Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Кафедра ЭВМ

Отчёт по лабораторной работе №5

“ Программирование клавиатуры ”

Проверил: Выполнил:

Минск 2023

#### **Постановка задачи**

#### Программируя клавиатуру помигать ее индикаторами. Алгоритм мигания произвольный. Условия реализации программы, необходимые для выполнения лабораторной работы:

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

#### 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.

#### Для определения момента получения кода возврата необходимо использовать аппаратное прерывания от клавиатуры.

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

#### **Алгоритм решения задачи**

Для работы с клавиатурой используется 2 регистра: 60h – регистр данных, 64h – регистр состояния (статуса).

Для управления индикаторами через 60h отправляется код EDh. Затем маска, в соответствии с которой должны загореться индикаторы.

Формат «маски»:

Биты 7-3 не используются Caps Lock Num Lock Scroll Lock

1 – включить

0 - выключить

Так клавиатура работает медленно, запись байтов команды должна выполняться только после проверки незанятости входного регистра контроллера клавиатуры. Проверка осуществляется считыванием и анализом регистра состояния контроллера клавиатуры (64h, бит 1).

На обработку каждого байта клавиатура отвечает кодом возврата. Если в регистре 60h находится код

FA – байт обработан успешно,

FE – произошла ошибка.

В случае ошибки передачу байта нужно повторить. Пересылка выполняется до 3 раз, если ошибка не исчезла, нужно вывести сообщение и выйти из программы.

При нажатии клавиши блок клавиатуры передает ее код сканирования центральному процессору. Когда клавиша отпускается, клавиатура снова передает ее код, но увеличенный на 128 (или шестнадцатеричное значение 80). То есть, коды для нажатия и отпускания клавиш различаются.

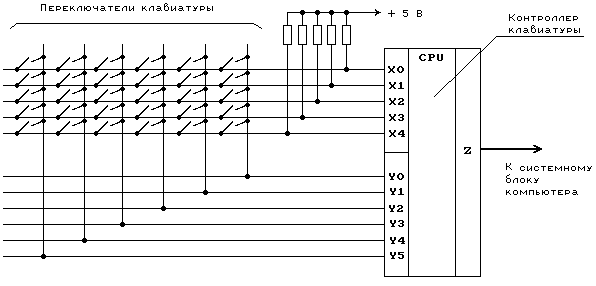
Когда выполняется какое-либо действие с клавишей (нажатие или отпускание), процессор клавиатуры обнаруживает его и запоминает в буфере. Затем формируется прерывание с номером 9.

**Принципы работы клавиатуры.**

Клавиатура выполнена, как правило, в виде отдельного устройства, подключаемого к компьютеру тонким кабелем. Малогабаритные компьютеры Lap-Top используют встроенную клавиатуру.

Внутри клавиатуры находится специализированная микросхема (контроллер клавиатуры), которая отслеживает нажатия на клавиши и посылает номер нажатой клавиши в персональный компьютер.

Если рассмотреть сильно упрощенную принципиальную схему клавиатуры, представленную на рисунке, можно заметить, что все клавиши находятся в узлах матрицы:



Все горизонтальные линии матрицы подключены через резисторы к источнику питания +5 В. Клавиатурный компьютер имеет два порта - выходной и входной. Входной порт подключен к горизонтальным линиям матрицы (X0-X4), а выходной - к вертикальным (Y0-Y5).

Устанавливая по очереди на каждой из вертикальных линий уровень напряжения, соответствующий логическому 0, клавиатурный контроллер опрашивает состояние горизонтальных линий. Если ни одна клавиша не нажата, уровень напряжения на всех горизонтальных линиях соответствует логической 1 (т.к. все эти линии подключены к источнику питания +5 В через резисторы).

Если оператор нажмет на какую-либо клавишу, то соответствующая вертикальная и горизонтальная линии окажутся замкнутыми. Когда на этой вертикальной линии контроллер установит значение логического 0, то уровень напряжения на горизонтальной линии также будет соответствовать логическому 0.

Как только на одной из горизонтальных линий появится уровень логического 0, клавиатурный контроллер фиксирует нажатие на клавишу. Он посылает в персональный компьютер запрос на прерывание и номер клавиши в матрице. Аналогичные действия выполняются и тогда, когда оператор отпускает нажатую ранее клавишу.

Номер клавиши, посылаемый клавиатурным процессором, однозначно связан с распайкой клавиатурной матрицы и не зависит напрямую от обозначений, нанесенных на поверхность клавиш. Этот номер называется скан-кодом (Scan Code).

Слово scan ("сканирование"), подчеркивает тот факт, что клавиатурный контроллер сканирует клавиатуру для поиска нажатой клавиши.

Но программе нужен не порядковый номер нажатой клавиши, а соответствующий обозначению на этой клавише ASCII-код. Этот код не зависит однозначно от скан-кода, т.к. одной и той же клавише могут соответствовать несколько значений ASCII-кода. Это зависит от состояния других клавиш. Например, клавиша с обозначением '1' используется еще и для ввода символа '!' (если она нажата вместе с клавишей SHIFT).

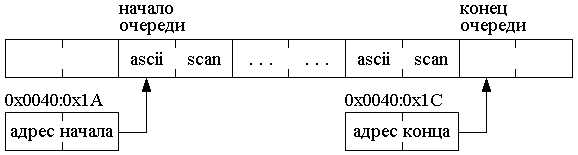
Поэтому все преобразования скан-кода в ASCII-код выполняются программным обеспечением внутри ПК. Как правило, эти преобразования выполняют модули BIOS. Для использования символов кириллицы эти модули расширяются клавиатурными драйверами.

Если нажать на клавишу и не отпускать ее, клавиатура перейдет в режим автоповтора. В этом режиме в центральный компьютер автоматически через некоторый период времени, называемый периодом автоповтора, посылается код нажатой клавиши. Режим автоповтора облегчает ввод с клавиатуры большого количества одинаковых символов.

**Структура буфера клавиатуры**

Клавиатура является основным средством ввода информации при работе на компьютере. В компьютерах IBM PC взаимодействие программ с клавиатурой организовано в асинхронном режиме. При нажатии клавиши на клавиатуре код нажатой клавиши поступает в буфер клавиатуры и хранится там до тех пор, пока программа не затребует его. Если скорость нажатия клавиш и скорость обработки нажатия клавиши программой отличаются, то или программа ожидает поступления кода очередной клавиши в буфер, или коды нажатых клавиш накапливаются в буфере клавиатуры. Хранение информации в буфере организовано в виде очереди (FIFO - First Input First Output), буфер клавиатуры находится в оперативной памяти компьютера и доступен для обычных операций чтения и записи.

Очередь буфера клавиатуры реализована следующим образом:



Каждой нажатой клавише в очереди соответствует ячейка размером в два байта, в первом байте записан ascii код, а во втором - scan код нажатой клавиши. Следует учитывать, что при считывании слова из буфера клавиатуры scan код оказывается в старшем байте, а ascii код в младшем байте слова в соответствии с порядком следования байтов принятом для IBM PC. В двух словах оперативной памяти по адресам 0x0040:0x1A и 0x0040:0x1C хранятся смещения адресов начала (головы) и конца (хвоста) очереди буфера клавиатуры. Следует обратить внимание, что эти смещения указаны относительно адреса 0040:0000h. Сам буфер размером 32 байта (16 слов) расположен в компьютере IBM PC/XT по адресу 0000h:041Eh и организован в виде кольцевой очереди. Следует отметить, что рост буфера (клавиши нажимаются, а символы из буфера не считываются) происходит в сторону хвоста.

В компьютерах моделей IBM PC/AT и IBM PS/2 расположение клавиатурного буфера задается содержимым двух слов памяти с адресами 0000h:0480h (смещение адреса начала буфера относительно 0040h) и 0000h:0482h (смещение конца буфера относительно 0040h). Обычно эти ячейки памяти содержат значения, соответственно, 001Eh и 003Eh, что соответствует расположению клавиатурного буфера в IBM PC/AT и IBM PS/2 его расположению в IBM PC/XT.

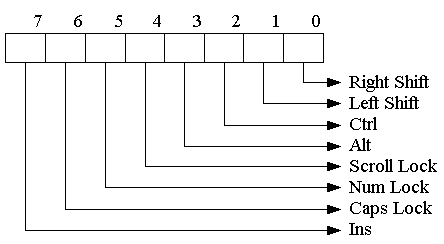
Указателями на начало и конец клавиатурного буфера обычно управляют обработчики прерываний INT 09h и INT 16h. Программа извлекает из буфера коды нажатых клавиш, используя различные функции прерывания INT 16h.

Заметим, что вы можете удалить все символы из буфера клавиатуры, установив оба указателя на начало буфера. Однако есть более предпочтительный способ с использованием, например, прерывания BIOS INT 16h.

Для чтения scan-кода последней нажатой клавиши (даже в том случае если буфер уже заполнен) можно использовать 60-й порт. При этом следует учитывать, что многократное обращение к порту будет возвращать один и тот же результат до тех пор, пока не будет нажата следующая клавиша на клавиатуре.

При переполнении внутреннего буфера клавиатуры или буфера, расположенного в области данных BIOS программа-обработчик прерывания INT 09h генерирует звуковой сигнал.

Кроме буфера клавиатуры и ячеек памяти с указателями головы и хвоста очереди с обработкой нажатия клавиш клавиатуры связано ещё некоторые элементов в области памяти данных BIOS. Одним из этих элементов является флаг состояния клавиатуры, располагающийся в области переменных BIOS по адресу 0000:0417h. Его размер составляет один байт, отдельные биты флага содержат информацию о состоянии клавиш Shift, Ctrl, Alt, Scroll Lock, Num Lock, Caps Lock и Ins. Соответствие перечисленных клавиш и битов флага показано на рисунке:



1. **Листинг программы**

#include <dos.h>

#include <conio.h>

#include <stdio.h>

#include <iostream.h>

// Объявление функций

void interrupt far IRQ1(...);//обработчик клавиатуры

void interrupt far(\*originalIRQ1)(...);//сохранение адреса обработчика перед установкой

void indicator(unsigned char mask);//управление индикаторами клавиатуры

void blinking(void);//мигание

// Глобальные переменные

int isResend = 1; // Флаг повторной отправки команды

int quitFlag = 0; // Флаг выхода из программы

int blinkingFlag = 0; // Флаг мигания индикаторами

int main() {

originalIRQ1 = getvect(0x09); // Сохранение оригинального обработчика прерывания клавиатуры

setvect(0x09, IRQ1); // Установка собственного обработчика прерывания клавиатуры

blinkingFlag=1; // Установка флага мигания индикаторами

blinking(); // Запуск мигания индикаторами

while (!quitFlag) { // Цикл программы до выхода по флагу quitFlag

if (blinkingFlag) // Если установлен флаг мигания индикаторами, то запускается функция blinking()

blinking();

}

setvect(0x09, originalIRQ1); // Восстановление оригинального обработчика прерывания клавиатуры

cout << "Press any key to continue..." << endl;

getch();

return 0;

}

// Обработчик прерывания клавиатуры

void interrupt far IRQ1(...) {

unsigned char value = 0;

value = inp(0x60); // Получение значения скан-кода клавиши

if (value == 0x10) { // Если нажата клавиша "Q", то устанавливается флаг выхода из программы

quitFlag = 1;

}

if (value == 0x39 && blinkingFlag == 0) { // Если нажата клавиша "Пробел" и флаг мигания индикаторами не установлен, то устанавливается флаг мигания индикаторами

blinkingFlag = 1;

}

else {

if (value == 0x39 && blinkingFlag == 1) { // Если нажата клавиша "Пробел" и флаг мигания индикаторами установлен, то снимается флаг мигания индикаторами

blinkingFlag = 0;

}

}

if (value != 0xfa && blinkingFlag == 1) { // Если значение не является успешным кодом и флаг мигания индикаторами установлен, то устанавливается флаг повторной отправки команды

isResend = 1;

}

else {

isResend = 0;

}

cout << "Key scan-code: " << hex << (int)value << endl; // Вывод скан-кода клавиши в шестнадцатеричном формате

outp(0x20, 0x20); // Отправка команды контроллеру прерываний для подтверждения обработки прерывания

}

// Функция управления индикаторами клавиатуры

void indicator(unsigned char mask) {

isResend = 1; // Установка флага повторной отправки команды

while (isResend) { // Цикл, пока флаг повторной отправки команды установлен

while ((inp(0x64) & 0x02) != 0x00); // Ожидание готовности порта для отправки команды

outp(0x60, 0xED); // Отправка команды включения индикаторов

delay(50); // Задержка для стабилизации работы

}

isResend = 1; // Установка флага повторной отправки команды

while (isResend) { // Цикл, пока флаг повторной отправки команды установлен

while ((inp(0x64) & 0x02) != 0x00); // Ожидание готовности порта для отправки команды

outp(0x60, mask); // Отправка команды управления индикаторами

delay(50); // Задержка для стабилизации работы

}

}

// Функция мигания индикаторами клавиатуры

void blinking() {

indicator(0x02); // Включение индикатора Caps Lock

delay(200); // Задержка

indicator(0x04); // Включение индикатора Num Lock

delay(200); // Задержка

indicator(0x06); // Включение обоих индикаторов

delay(200); // Задержка

indicator(0x00); // Выключение всех индикаторов

delay(200); // Задержка

indicator(0x06); // Включение обоих индикаторов

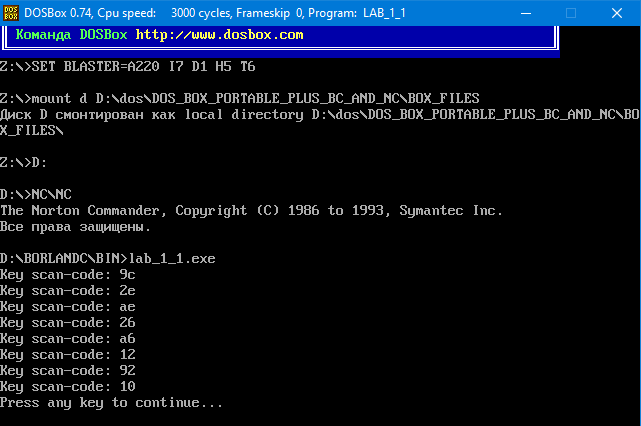
delay(200); // Задержка

indicator(0x00); // Выключение всех индикаторов

cout << "Blinking end!" << endl; // Вывод сообщения о завершении мигания

}

**Тест**



#### **Вывод**

Запрограммировал клавиатуру таким образом, чтобы она мигала ее индикаторами. Алгоритм мигания использовал произвольный. Выполнил следующие условия:

1. Запись байтов команды выполняется только после проверки незанятости входного регистра контроллера клавиатуры. Проверка осуществляет считывание и анализирует регистр состояния контроллера клавиатуры.

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

3. Для определения момента получения кода возврата использовал аппаратное прерывания от клавиатуры.

4. Все коды возврата выведены на экран в шестнадцатеричной форме.