BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Volumul 62 (66), Numărul 1, 2016
Secția
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ

SOFTWARE IMPLEMENTATION OF A REAL-TIME CLOCK USING A MICROCONTROLLER FROM ATMEL FAMILY

BY

PETRUȚ DUMA[*]

Technical University “Gheorghe Asachi” of Iași,
Faculty of Electronics, Telecommunications and Information Technology

Received: March 10, 2016
Accepted for publication: March 28, 2016

Abstract. The work describes a modality to implement by software a real-time clock on a system equipped with a microcontroller from ATMEL family. From the microcontroller’s resources, the application uses a 16-bit counter that, when overflowed, triggers interrupt requests. The command program counts the periodical interrupt requests and uses separate variables to count seconds, minutes, hours, days, months and years. The application also implements by software the features that allow use either daylight saving time or standard time, to determine leap years, to count months with different number of days, to count days of the week and so forth. The command program was extended to allow the display of the clock and to enable the user to set the clock.

Keywords: real time clock; ATMEL microcontroller; command program.

1. Introduction

The management of various parameters in time, such as digital or analog inputs/outputs, as well as signal acquisition from the sensors used in applications require a real-time clock that can be implemented by means of software. This paper describes a software real-time clock that provides a complete time format that includes second, minute, hour, day, month and year.

[*] *e-mail*: pduma@etti.tuiasi.ro

The user process manages all the system resources, including the real-time clock, being commanded and controlled by an application system equipped with a microcontroller from ATMEL family. Fig. 1 shows the basic structure of a process working in real-time for monitoring various sensors, command inputs/outputs or any other parameter. The notations have the following meanings: AS_µC – application system equipped with microcontroller; S_RTC – software real-time clock; DI – digital inputs; DO – digital outputs; AI – analog inputs; AO – analog outputs; ADC – analog to digital converter; DAC – digital to analog converter; DDC – data display console; SFM – serial flash memory; RS232_SI – RS232 serial interface; PC – personal computer.

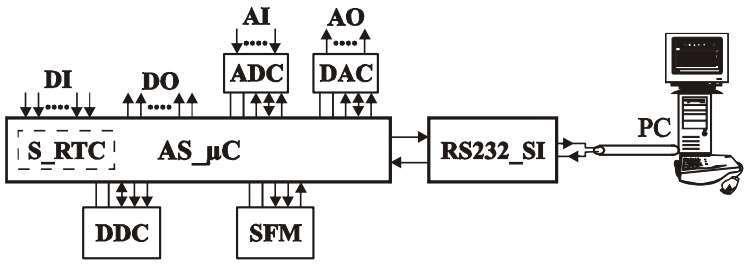


Fig. 1

The serial FLASH memory stores the parameters acquired at certain preset time intervals, their extreme and accidental values along with the real-time clock value. The data display console is optional and permanently displays the real-time clock and the values of the data acquired. The system communicates through the serial asynchronous interface with a personal computer in order to transmit and receive various commands and data, to display the parameters acquired in various formats and for downloading the FLASH memory into the computer.

This work describes the software implementation of the real-time clock while subsequent works will present the sensors monitoring, various digital and analog signal acquisition, data storage into the FLASH memory and data transfer to the personal computer.

2. Software Real-time Clock

The software application uses, from the ATMEL microcontroller resources, the T0 counter and the interrupt system. The T0 counter is set to work in timer operation mode and in 0 operation mode.

The hardware structure of the T0 counter used in this application is shown in fig. 2. and includes the following notations: Osc – microcontroller's internal clock oscillator; XTAL – quartz crystal; FD12 – frequency divider by 12, used for the clock signal; S_O – operation switch; $C/\overline{T}0$ – counter or timer

operation indicator; T0I – T0 timer input (external); S_C – control switch; TR0 – timer run control indicator; $\overline{\text{INT0}}$ – external interrupt request input (hardware counter enable); GATE0 – timer gating control indicator; T0 – counter, consisting of TL0 (low counter section) and TH0 (high counter section); TF0 – timer overflow flag.

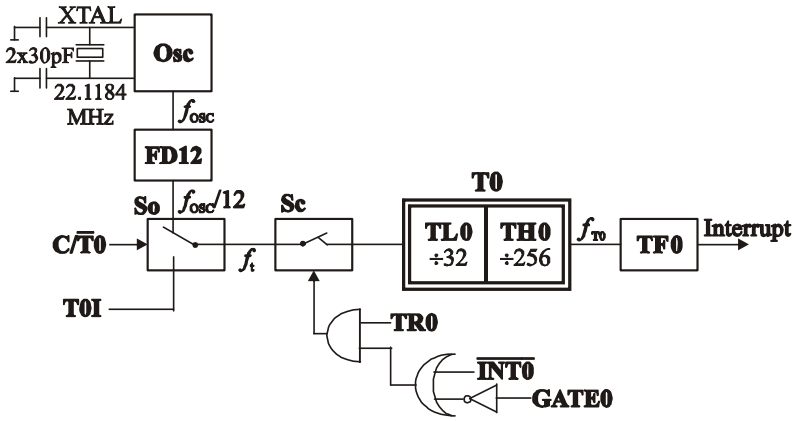


Fig. 2.

The T0 counter in operation mode 0 is basically a 13-bit counter, consisting of an 8-bit counter (TH0) that performs a division by 256, preceded by a 5-bit pre-divisor (TL0) that performs a division by 32; thus, T0 performs in all a division by $2^5 \times 2^8 = 8{,}192$. The counter works if the control switch is closed. In the application described, the counter enabling is made by setting indicator TR0 and resetting indicator GATE0; therefore, the input of the enabled counter is connected to the operation switch. The timer operation mode is used with the C/$\overline{\text{T}}$0 indicator reset when counting the falling edges of the clock signal divided by 12.

Thus, the clock oscillator signal divided by 12 passes through the operation and control switches and reaches the T0 counter that performs a division by 2^{13}. The counter overflow is signaled by setting TF0 indicator.

The application counts the overflows of the interrupt-operating T0 counter. In this case, the hardware structure of the interrupt system that is used is depicted in Fig. 3. The notations used have the following meanings: $\overline{\text{INT0}}$ – external interrupt; TF0 – timer T0 overflow flag; $\overline{\text{INT1}}$ – external interrupt; TF1 – timer T1 overflow flag; TI/RI/SPIF – serial interface flags; TF2/EXF2 – timer T2 flags; EA – global enable/disable indicator; ET0 – timer T0 interrupt enable indicator; S_{PL} – switch for setting the interrupt priority level; PT0 – timer T0 interrupt priority indicator; HILMB – high interrupt level management block; LILMB – low interrupt level management block.

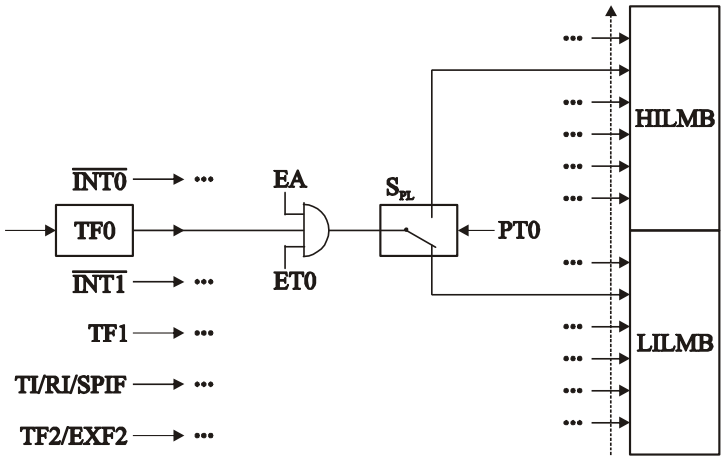


Fig. 3

The interrupt enabling for counter T0 is made by software, setting the interrupt system general enable indicator (EA) and setting the interrupt validation indicator for counter T0 (ET0).

If the interrupt system is active and an interrupt request is sent from counter T0, then the microprocessor finishes to execute the current instruction and forcedly executes a subroutine call instruction; the interrupt request is accepted after sampling and analyzing the interrupt indicators using an internal hardware structure.

The execution of the subroutine call instruction is made with the purpose to save into the stack the address of the following instruction that must be executed within the interrupted program and to load the program counter with the jump address 000BH for counter T0. An internal indicator signalizes the start of the interrupt processing program sequence.

Interrupt from counter T0 is selected for this high level application using the switch that defines the adequate priority level by setting PT0 indicator. While dealing with a high level interrupt, no other interrupt request is accepted.

The subroutine that processes the interrupt request ends with the instruction that resumes the execution of the interrupted program (RETI). This is a subroutine return instruction that actually resumes the execution by loading the program counter with the address previously saved in the stack, but also the update of the internal interrupt indicator, that authorizes new interrupt requests.

The frequency of the internal clock oscillator allows for the following factorization:

$$f_{OSC} = 22.1184\,\text{MHz} = 2^{15} \times 3^3 \times 5^2\,\text{Hz}. \quad (1)$$

The frequency of the operation switch output signal in timer mode is:

$$f_t = \frac{f_{OSC}}{12} = 1.8432 \text{ MHz} = 2^{13} \times 3^2 \times 5^2 \text{ Hz}. \quad (2)$$

The frequency of the signal at the counter T0 output in operation mode 0, that is also the frequency of the periodical interrupt requests, is:

$$f_{T0} = f_{INT} = \frac{f_t}{2^{13}} = \frac{f_{OSC}}{2^{15} \times 3}, \quad (3)$$

$$f_{T0} = f_{INT} = 3^2 \times 5^2 \text{ Hz} = 225 \text{Hz}. \quad (4)$$

The period of the periodical interrupt requests is:

$$T_{INT} = \frac{1}{f_{INT}} = 4.(4) \text{ ms} \quad (5)$$

The real time clock software operates in periodic interrupts set by counter T0, with the duration of 4.(4) ms and relies on the basic principle of counting the interrupt requests.

By counting 225 interrupts with period T_{INT} is produced a duration of one second, that is the basic period for the clock.

The one second duration is obtained by software, using a counter $C1$ that counts down from 225 to 1

$$C1 = 225 \times T_{INT}. \quad (6)$$

A counter for seconds $C2$ is used to obtain one minute duration, counting up from 0 to 59

$$C2 = 60 \times C1. \quad (7)$$

A minute counter $C3$ is used to obtain one hour duration, counting up from 0 to 59

$$C3 = 60 \times C2. \quad (8)$$

An hour counter $C4$ is used to obtain one day duration, counting up from 0 to 23

$$C4 = 24 \times C3. \quad (9)$$

After incrementing the hour counter, it must be checked if the conditions are met for passing to the daylight saving time, or the opposite, to standard time.

The daylight saving time is legally adopted in some countries during a part of the year, starting from a specific date in spring up until another specific date in the autumn. The daylight saving time is earlier by one hour compared to standard time. This system is designed to use as much as possible the natural daylight for human activities, in order to save the energy that would otherwise be spent for lightning.

In Romania, the daylight saving time starts on the last Sunday of March, as 03:00 AM becomes 04:00 AM. The return to standard time is made in the last Sunday of October, as the reverse process takes place, 04:00 AM

becoming 03:00 AM.

A day of week counter $C5$ is used to obtain one week duration, counting up from 1 to 7

$$C5 = 7 \times C4 \tag{10}$$

The numeric values for the days are obviously sequential, numbered from 1 to 7, according to the days of the week (Monday, Tuesday … Sunday) as shown in Table 1.

Table 1

Day of the week	Numeric value
Monday	1
Tuesday	2
Wednesday	3
Thursday	4
Friday	5
Saturday	6
Sunday	7

Using a day of month counter $C6$, one month duration is obtained, by counting up from 1 to N, where N is number of days of the current month, as follows: 28 for February in a standard year; 29 for February in a leap year; 30 for April, June, September and November; 31 for the other months

$$C6 = N \times C4;\ N = 28 / 29 / 30 / 31. \tag{11}$$

The numeric values for the months of the year and the number of days in each month (N) are included in Table 2.

Table 2

Month	Numeric value	Number of days (N)
January	1	31
February – standard year	2	28
February – leap year	2	29
March	3	31
April	4	30
May	5	31
June	6	30
July	7	31
August	8	31
September	9	30
October	10	31
November	11	30
December	12	31

The program memory of the microcontroller stores a data table starting from the symbolic address TD that includes the number of days for each month (*N*) for a standard year.

TD: 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31.

The real time clock from this application is implemented for Gregorian calendar that has an average length of the year of 365.2425 day and tends to get close to the astronomic year (365 days, 5 hours, 48 minutes and 45 seconds).

According to the Gregorian calendar, there are following rules: the years divisible by 400 are leap years; the other years that are divisible by 100 are standard years; of the rest years, those divisible by 4 are leap years.

Using a month counter *C*7, one year duration is obtained, by counting up from 1 to 12

$$C7 = 12 \times C6 . \tag{12}$$

A year counter *C*8 counts up from 00 to 99 years in order to obtain a duration of a century. Counter *C*8 is, in fact, the low part of the year counter and is a byte-long variable

$$C8 = 100 \times C7 . \tag{13}$$

Counter *C*9, which includes the high part of the year, is also a one-byte variable and is initialized to 20. For a clock covering a duration longer as 100 years, it would be required to increment this counter.

$$C9 = 20 . \tag{14}$$

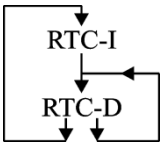
The counters described above (*C*1, *C*2, … *C*9) are presented in a compact manner in table 3, including the numeric range of their possible values (RV), the values used to test the counters (TV) in order to perform a transport, the values used for the counter initialization (IV) and the decrement/ increment manner of each counter (Dec/Inc).

Table 3

Counter	Name	RV	TV	IV	Dec/Inc
Interrupt counter	*C*1	255 … 1	0	255	-1
Second counter	*C*2	0 … 59	60	00	+1
Minute counter	*C*3	0 … 59	60	00	+1
Hour counter	*C*4	0 … 23	24	00	+1
Day of week counter	*C*5	1 … 7	8	1	+1
Day of month counter	*C*6	1 … 28/29/30/31	29/30/31/32	1	+1
Month counter	*C*7	1 … 12	13	1	+1
Year counter (low)	*C*8	00 … 99	100	00	+1
Year counter (high)	*C*9	20	-	20	+1

The basic variable flowchart of the program sequence that initializes the real-time clock is presented below.

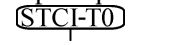
SP←60H	Initialize the stack pointer.
*C*1←225	Initialize interrupt counter with 225.
*C*2←0	Initialize second counter with 00.
*C*3←30	Initialize minute counter with 30.
*C*4←11	Initialize hour counter with 11.
*C*5←1	Initialize day of week counter with 1.
*C*6←15	Initialize day of month counter with 15.
*C*7←2	Initialize month counter with 2.
*C*8←16	Initialize the low part of the year counter with 16.
*C*9←20	Initialize the high part of the year counter with 20.
VI←0	Initialize standard (winter) time indicator with 0.
T0←0	Initialize counter T0 (high and low parts) with 0.
M1M0←0	Initialize counter T0 operation mode 0.
C/T0←0	Initialize counter T0 operation mode timer.
GATE0←0	Initialize software enable indicator for counter T0.
TR0←1	Initialize general enable indicator for counter T0.
EA←1	Initialize general interrupt enable indicator.
ET0←1	Initialize counter T0 interrupt enable indicator.
PT0←1	Initialize high priority level indicator for T0 interrupt.
OI	Other initializations.
PCP	Peripheral circuits programming.
BP	Basic program – user process.



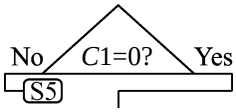
Real-time clock initialization.

Real-time clock display.

The basic variable flowchart of the subroutine that processes the interrupt request from counter T0 is outlined below.



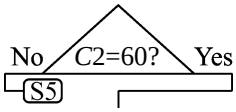
(SP)←Reg.	Save registers into the stack (A, PSW etc.).
*C*1←*C*1-1	Decrement interrupt counter *C*1.



Test if the interrupt counter is down to 0 (225 interrupts received).

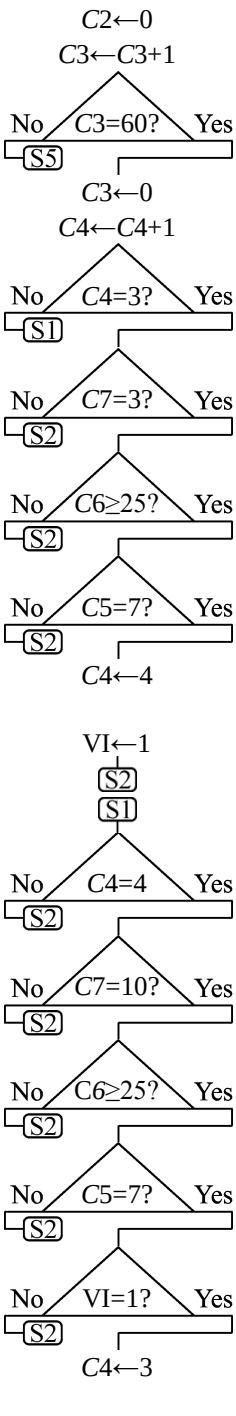
*C*1=0; a second has elapsed.

*C*1←225	Reinitialize interrupt counter with 225.
*C*2←*C*2+1	A second is counted into *C*2.



Test if the second counter has reached 60 (60 seconds elapsed).

*C*2=60; a minute has elapsed.



Reinitialize second counter with 0.

A minute is counted into $C3$.

Test if the minute counter has reached 60 (60 minutes elapsed).

$C3=60$; an hour has elapsed.

Reinitialize minute counter with 0.

An hour is counted into $C4$.

Test for daylight saving time.

Test if the hour counter has reached 3.

$C4=3$; the hours is 03:00 AM.

Test if the month counter is 3.

$C7=3$; the month is March.

Test if the day of month counter is higher than or equal to 25

$C6\geq25$; the day is one of the last seven of the month.

Test if the day of week counter is 7.

$C5=7$; the day is Sunday.

The day is the last Sunday of March and the time becomes 04:00AM from 03:00 AM.

The daylight saving time indicator is set.

Jump to S2 to continue STCI-T0.

Test for the end of daylight saving time and return to standard time.

Test if the hour counter is 4.

$C4=4$; the hour is 04:00AM.

Test if the month counter is 10.

$C7=10$; the month is October.

Test if the day of month counter is higher than or equal to 25

$C6\geq25$; the day is one of the last seven of the month.

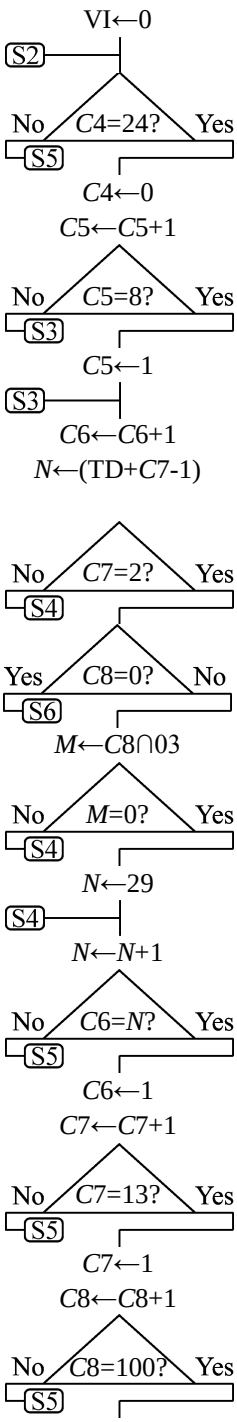
Test if the day of week counter is 7.

$C5=7$; the day is Sunday.

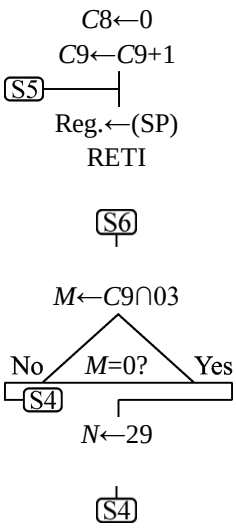
Test if the clock is still set to daylight saving time.

$VI=1$; the clock is still set to daylight saving time.

The day is the last Sunday of October and the time becomes 03:00AM from 04:00 AM.



The daylight saving time indicator is reset.
Continue STCI-T0
Test if the hour counter has reached 24 (24 hours elapsed).
*C*4=24; a day has elapsed.
Reinitialize hour counter with 0.
A day of week is counted in *C*5.
Test if the day of week counter has reached 8 (7 days elapsed).
*C*5=8; a week has elapsed.
Reinitialize day of week counter with 1.
A day of month is counted in *C*6.
The variable *N* is assigned the value retrieved from the memory location with address TD+*C*7-1 representing the number of days of the current month in a standard year.
Test if the month counter is 2.
*C*7=2; the month is February.
Test if the low part of the year counter is 00 (dividable by 100).
*C*8≠00; year not dividable by 100.
The variable *M* is assigned the two less significant bits of the low part of the year counter.
Test if variable *M* is 0.
M=0; the year is a leap year (dividable by 4).
The variable *N* is assigned the value 29, the number of days in February, in leap years.
Update variable *N* to obtain the testing value.
Test if the day of month counter is equal to *N* (if a time period of 28/29/30/31 day is elapsed).
C6=*N*; a month's time is elapsed.
Reinitialize day counter with 1.
A month is counted in *C*7.
Test if the month counter is equal to 13 (if a time period of 12 months is elapsed).
*C*7=13; a year's time is elapsed.
Reinitialize month counter with 1.
A year is counted in *C*8 (the low part of the year counter).
Test if the low part of the year counter is equal to 100 (if a time period of 100 years is elapsed).
*C*8=100; a century's time is elapsed.



*C*8←0 Reinitialize the low part of the year counter with 0.

*C*9←*C*9+1 A century is counted in *C*9 (the high part of the year counter)

S5

Reg.←(SP) Restore the registers from the stack.

RETI Return from the subroutine for processing interrupt request, to resume the interrupted program.

S6 *C*8=0 (the low part of the year counter is 00); is an year dividable by 100.

M←*C*9∩03 The variable *M* is assigned the two less significant bits of the high part of the year counter.

No *M*=0? Yes Test if variable *M* is 0.

S4 *M*=0; the year is a leap year (dividable by 400).

N←29 The variable *N* is assigned the value 29, the number of days in February, in leap years.

S4 Jump to S4 to continue STCI-T0.

3. Conclusions

The real-time clock presented is software implemented for application/development systems equipped with microcontroller AT89S8253, but it can be used for any microcontroller from ATMEL family.

A minimal program version contains the clock initialization sequence and the subroutine for processing interrupt requests from counter T0. In this particular case, the user process makes use of the counter variables of the clock (second, minute, hour, day, month and year) in order to save in a nonvolatile memory the values of the process's inputs and outputs at certain time intervals, and also the extreme and accidental values that appear. The command program uses 0.3 KB of memory space.

Other program versions include minimal functions, along with the clock display and the user's enable to set the clock. Two software versions were written. One of the versions displays and initializes the clock using a serial console from the personal computer and the command program uses 1.9 KB of memory. The other version displays the clock on a time-multiplexed display consisting of six display cells, each of them with 7 segments. The clock setting is made through four programming switches. The command program for this version uses 1.4 KB of memory. The application implements all the functions described and is remarkable by the small amount of memory required compared to its features.

A disadvantage of this implementation is that the microcontroller must be powered from a DC source and also from batteries. This dual powering is necessary in order to maintain the clock information by keeping at least the microcontroller powered during power breaks.

REFERENCES

Duma P., *Microcontrolerul INTEL 8051. Aplicații*, Edit. „TEHNOPRESS", Iaşi, 2004.
Hintz J.K., Tabak D., *Microcontrollers. Arhitecture, Implementation and Programming*, McGraw Hill, New York, 1993.
Lance A., *Leventhal, Programmation en langage assembleur*, Edit. Radio, Paris, 1998.
Peatmann B.J., *Design with Microcontrollers*, McGraw Hill, New York, 1998.
* * * ATMEL, Family Microcontroller, Data Book, 1998.
* * * MAXIM, Multichannel RS232 Drivers/Receivers, MAX232A Data Sheet, 2006.

IMPLEMENTARE CEAS DE TIMP REAL SOFTWARE CU MICROCONTROLER DIN FAMILIA ATMEL

(Rezumat)

Lucrarea descrie modul de implementare prin soft a unui ceas de timp real pe un sistem echipat cu microcontroler din familia ATMEL. Din resursele microcontrolerului este utilizat un numărător de 16 biți care la depăşirea capacității de numărare solicită cereri de întrerupere. Programul de comandă numără cererile de întrerupere periodice şi contorizează în variabile distincte secundele, minutele, orele, zilele, lunile şi anii. Sunt asigurate prin soft facilități care permit trecerea la ora de vară şi de iarnă, determinarea anilor bisecți, contorizarea lunilor cu număr de zile diferite, contorizarea zilelor din săptămână, etc. Programul de comandă este extins pentru a afişa ceasul şi pentru a fi inițializat de utilizator.