99000682

RCSL No: 52-AA1172

Edition: October, 1983

Author: Henning Jakobsen

Title:

RC3502
Diagnostic Program, User's Guide

RC International

Keywords:

RC3502, Real-Time Pascal, Test program.

Abstract:

This manual describes the diagnostic test program for the RC3502 system. By this program you may make your own small program loops and run them on an RC3502 for scope looping.

(26 printed pages).

RCSL No 42-I 2164

TABLE OF CONTENTS PAGE

1. SURVEY 1

2. INPUT TO DIAG 4
   2.1 Commands 4
   2.2 Instructions 5
   2.3 Program Lines 5

3. OUTPUT 8
   3.1 Command Responses 8
   3.2 Error Messages 8
   3.3 Programmed Output 8
   3.4 Other Output Types 9

4. FUNCTIONS 10
   4.1 Program Control 10
   4.2 Computing 11
   4.3 Printing 11
   4.4 I/O Instructions 12
   4.5 Datatransfer 15
   4.6 Wait Functions 16

5. SUMMARY 17
   5.1 List of Commands 17
   5.2 List of Instructions 17

1. SURVEY

The "DIAG" system is a utility, allowing you to make your own diagnostic test program loops and run them instantly on an RC3502.

DIAG is started via opsys.

```
 >opsys
 run diag
```

DIAG works as a little stand-alone simple BASIC-like system without backing storage.

Example
Reading of the registers in a VDC controller.

```
 10:  CONTROLC 96               ; RESET VDC
 20:  CONTROLC 1312             ; SET TIMER
 30:  LET REG 48
 40:  WHILE REG < 64
 50:    ADD REG, 3, SREG
 60:    PRINT "  REGISTER  ", REG, ".. ", SREG, ":"
 70:    REPEAT
 80:      GOSUB LINE 150
 90:      INCR REG
100:    UNTIL REG > SREG
110:    PRINTNL
120:  ENDW
130:  STOP
140:  GOTO LINE 30
150:  MUL REG, 256, A            ; A:= REG * 256
160:  ADD A, 64, CW              ; CW:= A + 64
170:  CONTROLC CW
180:  INWORD VAL                 ; VAL := VDC´S DATAIN
190:  PRINTH "  ", VAL           ; PRINT AS HEX NUMBER
200:  RETURN

RESERVE 18                       ; RESERVE INTERRUPT LEVEL
RUN
  REGISTER  48 .. 51 :  0205  0200  0200  0200
  REGISTER  52 .. 55 :  0214  0200  0200  0200
  REGISTER  56 .. 59 :  0000  0200  0287  0000
  REGISTER  60 .. 63 :  0200  0200  0200  0200
STOP AT 130
DUMP
   REG    64  0040   SREG   63   003F
   A   16128  3F00   CW  16192   3F40
   VAL   512  0200
RELEASE 18
```

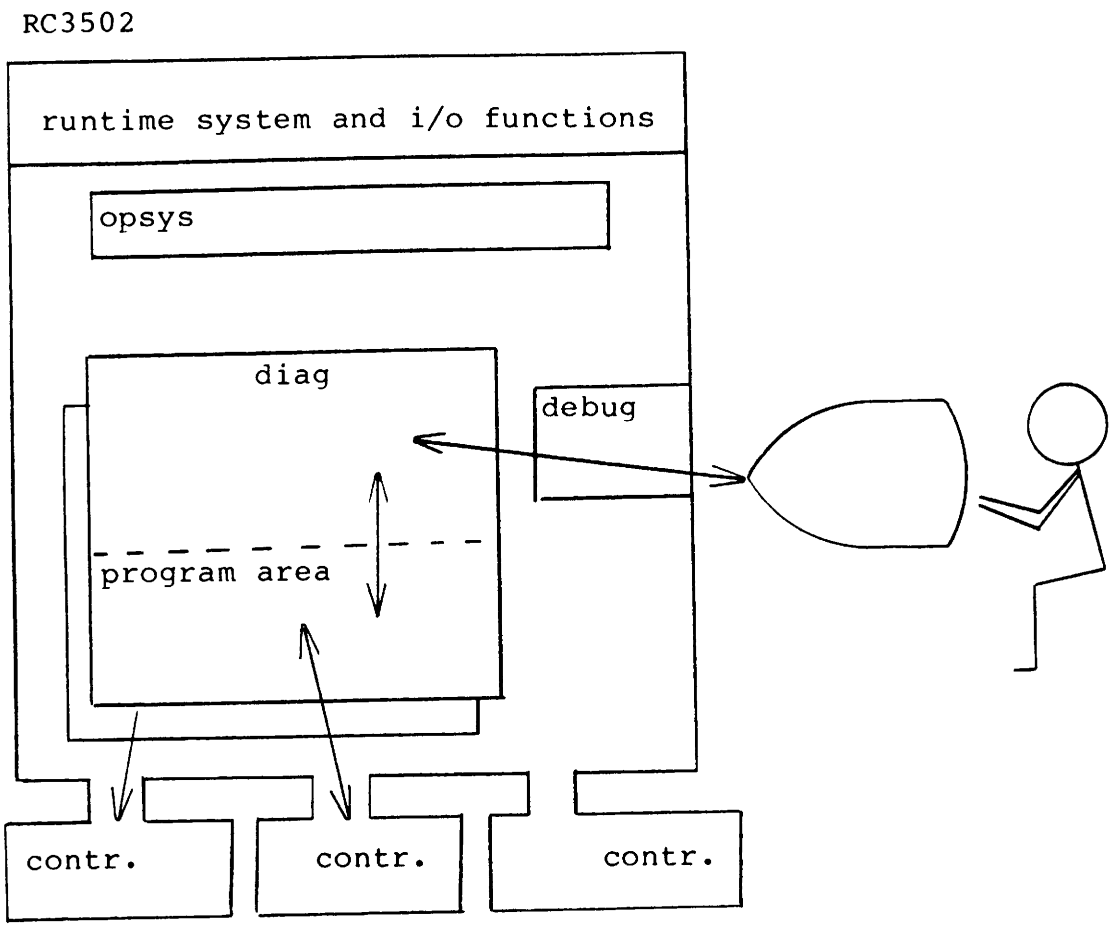

Fig. 1. Configuration.

Components of the Diagnostic System

1. An RC3502 with debug console and suitable memory and controllers.

2. The normal REAL-TIME PASCAL (RTP) runtime system including all I/O-functions.

3. The normal ´opsys´ for operator communication and process management.

4. The DIAG program.

5. The user generated program (typed in at the debug console), which can perform all wanted I/O-functions at the installed controllers.

6. Manuals:

   RC3502 Reference Manual
   RC3502 Operator Guide
   REAL-TIME PASCAL Reference Manual

   This manual

   Manuals for the controllers in question.

2. INPUT TO DIAG 2.

Input is given line for line from console keyboard.

DIAG takes 3 forms of input:

- Commands, specifying operations on the user program.
- Instructions, specifying REAL-TIME PASCAL functions, for instant execution.
- Numbered program lines to be saved in the program area for later execution.

2.1 Commands 2.1

The underlined letters must be given. Line numbers are given as numbers in 1..3999 .

new	Erase all program lines and variables.
clear	Set all variables to zero.
delet f t	Delete a range of program lines.
list	List all program lines.
list f	List program lines starting with line f.
list f t	List program lines in the interval f..t .
renumber	Set line numbers to 10, 20, 30, ...
run	Renumber program lines and execute from line 10.
run f	Renumber and execute from line f.
stop	Stop execution and print actual line number. Only this command is active after run.
dump	List all variable names and values.
continue	Continue after a stop.

reserve i	Reserve interrupt level i. Level range is 5..122 . After reservation all executions are performed in a channel statement. Only one level may be reserved at the same time. But you can start another incarnation of DIAG via opsys for another level.
release	Release the reserved interrupt level.
help	List command names and instruction names implemented in your version, not only those mentioned in this obsolete manual.

2.2 Instructions

Instructions are typed as program lines without line number. After syntax check the line is saved as line 4010.

```
4010: <instruction>
4020: stop
4030: goto 4010
```

Then a ´run 4010´ is performed automatically. After ´stop at 4020´, command ´con´ will repeat the instruction.

2.3 Program Lines

Program lines are saved in the program area according to the line number. The syntax is:

```
<lineno> : <instr-name> <parameter list>
```

lineno : must be in the interval 1..3999 .
instr-name: must be from the instruction list (see 5.2).
parameterlist: according to the instr-name.

Parameter types are:

const	A value typed as a decimal number in -32678..32676 or a hexadecimal number typed as # followed by up to 4 hex digits.

lineno	A line number.
text	Up to 12 characters inclosed in " or ´ .
relation	One of these: =, <>, <, <=, >=, >, ult, andnz. ult is "unsigned less than" x andnz y is: x and y <> 0.
identifier	A name of an integer variable. The name is up to 11 characters, starting with a letter.
val	Const or identifier.

DIAG has room for 30 texts, 30 variables, and 399 program lines.

Instructions are provided for program control (while, endwhile, stop, goto), computing, printing, input/output, data transfer, and interrupt handling.

It is recommended to use the following delimiters between the elements of a line:

space, comma, (and). Use : after lineno.

<u>Example</u>
Simulate the debugger M command:

```
 10: LET D = # 600
 20: LET L =     0
 30:  WHILE L < 4
 40:    PRINTH "   ", M, &8, ´:´, D
 50:    LET N = 0
 60:    REPEAT
 70:      GETW M, D, W
 80:      PRINTH " ", W
 90:      ADD 2, D, D
100:      INCR N
110:    UNTIL N >= 8
120:    PRINTNL
130:    INCR L
140:  ENDW
150:  STOP
160:  GOTO LINE 20
LET M # C2
STOP AT 4020
RUN
   00C2:0600  D78C  0002  6700  EF45  AFAF  AFAA  1736  6400
   00C2:0610  0967  0054  E600  0123  1111  2222  3344  AADD
   00C2:0620  D080  60AF  07AF  AA12  CC33  0009  E800  00C2
   00C2:0630  0001  4000  CF11  2234  C207  246E  6100  1044
STOP AT 150
```

3. OUTPUT 3.

After start DIAG prints version date,
e.g. 831005 diag:

3.1 Command Responses 3.1

list	Lists some or all program lines.
help	Lists commands and insructions.
renumber	Checks ´goto´, ´until´, and ´endwhile´, so some errors may be reported.
run	Executes a renumber, so some errors may appear here, too.
stop	After stopping, the actual line number will be printed.
dump	List all variable names and values.
reserve	An error occurs if the level is in use or no device is installed at that level.

3.2 Error Messages 3.2

Program lines are checked for syntaxical errors, so some error printout may appear.

The format is: ? <explanation> <number>

3.3 Programmed Output 3.3

stop	prints "stop at line <number>" when executed.

print instructions makes the output you want.

3.4 Other Output Types

Because DIAG allows you to misuse the whole system, any strange output may appear. For instance ´exceptions´ may occur because of misuse. Exceptions may also arise from errors in the DIAG program, so please report incomprehensible exceptions to the RC3502 test program group (1983: HEJ in Aarhus).

How to continue after exception:

```
  >opsys
  remove diag  run diag
```

The ´list´ and ´dump´ commands may be used to show what survived the situation.

4. FUNCTIONS

The function of most of the instructions can be found in the Instruction list and RTP Reference Manual. Here the special diag instructions are explained.
The typical execution time for an instruction is ca 1 m sec.

4.1 Program Control

The command ´run´ starts execution of the user program line for line. By using the program control statements you can alter the sequential execution of the program.

GOTO s	continue execution at specified line.
GOSUB s RETURN	execute subroutine. The RETURN statement in the subroutine then works as a goto to the line just after GOSUB.
IF a rel b, t, f	The condition a rel b, where rel is a relation between a and b is computed. If the condition is true then goto line t else goto line f. f may be omitted.

IF n<128, 70
IF a>4,90,30
	if n<128 then goto line 70.
	if a > 4 then goto 90 else goto 30

REPEAT ... UNTIL a rel b	The program loops, the lines between REPEAT and UNTIL is executed as long as condition a rel b is false. After 1024 loops is a 2 seconds pause to give time for a ´stop´.
WHILE a rel b ... ENDWHILE	The program loops as long as condition a rel b is true.
NOOP	Dummy operation, goto next line.

STOP	Print ´stop at line <number>´ and wait for commands.

4.2 Computing

Notation: i is an identifier = name of an integer (16 bits) variable.
v, w is an identifier or a constant.
Expressions are not allowed.

LET i v	v is copied into i. RTP: i:= v.
INCR i	i:= i + 1 modulo 64K.
ADD v w i	i:= v + w without overflowcheck.
SUB v w i	i:= v - w without overflowcheck.
MUL v w i	i:= v * w multiplication with check.
DIV v w i	i:= v DIV w division with check, decimals are thrown away.
MOD v w i	i:= v MOD w v modulo w
OR v w i	i:= v OR w bitwise addition.
AND v w i	i:= v AND w bitwise multiplication.
MADD v w i	same as ADD
MSUB v w i	same as SUB
UADD v w i	unsigned add.
USUB v w i	unsigned sub.
UMUL v w i	unsigned mul.
UDIV v w i	unsigned div.
UMOD v w i	unsigned mod.

4.3 Printing

The print instructions have max. 7 parameters. Printparameters are of 3 types:

1. text : max. 12 characters inclosed in " or ´

2. value: identifier, number, or hexnumber. Expressions are not allowed.

3. char : & <number> with number in 0..127

Texts are printed with outtext(). Chars are printed with outchar(). Values are printed with outinteger() in PRINT and PRINTNL, and with outhex() in PRINTH and PRINTHNL. After each value a space character is printed.
PRINTNL and PRINTHNL prints a new-line after the

parameters.

PRINTHNL 1983 — prints a decimal value converted to hexadecimal.

PRINT "This is a lo", "ng text. "

PRINT "short # "

PRINT "BELL", &7

4.4 I/O Instructions

BYTEC m,d,v

- The current interrupt level is cleared and the contents of parameter v are transferred to the position given by m,d . Often m,d points to an external memory address, e.g. in COM204.

WORDC m,d,v

- As byteclear, but a word is transferred.

CONTR v

- Parameter v is transferred to the CONTROL register of the reserved interrupt level. The level is not cleared so the program continues without wait for interrupt.

CONTROLC v

- Parameter v is transferred to the CONTROL register of the reserved interrupt level. If the level is not ´timed out´ then it is cleared so the program continues when a new interrupt arrives.

CLEARL

- clears the interrupt level and waits for an interrupt. If the level has status ´timed out´, the call has no effect.

INBYTEB i v

- Read-block-of-bytes from the reserved level to msg databuffer. v is number of bytes wanted. Terminates when EOI=true or wanted bytes are read.

INWORD i

- The DATAIN register of the reserved level is transferred to i. If EOI=true after the call then i is undefined.

INWORDB i,v

- Read-block-of-words, works as INBYTEB, but in wordmode.

INWORDC i

- The DATA-IN register of the reserved level is transferred to i. IF EOI=true after the call then i is undefined. If the level is ´timed out´ then the program continues, else the interrupt level is cleared and waiting for an interrupt.

IOGI i,f,v

- General input. Function f is performed on the device. The word v is transferred to STATUS-OUT, and the resulting word obtained, according to f, from DATA-IN or STATUS-IN is moved to i.

IOGO f,v

- General output. Function f is performed on the device. The word v is transferred, according to f, to the DATA-OUT, STATUS-OUT, or CONTROL register.

IOWBWC v

- Write-block-of-words and clear level after last word. The words are the first v words from req databuffer.

OUTBYTEB i v

- Write-block-of-bytes, v bytes from req databuffer.

OUTWORDB i,v

- Works as OUTBYTEB, but in wordmode.

OUTWORD v

- The word v is transferred to the DATA-OUT register.

OUTWORDC v

- The word v is transferred to the DATA-OUT register. If status is ´timed out´ then continue, else clear interrupt and wait for next interrupt.

SENS i,v

- The word v is transferred to the STATUS-OUT register and the response is transferred from STATUS-IN to i. The level is not cleared.

SENSEC i,v,c,m

- The word v is transferred to the STATUS-OUT register. If STATUS-IN and m = c then i:= STATUS-IN and continue, else the level is cleared and the procedure is repeated when the next interrupt arrives, unless the status is changed to ´timed out´.

SETI

- Set interrupt on the reserved level.

TIMED t (f)

- If the level is ´timed out´ then the status is cleared and program continues in line t, else goto next line (or line f if specified).

4.5 Datatransfer

GETID i	Reads the value of the ID register, IDR201.
GETB m,d,i	Fetch of a byte from the memory address given by (m,d).
GETW m,d,i	As GETB, a 16 bit word is transferred.
PUTB m,d,v	Transfer of a byte to a memory address.
PUTW m,d,v	Transfer of a word to a memory address.
READB m,d,i	As GETB without parity check.
READW m,d,i	As GETW without parity check.
COMPA n t f	If the first n bytes are equal then goto t else goto f. f may be omitted.
EXCH	The databuffers msg and req are exchanged.

FILL a b c d e f g

```
Fill bytes into the req-buffer.
a= -1 : The first b bytes are set to 0.
a= -2 : the first b bytes are set to 255.
a= -3 : the first b bytes are set to hex 55, aa, 55, ...
a= -4 : the first b bytes are set to 1, 2, 3, ...
a> -1 : The values b to g are filled into
        the words a, a+1, a+2, ...
Parameters c to g are optional.
```

PRINTM a b n	The msg databuffer is printed from byte a to byte b, n bytes pr line.

4.6 Wait Functions

In DIAG the following variables are declared:

```
my_sem  : semaphore
req     : reference (allocated to my_sem)
msg     : reference (always nil before a wait)
chn_msg : reference
```

The instruction:

```
     CWAITISD v d j k l
```

is executed as:

```
     if not nil (msg) then return (msg);
     case ctrwaitisd (v, msg, my_sem, d) of
a_interrupt : goto j;
a_semaphore : goto k;
a_delay     : goto l
     end;
```

The other waits are handled in similar way.

5. SUMMARY 5.

5.1 List of Commands 5.1

```
NEW
CLEAR
DELETE  first last
LIST    first last
RENUMBER
RUN    first
STOP
DUMP
CONTINUE
RESERVE   level
RELEASE
HELP
```

5.2 List of Instructions 5.2

```
m   : memno in 126..254
d   : displacement        (m,d) is a memory address
i   : identifier
v   : value or identifier
r   : relation ( =  <>  <  <=  >=  >  ult  andnz  )
l   : line no
( ) : optional parameter
```

Name Parameters	RTP Equivalence
GOTO l	goto line l
GOSUB l	call routine at line l
RETURN	return from routine
IF v r v l (l)	if v r v then goto l else goto l
REPEAT	repeat
UNTIL v r v	until v r v
WHILE v r v	while v r v do begin
ENDW	end; (* while *)
NOOP	(* empty statement *)
STOP	outtext ("stop at line xxx") end
ADD v v i	i := madd (v, v)
AND v v i	i := v and v
DIV v v i	i := v div v
INCR i	increment_mod_64K (i)
LET i v	i := v
MADD v v i	i := madd (v, v)
MOD v v i	i := v mod v

MSUB v v i	i := msub (v, v)
MUL v v i	i := v * v
OR v v i	i := v or v
SUB v v i	i := msub (v, v)
UADD v v i	i := uadd (v, v)
UDIV v v i	i := udiv (v, v)
UMOD v v i	i := umod (v, v)
UMUL v v i	i := umul (v, v)
USUB v v i	i := usub (v, v)

PRINT params	print values as decimal numbers
PRINTNL params	as PRINT followed by a new line.
PRINTH params	print values as hex numbers
PRINTHNL params	as PRINTH followed by a new line.

BYTEC m d v	byteclear ((m,d) , v)
CLEARL	clearlevel
CONTR v	control (v, chn)
CONTROLC v	controlclr (v)
EOI l (l)	if eoi then goto l else goto l
INBYTEB i v	inbyteblock (i, 6, 5+v, msg)
INWORD i	inword (i, chn)
INWORDB i v	inwordblock (i, 6, 5+v, msg)
INWORDC i	inwordclr (i)
IOGI i v v	iogi (i, v, v, chn)
IOGO v v	iogo (v, v, chn)
IOWBWC v	iowbwc (6, 5+v, req)
OUTBYTEB i v	outbyteblock (i, 6, 5+v, req)
OUTWORD v	outword (v, chn)
OUTWORDB i v	outwordblock (i, 6, 5+v, req)
OUTWORDC v	outwordclr (v)
SENS i v	sense (i, v, chn)
SENSEC i v v v	senseclr (i, v, v, v)
SETI	setinterrupt (chn)
TIMED l (l)	if timedout then goto l else goto l
WORDC m d v	integerclear ((m,d) , v)

COMP v	compare v bytes in msg and req
EXCH	msg :=: req
FILL v v (v v v v v)	fill data into req-buffer
GETB m d i	i:= (m,d)
GETID i	i:= getid
GETW m d i	i:= (m,d)
PUTB m d v	(m,d) := v
PUTW m d v	(m,d) := v
PRINTM v v v	printmessage (msg, v, v, v)
READB m d i	readbyte (i, (m,d))
READW m d i	readword (i, (m,d))
READRA i v	readram (i, v)
WRITERA v v	writeram (v, v)

WRITERAMC	v v	writeramclr (v, v)
CWAITID	v v l l	ctrwaitid (v, v)
CWAITIS	v l l	ctrwaitis (v, msg, mysem)
CWAITISD	v v l l l	ctrwaitisd (v, msg, mysem, v)
WAITD	v	waitd (v)
WAITI		waiti
WAITID	v l l	waitid (v)
WAITIS	l l	waitis (msg, mysem)
WAITISD	v l l l	waitisd (msg, mysem, v)

RETURN LETTER

Title: RC3502 Diagnostic Program, User's Guide RCSL No.: 52-AA1172

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

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

____

____

____

____

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

____

____

____

____

How can this manual be improved?

____

____

____

____

Other comments?

____

____

____

____

____

Name: ______________________ Title: ______________________

Company: ____

Address: ____

Date: __________

Thank you

42-i 1288

. . . . Fold here . . . .

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

Affix
postage
here

A/S REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark