

RC3500 AUTOLOADSYSTEM.

VERSION 2

Ole Brun Madsen

RECAU-75-

RC3500 AUTOLOAD

RC3500 AUTOLOAD SYSTEM

1. General - Information:

The RC3500 autoloader system consists of a set of input drivers, a primitive loader and a set of reset options.

The autoloader system is placed in a Programmable Read Only Memory (PROM) with 256 16-bit words.

The autoloader system can be activated by pushing the autoloader switch on the front panel, by sending an autoloader signal on the Ext. Autoloader Line, or by a programmed jump into the read only memory from a running program.

The autoloader system will use either the front panel switches or the internal switches as parameters for selection of the actual type of autoloader.

Ref: RCSL:52 - AA059 RC3500 General Purpose Controller,
General Information.

2. Implementation:

The PROM is placed as the last 256 locations (512 bytes) of the physical memory.

The autoloader system consists of the following sections:

INIT: read the switches and select between the two main autoloader functions, LOAD or JMP.

LOAD: used to load the memory from an external source, and, depending of the switches and the load information, stop or jump to execution.

JMP: used to force the CPU to execute at a memory location specified by the switches.

RESET: reset all interrupt levels to a dummy loop except the level selected for input. This level is reset to a driver entry also by the switches.

MEMORY: write all memory locations. Used to secure that all memory locations have correct parity. (Used after power off or memory errors).

MOVE: move the variable part of the driver subroutines to the first 32 locations of the memory, and modify I/O instructions with the selected device number.

LOADER: a program able to transform binary packed input and load the resulting output string into memory.

PROC: a set of procedures used by the drivers to control the devices and to communicate with the loader.

DRIVERS: a set of address tables, one for each driver.
The tables represents the flow through the PROC's.

The PROM memory lay out is described in appendix A.

The actual autoloader system program is described in appendix B.

3. Switches:

The 16 switches is used to specify the detailed structure of the autoloader:

<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>	<u>9</u>	<u>10</u>	<u>11</u>	<u>12</u>	<u>13</u>	<u>14</u>	<u>15</u>
E					D						C	B		A	

A. Main autoloader function.

- 0: jump to the memory location specified by switch 0- 14.
- 1: load as specified by the B, C, D and E switch part.

B. Driver Type.

- 0: RC1500 papertape reader
- 1: CDI100 CDC6000 - channel interface
- 2: LAM116 Asynchronous multiplexor entry
- 3 - 7: not implemented.

C. Execution.

- 0: jump to entry address as specified by the loaded information.
- 1: halt after load.

D. Parameters.

The 6 bit information is used individually by the different driver types, f. ex. the port number on the LAM116 multiplexor.

E. Driver level (0-31).

4. Input format:

In the initial phase the loader accepts a subset of the general loader format (appendix C):

- 1) All addresses are regarded as absolute addresses.
- 2) Only the following two load control chares are affective:
 - [84] define load address.
 - [86] checksum.

The first part of the input string may contain the necessary routines to interpret the whole set of the loader format, as the entry handle the other control chars are placed in the variable section of the autoloader system, and may be overwritten . If an checksum error occurs at load time the autoloader system is stopped.

APPENDIX A MEMORY LAYOUT

e10:	INIT	
	JMP	
	MOVE	
e12:	MEMORY	
e20:	VARIABLE PART:	
	c 20:	PARAMS
	c 30:	I/O SUBROUTINES
	c 40:	RESET
	c 50:	VARIABLES
e30:	LOAD_ _ _ER	
e40:	DRIVERS & PROC	
e42:	UNUSED	
e43:	DRIVER ENTRY AREA	
e44:	AUTOLOAD ENTRY	
c20: c30: c40: c50:	VARIABLE PART AFTER MOVE	


 PROM

APPENDIX B.

AUTOLOAD-AUTOC4A-750430

PAGE 1

REC35SI - 16.10.06 - 30/04/75

SLAM2 - PASCALVERS - 01/01/75

LINE ADDR LIST

PASS 1

1		B.C127,D127,E127,I127			; AUTOLOAD-AUTOC4A-750430
2		B.A127			; MOVE AUTOLOAD PROGRAM
3		K =	256		
4	000400	A0:	LDR	W3 I E0	; W3:=FIRST WORD TO BE MOVED
5	000404		LDR	W2 I -512	; W2:=FIRST WORD IN ROM
6	000410	A1:	LDR	W1 N X3	; W1:=[W3]; W3:=W3+2
7	000412		STR	W1 N X2	; [W2]:=W1; W2:=W2+2
8	000414		BLW	W2 I 0,A1	; LOOP ON ALL WORDS
9	000420	A2:	LDR	W0 I A2	; STOP A2: LOOP
10		E.			; END MOVE PROGRAM
11		P.			

LINE ADDR LIST

PASS 1

```

12      ; CXX:      VARIABLE AREA ADDRESSES:
13      ;          =====
14      ;
15      ; C0 :      CELL ZERO TRAP:
16      ; C2 :      SAVE SWITCHES
17      ; C4 :      SAVE DEVICE
18      ; C6 :      CURRENT CHECKSUM
19      ; C8 :      SAVE CHECKSUM
20      ; C10:      LOAD ADDRESS
21      ; C12:      ENTRY ADDRESS
22      ; C14:      SAVE CHAR
23      ; C16:      SAVE PARAMETER
24      ; C20:      SUBROUTINE AREA START
25      ; C30:      SUB0: RWC W3 X3
26      ; C31:      SUB1: RSC W3 X3
27      ; C32:      SUB2: WWC W3 X3
28      ; C33:      SUB3: WCC W3 X3
29      ; C34:      SUB4: WCC W0 X0
30      ; C35:      SUB5: IST W2 S X0
31      ; C40:      UNMODIFIED VARIABLES
32      ; C50:      CURRENT LOADER ENTRY ADDRESS
33      ; C52:      CURRENT LOADER CTR ADDRESS
34      ; C99:      END VARIABLE AREA
35      ;
36      ; DXX:      DRIVER ADDRESSES:
37      ;          =====
38      ;
39      ; D0 :      DRIVER NEXT ADDRESS SUBROUTINE
40      ; D2 :      DRIVER INIT ENTRY
41      ; D4 :      DRIVER DUMMY LOOP ADDRESS
42      ; D10:      RC500 DRIVER CTR TABEL
43      ; D11:      CDI100 DRIVER CTR TABEL
44      ; D12:      LAM1XX DRIVER CTR TABEL
45      ;
46      ; EXX:      AUTOLOADER GLOBAL ADDRESSES:
47      ;          =====
48      ;
49      ; E10:      MAIN ENTRY
50      ; E12:      MEMORY OVERWRITE
51      ; E20:      VARIABLE PART
52      ; E30:      LOADER:
53      ; E32:      LOADER INITIAL ENTRY
54      ; E34:      LOADER CONTROL INITIAL ENTRY
55      ; E40:      DRIVERS:
56      ; E41:      DRIVER START AREA
57      ; E42:      DRIVER END AREA
58      ; E43:      DRIVER ENTRY TABEL
59      ; E44:      AUTOLOAD INITIAL ENTRY

60      I31 = 31      ; AUTOLOAD LEVEL

```

61 P.

LINE	ADDR	LIST	PASS 1
62		B.A127,B127	: AUTOLOAD MAIN ENTRY:
63		E0:	:
64		K = -512	: ROM FIRST ADDRESS
65		E10:	: AOTOLOAD FIRST SECTION:
66		A10:	: DECODE AUTOLOAD TYPE:
67	177000	RSW W1 X1 0	: W1:=SWITCHES
68	177002	BOP W1 I 1,A20	: GOTO A20 IF AUTOLOAD
69	177006	IST W1 S X0 I31	: ELSE AUTORESET(ADDR:=W1)
70		A20:	: AUTOLOAD:
71	177010	LDR W2 I 0	: W2:=0
72	177012	STI A W2 I C0	: C0:=W2
73	177014	STI A W1 I C2	: C2:=W1
74	177016	LSR W1 11	: W1:=DEVICE ^ INTERRUPT
75	177020	STI A W1 I C4	: C4:=W1
76		A30:	: MOVE AND MODIFY
77			: AUTOLOAD VARIABLE PART:
78	177022	LDR A W3 I C20	: W3:=C20
79	177024	A32: LDR W2 X3+E20-C20	: W2:=WORD TO BE MOVED
80	177030	BGW A W3 I C35,A34	: TEST FOR NO MODIFICATION
81	177034	BLW W2 IN 0 ,A34	: TEST FOR NO MODIFICATION
82	177040	LOR W2 I X1	: MODIFY WITH DEVICE OR INTERRUPT
83	177042	A34: STR W2 N X3	: MOVE WORD; W3:=W3+2
84	177044	PLW A W3 I C99,A32	: LOOP UNTIL C99
85		A40:	: PREPARE RESET ALL LEVELS
86	177050	LDR W1 I 0	: W1:=CURRENT LEVEL:=0
87	177052	LDR W2 I D2	: W2:=ACTUAL DRIVER ENTRY:
88	177056	LDR A W3 I D4	: W3:=DUMMY LOOP ADDRESS
89	177060	LDR A W0 I C35	: GOTO RESET ALL LEVELS
90		E12:	: MEMORY OVERWRITE
91	177062	STI W1 IN X1	: [W1]:=W1; W1:=W1+2
92	177064	SUB W0 I 2	: LOOP FOREVER
93		E.	: END SECTION E10:
94		P.	

```

LINE  ADDR  LIST
95      B.A127,B127      ; AUTOLOAD VARIABLE PART:
96      E20:              ; AUTOLOAD SECTION E20:
97      K = 6             ;
98      C20:              ; FIRST ADDRESS VARIABLE PART
99      C0= 0              ; C0 : CELL ZERO TRAP
100     C2= 2              ; C2 : SAVE SWITCHES
101     C4= 4              ; C4 : SAVE DEVICE
102 000006 C6: 0          ; C6 : CURRENT CHECKSUM
103 000010 C8: 0          ; C8 : SAVE CHECKSUM
104 000012 C10: 256       ; C10: LOAD ADDRESS
105 000014 C12: 256       ; C12: ENTRY ADDRESS
106 000016 C14: 0         ; C14: SAVE CHAR
107 000020 C16: 0         ; C16: SAVE PARAMS
108 000022 C30: RWC W2 X2 0 ; SUB0
109 000024 LDR A W0 I D0    ; GOTO D0
110 000026 C31: RSC W2 X2 0 ; SUB1
111 000030 LDR A W0 I D0    ; GOTO D0
112 000032 C32: WWC W2 X2 0 ; SUB2
113 000034 LDR A W0 I D0    ; GOTO D0
114 000036 C33: WCC W2 X2 0 ; SUB3
115 000040 LDR A W0 I D0    ; GOTO D0
116 000042 C34: WCC W0 X0 0 ; SUB4
117 000044 LDR A W0 I D0    ; GOTO D0
118      C35:              ;
119 000046 IST W2 S X0 0    ; RESET AULOAD DRIVER
120      C40:              ; UNMODIFIED PART:
121 000050 PGW W1 I 30,E30  ; GOTO LOADER IF ALL READY
122 000054 C42: IST W3 X0 0  ; ELSE RESET LEVEL W1
123 000056 ADD W1 I 1       ; W1:=W1+1
124 000060 STB W1 C42+1     ; MODIFY IST-INSTR.
125 000064 LDR A W0 I C35   ; LOOP ON C35
126 000066 C50: E32         ; INITIAL LOADER ENTRY
127 000070 C52: E34         ; INITIAL LOAD CTR SEQ
128      C99:              ; END VARIABLE PART:
129      K = E20+C99-C20
130      E.                ; END SECTION E20:
131      P.

```

LINE ADDR LIST

PASS 1

```

132          B.A127,B127          ; AUTOLOAD: LOADER:
133          E30:                  ; AUTOLOAD SECTION E30:
134          A0:                    ; LOADER ENTRY:
135 177152      LCE   W3 I   A12      ; W3:=DUMMY RETURN ADDRESS
136 177156      PEW   W1 IN 2,A0      ; SKIP TO STX
137          A10:                  ; NORMAL INPUT ENTRY:
138 177164      LDI A W0 I   C50      ; GOTO [C50]
139          E32:                  ; INITIAL LOADER ENTRY
140 177166      STI A W1 I   C14      ; C14:=W1
141 177170      BGW   W1 I   31,A20    ; GOTO A20 IF<>ASCIICTR
142 177174      PEW   W1 I   3,A14     ; GOTO A14 IF ETX
143          A12:                  ; NORMAL WAIT READ EXIT
144 177200      LCE   W0 I   A10      ;
145          A14:                  ; ETX:
146 177204      LDI A W1 I   C2       ; W2:=SWITCHES
147 177206      A16: BOP   W1 I   16,A16 ; LOOP IF NOGO
148 177212      LDI A W0 I   C12      ; GOTO C12
149          A20:                  ; NON ASCII CONTROL:
150 177214      PEW   W1 I   86,A50    ; GOTO A50 IF SUMCHECK
151 177222      ADD   W1      C6       ; W1:=W1+CHEKSUM
152 177226      STI A W1 I   C6       ; SUMCHECK:=W1
153 177230      LDI A W1 I   C14      ; W1:=CHAR
154 177232      BGW   W1 I   79,A30    ; GOTO A30 IF LOAD CTR
155          A22:                  ; COLLECT DATA:
156 177240      AND   W1 I   15       ; W1:=W1[4 BIT]
157 177242      LSL   W2      4       ; W2:=W2*16
158 177244      ADD   W2 I   X1       ; W2:=W2+W1
159 177246      LDI A W1 I   C14      ; W1:=C14
160 177250      BLW   W1 I   48,A12    ; GOTO A12 IF MORE DATA
161 177254      LDR   W0 I   X3       ; ELSE RETURN W3
162          P.

```

LINE ADDR LIST

PASS 1

```

163          A30:          ; LOAD CONTROL:
164 177256      LDR      W3 I  A12      ; W3:=A12
165 177262      LDR      W2 I  0        ; W2:=0
166 177264      LDI  A  W0 I  C52      ; GOTO [C52]
167          E34:          ; INITIAL LOAD CTR SEQ:
168 177266      BEW      W1 IN 84,A12    ; SKIP I<> LOAD ADDRESS
169          ; SKIP MODE

170          A40:          ; LOAD ADDRESS:
171 177274      XSR      W0 I  X3        ; GET LOAD ADDRESS
172 177276      STI  A  W2 I  C10      ; C10:=W2
173 177300      STI  A  W2 I  C12      ; C12:=W2
174          A42:          ; LOAD DATA:
175 177302      LDR      W2 I  0        ; W2:=0
176 177304      LDR      W3 I  A12      ; W3:=A12
177 177310      XSR      W0 I  X3        ; GET DATA
178 177312      LDI  A  W1 I  C10      ; W1:=LOAD ADDRESS
179 177314      STR      W2 N  X1        ; LOAD DATA
180 177316      STI  A  W1 I  C10      ; AND UPDATE LOAD ADDRESS
181 177320      SUB  A  W0 I  A46-A42    ; GOTO A42
182          A46:

183          A50:          ; SUMCHECK:
184 177322      LDI  A  W1 I  C6        ; W1:=C6
185 177324      STI  A  W1 I  C8        ; C8:=W1
186 177326      LDR      W3 I  A12      ; W3:=A12
187 177332      XSR      W0 I  X3        ; INPUT CHECKSUM
188 177334      A52:      BEW      W2 I  C8,A52 ; SUMCHECK TRAP
189 177340      LDR      W2 I  0        ; W2:=0
190 177342      STI  A  W2 I  C6        ; RESET CHECKSUM
191 177344      SUB  A  W0 I  A54-A42    ; CONTINUE A42
192          A54:
193          E.
194          P.

```

; END AUTOLOAD LOADER

 LINE ADDR LIST

PASS 1

```

195          B.A127,B127          ; DRIVER SECTION:

196          E40:                  ; AUTOLOAD SECTION E40:

197          E41=-256
198 177346    0,R.(E41-E40)>1

199 177400    A0:  LSR    W1    8          ; A0: W1:=W1>8
200 177402    A2:  AND    W1 I 127        ; W1:= 7 BIT DATA
201 177406          IST    W1 S X1 I31    ; TRANSFERE TO LOADER
202 177410          DEW    W1 I 3,D4      ; GOTO D4 IF ETX
203 177414    A4:  LDC    W1 I 1          ; A4: W1:=-1
204 177416          LDR A W0 I D0        ; GOTO D0

205 177420    A10: LDI A W1 I C2          ; A10: W1:=SWITCHES
206 177422          LSR    W1    5        ; W1:=PARAM
207 177424          AND    W1 I 63        ; MASK OFF
208 177426          STI A W1 I C16       ; C16:=PARAMS
209 177430          LDR    W0 I D0        ; GOTO D0

210 177434    A12: LOR    W1 I 13<5+3<11 ; W1:=LAM CTRW
211 177440    A13: LOR    W0 I C33        ; CALL OUTCTR C33
212 177442    A14: SUB    W1    C16       ; W1:=W1-C16
213 177446          BSZ    W1 I 31,A0     ; TEST FOR CORRECT
214          ; PORTNO AND INPUT
215 177452          SUB A W0 I A16-A4     ; GOTO A4

216 177454    A16: LDB    W2    X2        ; W2:=NEXT BYTE
217 177456          LDC    W2 I X2        ; W2:=NEXT TABEL ADDRESS
218 177460          LDR A W0 I D0        ; GOTO D0

219 177462    A18: LDR    W1 I 0          ; W1:=0
220 177464    A20: LDR A W0 I C34        ; CALL OUTCTR C34:
221 177466    A22: LDR A W0 I C30        ; CALL READ C30:
222 177470    A23: LCE A W0 I C31        ; WAIT NEXT INTERRUPT
223 177472    A24: LDR A W0 I C31        ; CALL STATUS C31

224          P.
```

PASS 1

```

225 H. ; HALFWORD MODE:
226 177474 D10: -A24 ; RSC CLEAR STATUS
227 177475 -A20 ; WCC WAIT INTERRUPT
228 177476 -A22 ; RWC INPUT
229 177477 -A2 ; IST TO LOADER
230 177500 -D2 : JMP LOOP ALL

231 177501 D11: -A23 ; WAIT NEXT STATUS
232 177502 -A2 ; IST TO LOADER
233 177503 -D2 : JMP LOOP ALL

234 177504 D12: -A10 ; SUB FETCH PARAMS
235 177505 -A12 ; 110B CTRW
236 177506 -A4 ; SUB W1:=-1
237 177507 A25: -A20 ; WCC WAIT INTERRUPT
238 177510 -A14 ; IST CHECK AND IST TO LOADER
239 177511 -A16 ; JMP TABEL JUMP
240 177512 -A25 ; TO A25
241 177513 W. ; FULLWORD MODE

242 E42:
243 H. ; HALFWORD MODE
244 E43=-32
245 177514 0,R.(E43-E42)
246 E43: ; DRIVER CONTROL TABEL
247 177740 -A16,-D10 ; TYPE 0: RC500
248 177742 -A16,-D11 ; TYPE 1: CDI100
249 177744 -A16,-D12 ; TYPE 2: LAM116
250 177746 0, 0 ; TYPE 3
251 177750 0, 0 ; TYPE 4
252 177752 0, 0 ; TYPE 5
253 177754 0, 0 ; TYPE 6
254 177756 0, 0 ; TYPE 7

255 W. ; FULLWORDMODE:

256 177760 D4: LCE A W0 I D4 ; DUMMY LOOP
257 177762 D2: LDI A W2 I C2 ; W2:=SWITCHES
258 177764 AND W2 I 14 ; W2:=DRIVER TYPE
259 177766 ADD A W2 I E43 ; W2:=TABEL ENTRY E43
260 177770 D0: LDB W3 N X2 ; W3:=[W2]; W2:=W2+1
261 177772 LDC W0 I X3 ; GOTO -W3

262 177774 -512 ;
263 E44: ; AUTOLOAD INITIAL ENTRY:
264 177776 LDI A W0 I -4 ; GOTO [-4]

265 E. ; END SECTION E40:
266 E50: ; END AUTOLOAD ADDRES
267 E.
```

APPENDIX C: LOADER FORMAT:

The external representation is packed 4-bit binary information and loader directives in a 7-bit standard ASCII character set.

The basic units are addresses and data . Addresses and data are regarded as 16 bits units.

Addresses are regarded as virtual segment addresses and transformed into the actual segment addresses before they are loaded.

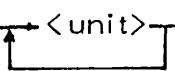
Data or addresses are represented by up to four chars (see below). Leading Zeros may be omitted. The most significant bits are placed in the first char.

<compiler>:	Defines which compiler version has generated the source text.
<source>:	The source text to be loaded Source texts may be nested, which means that the loader will stack the current load environment and create a new segment space to load into.
<external procedure>:	The name of the external procedure is substituted by the address of the external procedure.
<standard procedure>:	The name of the standard procedure is substituted by the calling sequence of the standard procedure.
<load address>:	The current load address is set to the defined address.
<address chain>:	The first address is regarded as the start of an address chain that successively is substituted with the second address.
<check sum>:	The loader performs a checksum operation and reset current checksum.

<load text>::=

→ STX → SO → <compiler> → <source> → SI → EXT →

<source>::=

→ [80] → <unit> → [81] →


<unit>::=

→ <external procedure>

→ <standard procedure>

→ <load address>

→ <address chain>

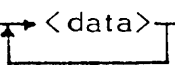
→ <checksum>

→ <source>

→ <data>

→ <address>

<external procedure>::=

→ [82] → <data> → <address> →


<standard procedure>::=

→ [83] → <data> →

<load address>::=

→ [84] → <address> →

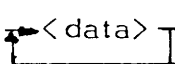
<address chain>::=

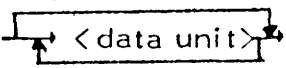
→ [85] → <address> → <address> →

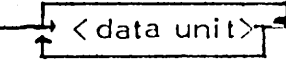
<check sum>::=

→ [86] → <data> →

<compiler>::=

→ [87] → <data> →


$\langle \text{data} \rangle ::=$
 $\langle \text{data end} \rangle \rightarrow$

$\langle \text{address} \rangle ::=$
 $\langle \text{address end} \rangle \rightarrow$

$\langle \text{data unit} \rangle ::=$
 $\rightarrow [32 - 47] \rightarrow$

$\langle \text{data end} \rangle ::=$
 $\rightarrow [48 - 63] \rightarrow$

$\langle \text{address end} \rangle ::=$
 $\rightarrow [64 - 79] \rightarrow$