and will be stopped, affecting input and attention operations in progress on the line.

The set conversion message is returned immediately.

3.4.4 Set Timers 3.4.4

Format:

	message	answer
u1	12	12
u2	-	result
u3	lineno	lineno
u4	-	unchanged
buf	timers	timers

```
timers =
  record
    f, l, n: integer; (* not used *)
    itimer1: integer;
      (* defines the timeout before first character
         input to a buffer *)
    itimer2: integer;
      (* defines the timeout between chars. input *)
    otimer1: integer;
      (* defines the timeout per character output
         from a buffer or echo *)
   end;
```

All timers are in units of 100 ms = 0.1 second and a zero value means no timeout.

Note, however, that the value n means a period between n-1 and n seconds, e.g.: the value 1 is senseless.

Function:

The timer values are set and the answer returned.

3.4.5 Reset

Format:

	message	answer
u1	16	16
u2	-	result
u3	lineno	lineno
u4	-	unchanged
buf	-	unchanged

Function:

All operations in progress on the line are terminated and all messages are returned with status: reset (= not processed). A modem_attention request is not returned.

Likewise all messages received for the line during the reset operation until the reset message is returned, are returned with result reset.

3.4.6 Modem Attention

Format:

	message	answer
u1	20	20
u2	-	result
u3	lineno	lineno
u4	-	unchanged
buf	-	unchanged

Function:

A change in the incoming modem_signals at a line will be recorded by a returnal of a Modem Attention message, if present. Note that the Modem Attention as opposed to all other control messages is not returned immediately. Likewise a modem_attention request is not returned by the usual reset operation. Result attention will be set by returnal of the message. message.

3.4.7 Reset Modem Attention

3.4.7

Format:

	message	answer
u1	24	24
u2	-	result
u3	lineno	lineno
u4	unchanged	unchanged

Function:

Returns a modem_attention request.

3.5 Output

3.5

Format:

	message	answer
u1	outfunc	outfunc
u2	-	result
u3	lineno	lineno
u4	-	unchanged
buf	data	unchanged

Function:

The data in the buffer is output and the answer returned.

outfunc 2: Basic output function.

+32: ALC output function.
An ALC message is output, i.e.

<STX> <OPK><BBL><INFO> <ETX><CHS>

The framed contents must be supplied in the output message. <STX>, <ETX><CHS> are supplied by the IMS. (For detailed specification refer to ref. 3).

If the line_status (see section 3.2) has XON/XOFF specification, receiving an XOFF (dec : 19) character suspend further output until an XON (dec : 17) character is received. However, the timeout control continues.

Output can be terminated prematurely because of either receipt of an attention character, a timeout, or a reset operation. Next is updated according to the amount of data transmitted.

result	interpretation
timeout	a timeout has occured
reset	a reset operation has been issued
attention	the output operation has been terminated prematurely due to receipt of an attention character.

3.6 Input

Format:

	message	answer
u1	inpfunc	inpfunc
u2	-	result
u3	lineno	lineno
u4	-	unchanged
buf	-	rec-data

Function:

The value of inpfunc in u1 defines the actual input function for the message and is composed of a sum of binary numbers, each defining a specific characteristic of the input function:

inpfunc 1: basic input function.

inpfunc +2: echoing of characters according to the specified classification and conversion is performed.

inpfunc +4: continuing input, i.e. characters received from returnal of the previous input operation and onto initiation of this input message are accepted instead of skipped.

inpfunc +16: flow control by means of XON/XOFF characters are to be performed. The following action is taken:

when the driver initiates execution of a flow control input message an XON-character (dec : 17) is output.

when the driver terminates execution of a flow control input message an XOFF-character (dec : 19) is output.

inpfunc +32: ALC protocol input.

<STX> <OPL><BBL><INFO> <ETX><CHS>

is received, the format of the message is checked, <CHS> is calculated and checked, and the framed contents returned.

The input operation is terminated, when one of the following events occurs:

- a character classified as termination is received.
- a character classified as attention is received.
- a timeout occurs.
- a reset operation is issued.
- overrun or character lost. Internal buffer overrun or hardware overrun.
- a complete ALC message is received (in case of ALC input). Upon termination of an input request next is updated.

Result	interpretation:
timeout	one of the two input timers is decremented from one to zero.

4. IMS-DRIVER PROCESS

The IMS driver process is defined as:

```
PROCESS ims_driver (var imssem: semaphore;
                    imslevel: integer;
                    imsmodule: integer;
                    ims16K: 0..3);
```

imssem — is the main semaphore to which all messages are sent.

imslevel — is the interrupt level of the IMS as strapped on the IMS board.

imsmodule — is the module number as selected on the IMS board. The IMS will usually be strapped to the external address space of the RC3502, in which case it will not conflict with RAM modules.

ims16K — an integer in the range 0..3 indicating which quarter of the 64 K bytes module is used for this IMS. This must correspond to the selection on the IMS board. Up to four IMSes can in this way have the same module number.

IMS-driver stack size: 600 words

IMS-driver code size: 5000 bytes

Recommended IMS-driver priority : 0

A. DIFFERENCES BETWEEN IMS AND LAM DRIVER A.

The format of the line status record is changed.

Conversion tables are returned immediately by the IMS driver.

The timers are for the IMS in units of 0.1 s = 100 ms, while they are in units of 1 s for the LAM.

The IMS supports the following additional functions:

- flow control on output
- ALC protocol
- Modem Attention

B. REFERENCES

B.

1 RCSL No. 52-AA1038.
Reference Manual for the IMS208.
Communication controller to RC3502.
March 1981, Knud E. Hansen.

2 RCSL No. 43-GL11703.
RC3502 LAM108/116 Driver.
Reference Manual.
November 1981, Harald Villemoes.

3 ALARMSYSTEM
AL.RC.22/3
Tilslutning af vagtcentraler.
December 1981, Ole Ejby Reinau, Joes Clausen.

4 RCSL No. 31-D617.
PASCAL80 Driver conventions.
October 1980, Ejvind Lynning.

5 RCSL No. 52-AA1119
RC3542 8 channel Asynchronous Multiplexor.
General Information.
May 1982, Harald Villemoes.

6 RCSL No. 52-AA1092
Technical Manual for IMS 208, Rev. 0.

C. CONVERSION TABLE - AN EXAMPLE

The conversion mechanism is illustrated by the following example:

```
conv_integer = PACKED RECORD
                 !  normal_conv  : boolean;
                 !  attention    : boolean;
                 !  termination  : boolean;
                 !  blind        : boolean;
                 !  noecho       : boolean;
                 !  erase_last   : boolean;
                 !  erase_all    : boolean;
                 !  mark         : boolean;
                 !  conv_char    : byte
                 END;

spec_conv_record = PACKED RECORD
                     !  spec_conv_integer : conv_integer;
                     !  length            : integer;
                     !  spec_echo         : ARRAY (1..4) OF byte
                     END;

conversion_table = ARRAY( -1..127 ) OF conv_integer;
special_table    = ARRAY(  1..4   ) OF spec_conv_record;


conv_buf_type =
RECORD
!  f, l, n : integer;
!  conv_cont : integer;
!  conv_tab : conversion_table;
!  spec_tab : special_table
END;
```

The following routine initializes the conversion buffer referenced by m. The example initializes a table to convert a 7-bit character.

The character esc (27) will be classified as attention and echoed as CR(13)LF(10).

The character CR(13) will be echoed as CR(13)LF(10) and CR(13) will be delivered in the buffer and input will be terminated.

The character bs(8) is echoed as bs(8), sp(32), bs(8). The preceding character is erased from the buffer, and bs is not delivered, i.e. the preceding character will be removed both on a terminal screen and in the buffer.

```
PROCEDURE init_conv_table( VAR m: reference );
VAR i : integer;
CONST
  t = true;
  f = false;
  dummy    = conv_integer( t, t, f, f, f, f, f, f,  0 );
  cr_index = conv_integer( f, f, f, f, f, f, f, f,128 );
  bs_index = conv_integer( f, f, f, f, f, f, f, f,132 );
  at_index = conv_integer( f, f, f, f, f, f, f, f,136 );
  cr_conv  = conv_integer( t, f, t, f, f, f, f, f, 13 );
  bs_conv  = conv_integer( t, f, f, t, f, t, f, f,  8 );
  at_conv  = conv_integer( t, t, t, f, f, f, f, f, 13 );
BEGIN
! LOCK m AS buf: conv_buf_type DO
!   WITH buf DO
!     BEGIN
!       ! conv_cont:= -1;
!       ! FOR i:= -1 TO 127 DO
!       !   BEGIN
!       !     ! conv_tab(i):= dummy;
!       !     ! conv_tab(i).conv_char:= i AND 127;
!       !     END;
!       !
!       ! conv_tab(27):= at_index;
!       ! conv_tab(13):= cr_index;
!       ! conv_tab( 8):= bs_index;
!       ! spec_tab( 1).spec_conv_integer:= cr_conv;
!       ! spec_tab( 1).length := 2;
!       ! spec_tab( 1).spec_echo(1):= 13;
!       ! spec_tab( 1).spec_echo(2):= 10;
!       ! spec_tab( 2).spec_conv_integer:= bs_conv;
!       ! spec_tab( 2).length := 3;
!       ! spec_tab( 2).spec_echo(1):=  8;
!       ! spec_tab( 2).spec_echo(2):= 32;
!       ! spec_tab( 2).spec_echo(3):=  8;
!       ! spec_tab( 3).spec_conv_integer:= at_conv;
!       ! spec_tab( 3).length := 2;
!       ! spec_tab( 3).spec_echo(1):= 13;
!       ! spec_tab( 3).spec_echo(2):= 10;
!       !
!     END;
!
END;
```

D. HARD ERRORS

The error indications described are hard errors detected either by the firmware or the software driver. After a hard error all messages are returned and a software reset is performed.

error number

0 - 7: impossible.

8: software channel already reserved.

9, a: software error in communication between firmware and driver.
If one of these errors are reported when initializing the driver, it will usually be caused by:

a. an inconsistency between the hardware strapping and the driver parameters.

b. an illegal placement of the IMS208 board in the RC3502 crate. No spare slots must be left between the CPU and the IMS208 board.

c. a hardware error detected by the build in testprograms. This can be verified by using the testplug as decribed in ref. 6.

b - d: protocol failures between firmware and driver.

e - 12: internal firmware failures.

RETURN LETTER

Title: RC3502 IMS Driver Reference Manual RCSL No.: 52-AA1116

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