Title:

RC3600 DRIVERS REFERENCE MANUAL

A/S REGNECENTRALEN

RC SYSTEM LIBRARY: FALKONERALLE 1 DK-2000 COPENHAGEN F

RCSL No: 44-RT 1522

Edition: June 1977

Author: Dan Holmer Andersen

Keywords:

RC3600, MUS, MUSIL, Devicehandling, Driver, Driver Description.

Abstract:

This manual obsoletes RCSL: 44-RT 508,
RCSL: 44-RT 759 and
RCSL: 44-RT 1307

This manual contains descriptions of all standard Driver Processes available in the RC3600 MUS - system.

Printed by A/S Regnecentralen, Copenhagen

Introduction

This manual was former a section in the manual MUS PROGRAMMERS GUIDE (1) in which the general description of the devicehandling can be found.

An explanation of I/O handling in the MUS system can also be found in the MUSIL manual (2).

In section IV you will find the relocatable binary modules of which the driver description in this manual are valid.

REFERENCES

1) RCSL: 44-RT 1306 MUS INTRODUCTION and MUS PROGRAMMER's GUIDE.

2) RCSL: 44-RT 740 MUSIL

3) RCSL: 43-GL 3692 RC3600 BASIC SOFTWARE KEY

4) RCSL: 42-i0344 MUSIL PROGRAMMING GUIDE

CONTENTS

CONTENTS (continued)

SECTION III:

STANDARD DRIVER PROCESSES.

General Rules

This is an operator driver.

About operator process in general, see Part 2, Chap. 5.
The teletype driver operates on bytes placed left to right in words, allowing odd addresses and odd byte counts. The driver has to be in the same memory as the user process.

CONTROL

Control messages are ignored and returned with status = 0.

INPUT

Operation:	1	normal input
	5	input with timeout
	17	input with attention request

Operation code is tested by the driver as bit pattern.

Input General:

A line of characters can be input to the storage area defined by the message. The characters are echoed on the teletype during input.

In all input and character modes, characters CR, NL, and BELL have special effects:

BELL	Master Attention Request. Previous input is forgotten, and output in progress is terminated.
RETURN	New Line. An answer is delivered with status = 0, and bytes input defined.
LINE FEED	Same action as for RETURN.

Other control characters with ASCII values <32 terminate input. An answer with status = 0 and bytes input defined is returned.

If driver is in normal character mode, RUBOUT acts as character cancel:

RUBOUT — Char. cancel.
The character "←" is output on the teletype, and last character is cancelled in storage area. [1)]

Normal Input

Input is terminated when storage area is full or when one of the special actions causes termination. Input is terminated temporarily when no characters have been input for 5 seconds. If an output message is pending, input is cancellec, otherwise input is continued.

Input with Timeout

The same as for normal input, except that input is terminated when no characters have been input for a specified number of seconds placed in mess3 of input buffer. An answer is returned with status = timer (1b14) and bytes input defined.

Input with Attention Request

Present sender is inserted as user of attention. Driver will shift to second character mode, in which the following characters have special effects:

BELL)
RETURN) The same as for normal input
LINE FEED)

& — Char. cancel.
The character is echoed, and last character is cancelled in storage area.

% — Line cancel.
The character is echoed, and all characters are cancelled in the storage area.

ESCAPE — Attention.
An answer with status = attention (1b5) is returned to user of attention. If no user is present, the character is stored in the normal way.

---

1) RUBOUT is echoed as backspace on F13.

If ESC-key is activated, the buffer returned is found in the following way:

1. If driver is busy with output from user of attention, output is terminated, and buffer is returned with status = attention (1b5) and bytes output defined.

2. If driver is busy with input to user of attention, character is stored and input is terminated. An answer is returned with status = attention and bytes input defined.

3. If driver is busy with input or output from other processes, input is cancelled and output is terminated. Driver now searches for an input buffer with sender equal to user of attention. If found, character is stored, and the buffer is returned with status = attention. If no input buffer is found, the attention buffer is returned in the same way.

   Note that if driver is in second character mode, it cannot return to normal mode.

Master Attention Request

Attention request is set when BELL-key is operated. If driver is busy with output, it is terminated and repeated later.

OUTPUT

The storage area defined by the message is output as 8-bit characters. Output is terminated when the area is empty, after a NULL character, after BELL is operated or if ESC-key is activated and an attention answer is delivered. Byte values <13> and <10> are treated in a special way:

<not 13> <10> is output as <not 13> <13> <10>,

<13> <not 10> is output as <13> <10> <not 10>, and

<13> <10> is output as <13> <10>

Byte values < 16 (except 7, 10, and 13) are skipped at output.

The alphabet used on different hardware is:

F12 KSR TELETYPE:

	0	16	32	48	64	80	96	112	128
0			SPACE	0	@	P			
1			!	1	A	Q			
2			"	2	B	R			
3			#	3	C	S			
4			$	4	D	T			
5			%	5	E	U			
6			&	6	F	V			
7	BELL[1]		'	7	G	W			
8			(	8	H	X			
9			)	9	I	Y			
10	LF		*	:	J	Z			
11		ESC	+	;	K	[			
12			,	<	L	\			
13	CR		-	=	M	]			
14			.	>	N	↑			
15			/	?	O	←		RUBOUT	

Remarks: 1. Master attention request at input.

F13 ALPHANUMERIC DISPLAY/KEYBOARD:

	10	16	32	48	64	80	96	112	128
0			SPACE	0	@	P			
1			!	1	A	Q			
2			"	2	B	R			
3			#	3	C	S			
4			$	4	D	T			
5			%	5	E	U			
6			&	6	F	V			
7	BELL[3]		'	7	G	W			
8		RIGHT CURSOR	(	8	H	X			
9		LEFT CURSOR	)	9	I	Y			
10	NL	UP CURSOR	*	:	J	Z			
11		ESC	+	;	K	[			DOWN CURSOR[1]
12		HOME DOWN	,	<	L	\			
13	CR	HOME UP	-	=	M	]			
14		EOL[2]	.	>	N	↑			SPOW LATCH[1]
15		EOF[2]	/	?	O	LEFT CURSOR		RUBOUT	SPOW UNLATCH[1]

Remarks:

1. Only at output. Value -128 delivered at input.
2. EOL: Erase end of line.
   EOF: Erase end of frame.
3. Master attention request at input.

F14 SILENT PRINTER/KEYBOARD:

	0	16	32	48	64	80	96	112	128
0			space	0	@	P	\ [1]	p [1]	
1			!	1	A	Q	a [1]	q [1]	
2			"	2	B	R	b [1]	r [1]	
3			#	3	C	S	c [1]	s [1]	
4			$	4	D	T	d [1]	t [1]	
5			%	5	E	U	e [1]	u [1]	
6			&	6	F	V	f [1]	v [1]	
7	bell [3]		'	7	G	W	g [1]	w [1]	
8			(	8	H	X	h [1]	x [1]	back space [2]
9			)	9	I	Y	i [1]	y [1]	
10	NL		*	:	J	Z	j [1]	z [1]	
11		ESC	+	;	K	[	k [1]	{ [1]	
12			,	<	L	\	l [1]	¦ [1]	
13	CR		-	=	M	]	m [1]	} [1]	
14			.	>	N	^	n [1]	~ [1]	
15			/	?	O	_	o [1]	rub-out	

Remarks:
1. Only at output
2. Only at output. Value -128 delivered at input.
3. Master attention request at input.

F15 and F16 ALPHANUMERIC DISPLAY/KEYBOARD:

	0	16	32	48	64	80	96	112	128
0			SPACE	0	@	P	` [1]	p [1]	
1			!	1	A	Q	a [1]	q [1]	
2			"	2	B	R	b [1]	r [1]	
3			#	3	C	S	c [1]	s [1]	
4			$	4	D	T	d [1]	t [1]	
5			%	5	E	U	e [1]	u [1]	
6			&	6	F	V	f [1]	v [1]	CURSOR POS [4]
7	BELL [5]		'	7	G	W	g [1]	w [1]	
8		RIGHT CURSOR	(	8	H	X	h [1]	x [1]	BACK CURSOR [2]
9			)	9	I	Y	i [1]	y [1]	TAB [2]
10	NL	UP CURSOR	*	:	J	Z	j [1]	z [1]	
11		ESC	+	;	K	[	k [1]	{ [1]	
12			,	<	L	\	l [1]	¦ [1]	ERASE ALL [2]
13	CR	HOME	-	=	M	]	m [1]	} [1]	
14		EOL [3]	.	>	N	^	n [1]	~ [1]	PRINTER ON [2]
15		EOP [3]	/	?	O	_	o [1]	RUBOUT	PRINTER OFF [2]

Remarks:

1. Only F16
2. Only at output. Value -128 delivered at input.
3. EOL: Erase end of line.
   EOP: Erase end of page.
4. Cursor position: Next character will be the character address, and the following character the row address. Add 128 to wanted position to avoid skip of character.
5. Master attention request at input.

General Description

Control and Input messages are accepted. Output messages are interpreted as input messages. Leading blanks are skipped after a control message if the conversion bit is set.

Control

Reservation and Conversion are accepted. Conversion table address should be an even byte address.

Input Operation

Three modes of operation exist:

1	Binary, the input character is delivered.
5	Odd parity, the most significant bit is regarded as a complemented parity and removed.
9	Even parity, the most significant bit is regarded as parity and removed.

Common mode bit:

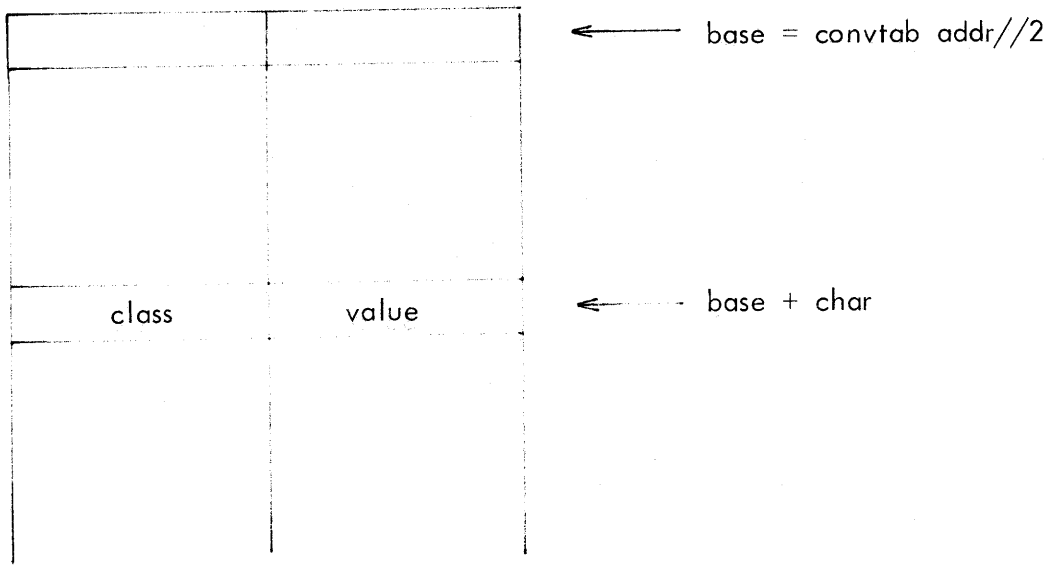
Bit 11 = +16	Do not skip leading blanks.
Bits 0-7	The first byte of the mode word is used as a defectmask. If the read character has bits in common with this mask, it is skipped, and the error is counted.

When operations 5, 9, 21 or 25 are used, the further treatment of the input character is:

```
repeat
char : = dia (device);
until char <> 0 and char <> 255; !CHAR SKIP!
if -, parity_ok (char) then char : = 26;
char : = char mod 128
```

At return MESS2 of buf holds the number of defective characters and characters with parity error in the block.

Conversion is done with the following table:



class = 0	normal character	the value is delivered.
class = 128	skip character	the character is skipped.
class = 129	shift in	base : = base - if value = 0 then 256 else value
class = 130, 131	shift out	base : = base + if value = 0 then 256 else value
132 $\leq$ class $\leq$ 135	end of block	the character delimits the current block.
class $\geq$ 136		parity error indicated and counted; value delivered.

Status

1B3	Char defect.
1B6	Device reserved.
1B10	Parity error is set if one or more characters with wrong parity occur in the block
1B11	End medium is set if the device is not ready to deliver a character within 1 second.

PAPER TAPE PUNCH DRIVER

General Description

Control- and Output messages are accepted. Input messages are interpreted as control messages.

Control

Reservation and Conversion is accepted. Conversion table address should be a byte address, and the characters areconverted before output by simple indexing.

conv character: = byte (convtable + character).

Position, Termination and Disconnect results in output of 128 characters blank tape.

Output operation

Three modes of operation exist:

3:	binary, the converted character is output.
7:	odd parity, the converted character is augmented by the complement of its parity in the most significant position.
11:	even parity.

Status

1B6:	device reserved
1B11:	em, is set if the hardware status bit 15 (paper low) is set.
1B14:	timer, is set if a character is not output within 600 ms.

LINE PRINTER DRIVER

General Description

Control- and Output messages are accepted. Input messages are treated as Control messages.

Control

Reservation and Conversion is accepted.

Conversion table address should be a byte address. The characters to be output are converted as

```
conv char:= byte (convtable + char);
```

The position command has only effect if Output mode is 7. Then the mess2 part (file) of the control message gives a number of spaces to be output in front of the printline.

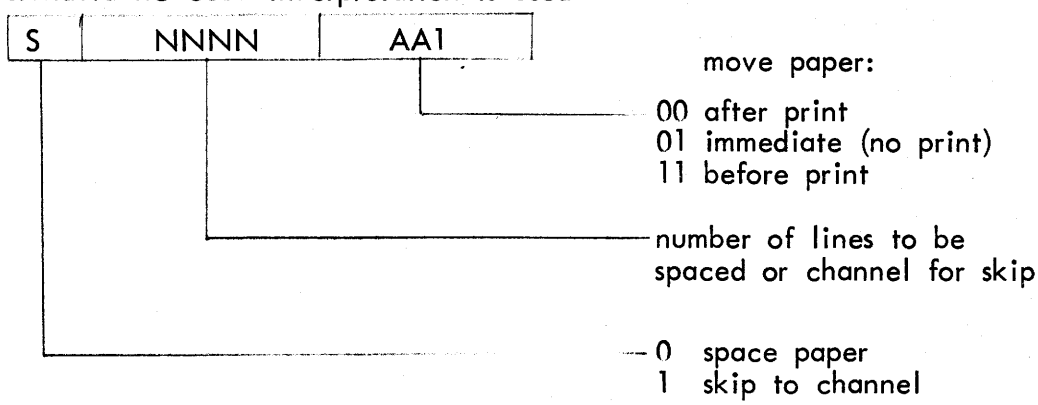
Output Operation

Two modes of operation exist:

3 : The converted characters are output to the line printer.

7 : The first byte of an output message is interpreted as a carriage control word.

The standard RC 3600 interpretation is used



Status:

1B0:	disconnected, line printer disconnected or unable to accept command.
1B1:	off-line, the line printer is in off-line state.
1B3:	device mode 1, channel 12 in carriage control tape has been encountered.
1B6:	illegal, device reserved.
1B8:	block error, paper is out, paper torn or paper run away.
1B9:	data late, line printer not in ready state.
1B10:	parity, a ccw contains a zero bit in the least significant position.
1B11:	end medium, end of paper, less than 1.5 forms left.
1B14:	timer paper run away.

General Description

Control, Input, and Output messages are accepted. On return mess2 and mess3 of the message contain the actual file and block number.

Control

Reservation, Termination, Positioning, Disconnect, Erase, and Sense are accepted. Conversion is ignored.

Bit 11 = 1	Termination:	Writes two tape marks and positions between them.
Bit 10 = 1	Positioning:	Positions the tape at the file and block number given by mess2 and mess3 of the message. Position command with file and block number = 1 acts as a rewind command. Position command with file number = X and block number = -1 positions before leading file mark in file X, i.e. after the last block in file X-1.
Bit 9 = 1	Disconnect:	Rewinds the tape and when BOT is sensed sets the unit off-line.
Bit 8 = 1	Erase:	Erase 3.7 inches of the tape

Input

Operation:

1	Read packed, byte limit = 12
5	Read packed, byte limit = 0
9	Read unpacked, byte limit = 12
13	Read unpacked, byte limit = 0

Output

Operation:

3 Write

Common Input/Output

Common mode bits:

Bit 4 = +8192: Selects the lower of two possible densities.

Bit 5 = +4096: Selects even parity.

Status

1B1	Off-line, the unit is off-line.
1B2	Busy, the unit is rewinding.
1B3	Device mode 1, noise record.
1B4	Device mode 2, PE.
1B5	Device mode 3, write lock.
1B6	Illegal, byte limit conflict or write lock, device reserved.
1B7	eof, end of file.
1B8	Block error, block length error (read).
1B9	Data late.
1B10	Parity error, read or write.
1B11	End medium, end of tape (EOT). This status bit is returned if after treatment of a message the read/write head is positioned behind the EOT mark. It is returned on read, write or position commands.
1B12	Position error, a file mark is read while positioning at a block. The blockno of answer is one greater than the number of blocks in the current file. Position before the filemark.
1B14	Timer, time out at wait interrupt, or at rewind.

CARD READER DRIVER

General Description

Control and input messages are accepted. Output messages are returned with illegal status.

Control

Only Control modes: Reservation and Conversion are performed, other modes are skipped. A sense of the device is always executed when the driver receives a control message.

Input Operations

Three modes of operation exist:

1	Read binary bytes
5	Read binary punched cards
9	Read decimal punched cards, skip trailing blank columns, and terminate with byte values <13><10>. A maximum of 72 card columns is delivered.
21	Read decimal punched cards.
33	Read decimal punched cards and skip trailing blank columns. A minimum of 10 columns is returned.

Common mode bit:

bit 9 = +64: if parity error then char:=31
Must not be added to mode 5

All other operation modes are returned with illegal status.

Binary Read Mode

Fig. 2 shows how the row numbers in each column are placed in the buffer.

Not used				12	11	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Fig. 2 Buffer word

Decimal Read Mode

Fig. 3 shows how the 12 bits in a column will be converted to a 8 bit character.

E								12	11	0	8	9	x	x	x
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Fig. 3 Buffer word

The table below shows how the E-bit (error) and the three x bits are converted.

Punch position number							Buffer word bit			
1	2	3	4	5	6	7	0	13	14	15
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	1	0	1
0	0	0	0	0	1	0	0	1	1	0
0	0	0	0	0	0	1	0	1	1	1
More than one 1's							1	x	x	x

Data Conversion

The conversion Table Address should be a byte address. If the operation is 5 (Read binary punched cards), no conversion of data is performed, and the card columns (Fig. 2) are delivered to the user. E.g. one card column fills up one user buffer word.

If the operation is 1, 9, 21 or 33 (Read decimal punched cards)(+64), the characters are converted before output by simple indexing:

conv character := byte (conv table + character)

e.g. one card column fills up one user buffer byte.

If Operation = 9 (+64), the last two bytes delivered are <13><10> generated by the driver.

Status

1B1	Off-line, card reader is not ready: Stacker jam, Stacker full, Stop five, power off, test mode, open interlock, start true, disconnected.
1B2	Busy, a card has failed to feed from hopper, a card has not passed over the read station in the specified time.
1B3	Device mode 1, reject command failed.
1B4	Device mode 2, 51 column cards.
1B6	Illegal command, device reserved.
1B8	Block length error, card reader has delivered less than 80 columns, user byte count less than received number of bytes from card reader.
1B9	Data late.
1B10	Parity error, light senser failure, at least one column in a card read in decimal mode contains a conversion error.
1B11	End Medium, hopper empty (no data have been transferred).
1B14	Timer, in wait interrupt has occurred (300 msec). The received data are delivered.

OPERATOR CONTROL PANEL DRIVER.

General Description and Rules.

This is an operator driver. The driver operates on bytes placed left to right in words. Odd addresses are not allowed but odd byte counts are.

Attention Request.

When the load button is pressed, the driver accepts this as a request to choose the operating system 's' as user. All input is then directed to 's' until 'NL' button or 'CAN' button is pressed. In attention mode the lamp labelled 'LOAD' is lit.

Control.

Control messages are ignored.

Input Operation.

A string of digits or an indicator message can be input to the storage area defined by the message. Only the digit, +, - buttons are echoed from position 12 on the display.

The other indicators have following effects:

CAN:	Echoed characters are cancelled and previous input is ignored. The operator can input a new text.
NL:	Terminates the previous input message by placing a NL as the last character of the string of digits.
START, CONT, STOP, INT:	(1) If an input operation is in progress from keyboard it terminates the previous input message by placing a NL as the last character of the string of digits. The button is then treated as for (2) if a new input message arrives.

START, CONT, STOP, INT: (2) If no previous input has occured the corresponding text: 'START'. 'CONT', etc. is delivered as answer.

Note: If more than one indicator is activated the left most is delivered.
On Keyboard in sequence: CAN,NL,SP,+,-, 0,1,...,8,9.

Input is terminated when the storage area is full, or as explained above, whichever occurs first.

An answer is delivered with status = 0 and bytes input defined.

Output Operation.

	$b_7b_6b_5$							
$b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0000	terminator		SP	Ø	@	P		
0001	terminator		!	1	A	Q		
0010	terminator		"	2	B	R		
0011	CLEAR ALL		#	3	C	S		
0100	START		$	4	D	T		
0101	CONT		%	5	E	U		
0110	STOP		&	6	F	V		
0111	ALARM		'	7	G	W		
1000	C.STOP		(	8	H	X		
1001	C.ALARM		)	9	I	Y		
1010	NL		*	:	J	Z		
1011			+	;	K	[		
1100			,	<	L	∿		
1101			-	=	M	]		
1110	C.START		.	>	N	{		
1111	C.CONT		/	?	O	}		

The figure shows the character repertoire and coding (binary). Empty positions indicate blind characters.

Output is delimited by the bytecount or by a character < 3.

Outputting the NL character causes all 16 character-positions in display to be cleared in one operation.

Displayable characters delivered after the NL character are displayed in sequence from left to right. If more than 16 displayable characters are delivered after the last NL character, the previous message is overwritten cyclically.

Blind characters have no effects.

The characters 'CLEAR ALL' to 'C.CONT' on figure are used to indicate the corresponding function indicators:

<3>	means	clear all lamps and audio alarm.
<4>	means	light the lamp labelled 'START'
<5>	-	- - - - - 'CONT'
<6>	-	- - - - - 'STOP'
<7>	-	turn the audio alarm on
<8>	-	clear the lamp labelled 'STOP'
<9>	-	turn the audio alarm off
<10>	-	NL
<14>	-	clear the lamp labelled 'START'
<15>	-	- - - - - 'CONT'

Note: Depressing any indicator turns the audio alarm off.

Status.

Not applicable.

READER PUNCH DRIVER

General Description

Control. Input. Output messages are accepted.

Control

Whenever the driver receives a control message, a sense of the device is executed.

Control modes. Reservation, Conversion, and Disconnect are performed.

The Disconnect mode causes the card in the wait station, if any, to be led out to the stacker in question.

Input/Output Operations

General

In the Input Output Operations bits (7:15) are used to define the type of command.

Bit 7 = 1 selects stacker # 2 or the secondary hopper depending on the mode of operation.

Bit 8 = 1 inhibits the input feed of the next card.

Bits (9:15) define the mode of operation.

The driver communicates with the device in either binary or decimal mode.

Binary Mode

In binary mode the driver words correspond to the card columns as shown in Fig. 1.

Not used				12	11	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Fig. 1. Driver Word

Decimal Mode

In decimal read mode the card column will be transferred to the driver as an eight-bit character as shown in Fig. 2.

In decimal write and in load print modes the driver word will be transferred to the device as shown in Fig. 3.

Fig. 4 shows the correspondence between bits (13:15) in the driver word and rows (1:7) in the card column, and for the decimal read mode how the E-bit (Error) is converted.

E								12	11	0	8	9	x	x	x
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Fig. 2. Driver word in decimal read mode

								12	11	0	8	9	x	x	x
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Fig. 3. Driver word in decimal write mode

Card Row Number							Driver Word Bit			
1	2	3	4	5	6	7	0	13	14	15
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	1	0	1
0	0	0	0	0	1	0	0	1	1	0
0	0	0	0	0	0	1	0	1	1	1
More than one 1							1	0	0	0

Fig. 4. Data conversion in decimal mode

Input Operations

Bit 7 = 1 selects the secondary hopper.

Bit 9 = 1 denotes that the card in the wait station will be led out to stacker = 2.

Bits (10:15) denote the mode of operation.

Five modes of operation exist:

1 : Read binary bytes with conversion.

5 : Read binary punched cards.

9 : Read decimal punched cards, convert, skip trailing blank columns, and terminate with byte values <13><10>. A maximum of 72 card columns are delivered.

21 : Read decimal punched cards and convert.

33 : Read decimal punched cards, convert, and skip trailing blank columns. A minimum of 10 columns is delivered.

All other modes of operation are returned with illegal status.

Except for the above-mentioned case (bit 9 = 1) stacker # 1 is used.

Bytecount must be equal to or greater than the number of bytes to be returned, or else the operation is returned with block length error and a number of bytes transferred equal to bytecount. the rest of the columns will be skipped. The bytevalues of the last two bytes delivered in mode 9 will always be <13><10>.

Byteaddress must be even or else the operation is returned with illegal status.

If the whole card is to be transferred, bytecount must be at least 80 for bytetransfer and 160 for word transfer.

Output Operation

Bit 6 = 1 Separates print data.
The first 80 (51) bytes are interpreted as the punch data, and the rest as the print data. (If binary words, the first 160 (102) bytes are the punch data). Only relevant in modes 3, 7, and 11.

Bit 7 = 1 Selects stacker # 2.

Bit 9 = 1 The card in the wait station is led out to the appropriate stacker, and a card is fed before the write operation (read before write).

Bit 10 = 1 Selects the primary hopper.

Bits (11:15) denote the mode of operation.

Six modes of operation exist:

3 : Punch binary byte, no conversion.

7 : Punch binary word, no conversion.

11 : Punch decimal byte with conversion.

19 : Print decimal byte with conversion.

27 : Punch and print decimal byte with conversion.

All other modes of operation are returned with illegal status.

Except for the above-mentioned case (bit 10 = 1), hopper # 2 is used.

In the decimal modes and the punch binary byte mode, bytecount must be less than or equal to 80 (51). If not, the operation is returned with block length error, and 80 (51) bytes will be transferred.

In the punch binary word mode bytecount and byteaddress must be even numbers, or else the operation is returned with illegal status. Bytecount must be less than or equal to 160 (102), or else the operation will be returned with block length error and 80 (51) words transferred.

If the decimal mode or the binary mode is used with separate print data, bytecount must be less than or equal to 160 (102). If not, the operation is returned with

block length error, and 80 (51) punch data and 80 (51) print data will be transferred.

If the binary word mode is used with separate print data, bytecount must be less than or equal to 240 (153). If not, the operation is returned with block length error, and 160 (102) punch data and 80 (51) print data will be transferred.

In all modes the card will be filled up with trailing blanks if bytecount is less than the maximum allowed.

Data Conversion

The Conversion Table Address should be a byteaddress.

If the operation is 5 (read binary word) or 7 (punch binary word) no conversion of data is performed, and the twelve last bits in a user word correspond to a card column. (Fig. 1).

If the operation is 1, 9, 11, 19, 21, 27, or 33, the characters are converted before output/input by simple indexing:

conv char : = byte (conv table + char)

E.g. one card column corresponds to one user buffer byte. If the operation is 3 or 7, only the print data, if any, is converted in this way.

Status

1B1 : Off-line, Reader Punch is not ready. Power off, power start-up, open interlock, Output Error, Stacker Full, Stop/Reset key was pressed, disconnected.

1B2 : Busy, a card has not reached the read station within a specified time, or a light check failure.

1B3 : Device mode 1: Stacker Full. (Is returned with 1B1).

1B4 : Device mode 2: 51 column cards.

1B5 : Device mode 3: Output Error, card jam in the output section. (Is returned with 1B1).

1B6 : Illegal, illegal command or driver process reserved.

1B7 : End of file, secondary hopper empty.

1B8 : Block length error, user bytecount does not comply with the number of columns in a card.

1B10 : Data error, read error, dark check failure, punch error, at least one column in a card contains a parity error.

1B11 : End medium, primary hopper empty.

1B13 : Rejected.

1B14 : Timer, in wait interrupt.

1B4 is one if the card equipment is for 51 column cards. Normally it should be disregarded.

1B10 indicates in input operations that the delivered data are erroneous. The programmer must decide what action is to be taken. In output modes 1B10 indicates a bad punching of the card. The operation must be repeated. For all other status bits the action to be taken is a repetition of the operation.

In output operations the answer to an operation is delayed the time required to perform another output operation. Therefore the multibuffer technique should be used in output operations where possible in order to operate the device at its full speed.

CASSETTE TAPE DRIVERS

General Information

The RC3625, RC3626 Cassette Tape Unit is in hardware equipped with logic, which automatically generates two CRC-16 blockcheck bytes. These bytes are during write-operations inserted at the end of each datablock and the read-after-write logic of the unit concurrently reads the block again and at the end of each block performs a comparison between the two CRC bytes generated by the read-logic and the two bytes read. This means that automatic check-read is performed during writing,and the block-check comparison is performed in hardware during reading.

This type of block-check is described in ECMA 34 standard dated July 1973, and the RC cassette tape unit was build in accordance to this standard.

However, there exists an earlier version (September 1971) of the ECMA 34 standard, which differs from the latest in aspect to the block-check bytes, because this standard defines one LRC byte as check byte.

Because the cassette tape unit, through the standard parallel interface, also delivers the block-check bytes to the program, it is possible to ignore the hardware generated block-check status and in stead generate a block-check byte(s) in the driver. This means that the standard driver is able to read cassette tapes recorded according ECMA 34 version 1 (September 1971).

The above-mentioned characteristics are valuable for the cassette tape driver CT007, and this driver is the standard MUS-driver, which always must be used, when the problem in question can be solved by CT007.

In the cassette tape unit (under the cover) a build-in switch makes it possible to switch off the automatics block-check generating and the automatic read-after write function. When doing so all check must be performed in the driver, and for this purpose an alternative cassette tape driver CT400 has been implemented. This driver has the same possibilities as CT007 except that, it is also able to write datablocks according ECMA 34 version 1.

Furthermore the read-after write feature is substituted by a special write-mode performing check-reading, by means of repositioning.

NOTICE! The CT 400 will load the CPU more as CT007, and must therefore only be used if it is necessary to write according ECMA 34 version 1 standard.

The CRC-MODE switch must be in the position AUT CRC CHECK ON when CT007 is used, because data else will be erroneous without any status indication.

CASSETTE TAPE DRIVER CT007

General

This driver requires that the CRC-mode switch in the tape unit (RC3625, RC3626) is set to the position AUT-CRC CHECK-ON!.

Control, input messages, and output messages are accepted. At return mess2 and mess3 of the message contain the actual file- and block-number.

Control

Reservation, Conversion, Termination, Positioning, Disconnect, Erase, and Sense are accepted.

Bit 12 = 1 :		Conversion, Byte: = byte (convtable + byte);
Bit 11 = 1 :		Termination; writes two tape marks (according to ECMA 34 vers. 2) and positions between them.
Bit 10 = 1 :		Positioning; positions the tape at the file-and block-number given by mess2 and mess3 of the message.
Bit 9 = 1 :		Disconnect; rewinds the tape and ejects the cassette.
Bit 8 = 1 :		Erase; erases a length of tape corresponding to approximately 150 characters.

Input

As it is possible to read data according to several standards, the following input modes are valid:

Operation:

1 :	Read one block, ECMA 34 vers. 2 (CRC 16 check of data block).
9 :	Read one block, ECMA 34 vers. 1 (LRC check of data block).
17 :	Read one block, no check is performed.

Read continuously (slew mode):

Operation :

op +4 : The cassette tape unit will read the data blocks without stopping at inter-block gaps if input messages are contiguous within 20-40 ms.

Output

Operation :

3 : Write one block according to ECMA 34 vers 2.

Status

1B1 : Off-line, disconnected, cassette released, or power interrupt.

1B3 : Device mode 1: End of Data, erased tape of a length of 250 to 400 mm has been found.

1B6 : Illegal, illegal command.

1B7 : EOF, End of File, tape mark according to ECMA 34 vers 1 and 2 was read.

1B8 : Block-length error.

1B9 : Data late,

Output: Output buffer has been emptied before the termination of write operation.

Input: Input buffer has been filled.
Notice: After this status the position might be lost.

1B10 : Block-check error.

1B11 : End Medium, BOT/EOT hole is sensed in the tape during operation (data is not lost).

1B12 : Position error.

1B14 : Timeout; the operation has been started before the reader is ready (at rewinding etc.) - or possibly hard error in the cassette tape unit, or it has been attempted to write on a cassette with the write-enable plugs removed.

CASSETTE TAPE DRIVER CT 400.

General

This driver implements reading and writing of cassettetapes according to ECMA 34 vers. 1 and 2. Requires that the modeswitch in the tape unit (RC 3625, RC 3626) is set to position: AUT.CRC CHECK-OFF.

All necessary blockcheck generation and accumulating is performed in software.

Control, input messages, and output messages are accepted. At return mess2 and mess3 of the message contain the actual file- and blocknumber.

Control

Reservation, conversion, termination, positioning, disconnect, erase and sense are accepted.

Bit 12 = 1:	Conversion. char:=byte (convtable + char).
Bit 11 = 1:	Termination; writes two tape marks and positions between them. NOTICE ! These tapemarks are always written according ECMA 34 vers. 2.
Bit 10 = 1:	Positioning; positions the tape at the file- and blocknumber given by mess2 and mess3 of the message.
Bit 9 = 1:	Disconnect; rewinds the tape and ejects the cassette.
Bit 8 = 1:	Erase; erases the tape in a length corresponding to approximately 150 characters.

Output

Operation

3:	Write according ECMA 34 vers. 2
7:	Write according ECMA 34 vers. 2 with control read.
11:	Write according ECMA 34 vers. 1
15:	Write according ECMA 34 vers. 1 with control read.

Input

Operation:

1:	Read one block according ECMA 34 vers. 2
5:	Read continously according ECMA 34 vers. 2
9:	Read one block according ECMA 34 vers. 1
13:	Read continously according ECMA 34 vers. 1
17:	Read one block with no block check.
21:	Read continously with no block check.

Read continously: The CTU will read the datablocks without stopping at interblockgabs, if the input messages are contigous within 20-40 ms.

Status

1B1 : Off-line, disconnected, cassette released, or power interrupt.

1B3 : Device mode 1: End of Data, erased tape of a length of 250 to 400 mm has been found.

1B6 : Illegal, illegal command.

1B7 : EOF, End of File, tape mark according to ECMA 34 vers 1 and 2 was read.

1B8 : Block-length error.

1B9 : Data late.

Output: Output buffer has been emptied before the termination of write operation.

Input: Input buffer has been filled.
Notice: After this status the position might be lost.

1B10 : Block-check error.

1B11 : End Medium, BOT/EOT hole is sensed in the tape during operation (data is not lost).

1B12 : Position error.

1B14 : Timeout; the operation has been started before the reader is ready (at rewinding etc.) - or possibly hard error in the cassette tape unit, or it has been attempted to write on a cassette with the writeenable plugs removed.

DISC DRIVER

General Description

Control, Input, and Output messages are accepted. The driver can handle up to 4 disc drives connected to the same controller.

Each disc pack mounted on a drive contains 2 surfaces of 203 cylinders of 12 sectors, giving a total of 2 x 203 *12 = 4872 sectors of 512 bytes.

For addressing drives and sectors logical segment numbers are used. The physical addresses are computed like this:

driveno = logical segment/4872
cylinder = logical segment/24 mod 203
surface = logical segment/12 mod 2
sector = logical segment mod 12

Control

Reservation and Position are accepted. Other control messages are ignored.

Position uses blockno = mess3.buf as current logical segment number, and positions the disc accordingly.

Input

Two modes of operation exist:

1	sequential read
5	random read

Bytecount of message must be an integral multiple of 512. Address of message must be an even byteaddress.

In sequential read mode bytecount/512 sectors are read from current segment and forward, current segment being increased by one for each sector read.

In random read mode current segment is first assigned the value of mess3.buf, and then input takes place as for sequential read.

Note: When the standard I/O system is used, sharelength should be exactly 512, otherwise repetition will function incorrectly.

Output

Four modes of operation exist:

3	write sequential
7	write random
11	write/read
15	write/read random

Bytecount of message must be an integral multiple of 512. Address of message must be an even byteaddress.

In sequential write mode bytecount/512 sectors are written from current segment and forward. Current segment is increased by one for each sector.

In random write mode current segment takes the value of mess3.buf, and output is then performed as in sequential mode.

Note: When the standard I/O system is used, sharelength should be exactly 512, otherwise repetition will function incorrectly.

Answers

In any answer from the driver mess1 gives the number of bytes transferred.
Mess3 contains current logical segment number.

Status

1b0	disconnected	the selected drive is not available, but is expected to be, malfunction (hardware bit 12).
1b5	drive does not exist	the drive addressed is outside the system.
1b6	illegal	the disc driver is reserved, address is an odd byteaddress, drive does not exist (as 1b5).
1b8	blockerror	the bytecount is not an integral multiple of 512 bytes, address is an odd byteaddress.
1b9	data late	data channel overrun.
1b10	parity	parity error on disc sector in read operation.
1b11	end medium	end of cylinder detected without zero sector counter (hardware bit 11).
1b12	position error	a seek has failed (hardware bit 10).
1b14	timeout	an operation has not terminated within 200 ms (seek, I/O) or 640 ms for recalibrate.

MULTIPLEXER DRIVER (RC3683)

GENERAL DESCRIPTION

Control, input- and output messages are accepted. The driver exists in 5 versions :
one for 16 half duplex channels, one for 24, one for 32, one for 40, and one for 64 half duplex channels.

Initialization

A message of the format :

mess0 : -1
mess1 : count
mess2 : address
mess3 : irrelevant

is used to initialize the line descriptions.

When the driver is loaded, all line descriptions are cleared.

When an initialization message arrives, the initialization information is added to the existing line descriptions with the possibility of redefining old initialization information.

Address and count points to a block of words giving the type and physical channelnumber of each line.

address/2 : line descriptor 0
.
.
line descriptor N

} "count" bytes.

Where each line description occupies 4 bytes

echo	channel, kind
time0	time1

echo is the echo channel for a full duplex input line, otherwise 255. Channel, kind is the physical channel number * 2 + 1 if output or combined input/output channel.

If the kind of the specified echo line is not output, or the specified echo line is not idle, the echo line specification is ignored.

Time0 is a maximal wait time for first character (in full seconds) or total wait time for output. Time = n means n-1 < TIME < n.

Time1 is the allowable wait time between characters (in full seconds). Time = n means n-1 < TIME < n.
The initialization message will cause all pending messages to be returned with status : disconnect, except the initialization message itself, which is returned with zero status. All channels will be set idle and reinitialized.

Line Addressing

For all normal messages, the line number is given in the left byte of mess0 (operation).

operation (0 : 7) = line

Special Messages

A control message with mess0 (operation) equal to zero is always accepted, and returned with zero status.

Break, Power Interrupt

When a break process or power interrupt occurs, all messages in queue are returned with status disconnected, and all channels are set idle and reinitialized.

For each output channel DTR is set to its latest state.

Initial Values

At load time, the below default values are valid :

DTR is cleared,
Line conversion table = 0 (no input conversion),
No linenumbers known.

Data Channel Overrun, Input buffer overflow

If one of these events are detected, all messages in queue are returned with status data late or buffer overflow, and all channels are set idle and reinitialized.

Interrupts

If an interrupt for a channel without messages is detected, the cause is returned in next buffer, and the channel is set idle.

CONTROL

Reservation, conversion, position, and disconnect are executable. Termination and erase have no effect.

A control message with mess0 (operation) equal to zero is always accepted, and returned with zero status (may be generated by the I/O system).

A control message with mess0 equal to -1 is used as an initialization message.

All other control messages will suspend input messages held by the line.

Reservation

In case of output line and mess1 <> 0, DATA TERMINAL READY will be set.

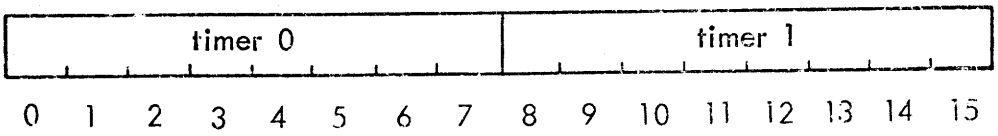
Conversion

Mess2 is taken as a byte address of an input conversion table. The table should start on an even byte (full word boundary).

Mess2 = 0 means no conversion.

Position

Mess2 is taken as new timer 0, timer 1 value for the specified line.



Timer 0 and Timer 1 are measured in seconds between 0 and 255.
Time = n means n-1 ≤ TIME < n.

For input operations timer 0 is the maximum time allowed before the first character and timer 1 the maximum time allowed between characters.

For output operations timer 0 is the maximum allowable time for the output operation.

Disconnect

In case of output line, DATA TERMINAL READY is cleared, and the channel is stopped.

Input/attention messages held by the line are returned with their present status.

INPUT

Operation (8 : 15) :

1	normal input
17	attention
33	input continued
+4	parity check for even parity of characters
+8	echoing of input takes place to echo channel.

Operation of channel

When an input message arrives to a channel in idle state, a READ command is executed on the channel. The channel is kept in this state regardless of bufferchanges. The channel will be returned to idle state in case of the following events.

1) A power interrupt or other break.
2) An interrupt from the channel.
3) Arrival of a control or output message; then a HALT command is executed, and the input buffer is cleared for characters to this channel.

Input of characters to cyclic buffer

When a channel has been started all characters received in the buffer are kept there when the channel is not ready to accept them.

If this leads to a situation where the distance between the first waiting character and the most recently entered, is greater than 1/2 of the buffer-length, then the channel belonging to the first character is alerted with status character lost and the character is removed. The status is returned in the next arriving message for the channel irrespective of the type of operation.

Attention

When an attention message is the single message present, it will accept all characters. The message will be returned when

1) it is active and an attention character arrives.
2) it is active and an interrupt arrives.
3) a control operation with DISCONNECT arrives.
   Then status is zero in return.

The use of the message will be temporarily suspended when :

another message arrives.

Only an attention character is stored in the buffer.

Normal input

When a normal input message is activated, the input buffer is cleared for characters to this channel. Then it will accept all characters.
It will be returned when

1) an attention character, or a termination character arrives.
2) the specified number of characters have been read.
3) an interrupt occurs.
4) a timeout occurs.
5) a control operation with DISCONNECT arrives.

Input continued

When a input continued message is activated, it will accept all characters, in this way no characters are cleared. It is returned as described for normal input. This mode may only be used for full duplex input channels.

Operation of a Line

The driver event queue is treated as a set of independent line event subqueues.

Input Suspension

A current input message may be temporarily suspended, being replaced by some later message for the line (subqueue look ahead). Attention messages are suspened by any later message for the line. Other input messages are suspended by any later control or (half duplex) output message for the line.

When the current message has been returned, the oldest item in the line subqueue will be the next current message. (A previously suspended input message will be resumed with character buffer erased).

Kill Output

Termination by attention on an input line (full duplex) will stop concurrent channel output. The output message is returned with status attention.

Echo

If the output line is idle when the channel transmitter is requested for the first echo character, the input line will delay the corresponding output line (echo reservation) until input termination. Otherwise (output line busy), the input message is returned with status character lost.

Treatment of Characters

When a character is received, parity check is performed if specified in the input message and if a parity error is detected, the valve 26 (decimal) is substituted. When parity checking is specified, the 7 last significant bits are taken as character value.

When the conversion table address is non-zero, conversion is performed as table lookup giving a class and a value byte for each character.

convtab//2

class	value

; value "0"

class	value

; value "char"

class = 0 — Normal character, the value is delivered and if specified echoed.

class 128 <> 0 — Attention character. Input is terminated and the input or att buffer returned with status attention. If an output operation is in progress, it will be stopped and returned with status attention. If specified the value is delivered and if specified echoed.

class 64 <> 0 — Termination character. Input is terminated with this character. If specified the character is delivered and if specified echoed.

class 32 <> 0 — Special echo, the word class concatenate value bits 3 to 15 are taken as a displacement in words from convtab//2 to a subtable:

convtab//2:

XX1	displacement

; value "char"

displacement

subclass	value
echo 1	echo 2
echo 3	echo 4

subclass	128 <> 0	Erase current input buffer.
subclass	64 <> 0	Erase last character if any from input buffer. If the input buffer is empty, the string echo 1, echo 2, - - - is not output.
subclass	32 <> 0	Value is not delivered.

Value gives the value delivered, and echo 1, echo 2, - - - is a string terminated by 128, which is echoed on the output line if specified.

class	16 <> 0	Mark, sets the mark answer status bit.
class	8 <> 0	Shift character, conversion table address is changed as follows :

```
convtab : = convtab +
            (if class    4 = 0 then value
            else -value) * 2;
```

Then conversion is repeated.

Echo

If an error is detected on the echo channel, input is terminated, and this status is delivered to the input channel.

OUTPUT

operation (8 : 15):

3	write
67	break.

The text to be output or used in timing the break sequence should be terminated by a stop character (value 128).

Answer

Mess2 of an answer contains the line number (except mess 2 = 1b6 by unknown line).

STATUS :

1B0	disconnected
1B1	data set not ready
1B2	colling indicator
1B3	carrier off
1B4	mark
1B5	attention char received
1B6	channel unknown, illegal command.
1B7	character lost (echoline busy or interrupt buffer overflow or compress cyclic input buffer).
1B8	cyclic input buffer overflow
1B9	data late
1B10	parity error
1B11	break received
1B14	time out

INCREMENTAL PLOTTER DRIVER

General Description.

Control- and Output messages are accepted.

Control.

Reservation and Conversion is accepted. Conversion table address should be a byte address. The characters to be output are converted as

```
conv. char: = byte (convtable + char);
```

Output Operation.

One mode of operation exist:

3: The converted characters are output to the incremental plotter.

Only the last 4 bits in a byte are significant; if the converted byte contains more bits then the byte will be skipped.

The meaning of the last four bits in the byte is:

Decimal	Binary	
0	0000	= step (+dy, 0)
1	0001	= step (+dy, +dx)
2	0010	= step (0, +dx)
3	0011	= step (-dy, +dx)
4	0100	= step (-dy, 0)
5	0101	= step (-dy, -dx)
6	0110	= step (0, -dx)
7	0111	= step (+dy, -dx)
8	1000	= step all pens up
9	1001	= pen 1 down, pen 2 and pen 3 up
10	1010	= pen 2 down, pen 1 and pen 3 up
11	1011	= pen 3 down, pen 1 and pen 2 up
12	1100	= not used
13	1101	= not used
14	1110	= not used
15	1111	= not used

Status.

1B0	:	Power break, pen position undefined.
1B6	:	illegal operation (input)
1B14	:	timer

SERIAL PRINTER DRIVER

General Description

Control and Output messages are accepted. Input messages are treated as Control messages.

Control

Reservation and Conversion is accepted.

Conversion table address should be a byte address. The characters to be output are converted as

```
conv char :=byte(convtable+char);
```

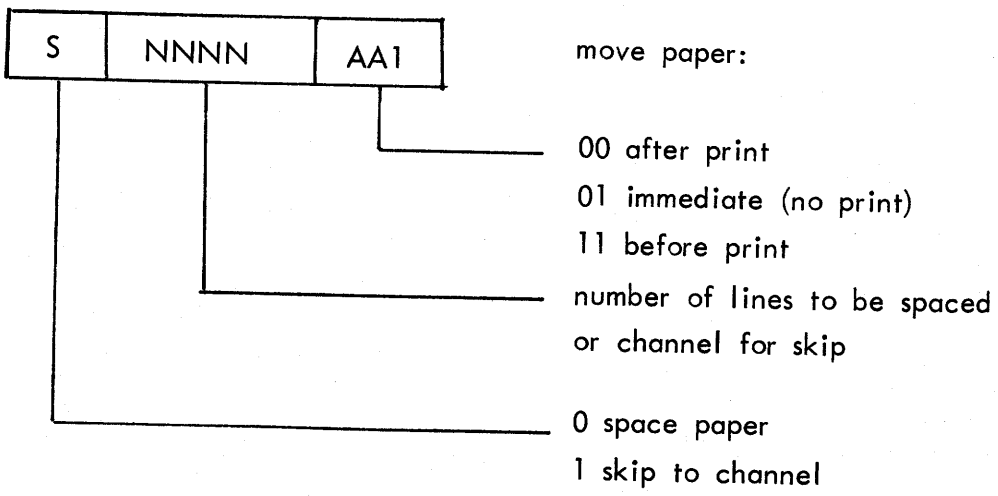
The character value 26 is skipped, 31 and 255 is printed as space. The position command has only effect if Output mode is 7. Then the mess2part (file) of the control message gives a number of spaces to be output in front of the printline.

Output Operation

Two modes of operation exist:

3: The converted characters are output to the line printer.
7: The first byte of an output message is interpreted as a carriage control word.

The standard RC 3600 interpretation is used



If the carriage control word indicates a skip to channel 1, a form feed will be executed. If a skip to channel 2 to 12 is indicated, a vertical tabulation will be executed, i.e. the paper will be moved to the next hole in channel 5 of the Vertical Format Unit tape is detected.

Status:

1B0	disconnected, line printer disconnected or unable to accept command, power break.
1B1	off-line, the line printer is in off-line state.
1B6	illegal, device reserved.
1B8	Block error.
1B9	data late, line printer not in ready state, end of paper, paper run-away, or a malfunction in the video-circuit of the printer.
1B10	parity, a ccw contains a zero bit in the least significant position.
1B11	end medium, end of paper, less than 1.5 forms left.
1B14	timer, paper run-away.

FLEXIBLE DISC DRIVER

Control, input messages, and output messages are accepted.
The diskette is handled as a random access device.
The size of a segment is 128 bytes, and an arbitrary slice size is handled.

When blockcount and bytecount are given, the slice size, the track, and the sector are calculated as:

```
slice  : = bytecount // 128;
track  : = ((blockcount -1) * slice) // 26;
sector : = if physical position in mode then

           ((blockcount -1) * slice) mod 26 + 1

           else

           begin !logical position!
           (((blockcount -1) * slice * n) +1) mod 26;
           if sector = 0 then sector : = 26;
           end !logical position!;
```

where every n'th physical sector is read or write in sequence.

Control

Reservation, conversion, position, and erasure are accepted.

Conversion : The content of mess2.buf is used as conversion table address. The conversion is performed as:

Converted char.: = byte (conversion table + char.)

Position : The logical position is set according to the content of

mess2.buf : - deletecount
mess3.buf : blockcount

where blockcount > 0.

If the content of mess2.buf < = 0 the numerical value is used to calculate a trackdisplacement:

displacement : = -(mess2.buf) // 26; (1)

The displacement denotes the number of tracks prior to the current

one which has been deleted, i.e. the mess2.buf is the negative number of deleted segments. The last specified displacement is used by the driver prior to all transput messages.

The displacement is cleared if status (off-line) or the break address is entered. Otherwise only the user can change the displacement.

The trackdisplacement has been introduced in order to make it possible to change the content of the track address register in the controller. If one or two tracks on the diskette are deleted (i.e. replaced by the succeeding track), all the address fields on the succeeding tracks may be changed according to the number of deleted tracks.

If track 'i' is deleted, the content of the track field in all address fields succeeding track 'i' is changed to:

addressfield.track = n - 1;

where n (= physical trackno.) > i

It is not possible to change the address fields by means of the driver.

Erasure : The current block is deleted by writing a delete data sync. byte (hex(F8)) in front of the data field of the first sector in the slice. The data content is undefined. The blockcount is increased by one.

Transput

General		Message	Answer
mess0.buf	:	operation	status
mess1.buf	:	bytecount	bytecount
mess2.buf	:	first byteaddress	displacement*
mess3.buf	:	blockcount	blockcount

*) See (1) under 'Position'

where bytecount mod 128 <> 0 will give status (blockerror) and

bytecount : = bytecount // 128 * 128.

The operation is continued.

The blockcount is increased by one, if the bytecount <> 0 after a successful transput operation.

Mode

Input	x + 1	: read, sequential
	x + 5	: read, random
	x + 17	: read, sequential, nonskip
	x + 21	: read, random, nonskip
Output	x + 3	: write, sequential
	x + 7	: write, random
	x + 11	: write/read, sequential
	x + 15	: write/read, random

where x : = if logical position then 32 else 0.

Transput Operation

Sequential : The blocks are transferred successively through the diskette. The position may be redefined by the control command 'position' or by a random transput operation.

Random : The content of mess3.buf is used to set the position of the diskette, and this blocknumber is transferred.

Logical Position : The sectors are organized so that every 7th sector corresponds to successive segments.

Input Operation

Description of the read modes:

Read : If the data filed sync.byte = hex(FB), the positioned block is transferred to the core;

if the data field sync.byte = hex(F8) in any of the sectors in the block, it is skipped, and the next block is read. This will go on until either a successful operation or end medium is sensed. The blockcount is increased by one for each skipped block;
if the data field sync.byte <> hex(FB, F8) the block is transferred with parity error;

Nonskip : The data are transferred to the core regardless of the data field sync.byte.

Output Operation

Description of the write modes:

Write : If the write protection bit is sensed, the status (illegal) is set, and the answer is returned immediately, else the content of the specified data buffer is written in the positioned block.

The data field sync.byte := hex(FB);

Write/read : The output is set up as for write. Then a nonskip read operation is executed, yet no data are transferred to the core. The status is investigated and returned.

Status

b0 : disconnected, the disc is disconnected or unable to accept commands.

b1 : off-line, the cartridge door is or has been open; the drive is empty (no diskette); the power has been turned on.

b3 : address field error, parity error in the address of the positioned block.

b4 : skip block, a delete data field sync.byte hex(F8) is sensed or one or more blocks are skipped.

b5 : write-protected, the write protection button is ON.

b6 : illegal, device reserved;
write-protected in output operation;

b8 : block error, specified bytecount is not an integral multiple of 128;

b10 : parity error, the parity of the address field and/or the data field is wrong; an undefined data field sync.byte is sensed;

b11 : end medium, the position is outside the disc;

b12 : position error, position command has been attempted to a non-existing block;
transput operation: the track address read is wrong;
if transput = read the data is transferred;
if transput = write the operation is terminated;
parity error is masked out.

b14 : timer, a time-consuming operation was impossible to execute or has lasted more than 2.4 sec.

Other Information

cur + 27 : the last read (track < 8 + data field sync.byte)

cur + 28 : unmodified status

IBM BINARY SYNCHRONOUS COMMUNICATION DRIVER

General Description

The driver conforms to the IBM BSC standard of point-to-point communication over telephone lines. The main purpose of the driver is to simulate the facilities of an IBM Batch Terminal such as IBM 2780.

Applicable documents:

IBM Systems Reference Library:

General Information : Binary Synchronous Communications GA27-3004

Component Description : IBM 2780 Data Transmission Terminal GA27-3005

Input Operation

The input mode is 1. The mode is used to receive one block of data from the other side. The blocks may consist of any number of units of data.

The driver automatically handles transparent/non-transparent mode of operation. The received blocks of data are checked, and stripped of all transmission control sequences before they are returned to the user as VB-formatted blocks, each unit corresponding to one variable format record in the block. No check is performed in the driver on the length of a unit or block, nor on the end-to-end control characters.

Correct acknowledgement (ACK 0/1) or call for retransmission (NAK: up to 3 times) is handled automatically by the driver. Recognition of bid-for-the-line, call-for-reply (ENQ), temporary-text-delay (TTD), and end-of-transmission (EOT) is also handled automatically.

In the case of persistent errors, or if the user does not supply new input commands within given time limits, the receiving of blocks is temporarily aborted by means of an end-of-transmission communication (EOT) to the other side.

Output Operation

Four modes of operation are allowed:

3 : Non-transparent mode, block terminator = ETB

7 : Transparent mode, block terminator = ETB

11 : Non-transparent mode, block terminator = ETX

15 : Transparent mode, block terminator = ETX

Modes 3 and 7 are used during normal output operation, indicating at the other end that more data is pending. Modes 11 and 15 are only used when the last block of data in a text is to be transmitted, i.e. just before procedure close is called.

The driver must receive data blocks from the user in V- or VB-format. The data blocks are automatically extended by the necessary transmission control sequences and check characters. Each variable format record is represented as one unit of data in the block. No check is performed on the length of a unit or block, nor on the end-to-end control characters.

The driver recognizes correct acknowledgements (ACK 0/1) or call for retransmissions (NAK) and performs retransmissions (up to 3 times) automatically. Also the driver recognizes end-of-transmission (EOT), wait-for-acknowledgement (WACK), and reverse-interrupt (RVI), and performs the appropriate control actions automatically.

Bidding for the line and calling for reply (ENQ) is automatically performed by the driver when necessary.

If the user does not supply new data blocks within given time limits, the driver holds the line active by means of temporary-text-delay (TTD) communications.

In the case of persistent errors the transmission of data blocks is temporarily aborted by means of an end-of-transmission (EOT) communication to the other side.

Control Operation

Reservation (R) :	is accepted in any combination, and respected.
Conversion (C) :	is accepted in any combination, but ignored.
Termination (T) :	is accepted in any combination, but ignored.
Position (P) :	is not accepted with D set also. The contents of mess 2 (file) and mess 3 (block) are taken as base values for the retransmission- and record-counters respectively.
Disconnection (D) :	is not accepted with P set also. End-of-transmission (EOT) is sent, and the retransmission- and record-counters are cleared.
All other bits :	illegal in any combination.

Special Information

Mess 2 (file) and mess 3 (block) of every answer from the driver indicate the number of retransmissions and the number of correctly transferred records respectively. These counters are reset to zero after the return of each D-control command, and may be reinitialized by means of a P-control command.

The maximum blocklength that can be received by the driver is an assembly constant which is 800 bytes at present. This length includes all transmission control sequences, check characters, and intermediate synchronization (SYN) characters.

The maximum blocklength that can be transmitted from the driver is determined by the same assembly constant. This length includes all transmission control sequences and check characters. No intermediate synchronization (SYN) characters are inserted in the block.

Status

1B0 : Data set not ready

1B1 : Carrier off while receiving

1B2 : No legal bid received (input)
Bid failed (output)

1B3 : Reverse interrupt received (output)

1B4 : Call for reply (ENQ) received on reply (input)
Wrong acknowledge (ACK) received (output)

1B5 : Receive buffer (word count) overflow
Illegal heading on block or unit (input)
Illegal reply to block (output)

1B6 : Illegal command

1B7 : End-of-text (ETX) received in previous block (input)

1B8 : Share too small to hold block (input)
Share too big to be transmitted (output)
Illegal length fields in share (output)

1B9 : Data channel late while receiving

1B10 : Negative acknowledge (ACK) sent on bcc-error (input)
Negative acknowledge (NAK) received on block (output)

1B11 : End-of-transmission (EOT) received

1B14 : Timeout on block or unit (input)
Timeout on reply (output)

Many of the above-mentioned status bits may occur in combinations which describe the actual events that have occurred during the automatic error recovery of the driver. E.g. if B10 and B11 are set in input mode this means that the driver has found a bcc-error and thus replied with negative acknowledgement (NAK), and the other side has tried to retransmit the block. Again the driver found a bcc-error, replied with NAK, etc., until the other side gave up, aborting the transmission with EOT. Another example is at the end of reception of a text. In order to consider the reception of a text to be final, B7 and B11 must both be set, indicating that the last block before EOT contained ETX.

Apart from the above-mentioned status combinations which describe actions and events there exist status combinations which have a special meaning:

1B8 + 1B10	Illegal character in share (output)
1B2 + 1B0	Data set not ready in bid phase
1B2 + 1B1	Carrier off in bid phase
1B2 + 1B4	Bid (ENQ) received in bid phase (output)
1B2 + 1B5	Receive buffer (word count) overflow in bid phase Illegal heading on bid (input) Illegal reply to bid (output)
1B2 + 1B9	Data channel late in bid phase
1B2 + 1B10	Negative acknowledge (NAK) received on bid (output)
1B2 + 1B11	End-of-transmission (EOT) received in bid phase
1B2 + 1B14	Timeout waiting for bid (input) Timeout on reply to bid (output)

Reverse Interrupt status (B3) will always appear alone, and according to the BSC standard the transmitted block which received this reply was accepted

by the other side. This means that output of the block in this must not be repeated (by means of repeatshare). The driver will continue to process pending transput messages as if a correct acknowledgement had been received.

Current Limitations

EBCDIC character set only.
Half-duplex only.
No multi-point communication.
BEL feature not implemented.
No horizontal tabulation (not a device-driver function).
No automatic answerback.

RC 3600 - RC 4000 COMMUNICATION DRIVERS.

General description

The drivers are intended for synchronous communication with RC 4000 using the controllers SCC 702 on RC 3600 and SCC 401 on RC 4000. The blockformat therefore is the one recognized by SCC 401:

{SYN} 2 < start > < data > < bcc >

where < start > consists of one character different from SYN
< data > consists of a number of 8-bits characters
< bcc > consists of two crc 16 characters accumulated over data.

The drivers consists of two separate processes,

R40X for the transmitter controller
R40R for the receiver controller

The two processes have some common procedures and variables located in the code and processdescription of R40X. Their addresses are retrieved at load time by R40R, so the drivers must be loaded in this sequence = R40X, R40R.

Control

Reservation, disconnect and conversion have effect.
Reservation with mess1 < > 0 will cause the modem signal DATA TERMINAL READY to be set.

Disconnection of R40X will cause the signal to be cleared. Having changed the signal, the driver will delay the answer for 200 msec.

Conversiontable has the only purpose of passing parameters to the drivers.
It has the following format:

+0	timer 1
+1	timer2
+2	SOH
+3	STX
+4	ENQ
+5	SYN
+6-7	First word in testbuffer
+8-9	First word after testbuffer (see testoutputsection).

Timer1 is used by the receiver as the maximum time to wait for the start of a block. It is used by the transmitter in delayed transmit mode.

Timer2 is the maximum time allowed for the transmission or reception of a single block.
Unit for timers is 100 msec, exept that the transmitter uses the unit 20 msec for timer 1.

SOH, STX and ENQ are values of the character < start > specifically recognized by the receiver (see section status). Receiverhardware will be set to operate in mode 0 at high speed. Character length is set to 8 bits and SYN is set as synchronization character.
Bytes 2-5 of conversiontable are ignored by R40X.

Generally a controlmessage causes the controller to be cleared. The receiver sets statusbit 0 in the answer if the modemsignal DATA SET READY is off. All other statusbits are zero.
The transmitter will always answer a controlmessage with status zero.

TRANSPUT

a. RECEIVER

Two modems are recognized:

1 receivercontroller is cleared at timeout

9 receivercontroller is allowed to run after timeout, and when next inputmessage arrives, no startcommand is executed to the controller. So the next message must refer to the same buffer.

Bytecount of the message defines the size of the block including < start > and < bcc > so the number of datacharacters is moved one character forward in the buffer, destroying the character < start >. The value of this character may be seen in the status.

The receiverdriver checks the state of the transmitterdriver, so that it will not start to count timerperiods until the transmitterdriver has terminated. If no data arrives, receiver will answer with timeoutstatus timer 1/10 seconds later.

Bytecount of the answer is the number of characters received, including < start > and < bcc >.

b. TRANSMITTER

The buffer to be transmitted must have the following format when R40X receives the outputmessage:

$\left\{ \text{SYN} \right\}_{5}^{5}$ < start > < data > < tail >

where < tail > consists of three bytes. The transmitterdriver will then accumulate CRC 16 over < data > and place the result followed by a PAD character, in < tail > part. It is seen, that the number of characters in < data > is bytecount - 9.
A PAD character has the value 255.

Two modes are recognized:

3: normal mode. The transmission will be started immediately after the CRC 16 has been computed.

7: delayed transmission mode. The transmitterdriver will wait timer 1*20 msec before transmission is started. This mode may be used for testpurposes, but may also be valuable on half duplex lines.

The bytecount of answer is the number of characters transmitted. It will be the same as the bytecount of the message, unless a hard error occurred.

STATUS

The following statusbits are used:

0: dataset not ready
1: carrier lost
2: calling indicator
3: received startcharacter was SOH
4: received startcharacter was STX
5: received startcharacter was ENQ
6: illegal operation
7: startcharacter different from SOH, STX and ENQ
9: data late
10: blockcheck error
14: timer

bits 1, 2, 3, 4, 5, 7 and 10 are not used by transmitter.

TESTOUTPUT

When testoutput is produced, all characters received or transmitted will be stored cyclically in a core buffer in the user program. The buffer is defined in bytes 6-9 of the conversiontable. Convtab(6:7) is first wordaddress of the buffer, and will also be used as current pointer in the buffer. Current pointer is updated by driver. Convtab (8:9) is top wordaddress of the buffer.

If first > = top then no testoutput will be produced.

The format of the testrecord is:

+0	identification of driver. 11111 for receiver, 22222 for transmitter
+1	status of the answer
+2	operation of the message
+3 ff	the transmitted or received block including < start > and < bcc >.

The block will be stored with one character in each word.

Testoutput is produced immediately before the answer is sent.

CHARABAND PRINTER DRIVER

General Description

Control and output messages are accepted. Input messages are treated as control messages.

Control

Reservation, Conversion, and Position are accepted. Conversion table address should be a byte address. The characters to be output are by hardware converted as

conv char : = byte (conv table + char) ;

Note that you have to set up the conversion table before you open your zone to the printer.

The Charaband alphabet is:

	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
0	SP		SP	0	@	P	`	p	¢	Æ	æ		А	Р		
1			!	1	A	Q	a	q	¢	Ø	ø		Б	С		
2			"	2	B	R	b	r	¤	Å	å		В	Т		
3			#	3	C	S	c	s	¤	Ä	ä		Г	У		
4			$	4	D	T	d	t	‡	Ö	ö		Д	Ф		
5			%	5	E	U	e	u	¬	Ü	ü		Е	Х		
6			&	6	F	V	f	v	¶	É	é		Ж	Ц		
7			'	7	G	W	g	w	^	Á	á		З	Ч		
8			(	8	H	X	h	x	ƒ				И	Ш		
9			)	9	I	Y	i	y					Й	Щ		
10	LF		*	:	J	Z	j	z	Illegal				К	Ъ		
11			+	;	K	[	k	{					Л	Ь		
12	FF		,	<	L	\	l	|	Illegal				М	Ю		
13	CR		-	=	M	]	m	}	Illegal				Н	Я		
14			.	>	N	↑	n	~					О			
15		SP	/	?	O	←	o	_					П			Ignore

761008

Characters shown in the table, but not defined to be on an actual charaband must be interpreted as spaces.
Note the characters from 192 to 221 are cyrillic.

In output mode 7 it is possible to use margin, i.e. the number of spaces to be put in front of every print line. The position command only has effect if output mode is 7. Then the mess2 part (file) of the control message gives the margin, i.e. setposition(zone,margin,1).

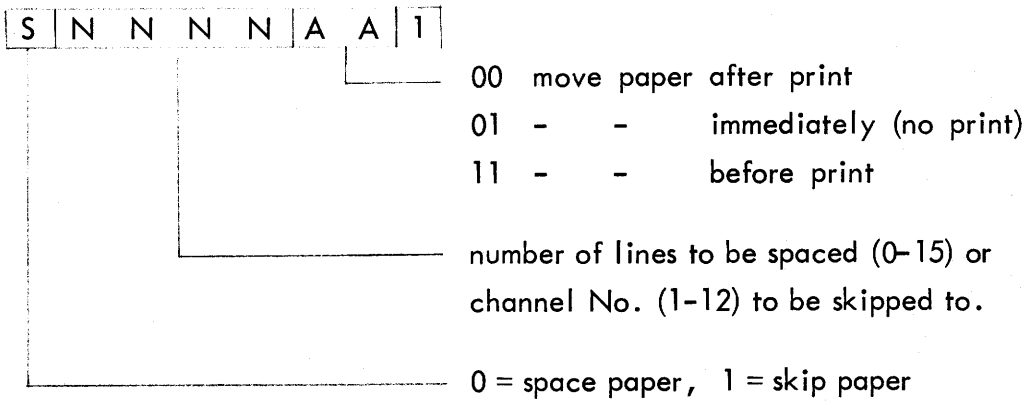
Output Operation

It is possible to output 136 characters per line.

Four modes of operation exist:

3 : The converted characters are printed.

7 : The first byte of the output messages is interpreted as a carriage control word (ccw). It must be built up as



15 : Output to the Direct Acces Vertical Format Unit (VFU), which may exist instead of a carriage control tape. The line spacing is 6 lpi (normal).

The data output format is:

1st byte : number of lines in VFU (max. 144, starting at line No. 1)

2nd byte : line No. in VFU, for which the next two bytes define channel holes.

3rd byte : a one in one of the 6 rightmost bits means a hole in channels 1-6, the two first bits must be zero.

	0	0	X	X	X	X	X	X
channel No.:			6	5	4	3	2	1

X=1 means a hole.

4th byte : a one in one of the 6 rightmost bits means a hole in channels 7-12, the two first bits must be zero.

0	0	X	X	X	X	X	X
		12	11	10	9	8	7

channel No.: X=1 means a hole

The format of the next 3 bytes is the same as for byte 2, byte 3, and byte 4, and so on. The line numbers must be ascending. It is not necessary to output data to lines without holes. Note that if you get a status error after loading the VFU data, you have to reload all the VFU data. If the VFU load succeeds, then the VFU is positioned to top of form, i.e. a hole in channel 1.

31 : As for mode 15, but the line spacing is 8 lpi.

Status

1b0	:	disconnected,	the Printer disconnected, Power off, not ready, VFU error, no hole in VFU channel 1.
1b1	:	off-line,	the Printer is in off-line state, VFU error, not ready.
1b3	:	device mode 1,	channel 12 in carriage control has been encountered.
1b4	:	device mode 2,	direct access VFU exists.
1b6	:	illegal,	device reserved.
1b8	:	block error,	paper fault or overprinting of a line has occurred more than 8 times.
1b9	:	data late,	printer not ready (e.g. a papermove or a another command not finished after a break situation.
1b10	:	parity error,	parity error during loading or printing of a line, or a ccw contains a zero in the least significant bit.
1b11	:	end medium,	end of paper.
1b14	:	timer,	time out.

RC 3682 ASYNCHRONOUS MULTIPLEXER DRIVER

1.1. General Description

Control, input, and output messages are accepted.

The driver exists in 3 versions with 1, 2, or 3 AMX devices controlled by one driver process.

1.2. Channels

Each AMX comprises 8 full or half duples channels. Logical channel numbers 0:7, 16:23, and 32:39 are used for the first, second, and third AMX respectively.

1.3. Lines

A channel is represented by a pair of lines (imitating the half duplex lines of the RC 3683 TMX).

Input line No. = channel No.

Output line No. = channel No. + 8

For 1, 2, or 3 AMX devices, the range of legal line numbers will be 0:15, 0:31, or 0:47.

1.4. Half/Full Duplex

For full duplex operation input messages should be sent to the input line No. and output messages to the output line No.

For half duplex operation both input and output messages should be sent to the input line No. (see 6.4.4.).

2.1. Initialization

A message with mess0 = -1 is treated as a dummy message (3.2). The channel or line parameters mentioned below may be changed by control messages:

Channel character format	(5.4. position)
DATA TERMINAL READY	(5.2 reservation)
Receiver bitrate	(5.4. position)
Transmitter bitrate	(5.4. position)
Line conversion table	(5.3. conversion)
Line timer 0, timer 1	(5.4. conversion)

2.2. Initial Values

At load time, the below default values are valid:

Channel character format = 2 stopbits,
no parity, 8 data bits in character.

DATA TERMINAL READY is cleared.

Receiver bitrate = transmitter bitrate = 2400 bits/sec.*

Line conversion table = 0 (no input conversion).

Timer 0 = timer 1 = 30 seconds. See 5.3 position.

3.1. Line Addressing

For all normal messages, the line number is given as the left byte of mess0:

Mess0 (0:7) = line

3.2. Special (Dummy) Messages

A message with mess0 equal to zero or -1 is always accepted and answered with zero status.

---

* Liable to change.

4.1. Break, Power Restart

When a break process or a power restart occurs, all pending messages are returned with status disconnected. The channels are reset with cleared hardware buffers.

DATA TERMINAL READY is set to its latest current value.

The channel initiantion registers are set to their latest current values. (Default values or latest value set by position (see 5.3)).

5. Control

Reservation, conversion, position, and disconnect are executable. Termination and erase have no effect.

A control message with mess0 (operation) equal to zero is always accepted, and returned with zero status (may be generated by the I/O system).

A control message with mess0 equal to -1 is treated as a control message with mess0 equal to zero. (Initialization message for the RC 3683 driver).

All other messages will suspend input messages held by the line (See 6.4.1.).

5.1. Reservation

In case of output line and mess1 <> 0, DATA TERMINAL READY will be set.

5.2. Conversion

Mess2 is taken as a byte address of an input conversion table. The table should start on an even byte (full word boundary).

Mess2 = 0 means no conversion.

5.3. Position

Mess2 is taken as new timer 0, timer 1 value for the specified line.

Timer 0								Timer 1							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Timer 0 and Timer 1 are measured in seconds between 0 and 255.

For input operations timer 0 is the maximum time allowed before the first character and timer 1 the maximum time allowed between characters.

For output operations timer 0 is the maximum allowable time for the output operation.

Mess3 is ignored when positive or zero.

A negative mess3 is used to define the channel characteristics in the following format:

1	X	X	S	P		L		rec. bitrate				xmit bitrate			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

where

X means 'not used' and

S specifies the number of stop bits to be used:

bit 3	Number of Stopbits
0	1 stopbit
1	2 stopbits

P specifies the parity condition

bit 4	bit 5	Parity
0	0	Odd Parity
0	1	Even Parity
1	0	No Parity
1	1	No Parity

When even or odd parity is specified, the parity bit (most significant bit) will be supplied by the transmitter and checked and removed by the receiver.

L specifies the character length of the channel excluding a possible parity bit:

bit 6	bit 7	Char. Length
0	0	5
1	0	6
0	1	7
1	1	8

Rec. bitrate and xmit bitrate specifies the receiving and xmitting line transmission bit rate respectively:

Bit 8:11 Bit 12:15	Bit Rate b/s	t_{EOT} ref. 6.4.4
0000	9600	80 ms
0001	4800	80 ms
0010	2400	80 ms
0011	1200	80 ms
0100	600	80 ms
0101	300	120 ms
0110	220	160 ms
0111	200	160 ms
1000	150	200 ms
1001	134.5	240 ms
1010	110	280 ms
1011	75	360 ms
1100	50	520 ms
1101	40	640 ms
1110	40	640 ms
1111	40	640 ms

Input at 9600 bps should be intermittent as the channel throughput is limited by the 40 msec driver time cycle :

Input : :	max 32 chars/cycle	= 800 cps,
Output : :	34 chars/cycle	= 850 cps.

5.4. Disconnect

In case of output line, DATA TERMINAL READY is cleared, and the channel is stopped.

Input/attention messages held by the line are returned with their present status (see 8).

6. Input

Mess0 (8:15):

1	normal input
17	attention
33	input continued
+4	soft parity check for even parity of characters (see 6.5.)
+8	echoing of input.

The channel receiver is kept in receive mode regardless of buffer changes. In half duplex operation (output messages sent to the input line) the channel receiver is stopped during output and then restarted without loss of characters that arrived before the output operation. When no input message is present, characters received are buffered in the device (max. 32 characters per channel).

6.1. Attention

A current attention message will accept all characters. Only the terminating attention character will be delivered.

The message will be returned when

1. an attention character is met,
2. control with disconnect arrives (zero status),
3. a modem signal status error is detected.

6.2. Normal Input

When a normal input message is activated, the input buffer is cleared for characters.

The message will be returned when

1. an attention or termination character arrives,
2. the specified number of characters have been stored,
3. a timeout occurs,
4. control with disconnect arrives,
5. a modem signal status error is detected.

6.3. Input continued

Characters previously buffered in the channel receiver are accepted. If this previous input exceeds 32 characters or demands echo exceeding 32 characters, the message will be returned with status character lost.

The message is returned as described for normal input.

6.4. Operation of a Line

The driver event queue is treated as a set of independent line event subqueues.

6.4.1. Input Suspension

A current input message may be temporarily suspended, being replaced by some later

message for the line (subqueue look ahead). Attention messages are suspended by any later message for the line. Other input messages are suspended by any later control or (half duplex) output message for the line.

When the current message has been returned, the oldest item in the line subqueue will be the next current message. (A previously suspended input message will be resumed with character buffer erased).

6.4.2. Kill Output

Termination by attention on an input line (full duplex) will stop concurrent channel output. The output message is returned with status attention.

6.4.3. Echo

If the output line is idle when the channel transmitter is requested for the first echo character, the input line will delay the corresponding output line (echo reservation) until input termination. Otherwise (output line busy), the input message is returned with status character lost.

6.4.4. Transmission Mode Considerations

When a channel is idle, the channel receiver is kept in receive mode.

The channel transmitter is started (causing the modem signal REQUEST TO SEND to change from OFF to ON) when an output operation is initiated on an output line or an input operation specifying echo is initiated on an input line.

When an output operation is initiated on an input line (half duplex), the channel receiver is stopped and the channel transmitter started. After the output operation has been terminated, the channel transmitter is stopped within t_{EOT} msec (see table in 5.3) and the channel receiver restarted without loss of characters in the hardware input buffer. When the channel transmitter is stopped, the modem signal REQUEST TO SEND is changed from ON to OFF.

When an input operation specifying echo is sent to an output line, the echo specification is ignored.

An input operation specifying echo sent to an input line in half duplex mode will cause unpredictable results.

A disconnect control message sent to an output line will stop the channel transmitter.

6.4.5. Modem Status

When the modem signal CARRIER changes from ON to OFF during an input operation, the input message is returned with status 1B3: carrier off. When an output operation sent to an input line is initiated and CARRIER is ON, the output message is returned with status : 1B8 (carrier on).

Both DATA SET READY being OFF and CALLING INDICATOR being ON will terminate any transput message with the corresponding status bit set in the answer. During input the error will be detected within 40 msec and during output within 1 second.

A BREAK received will terminate an input operation only, and status 1B11 (break received) will be set in the answer.

6.5. Treatment of Characters

When a character is received, parity check is performed as defined for the receiving channel and if a parity error is detected, the valve 26 (decimal) is substituted. When parity checking is specified, the parity bit (the most significant bit) is cleared.

When the mode of the input operation specifies soft parity check and the hardware parity condition is: no parity, a software parity check is performed, if a parity error is detected, the valve 26 (decimal) is substituted, and the 7 least significant bits are taken as character value.

When the conversion table address is non-zero, conversion i performed as table lookup giving a class and a value byte for each character.

convtab//2

class	value

; value "0"

⋮

class	value

; value "char"

⋮

class = 0 — Normal character, the value is delivered and if specified echoed.

class 128 <> 0 — Attention character. Input is terminated and the input or att buffer returned with status attention. If an output operation is in progress, it will be stopped and returned with status attention. If specified the value is delivered and if specified echoed.

class 64 <> 0 — Termination character. Input is terminated with this character. If specified the character is delivered and if specified echoed.

class 32 <> 0 — Special echo, the word class concatenate value bits 3 to 15 are taken as a displacement in words from convtab//2 to a subtable:

convtab//2:

⋮

XX1	displacement

; value "char"

displacement

⋮

subclass	value
echo 1	echo 2
echo 3	echo 4

⋮

128

subclass	128 <> 0	Erase current input buffer.
subclass	64 <> 0	Erase last character if any from input buffer. If the input buffer is empty, the string : echo 1, echo 2, --- is not output.
subclass	32 <> 0	Value is not delivered.

Value gives the value delivered, and echo 1, echo 2, - - - is a string terminated by 128, which is echoed on the output line if specified.

class	16 <> 0	Mark, sets the mark answer status bit.
class	8 <> 0	Shift character, conversion table address is changed as follows :

convtab : = convtab +
(if class 4 = 0 then value
else -value) * 2,

The conversion is repeated.

7. <u>Output</u>

operation (0:7) = line

operation (8:15:

3	write, characters are output until 128 is found.
67	break, break is output during (messl * 40) msec.

An output message is returned when

1. a character with valve 128 is to be output,
2. control with disconnect arrives,
3. a modem signal status error is detected,
4. a timeout occurs.

8. Answer

Mess2 of an answer contains the line number (except mess2 = 1B6 by unknown line).

Status:

1B0	disconnected (power restart)
1B1	data set not ready
1B2	calling indicator
1B3	carrier lost (during input)
1B4	mark
1B5	attention received
1B6	line unknown
1B7	character lost
1B8	carrier on (half duplex output only)
1B10	parity or stopbit error
1B11	break received
1B14	timeout

RC 74060 ASYNCHRONOUS MULTIPLEXOR DRIVER

1.1. General Description

Control, input, and output messages are accepted.

The driver exists in 2 versions with 1 or 4 RC 74060 devices controlled by one driver process.

1.2. Channels

Each RC 74060 comprises 4 full duplex channels. Logical channel numbers 0:3 and (0:7, 16:23) are used for 1 and 4 RC 74060 devices respectively.

1.3. Lines

A channel is represented by a pair of lines (imitating the half duplex lines of the RC 3683 TMX).

Input line No. = channel No.
Output line No. = channel No. + 8

For 1 or 4 RC 74060 devices, the range of legal line numbers will be (0:3,8:11) or 0:31.

1.4. Full Duplex

The driver supports full duplex operation only and input messages must be sent to the input line No. and output messages to the output line No.

Input messages sent to output lines and output messages sent to input lines will be returned with status Illegal.

Control messages may, however, be sent to input or output line numbers with the same result.

2.1. Initialization

A message with mess0 = -1 is treated as a dummy message.

No initialization is needed since all channel and line parameters are defined by hardware jumpers.

3.1. Line Addressing

For all normal messages, the line number is given as the left byte of mess0:

mess0 (0:7) = line

3.2. Special (Dummy) Messages

A message with mess0 equal to zero or -1 is always accepted and answered with Zero status.

4.1. Break, Power Restart

When a break process or a power restart occurs, all pending messages are returned with status Disconnected.

5. Control

Conversion, position, and disconnect are executable. Reservation, termination, and erase have no effect.

A control message with mess0 (operation) equal to zero is always accepted, and returned with Zero status (may be generated by the I/O system).

A control message with mess0 equal to -1 is treated as a control message with mess0 equal to zero. (Initialization message for the RC 3683 driver).

All other control messages will suspend input messages held by the line (see 6.4.1.).

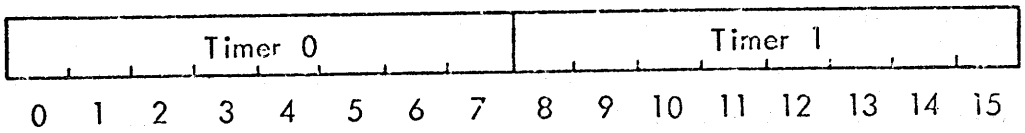
5.1. Conversion

Mess2 is taken as a byte address of an input conversion table. The table should start on an even byte (full word boundary).

Mess2 = 0 means no conversion

5.2. Position

Mess2 is taken as new timer 0, timer 1 value for the specified line.



Timer 0 and Timer 1 are measured in seconds between 0 and 255.

For input operations timer 0 is the maximum time allowed before the first character and timer 1 the maximum time allowed between characters.

For output operations timer 0 is the maximum allowable time for the output operation.

Mess3 is always ignored.

5.3. Disconnect

Input/attention messages held by the line are returned with their present status (see 8).

6. Input

Mess0 (8:15)

1 normal input
17 attention

33 input continued
+4 even parity check
+8 echoing of input

The channel receiver is kept in receive mode regardless of buffer changes.

When no buffer is present, characters received are buffered in a local buffer (max. 16 characters per channel).

6.1. Attention

A current attention message will accept all characters. Only the terminating attention character will be delivered.

The message will be returned when

1. An attention character is met
2. Control with disconnect arrives (Zero status)

6.2. Normal Input

When a normal input message is activated, the input buffer is cleared for characters.

The message will be returned when

1. An attention or termination character arrives
2. The specified number of characters have been stored
3. A timeout occurs
4. Control with disconnect arrives

6.3. Input Continued

Characters previously buffered in the local buffer are accepted. If this previous

input exceeds 16 characters or demands echo exceeding 16 characters, the message will be returned with status Character Lost.

The message is returned as described for Normal Input.

6.4. Operation of a Line

The driver event queue is treated as a set of independent line event subqueues.

6.4.1. Input Suspension

A current input message may be temporarily suspended, being replaced by some later message for the line (subqueue look ahead). Attention messages are suspended by any later message for the line. Other input messages are suspended by any later control message for the line.

When the current message has been returned, the oldest item in the line subqueue will be the next current message. (A previously suspended input message will be resumed with character buffer erased).

6.4.2. Kill Output

Termination by attention on an input line (full duplex) will stop concurrent channel output. The output message is returned with status Attention.

6.4.3. Echo

If the output line is idle when the channel transmitter is requested for the first echo character, the input line will delay the corresponding output line (echo reservation) until input termination. Otherwise (output line busy), the input message is returned with status Character Lost.

6.5. Treatment of Characters

When a character is received, parity check is performed if specified and if a parity

error is detected, the valve 26 (decimal) is substituted. When parity checking is specified, the 7 least significant bits are taken as character value.

When the conversion table address is non-zero, conversion is performed as table lookup giving a class and a value byte for each character.

convtab//2

class	value

; value "0"

⋮

class	value

; value "char"

⋮

class = 0 — Normal character, the value is delivered and if specified echoed.

class 128 <> 0 — Attention character. Input is terminated and the input or att buffer returned with status attention. If an output operation is in progress, it will be stopped and returned with status attention. If specified the value is delivered and if specified echoed.

class 64 <> 0 — Termination character. Input is terminated with this character. If specified the character is delivered and if specified echoed.

class 32 <> 0 — Special echo, the word class concatenate value bits 3 to 15 are taken as a displacement in words from convtab//2 to a subtable:

convtab//2:

⋮

XX1	displacement

; value "char"

displacement

⋮

subclass	value
echo 1	echo 2
echo 3	echo 4

⋮

128

subclass	128 <> 0	Erase current input buffer.
subclass	64 <> 0	Erase last character if any from input buffer. If the input buffer is empty, the string echo 1, echo 2, - - - is not output.
subclass	32 <> 0	Value is not delivered.

Value gives the value delivered, and echo 1, echo 2, ----- is a string terminated by 128, which is echoed on the output line if specified.

class	16 <> 0	Mark, sets the mark answer status bit.
class	8 <> 0	Shif character, conversion table address is changed as follows :

convtab : = convtab +
(if class 4 = 0 then value
else -value) * 2,

The conversion is repeated.

7. Output

operation (0:7) = line

operation (8: 15) :

3 write, characters are output until 128 is found.

An output message is returned when

1. A character with valve 128 is to be output
2. A timeout occurs.

8. Answer

Mess2 of an answer contains the line number (except mess2 = 1B6 by unknown line).

Status:

1B0	Disconnected (power restart)
1B4	Mark
1B5	Attention Received
1B6	Line Unknown
1B7	Character Lost
1B10	Parity Error
1B14	Timeout

DIGITAL OUTPUT TERMINAL

General

The driver may handle 4096 digital output terminals. Two different modes of treating the channels are possible. The first is a static operation where the output level is changed on an explicit message from the user. The second is a pulsing operation where the channels in the message are switched ON and OFF at the time intervals specified by the user.

Control

Position and disconnection are accepted.

Position — The time intervals for the pulse operation is redefined.

mess0.buf	operation
mess1.buf	Ø
mess2.buf	ON-interval (one state)
mess3.buf	OFF-interval (zero state)

where 'interval' is the number of time slices (>0) of 20 msec. Then

(interval -1) < = t < interval

where t *20 = the ON/OFF time in msec.

The initial values are 25 (~500 msec) for both intervals.

Disconnection — If a share with pulsating channels is in progress, the execution is stopped when the next OFF-interval is running out and the answer is returned.

Output

mess0.buf	operation
mess1.buf	bytecount = (2, 4, 6, ..., 1)
mess2.buf	byteaddress
mess3.buf	Ø

where l/2 = number of words in the share, i.e. number of channels.

If bytecount or byteaddress is odd then status : = odd address and the answer is returned.

Mode:

3: Set channels.

Each 2 adjacent bytes in the addressed share should contain an output channel number, the output value, and the static/pulsing mode.

If the bytes addressed contain all ones (i.e. -1) the answer is returned immediately.

word (bytecount/2-1):	bit (0)	=	output value,
	bit (1)	=	if 0 then set channel static, if 1 then set channel pulsing,
	bit (2:3)	=	not used,
	bit (4:15)	=	channel number

If bit (1) = 0 the value is sent to the addressed channel number. Then the channel address is searched in the pool of pulsing channels. If the channel is found, it is removed from the pool.

If bit (1) = 1 the channel addressed is inserted in the pool of pulsing channels if it does not already exist in the pool. If the pool is full, then status : = pool full, and the answer is returned immediately.

7: Set pool of pulsing channels.

The sender allocates a core area to the driver. It will be used as a pool of pulsing channels. The last word of the pool is used as a pointer, i.e. the minimum size is 4 bytes.

The size of the pool should be equal to the maximum number of channels which are pulsing at the same time + 1.

The channel numbers in the pool are sent to the digital output terminal with alternating values with a loop-timeslice defined in the control (position). When receiving a message in mode 7, it is tested if another message in this mode is in progress. If this is so, the old message is returned at the end of the next OFF-period, and the new one is executed.

To stop a pulsating operation a disconnection command should be sent or a message in mode 3 containing the pulsing channel address which should be stopped, and the stop-value.

Input

Input message has no meaning. Status (input message) is set and the answer is returned.

Answer

mess0.buf	status
mess1.buf	number of channels processed
mess2.buf	Ø
mess3.buf	Ø

Status

1b4	pool full.
1b5	input message, an inputmessage has been received.
1b8	odd byteaddress.

DIGITAL SENSE TERMINAL

General

The driver may handle 4096 digital sense terminals. The scanning of the channels is done synchronously with a time interval defined by the user. When scan is running, the user should send input messages to the driver to get the numbers and the state of those channels which have changed after the last input message.

Control

Position and disconnection are accepted.

Position

The user may redefine the first channel in scan (start channel) and the number of channels in each scan (number of channels).

When receiving an outputmessage the startchannel := O and the number of channels := number of channels on the installation.

The scanning cycle time may be redefined by the user by setting mess2.buf to the negative number of time slices of 20 msec each.

	scanning window	time slice
mess0.buf	operation	operation
mess1.buf	Ø	Ø
mess2.buf	start channel	-number of time slices
mess3.buf	number of channels	Ø

If start channel + number of channels > 4096 or > number of bytes in output buffer then status (end medium) is set and the answer is returned.

If m = number of time slices (>0) then

$$(m-1)*20 <= t < m*20 \quad \text{(msec)}$$

where t = time between adjacent scans.

The initial value of m is 25 (= 500 msec).

Disconnection Stop scanning. All messages in the event queue are returned. Messages different from output are returned with status (empty return) and bytecount := 0.

Input

The user should have one or more input messages pending. Every time a scan has been completed, the first input message will most likely be executed.

If a channel has changed, the channel number, the channel value, and a bit telling whether it is a new or an old piece of information is saved in the addressed share, 2 bytes at a time. The channel is now able to accept changes again. This will go on until either 'number of bytes' channels have been searched through or the addressed input share is full.

In the latter case status (block full) is set and the answer is returned. The search will continue from the point where it was stopped when another input message is found in the message queue.

If an output message or an control message with new specification of first channel and number of channels in a scan is received, the rest of the information is lost.

If no changed channel is met, the message is still pending.

If no changes are met after 120 scans, the first input message is returned with status (timeout).

mess0.buf	mode
mess1.buf	bytecount (= 2, 4, ..., l)
mess2.buf	byteaddress
mess3.buf	Ø

where l/2 = number of words in the share.

The format of the information in the addressed input share is:

word:	b(0)	channel value
	b(1)	old information. This bit indicates that the channel has been scanned one or more times after a change has been accepted. The actual value of the channel may have been changed again, but this will be recognized in the next scan.
	b(2)	if b(1) = 1 then the last sensed channel value.
	b(3)	0
	b(4:15)	channel number.

Output

The information about the state of the channels is stored in an output buffer. The number of bytes should be equal to the number of channels on the installation.

mess0.buf	operation
mess1.buf	bytecount (= number of channels)
mess2.buf	byteaddress
mess3.buf	Ø

If an output buffer exists already, it is returned.

Bytecount and byteaddress must be modulo 2. If not, the answer is returned with status (odd byteaddress).

The output buffer should be initiated as follows (the bit numbers are on byte level):

b(0)	initial value of the channel in question,
b(1:3)	0,

b(4) idling channel, the channel is skipped during scan,

b(5) scanmode,

0 : immediate channels, each change is accepted uncritically.

1: double sense, if a change is sensed, the channel value of the subsequent scan must be equal to the changed value to be accepted, else it is regarded as noise.

b(6:7) 0

Answer	input	output
mess0.buf	status	status
mess1.buf	number of changed channels *2	number of channels on installation
mess2.buf	Ø	Ø
mess3.buf	Ø	Ø

Status

b3	timeout	, no changes met during 120 scans.
b4	block full	, the input share is too small to hold all information
b5	empty return	, returned messages after a break or a disconnection. In the latter case this is only true if message is different from an output message.
b8	odd byteaddress	
b11	end medium	, the upper limit for the specified scanning window > 4096 or > the number of bytes in output buffer, or output buffer is missing (output buffer size = 0).

GENERAL SYNCHRONOUS COMMUNICATIONS DRIVER

1. General Description

All message types are accepted. Input and output messages are rejected if conversion table has not been specified or the driver has not been reserved by the process.

1.1. Control Messages:

Reservation	is accepted.
Conversion	The driver parameters and functions are set up according to the conversion table. See Section 2. Signal (V24) "dataterminal ready" is turned ON after a 200 ms delay if "dataset ready" is not ON.
Termination	ignored.
Position	ignored.
Disconnection	The signal (V24) "dataterminal ready" is turned OFF, thus disconnecting the line. The conversion table is set to not used.
Erasure	ignored.

1.2. Input Messages

0	34	7 8	13
OFFSET	TIMEOUT	RING	01

OFFSET : 0 or 1. The given offset is used.

TIMEOUT: Value in the interval 0..6.

CONVT. TIMEOUTS [TIMEOUT] is used as the between-blocks timeout, and
CONVT. TIMEOUTS [TIMEOUT + 1] is used as blocklength timeout.

RING:	0	No wait for calling indicator, simple input message.
	<> 0	Wait for calling indicator. CONVT. TIMEOUTS [TIMEOUT] is used as maximum wait time.

A normal (RING = 0) input message starts the receiver hardware and initiates processing by the input program at INADDRESS.
The receiver is sensed at least every 20 msec for incoming data, which are then processed. (See GET. sect. 2.4.).

The receiver hardware is stopped when one of the following events occurs:

1. TRMBLK (terminate block) is called (see section 2.3)
2. Error status occurs:

   CARRIER LOST } before all characters
   DATASET NOT READY } needed were input

3. POWER BREAK.
4. Input program illegal function.
5. TIMEOUT occurs.
6. (bytecount - OFFSETSIZE) bytes were received.

Bytecount returned is number of bytes generated by input program (see 2.1., description of OFFSETSIZE). The contents of the buffer is without parity bits.

1.3 Output Messages:

0	3 4	7 8	13
OFFSET	TIMEOUT	DIRECT	1 1

OFFSET	0 or 1. The given offset is used
TIMEOUT	Value in the interval 0..7. CONVT. TIMEOUTS [TIMEOUT] is used as a maximum timeout.

DIRECT	0	Initiate the output program.
	<> 0	Transmit data in the buffer without processing by output program. Used e.g. when retransmitting a previously transmitted block.

A normal (DIRECT = 0) output message initiates processing by the output program at OUTADDRESS. The transmitter hardware is started when the processing is finished (TRMBLK and no errors).

Bytecount given is number of databytes + OFFSETSIZE. First databyte is assumed to be positioned at firstbyte + OFFSETSIZE. If the output program does not fetch all databytes, status bit 8 (block syntax) is added to statusword and transmitter hardware is not started.
Bytecount returned is actual number of bytes given to the transmitter logic, i.e.

DIRECT = 0 Number of bytes generated by output program.

DIRECT <> 0 Bytes transferred.

(See 2.1., description of OFFSETSIZE).

A PAD character (all ones) is generated by the driver when TRMBLK is called.

1.4. Status Bits

Bit	Description	Generated by
0	Dataset not ready	CCITT V.24 Circuit 107 1)
1	Carrier lost (input)	CCITT V.24 Circuit 109 1)
2	Calling indicator (input)	CCITT V.24 Circuit 125 1)
3, 4, 5	Available to user	SETSTA (page 22.27)
6	Reserved by other program Not reserved Conversion table not specified Bytecount ≤ offset Illegal function in program	central logic
7	Hardware overrun (multiplexer)	1)
8	Bytes not processed (output)	TRMBLK (page 22.22)
	Share too small (input)	GET (page 22.23)
9	Offsetsize too small	PUT (page 22.24) BEGSTR (page 22.27)
	Data late	1)
10	Character parity (input)	GET (page 22.23)
	Block check error (input)	COMPBC (page 22.30)
11	String length overflow (input)	TRMSTR (page 22.28)
12	Used by I/O system	
13	Driver missing - generated by I/O system	
14	Timeout	1)
15	Used by I/O system	

Note 1) : These bits (0,1,2,7,9,14) are generated by hardware malfunction during TRMBLK (page 22.22), PUT (page 22.24), GET (22.23) and functions using these.

The SETSTA (set status bit - page 22.27) function will set any of these bits. Caution is required when using bits other than 3,4,5, and 7, especially bit 13 and 15, as these are processed by the I/O system, and are generated by the driver or the I/O system.

Bit 1 is returned on input only if the event occured inside a block.

Bit 8 (output) is returned if not all bytes were processed by means of GET etc.

2. Conversion Table Format

The conversion table is mandatory.

It is not used for code conversion, but contains controller parameters, description of elements of transmission code, and programs for treatment of input and output data such as parity check, block check, and deletion/addition of control sequences.

2.1. Controlling Parameters:

The conversion table is structured as follows:

Rel. Byte Address	Byte Contents	
+0,1	TIMING (see below)	
+2	Character length (including parity bit)	0 undefined 1 7 bit 2 6 bit 3 8 bit
+3	Synchronization character (without parity)	
+4	Character parity on communication line	0 no 1 even 2 odd
+5	Substitute character. This character is delivered instead of parity infected characters in input	
+6	Block check method (see below)	0 no check 1 LRCE 2 LRCO 3 SUM 4 NEGSUM 5 CRC 16 6 CRC 12
+7	Zero	
+8,9	Offset 0 (even value) to allow expansion of data.	
+10,11	Offset 1 (even value)	
+12 ... +19	Timeouts. This table contains timercounts in 0.1 sec units (maximum 25.5 sec.). It is used in connection with input and output operations (see section 1.2. and 1.3.)	

(Conversion table continued)

+20,21	Input program start address. Absolute byte address.
+22,23	Output program start address. Absolute byte address.

TIMING is a counter which is zero when messages are processed by the driver. As soon as the event queue is empty, it is increased by 1 every 100 msec.

BLOCKCHECK METHOD: Determines which blockcheck character(s) are to be added when output is terminated (TRMBLK), or to be expected as last character(s) input:

0:	NO	No block check characters.
1:	LRCE	One block check character making the
2:	LRCO	longitudinal parity EVEN or ODD. Note: The block check character has the parity specified by PARITY (+4).
3:	SUM	One block check character. When using
4:	NEGSUM	SUM, the summed character should be equal to the block check, in NEGSUM the summed character and the block check should have zero sum. Parity of block check characters is determined by PARITY.
5:	CRC 16	A polynomial double check character is used.
6:	CRC 12	Error when generated and received characters are not equal.

OFFSETS determines the maximum number of additional characters generated at any instant by the program. It is assumed even.

If at any instant

sum of generated - sum of removed > OFFSETSIZE

status 1b9, offsetsize too small, is delivered.

The following table shows bytes generated/deleted:

+ indicates generated
- indicated deleted

Operation		Receival	Transmitting
TRMBLK		0	+ 1 (PAD)
GET		- 1	- 1
PUT		+ 1	+ 1
BEGSTR		+ 2/ + 1	- 1/0
COMPBC:			
	NO	0	0
	LRC/SUM/NEGSUM	- 1	+ 1
	CRC12/CRC16	- 2	+ 2
all others		0	0

The bytecount which is returned to the main program is obtained by summing all positive numbers, i.e. sum of generated.

The following table shows combinations of parity, character length, and block length method.

	Check Method						
Length & Parity	NO	LRCE	LRCO	SUM	NEGSUM	CRC 16	CRC 12
6 NO	*	*	*	*	*		*[1]
6 EVEN	*	*	*	*	*		
6 ODD	*	*	*	*	*		
7 NO	*	*	*	*	*		
7 EVEN	*	*	*	*	*		
7 ODD	*	*[2]	*	*	*		
8 NO[3]	*	*	*	*	*	*[1]	

1) The only sensible combinations are 6 bit-CRC12 and 8 bit-CRC16. The last combination is used by IBM for BSC transmission in EBCDIC code.

2) ISO R1155 and R1177.

3) As maximum number of transmitted bits per character is 8, no parity bit is available.

2.2. Input and Output Programs

As the SCC 7Ø2 controller performs only the most necessary functions, a simple machine is simulated in the driver, thus enabling all simple character-for-character processing on driver level. This driver simulated machine has a set of instructions suited to operate on data strings as they occur in most data transmission protocols.

The instruction set includes instructions for:

1. Single character input and output with automatic parity addition and checking as specified.
2. Program flow control
3. Block check accumulation and control.
4. Substring functions.
5. Communication with the rest of the MUS system.

Section 2.3 is a short presentation of the instruction set, and section 2.4 contains the full, detailed description of the driver logic.

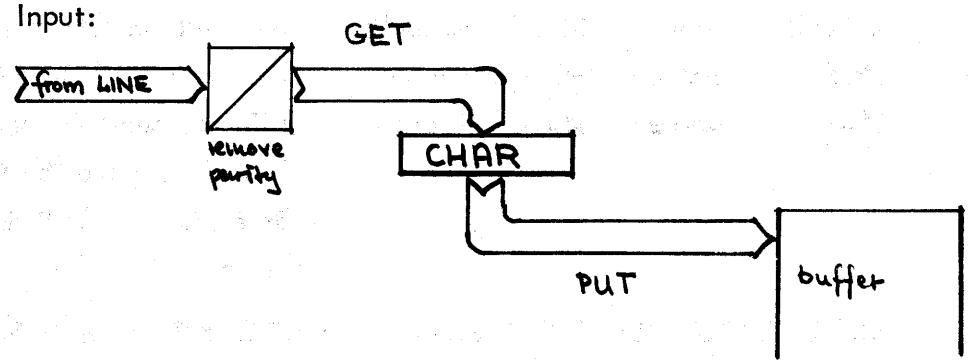
2.3. Summary of Instruction Set.

2.3.1 Character functions

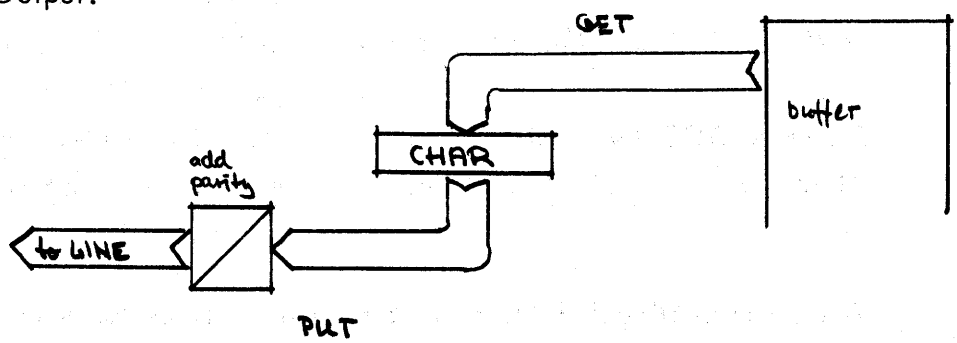
Name	Mnemonic
GET CHARACTER	GET
PUT CHARACTER	PUT
SAVE CHARACTER	SAVE
RESTORE CHARACTER	UNSAVE
LOAD VALUE	LOAD

The function, GET and PUT, depends on whether the data flow is from or to the transmission line.

Input:



Output:



GET always leaves a value in CHAR, PUT fetches a character value from CHAR, without destroying the contents. Parity bit is specified as shown previously.

SAVE saves the contents of CHAR in SAVECHAR, UNSAVE restores the contents from SAVECHAR.

LOAD sets a constant value in CHAR:

```
        LOAD 3          load on ASCII ETX into CHAR.
```

2.3.2.

Program flow control

GOTO <label>		Unconditional jump to <label>
IFEQ <value> GOTO <label>		Conditional jumps.<value>is compared
IFNEQ <value> GOTO <label>		to CHAR. If CHAR is equal (EQ),
IFLT <value> GOTO <label>		not equal (NEQ), greater than
IFGT <value> GOTO <label>		(GT), respectivly less than ('LT)
		<value> a jump is executed.
		Otherwise the instruction has no
		effect.
SET DCOUNT (SETCNT <value>)		DCOUNT is set to <value>
		e.g.
		SETCNT 80 sets DCOUNT to 80
IFMBIT <bitno> GOTO <label>		Jump if the bit is set in the trans-
		put operation.
Decrease DCOUNT and Jump if		DCOUNT is decrease by 1. If the
NONZERO (DECJNZ <label>)		new value is not equal to zero,
		a jump is done.

A loop outputting 8 SYN characters, may look as follows:

```
        LOAD      SYN          ; CHAR = value of SYN
        SETCNT    8            ; i: = 8
.OSYN PUT                      ; repeat PUT;  i: = i-1
DECJNZ OSYN                    ; until i = 0
```

(.OSYN means "define OSYN as label").

2.3.3. Block checking

The method used has been described previously.

Initialize Blockcheck (ZEROBC)	BCC is preset.
Accumulate Block check (ACUMBC)	Compute a new, intermediate block check value.
Complete Block check (COMPBC)	Receival: read one or two characters, and compare with accummulated BCC. Transmitting: Generate and output the final block check sequence.

2.3.4. Substring functions.

SCD will treat data structured as follows:

length	<length> characters

Three operations work on substrings:

Set length field size	SETLFS <value>
Start string	BEGSTR
End of string	TRMSTR

Their effect differs, depending on input or output:

BEGSTR Input: make room for a 2 or 1 byte long length field
Output: set DCOUNT to a proper value.

TRMSTR has only effect at input. PUT counts automatically, and the number of generated characters is put into the place, where BEGSTR made room.

SETLFS with value 0 determines a two bytes length field, 1 determines one byte.

2.3.5. Interfacing MUS

Set status bit	SETSTA <bitno>	sets the bit to one in the status word returned in answ 0
Set counter	SETSCN	set DCOUNT to the length of bufferarea specified together with operation (mess 1)
Terminate Block	TRMBLK	stop SCD and return an answer.

2.4. Execution times for SCD operations.

Following execurtion times for SCD operations are all in μs, and includes an operationfetch of 38.40 μs. The values are for a 3601C CPU, a 3601D may be up to 50% faster.

Unconditional instructions:

03	LOAD	66.15
04	SAVE	44.85
05	UNSAVE	44.85
06	GOTO	63.15
09	SETSTA	70.05
10	SETCNT	66.15
14	ZEROBC	43.65
17	SETSCN	55.05
18	SETLFS	70.20

Conditional instructions

07	IFEQ	96.55	71.80
08	IFNEQ	96.55	73.15
11	DECJNZ	67.65	46.40
19	IFMBIT	99.45	76.05
20	IFLT	95.55	72.15
21	IFGT	95.55	72.15

Others:

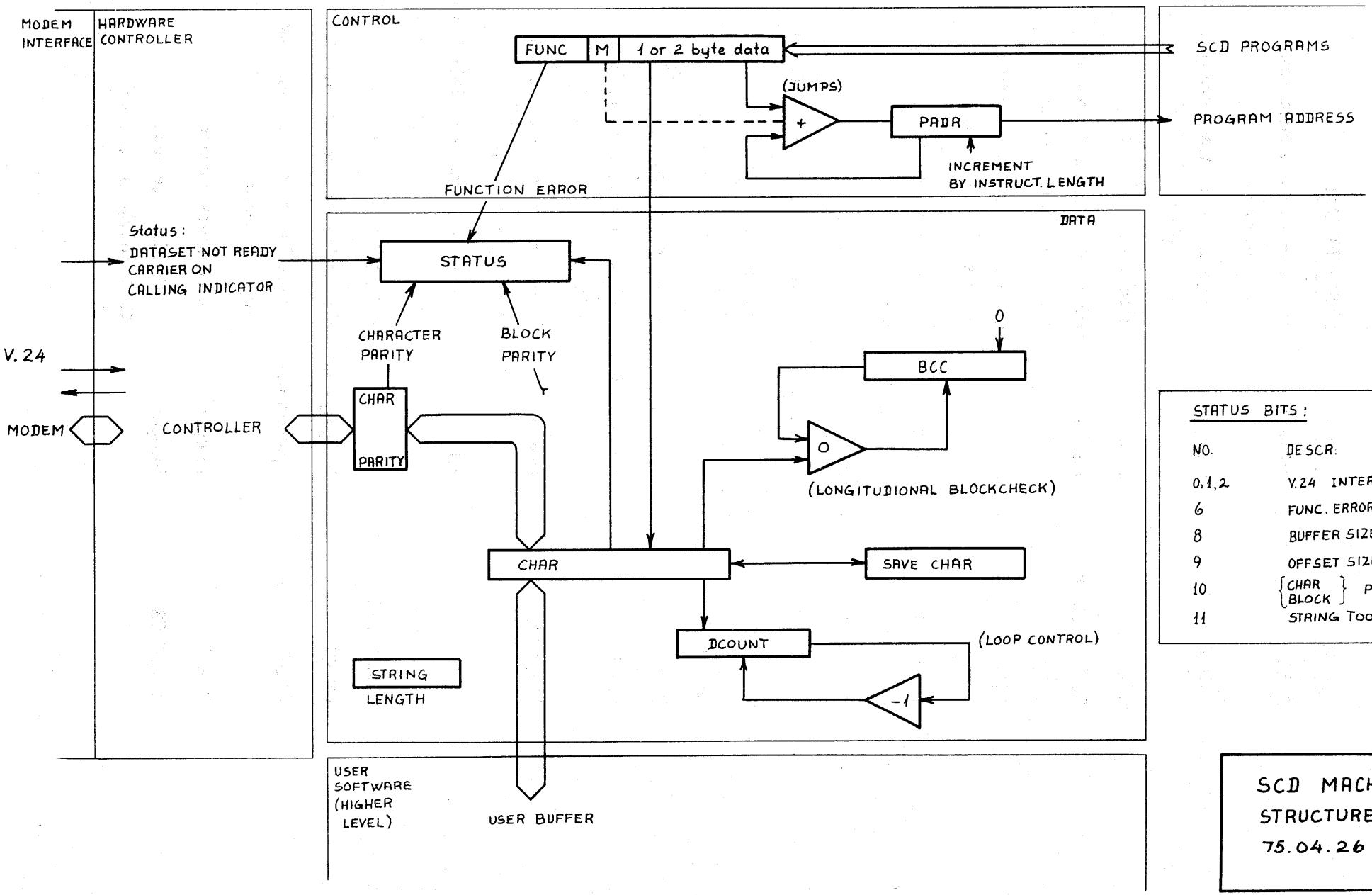
00	TRMBLK	in 124.45	plus driver/MUS time
		out 357.20	
01	GET	in, no parity	97.15
		in, parity	131.80
		out	79.80
02	PUT	in	93.85
		out, no parity	102.90
		out, parity	141.30
12	BEGSTR	in	75.90
		out	110.70
13	TRMSTR	in	133.70
		out	47.55
15	ACUMBC	no	43.65
		LRC	71.10
		SUM NEGSUM	64.50
		CRC12	137.10
		CRC16	131.70
16	COMPBC	no	43.65
		in, LRCE, SUM	69.00 *
		in, LRCO, NEGSUM	77.10 *
		in, CRC12	91.50 **
		in, CRC16	75.45 **
		out, LRCE, SUM	71.55 *
		out, LRCO, NEGSUM	79.65 *
		out, CRC12	86.25 **
		out, CRC16	70.20 **

* add a GET by input , a PUT by output

** add two GET by input, two output, and subtract 38.40.

3. Driver Logic

This section contains a description of the internal workings of the driver written in an informal higher language. The variables used refers mainly to the figure which shows a hypothetical machine structure.



STATUS BITS:

NO.	DESCR.
0,1,2	V.24 INTERFACE
6	FUNC. ERROR
8	BUFFER SIZE
9	OFFSET SIZE
10	{CHAR, BLOCK} PARITY
11	STRING TOO LONG

SCD MACHINE STRUCTURE
75.04.26

3.1. Variables

NAME	TYPE	DESCRIPTION
AREC	bitpattern	Image of receiver hardware status register.
AXMT	bitpattern	Image of transmitter hardware status register.
BCC	bytes	Blockcheck character(s).
BLOCKTIMER	integer	Number of 20 msec periods left in current interval
CHAR	byte	Character register. Content is always without parity.
CHECKMETHOD	integer	Identification of block check method.
CSL	integer	Character length.
CSLMASK	bitpattern	Mask used to remove parity bit.
CONVT		See below.
DCOUNT	integer	Counter register.
DMODE	bitpattern	Mode in last message.
LADR	byteaddress	Address of next byte in line buffer.
LCOUNT	integer	Bytecount, No. of bytes generated by PUT etc.
M	bit	0: Next address of GOTO is forward 1: - - - - - backwards
ODDCHAR	bit	Odd/even byte pointer.
OFFSET	integer,even	Offset from conversion table.
PADR	byteaddress	Instruction pointer in input or output program
PARITY	integer	Parity check method.
RESERVER	integer	0: Not reserved. <> 0: Driver is reserved.

SADR	byteaddress	Address of next byte in share.
SAVECHAR	byte	Saved CHAR register.
SCOUNT	integer	Bytecount, number of bytes remaining in share.
STATUS	bitpattern	Status to be returned to main program.
STRADR	byteaddress	0: No string <> 0: Address of first length field byte.
STRING	integer	Length of current string (if STRADR <> 0).
TIMEOUT	integer	Base index of timeouts to be used, origin is operation in last message.
TIMERS	byteaddress	Address of table with timeouts.

CONVT is defined as follows:

```
record
        TIMING          : integer ;
        CSL             : byte    ;   ! character set length !
        SYNC            : byte    ;
        PARITY          : byte    ;
        SUBST           : byte    ;
        BLOCKCHECK      : byte    ;
        DUMMY           : byte    ;   ! reserved for future use !
        OFFSET 0,
        OFFSET 1        : integer ;   ! even                    !
        TIMERS          : array 0..7  of byte;
        INPUTADR        : integer ;
        OUTPUTADR       : integer ;

end;
```

BUF has the following definition:

```
record
        OP          : integer ;
        COUNT       : integer ;
        ADDRE,
        MESS2       : integer ;
end ;
```

3.2 Use of Buffers

As transmission blocks vary in size from a few characters through approximately 80 characters to about 1200 characters, an internal buffer in the driver would be a little inconvenient. The problem is solved in the present driver by using the buffer given in the message for data communicated to or from the main program as well as data transmitted to or received from the communication line. As the input/output processing by the driver may generate characters, the source data is displaced OFFSETSIZE from the final destination.

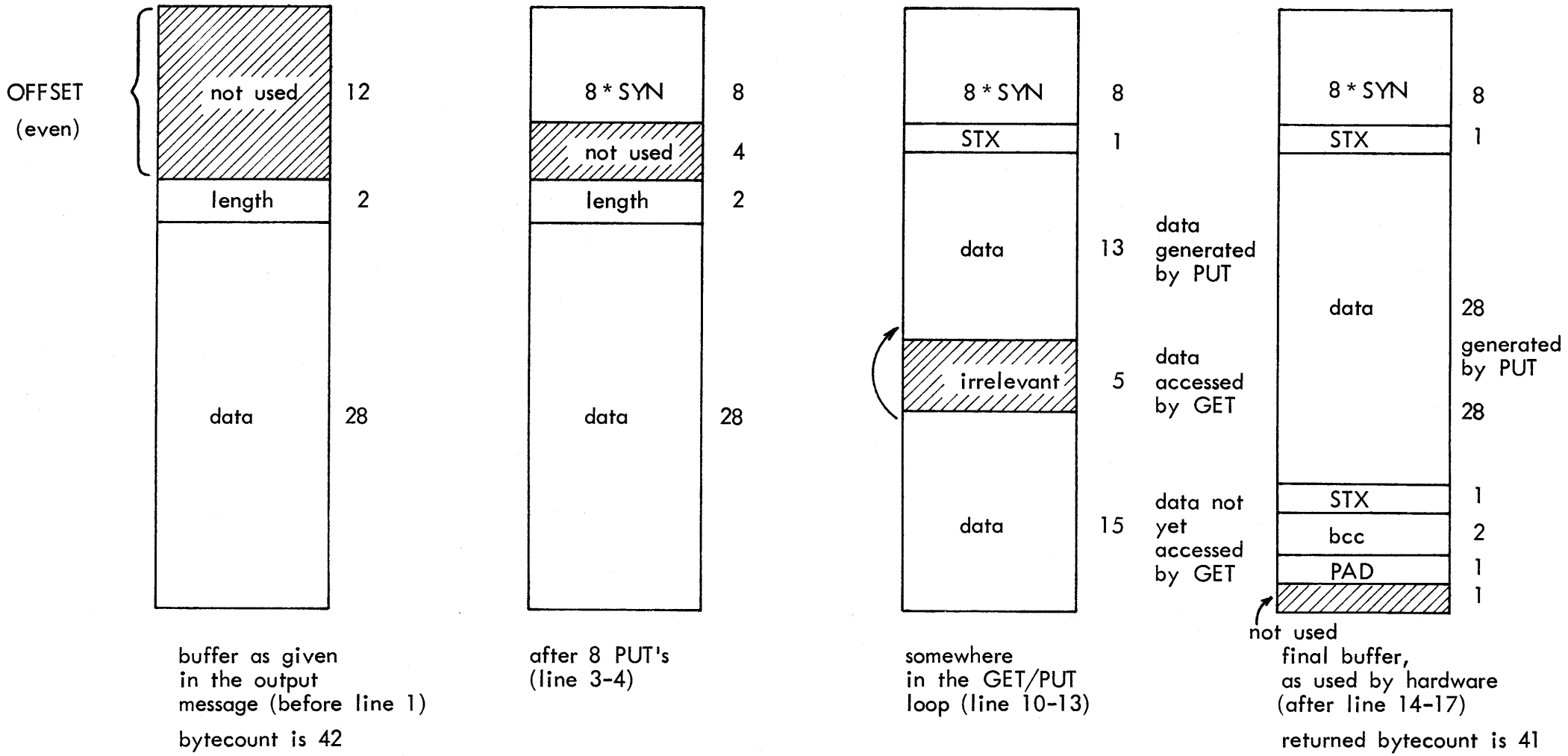
Following SCD program will need offsetsize 12, as CRC 16 is used (12 = 1 + 8 * SYN + STX + ETX + 2 * bbc + PAD - 2 * length) :

```
 1    .OUT     LOAD SYN        generate 8 SYN
 2             SETCNT 8
 3    .GSYN    PUT
 4             DECJNZ GSYN
 5             LOAD STX        generate STX
 6             PUT
 7             GET
 8             BEGSTR          2 bytes length 'n'
 9             ZEROBC
10    .PDATA   GET
11             ACUMBC          move 'n' characters
12             PUT
13             DECJNZ PDATA
14             LOAD ETX        generate ETX
15             PUT             and
16             COMPBC          blockchecksequence
17             TRMBLK          transport the block
```

internal format : length data

line format : SYN_8STX data ETX bcc PAD

Next page shows the buffer at different instances.



A buffer, at 4 different instances, when doing output.

An inputprogram checking blocks with data surrounded by STX ... ETX may look as follows :

```
 1    .IN        SETLFS 0                  two bytes length fields
 2               GET
 3               IFNEQ STX GOTO ERROR      first char STX

 4               ZEROBC                    prepare data
 5               BEGSTR

 6    .NEXT      GET
 7               IFEQ SYN GOTO NEXT        skip SYN inside data
 8               IFEQ ETX GOTO FINIS
 9               ACUMBC
10               PUT                       transfer to buffer
11               GOTO NEXT

12    .FINIS     TRMSTR                    end data
13               COMPBC                    perform block check
14               TRMBLK                    end processing

15    .ERROR     SETSTA 8                  set bit 8 :
16               TRMBLK                    Syntax error
```

The block may be structured on the line as this

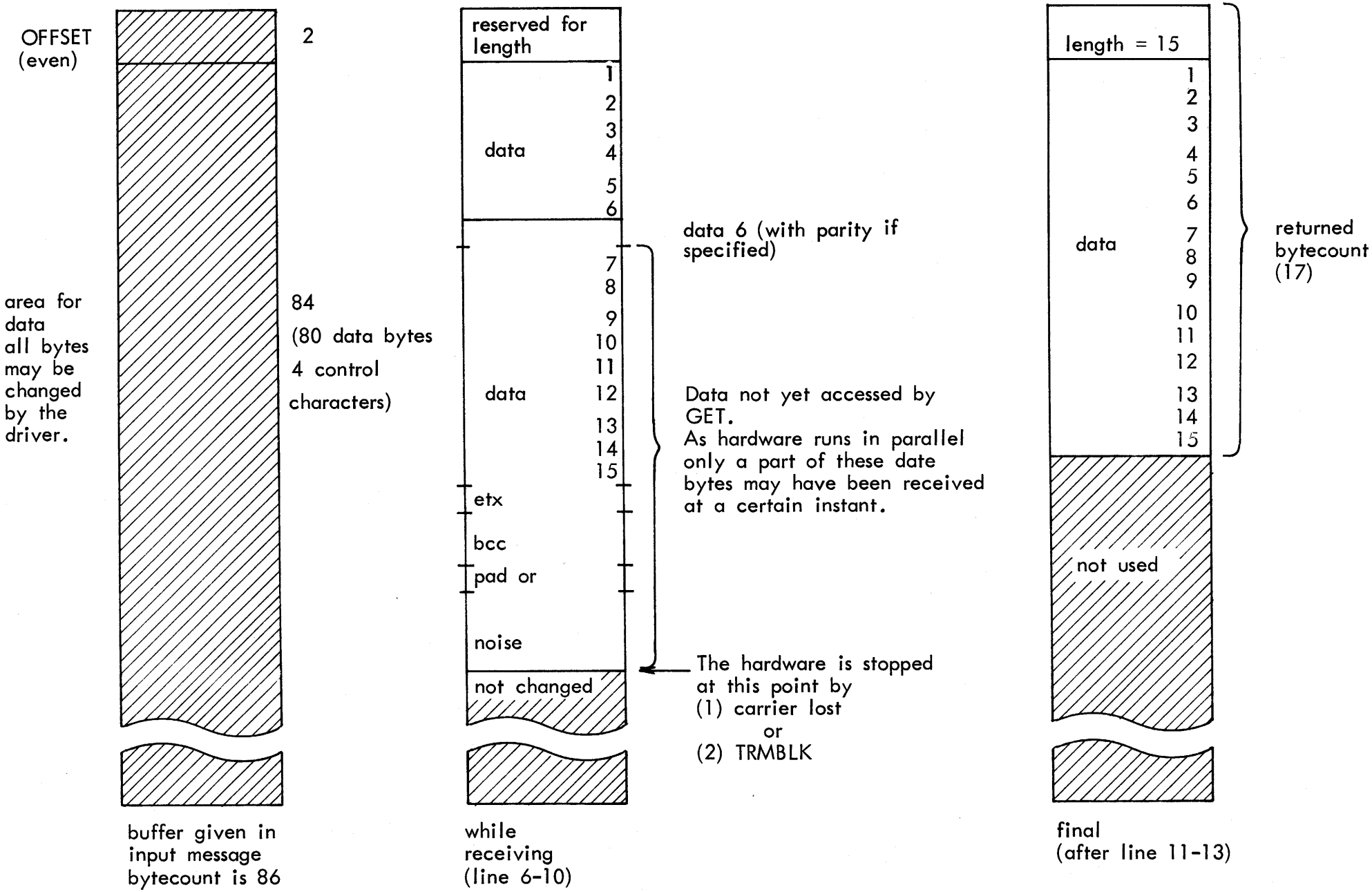
SYN SYN ... SYN STX (data, bytes <> ETX) ETX blockcheck PAD

The buffer is structured as this :

length data , length is two byte long.

The program will need offsetsize = 2 (in fact 1, but the value must be even).

Next page shows the use of the buffer when a block with 15 data bytes is input.


OFFSET
(even)
2
area for
data
all bytes
may be
changed
by the
driver.
84
(80 data bytes
4 control
characters)
buffer given in
input message
bytecount is 86
reserved for
length
1
2
3
data
4
5
6
7
8
9
10
11
data
12
13
14
15
etx
bcc
pad or
noise
not changed
data 6 (with parity if
specified)
Data not yet accessed by
GET.
As hardware runs in parallel
only a part of these date
bytes may have been received
at a certain instant.
The hardware is stopped
at this point by
(1) carrier lost
or
(2) TRMBLK
while
receiving
(line 6-10)
length = 15
1
2
3
4
5
6
data
7
8
9
10
11
12
13
14
15
not used
returned
bytecount
(17)
final
(after line 11-13)

3.3 Description of functions

```
'central  fetch'
CENTRAL:

        getbyte(function, PADR) ;
        M:=        function extract 1 ;
        function:= function shift   -1 ;
        if function outside interval then
        begin
                status:= status add 'illegalfunction' ;
                stop
        end ;

        goto case function of
        0:  TRMBLK ;
        1:  GET    ;
        2:  PUT    ;

          . . .

        17: SETSCN ;
        endcase ;

RETURN:

        PADR:= PADR + 1 ;
        goto CENTRAL ;

        'end of central fetch'
```

```
TRMBLK:

        perform TRMSTR;

        if OUTPUT then
        begin
           if data generated then
           begin
              if no error then
              begin
                 if not all characters used then
                 begin
                    status:= status add 'blocksyntax' ;
                    goto ReturnOutError ;
                 end ;

              'send block'
                putbyte( PAD , LADR);   LADR:= LADR + 1 ;
                set_address_and_start(xmt, address, count)

                case waitinterrupt(xmt, block length) ot
                timer :    status:= status add 'timeout' ;
                interrupt: if hardwarestatus <> 0 then status:=hardwarestatus;
                           goto ReturnOut ;
                endcase ;

           end;       'no error'

ReturnOutError:
           timing:= 128 ;
        end ;          ' data      '

ReturnOut:
        count:= LADR ;
   end ;            ' OUTPUT   '

ReturnIn:
   returnanswer(status, count);
   stop ;
```

```
GET:

  if INPUT then
  begin      ' fetch next character from line

     waitnext;
     getbyte(char, LADR);  LADR:= LADR + 1;
     case character parity of
         NO:   goto GoodCharacter ;
         ODD:  if parity (char) is odd   then goto ParityOk ;
         EVEN: if parity (char) is even  then goto ParityOk ;
     endcase ;

     'parityerror'
     status:= status add 'parityerror' ;
     char := SUBchar ;

ParityOk:
     char:= char extract characterlength

GoodCharacter:
     CHAR:= char ;

     goto RETURN;
  end ;

  'OUTPUT'

  if buffer empty then
  begin
     status:= status add'sharelength' ;
     stop ;
  end ;

  getbyte(CHAR, SADR);  SADR:= SADR + 1 ;
  SCOUNT:= SCOUNT - 1 ;

  goto RETURN ;
```

```
PUT:

  if INPUT then
  begin    'put CHAR into buffer'
     if no room then
     begin
        status:= status add ´offsetsize´ ;
        stop ;
     end ;
     putbyte(CHAR, SADR) ;   SADR:= SADR + 1 ;
     STRING:= STRING + 1 ;
     goto RETURN ;
  end ;
  'OUTPUT - output character on line'
  if no room then
  begin
     status:= status add ´offsetsize´ ;
     stop ;
  end ;
  putbyte(GenerateParity(CHAR, characterparity, LADR) ;
  LADR:= LADR + 1 ;
  goto RETURN ;

SETSCN:

  DCOUNT:= bytecount - selected OFFSET;
  goto RETURN ;
```

```
LOAD:

  PADR:= PADR + 1 ;
  getbyte(CHAR, PADR) ;
  goto RETURN ;

SAVE:

  SAVECHAR:= CHAR ;
  goto RETURN ;

UNSAVE:

  CHAR:= SAVECHAR ;
  goto RETURN ;

GOTO:

  PADR:= PADR + 1 ;            ' displacement in next byte '
  getbyte(disp, PADR) ;
  if M=1  then  disp:= -disp ;
  PADR:= PADR + disp ;
  goto CENTRAL ;               ' escape PADR increase '

IFEQ:
IFNEQ:

  PADR:= PADR + 1 ;
  getbyte(char, PADR) ;
  if function = 'ifeq' and char = CHAR
                 or
    function = 'ifneq' and char <> CHAR    then goto GOTO ;
  PADR:= PADR + 1 ;            ' skip displacement '
  goto RETURN ;
```

```
SETLFS:        ! set length field size        !

        PADR:= PADR + 1 ;
        getbyte (c,     PADR) ;
        DMODE (0:0 ) :=  c  extract 1 ;
        goto RETURN ;

IFMBIT:

        PADR:= PADR + 1 ;
        getbyte (bitno, PADR) ;
        if DMODE (bitno) <> 0 then goto GOTO ;
        PADR:= PADR + 1;  ! skip displacement !
        goto RETURN '

IFLT:
IGFT  :

        PADR:= PADR + 1 ;
        getbyte (char,     PADR) ;
        if function = 'IFLT' and char > CHAR
                            or
          function = 'IFGT' and char < CHAR then goto GOTO ;
        PADR:= PADR + 1 ; ! skip displacement !
        goto RETURN ;
```

```
SETSTA:

  PADR:= PADR + 1 ;
  getbyte(bitno, PADR) ;
  status:= status add ( 1 shift (15-bitno) ) ;
  goto RETURN ;

SETCNT:

  PADR:= PADR + 1 ;
  getbyte(DCOUNT, PADR) ;
  goto RETURN ;

DECJNZ:

  DCOUNT:= DCOUNT - 1 ;
  if DCOUNT <> 0  then goto GOTO ;
  PADR:= PADR + 1 ;          ' skip displacement '
  goto RETURN ;

BEGSTR:

  if INPUT then
  begin     ' make room for length field '
     if STRADR <> 0  then perform TRMSTR ; ' unfinished string '
     if no room  then
     begin
        status:= status add 'offsetsize' ;
        stop ;
     end;

     STRADR:= SADR ;    STRING:= 0 ;
     SADR:= SADR + 2 - DMODE (0:0) ;
     goto RETURN ;
  end ;
```

```
'OUTPUT - fetch length

  if DMODE (0:0) = 0 then
  begin  ! two byte - one in CHAR !
     DCOUNT:= CHAR * 256 ;
     perform GET ;
     DCOUNT:= DCOUNT + CHAR
  end
  else DCOUNT:= CHAR

  goto RETURN ;



TRMSTR:

  if INPUT and STRADR <>0 then
  begin
     if DMODE (0:0) <>0  then
     begin
       if STRING ≥ 256 then
       begin ! too big for one byte !
         status:= status add 'string overflow' ;
         stop
       end ;
       putbyte (STRING, STRADR) ;
  end
  else
  begin
       putbyte (STRING//256, STRADR) ;
       putbyte (STRING mod 256, STRADR + 1) ;
  end;
       STRADR:= 0 ;
  end ;

  goto RETURN ;

ZEROBC:

  BCC:= 0 ;
  goto RETURN ;
```

```
ACUMBC:

  case checkmethod of
    NO:                ;
    LRCE:  LRCO:
       BCC:= BCC  xor  CHAR ;
    SUM:   NEGSUM:
       BCC:= BCC + CHAR ;
    CRC12:
       char:= CHAR extract 6 ;
       byte:= BCC   extract 6 ;
       crc := BCC   shift  -6 ;
       c   := byte xor char   ;
       crc := crc xor (c shift 3) xor (c shift 5) ;
       BCC:= crc xor (8'5001 * parity(c)  ) ;
    CRC16:
       byte:= BCC extract 8 ;
       crc := BCC shift  -8 ;
       c   := byte xor CHAR ;
       crc := crc xor (c shift 7) xor (c shift 6) ;
       BCC:= crc xor (8'140001 * parity(c)  )  ;
  endcase ;
  goto RETURN ;
```

```
COMPBC:

  if checkmethod = NO then goto RETURN ;

  if INPUT then
  begin
    perform GET ;       c:= CHAR ;
    if checkmethod is CRC 12/CRC16  then
    begin  ' one more char  '
      perform GET ;
      c:= CHAR shift characterlength + c ;
    end

    case checkmethod of
    LRCE:  SUM:  CRC12:  CRC16:
         if c = BCC then goto InOk ;
    LRCO:
         if -,c = BCC then goto InOk ;
    NEGSUM:
         if -c  = BCC then goto InOk ;
    endcase ;

    status:= status add 'parityerror ' ;
InOk: BCC:= 0 ;
    goto RETURN ;

  end ;
'OUTPUT - generate checksequence '

  case checkmethod of
  LRCO:
       c:= -,BCC ;
  NEGSUM:
       c:= -BCC ;
  LRCE:  SUM:  CRC12:  CRC16:
       c:= BCC ;
  endcase ;

  CHAR:= c extract 8 ;      perform PUT ;
  if checkmethods = CRC12/CRC16 then
  begin
    CHAR:= c shift(- characterlength);
    perform PUT ;
  end ;
  goto InOk ;
```

4. Use of CCITT V24 Circuits

The following circuits are used by SCC 702 controller :

TO FROM MODEM	CIRCUIT		DESCRIPTION
-	101	*	Protective ground or earth
-	102	*	Signal ground or common return
TO	103	*	Transmitted data
FROM	104	*	Received data
TO	105	*	Request to send
FROM	106	*	Ready for sending
FROM	107		Dataset ready
TO	108/2		Data terminal ready
FROM	109		Datachannel received line signal detector
TO	111		Data signaling rate selector
TO	113	*	Transmitter signal element timing
FROM	115	*	Receiver signal element timing
FROM	125	*	Calling indicator

Circuits with * are set up automatically by the controller.
The use of circuit 119 (transmitting timing) depends on whether the SCC 702 is equipped with clock option.

Circuits 107, 109, and 125 are given as status bits if relevant, 107 and 109 when they go from ON to OFF.

Circuit 108/2 is set to ON when a control message with conversion has arrived, but only if circuit 107 is OFF. It is set to OFF when a disconnect control message arrives.

Circuit 111 (rate selector) is allways set to HIGH by the driver. It is possible, however, to make a strap in the SCC 702 (position 7) to select between HIGH, LOW, and software selected. This strap is normally HIGH.

APPENDIX A

The programs for in- and output are interpreted by SCD. The internal format is a sequence of bytes, where each instruction are of length 1, 2, or 3 bytes :

function	length	format +0	+1	+2
TRMBLK	1	0		
GET	1	2		
PUT	1	4		
LOAD	2	6	value	
SAVE	1	8		
UNSAVE	1	10		
GOTO	2	12+M	disp	
IFEQ	3	14+M	value	disp
IFNEQ	3	16+M	value	disp
SETSTA	2	18	bitno	
SETCNT	2	20	value	
DECJNZ	2	22+M	disp	
BEGSTR	1	24		
TRMSTR	1	26		
ZEROBC	1	28		
ACUMBC	1	30		
COMPBC	1	32		
SETSCN	1	34		
SETLFS	2	36	value	
IFMBIT	3	38+M	bitno	disp
IFLT	3	40+M	value	disp
IFGT	3	42+M	value	disp

value	in the interval 0..255
bitno	in the interval 0..15
disp	in the interval 0..255
M	0: resulting address is = addr(disp byte) + disp
	1: resulting address is = addr(disp byte) - disp
	i.e. relative to the address of the byte contains displacement

FORMAT OF SCD PROGRAM HEAD :

0	+ 0	TIMER	(2)
1	1		
2	+ 2	CLENG	(1)
3	+ 3	SYNC	(1)
4	+ 4	PARITY	(1)
5	+ 5	SUBST	(1)
6	+ 6	CHECK	(1)
7	+ 7	(RESERVED)	
10	+ 8	OFFSET 0	(2)
11	9	[EVEN]	
12	+ 10	OFFSET 1	(2)
13	11	[even]	
14	+ 12		
15	13	TIMEOUTS	(8)
16	14		
17	15		
20	16		
21	17		
22	+ 18		
23	19		
24	+ 20	INADR	(2) ABSOLUTE ADDRESS OF FIRST BYTE
25	21		
26	22	OUTADR	(2) ABSOLUTE ADDRESS OF FIRST BYTE
27	23		

APPENDIX B

DRIVER		PROCESS NAME	DEVICE CODE OR CHANNEL	RCSL.BIN.TAPE
SINGLE CHANNEL:				
SCD03	(STANDARD)	'SCD32'	40/41	43-GL 1254
SCD04	(OVERLAPPING OUTPUT)	'SCD32'	40/41	43-GL 1797
S7000	(FOR DGC 4015)	'SCD32'	-	43-GL 1689
SCD14	(as SCD04)	'SCD34'	42/43 (sec.channel)	43-GL 1923
MULTIPLEXER:				
MAIN MODULE (ALWAYS PRESENT)				
SMD02		-	0 - 7	43-GL 1419
FOR THE CHANNELS:				
MULTIPLEXER USED AS SINGLE CHANNEL				
SX001		'SCD32'	0	43-GL 1122
SX101		'SCD32'	1	43-GL 1932
SX201		'SCD32'	2	43-GL 1935
SX301		'SCD32'	3	43-GL 1938
MULTIPLEXER USED AS MULTI CHANNEL				
SMX0		'SMX0'	0	43-GL 915
SMX1		'SMX1'	1	43-GL 918
SMX2		'SMX2'	2	43-GL 921
SMX3		'SMX3'	3	43-GL 924
SMX4		'SMX4'	4	43-GL 1125
SMX5		'SMX5'	5	43-GL 1128
SMX6		'SMX6	6	43-GL 1131
SMX7		'SMX7'	7	43-GL 1134

FRONT-END PROCESSOR DRIVERS

General Description

The drivers are intended for communication with RC 3500, RC 3600, and RC 8000, using the FPA 702 Front-End Processor Adaptor. So the dataformats are the ones recognized by FPA 702, datablock and status-byte.

A datablock has the format:

<start> <data>

where <start> consists of one 8-bits character

<data> consists of a number of 8-bits characters.

A statusbyte is a single 8-bits character.

The drivers are two fully independent processes,

FPAX for the transmitter controller

FPAR for the receiver controller.

Only single buffer processing is implemented.

Control

Reservation, conversion, position, and termination have effect.

Disconnect and erase are ignored.

Conversiontable has the only purpose of passing parameters to the drivers.

It has the following format:

+ 0-1 timer

+ 2-3 first word in testbuffer (word address)

+ 4-5 first word after testbuffer (see testoutputsection)

+ 6-7 current word in testbuffer

Timer is used by the drivers as the maximum time to wait for an operation to terminate. Unit 0.02 sec.

Termination is ignored by receiver. On the transmitter a termination without conversion causes a "reset" command to be executed.

Position is used to give a new timer to the drivers. MESS2 is the new timer. On the transmitter a control operation with 1b7 sat causes an autoload signal to be transmitted.

mess 3 contains a parameter with the following meaning:

mess 3 = ϕ: transmit autoload signal and receive statusbyte.

mess 3 <> ϕ: transmit autoload signal and return answer immediately.

The autoload signal is intended for use when the connected computer should be autoloaded via FPA 702. First an autoload pulse is generated and the driver loops in mess1 * 2 usec. Then the controller is started in receive status mode. This is consistent with the fact that the IORST instruction sets the receiver into transmit-receive mode, so that a NIOS REC will cause the receiver to transmit a zero statusbyte before it waits for a block.

Generally, a control message causes the controller to be cleared. It does not affect the status register of any of the controllers, and the status of the answer will always be zero.

Transput

a. Receiver

After break all operations are returned.

First operation after powerbreak is returned immediately.

Three modes are recognized:

1 transmit statusbyte - receive block

5 transmit statusbyte

9 receive block

Messageformat:

messφ	mode	
mess1	bytecount	the maximum No. of bytes in the field <data> of the received block. Must be even. Irrelevant when mode = 5.
mess2	address	must be even. The field <data> of received block is stored at this address and onwards. Irrelevant when mode = 5.
mess3	statusbyte	In mode 5 and 1 the rightmost byte of this word is transmitted as a statusbyte.

Answerformat:

messφ	status	
mess1	bytecount	the number of characters in the field <data> of received block. Irrelevant when mode = 5.
mess2	-	
mess3	startchar.	The character of the field <start> in received block.

Status:

1b6	illegal command
1b7	power break
1b8	block length error
1b10	parity error in a received character
1b11	reset signal received
1b14	timeout

b. Transmitter

Three modes are recognized:

3	transmit block - receive statusbyte
7	receive statusbyte
11	transmit block

Messageformat:

messφ	mode	
mess 1	bytecount	the number of characters in the field <data> of the block to be transmitted. Only relevant in modes 3 and 11.
mess2	address	address of the field <data> to be transmitted. Must be even. Only relevant in modes 3 and 11.
mess3	startchar	The <start> of the block to be transmitted. Only relevant in modes 3 and 11.

Answerformat:

messϕ	status
mess1	bytecount
mess2	-
mess3	received statusbyte. Only relevant in modes 3 and 7.

Status:

1bϕ	disconnect
1b1	receiver not ready
1b6	illegal
1b14	timeout

Testoutput

When testoutput is produced, all operations and up to 40 characters of the datablock will be stored cyclically in a core buffer in the user program. Convtab (2 : 3) is first word address of the buffer, convtab (4 : 5) is top word address of the buffer, and convtab (6 : 7) is used as current pointer in the buffer (this word is updated by the drivers).

If first < top and switch 15 on the NOVA is on (up), then testoutput is produced.

The format of the testrecord is:

+ 0:	identification of driver. 11111 for receiver, 22222 for transmitter.
+ 1:	operation. MESS 0 of message.
+ 2:	MESS3 (MESS2 if control operation) of message. - 1 if unused.
+ 3:	status. MESS 0 of answer.
+ 4:	bytes transferred. - 1 if unused.
+ 5:	special. MESS 3 of answer. - 1 if unused.
+ 6 ff:	up to 40 bytes of datablock, one byte per word.

FIXED HEAD DISC DRIVER

Control, input and output messages are accepted. The disc is a random accers device but the driver may be called as it was a sequential device too.

The size of one block is 512 bytes and an arbitrary number of blocks (>0) may be transferred in one message.

Each disc unit contains 1024 blocks, and the maximum number of blocks which may be accessed are 1024 x 3 = 3072 blocks, i.e. at most 3 units may be handled by the driver.

When the blockcount (= first segment) and the bytecount are given the position is calculated as :

Unit : = (blockcount - 1) // (1024);

Block : = (blockcount - 1) mod (1024);

Track : = block//8;

Sector : = block mod 8;

No. of blocks to

Transfer : = bytecount//512;

The driver can only treat one message at the time.

CONTROL

Reservation and position are accepted.

Position : The logical position is set in accordance to the content of:

Mess3.buf : blockcount

where blockcount > 0.

TRANSPUT

General:

	Message	Answer
mess0.buf	operation	status
mess1.buf	bytecount	bytecount
mess2.buf	first byte address	unchanged
mess3.buf	blockcount	blockcount

Where bytecount mod 512 <> 0 will give status (blockerror) and

bytecount: =bytecount//512 * 512,

and the operation is continued.

The mess3.buf of the message is only significant in random transput operations.

The blockcount is increased by one if the bytecount <> 0 after a successful transput operation.

MODE

Input: 1 : read, sequential

5 : read, random

Output: 3 : write, sequential

7 : write, random.

TRANSPUT OPERATIONS

Sequential : The blocks are transferred successively through the disc. The position may be redefined by the control command 'position' or by a random transput operation.

Random : The content of mess3.buf is used to set the position of the disc prior to execution of the transput operation.

STATUS:

b0: disconnected, the selected unit does not exist.

b5: write-protected, the selected block is writeprotected.

b6: illegal, device reserved;
write protected in output operation.

b8: block error, the specified bytecount is not an integral multiple of 512.

b9: data late.

b10: parity error.

b12: position error, the attempted position is outside the possible number of units.

b14: timer, a timeconsuming operation has lasted more than 5.12 sec.

Driver for RC 866 printer with ACK/NAK.

General Description.

Control- and output-messages are accepted. Input messages are returned with status illegal. This driver will not start or initialize AMX drivers. The multiplexer line No. is given as left byte of word 34 (octal) in the process description. Word 33 is used as byte address of standard table.

Control.

Reservation, conversion, position and termination are accepted.

Conversion table address should be a byteaddress. If RC 3683 TMX is used then the driver must send characters with correct parity, this may be done by means of a conversion table.

Position. If mess. 3, block, is negative then the message is sent to the AMX driver, else mess. 2, file, is used as margin for output messages with mode = 7 or 23.

Termination is sent to the AMX driver as a disconnection.

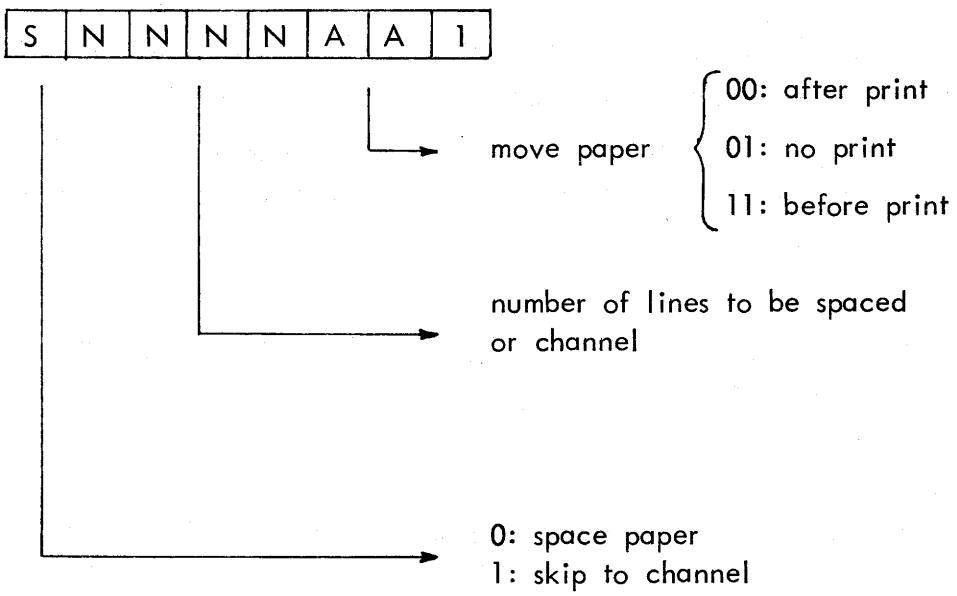
Output.

Two modes of operation exist:

3: The converted characters are sent to AMX.

7: The first byte of the message is interpreted as a carriage control word.

The carriage control word:



The RC 866 does not have a normal carriage control tape (see RCSL: 52-AA360), so the result of skip to channel is:

Channel	Character sent
0	LF
1	LF
2	VT
3	FF
4	DC1
5	DC2
6	DC3
7	CR
all others	LF

Mode + 16 is used if full duplex operation is wanted.

Status.

Status is the status from the AMX driver in use.

CCW error gives 1b10, parity.

Answer NAK from the printer gives 1b9, data late.

READER PUNCH DRIVER RC3663, RC3664

The reader punch is a combined input/output device which can preforme reading, puncing, and printing of cards. As cards can be fed from two hoppers and stacking can be done in two stackers, each operation send to the driver contains controll information used to route cards in the way the programmer wants.

A third stacker, the reject stacker, is used to stack cards with punch errors, and can not be controlled by the user.

Control

Whenever the driver receives a control message, a sense of the device is executed.

Driver can be reserved and is able to convert data by means of a conversion table transferred by the user in a control message with conversion bit set.

Only one conversion table is known by the driver, and this has to be exchanged by the user if mixed read and punch/print is preformed.

If the disconnect bit in mode is set, the cards in track, if any, is led out in the stacker in question.

Input

Data to the user can be transferred in three ways :

As binary words where each user word corresponds to the card columns as shown

Card column hole :	X	X	X	12	11	0	1	2	3	4	5	6	7	8	9
Bit no.:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Byte :				n							n+1				

The maximum number of bytes transferred in this way is 160 (102 if 51 0 cards).

As binary bytes where each user byte corresponds to the card columns as shown :

Card column hole :	2	3	4	5	6	7	8	9
Bit no.:	0	1	2	3	4	5	6	7

Maximum numbers of bytes transferred in this way is 80 (51).

As EBCDIC characters as a result of conversion by the given conversion table.

Maximum numbers of bytes transferred in this way is 80 (51).

Input modification and control

Special modebits set :

bit 6 = 1 — Read only. Cards are stacked immediately af the read is preformed. Use of this modification increases card speed to 400 CPM, but the cards cannot be punched or printed by following operations.

bit 7 = 1 — The card to be read is fed from hopper # 2.

bit 8 = 1 — Inhibits the input feed of the card i.e. the card in track is stacked and no bytes are transferred.

bit 9 = 1 — The card in wait station will be stacked in stacker # 2.

Bits (10:15)	The mode of operation.
	Five modes exist :
1 :	Read binary bytes, no conversion is preformed.
5 :	Read binary words, no conversion.
9 :	Read decimal punched cards, skip trailing blanks, and terminate with byte values <13> <10>. A maximum of 72 bytes is delivered. Data is converted if conversion table is specified.
21 :	Read decimal punched card and convert.
33 :	Read decimal punched cards, convert, and skip trailing blanks. If a blank card is read, one blank character is delivered.

All other modes of operation are returned with illegal status.

Except for the above mentioned case (bit 9 = 1) stacker # 1 is used.

Bytecount must be equal to or greater than the number of bytes to be returned, or else the operation is returned with block length error and a number of bytes transferred equal to bytecount. The rest of the columns will be skipped. The byte values of the last two bytes delivered in mode 9 will always be <13> <10>.

In mode 5 byte address and byte count must be even or else the operation is returned with illegal status.

If a card read in mode 9, 21 or 33 contains an illegal hollerith punch code, parity error status is returned with the data. A zero byte is substitutet for all illegal columns and converted.

Output

Data to be punched in the card can be transferred in the same three formats as the read data described in Input section.

If multi line print is installed, print data is transferred in a special format. The format makes it possible to specify the line number to be printed and the number of characters in the current line as shown :

PRINT DATA START <line no> <no of char in line> <first char> ....<last char>
<line no> <no of char in line> <first char> ....<last char>

Line numbers must follow in increasing order and number of characters must be less than 65, if not only 64 characters are transferred and block length error is indicated.

Output modification and control

bit 5 = 1 — Punch after read.

Is used if the cards to be punched are not equal. If not used in this case auto punch retry will function incorrectly.

Hopper # 2 should contain blank cards for auto punch retry routine.

bit 6 = 1 — Print after punch.

The first 80 (51) bytes are interpreted as the punch data, and the rest as the print data. (If binary words, the first 160 (102) bytes are the punch data).

If multi line print option is installed, print data is interpreted as shown in the general information.

If single line print is installed, the remaining bytes are taken as the print line, and the line can be maximally 80 bytes long.

Only relevant in mode 3,7, and 11.

bit 7 = 1 — Selects stacker # 2.

bit 8 = 1 — Inhibits the input feed of the next card from hopper.

bit 9 = 1 — The card in the wait station is led out to the appropriate stacker, and a card is fed before the punching/printing is performed.

bit 10 = 1 — Selects the primary hopper.

bits (11:15) = — The mode of operation.
Six modes of operation exist.

3 : Punch binary bytes, no conversion.

7 : Punch binary words, no conversion.

11 : Punch decimal bytes with conversion.

19 : Print decimal bytes with conversion.

If multi line print option is installed the print data is interpreted as print data used with separate print.

If single line print data is transferred directly from the buffer. Max. 80 bytes are printed.

27 : Punch and print buffer.

If multi line print is installed, the first 64 punched card columns are printed on the top row and the remaining 16 on the second line right justified.

If single print is installed, the punch data is printed on the top of the card.

All other modes of operation are returned with illegal status.

Except for the above mentioned case (bit 10 = 1), hopper # 2 is used.

In the decimal modes and the punch binary bite mode, bytecount must be less than or equal to 80 (51). If not, the operation is returned with block length error, and 80 (51) bytes will be transferred.

In the punch binary word, mode bytecount and byteaddress must be even numbers, or else the operation is returned with illegal status. Bytecount must be less than or equal to 160 (102), or else the operation will be returned with block length error and 80 (51) words transferred.

When using seperate print or print only, the data format depends on print option installed.

If multi line print option print data is transferred with two bytes information code in front of each print line. First byte is the line number ($1 \leq$ line no ≤ 12) and the second number of bytes in the line ($0 \leq$ count ≤ 64), this can give a maximal buffer size of 2 * 12 + 64 * 12 = 792 bytes + the punch data.

If more than 64 bytes pr. line are specified, block length error is set and 64 bytes are printed.

If single line print the first 80 (51) is the punch data, and the rest the print data, and total bytecount must be less than or equal to 240 (153). If not, operation is returned with block length error, and 80 (51) punch data and 80 (51) print data will be transferred.

In case of binary word transfer bytecount must be less than or equal to 240 (153), if not, 160 (102) bytes are punched and 80 (51) bytes are printed. Operation is returned with block length error.

In all mode unpunched/unprinted columns are filled with blanks.

Data conversion

If the operation is 1, 9, 11, 19, 21, 27 or 33 the bytes are converted before input/output by simple indexing.

If the operation is 3 or 7, only print data, if any, is converted in this way.

Status

In case of error, the reader punch device displays a two digit error condition number on the operator panel. Operator action to be taken is given by this number and can be found in table 1.

The error code is transferred as a binary number in the answer in Mess2 of the message, and can be found in ZONE.ZFILE in case of file descriptors is used to communicate with the reader punch driver.

The code is transformed to normal MUS status information with following interpretation :

1b0 : Auto Punch retry failed, cards not cleared from track as indicated by error code, (with 1b1) top cover raised during operation, Miss stack during auto punch retry (with 1b5).

1b1 : Device offline, stacker full (with 1b3), chad box full, cards not cleared from track as indicated by error code (with 1b0), stack fault (with 1b5), read check (with 1b10), Motion check (with 1b2).

1b2 : Pick check, motion check (with 1b1).

1b3	:	Stacker # 1, # 2 or Reject stacker full (with 1b1).
1b4	:	51 column cards.
1b5	:	Miss stack, Output error, track jam in stacker area.
1b6	:	Device reserved, unknown command.
1b7	:	Secondary hopper empty.
1b8	:	Block length error.
1b10	:	Parity error, Read check, double feed (with 1b1).
1b11	:	Primary hopper empty.
1b12	:	EOF switch activated. Returned only on input message and with either 1b7 or 1b11 indicating the empty hopper.
1b13	:	Driver not loaded.
1b14	:	Time out, device not ready within specified time.

As the device has post-read feature installed, all data punched is compared with the data read by the card feed and the data transferred to the punch.

If any difference occurs the card is stacked in the reject stacker and a new card is fed from same hopper and punched with the dame data.
If data is correct punched, this card is stacked too in the reject stacker and a final card is punched and placed in the original specified stacker.

If, however, any error occurred in card two or three, the reader punch is stopped and the error is indicated on the operator panel and the message is returned with status <> 0.

If modification Punch-after-read is used in the output message blank cards for punch retry is fed from hopper # 2.

TABLE 1

Error codes returned in Mess2 of message :

Error codes X1 to X7 are duplicated for every group 1X to 8X, operator action is taken as combination of both, and the status is the or'ed value.

ERROR CODE	ERROR DESCRIPTION	OPERATOR ACTION	STATUS
00	OFFLINE	Press START	1b1
01	Hopper # 1 empty	Correct and press START	1b11
02	Hopper # 1 pick check	Correct and press START	1b2
03	Stacker # 1 full	Correct and press START	1b1 +1b3
04	Hopper # 2 empty	Correct and press START	1b7
05	Hopper # 2 pick check	Correct and press START	1b2
06	Stacker # 2 full	Correct and press START	1b1 +1b3
07	Reject stacker full	Correct and press START	1b1 +1b3
08	Chadbox full	Correct and press START	1b1
09	Cards not cleared from track	Correct and press START	1b0+1b1
X1	Card miss-stack # 1	Correctly stack all cards in track except last in track	1b1 +1b5
X2	Card miss-stack # 2		
X4	Card failed to reack stacker		
X5	Card miss-stacked # 1 Jam in output		
X6	Card miss-stacked # 2 Jam in output		
X7	Card miss stacked to reject stacker. Jam.		
X8	Track jam during Punch Retry	Place last cards in punch area into reject stacker.	

ERROR	ERROR DESCRIPTION	OPERATOR ACTION	STATUS
1X	Miss stack or jam	Stack all cards in track	1b1 +1b5
3X	Motion check	Stack all cards except last. Return this to hopper # 1.	1b1 +1b5
4X	Read check. Double feed from hopper # 1	Stack all cards except last (last two). Return rest to hopper 1.	1b1 +1b10
5X	Read check. Double feed from hopper # 2.	Stack all cards except last (last two). Return rest to hopper # 2	1b1 +1b10
6X	Motion check	Stack all cards except last. Replace last with a blank card. Place blank card in hopper # 1.	1b1 +1b2
8X	Card pick error	Return last card to hopper # 1.	1b1 +1b2
94	Miss stack during Auto punch retry	Press CANCEL	1b0 +1b5
95	Registration failure	Press CANCEL	1b0
96	Error require multiple recovery procedure	Press CANCEL	1b0
97	Hood raised	Press CANCEL	1b0
98	Auto punch retry failed	Press CANCEL	1b0

1b10 single indicates in input operation that the card read had an illegal hollerith code. The programmer must decide the action to be taken. Each column with an illegal hollerith code is returned as a converted 0 byte.

Any other messages rejected by an error should be repeated after properly operator action.

In case of error in group 9x CANCEL must be activated and job restart is necessary.

If CANCEL is used in other error conditions data can be lost, therefore, the proper operator action is correction of the error and activation of the START button.

TABLE 2.

Print drum character set.

	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
0					SPACE	&	-								\	0
1							/		A	J			A	J		1
2									B	K	S		B	K	S	2
3									C	L	T		C	L	T	3
4									D	M	U		D	M	U	4
5									E	N	V		E	N	V	5
6									F	O	W		F	O	W	6
7									G	P	X		G	P	X	7
8									H	Q	Y		H	Q	Y	8
9									I	R	Z		I	R	Z	9
10					¢	!		:								
11					.	$	,	#								
12					<	*	%	@								
13					(	)	_	´								
14					+	;	>	=								
15					|	>	?	"								

Any code not specified is printed as underscore (_).

DISC STORAGE MODULE DRIVER
RC8221, RC8222, RC8223, RC8224.

GENERAL

Control, input and output messages are accepted. The driver may handle up to 4 disc drives connected to the same controller.

Moreover the driver can handle 12,18,33,66,125 and 250 M byte drives in any combination.

Addressing of drives and sectors uses a logical segment number. To address no. x on drive no. i the message should contain the following block number:

blockcount = x + (i *(2**19));

i.e. add the unit number in the mode (3:4) of the message.

Two different kind of drivers exist:

1. normal driver, used for normal use and reliability tests.
2. initiation driver, used for initiation of new disc cartridges and for test programs. The driver has most of the modes of the normal drive, but it may be hazardous to use during test of new program because all the information on the cartridge may be spoiled if a wrong mode is used.

CONTROL

Reservation and position are accepted.

Position uses block number = mess 0 buf (0:7)*2^{16} + mess 3. buf. as current segment number and first byte = mess 2. buf (byteoperation only). The drive is positioned just before a following transputoperation.

mess 0	block no.(high)*256 + position
mess 1	0
mess 2	first byte
mess 3	block no.(low)

If blocknumber is outside the existing disc drives then status: = end medium.

TRANSPUT

General

Input and output operations are possible.

If a position or a parity error is sensed after an input command with data transfer the operation is repeated with off-set, which means that the position of the heads is moved a little off the track. This will go on until either the error disappears or all the off-set possibilities have been tried (ref. 1, chapter 3.7.3.). Regardless of the kind of disc the offset magnitude is set up even it is only meaningful in case of the 125 M and the 250 M disc drives. The number of repeats will then be: 9 x 16 x 3 = 432. The off-set mode may be suppressed as described below.

Transput message buffer content:

mess 0	pos(high)*256 [1] + mode
mess 1	bytecount
mess 2	byteaddress
mess 3	pos(low) [1]

1) only used if mode = random

The current drive is the one the last transput message has been sent to.

After break and start of the driver process the current drive is the last existing drive described inside the driver normally drive number 3. Therefore the driver should be reserved if sequential operations are used.

For all data transferring modes the block count is increased at return to point at the next block after the transferred block(s). If the bytecount is zero the blockcount is unchanged.

Below the transput modes for the two different drivers are described.

TRANSPUT MODES OF THE NORMAL DRIVER

BIT	OPERATION	INPUT	OUTPUT
13	RANDOM	x	x
12	WRITE/READ		x
12	WAIT	x	
11	BYTES	x	x
10	INHIBITOFFSET	x	x
9	SENSE ALL	x	x
8	WORD ADDRESS	x	x

TRANSPUT MODES OF THE INITIALITIATION DRIVER

BIT	OPERATION	INPUT	OUTPUT
13	RANDOM	x	x
12	WAIT	x	
11	ADDRESS FIELDS	x	x
10	FORMAT WRITE		x
9	SENSE ALL	x	x
8	WORDADDRESS	x	x

Below the meaning of the above are described.

RANDOM

The position is calculated from the mess 0 and the mess 3 as described for the control message (position).
If mode (random) is false, the position is taken from the current drive description. In this way sequential transfer is possible.
Byte count must be an integral multiple of 512. In case of mode (sense all), 14 bytes must be added.

WRITE/READ

After output of the positioned block the segment is read for check of the written segment. The data are not transferred to the memory. The size of the positioned block must be equal to one segment. If not so status (block error) is set. The size of the buffer may then be 512 bytes and in case of mode (sense all) too it should be on 526 bytes.

WAIT

The message is kept in the event queue until the selected unit changes to or from its ready state. All other modes except mode (random) are dummy.

BYTES

Bytecount bytes are read from current byte and on from the positioned segment. The current byte is defined as the content of mess 2 of a preceeding setposition command.
If bytecount+currentbyte >510 bytes or odd then status (block error) is set.

INHIBIT OFFSET

In case of parity or position error the operation is repeated at least 3 times. If the error has not been removed the position of the heads is moved a little off the track and the operation is again repeated

3 times. This offset operation is avoided if INHIBIT OFFSET is true. In case of output, INHIBIT OFFSET is always set to true regardless of the content of the mode. In case of the initiation driver INHIBIT OFFSET is set to true, too.

SENSE ALL

14 bytes of special status is available with the following content:

byte:	(0:3):	Address Field of last transferred sector
	(4:9):	Error correction Information (ECC)
	(10:13):	Detailed Status

The ECC may be used to error correct an erroneous sector and the Detailed Status for test program purposes. For further details the reader must consult ref. 1. The special status is placed in the last 114 bytes of the specified data buffer, i.e. the bytecount must be enlarged with the 14 bytes.

WORD ADDRESS

The address of mess 2 is interpreted as a word address. In this way a data buffer may be situated in an extended memory.

ADDRESS FIELDS

The address fields on the disc cartridge may be written and read without affecting the data content of the sectors. An address field has the following format:

byte:	(0:1):	Cylinder
	(2:3):	Head shift 8 + Sector
	(4:5):	Next Cylinder
	(6:7):	Next Head shift 8 + Next Sector

The Next Cylinder etc. are used by the controller when multiple data sectors are transferred. Bytecount must be a multiple integral of 8 bytes.

The next sector in sequence is found in "next"-address field. This mode overrides normal data transfer.

FORMAT WRITE

This mode is only allowed during output. The selected track is cleaned by writing all zeros the whole track around, i.e. both address fields and data fields are cleaned.
It should be used prior to writing on address fields on a cartridge the very first time.
This mode overrides both normal data transfer, and transfer of address fields.

ANSWER

All accepted messages are returned with the following content:

mess 0	status
mess 1	bytecount
mess 2	block no. (high)*256 + drive no.
mess 3	block no. (low)

In case of control messages bytecount:=0 and mess 2 and mess 3 are unchanged.

If control(setposition) the status:=current ready state, i.e. ONLINE (zero status) or ATTENTION.
For all other control messages the status:=0.

At return of wait-messages the bytecount:= 2 to avoid troubles if status = 0.

The mess 2 contains the drive number and mess 3 is unchanged.

If the message is unintelligible the bytecount:= 0 and the mess 2 and the mess 3 are unchanged.

STATUS

1b0: disconnected,	power off, hard error, writeprotected during output.
1b1: attention,	not ready

1b5: write protect,	the drive is write protected.
1b6: illegal,	driver is reserved, the specified bytecount is odd, format write has been attempted in an input message.
1b8: block error,	the bytecount in the message does not obey the rules according to the specified mode.
1b9: illegal program,	a channel program command is interpreted as an illegal one.
1b10: parity,	a parity error is detected and could not be corrected by repeating the operation.
1b11: seek error,	the attempted position is outside existing disc drives, multiple segment transfer goes outside the positioned drive.
1b12: position,	the attempted position does not contain a proper address field and it could not be corrected by repeating the operation.
1b14: timer,	no interrupt has been received for 2 sec. after a transput operation.

Ref. 1 : Programmers Reference Manual
RCSL : 44-RT 1191, 1976
By : Lars Myrup Jacobsen

The descriptions found in this manual are valid for following reloctable binary modules :

TTY -	TT005, TT101, and TT200.
PTR -	PR006 and PR100
PTP -	PP002 and PP100
LPT -	LP010 and LP104
MT -	MT006, MT102, MT201, MT301, MT403, MT502, MT805, MT941, and MT951.
CDR -	CR003
OCP -	OP003
RDP -	RP001
CT -	CT009, CT400, and CT109
DKP0 -	DD303
AMX (RC3683) -	AX004
PLT -	PLT03
SP -	SP002, SP101, SP201, SP301, and SP401
FD0 -	FD204 and FD302
BSC -	BSC08
R40X-R40R -	R40X, R40R, R4MR, and R4MX
CPT -	CP003 and CP102
AMX (RC3682) -	AX861
AMX (RC74060) -	AX911
DST0 -	DST00
DOT0	DOT00
SCD -	SCD03, SCD04 (overlapping output) SCD14, SMD02, SX001 - SX301, SMX0 - SMX7
FPAR, FPAX -	FX001, FR000, FX100, and FR100
FH0 -	FH000
LPTM -	LPM00
RC3664 RDP -	CRP00
MD0 -	MD000, MDI00