Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Кафедра ЭВМ

Лабораторная работа №3
“Контроллер клавиатуры”

Выполнил:
Студент группы 724402
Чернявский Я. А.

Проверил:
к.т.н., доцент Селезнев И.Л.

Минск 2019

1 Цель работы

1.1 Задача

Программируя контроллер клавиатуры, 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 (H).

1.2 Порядок выполнения работы:

Написать собственный обработчик 9Int;

Сохранить указатель на старый обработчик и определить новый;

Вернуть старый обработчик;

2 Теоретические сведения

Микропроцессор 8048 выполняет:

1. Слежение за нажатиями клавиш и передачи их состояния процессору, самодиагностику (после включения питания компьютера), проверку нажатия клавиш и противодребезговую защиту (что не позволяет воспринимать одну нажатую клавишу как две).

2. Буферизацию до 20 нажатий клавиш, если центральный процессор не может их принять сразу.

Блок клавиатуры не связывает с клавишами никаких конкретных значений. Вместо этого, блок клавиатуры идентифицирует клавишу по ее номеру или коду сканирования. Все клавиши имеют коды сканирования от 1 до 83. 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). Таким образом, имеются различные коды для нажатия и освобождения клавиш.

Клавиатура выполняет еще и функцию повторения клавиши. Блок клавиатуры следит за тем, сколько времени клавиша остается нажатой и формирует сигнал повторения.

Функция повторения распространяется на все клавиши блока клавиатуры.

Каждый переданный компьютеру скан-код (числовое значение) обрабатывается и преобразовывается в код ASCII, который и применяется для передачи смыслового содержания нажатой клавиши. Скан-код для стандартной клавиатуры (84 клавиши) имеет размер 1 байт, а для расширенной — от 2 до 4 байтов. Чтобы отличить расширенный скан- код от обычного, в качестве первого байта всегда выступает значение E0h (например, код левой клавиши равен 38h, а правой — E0h,38h). Кроме уникального кода нажатия, каждая клавиша имеет свой код отпускания. Как правило, этот код состоит из двух байт, первый из которых всегда равен F0h. На расширенных клавиатурах коды отпускания имеют размер три байта, где первые два байта всегда равны EOh, FOh, а третий байт является последним байтом скан-кода нажатия.

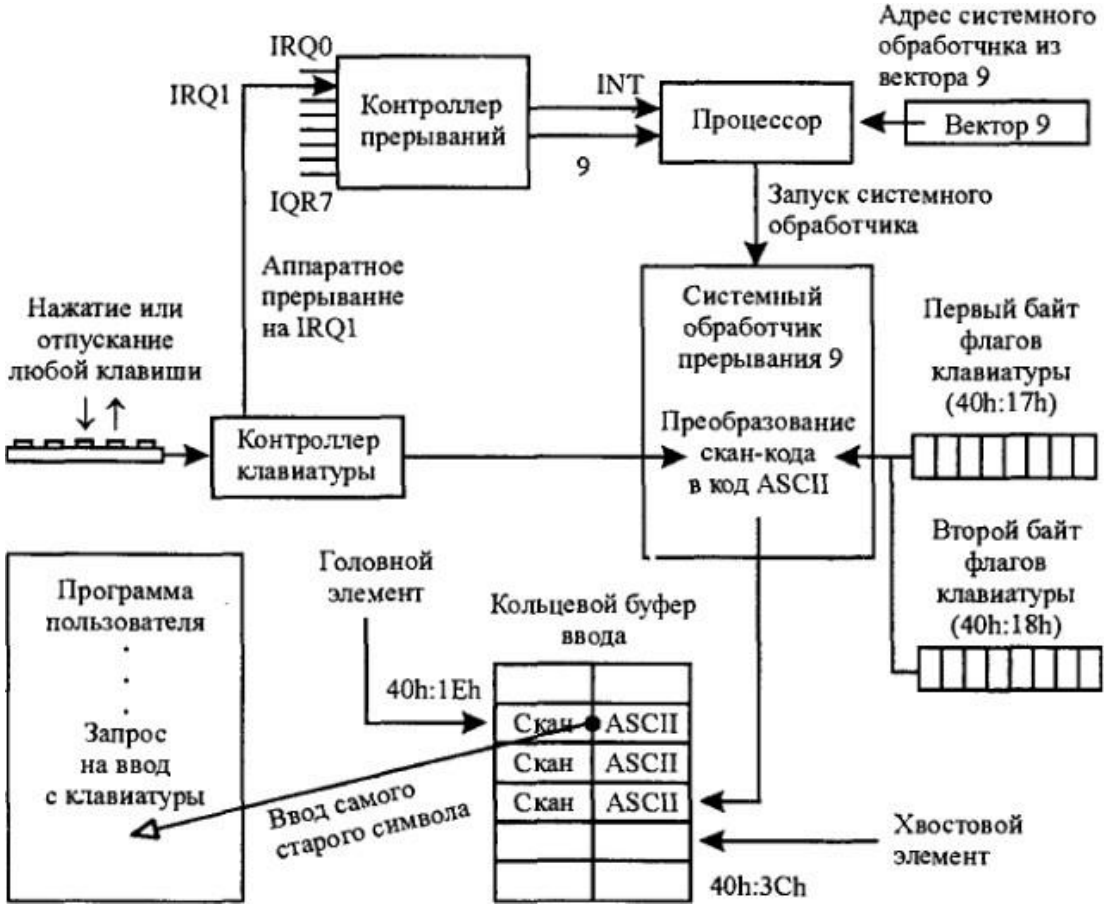

Рисунок 2.1 – Схема управления контроллером клавиатуры.

Программа int09, помимо порта 60h, работает еще с двумя областями оперативной памяти: кольцевым буфером ввода, располагаемым по адресам от 40h:lEh до 40h:3Dh, куда в конце концов помещаются коды ASCII нажатых клавиш, и 2 байтами флагов клавиатуры, находящимися по адресам 40h:17h и 40h:18h. В этих байтах фиксируется состояние управляющих клавиш (Shift, Caps Lock, Num Lock и др.). 64h для чтения - регистр состояния клавиатуры, возвращает следующие биты: бит 1 - в буфере ввода есть данные (для контроллера клавиатуры), бит 0 - в буфере вывода есть данные (для компьютера). При записи в этот порт он играет роль дополнительного регистра управления клавиатурой, но его команды сильно различаются для разных плат.

Клавиша	Код нажатия	Код отпускания	Клавиша	Код нажатия	Код отпускания
F1	05	F0, 05	'	0E	F0, 0E
F2	06	F0, 06	-	4E	F0, 4E
F3	04	F0, 04	=	55	F0, 55
F4	0C	F0, 0C	Backspace	66	F0, 66
F5	03	F0, 03	Tab	0D	F0, 0D
F6	0B	F0, 0B	Space	29	F0, 29
F7	83	F0, 83	Caps Lock	58	F0, 58
F8	0A	F0, 0A	Esc	76	F0, 76
F9	01	F0, 01	Enter	5A	F0, 5A
F10	09	F0, 09	Left Ctrl	14	F0, 14
F11	78	F0, 78	Right Ctrl	E0, 14	E0, F0, 14
F12	07	F0, 07	Left Alt	11	F0, 11
0	45	F0, 45	Right Alt	E0, 11	E0, F0, 11
1	16	F0, 16	Left Shift	12	F0, 12
2	1E	F0, 1E	Right Shift	59	F0, 59
3	26	F0, 26	Left Win	E0, 1F	E0, F0, 1F
4	25	F0, 25	Right Win	E0, 27	E0, F0, 27
5	2E	F0, 2E	Apps	E0, 2F	E0, F0, 2F
6	36	F0, 36	Print Screen	E0, 12, E0 7C	E0, F0, 7C E0, F0, 12
7	3D	F0, 3D	Scroll Lock	7E	F0, 7E
8	3E	F0, 3E	Pause	E1, 14, 77 E1, F0, 14 F0, 77	Нет
9	46	F0, 46	[	54	F0, 54

Рисунок 2.2 – Скан-коды клавиатуры. Часть 1

Клавиша	Код нажатия	Код отпускания	Клавиша	Код нажатия	Код отпускания
A	1C	F0, 1C	]	5B	F0, 5B
B	32	F0, 32	;	4C	F0, 4C
C	21	F0, 21	\	52	F0, 52
D	23	F0, 23	,	41	F0, 41
E	24	F0, 24	.	49	F0, 49
F	2B	F0, 2B	/	4A	F0, 4A
G	34	F0, 34	Insert	E0, 70	E0, F0, 70
H	33	F0, 33	Home	E0, 6C	E0, F0, 6C
I	43	F0, 43	Delete	E0, 71	E0, F0, 71
J	3B	F0, 3B	End	E0, 69	E0, F0, 69
K	42	F0, 42	Page Up	E0, 7D	E0, F0, 7D
L	4B	F0, 4B	Page Down	E0, 7A	E0, F0, 7A
M	3A	F0, 3A	Up Arrow	E0, 75	E0, F0, 75
N	31	F0, 31	Down Arrow	E0, 72	E0, F0, 72
O	44	F0, 44	Left Arrow	E0, 6B	E0, F0, 6B
P	4D	F0, 4D	Right Arrow	E0, 74	E0, F0, 74
Q	15	F0, 15	Num Lock	77	F0, 77
R	2D	F0, 2D	(+) Enter	E0, 5A	E0, F0, 5A
S	1B	F0, 1B	(+) /	E0, 4A	E0, F0, 4A
T	2C	F0, 2C	(+) *	7C	F0, 7C
U	3C	F0, 3C	(+) -	7B	F0, 7B
V	2A	F0, 2A	(+) +	79	F0, 79
W	1D	F0, 1D	(+) .	71	F0, 71
X	22	F0, 22	(+) 0	70	F0, 70
Y	35	F0, 35	(+) 1	69	F0, 69
Z	1A	F0, 1A	(+) 2	72	F0, 72
(IE) Search	E0, 10	E0, F0, 10	(+) 3	7A	F0, 7A
(IE) Home	E0, 3A	E0, F0, 3A	(+) 4	6B	F0, 6B
(IE) Stop	E0, 28	E0, F0, 28	(+) 5	73	F0, 73

Рисунок 2.3 – Скан-коды клавиатуры. Часть 2

Клавиша	Код нажатия	Код отпускания	Клавиша	Код нажатия	Код отпускания
(IE) Refresh	E0, 20	E0, F0, 20	(+) 6	74	F0, 74
(IE) Forward	E0, 30	E0, F0, 30	(+) 7	6C	F0, 6C
(IE) Back	E0, 38	E0, F0, 38	(+) 8	75	F0, 75
(IE) Favorites	E0, 18	E0, F0, 18	(+) 9	7D	F0, 7D
Volume Up	E0, 32	E0, F0, 32	Next Track	E0, 4D	E0, F0, 4D
Volume Down	E0, 21	E0, F0, 21	Previous Track	E0, 15	E0, F0, 15
Mute	E0, 23	E0, F0, 23	Wake	E0, 5E	E0, F0, 5E
Play	E0, 34	E0, F0, 34	Power	E0, 37	E0, F0, 37
Stop	E0, 3B	E0, F0, 3B	Sleep	E0, 3F	E0, F0, 3F

Рисунок 2.4 – Скан-коды клавиатуры. Часть 3

60h для записи - регистр управления клавиатурой. Байт, записанный в этот порт (если бит 1 в порту 61 h равен 0). Интерпретируется как команда. Некоторые команды состоят из более чем одного байта - тогда следует дождаться обнуления этого бита еще раз перед тем, как посылать следующий байт. Команда EEh. Команда позволяет протестировать клавиатуру на предмет работоспособности. Если в работе клавиатуры возникли сбои, следует сделать сброс (команда FFh) и послать эту команду. Возвращаемое значение, отличное от EEh. явно укажет на сбои в работе клавиатуры. Команда F2h. Эта команда позволяет получить идентификатор клавиатуры и убедиться в ее наличии. После выполнения команды клавиатура вернет код подтверждения FAh, а затем идентификатор.

Бит	Описание
0	Наличие данных в выходном буфере клавиатуры (0 — выходной буфер пустой, 1 — в буфере есть данные)
1	Наличие данных во входном буфере клавиатуры (0 — входной буфер пустой, 1 — в буфере есть данные)
2	Результат самотестирования (0 — сброс, 1 — тест прошел успешно)
3	Порт, используемый для последней операции (0 — 60h, 1 — 64h)
4	Состояние клавиатуры (0 — заблокирована, 1 — включена)
5	Ошибка передачи (0 — ошибок нет, 1 — клавиатура не отвечает)
6	Ошибка тайм-аута (0 — ошибка отсутствует, 1 — ошибка)
7	Ошибка четности (0 — ошибка отсутствует, 1 — ошибка) указывает на последнюю ошибку, произошедшую при передаче данных

Формат регистра команд (64h) показан в табл. 3.11.

Таблица 3.11. *Регистр команд (64h)*

Бит	Описание
0	Прерывания для клавиатуры (0 — отключить, 1 — включить)
1	Прерывания для мыши (0 — отключить, 1 — включить)
2	Системный флаг (1 — инициализация через самотестирование, 0 — инициализация по питанию)
3	Не используется
4	Доступ к клавиатуре (0 — открыт, 1 — закрыт)
5	Доступ к мыши (0 — открыт, 1 — закрыт)
6	Трансляция скан-кодов (0 — не использовать, 1 — использовать)
7	Резерв

Рисунок 2.5 – Регистр состояния 64h порта и регистр команд 64h порта

Код команды	Описание
20h	Прочитать байт из регистра команд
60h	Записать байт в регистр команд
A1h	Получить номер версии производителя
A4h	Получить пароль (возвратит FAh, если пароль существует, и F1h — в обратном случае)
A5h	Установить пароль (посылает строку с нулевым символом в конце)
A6h	Проверить пароль (сравнивает введенный с клавиатуры пароль с текущим)
AAh	Выполнить самотестирование контроллера (в случае успеха возвратит 55h)
ABh	Проверка интерфейса клавиатуры (00h — все хорошо, 01h — низкий уровень сигнала синхронизации, 02h — высокий уровень сигнала синхронизации, 03h — низкий уровень сигнала на линии данных, 04h — высокий уровень сигнала на линии данных)
ADh	Отключить интерфейс клавиатуры (устанавливает бит 4 в регистре команд)
AEh	Включить интерфейс клавиатуры (очищает бит 4 в регистре команд)
AFh	Получить версию
C0h	Прочитать входной порт
D0h	Прочитать значение из выходного порта
D1h	Записать параметр в выходной порт
D2h	Записать параметр в буфер клавиатуры
E0h	Тестирование порта (возвращает тестовое значение для порта)

Рисунок 2.6 – Команды управления контроллером клавиатуры

3 Листинг кода программы

```
//724402-2. Chernyavsky Y. L3. Kontroller klaviaturi
#include <stdio.h>
#include <locale.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>

int x = 0;          // Глобальные переменные строки и столбца
int y = 10;         // для вывода на экран
int count = 0;      // Счетчик символов на экране
char header[] = {"724402-2. Chernyavsky Y. L3. Kontroller klaviaturi"};    // Шапка
char menu[] = {"0) Exit \n1) Clean \n2) Blink \n3) Stop blinking "};       // Меню
char regStatus[] = {"Status of registers:"};    // Состояние регистров

void PrintToPoint(char *str, int line, int column);    //Функция вывода
void ClearScreen(int line, int counts);                //Функция очистки экрана
void PrintToField(char *str);                          //Функция вывода
void interrupt newInt9(void);                          //Функция обработки прерывания
void interrupt (*oldInt9)(void);                       //Указатель на обработчик прерывания
void Indicators(unsigned char mask);                   //Функция управления индикаторами
void Blink(void);                                      //Функция мигания индикаторами
void ReadingReg(void);                                 //Функция вывода состояния регистров
void Scroll(void);                                     //Функция смещения строк
void myitoa(int number, char *destination, int base); //Функция перевода чисел в заданную
систему счисления
```

```
int isResend = 1;          //Флаг ошибки
int quitFlag = 0;          //Флаг выхода из программы
int blinkingON = 0;        //Флаг мигания индикаторами

void main()
{
	clrscr();
	PrintToPoint(header, 0, 0);         // Вывод шапки
	PrintToPoint(regStatus, 6, 0);     //Вывод состояния регистров
	PrintToPoint(menu, 1, 0);          // Вывод меню
	oldInt9 = getvect(9);              //Сохраняем указатель на старый обработчик
	setvect(9, newInt9);               //Меняем на новый
	while (!quitFlag)
	{
		if (blinkingON)
			Blink();
	}
	setvect(9, oldInt9);               //Восстанавливаем старый
	clrscr();
	return;
}
void interrupt newInt9()               //Функция обработки прерывания
{
	char str6[9];
	unsigned char value = 0;
	ReadingReg();                      //Вывод функции вывода состояния регистров
	if (count >= 1199)
	{
		Scroll();                  //Вызов функции смещения строк
		y = 24;                    //Установка курсора в 24 строку
		x = 0;                     //0 позицию строки
		count = 1120;              //счетчик символов (-80)
	}
	value = inp(0x60);                 //Получение значения из порта 60h
	if (value == 0x8b || value == 0x01) //Если 0 или ESC, то устанавливаем флаг выхода
		quitFlag = 1;              //Установка флага выхода
	if (value == 0x02)                 //Если 1 , то производим очистку экрана
	{
		ClearScreen(10, 1279);     //Очистка экрана
		y = 10;                    //Установка курсора в начальную позцию
		x = 0;
		count = 0;                 //Счетчик смволов 0
	}
	if (value == 0x03 && blinkingON == 0) //Если 2 , то поставить или снять флаг мигания
		blinkingON = 1;
	else if (value == 0x02 && blinkingON == 1)
		blinkingON = 0;
	if (value != 0xFA && blinkingON == 1) //Если нет подтверждения успешного выполнения
		isResend = 1;
	else
		isResend = 0;                      //то устанавливаем флаг повторной передачи
	if (value == 0x04)
		blinkingON = 0;

	myitoa(value, str6, 16);   //Перевод скан-кода в шестнадцатеричную систему счисления
	PrintToField(str6);        //Вывод скан-кода символа
	outp(0x20, 0x20);          //Сброс контроллера прерывания
}

void Indicators(unsigned char mask) //Функция управления индикаторами
{
	isResend = 1;
	while (isResend)
```

```
	{
		while ((inp(0x64) & 0x01) != 0x00)
			;
		outp(0x60, 0xED); // Команда включения светодиода
		delay(50);
	}
	isResend = 1;
	while (isResend)
	{
		while ((inp(0x64) & 0x01) != 0x00)
			;
		outp(0x60, mask);
		delay(50);
	}
}
void Blink()  // Функция мигания индикаторами
{
	Indicators(0x02);
	delay(50);
	Indicators(0x01);
	delay(30);
	Indicators(0x04);
	delay(50);
	Indicators(0x00);
	delay(20);
	Indicators(0x06);
	delay(30);
	Indicators(0x07);
	delay(50);
}
void PrintToPoint(char *str, int line, int column) //Функция вывода в определённую позицию
{
	int i;
	char far *start = (char far *)0xb8000000; //Начальный адрес видео буфера
	char far *v;
	for (i = 0; str[i] != '\0'; i++)
	{
		if (str[i] == '\n')		//Если в строке находится символ \n
		{
			line++;			//Переход на следующую строку
			column = 0;		//Установка курсора в начало строки
			continue;		//Пропуск вывода символа \n
		}
		v = start + line * 160 + column * 2;
		column++;
		*v = str[i];			//Посимвольный вывод
		v++;
	}
}
void ClearScreen(int line, int counts) //Функция очистки экрана
{
	int i;
	int column = 0;
	char far *start = (char far *)0xb8000000; //Начальный адрес видео буфера
	char far *v;
	for (i = 0; i < counts; i++)
	{
		v = start + line * 160 + column * 2;
		column++;
		*v = ' ';
		v++;
	}
}
void PrintToField(char *str) //Функция вывода в поле для вывода
```

```
{
	int i;
	char far *start = (char far *)0xb8000000; //Начальный адрес видео буфера
	char far *v;
	for (i = 0; str[i] != '\0'; i++)
	{
		v = start + y * 160 + x * 2;
		x++;
		*v = str[i];		//Посимвольный вывод
		v++;
		count++;		//Увеличение счетчика количества символов
	}
	v = start + y * 160 + x * 2;
	x++;
	*v = ' ';
	v++;
	count++;
}
void ReadingReg() //Функция вывода состояния регистров
{
	unsigned char temp;
	char str[4];
	char str1[9];
	temp = inp(0x64);		// Получение регистра 64 порта
	myitoa(temp, str, 16);
	myitoa(temp, str1, 2);
	PrintToPoint("(64h)", 7, 0);
	PrintToPoint(str, 7, 6);
	PrintToPoint("000", 7, 9);
	PrintToPoint(str1, 7, 12);

	temp = inp(0x60);		// Получение регистра 60 порта
	myitoa(temp, str, 16);
	myitoa(temp, str1, 2);
	PrintToPoint("(60h)", 7, 21);
	PrintToPoint(str, 7, 27);
	PrintToPoint(str1, 7, 30);

	temp = inp(0x61);		// Получение регистра 61 порта
	myitoa(temp, str, 16);
	myitoa(temp, str1, 2);
	PrintToPoint("(61h)", 8, 0);
	PrintToPoint(str, 8, 6);
	PrintToPoint("00", 8, 9);
	PrintToPoint(str1, 8, 11);
}
void Scroll() //Функция прокрутки экрана
{
	int i;
	int column = 0;
	char *str = malloc(1280);		//Выделение памяти для данных
	char far *start = (char far *)0xb8000000; //Начальный адрес видео буфера
	char far *v;
	int line = 11;				//Номер строки , откуда начнется считывание
	for (i = 0; i < 1200; i++)
	{
		v = start + line * 160 + column * 2;
		column++;
		str[i] = *v;			//Считывание блока данных
	}
	PrintToPoint(str, line - 1, 0);		//Перезапись блока данных
	ClearScreen(24, 80);			//Очистка последней строки
	free(str);
}
```

```
void myitoa(int number, char *destination, int base) //Функция перевода в заданную систему
счисления
{
	int count = 0;
	int i;
	do
	{
		int digit = number % base;
		destination[count++] = (digit > 9) ? digit - 10 + 'A' : digit + '0';
	} while ((number /= base) != 0);
	destination[count] = '\0';

	for (i = 0; i < count / 2; ++i)
	{
		char symbol = destination[i];
		destination[i] = destination[count - i - 1];
		destination[count - i - 1] = symbol;
	}
	}
```

4 Внешний вид выполнения программы

Ниже представлены скриншоты выполнения программы. На рисунке 4.1 – начальное меню программы и вывод скан-кодов. На рисунке 5.2 – моргание индикатором.

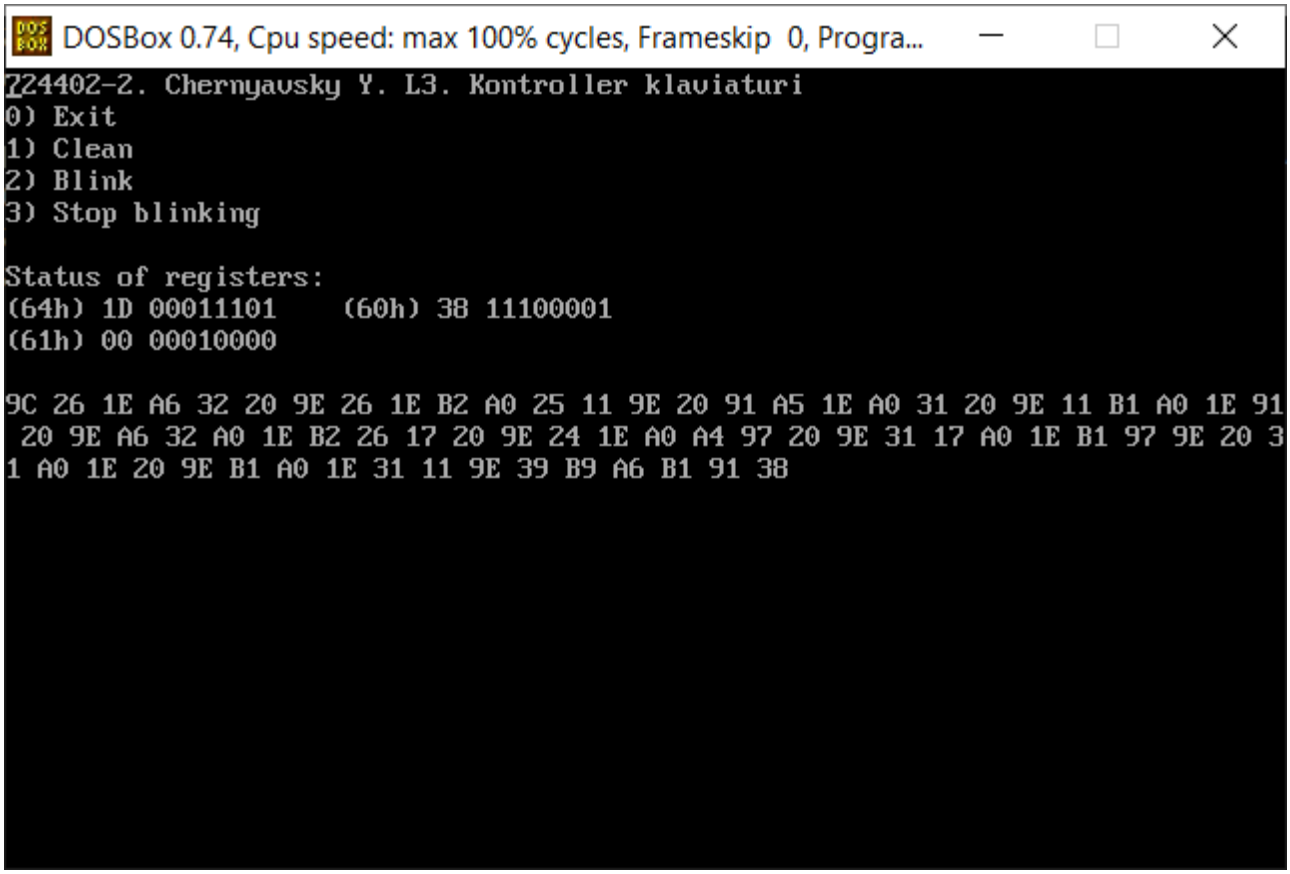

Рисунок 4.1 – Вывод скан-кодов клавиатуры

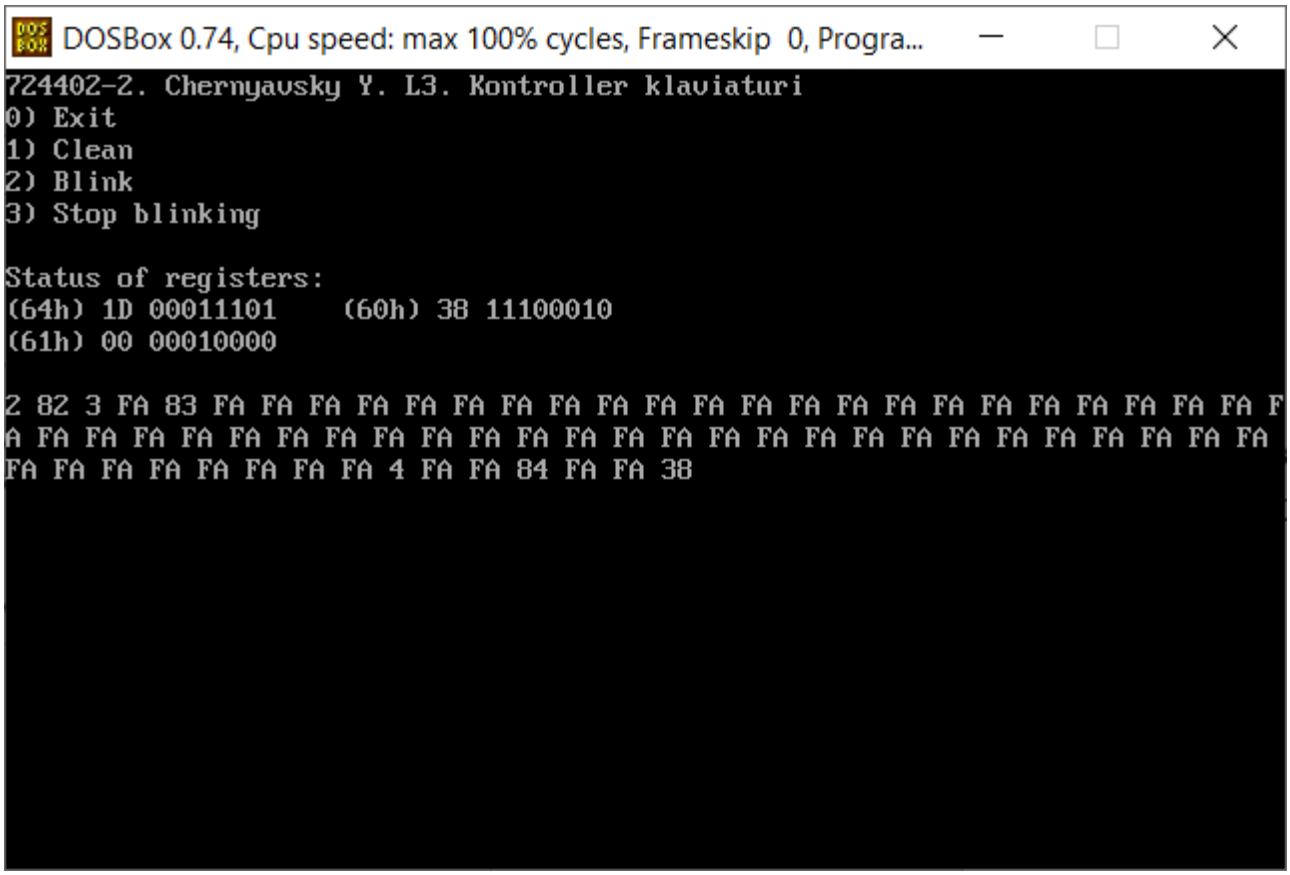

Рисунок 4.2 – Моргание индикатором

5 Вывод

При выполнении работы был разработан свой контроллер прерываний на языке C в эмуляторе MSDOS, который перехватывает сигналы с клавиатуры и выводит информацию на экран.