Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №4

“Программирование часов реального времени”

Проверил: Выполнил:

к.т.н., доцент студент гр.150502

Одинец Дмитрий Николаевич Альхимович Н. Г.

Минск 2023

**Задача**

В программе должно быть реализовано меню, позволяющее выбрать тестируемый функционал (установка времени, считывание времени, задержка и т.д.).

1. Написать программу, которая будет считывать и устанавливать время в часах реального времени. Считанное время должно выводиться на экран в удобочитаемой форме.
2. Используя аппаратное прерывание часов реального времени и режим генерации периодических прерываний, реализовать функцию задержки с точностью в миллисекунды (с возможностью изменения частоты генерации).
3. Используя аппаратное прерывания часов реального времени и режим будильника ЧРВ (4A использовать не желательно), реализовать функции программируемого будильника.

**Алгоритм**

Программа состоит из нескольких частей, представляющих собой некоторые функции, с помощью которых осуществляется следующее:

1. для считывания времени надо выполнить следующие действия:
   * + проверить доступность регистров;

* в порт 70h отправить номер требуемого байта памяти;
* считать значение времени из порта 71h.

1. для установки времени надо выполнить следующие действия:
   * + проверить доступность регистров;

* запретить обновление часов;
* отключить цикл обновления часов;
* с помощью портов 70h и 71h установить нужные значения времени;
* включить цикл обновления часов.

1. для установки задержки надо выполнить следующие действия:
   * + сохранить указатель на старый обработчик прерывания ЧРВ;

* установить новый обработчик прерываний ЧРВ;
* обеспечить вызов периодического прерывания;
* восстановить старый обработчик.

1. для установки будильника надо выполнить следующие действия:
   * + сохранить указатель на старый обработчик прерывания ЧРВ;

* установить новый обработчик прерываний ЧРВ;
* разрешить прерывания на линии ЧРВ;
* обеспечить вызов периодического прерывания.

**Листинг программы**

#include <stdio.h>

#include <string.h>

#include <dos.h>

void getCurrentTime(void); //функция считывания времени в ЧРВ

void setCurrentTime(void); //функция установки времени в ЧРВ

void setDelay(void); //функция задержки

void setFrequency(void); //функция изменения частоты генерации периодических прерываний

void setAlarm(void); //функция установки будильника

void wait(void); //функция ожидания конца цикла корректировки

int BCDToDecimal(int); //функция перевода числа из двоично-десятичного формата в десятичный

int DecimalToBCD(int); //функция перевода числа из двоично-десятичного формата в десятичный

volatile unsigned long delay\_counter;

void interrupt(\*systemRTCDelay)(void); //указатель на прерывание ЧРВ

void interrupt customRTCDelay(void) //прототип функции обработки прерывания

{

if(delay\_counter > 0)

delay\_counter--;

(\*systemRTCDelay)();

}

volatile int hours\_alarm, mins\_alarm, secs\_alarm, trigger = 0;

void interrupt(\*systemRTCAlarm)(void);

void interrupt customRTCAlarm(void)

{

int temph, tempm, temps;

wait();

outp(0x70, 0x04);

temph = BCDToDecimal(inp(0x71));

wait();

outp(0x70, 0x02);

tempm = BCDToDecimal(inp(0x71));

wait();

outp(0x70, 0x00);

temps = BCDToDecimal(inp(0x71));

if(tempm == mins\_alarm && temph == hours\_alarm && temps == secs\_alarm)

trigger = 1;

(\*systemRTCAlarm)();

}

//0x70 - порт индекса ЧРВ, 0x71 - порт данных ЧРВ

int main()

{

while(1)

{

puts("\nChoose an option: 1-get current time;");

puts(" 2-set current time;");

puts(" 3-set delay;");

puts(" 4-set alarm;");

puts(" 0-end");

fflush(stdin);

switch(getchar())

{

case '1':

getCurrentTime(); //считывание времени в ЧРВ

break;

case '2':

setCurrentTime(); //установка времени в ЧРВ

break;

case '3':

setDelay(); //задержка

break;

case '4':

setAlarm(); //программируемый будильник

break;

case '0':

return 0;

default:

fprintf(stderr, "Wrong parameter\n");

return -1;

}

}

return 0;

}

void getCurrentTime()

{

printf("Current time: ");

wait();

outp(0x70, 0x04); //в порт 70h посылается номер требуемого байта памяти, текущий час

printf("%d:", BCDToDecimal(inp(0x71))); //чтение байта памяти через порт 71h

wait();

outp(0x70, 0x02); //текущая минута

printf("%d:", BCDToDecimal(inp(0x71)));

wait();

outp(0x70, 0x00); //текущая секунда

printf("%d\n", BCDToDecimal(inp(0x71)));

}

void setCurrentTime()

{

int hours, mins, secs;

printf("Enter exact time you want to set: ");

scanf("%d:%d:%d", &hours, &mins, &secs); //извлечение значений часов, минут и секунд

\_disable(); //запретить прерывания (= cli)

wait();

outp(0x70, 0x0B); //0Bh - регистр состояния B (бит 7 - запрещено обновление часов)

outp(0x71, inp(0x71) | (1 << 6)); //отключить цикл обновления ЧРВ, установив 7-й бит в 1

outp(0x70, 0x04);

outp(0x71, DecimalToBCD(hours));

outp(0x70, 0x02);

outp(0x71, DecimalToBCD(mins));

outp(0x70, 0x00);

outp(0x71, DecimalToBCD(secs));

outp(0x70, 0x0B);

outp(0x71, inp(0x71) & ~(1 << 6)); //включить цикл обновления ЧРВ, сбросив 7-й бит (~ - побитовое инвертирование)

\_enable(); //разрешить прерывания (= sti)

}

void setDelay()

{

printf("Enter the length of delay in milliseconds: ");

scanf("%u", &delay\_counter);

\_disable(); //запретить прерывания (= cli)

systemRTCDelay = getvect(0x70); //сохранить указатель на старый обработчик прерываний ЧРВ

setvect(0x70, customRTCDelay); //установить новый обработчик прерываний ЧРВ

\_enable(); //разрешить прерывания (= sti)

printf("Delay started\n");

//setFrequency();

//0xA1 - регистр маски для второго контроллера прерываний PIC

outp(0xA1, inp(0xA1) & ~1); //~1 = 1111 111 0 - разрешить прерывания на линии ЧРВ (IRQ8)

outp(0x70, 0x0B); //регистр состояния B

outp(0x71, inp(0x71) | (1 << 6)); //6 бит - в 1: вызов периодического прерывания (IRQ8)

while(delay\_counter > 0);

printf("Delay is over\n");

setvect(0x70, systemRTCDelay); //восстановить старый обработчик прерываний

}

void setFrequency()

{

unsigned char freq;

printf("Enter the frequency (in Hz) you want to set for RTC: ");

scanf("%hhu", &freq);

freq = (unsigned char)(32768 / freq);

outp(0x70, 0x0A);

outp(0x71, (inp(0x71) & 0xF0) | freq); //сохранить старшие 4 бита регистра А (0xF0 = 11110000), с помощью | меняются 4 младший бита

}

void setAlarm()

{

printf("Enter the alarm time: ");

scanf("%d:%d:%d", &hours\_alarm, &mins\_alarm, &secs\_alarm);

\_disable();

systemRTCAlarm = getvect(0x70); //сохранить указатель на старый обработчик прерываний ЧРВ

setvect(0x70, customRTCAlarm); //установить новый обработчик прерываний ЧРВ

\_enable();

outp(0x70, 0x0B); //регистр состояния B

outp(0x71, inp(0x71) | (1 << 5)); //5 бит - в 0: вызов периодического прерывания (IRQ8)

while(!trigger);

printf("ALARM!\n");

setvect(0x70, systemRTCAlarm); //восстановить старый обработчик прерываний

}

void wait()

{

do

{

outp(0x70, 0x0A); //0Аh - регистр состояния A (бит 7 - заняты/не заняты часы

} while(inp(0x71) & (1 << 6)); //проверка значения 7-го бита: пока он не сброшен, происходит корректировка частка КМОП-памяти (первые десять байтов)

}

int BCDToDecimal(int bcd)

{

int decimal = 0, factor = 1;

while(bcd != 0)

{

decimal += (bcd & 0x0F) \* factor; //bcd & 0x0F - младшие четыре бита

bcd >>= 4; //обеспечение обработки следующих четырех битов

factor \*= 10; //учет каждой десятичной позиции

}

return decimal;

}

int DecimalToBCD(int decimal)

{

int bcd = 0, multiplier = 1;

while(decimal > 0)

{

bcd += (decimal % 10) \* multiplier; //обработка каждой цифры десятичного числа

multiplier \*= 16; //обеспечение перехода к следующему разряду кода

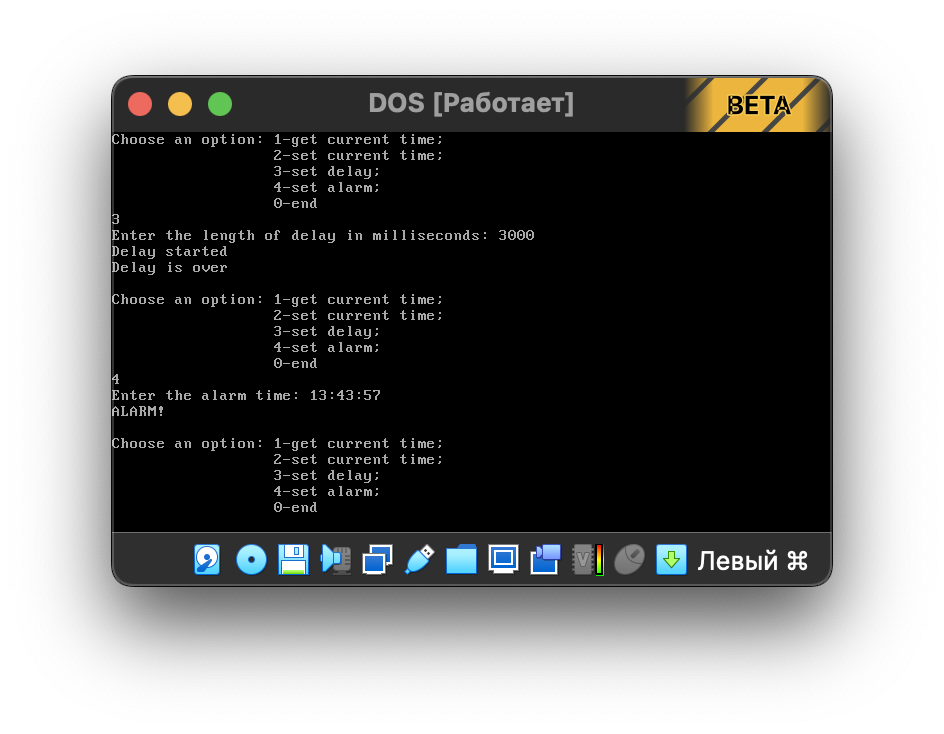
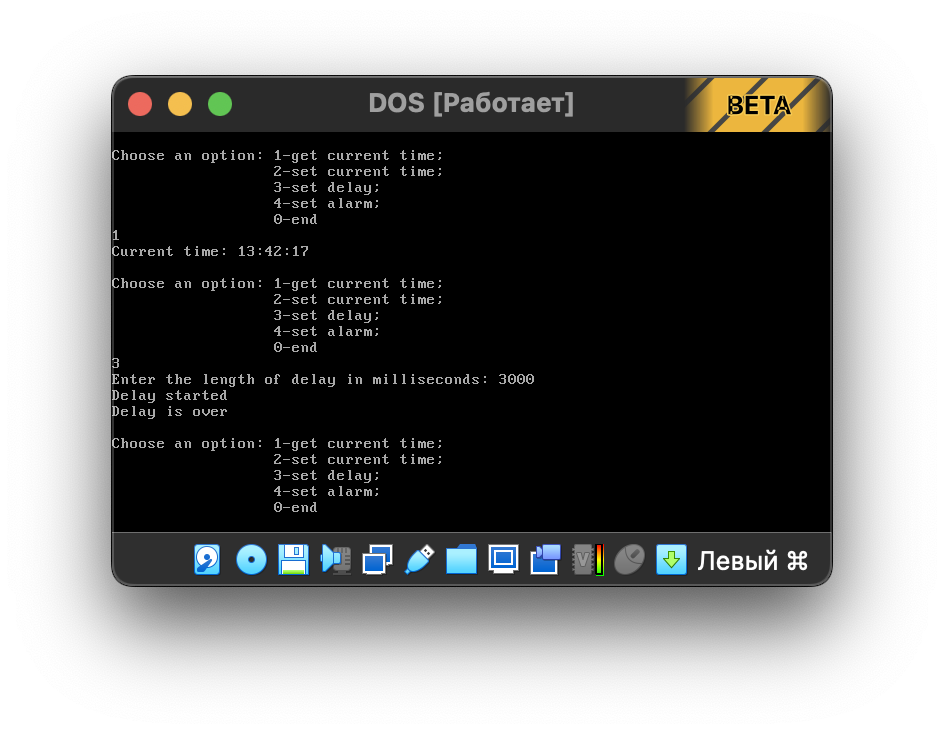
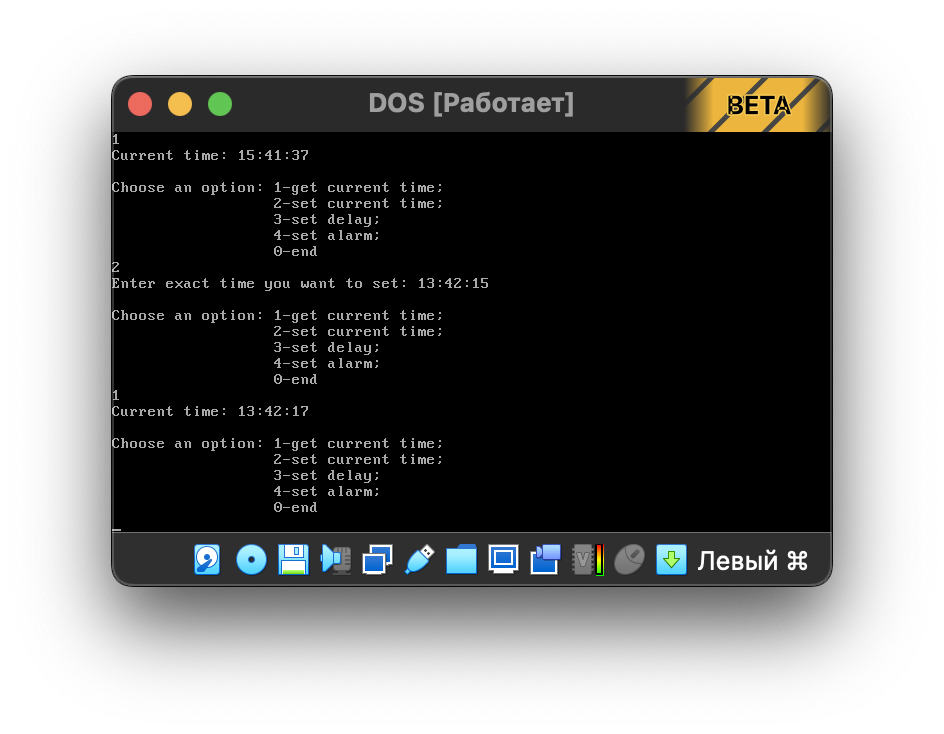
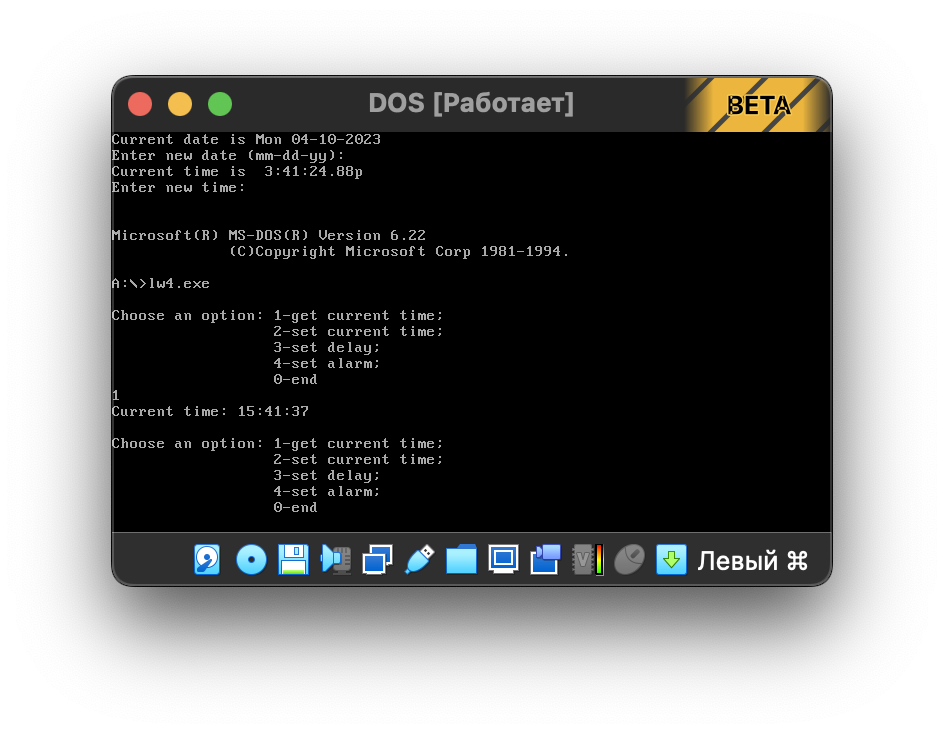
decimal /= 10; //обеспечение перехода к следующей цифре

}

return bcd;

}

**Тест**

****

**Заключение**

В данной лабораторной работе разработана программа, которая позволяет считывать и устанавливать время посредством аппаратного прерывания часов реального времени, а также реализует функцию задержки и будильника.

Для компиляции программы использовался TurboC++, а для запуска исполняемого файла – MS-DOS, эмулированная с помощью VirtualBox.