Белорусский Государственный Университет Информатики и Радиоэлектроники

Кафедра ЭВМ

Отчет по лабораторной работе № 6
Тема: «Программирование клавиатуры»

Выполнил:
студент гр.950501
Поддубный Д.П.

Проверил:
Одниец Д.Н.

Минск 2021

1. Постановка задачи.

Программируя клавиатуру помигать ее индикаторами. Алгоритм мигания произвольный. Условия реализации программы:

1) Запись байтов команды должна выполняться только после проверки незанятости входного регистра контроллера клавиатуры. Проверка осуществляется считывание и анализом регистра состояния контроллера клавиатуры.

2) Для каждого байта команды необходимо считывать и анализировать код возврата. В случае считывания кода возврата, требующего повторить передачу байта, необходимо повторно, при необходимости – несколько раз, выполнить передачу байта. При этом повторная передача данных не исключает выполнения всех оставшихся условий.

3) Для определения момента получения кода возврата необходимо использовать аппаратное прерывание от клавиатуры.

4) Все коды возврата должны быть выведены на экран в шестнадцатеричной форме.

2. Алгоритм решения задачи.

Для вывода на экран скан-кодов или кодов возврата необходимо заменить обработчик прерывания 09h. При вызове данного обработчика выводится значение из порта 60h на экран. При управлении индикаторами значение из порта 60h (код возврата) необходимо анализировать на случай необходимости повторной передачи байтов команды.

В случае считывания кода возврата, требующего повторить передачу байта, устанавливается флаг commandIsExecuted, сигнализирующий о том, что нет необходимости повторить передачу команды в порт 60h.

Для управления индикаторами клавиатуры используется команда *EDh*. Второй байт этой команды содержит битовую маску для настройки индикаторов (бит 0 – состояние Scroll Lock, бит 1 – состояние Num Lock, бит 2 – состояние Caps Lock). В данной программе управление индикаторами реализовано в функции void SetMask (unsigned char mask), где mask – битовая маска, определяющая состояние индикаторов.

Перед каждой командой записи происходит ожидание освобождения входного буфера клавиатуры: while((inp(0x64) & 0x02) != 0x00).

Главная процедура выполняет следующие действия:

1) Запоминает адрес старого обработчика прерывания 9h, вызывая функцию getvect() с параметром INT = 9.
2) Записывает в таблицу векторов прерываний адрес нового обработчика прерывания с помощью функции setvect().
3) В цикле (пока не установлен флаг выхода quitFlag) производится вызов функции мигания индикаторами Highlight() (если установлен флаг мигания индикаторами needHighlight).

4) Параллельно с этим происходит отслеживание прерываний 09h, и при его возникновении выводится скан-код нажатой клавиши или код возврата.
5) В конце работы программа восстанавливает в таблице векторов прерываний адрес старого обработчика.

3. Листинг программы.

```
#include <dos.h>
#include <conio.h>
#include <stdio.h>

const unsigned char QUIT_CHAR = 0x01;
const unsigned char HIGHLIGHT_CHAR = 0xa3;
const int TARGET_INTERRUPT = 9;
const int true = 1;
const int false = 0;

void interrupt NewInterrupt(void);
void interrupt (*oldInterrupt)(void);
void SaveOldInterrupt();
void SetNewInterrupt();
void RestoreOldInterrupt();
void UpdateHighlightFlag(unsigned char);
void UpdateQuitFlag(unsigned char);
void WaitInputFree();
void SetMask(unsigned char mask);
void Highlight (void);
void EndInterrupt();

int commandIsExecuted;
int quitFlag;
int needHighlight;

void main()
{
commandIsExecuted = false;
quitFlag = false;
needHighlight = true;
clrscr();
SaveOldInterrupt();
SetNewInterrupt();
while(!quitFlag)
{
if (needHighlight)
{
Highlight();
needHighlight = false;
}
```

```
        }
        RestoreOldInterrupt();
        clrscr();
        return;
        }

        void SaveOldInterrupt()
        {
        oldInterrupt = getvect(TARGET_INTERRUPT);
        }

        void SetNewInterrupt()
        {
        setvect(TARGET_INTERRUPT, NewInterrupt);
        }

        void RestoreOldInterrupt()
        {
        setvect(TARGET_INTERRUPT, oldInterrupt);
        }

        void interrupt NewInterrupt()
        {
        unsigned char value = 0;
        oldInterrupt();
        value = inp(0x60);
        UpdateQuitFlag(value);
        UpdateHighlightFlag(value);
        commandIsExecuted = commandIsExecuted || (needHighlight == false) || (value
== 0xFA);
        printf("\t%x", value);
        EndInterrupt();
        }

        void EndInterrupt()
        {
        outp(0x20, 0x20);
        }

        void UpdateQuitFlag(unsigned char value)
        {
        if (value == QUIT_CHAR)
        {
        quitFlag = true;
        }
        }

        void UpdateHighlightFlag(unsigned char value)
        {
        if (value != HIGHLIGHT_CHAR)
```

```
{
return;
}
if (needHighlight)
{
needHighlight = false;
}
else
{
needHighlight = true;
}
}

void SetMask(unsigned char mask)
{
commandIsExecuted = false;
while (!commandIsExecuted)
{
WaitInputFree();
outp(0x60, 0xED);
outp(0x60, mask);
delay(50);
}
}

void WaitInputFree()
{
while((inp(0x64) & 0x02) != 0x00);
}

void Highlight ()
{
printf("\tHere comes the light!");
SetMask(0x02);
delay(1000);
SetMask(0x00);
delay(1000);
SetMask(0x02);
delay(500);
SetMask(0x00);
delay(1000);
printf("\tHere comes the darkness!");
}
```

4. Результаты работы программы.

Программа выводит на экран скан-коды клавиш при их нажатии и отпускании. Мигание индикаторов включается или выключается нажатием

на клавишу H. Во время мигания на экран выводятся коды возврата для каждого байта команды. Выход из программы производится по нажатию Esc.