Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №4

«Программирование часов реального времени»

Вариант 14

Выполнил: Проверил:

Студент группы 150504 Преподаватель

Желубовский С.В. Одинец Д.Н.

Минск, 2023

1. Постановка задачи

Написать программу, которая будет считывать и устанавливать время в часах реального времени. Считанное время должно выводиться на экран в удобочитаемой форме.

1. Используя аппаратное прерывание часов реального времени и режим генерации периодических прерываний реализовать функцию задержки с точностью в миллисекунды.
2. Используя аппаратное прерывания часов реального времени и режим будильника реализовать функции программируемого будильника.
3. Алгоритм

#### Перед установкой значений времени вызывается функция, которая считывает и анализирует старший байт регистра состояния 1 на предмет доступности значений для чтения и записи. Когда этот бит установлен в '0', отключается внутренний цикл обновления часов реального времени: для этого старший бит регистра состояния 2 устанавливается в '1'.

#### Считывание или запись значений времени происходит следующим образом: в порт 70h отправляется индекс регистра CMOS, соответствующий значению времени (секунды, часы и т. д.), затем происходит чтение значения из порта 71h (или запись значения в порт).

#### После установки значений времени вызывается функция, которая возобновляет внутренний цикл обновления часов реального времени.

#### Для реализации функции задержки заменён обработчик прерывания 0x70, в котором происходит отсчёт миллисекунд. Для включения периодического прерывания, происходящего примерно каждую миллисекунду, 6-й бит регистра B устанавливается в '1'.

1. Листинг программы

Далее приведен листинг программы, реализующей все поставленные задачи.

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <dos.h>

#define TIMER\_REG\_AMOUNT 3

#define TIME\_REG\_AMOUNT 6

typedef unsigned char byte;

enum time\_registers {

sec = 0x00,

sec\_timer = 0x01,

min = 0x02,

min\_timer = 0x03,

hour = 0x04,

hour\_timer = 0x05,

day = 0x07,

month = 0x08,

year = 0x09,

};

enum state\_registers {

// 7 - is time update

// 4-6 - divider

// 0-3 interrupt frequency

A = 0x0A,

// 7 - refresh clock control

// 6 - irq8 control

// 5 - timer interrupt allowed

// 4 - interrupt after cycle allowed

// 3 - meander generation allowed

// 2 - time format: 1 is binary, 0 is BSC

// 1 - time format: 1 is 24h, 0 is 12h

// 0 - summer time allowed

B = 0x0B,

// 7 - was interruption

// 6 - period interruption is allowed

// 5 - timer interruption

// 4 - interrupt after clock

// 0-3 - must be zero

C = 0x0C,

};

unsigned time[TIME\_REG\_AMOUNT];

const byte time\_registers[TIME\_REG\_AMOUNT] = {sec, min, hour, day, month, year};

const byte timer\_registers[TIMER\_REG\_AMOUNT] = {sec\_timer, min\_timer, hour\_timer};

unsigned bcd\_to\_dec(unsigned bcd) { return (bcd / 16 \* 10) + (bcd % 16); }

unsigned dec\_to\_bcd(unsigned dec) { return (dec / 10 \* 16) + (dec % 10); }

void print\_time(void) {

int i;

for (i = 0; i < TIME\_REG\_AMOUNT; ++i) {

outp(0x70, A);

// Don't read time while 7 bit is set

if (inp(0x71) & 0x80) {

i--;

continue;

}

outp(0x70, time\_registers[i]);

time[i] = inp(0x71);

time[i] = bcd\_to\_dec(time[i]);

}

printf("%02u:%02u:%02u %02u.%02u.20%02u\n",

time[2], time[1], time[0], time[3], time[4], time[5]);

}

void set\_time(void) {

int i;

puts("Enter time in format hours:min:sec");

if (scanf("%u:%u:%u", &time[2], &time[1], &time[0]) != 3) {

fprintf(stderr, "Failed to enter time\n");

}

puts("Enter date in format day.month.year");

if (scanf("%u.%u.%u", &time[3], &time[4], &time[5]) != 3) {

fprintf(stderr, "Failed to enter date\n");

}

for (i = 0; i < TIME\_REG\_AMOUNT; ++i) {

time[i] = dec\_to\_bcd(time[i]);

}

disable();

// Don't read time while 7 bit is set

do {

outp(0x70, 0x0A);

} while (inp(0x71) & 0x80);

// Off clock refresh

outp(0x70, 0x0B);

outp(0x71, inp(0x71) | 0x80);

for (i = 0; i < TIME\_REG\_AMOUNT; i++) {

outp(0x70, time\_registers[i]);

outp(0x71, time[i]);

}

// On clock refresh

outp(0x70, 0x0B);

outp(0x71, inp(0x71) & 0x7F);

enable();

}

void interrupt(\*old\_interrupt)(void);

void interrupt new\_interrupt(void) {

puts("New alarm handler called");

outp(0x70, 0x0C);

if (inp(0x71) & 0x20) {

puts("5 bit in C register is set");

}

// Send EOI to master and slave

outp(0x20, 0x20);

outp(0xA0, 0x20);

// Set old interrupt

disable();

setvect(0x70, old\_interrupt);

enable();

// Stop alarm

outp(0x70, 0x0B);

outp(0x71, inp(0x71) & 0xDF);

}

void set\_alarm(void) {

int i;

puts("Enter time in format hours:min:sec");

if (scanf("%u:%u:%u", &time[2], &time[1], &time[0]) != 3) {

fprintf(stderr, "Failed to enter time\n");

}

for (i = 0; i < TIMER\_REG\_AMOUNT; ++i) {

time[i] = dec\_to\_bcd(time[i]);

}

disable();

// Get old RTC interruption handler

old\_interrupt = getvect(0x70);

// Set new RTC interruption handler

setvect(0x70, new\_interrupt);

outp(0xA1, (inp(0xA0) & 0xFE));

// Don't read time while 7 bit is set

do {

outp(0x70, 0x0A);

} while (inp(0x71) & 0x80);

for (i = 0; i < TIMER\_REG\_AMOUNT; i++) {

outp(0x70, timer\_registers[i]);

outp(0x71, time[i]);

}

enable();

// Allow alarm

outp(0x70, 0x0B);

outp(0x71, inp(0x71) | 0x20);

puts("Timer set");

}

int main(void) {

while (1) {

puts("1 - print time");

puts("2 - set time");

puts("3 - set alarm");

puts("0 - exit");

switch (getch()) {

case '1':

print\_time();

break;

case '2':

set\_time();

break;

case '3':

set\_alarm();

break;

case '0':

exit(EXIT\_SUCCESS);

default:

break;

}

}

}

1. Тестирование программы

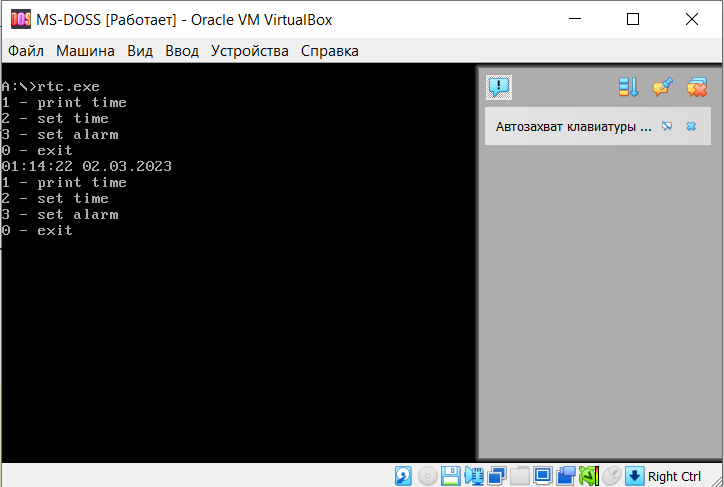


Рисунок 4.1 — Меню пользователя с выводом текущего времени.

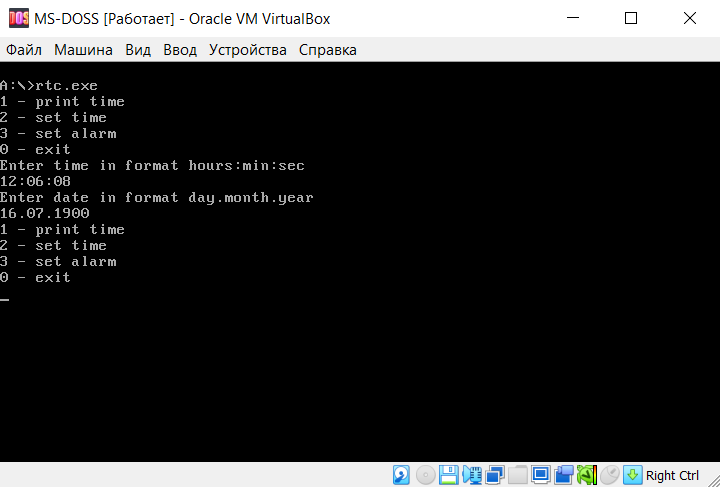


Рисунок 4.2 — Установка нового времени

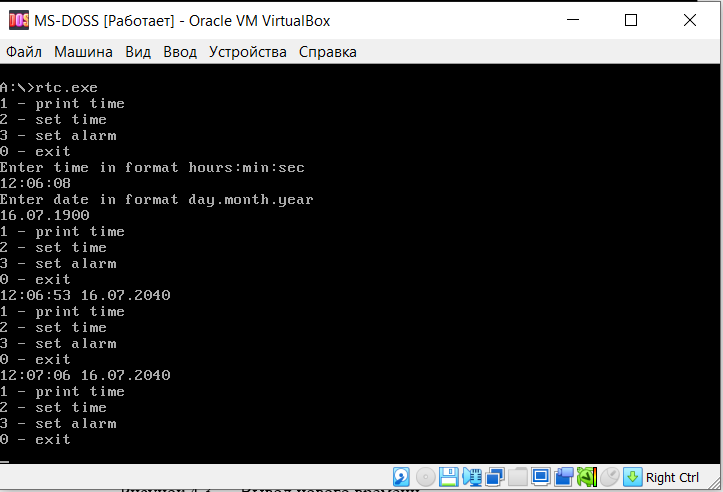


Рисунок 4.3 — Вывод нового времени.

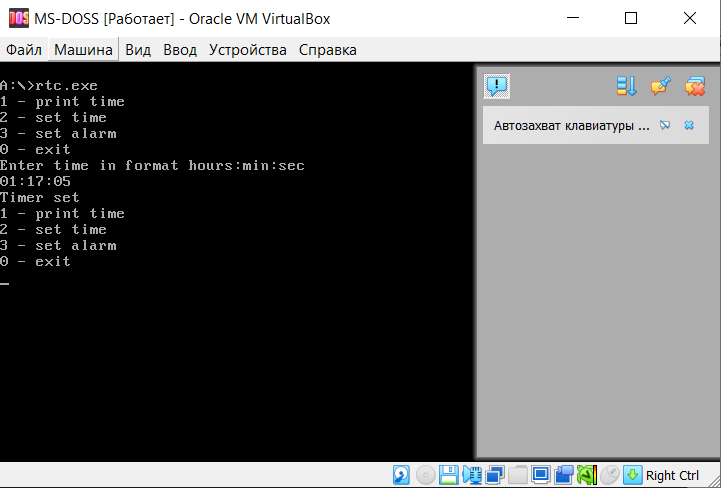


Рисунок 4.4 — Установка будильника.

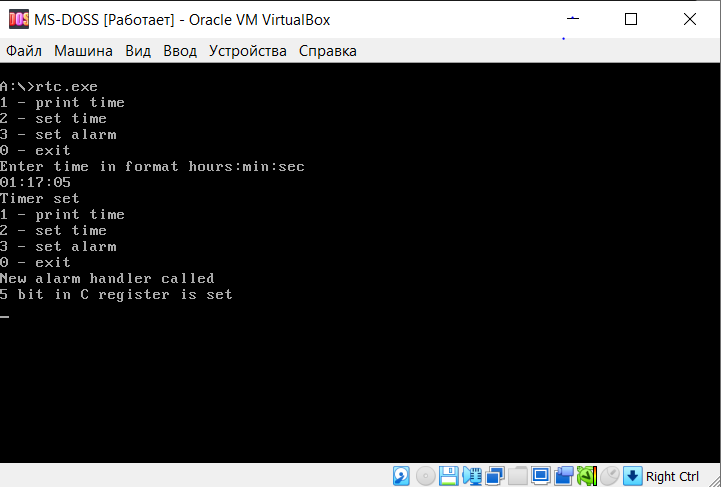


Рисунок 4.5 — Срабатывание будильника.

1. Заключение

В данной лабораторной работе были выполнены все поставленные задачи: написана программа, которая считывает и устанавливает время в часах реального времени, была реализована функция программируемого будильника, используя аппаратное прерывания часов реального времени и режим будильника.

Программа компилировалась в Turbo C++ и запускалась в DOS, который эмулировался с помощью VirtualBox.