Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ЭВМ

Отчет по лабораторной работе №4

«Программирование часов реального времени»

Проверил: Выполнил:

к.т.н., доцент студент гр. 150501

Одинец Д.Н. Почебут А.С.

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.

1.1 Задержка должна вводится с клавиатуры в миллисекундах в удобной для

пользователя форме.

2. Используя аппаратное прерывание часов реального времени и режим будильника реализовать функции программируемого будильника.

2.1. Время будильника вводится с клавиатуры в удобной для пользователя форме.

2.2. При срабатывании будильника программа должна сообщить об этом в любой форме.

1. 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, выполняющей все поставленные задачи.

#include <io.h>

#include <dos.h>

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

char new\_time[6]; // данные часов

unsigned int delay\_count = 0;

unsigned int time\_counters[] = { 0x00, 0x02, 0x04, 0x07, 0x08, 0x09 };

unsigned int time\_registers[] = { 0x01, 0x03, 0x05};

int alarm\_on = 0;

void interrupt time\_interrupt(...);

void interrupt alarm\_interrupt(...);

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

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

int dtob(int val);

int btod(int val);

void show\_time();

void set\_time();

void set\_alarm();

void set\_delay(unsigned int delay);

void reset\_alarm();

void enter\_time();

int enter\_value(int high\_measure, int low\_measure);

int dtob(int val)

{

return (val / 10 \* 16) + (val % 10);

}

int btod(int val)

{

return (val / 16 \* 10) + (val % 16);

}

void show\_time()

{

int i = 0;

for (i = 0; i < 6; i++) {

outp(0x70, time\_counters[i]); // выбор адреса в памяти CMOS

new\_time[i] = btod(inp(0x71)); // считывание значения по адресу в массив

}

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

new\_time[2], new\_time[1], new\_time[0], new\_time[3], new\_time[4], new\_time[5]);

}

int enter\_value(int high\_measure, int low\_measure)

{

int value = 0;

do

{

rewind(stdin);

scanf("%d", &value);

} while(value > high\_measure || value < low\_measure);

return dtob(value);

}

void enter\_time()

{

printf("Enter year:\n");

new\_time[5] = enter\_value(100, 21);

printf("Enter month:\n");

new\_time[4] = enter\_value(12, 1);

printf("Enter day:\n");

new\_time[3] = enter\_value(31, 1);

printf("Enter hours:\n");

new\_time[2] = enter\_value(23, 0);

printf("Enter minuts:\n");

new\_time[1] = enter\_value(59, 0);

printf("Enter seconds:\n");

new\_time[0] = enter\_value(59, 0);

}

void set\_time()

{

enter\_time();

disable();

// проверка на доступность значений для чтения/записи

unsigned int check;

do {

outp(0x70, 0xA); // выбор регистра А

check = inp(0x71) & 0x80; // 0x80 - 1000 0000

// 7-й бит в 1 для обновления времени

} while (check);

// отключение обновления часов реального времени

outp(0x70, 0xB); // выбор регистра B

outp(0x71, inp(0x71) | 0x80); // 0x80 - 1000 0000

// 7-й бит в 1 для запрета обновления часов

for (int i = 0; i < 6; i++) {

outp(0x70, time\_counters[i]); // выбор нужного значения данных

outp(0x71, new\_time[i]); // подача в регистр нужного значения

}

// включение обновления часов реального времени

outp(0x70, 0xB); // выбор регистра В

outp(0x71, inp(0x71) & 0x7F); // 0x7F - 0111 1111

// 7-й бит в 0 для разрешения обновления часов

enable(); // разрешение на прерывание

system("cls");

}

void set\_delay(unsigned int delay)

{

int freq;

printf("input frequence:\n");

do

{

scanf("%d", &freq);

}while(freq > 15 || freq < 3);

disable(); // запрет на прерывание

// установка нового обработчика прерываний

old\_time\_interrupt = getvect(0x70);

setvect(0x70, time\_interrupt);

// размаскировка линии сигнала запроса от ЧРВ

// 0xA1 - новое значение счетчика для системного таймера

outp(0xA1, inp(0xA1) & 0xFE); // 0xFE = 1111 1110

// 0-й бит в 0 для разрешения прерывания от ЧРВ

enable();

int state;

disable();

unsigned int check;

do

{

outp(0x70, 0xA); // выбор регистра A

check = inp(0x71) & 0x80;

}while(check);

outp(0x70, 0xA);

outp(0x71, (inp(0x71) & 0xF0) | freq);

// разрешение на прерывание

enable();

outp(0x70, 0xB); // выбор регистра В

outp(0x71, inp(0x71) | 0x40); // 0x40 = 0100 0000

// 6-й бит регистра B установлен в 1 для периодического прерывания

delay\_count = 0;

while (delay\_count <= delay);

disable();

setvect(0x70, old\_time\_interrupt);

enable();

delay\_count = 0;

return;

}

void set\_alarm()

{

enter\_time(); // ввод нового времени

disable(); // запрет на прерывание

// проверка на доступность значений для чтения/записи

unsigned int check;

do {

outp(0x70, 0xA); // выбор регистра A

check = inp(0x71) & 0x80; // 0x80 - 1000 0000

// 7-й бит в 1 для обновления времени

} while (check);

for(int i = 0; i < 3; i++)

{

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

outp(0x71, new\_time[i]);

}

outp(0x70, 0xB); // выбор регистра B

outp(0x71, (inp(0x71) | 0x20)); // 0x20 - 0010 0000

// 5-й бит регистра B установлен в 1 для разрешения прерывания будильника

// переопределение прерывания будильника

old\_alarm\_interrupt = getvect(0x4A); // 0x4A - обновление времени

setvect(0x4A, alarm\_interrupt);

outp(0xA1, (inp(0xA0) & 0xFE)); // 0xFE - 1111 1110

// 0-й бит в 0 для разрешения прерывания от ЧРВ

enable(); // разрешение на прерывание

alarm\_on = 1;

}

void reset\_alarm()

{

// проверка на наличие установленного будильника

if (old\_alarm\_interrupt == NULL)

return;

disable(); // запрет на прерывание

// возврат старого прерывания

setvect(0x4A, old\_alarm\_interrupt);

outp(0xA1, (inp(0xA0) | 0x01)); // 0x01 - 0000 0001 (пересчет частоты прерывания)

// проверка на доступность значений для чтения/записи

unsigned int check;

do {

outp(0x70, 0xA); // выбор регистра A

check = inp(0x71) & 0x80; // 0x80 - 1000 0000

// 7-й бит в 1 для обновления времени

} while (check);

// запись нулевых значений в регистр будильника

for (int i = 0; i < 3; i++)

{

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

outp(0x71, 0x00);

}

outp(0x70, 0xB); // выбор регистра B

outp(0x71, (inp(0x71) & 0xDF)); // 0xDF - 1101 1111

// 5-й бит в 0 для запрета прерывания будильника

enable(); // разрешение на прерывание

}

int main()

{

while (1)

{

system("cls");

show\_time();

printf("\n1 - Set time");

printf("\n2 - Set delay");

printf("\n3 - Set alarm");

printf("\n0 - Exit");

printf("\ndelay\_count = %d", delay\_count);

if(alarm\_on == 1) printf("\n\nALARM ON");

if(alarm\_on == 2) {

printf("\n\nALARM!");

delay(1000);

alarm\_on = 0;

}

printf("\n\nEnter choice: ");

delay(1000);

if (kbhit()) {

switch(getch())

{

case '0':

return 0;

default:

break;

case '1':

system("cls");

set\_time();

break;

case '2':

system("cls");

int delay = 0;

printf("Input delay (ms): ");

scanf("%d", &delay);

set\_delay(delay);

break;

case '3':

system("cls");

set\_alarm();

break;

}

}

}

}

void interrupt time\_interrupt(...) // новый обработчик прерываний часов

{

outp(0x70, 0x0C);

if (inp(0x71) & 0x40) // 0100 0000

{

delay\_count++;

printf("delay\_count: %d\n", delay\_count);

}

// посыл сигнала контроллерам прерываний об окончании прерывания

(\*old\_time\_interrupt)();

}

void interrupt alarm\_interrupt(...) // новый обработчик прерываний будильника

{

system("cls");

alarm\_on = 2;

old\_alarm\_interrupt();

reset\_alarm();

}

1. Тестирование программ

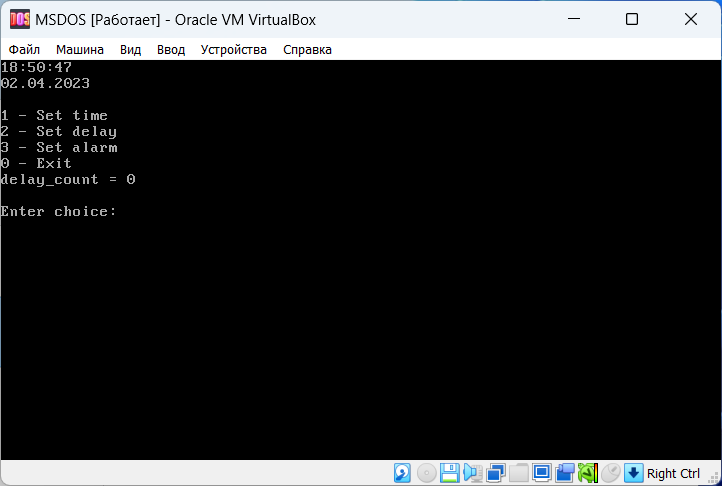


Рисунок 4.1 – Результат работы программы при запуске.

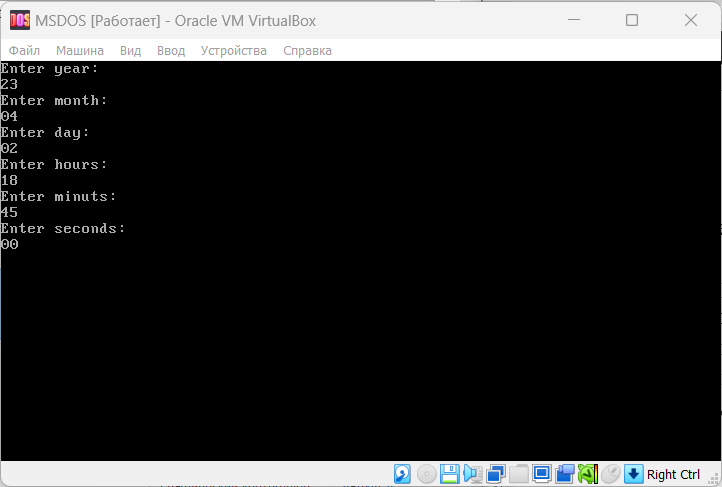


Рисунок 4.2 – Установка нового времени.

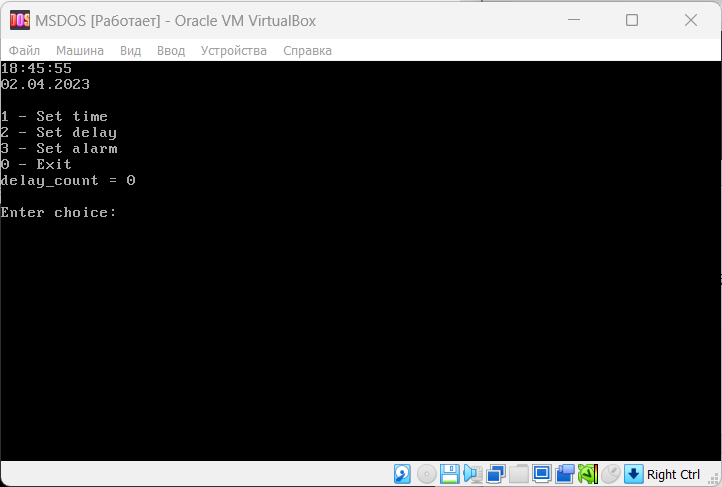


Рисунок 4.3 — Вывод нового времени.

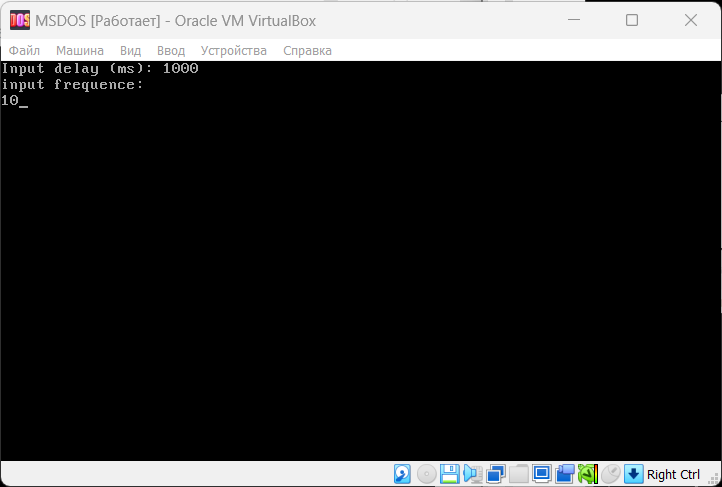


Рисунок 4.4 — Установка задержки.

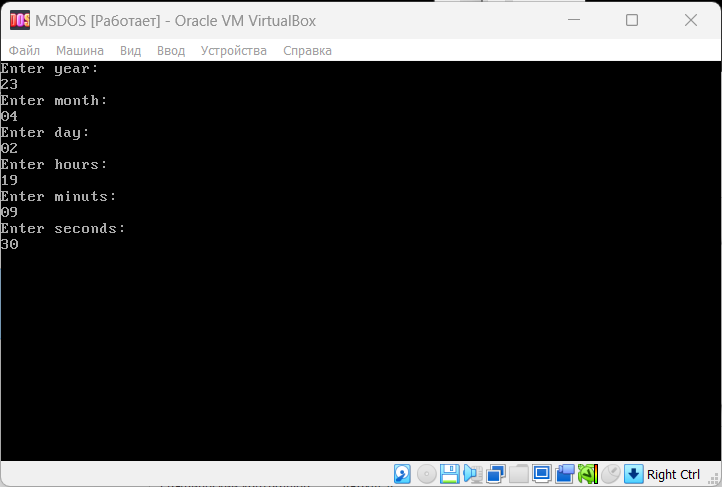


Рисунок 4.5 — Установка будильника.

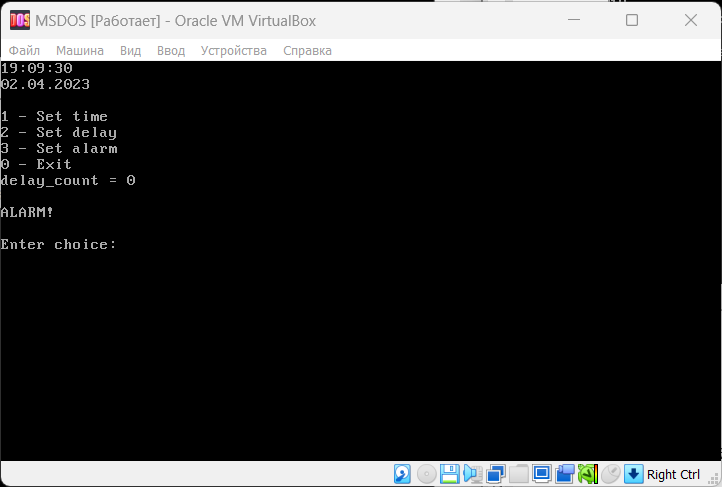


Рисунок 4.6 — Срабатывание будильника.

1. Заключение

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

Программа компилировалась в Borland C и запускалась в DOS, который эмулировался с помощью MS-DOS в VirtualBox.