

BASIC listing: RTCB.BAS

```
*****
! *                                     R T C B                                     *
! *-----*
! * Task           : Makes different routines available for reading      *
! *                : and writing realtime clock data.                      *
! * Author        : Michael Tischer                                       *
! * Developed on   : 07/24/87                                              *
! * Last update    : 04/07/95                                             *
! *****
!
```

DECLARE FUNCTION RTCRead% (Address%)

DECLARE FUNCTION RTCDT% (Address%)

```
CLS                                           'Clear screen
PRINT "RTCB (c) 1987, 92 by Michael Tischer": PRINT
PRINT "Information from the battery operated realtime clock"
PRINT "===== "
PRINT
```

```
IF (RTCRead(&HE) AND 128) = 128 THEN          'Bit 7 = 0 --> Battery O.K.
```

```
PRINT "          WARNING! Clock battery is low"
```

```
ELSE
```

```
PRINT "- The clock is running in";
```

```
PRINT (RTCRead(&HB) AND 2) * 6 + 12; "hour mode"
```

```
PRINT "- the time: ";
```

```
PRINT USING "##:"; RTCDT(&H4);
```

```
PRINT USING "##:"; RTCDT(&H2);
```

```
PRINT USING "##:"; RTCDT(&H0)
```

```
PRINT "- the date: ";
```

```

PRINT USING "##"; RTCDT(&H8);
PRINT "-";
PRINT USING "##"; RTCDT(&H7);
PRINT "-";
PRINT USING "####"; RTCDT(&H9) + 1900
PRINT

```

END IF

```

! *****
! * RTCDT: Reads the contents of a date or time memory location, and *
! * converts the contents to decimal. *
! *-----*
! * Input : ADDRESS% = The memory address (0-63) *
! * Output : The contents of this address in decimal notation *
! *****
!

```

FUNCTION RTCDT% (Address%)

```

Ret% = RTCRead(Address%) 'Read contents of the memory address
IF (RTCRead(&HB) AND 2) <> 0 THEN 'Test for BCD mode
    RTCDT% = (Ret% AND 15) + INT(Ret% / 16) * 10 'Convert BCD to DEC
ELSE
    RTCDT% = Ret%
END IF

```

END FUNCTION

```

! *****
! * RTCRead: Reads the contents of a memory location on the RTC. *
! *-----*
! * Input : ADDRESS% = The memory address (0-63) *

```

```

'* Output : The contents of this address, or -1 if this address      *
'*          contains an invalid number                               *
'*****
'

```

```

FUNCTION RTCRead% (Address%)

```

```

    IF (Address% < 0) OR (Address% > 63) THEN
        RTCRead% = -1
    ELSE
        OUT &H70, Address%      'Memory location in the RTC address register
        RTCRead% = INP(&H71)    'Read contents of the RTC data register
    END IF

```

```

END FUNCTION

```

```

'*****
'* RTCWrite: Writes a memory location to the RTC.                  *
'*-----*
'* Input   : ADDRESS% = The memory address (0-63)                  *
'*          : CONTENTS% = New contents of this memory location      *
'*****
'

```

```

SUB RTCWrite (Address%, Contents%)

```

```

    OUT &H70, Address%      'Memory location in the RTC address register
    OUT &H71, Contents%    'Write new contents to the RTC data register

```

```

END SUB

```

Pascal listing: RTCP.PAS

```
{*****}
{*                R T C P                *}
{*-----*}
{* Task          : Provides two functions for accessing the *}
{*               battery operated realtime clock.          *}
{*-----*}
{* Author        : Michael Tischer                        *}
{* Developed on   : 07/10/87                               *}
{* Last update    : 02/27/92                               *}
{*****}
```

program RTCP;

Uses Crt; { Add CRT unit }

const RTCAdrPort = \$70; { RTC address register }
RTCDtaPort = \$71; { RTC data register }

SECONDS = \$00; { Addresses for some RTC memory locations }
MINUTE = \$02;
AN HOUR = \$04;
DAY = \$07;
MONTH = \$08;
YEAR = \$09;
STATUSA = \$0A;
STATUSB = \$0B;
STATUSC = \$0C;
STATUSD = \$0D;
DIAGNOSE = \$0E;
HUNDREDEYEAR = \$32;


```

begin
  if (RTCRead(STATUSB) and 2 = 0)           { BCD or binary mode? }
  then RTCDT := RTCRead(Address)             { Binary }
  else                                         { BCD }
  begin
    Wert := RTCRead(Address);                { Get contents of memory location }
    RTCDT := (Wert shr 4) * 10 + Wert and 15  { Convert to binary }
  end
end;

```

```

{ ***** }
{ * RTCWrite: Writes a value to the RTC memory location. * }
{ * Input   : ADDRESS = Address of memory location in the RTC * }
{ *         : CONTENT = New value for this memory location * }
{ * Output  : None * }
{ * Info   : This address should range from 0 to 63 * }
{ ***** }

```

```

procedure RTCWrite(Address : integer;      { Address of memory location }
                   Content  : byte);       { New contents }

```

```

begin
  Port[RTCAAdrPort] := Address;             { Pass RTC address }
  Port[RTCDtPort]   := Content              { Write new value }
end;

```

```

{ ***** }
{ *                               MAIN PROGRAM                               * }
{ ***** }

```

```

begin

```

```

ClrScr;                                { Clear screen }
writeln('RTCP (c) 1987, 92 by Michael Tischer'#13#10);
writeln('Information from the battery operated realtime clock');
writeln('===== '#13#10);
if RTCRead(Diagnose) and 128 = 0 then    { Is the battery O.K.? }
begin                                   { Yes --> Battery O.K. }
    writeln('- The clock is in ', (RTCRead(STATUSB) and 2)*6+12,
        ' hour mode');
    writeln('- The time : ', RTCDT(AN HOUR), ':', RTCDT(MINUTE):2,
        ':', RTCDT(SECONDS):2);
    write('- The date : ');
    writeln(RTCDT(MONTH), '-', RTCDT(DAY), '-',
        RTCDT(HUNDREDEYEAR), RTCDT(YEAR));
end
else                                     { Dead battery }
    write('          Attention! Clock battery is dead')
end.

```