Белорусский Государственный Университет Информатики и Радиоэлектроники

Кафедра ЭВМ

Отчет по лабораторной работе № 3

Тема: «Программирование часов реального времени»

Выполнил:

студент гр.XXXXXX

XXXXX X.X.

Проверил:

Третьяков А.Г.

Минск 2016

#### Постановка задачи.

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

#### Используя аппаратное прерывание часов реального времени и режим генерации периодических прерываний реализовать функцию задержки с точностью в миллисекунды.

#### Алгоритм решения задачи.

#### Перед установкой значений времени вызывается функция LockClockUpdate(), 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 UnlockClockUpdate(), 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'.

#### Листинг программы.

#### #include <dos.h>

#### #include <ctype.h>

#### #include <stdio.h>

#### #include <conio.h>

#### #include <string.h>

#### #include <stdlib.h>

#### int msCounter = 0;

#### void interrupt far (\*oldInt70h)(void);

#### void interrupt far NewInt70Handler(void);

#### void LockClockUpdate();

#### void UnlockClockUpdate();

#### int BCDToInteger(int bcd);

#### unsigned char IntToBCD(int value);

#### void GetTime();

#### void SetTime();

#### void ShowValue(unsigned char regNum);

#### void CustomDelay();

#### void WaitClockIsFree();

#### void WaitForMSCounter(unsigned long delayPeriod);

#### void AllowClockInterrupts();

#### void EndOfInterruptForMasterInterruptController();

#### void EndOfInterruptForSlaveInterruptController();

#### void EnablePeriodicInterrupt();

#### void main()

#### {

#### char c, value;

#### clrscr();

#### printf("Press:\n'1' - Show time\n'2' - Set time\n'3' - Delay time\n'Esc' - quit\n\n");

#### 

#### while(c != 27)

#### {

#### c = getch();

#### switch(c)

#### {

#### case '1': GetTime();break;

#### case '2': SetTime();break;

#### case '3': CustomDelay();break;

#### case 27: break;

#### }

#### }

#### }

#### void WaitClockIsFree()

#### {

#### do

#### {

#### outp(0x70, 0x0A);

#### } while( inp(0x71) & 0x80 ); // Check for 1 in 7th bit

#### }

#### void GetTime()

#### {

#### unsigned char value;

#### 

#### WaitClockIsFree();

#### outp(0x70, 0x04); // Hours

#### value = inp(0x71);

#### printf("%d:",BCDToInteger(value));

#### 

#### WaitClockIsFree();

#### outp(0x70, 0x02); // Minutes

#### value = inp(0x71);

#### printf("%d:",BCDToInteger(value));

#### 

#### WaitClockIsFree();

#### outp(0x70, 0x00); // Seconds

#### value = inp(0x71);

#### printf("%d ",BCDToInteger(value));

#### 

#### WaitClockIsFree();

#### outp(0x70, 0x07); // Day of month

#### value = inp(0x71);

#### printf("%d.",BCDToInteger(value));

#### 

#### WaitClockIsFree();

#### outp(0x70, 0x08); // Month

#### value = inp(0x71);

#### printf("%d.",BCDToInteger(value));

#### 

#### WaitClockIsFree();

#### outp(0x70, 0x09); // Year

#### value = inp(0x71);

#### printf("%d ",BCDToInteger(value));

#### 

#### WaitClockIsFree();

#### outp(0x70, 0x06); // Day of week

#### value = inp(0x71);

#### 

#### switch(BCDToInteger(value))

#### {

#### case 1: printf("Sunday\n"); break;

#### case 2: printf("Monday\n"); break;

#### case 3: printf("Tuesday\n"); break;

#### case 4: printf("Wednesday\n"); break;

#### case 5: printf("Thursday\n"); break;

#### case 6: printf("Friday\n"); break;

#### case 7: printf("Saturday\n"); break;

#### default: printf("Undefined day of week\n"); break;

#### }

#### }

#### void SetTime()

#### {

#### int hours, minutes, seconds, weekDay, monthDay, month, year;

#### 

#### printf("Enter hours: ");

#### scanf("%d", &hours);

#### printf("Enter minutes: ");

#### scanf("%d", &minutes);

#### printf("Enter seconds: ");

#### scanf("%d", &seconds);

#### printf("Enter week day number: ");

#### scanf("%d", &weekDay);

#### printf("Enter day of month: ");

#### scanf("%d", &monthDay);

#### printf("Enter mounth: ");

#### scanf("%d", &month);

#### printf("Enter year: ");

#### scanf("%d", &year);

#### 

#### LockClockUpdate();

#### 

#### outp(0x70, 0x04);

#### outp(0x71, IntToBCD(hours));

#### 

#### outp(0x70, 0x02);

#### outp(0x71, IntToBCD(minutes));

#### 

#### outp(0x70, 0x00);

#### outp(0x71, IntToBCD(seconds));

#### 

#### outp(0x70, 0x06);

#### outp(0x71, IntToBCD(weekDay));

#### 

#### outp(0x70, 0x07);

#### outp(0x71, IntToBCD(monthDay));

#### 

#### outp(0x70, 0x08);

#### outp(0x71, IntToBCD(month));

#### 

#### outp(0x70, 0x09);

#### outp(0x71, IntToBCD(year));

#### UnlockClockUpdate();

#### }

#### void LockClockUpdate()

#### {

#### unsigned char value;

#### WaitClockIsFree();

#### outp(0x70,0x0B);

#### value = inp(0x71);

#### value|=0x80; // 1 to 7th bit

#### outp(0x70, 0x0B);

#### outp(0x71, value);

#### }

#### void UnlockClockUpdate()

#### {

#### unsigned char value;

#### WaitClockIsFree();

#### outp(0x70,0x0B);

#### value = inp(0x71);

#### value-=0x80; // 0 to 7th bit

#### outp(0x70, 0x0B);

#### outp(0x71, value);

#### }

#### void interrupt far NewInt70Handler(void)

#### {

#### msCounter++;

#### outp(0x70,0x0C); // Neccessary code for interrupt

#### inp(0x71); // controller initiate interrupt again

#### 

#### EndOfInterruptForMasterInterruptController();

#### EndOfInterruptForSlaveInterruptController();

#### }

#### void EndOfInterruptForMasterInterruptController()

#### {

#### outp(0x20,0x20);

#### }

#### void EndOfInterruptForSlaveInterruptController()

#### {

#### outp(0xA0,0x20);

#### }

#### void CustomDelay()

#### {

#### unsigned long delayPeriod;

#### 

#### disable(); // cli

#### oldInt70h = getvect(0x70);

#### setvect(0x70, NewInt70Handler);

#### enable(); // sti

#### printf("Delay in ms: ");

#### scanf("%ld", &delayPeriod);

#### AllowClockInterrupts();

#### 

#### EnablePeriodicInterrupt();

#### WaitForMSCounter(delayPeriod);

#### 

#### printf("\nEnd delay\n");

#### 

#### setvect(0x70, oldInt70h);

#### 

#### UnlockClockUpdate();

#### }

#### void EnablePeriodicInterrupt()

#### {

#### unsigned char value;

#### 

#### outp(0x70, 0x0B);

#### value = inp(0x0B);

#### outp(0x70, 0x0B);

#### outp(0x71, value|0x40); // 1 to 6th bit

#### }

#### void WaitForMSCounter(unsigned long delayPeriod)

#### {

#### msCounter = 0;

#### while(msCounter != delayPeriod);

#### }

#### void AllowClockInterrupts()

#### {

#### unsigned char value;

#### value = inp(0xA1);

#### outp(0xA1,value & 0xFE);// 0 to 0th bit

#### }

#### int BCDToInteger (int bcd)

#### {

#### return bcd % 16 + bcd / 16 \* 10;

#### }

#### unsigned char IntToBCD (int value)

#### {

#### return (unsigned char)((value/10)<<4)|(value%10);

#### }

#### Результаты работы программы.

#### В программе реализовано меню, позволяющее выбрать тестируемый функционал (установка времени, считывание времени, задержка): '1' - вывод текущего время, '2' – установка времени, '3' – задержка в миллисекундах. Выход из программы производится по нажатию Esc.