КОНТРОЛЛЕР КЛАВИАТУРЫ

Задание

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.

Основные принципы функционирования *клавиатуры*

микропроцессор 8048 выполняет :

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).
буферизацию до 20 нажатий клавиш, если центральный процессор не может их принять сразу.

Процедура ввода с клавиатуры системы BIOS в ПЗУ имеет собственный буфер. Буфер клавиатуры может содержать 20 символов, а буфер системы BIOS - только 15.

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.

Для работы с клавиатурой используются порты и прерывания. Когда выполняется какое-либо действие с клавишей(нажатие или освобождение), **процессор клавиатуры обнаруживает его и запоминает в своем буфере. Затем, процессор клавиатуры формирует прерывание с номером 9. В ответ на прерывание служебная процедура системы BIOS в ПЗУ считывает код сканирования клавиши из порта клавиатуры (порт номер 96(60h)) и затем пересылает в порт клавиатуры команду очистить буфер процессора клавиатуры.** Если системный блок не реагирует на прерывания клавиатуры, то коды сканирования накапливаются в буфере процессора клавиатуры,

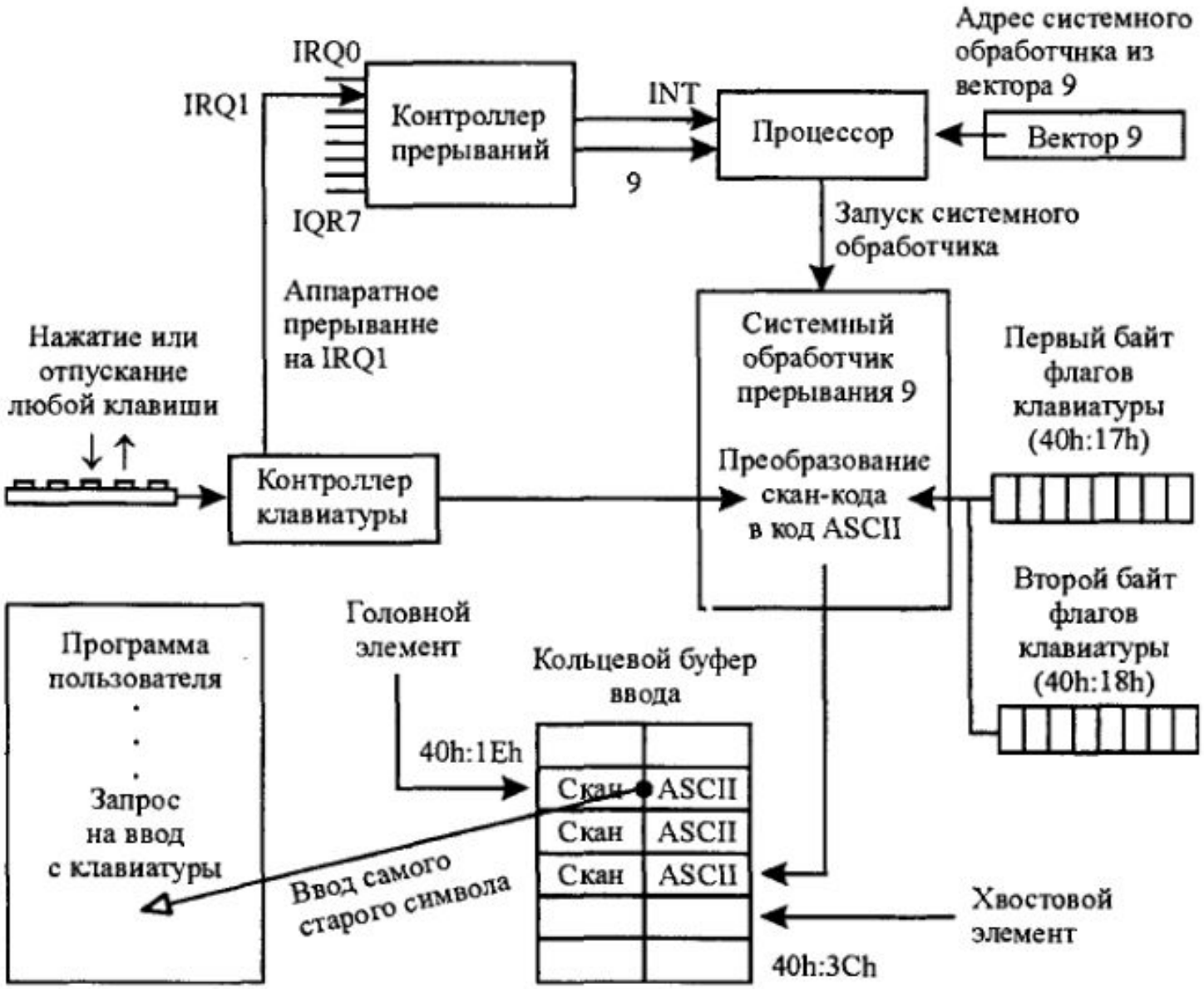
Специальный код сканирования 255, шестнадцатеричное значение FF, используется блоком клавиатуры, для сообщения, что его буфер заполнен.

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.

Процедуры системы BIOS в ПЗУ могут распознавать отличие повторных нажатий клавиши от повторения сигнала удерживаемой в нажатом состоянии клавиши, путем анализа кодов сканирования освобожденной клавиши. Если для одной и той же клавиши получены два кода нажатой клавиши и между ними не было кода освобожденной клавиши, значит клавиша удерживается процедурами системы BIOS для подавления функции повторения тех клавиш, которым она не нужна, таких как, например, клавиша смещения (shift).

Таким образом, Блок клавиатуры занимается физической стороной, механизмом функционирования клавиатуры, а программы системы BIOS в системном блоке выполняют все логические операции по интерпретации действий 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 <Alt> 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.


IRQ0
IRQ1
IQR7
Контроллер прерываний
INT
9
Процессор
Адрес системного обработчнка из вектора 9
Вектор 9
Запуск системного обработчика
Аппаратное прерываннe на IRQ1
Нажатие или отпускание любой клавиши
Системный обработчик прерывания 9
Преобразование скан-кода в код ASCII
Первый байт флагов клавиатуры (40h:17h)
Второй байт флагов клавиатуры (40h:18h)
Контроллер клавиатуры
Программа пользователя
Запрос на ввод с клавиатуры
Головной элемент
Кольцевой буфер ввода
40h:1Eh
Скан
ASCII
Ввод самого старого символа
Хвостовой элемент
40h:3Ch

Существует три основных способа программировании клавиатуры:

1. Поддержка клавиатуры посредством функций BIOS.
2. Работа с контроллером клавиатуры напрямую через порты.
3. Программирование клавиатуры в Win32 API.

- 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

Однако имеется ряд клавиш, которым не назначены отображаемые на экране символы. Это, например, функциональные клавиши F1...F12; При нажатии этих клавиш в кольцевой буфер ввода засылается расширенный код ASCII, в котором младший байт равен нулю, а старший является скан-кодом нажатой клавиши. Расширенные коды ASCII поступают в буфер ввода и в случае нажатия комбинаций управляющих и функциональных клавиш, например Shift+FI, Alt+Insert и др. В этом случае, однако, в старший байт расширенного кода ASCII помещается уже не скан-код клавиши, а некоторый код, специально назначенный этой комбинации клавиш.

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 OEDh 0?h - изменить состояние светодиодов клавиатуры. Второй байт этой команды определяет новое состояние:
бит 0: состояние Scroll Lock (1 - включена, 0 - выключена)
бит 1: состояние Num Lock
бит 2: состояние Caps Lock
При этом состояние переключателей, которое хранит BIOS в байтах состояния клавиатуры, не изменяется, и при первой возможности обработчик прерывания клавиатуры BIOS восстановит состояние светодиодов.

Перед началом работы с клавиатурой следует проверить наличие данных в буфере (бит 0 в регистре статуса). Кроме того, такую проверку необходимо выполнить перед любыми последующими операциями записи. После проверки буфера в регистр 64h записывается код желаемой команды

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.

Примеры

- Проверка наличия клавиатуры
- Управление индикатором <Num Lock>

Листинг 3.11. Проверка наличия клавиатуры

```
; ждем освобождения входного буфера клавиатуры
@keyb_wait:
in   AL, 64h    ; опрашиваем регистр состояния
test AL, 010b   ; если буфер занят
jnz  @keyb_wait; повторяем опрос
mov  AL, 0ABh   ; команду тестирования интерфейса
out  64h, AL    ; записываем в порт
```

Для программистов C++ этот пример представлен в листинге 3.12.

Листинг 3.12. Проверка наличия клавиатуры в C++

```
DWORD dwResult  = 0; // переменная для хранения результата
int iTimeWait = 50000;
// проверяем наличие данных во входном буфере клавиатуры
while ( -- iTimeWait > 0)
{
        // читаем состояние порта
        inPort ( 0x64, &dwResult, 1);
        if ( ( dwResult & 0x02) == 0x00) break;
        // закончилось время ожидания
        if ( iTimeWait < 1) return MY_ERROR_TIME;
}
// записываем в порт команду проверки интерфейса
outPort ( 0x64, 0xAB, 1);
```

Листинг 3.15. Управление индикатором <Num Lock>

```
@keyb_wait:
in   AL, 64h    ; опрашиваем регистр состояния
test AL, 010b   ; если буфер занят
jnz  @keyb_wait; повторяем опрос
mov  AL, 0EDh   ; команду управления индикаторами
out  60h, AL    ; записываем в порт данных
@data_wait:
in   AL, 64h    ; опрашиваем регистр состояния
test AL, 010b   ; если буфер занят
jnz  @data_wait; повторяем опрос
mov  AL, 010b   ; включаем Num Lock
out  60h, AL    ; записываем в порт значение
```

```
DWORD dwResult  = 0; // переменная для хранения результата
int iTimeWait = 50000;
// проверяем наличие данных во входном буфере клавиатуры
while ( -- iTimeWait > 0)
{
        // читаем состояние порта
        inPort ( 0x64, &dwResult, 1);
        if ( (dwResult & 0x02) == 0x00) break;
        // закончилось время ожидания
        if ( iTimeWait < 1) return MY_ERROR_TIME;
}
 // записываем в порт команду управления индикаторами
 outPort ( 0x60, 0xED, 1);
 int iTimeWait = 50000;
 // проверяем наличие данных во входном буфере клавиатуры
 while ( -- iTimeWait > 0)
 {
        // читаем состояние порта
        inPort ( 0x64, &dwResult, 1);
        if ( (dwResult & 0x02) == 0x00) break;
        // закончилось время ожидания
        if ( iTimeWait < 1) return MY_ERROR_TIME;
 }
 outPort ( 0x60, 0x02, 1);
```