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.150501

Одинец Дмитрий Николаевич Черноок А.Ю.

Минск 2023

**Задача**

Целью данной работы является написание программы, позволяющей работать с часами реального времени.

**Алгоритм**

Реализованная программа состоит из нескольких подпрограмм, выполняющих следующие функции:

* Вывод текущего времени;
* Изменение времени;
* Будильник;
* Задержка;
* Изменение частоты периодического прерывания.

Для выбора регистра часов реального времени используется порт 0х70, для работы с самими регистрами используется порт 0х71. Регистры ЧРВ с 0х00 по 0х09 хранят дату и время, далее следуют регистры состояний. В данной лабораторной работе используются регистры состояний – A, B, C. В регистре А выполняется проверка на доступность ЧРВ, а также регулируется частота периодического прерывания. Регистр В позволяет запретить/разрешить обновление часов, периодическое прерывание, прерывание будильника. Из регистра С можно получить информацию о происходящем прерывании.

**Листинг программы**

#include <dos.h>

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <windows.h>

typedef struct \_Time\_

{

short hour;

short min;

short sec;

} Time;

void interrupt\_init();

void show\_time();

void set\_time();

void make\_delay();

void set\_alarm();

void set\_frequency();

void print\_VideobuffString(char\* str, int offset);

void get\_state();

int delayTime = 0;

void interrupt(\*prevClockHandler)(...); // RCT Handler

void interrupt newClockHandler(...);

int main(){

interrupt\_init();

char ch;

while (1) {

clrscr();

printf("Choose option:\n'1' - Show time\n'2' - Set new time\n'3' - Make Delay\n'4' - Set Alarm");

printf("\n'5' - Change Frequency\n\n'8' - Vide0Buff Test\n'9' - Registers State\n'0' - End\n");

ch = getch();

switch (ch)

{

case '1':

show\_time();

break;

case '2':

set\_time();

break;

case '3':

make\_delay();

break;

case '4':

set\_alarm();

break;

case '5':

set\_frequency();

break;

case '8':

print\_VideobuffString("AAALLAAAAAAAARM!!!", 40);

getch();

break;

case '9':

get\_state();

break;

default:

break;

}

if (ch == '0')

break;

}

setvect(0x70, prevClockHandler);

return 0;

}

short get\_demicalNum(short \_BCD\_num) {

short newFirstNum = \_BCD\_num / 16;

short newSecNum = \_BCD\_num % 16;

return newFirstNum \* 10 + newSecNum;

}

void show\_time() {

Time clock;

while (1) {

clrscr();

//printf("Current time: \n");

outp(0x70, 0x04); //choose hour

clock.hour = inp(0x71);

clock.hour = get\_demicalNum(clock.hour);

outp(0x70, 0x02); //choose minute

clock.min = inp(0x71);

clock.min = get\_demicalNum(clock.min);

outp(0x70, 0x00); //choose minute

clock.sec = inp(0x71);

clock.sec = get\_demicalNum(clock.sec);

printf("Current time: \n%d:%d:%d\n", clock.hour, clock.min, clock.sec);

if (kbhit()) {

getch();

break;

}

delay(900);

}

}

short get\_BCDNum(short \_demic\_num) {

short newFirstNum = \_demic\_num / 10;

short newSecNum = \_demic\_num % 10;

return newFirstNum \* 16 + newSecNum;

}

void set\_time() {

Time time\_to\_set;

// Get new time

printf("New hour: ");

scanf("%d", &time\_to\_set.hour);

time\_to\_set.hour = get\_BCDNum(time\_to\_set.hour);

printf("New min: ");

scanf("%d", &time\_to\_set.min);

time\_to\_set.min = get\_BCDNum(time\_to\_set.min);

// Check on availability of clock registers

//unsigned short condition;

outp(0x70, 0x0A); // choose condition register A

char condition = inp(0x71);

condition &= 0x80; // 0x80 - 10000000b

if (condition) {

printf("It is impossible to set new time\n");

return;

}

else printf("It is OK to set new time\n");

// Disable RTC update cycle

outp(0x70, 0x0B); // choose condition register B

condition = inp(0x71);

condition |= 0x80; // 0x80 - 10000000b

outp(0x71, condition);

// Set new time

outp(0x70, 0x04); //choose hour

outp(0x71, time\_to\_set.hour);

outp(0x70, 0x02); //choose minute

outp(0x71, time\_to\_set.min);

// Allow RCT update cycle

outp(0x70, 0xB); // choose condition register B

condition = inp(0x71);

condition &= 0x7F; // 0x7F - 01111111b

outp(0x71, condition);

//printf("New chosen time: %d:%d", time\_to\_set.hour, time\_to\_set.min);

getch();

}

void print\_VideobuffString(char\* str, int offset)

{

char attrib = 0x5e;

char far\* videobuf = (char far\*)MK\_FP(0xB800, 0);

videobuf += (offset \* 2);

char temp;

for (int i = 0; str[i]; i++)

{

\*videobuf = str[i];

videobuf++;

\*videobuf = attrib;

videobuf++;

}

}

//!!!!!!!!!!!!!!!!!

void interrupt newClockHandler(...)

{

delayTime--;

char cond1;

char cond2;

outp(0x70, 0x0C); // choose condition register C

cond1 = cond2 = inp(0x71);

cond1 &= 0x80; // 1000 0000b

cond2 &= 0x20; // 0010 0000b

//if (cond1 && cond2) {

if (cond2) {

//printf("ALAAAAAAAAAAAAAAARM!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");

//getch();

print\_VideobuffString("AAALLAAAAAAAARM!!!", 40);

}

prevClockHandler();

}

//!!!!!!!!!!!!!!!!!!!!!!

void interrupt\_init() {

disable();

outp(0xA1, inp(0xA1) & 0xFE); // 1111 1110b - allow clocks interrupt

prevClockHandler = getvect(0x70); // real time clock handle

setvect(0x70, newClockHandler);

enable();

}

void my\_delay(unsigned int milSecs);

void make\_delay() {

// Get delay

printf("Enter delay in ms: ");

unsigned int milSeconds;

scanf("%d", &milSeconds);

printf("Start of Delay");

my\_delay(milSeconds);

printf("\nEnd of Delay");

getch();

}

void my\_delay(unsigned int milSeconds) {

delayTime = milSeconds;

// Permission on periodic interruption

outp(0x70, 0x0B); // choose register В

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

while (delayTime >=0);

outp(0x70, 0x0B); // choose register В

outp(0x71, inp(0x71) & 0xBF); // 0x40 = 1011 1111b // choose register В

return;

}

void set\_alarm() {

Time alarm\_time;

// Get new time

printf("Alarm hour: ");

scanf("%d", &alarm\_time.hour);

alarm\_time.hour = get\_BCDNum(alarm\_time.hour);

printf("Alarm min: ");

scanf("%d", &alarm\_time.min);

alarm\_time.min = get\_BCDNum(alarm\_time.min);

printf("Alarm sec: ");

scanf("%d", &alarm\_time.sec);

alarm\_time.sec = get\_BCDNum(alarm\_time.sec);

//// Check on availability of clock registers

////unsigned short condition;

//outp(0x70, 0xA); // choose condition register A

//char condition = inp(0x71);

//condition &= 0x80; // 0x80 - 10000000b

//if (condition) {

// printf("It is impossible to set alarm time\n");

// return;

//}

//else printf("It is OK to set alarm time\n");

// Set new time

outp(0x70, 0x05); //choose hour

outp(0x71, alarm\_time.hour);

outp(0x70, 0x03); //choose minute

outp(0x71, alarm\_time.min);

outp(0x70, 0x01); //choose second

outp(0x71, alarm\_time.min);

// Rermission on Alarm interruption

outp(0x70, 0x0B);

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

//

//

//

//outp(0x70, 0x0B);

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

}

void set\_frequency() {

printf("\nChoose frequency:");

printf("\n1 - 1024 Hz \n2 - 2 Hz \n3 - 8196 Hz");

outp(0x70, 0x0A); // choose condition register A

char condition = inp(0x71);

char chFreq = getch();

switch (chFreq)

{

case '1': // 3-0 0110

condition |= 0x06; // 0000 0110b

condition &= 0xF6; // 1111 0110b

break;

case '2': // 3-0 1111

condition |= 0x0F; // 0000 1111b

break;

case '3': // 3-0 0011

condition |= 0x03; // 0000 0011b

condition &= 0xF3; // 1111 0011b

break;

}

outp(0x71, condition);

}

void get\_state() {

printf("\nStates:\n");

unsigned short state;

outp(0x70, 0x0A); // choose condition register A

state = inp(0x71);

printf("Reg A:%d\n", state);

outp(0x70, 0x0B); // choose condition register B

state = inp(0x71);

printf("Reg B:%d\n", state);

outp(0x70, 0x0C); // choose condition register C

state = inp(0x71);

printf("Reg C:%d\n", state);

//outp(0x70, 0x0C); // choose condition register C

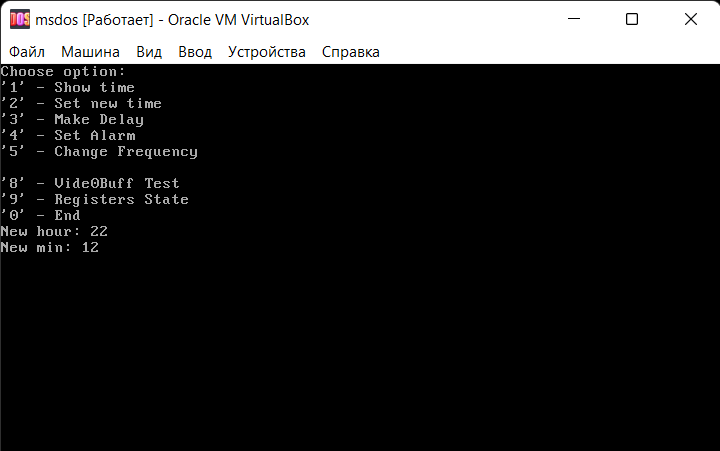
state = inp(0xA1);

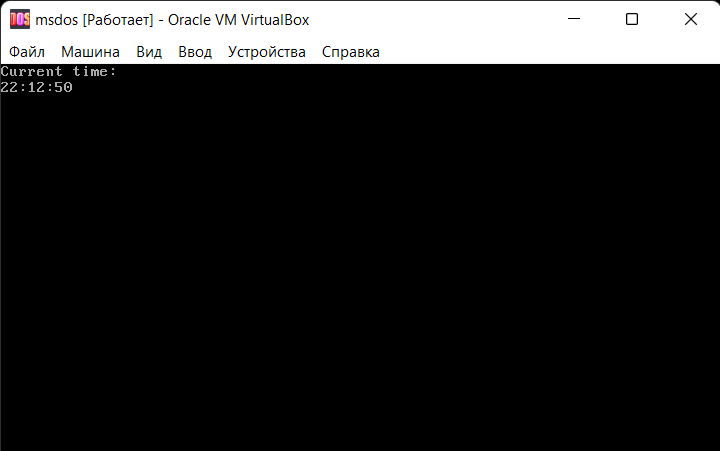
printf("Reg A1:%d\n", state);

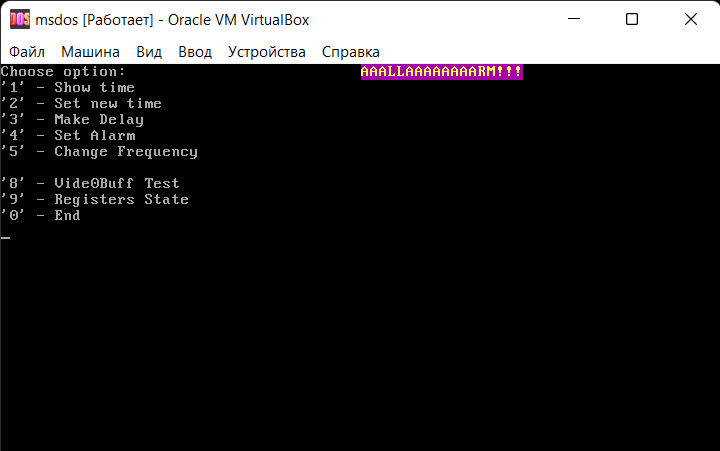
getch();

}

**Тест**







**Заключение**

В данной лабораторной работе разработана программа, которая способна выводить на экран текущее время, изменять его, выставлять будильник, создавать задержку, изменять частоту периодического прерывания.