CMOS RAM CONTROL FOR PC-AT

from an idea by H. van den Bosch

The program listed here is written in Turbo Pascal to compile a small program, RTC-NVR.EXE, that enables owners of PC/ATs or any other IBM PC compatible fitted with a real-time clock (RTC) circuit to examine and, if necessary, change the contents of the non-volatile CMOS RAM that holds the time and date, as well as system configuration ('set-up') information.

Any PC-AT or compatible has a 64-byte CMOS RAM and a real-time clock powered by a re-chargeable battery. In the standard IBM PC-AT (who still has such an oldie?), the two functions are combined in a Motorola Type MC146818 integrated circuit on the motherboard. In the latest generation of AT compatibles, however, the RTC and the RAM are usually contained in one of the VLSI components that form part of the manufacturer's chip set. In general, owners of a PC-AT need not worry about the exact location of the RTC and non-volatile RAM: the only thing that counts is that the two can be found at the right memory address. Fortunately, most clone manufacturers keep to the I/O address assignment drawn up by IBM.

A timed surprise?

To be informed by the computer that the set-up information is lost owing to a system malfunction or an exhausted battery is at best a temporary nuisance and at worst the beginning of a real hassle to unearth notes made a long time ago on hard disk drive parameters, RAM bank configurations, wait states, and so on.

Although the required information can be refurnished relatively quickly with the aid of the set-up utility provided with the computer, it is useful in many cases to have the RTC/RAM contents available on paper in the form of a hex dump.

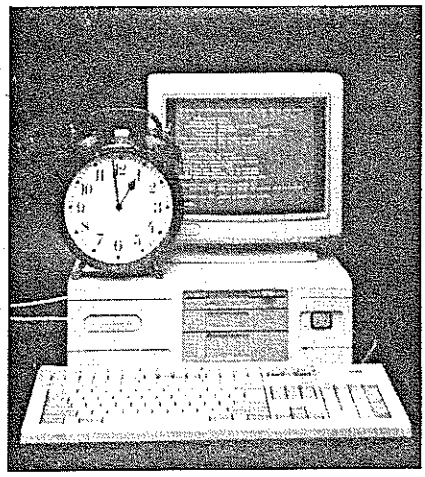
The author recently suffered a malfunction in the power supply of his PC-AT compatible. After this had been repaired, the machine on being switched on reported a hard disk controller fault, and could not be made to boot from drive C: as usual. The previously written utility RTC-NVR.EXE was available on floppy disk, however, and on being run indicated that the two checksum bytes at addresses 2E and 2F (see Tables 1 and 2) had been corrupted, probably as a result of the power supply malfunction. The bytes were recalculated, modified, and the machine performed a normal boot-up operation.

Document your settings

The listing in Fig. 1 is typed into Turbo Pascal and then compiled to obtain the required .EXE file. Routines 'upcase' and

```
program RTC_NVR;
{This program reads the contents of the Non-Volatile RAM on board the
 Real Time Clock chip of a PC/AT and allows the data to be changed.}

uses crt;

type str2 = string[2];

var  addr, data: byte;
     ch: char;

{*************************************************************************}
function cb_hex (b: byte): str2;
{cb_hex converts byte b into a string with the hexadecimal representation
 of byte b}

const hexsigns: array [0..15] of char = '0123456789ABCDEF';

begin
  cb_hex := hexsigns [b shr 4] + hexsigns [b and $0F]
end; {function cb_hex}

{*************************************************************************}
procedure read_RAM;
{read_RAM reads the contents of the non-volatile RAM}

var line, column: byte;

begin
  writeln ('hex address   hex data');
  writeln ('-----------   ------------------------------------');
  for line := 0 to 7 do
  begin
    write ('$',cb_hex(line * 8),'           '); {write address & 11 spaces}
    for column := 0 to 7 do
    begin
      port[$70] := line * 8 + column; {write address to RTC}
      write ('$',cb_hex(port[$71]),'  '); {read CMOS RAM and write contents}
    end;
    writeln
  end
end; {procedure read_RAM}

{*************************************************************************}
begin {program}
  repeat
    clrscr; {clear screen}
    writeln ('RTC-RAM information');
    read_RAM;
    write ('change RAM (choose N to end)? [Y,N] ');
    ch := upcase (readkey); {read keyboard and convert input character
                             to upper case if necessary}
    write (ch);
    if ch = 'Y'
    then
    begin
      {**** WARNING **** CHECKSUM IS NOT RECALCULATED ****}
      writeln;

      write ('address in hex (e.g $3f) or decimal format? '); readln (addr);
      write ('                                    data? '); readln (data);
      port[$70] := addr;
      port[$71] := data;
      read_RAM;

      write ('press any key to continue...');
      repeat until keypressed; {wait}

      {clear keyboard buffer without using dummy variables}
      repeat if readkey='' then; until not keypressed

    end {if}
  until (ch <> 'Y') and (ch = 'N');
end.
                                                          900015-11
```

Fig. 1. Listing of the Turbo-Pascal program used to compile RTC-NVR.EXE.



```
RTC-RAM information
hex address    hex data
-----------    ----------------------------------------
$00            $44  $03  $42  $00  $08  $00  $03  $10
$08            $11  $89  $26  $02  $50  $80  $00  $00
$10            $24  $00  $F0  $00  $71  $80  $02  $00
$18            $00  $29  $00  $00  $00  $00  $00  $00
$20            $00  $00  $00  $00  $00  $00  $00  $00
$28            $00  $00  $00  $00  $00  $00  $02  $30
$30            $00  $00  $19  $F9  $EC  $F4  $B0  $1E
$38            $E7  $45  $BA  $1C  $C4  $1F  $4F  $71
change RAM (choose N to end)? [Y,N] Y
address in hex (e.g $3f) or decimal format? $01
                                          data? $00
hex address    hex data
-----------    ----------------------------------------
$00            $10  $00  $43  $00  $08  $00  $03  $10
$08            $11  $89  $26  $02  $50  $80  $00  $00
$10            $24  $00  $F0  $00  $71  $80  $02  $00
$18            $00  $29  $00  $00  $00  $00  $00  $00
$20            $00  $00  $00  $00  $00  $00  $00  $00
$28            $00  $00  $00  $00  $00  $00  $02  $30
$30            $00  $00  $19  $F9  $EC  $F4  $B0  $1E
$38            $E7  $45  $BA  $1C  $C4  $1F  $4F  $71
press any key to continue...                  900015 - 12
```

Fig. 2. Print-out to illustrate the operation of the non-volatile RAM utility. The computer used for this test was a NEAT-286/4 AT compatibie. By studying the dump carefully, you can see when this article was written, and how many seconds it took to modify the data at address $01, the seconds alarm in the RTC.

Byte	Function	Address
0	Seconds	00
1	Second Alarm	01
2	Minutes	02
3	Minute alarm	03
4	Hours	04
5	Hour alarm	05
6	Day of week	06
7	Day of month	07
8	Month	08
9	Year	09
10	Status Register A	0A
11	Status Register B	0B
12	Status Register C	0C
13	Status Register D	0D

Table 2. Real-time clock addresses.

Addresses	Description
00–0D	Real-time clock information
OE	Diagnostic status byte
0F	Shut-down status byte
10	Diskette drive type byte (drives A and B)
11	Reserved
12	Fixed disk drive type (drives C and D)
13	Reserved
14	Equipment byte
15	Low base memory byte
16	High base memory byte
17	Low expansion memory byte
18	High expansion memory byte
19–2D	Reserved
2E–2F	2-byte CMOS checksum
30	Low expansion memory byte
31	High expansion memory byte
32	Date century byte
33	Information flags (set during power-on)
34–3F	Reserved

Table 1. CMOS RAM address map.

'readkey' are procedures resident in Turbo Pascal. Data is read via I/O port 71, while port 70 serves to address the 64 bytes. An example of the output of the program is shown in Fig. 2. The RAM addresses may be entered in hexadecimal (preceded by a $ sign) or in decimal.

The program is also suitable for the popular Amstrad PC1640, and should also work on PC-XTs equipped with a multi-I/O card, although this has not been tested.

Finally, be sure to copy RTC-NVR.EXE to a floppy disk if you have compiled the program on hard disk. Make the floppy bootable by adding CONFIG.SYS, COMMAND.COM and an AUTOEXEC.BAT file that calls up RTC-NVR.EXE.

Byte(s)	Usage	Default
0–9	RTC time and date parameters	--
10	RTC control register A	070H
11	RTC control register B	002H
12	RTC control register C	--
13	RTC control register D	--
14–19	Time and date when machine was last used	--
20	User RAM checksum	--
21-22	Enter key translation token	01C0DH
23-24	Forward key translation token	02207H
25-26	Joystick fire button 1 translation token	0FFFFH
27-28	Joystick fire button 2 translation token	0FFFFH
29-30	Mouse button 1 translation token	0FFFFH
31-32	Mouse button 2 translation token	0FFFFH
33	Mouse X direction scaling factor	00AH
34	Mouse Y direction scaling factor	00AH
35	Initial video mode and drive count	020H
36	Initial video character attributes	007H
37	Size of RAM disk in 2-Kbyte blocks	000H

Table 3. Amstrad PC1640 non-volatile RAM address map.

CMOS RAM CONTROL FOR PC-AT

from an idea by H. van den Bosch

The program listed here is written in Turbo Pascal to compile a small program, RTC-NVR.EXE, that enables owners of PC/ATs or any other IBM PC compatible fitted with a real-time clock (RTC) circuit to examine and, if necessary, change the contents of the non-volatile CMOS RAM that holds the time and date, as well as system configuration ('set-up') information.

Any PC-AT or compatible has a 64-byte CMOS RAM and a real-time clock powered by a re-chargeable battery. In the standard IBM PC-AT (who still has such an oldie?), the two functions are combined in a Motorola Type MC146818 integrated circuit on the motherboard. In the latest generation of AT compatibles, however, the RTC and the RAM are usually contained in one of the VLSI components that form part of the manufacturer's chip set. In general, owners of a PC-AT need not worry about the exact location of the RTC and non-volatile RAM: the only thing that counts is that the two can be found at the right memory address. Fortunately, most clone manufacturers keep to the I/O address assignment drawn up by IBM.

A timed surprise?

To be informed by the computer that the set-up information is lost owing to a system malfunction or an exhausted battery is at best a temporary nuisance and at worst the beginning of a real hassle to unearth notes made a long time ago on hard disk drive parameters, RAM bank configurations, wait states, and so on.

Although the required information can be furnished relatively quickly with the aid of the set-up utility provided with the computer, it is useful in many cases to have the RTC/RAM contents available on paper in the form of a hex dump.

The author recently suffered a malfunction in the power supply of his PC-AT compatible. After this had been repaired, the machine on being switched on reported a hard disk controller fault, and could not be made to boot from drive C: as usual. The previously written utility RTC-NVR.EXE was available on floppy disk, however, and on being run indicated that the two checksum bytes at addresses 2E and 2F (see Tables 1 and 2) had been corrupted, probably as a result of the power supply malfunction. The bytes were recalculated, modified, and the machine performed a normal boot-up operation.

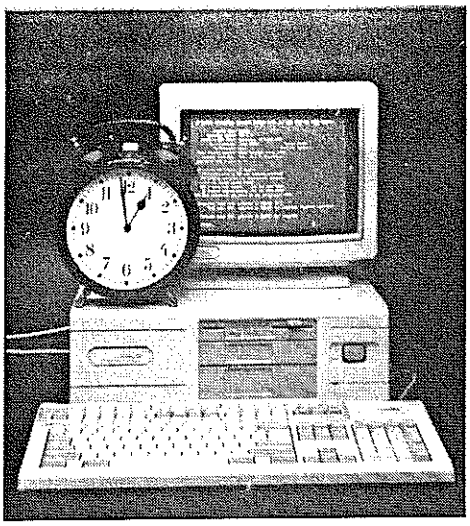
Document your settings

The listing in Fig. 1 is typed into Turbo Pascal and then compiled to obtain the required .EXE file. Routines 'upcase' and

```
program RTC_NVR;
{This program reads the contents of the Non-Volatile RAM on board the
 Real Time Clock chip of a PC/AT and allows the data to be changed.}

uses crt;

type str2 = string[2];

var  addr, data: byte;
     ch: char;

{*****************************************************************************}
function cb_hex (b: byte): str2;
{cb_hex converts byte b into a string with the hexadecimal representation
 of byte b}

const hexsigns: array [0..15] of char = '0123456789ABCDEF';

begin
  cb_hex := hexsigns [b shr 4] + hexsigns [b and $0F]
end; {function cb_hex}

{*****************************************************************************}
procedure read_RAM;
{read_RAM reads the contents of the non-volatile RAM}

var line, column: byte;

begin
  writeln ('hex address   hex data');
  writeln ('-----------   -----------------------------------');
  for line := 0 to 7 do
  begin
    write ('$',cb_hex(line * 8),'           '); {write address & 11 spaces}
    for column := 0 to 7 do
    begin
      port[$70] := line * 8 + column; {write address to RTC}
      write ('$',cb_hex(port[$71]),'  '); {read CMOS RAM and write contents}
    end;
    writeln
  end
end; {procedure read_RAM}

{*****************************************************************************}
begin {program}
  repeat
    clrscr; {clear screen}
    writeln ('RTC-RAM information');
    read_RAM;
    write ('change RAM (choose N to end)? [Y,N] ');
    ch := upcase (readkey); {read keyboard and convert input character
                             to upper case if necessary}
    write (ch);
    if ch = 'Y'
    then
    begin
      {**** WARNING **** CHECKSUM IS NOT RECALCULATED ****}
      writeln;

      write ('address in hex (e.g $3f) or decimal format? '); readln (addr);
      write ('                                      data? '); readln (data);
      port[$70] := addr;
      port[$71] := data;
      read_RAM;

      write ('press any key to continue...');
      repeat until keypressed; {wait}

      {clear keyboard buffer without using dummy variables}
      repeat if readkey='' then; until not keypressed

    end {if}
  until (ch <> 'Y') and (ch = 'N');
end.
```

900015 - 11

Fig. 1. Listing of the Turbo-Pascal program used to compile RTC-NVR.EXE.



```
RTC-RAM information
hex address    hex data
-----------    ----------------------------------------
$00            $44  $03  $42  $00  $08  $00  $03  $10
$08            $11  $89  $26  $02  $50  $80  $00  $00
$10            $24  $00  $F0  $00  $71  $80  $02  $00
$18            $00  $29  $00  $00  $00  $00  $00  $00
$20            $00  $00  $00  $00  $00  $00  $00  $00
$28            $00  $00  $00  $00  $00  $00  $02  $30
$30            $00  $00  $19  $F9  $EC  $F4  $B0  $1E
$38            $E7  $45  $BA  $1C  $C4  $1F  $4F  $71
change RAM (choose N to end)? [Y,N] Y
address in hex (e.g $3f) or decimal format? $01
                                            data? $00

hex address    hex data
-----------    ----------------------------------------
$00            $10  $00  $43  $00  $08  $00  $03  $10
$08            $11  $89  $26  $02  $50  $80  $00  $00
$10            $24  $00  $F0  $00  $71  $80  $02  $00
$18            $00  $29  $00  $00  $00  $00  $00  $00
$20            $00  $00  $00  $00  $00  $00  $00  $00
$28            $00  $00  $00  $00  $00  $00  $02  $30
$30            $00  $00  $19  $F9  $EC  $F4  $B0  $1E
$38            $E7  $45  $BA  $1C  $C4  $1F  $4F  $71
press any key to continue...                  900015 - 12
```

Fig. 2. Print-out to illustrate the operation of the non-volatile RAM utility. The computer used for this test was a NEAT-286/4 AT compatibie. By studying the dump carefully, you can see when this article was written, and how many seconds it took to modify the data at address $01, the seconds alarm in the RTC.

Byte	Function	Address
0	Seconds	00
1	Second Alarm	01
	Minutes	02
3	Minute alarm	03
4	Hours	04
5	Hour alarm	05
6	Day of week	06
7	Day of month	07
8	Month	08
9	Year	09
10	Status Register A	0A
11	Status Register B	0B
12	Status Register C	0C
13	Status Register D	0D

Table 2. Real-time clock addresses.

Addresses	Description
00–0D	Real-time clock information
	Diagnostic status byte
0F	Shut-down status byte
10	Diskette drive type byte (drives A and B)
11	Reserved
12	Fixed disk drive type (drives C and D)
13	Reserved
14	Equipment byte
15	Low base memory byte
16	High base memory byte
17	Low expansion memory byte
18	High expansion memory byte
19–2D	Reserved
2E–2F	2-byte CMOS checksum
30	Low expansion memory byte
31	High expansion memory byte
32	Date century byte
33	Information flags (set during power-on)
34–3F	Reserved

Table 1. CMOS RAM address map.

'readkey' are procedures resident in Turbo Pascal. Data is read via I/O port 71, while port 70 serves to address the 64 bytes. An example of the output of the program is shown in Fig. 2. The RAM addresses may be entered in hexadecimal (preceded by a $ sign) or in decimal.

The program is also suitable for the popular Amstrad PC1640, and should also work on PC-XTs equipped with a multi-I/O card, although this has not been tested.

Finally, be sure to copy RTC-NVR.EXE to a floppy disk if you have compiled the program on hard disk. Make the floppy bootable by adding CONFIG.SYS, COMMAND.COM and an AUTOEXEC.BAT file that calls up RTC-NVR.EXE.

Byte(s)	Usage	Default
0–9	RTC time and date parameters	--
10	RTC control register A	070H
11	RTC control register B	002H
12	RTC control register C	--
13	RTC control register D	--
14–19	Time and date when machine was last used	--
20	User RAM checksum	--
21-22	Enter key translation token	01C0DH
23-24	Forward key translation token	02207H
25-26	Joystick fire button 1 translation token	0FFFFH
27-28	Joystick fire button 2 translation token	0FFFFH
29-30	Mouse button 1 translation token	0FFFFH
31-32	Mouse button 2 translation token	0FFFFH
33	Mouse X direction scaling factor	00AH
34	Mouse Y direction scaling factor	00AH
35	Initial video mode and drive count	020H
36	Initial video character attributes	007H
37	Size of RAM disk in 2-Kbyte blocks	000H

Table 3. Amstrad PC1640 non-volatile RAM address map.