Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра электронных вычислительных машин

Лабораторная работа №4

«Программирование часов реального времени»

Выполнил: Проверил:

Студент группы 250502 Преподаватель

Грибовская А. А. Одинец Д.Н.

Минск, 2024

1. Постановка задачи

Написать программу, которая будет считывать и устанавливать время в часах реального времени. Считанное время должно выводиться на экран в удобночитаемой 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 <dos.h> // Подключаем заголовочный файл для работы с функциями DOS

#include <stdio.h> // Подключаем стандартную библиотеку ввода-вывода

#include <stdlib.h> // Подключаем стандартную библиотеку функций общего назначения

#include <conio.h> // Подключаем заголовочный файл для работы с консольным вводом-выводом

#include <io.h> // Подключаем заголовочный файл для работы с функциями ввода-вывода

#include <windows.h> // Подключаем заголовочный файл для работы с функциями Windows API

unsigned int delayTime = 0; // Объявляем переменную для хранения времени задержки

unsigned int delayMs; // Объявляем переменную для хранения задержки в миллисекундах

unsigned int date[6]; // Объявляем массив для хранения времени и даты

unsigned int dateReg[] = { 0x00, 0x02, 0x04, 0x07, 0x08, 0x09 }; // Значения регистров для чтения времени и даты

void interrupt(\*oldDelay)(...); // Объявляем указатель на прерывание для старого обработчика задержки

void interrupt(\*oldAlarm) (...); // Объявляем указатель на прерывание для старого обработчика будильника

void showTime(); // Прототип функции вывода времени

void setTime(); // Прототип функции установки времени

void delay(); // Прототип функции задержки

void setAlarm(); // Прототип функции установки будильника

void inputTime(int); // Прототип функции ввода времени

unsigned char todec(int); // Прототип функции перевода из BCD в десятичное

unsigned char tobcd(int); // Прототип функции перевода из десятичного в BCD

void changeFreqency(); // Прототип функции изменения частоты

unsigned char todec(int val) // Функция перевода из BCD в десятичное

{

return (unsigned char)((val / 16 \* 10) + (val % 16)); // Выполняем перевод и возвращаем результат

}

unsigned char tobcd(int val) // Функция перевода из десятичного в BCD

{

return (unsigned char)((val / 10 \* 16) + (val % 10)); // Выполняем перевод и возвращаем результат

}

void interrupt newDelay(...) // Обработчик прерывания для новой задержки

{

delayTime++; // Увеличиваем счетчик времени задержки

outp(0x70, 0x0C); // Записываем команду в регистр C

inp(0x71); // Считываем значение из регистра C

outp(0x20, 0x20); // Записываем команду в мастер-контроллер прерываний

outp(0xA0, 0x20); // Записываем команду в ведомый контроллер прерываний

if (delayTime == delayMs) // Проверяем, достигнута ли заданная задержка

{

puts("Delay's end"); // Выводим сообщение о завершении задержки

disable(); // Отключаем прерывания

setvect(0x70, oldDelay); // Восстанавливаем старый обработчик прерывания

enable(); // Включаем прерывания

outp(0x70, 0x0B); // Записываем команду в регистр A

outp(0x71, inp(0x71) & 0xBF); // Устанавливаем бит для разрешения прерывания таймера

}

}

void interrupt newAlarm(...) // Обработчик прерывания для нового будильника

{

puts("Alarm"); // Выводим сообщение о срабатывании будильника

outp(0x70, 0x0C); // Записываем команду в регистр A

inp(0x71); // Считываем значение из регистра B

outp(0x20, 0x20); // Записываем команду в мастер-контроллер прерываний

outp(0xA0, 0x20); // Записываем команду в ведомый контроллер прерываний

disable(); // Отключаем прерывания

setvect(0x70, oldAlarm); // Восстанавливаем старый обработчик прерывания

enable(); // Включаем прерывания

outp(0x70, 0x0B); // Записываем команду в регистр B

outp(0x71, inp(0x71) & 0xDF); // Устанавливаем бит для разрешения прерывания будильника (5 бит в 1)

}

int main() // Основная функция программы

{

while (1) { // Бесконечный цикл для вывода меню и обработки ввода

printf("1 - Time\n"); // Выводим пункт меню для отображения времени

printf("2 - Set time\n"); // Выводим пункт меню для установки времени

printf("3 - Set alarm\n"); // Выводим пункт меню для установки будильника

printf("4 - Set delay\n"); // Выводим пункт меню для установки задержки

printf("5 - Set freq\n"); // Выводим пункт меню для установки частоты

printf("e - Exit\n\n"); // Выводим пункт меню для выхода из программы

switch (getch()) { // Получаем ввод пользователя

case '1': // Если введен '1'

showTime(); // Выводим текущее время

break; // Завершаем выполнение switch

case '2': // Если введен '2'

setTime(); // Запускаем процесс установки времени

break; // Завершаем выполнение switch

case '3': // Если введен '3'

setAlarm(); // Запускаем процесс установки будильника

break; // Завершаем выполнение switch

case '4': // Если введен '4'

fflush(stdin); // Очищаем входной поток

printf("Input delay (ms): "); // Запрашиваем задержку в миллисекундах

scanf("%u", &delayMs); // Считываем задержку

delay(); // Задерживаем выполнение программы на указанное время

break; // Завершаем выполнение switch

case '5': // Если введен '5'

changeFreqency(); // Запускаем процесс изменения частоты

break; // Завершаем выполнение switch

case 'e': // Если введен 'e'

printf("\n\n"); // Печатаем пустые строки для отделения от предыдущего вывода

return 0; // Завершаем выполнение программы

default: // В случае ввода любого другого символа

printf("\n\n"); // Печатаем пустые строки для отделения от предыдущего вывода

break; // Завершаем выполнение switch

}

}

}

void showTime() // Функция для вывода текущего времени и даты

{

int i = 0; // Объявляем и инициализируем переменную для использования в цикле

for ( i = 0; i < 3; i++) // Цикл для вывода времени

{

outp(0x70, dateReg[i]); // Устанавливаем номер регистра часов

printf("%02d:", todec(inp(0x71))); // Выводим значение времени из регистра

}

printf("\n"); // Переходим на новую строку

for (i = 3; i < 6; i++) // Цикл для вывода даты

{

outp(0x70, dateReg[i]); // Устанавливаем номер регистра даты

printf("%02d.", todec(inp(0x71))); // Выводим значение даты из регистра

}

}

void setTime() // Функция для установки времени и даты

{

int a =1; // Объявляем и инициализируем переменную, указывающую, что устанавливается время

inputTime(a); // Запускаем функцию ввода времени и даты пользователем

disable(); // Отключаем прерывания

outp(0x70, 0x0A); // Устанавливаем команду для чтения состояния регистра A

do // Запускаем цикл проверки готовности регистра

{

a = inp(0x71) & 0x80; // Считываем 8-й бит регистра A

} while (a == 1); // Цикл повторяется, пока 8-й бит не станет равен 0

outp(0x70, 0x0B); // Устанавливаем команду для записи в регистр состояния 2

outp(0x71, inp(0x71) | 0x80); // Устанавливаем 8-й бит в 1 для разрешения записи

for (int i = 0; i < 6; i++) // Цикл записи времени и даты в регистры

{

outp(0x70, dateReg[i]); // Устанавливаем номер регистра

outp(0x71, date[i]); // Записываем соответствующее значение времени или даты

}

outp(0x70, 0x0B); // Устанавливаем команду для записи в регистр состояния 2

outp(0x71, inp(0x71) & 0x7F); // Сбрасываем 8-й бит в 0 для блокировки записи

enable(); // Включаем прерывания

printf("\nnew time\n"); // Выводим сообщение о успешном изменении времени и даты

showTime(); // Выводим новое время и дату

}

void inputTime(int a)

{

int n;

if(a==1){

do {

fflush(stdin);

printf("Year: ");

} while (scanf("%d", &n) != 1 || n > 99 || n < 0);

date[5] = tobcd(n);

do {

fflush(stdin);

printf("Month: ");

} while (scanf("%d", &n) != 1 || n > 59 || n < 0);

date[4] = tobcd(n);

do {

fflush(stdin);

printf("Day: ");

} while (scanf("%d", &n) != 1 || n > 30 || n < 0);

date[3] = tobcd(n);

}

do {

fflush(stdin);

printf("Hours: ");

} while ((scanf("%d", &n) != 1 || n > 23 || n < 0));

date[2] = tobcd(n);

do {

fflush(stdin);

printf("Minutes: ");

} while (scanf("%d", &n) != 1 || n > 59 || n < 0);

date[1] = tobcd(n);

do {

fflush(stdin);

printf("Seconds: ");

} while (scanf("%d", &n) != 1 || n > 59 || n < 0);

date[0] = tobcd(n);

}

void delay() // Функция для создания задержки в миллисекундах

{

delayTime = 0; // Устанавливаем начальное значение времени задержки

disable(); // Отключаем прерывания

oldDelay = getvect(0x70); // Сохраняем текущий обработчик прерывания

setvect(0x70, newDelay); // Устанавливаем новый обработчик прерывания

outp(0xA1, (inp(0xA0) & 0xFE)); // Разрешаем прерывания

enable(); // Включаем прерывания

outp(0x70, 0x0B); // Выбираем регистр состояния 2

outp(0x71, inp(0x71) | 0x40); // Устанавливаем бит разрешения обновления часовых данных (8 бит в 1)

return; // Возвращаемся

}

void setAlarm() // Функция для установки будильника

{

unsigned int alarmReg[] = { 0x01, 0x03, 0x05 }; // Массив регистров для установки времени будильника

int a = 0; // Переменная для определения типа установки времени (0 - необходимо ввести новое время)

inputTime(a); // Ввод времени для будильника

disable(); // Отключаем прерывания

oldAlarm = getvect(0x70); // Сохраняем текущий обработчик прерывания будильника

setvect(0x70, newAlarm); // Устанавливаем новый обработчик прерывания будильника

outp(0xA1, (inp(0xA0) & 0xFE)); // Разрешаем прерывания будильника

do // Запускаем цикл ожидания, пока часы не будут сброшены

{

outp(0x70, 0x0A); // Выбираем регистр состояния 1

} while (inp(0x71) >> 7); // Повторяем, пока бит сброса часов не станет равен 0

for (int i = 0; i < 3; i++) // Цикл установки времени будильника в регистры

{

outp(0x70, alarmReg[i]); // Выбираем соответствующий регистр

outp(0x71, date[i]); // Записываем значение времени в соответствующий регистр

}

enable(); // Включаем прерывания

outp(0x70, 0x0B); // Выбираем регистр состояния 2

outp(0x71, inp(0x71) | 0x20); // Устанавливаем бит разрешения прерывания будильника

printf("Alarm enable in: \n\n"); // Выводим сообщение о включении будильника

for (int k = 0; k < 3; k++) // Цикл вывода времени будильника

{

printf("%02d:", date[k]); // Печатаем значение времени

}

printf("\n"); // Переходим на новую строку

}

void changeFreqency() // Функция для изменения частоты

{

int freq; // Переменная для хранения значения частоты

int q; // Переменная для выбора частоты пользователем

int bin; // Переменная для хранения битового представления регистра

puts("set freq:"); // Выводим сообщение о выборе частоты

puts("1 - 2 hertz"); // Выводим варианты частоты

puts("2 - 4 hertz");

puts("3 - 8 hertz");

puts("4 - 16 hertz");

puts("5 - 32 hertz");

puts("6 - 64 hertz");

puts("7 - 128 hertz");

puts("8 - 256 hertz");

puts("9 - 512 hertz");

puts("10 - 1024 hertz");

puts("11 - 2048 hertz");

puts("12 - 4096 hertz");

puts("13 - 8192 hertz");

scanf("%d", &q); // Считываем выбор пользователя

freq = 0x0F - q + 1; // Вычисляем значение частоты

outp(0x70, 0x0A); // Выбираем регистр A

bin = inp(0x71); // Считываем текущее значение регистра

printf("before =%X= \n", bin); // Выводим текущее значение регистра

outp(0x70, 0x0A); // Выбираем регистр A

outp(0x71, (inp(0x71) & 0x1F) | 0x80); // Устанавливаем старший бит делителя частоты в 1

outp(0x70, 0x0A); // Выбираем регистр A

outp(0x71, (inp(0x71) & 0xE0) | freq); // Устанавливаем младшие биты делителя частоты

outp(0x70, 0x0A); // Выбираем регистр A

bin = inp(0x71); // Считываем текущее значение регистра

printf("\nAfter =%X= \n", bin); // Выводим значение регистра после изменения

}

1. Тестирование программы

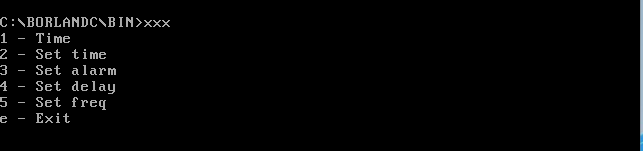


Рисунок 4.1 — Меню пользователя.

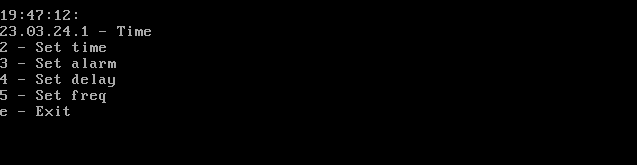


Рисунок 4.2 — Вывод текущего времени

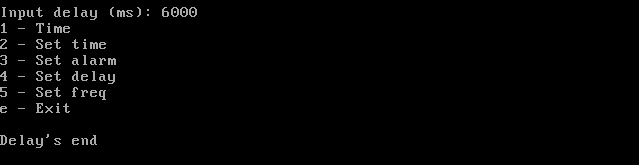


Рисунок 4.3 — Вывод задержки времени.