RCSL No: 52-AA1056

Edition: June, 1981

Author: Bo Bagger Laursen

Title:

RC3502 REAL TIME PASCAL
Character Input/Output Routines

RCSL 42-i1592

A/S REGNECENTRALEN
af 1979

Keywords:

REAL TIME PASCAL, RC3502, Software.

Abstract:

A set of primitive input/output routines for character input/output is described.

(28 printed pages).

Printed by A/S Regnecentralen af 1979, Copenhagen

RCSL 42-i1592

CONTENTS PAGE

1. INTRODUCTION

This manual describes a set of routines which may be used for input/output to the OPERATOR process or another character oriented input/output driver.

Communication takes place via variables of type zone.

The type zone is a simple implementation of the zone concept known from ALGOL8 and MUSIL.

Chapter 2 describes the initialization of variables of type zone.

Chapter 3 describes the output procedure OUTCHAR and a set of output procedures, which use the procedure OUTCHAR.

Chapter 4 describes the input procedure INCHAR and a set of input procedures, which use the procedure INCHAR.

An overview of the types and routines is listed in the appendices.

2. INITIALIZATION

The type zone is defined in the environment IOENVIR as:

```
zone = RECORD
          driver     ,
          answer     : ↑ semaphore;
          dataready  ,
          free       : semaphore;
          cur        : reference;
          u2val      ,
          state      : byte
          readstate  ,
          nextp      ,
          lastpos    : integer
       END;
```

driver - points to the driver semaphore (e.g. the OPERATOR semaphore).

answer - points to the semaphore, where answers arrive from the driver.

dataready - holds the answers from the driver, if this semaphore is specified as 'answer' in the call of openzone or openopzone (see 2.1 and 2.2). 'dataready' is normally specified, when the zone is used for input from OPERATOR.

free - holds the empty messages.

cur - refers the message which is currently in use for reading input or writing output.

u2val - whenever a message is signalled to the driver semaphore, the u2 field (result field) in the message is initialized to u2val.

state - holds the result field (u2) from the message referred by 'cur'. State is updated when a message is taken from 'free' as a new current message for output in the procedure outchar, or when a message is taken from 'dataready' in the procedure opwait as a new current message holding input data.

readstate - specifies the state of the zone when used for input after each call of one of the input routines.

Readstate is initialized to -1 after call of 'openzone' or 'openopzone'.

The following interpretation of readstate holds:

readstate = - 1

No input was ready ('cur = nil') when an input procedure was called or the current input buffer became empty during call.

readstate = 0

The call of an input procedure succeeded with no syntax errors.

readstate > 0

This value range is intended for indication of a syntax error detected during the call of the input procedure. The routines in this manual do not deliver positive values.

nextp - is the index of the next position in current message for reading or writing.

lastp - is the index of the last position in current message for reading or writing.

2.1 Openzone


```
PROCEDURE openzone (
            VAR z              : zone;
            driver             ,
            answer             : ↑semaphore;
            bufs               : integer;
            VAR home           : pool 1;
            v1, v2, v3, v4     : byte);
```

prepares the zone z for input/output.

z - the zone which is initialized.

driver - a pointer to the (OPERATOR or) driver semaphore.

answer - a pointer to a semaphore where answers arrive from the driver.

bufs - specifies the number of messages which the procedure will allocate the zone z. The messages are placed in 'z.free'.

home - the messages are allocated from the pool home. The messages must be able to hold a variable of type

```
RECORD
   first, last, next: integer
END;
```

v1, v2, v3, v4 - the u-fields in the messages are initialized to v1, v2, v3, and v4. v2 is used to reset the result field (u2), whenever a message is signalled to the driver.

2.2 Openopzone 2.2

```
PROCEDURE openopzone (
            VAR z              : zone;
            driver             ,
            answer             : ↑ semaphore;
            bufs               : integer;
            VAR home           : pool1;
            v1, v2, v3, v4     : byte);
```

- Messages from the pool 'home' must be able to hold variables of type:

```
RECORD
   first, last, next: integer;
   name             : alfa
END;
```

The procedure initializes the field 'name' to 'own.incname' in all messages, whereafter it performs as the procedure openzone.

3. OUTPUT

If the process does not want to be activated, when an output message returns from the driver, it uses:

ref (z.free)

as the actual parameter 'answer' in the call of 'openzone' or 'openopzone'. In this way empty output messages return directly to the semaphore 'z.free' as available messages for continued output.

If the process wants to supervise the answers, it uses:

ref ("mainsemaphore")

in the call of openzone or openopzone. The messages must be signalled to 'z.free' afterwards.

3.1 Outend

PROCEDURE outend (VAR z: zone);

- signals the current message 'z.cur' to the driver semaphore 'z.driver ↑'.

3.2 Outchar

PROCEDURE outchar (VAR z: zone; ch: char);

- places the character ch in the current message 'z.cur'.

  If no current message is available, a wait is performed on the semaphore 'z.free'.

  If the message becomes full, the procedure OUTEND is called.

3.3 Outtext

3.3

PROCEDURE outtext (VAR z : zone; text : alfa);

- writes the variable 'text' by calling outchar. The character '#' acts as a stop character.

3.4 Outfill

3.4

PROCEDURE outfill (VAR z: zone; filler: char; rep: integer);

- repeats the character 'filler', 'rep' times.

3.5 Outinteger

3.5

PROCEDURE outinteger (VAR z: zone; i, pos: integer);

- writes the number 'i' on decimal form. If the number occupies less than 'pos' positions (including the '-' character, if negative), the number is prefixed a number of spaces to complete the pos positions.

3.6 Outhex

3.6

PROCEDURE outhex (VAR z: zone; i, pos: integer);

- writes the number 'i' as four hexadecimal digits. If pos is greater than four, pos -4 space characters are prefixed the number.

3.7 Outdate

3.7

```
PROCEDURE outdate (VAR z: zone; date: coded_date);
```

- writes the parameter date with the layout:

YYYY.MM.DD

(For the definition of type coded_date, see appendix B. The type coded_date is used throughout the run time system, especially the timer system (see description of the procedure sendtimer).

3.8 Outtime

3.8

```
PROCEDURE outtime (VAR z: zone; time: coded_time);
```

- writes the parameter time with the layout:

HH.MM

(For the definition of type coded_time, see appendix B).

3.9 Outnl

3.9

```
PROCEDURE outnl (VAR z: zone);
```

- writes the character nl and signals the message to the driver by calling outend.

4. INPUT

If the process wants to be activated only when an input message returns from the driver, it uses:

ref (z.dataready)

as the actual parameter 'answer' in the call of 'openzone' or 'openopzone'.

More commonly, the process is also activated by other events. In that situation, it specifies:

ref (mainsemaphore)

as the actual parameter 'answer' in this call of 'openzone' or 'openopzone'.

4.1 Opwait

PROCEDURE opwait (VAR z: zone; VAR inputpool: pool 1);

- is used to wait for specific input to the zone, which returns directly to 'z.dataready', or to a mainsemaphore together with other messages. If a message is queued at 'z.dataready', this message is taken. Otherwise, a wait is performed on 'z.answer↑'.

  Opwait checks when a message arrives that it originates from the zone by means of the routine 'ownertest' and that (u1 MOD 8) = 1 (read). Other messages are queued temporarily in the zone until a zone message returns. The queued messages are put back in the mainsemaphore and the zone message prepared for later calls of INCHAR.

4.2 Optest


```
FUNCTION optest (VAR z: zone): boolean;
```

- is true if a message is queued at 'z.dataready', otherwise false. This may be used to avoid a wait in the procedure opwait.

Example:

```
IF optest (z) THEN
   BEGIN
      opwait (z, inputpool);
      (* process inputdate from zone z *)
   END
   ELSE
      (* do something else *)
```

4.3 Opin


```
PROCEDURE opin (VAR z: zone);
```

- signals a message from 'z.free', if any, to the driver semaphore 'z.driver↑'.

4.4 Inchar


```
PROCEDURE inchar (VAR z: zone; VAR ch: char);
```

- the next character from the current message 'z.cur' is read.

  If the message becomes empty, it is signalled to 'z.free'.

  After the call the variable 'z.readstate' indicates the state of the zone with the interpretation:

z.readstate	
0	Successful reading
- 1	This buffer was empty before call. The character nl is returned.

4.5 Ininteger

PROCEDURE ininteger (VAR z: zone; VAR i: integer);

- reads a decimal number according to the pseudo syntax:

$$\left\{\text{<nondigit>}\right\}_{0}^{n} \left\{\begin{matrix}+\\-\end{matrix}\right\}_{0}^{1} \left\{\text{<digit>}\right\}_{1}^{n}$$

Digits are read as long as the number is in the range -32768..32767.

z.readstate	
0	At least one digit is read.
- 1	No digit was met in the buffer. The buffer is empty and the value 0 is returned.

Examples:

Input	Result (i):
a b c 1 2 * a b c d e	12
a b c + - - + 1 7 9 a b	179
a b c + - 0 0 1 2 3 4 5 6 7	-12345
a b c 3 2 7 6 8	3276

(↓ indicates the value of nextp before and after the call).

4.6 Inhex

PROCEDURE inhex (VAR z: zone; VAR i: integer);

- reads a hexadecimal number according to the pseudo syntax:

$$\left\{\text{<non hex digit>}\right\}_{0}^{n} \quad \left\{\text{<hex digit>}\right\}_{1}^{n}$$

z.readstate	
0	At least one hex digit is read
- 1	No hex digit was met in the buffer. The value 0 is returned and the buffer is empty.

Hex digits are read as long as the number is in the range #h0000..#hffff.

4.7 Inname

PROCEDURE inname (VAR z: zone; VAR name: alfa);

- reads a name of maximum 12 characters after the syntax:

$$\left\{\text{<not letter or \_>}\right\}_{0}^{n} \quad \text{<letter or \_>} \left\{\text{<letter, digit or \_>}\right\}_{0}^{11}$$

Example:	Result:
a bc	a
_abc89_6c;	_abc89_6c
12ab	ab

The characters are delivered in the parameter 'name' from left to right. 'name' is not initialized by 'inname', so 'name' must be initialized before the call. The variable 'z.nextp' may be used to calculate the number of characters read (inclusive leading blanks).

z.readstate	
- 1	No <letter or _> was met in the buffer. The buffer is empty and possibly a number of spaces was skipped.
0	At least one <letter or _> is transferred to 'name'.

A. TYPE DEFINITIONS A.

```
TYPE (* coded_date and coded_time are predefined in PLATONENV *)

  coded_date = PACKED RECORD
                        year_after_1900 : 0 .. 127;
                        month           : 0 ..  15;
                        day             : 0 ..  31;
                      END;

  coded_time = PACKED RECORD
                        compiler_version : 0..31;   (* 5 bits *)
                        hour             : 0..31;   (* 5 bits *)
                        minute           : 0..63;   (* 6 bits *)
                      END;
(* zone is predefined in IOENVIR *)

  zone = RECORD
           driver : ↑semaphore;      (* operator process        *)
           answer : ↑semaphore;      (* answers returns here    *)
           dataready  : semaphore;   (* buffers with data       *)
           free   : semaphore;       (* free buffers            *)
           cur    : reference;       (* current buffer          *)
           u2val  : byte;            (* u2 to driver            *)
           state  : byte;            (* resultcode from answer *)
           readstate : integer;      (* 0: ok, >0: error,-1:cur=nil *)
           nextp  : integer;         (* next position in databuf *)
           lastpos : integer;        (* last position in databuf *)
         END;
```

B. ROUTINE DECLARATIONS B.

```
(* the following routines are predefined in IOENVIR *)

PROCEDURE openzone ( VAR z: zone; driv, answ: ↑semaphore;
  bufs: integer; VAR home: pool 1; v1, v2, v3, v4: byte     );
EXTERNAL;


PROCEDURE openopzone ( VAR z: zone; driv, answ: ↑semaphore;
  bufs: integer; VAR home: pool 1; v1, v2, v3, v4: byte     );
EXTERNAL;


PROCEDURE outend ( VAR z: zone);
EXTERNAL;


PROCEDURE outchar ( VAR z: zone; t: char);
EXTERNAL;


PROCEDURE outtext ( VAR z: zone; text: alfa);
EXTERNAL;


PROCEDURE outfill ( VAR z: zone; filler: char; rep: integer);
EXTERNAL;


PROCEDURE outinteger ( VAR z: zone; num, pos: integer);
EXTERNAL;


PROCEDURE outhex ( VAR z: zone; num, pos: integer);
EXTERNAL;


PROCEDURE outnl(VAR z : zone);
EXTERNAL;


PROCEDURE outdate(VAR z : zone; date : coded_date);
EXTERNAL;


PROCEDURE outtime(VAR z : zone; time : coded_time);
EXTERNAL;


PROCEDURE opin ( VAR z: zone);
EXTERNAL;
```

```
PROCEDURE opwait ( VAR z: zone; VAR inputpool: pool 1);
EXTERNAL;

PROCEDURE inchar ( VAR z: zone; VAR t: char);
EXTERNAL;

PROCEDURE ininteger ( VAR z: zone; VAR num: integer);
EXTERNAL;

PROCEDURE inhex ( VAR z: zone; VAR num: integer);
EXTERNAL;

PROCEDURE inname ( VAR z: zone; VAR name: alfa);
EXTERNAL;

FUNCTION optest ( VAR z: zone): boolean;
EXTERNAL;
```

C. EXAMPLES

testiolist 81.08.11. 16.42.

```
job a 1 time 6 0 size 100000 perm mini 25 2
mode list.yes
( btestio = SET 1 mini
btestio = pascal80 ioenvir
scope user btestio
finis
)

PROCESS testio( VAR sv : system←vector );

CONST
  version           = 3;
  empty             = "############";
  stop              = "stop########";
  linelength        = 80;
  firstindex        = 6 + alfalength;
  lastindex         = firstindex + linelength - 1;
  readcode          = 1;
  writecode         = 2;
  no←of←opbuffers   = 3;
  no←of←inbuffers   = 1;
  no←of←outbuffers  = no←of←opbuffers - no←of←inbuffers;

TYPE
  opbuffer = RECORD
               !  first ,
               !  last  ,
               !  next  : integer;
               !  name  : alfa;
               !  chars : ARRAY (firstindex..lastindex) OF char
             END;

  ktable   = ARRAY (1..10) OF integer;

VAR
  stat              ,
  i                 ,
  num               : integer;
  name              ,
  txt               : alfa;
  t                 : char;

  oppool            : pool no←of←opbuffers OF opbuffer;

  kb                ,
  z                 : zone;
```

```
testiolist  81.08.11.   16.42.

   47
   48           k                    : ktable:= ktable( 0, 1, 12, 123, 1234, 12345
   49                                                   32767, -32768, -12345, -1 )
   50
   51         BEGIN
   52     1   !
   53     2   !
   54     3   ! openopzone ( z, sv(operatorsem), ref(z.free),
   55     4   ! no←of←outbuffers, oppool, writecode,0,0,0 );
   56     5   !
   57     6   ! outtext ( z, own.incname );
   58     7   ! outfill ( z, ".", 3);
   59     8   ! outinteger ( z, version, 8);
   60     9   ! outnl ( z);
   61    10   !
   62    11   ! outhex ( z, #h0123, 2);
   63    12   ! outhex ( z, #h4567, 3);
   64    13   ! outhex ( z, #h89ab, 4);
   65    14   ! outhex ( z, #hcdef, 6 );
   66    15   ! outtext( z, "  out.#hidde");
   67    16   ! outnl  ( z);
   68    17   !
   69    18   ! FOR i:= 32 TO 127 DO
   70    19   !   outchar ( z, chr(i));
   71    20   ! outnl ( z);
   72    21   !
   73    22   ! FOR i:= 57 DOWNTO 0 DO
   74    23   !   outchar ( z, chr(i));
   75    24   ! outnl ( z);
   76    25   !
   77    26   ! txt:= "programtest.";
   78    27   ! outtext ( z, txt);
   79    28   ! outchar ( z, nl);
   80    29   !
   81    30   ! FOR i:= 1 TO 12 DO
   82    31   !   BEGIN
   83    32   !   ! txt(13-i):= "#";     (*  stop mark  *)
   84    33   !   ! outtext ( z, txt);
   85    34   !   ! outfill ( z, "-", i);
   86    35   !   ! outchar ( z, nl)
   87    36   !   END;
   88    37   !
   89    38   ! outend( z);
   90    39   !
   91    40   ! FOR i:= 1 TO 10 DO
   92    41   !   BEGIN
```

```
testiolist  81.08.11.   16.42.

   93    42   !   ! num:= -3;
   94    43   !   ! REPEAT
   95    44   !   ! ! outinteger ( z, k(i), num);   outchar ( z, ",");
   96    45   !   ! ! outhex ( z, k(i), num);   outchar ( z, ",");
   97    46   !   ! ! num:= num+3
   98    47   !   ! UNTIL num=9;
   99    48   !   ! outchar ( z, nl);
  100    49   !   END;
  101    50   !
  102    51   ! outend ( z);
  103    52   !
  104    53   ! openopzone ( kb, sv(operatorsem), ref(z.dataready),
  105    54   ! no-of-inbuffers, oppool, readcode, 1, 1, 1);
  106    55   !
  107    56   ! outtext ( z, "ininteger:");
  108    57   ! outnl ( z);
  109    58   ! outtext ( z, "ready: #    ");
  110    59   ! outchar ( z, bel);
  111    60   ! outend ( z);
  112    61   !
  113    62   ! REPEAT
  114    63   ! ! ininteger ( kb, num);
  115    64   ! ! IF kb.readstate < 0 THEN
  116    65   ! !   BEGIN
  117    66   ! !   ! opin ( kb);
  118    67   ! !   ! opwait ( kb, oppool);
  119    68   ! !   ! ininteger ( kb, num);
  120    69   ! !   END;
  121    70   ! ! stat:= kb.readstate;
  122    71   ! ! outinteger ( z, num, 6);
  123    72   ! ! outfill ( z, sp, 2);
  124    73   ! ! inchar ( kb, t);
  125    74   ! ! outchar ( z, t);
  126    75   ! ! outinteger ( z, stat, 4);
  127    76   ! ! outchar ( z, nl);
  128    77   ! ! IF t = nl THEN outchar ( z, bel);
  129    78   ! ! outend ( z)
  130    79   ! UNTIL num = -1 ;
  131    80   !
  132    81   ! outchar ( z, nl);
  133    82   ! outtext ( z, "inhex:    ");
  134    83   ! outnl ( z);
  135    84   !
  136    85   ! REPEAT
  137    86   ! ! inhex ( kb, num);
  138    87   ! ! IF kb.readstate = -1 THEN      (*  buffer empty  *)
```

```
testiolist  81.08.11.   16.42.

  139    88    ! !   BEGIN
  140    89    ! !   ! opin ( kb);
  141    90    ! !   ! opwait ( kb, oppool);
  142    91    ! !   ! inhex ( kb, num)
  143    92    ! !   END;
  144    93    ! ! stat:= kb.readstate;
  145    94    ! ! outinteger ( z, num, 6);
  146    95    ! ! outhex ( z, num, 6);
  147    96    ! ! outfill ( z, sp, 2);
  148    97    ! ! inchar ( kb, t);
  149    98    ! ! outchar ( z, t);
  150    99    ! ! outinteger ( z, stat, 4);
  151   100    ! ! outchar ( z, nl);
  152   101    ! ! IF t = nl THEN outchar ( z, bel);
  153   102    ! ! outend ( z)
  154   103    ! UNTIL num = -1 ;
  155   104    !
  156   105    ! outchar ( z, nl);
  157   106    ! outtext ( z, "inname:   ");
  158   107    ! outnl ( z);
  159   108    !
  160   109    ! REPEAT
  161   110    ! ! name:= empty;
  162   111    ! ! inname ( kb, name);
  163   112    ! ! IF kb.readstate = -1 THEN    (*  buffer empty  *)
  164   113    ! !   BEGIN
  165   114    ! !   ! opin ( kb);
  166   115    ! !   ! opwait ( kb, oppool);
  167   116    ! !   ! inname ( kb, name)
  168   117    ! !   END;
  169   118    ! ! stat:= kb.readstate;
  170   119    ! ! outtext ( z, name);
  171   120    ! ! outfill ( z, sp, 2);
  172   121    ! ! inchar ( kb, t);
  173   122    ! ! outchar ( z, t);
  174   123    ! ! outinteger ( z, stat, 4);
  175   124    ! ! outchar ( z, nl);
  176   125    ! ! IF t = nl THEN outchar ( z, bel);
  177   126    ! ! outend ( z)
  178   127    ! UNTIL name = stop ;
  179   128    !
  180   129    ! FOR num := minint TO maxint DO
  181   130    !   BEGIN
  182   131    !   ! outinteger ( z, num, 6);
  183   132    !   ! outhex ( z, num, 6);
  184   133    !   ! outchar ( z, nl)
```

```
testiolist  81.08.11.   16.42.

  185  134   !   END;
  186  135   !
  187  136   ! outend ( z)
  188  137   !
  189  138   END .
```

RETURN LETTER

RC3502 REAL TIME PASCAL

Title: Character Input/output Routines **RCSL No.:** 52-AA1056

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

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

_____

_____

_____

_____

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

_____

_____

_____

_____

How can this manual be improved?

_____

_____

_____

_____

Other comments?

_____

_____

_____

_____

_____

Name: _____ **Title:** _____

Company: _____

Address: _____

Date: _____

Thank you

42-i 1288

. . . . . Fold here . . . . . .

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

Affix
postage
here

REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark