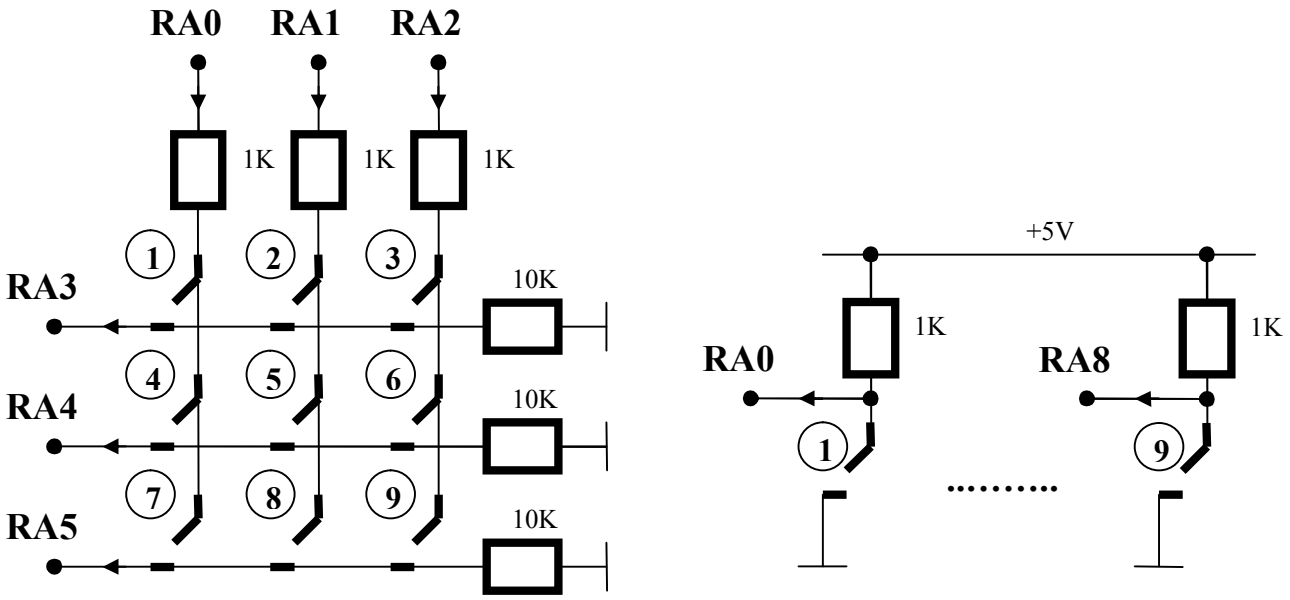
ГЛАВА 13 Матричные клавиатуры

Появление матричных клавиатур [17] было объективно вызвано ростом числа клавиш в них в случаях организации сложных алгоритмов управления, либо большого числа управляемых устройств. Использование для этих целей стандартных клавиатур, в виде кнопок с резисторами, включаемых между шинами «земли» и питания, нецелесообразно из-за пропорционального роста соединительных линий между ними и объектами управления. В этом случае 100 «обычных» кнопок требует, по крайней мере, 100 подводящих проводников, не считая цепей питания. Использование матричного принципа организации клавиатур позволяет существенно, в несколько раз, уменьшить число коммутирующих линий. Ввиду того, что при этом усложняется алгоритм обработки сигналов клавиатуры, для его реализации целесообразно использование либо специализированных, либо универсальных микроконтроллеров.

13.1 Матричные клавиатуры, структура, принцип работы, дребезг контактов

На рис. 93 приведены схемы матричной и «обычной» клавиатур из 9-ти клавиш, подключенных для определенности к контактам порта А (хотя, строго говоря, его кон-



матричная клавиатура «обычная» клавиатура

Рис. 93 Схема матричной клавиатуры с организацией 3х3

тактов в случае PIC 16F877 не хватит). Из рисунка видно, что для подключения матричной клавиатуры требуется 6 проводников (контактов контроллера с резисторами) при отсутствии источника питания, вместо 9-ти для «обычной». Название «матричная» отражает способ включения клавиш «1» - «9» как бы на пересечении строк и столбцов «матрицы». Эту же аналогию используют для характеристики числа клавиш в матричных клавиатурах, когда говорят не «9-ти контактная матричная клавиатура», а клавиатура «с организаций 3х3» по числу строк и столбцов.

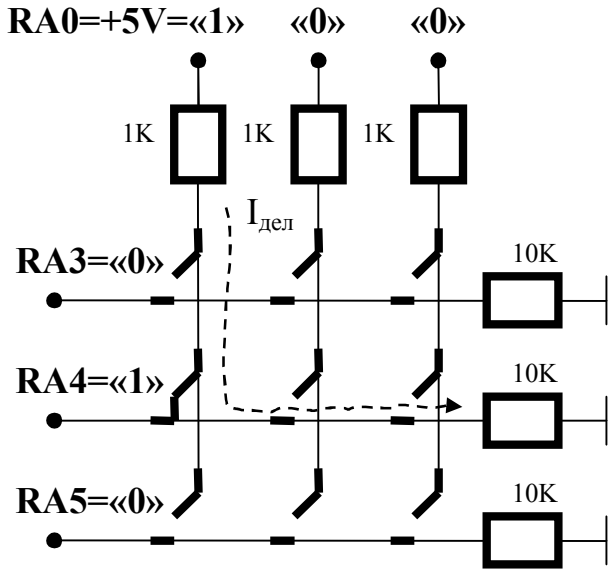
Отличается и способ подключения матричных клавиатур к контактам контроллера. «Столбцы» клавиатуры подключаются к выходам портов, которые являются «источ-

никами питания» для клавиш в столбце, а «строки» - к входам, определяющим факт нажатия клавиши по уровню входного сигнала. В этом заключается недостаток матричных клавиатур, т.к. усложняется алгоритм их опроса, поэтому наиболее часто их используют в микропроцессорных или микроконтроллерных системах, возлагая обработку на ПО контроллера. В «обычных» клавиатурах – клавиши подключаются к входам портов.

Таким образом, матричная клавиатура имеет следующие **особенности**:

- организация клавиш в виде матрицы **размерностью «m x n»**;
- клавиши **замыкают** между собой провода на **пересечении** строк и столбцов;
- **столбцы** подключены к **выходам** контроллера, **строки** – ко **входам**;
- роль **источника питания** для работы клавиатуры выполняют выходные контакты порта контроллера во время формирования на них уровня **логической «1»**;
- клавиатура представляет собой пассивное устройство, работу которого обеспечивает **контроллер, формирующий требуемые сигналы на столбцах и считывающий их на строках**.

На рис. 94 приведены уровни сигналов, формируемых контроллером на контактах портов в процессе работы клавиатуры. Этот процесс часто называют **опросом (сканированием) клавиатуры**. Поскольку сигналы периодичны, процесс опроса удобно представить в виде **алгоритма опроса**:



Клавиша 4 – «нажата»

$$I_{дел} = \frac{+5V}{1+10}$$

$$U_{RA4} = I_{дел} \times 10 = +5V \frac{10}{10+1} \approx +5V = \text{"1"}$$

Клавиша 4 – «отпущена»

$$I_{дел} = 0$$

$$U_{RA4} = 0V = \text{"0"}$$

Рис. 94 Принцип работы матричной клавиатуры и уровни сигналов на контактах контроллера

- на контакте RA0 контроллер формирует уровень «1», на RA1 = RA2 = «0». В этом случае возможно протекание тока через нажатые кнопки первого (левого) столбца: клавиши «1», «4», «7» (нумерация клавиш приведена на рис. 93). Через все остальные клавиши, ток протекать не будет. На рис. 94 показан путь протекания тока при нажатой клавише «4»: +5V (RA0) → резистор 1К → клавиша «4» → резистор 10К → корпус. Соединенные в этом случае резисторы 1К и 10К образуют делитель, через который начинает протекать ток (величина тока приведена на рис. 94). Этот ток создает падение напряжения на резисторе 10К, которое поступает на вход RA4. При нажатой кнопке «4» и указанном соотношении резисторов это напряжение будет равно логической «1». При отпущенной кнопке ток не протекает и напряжение на резисторе 10К, а соответственно

и на контакте RA4 равно «0». Таким образом, комбинация сигналов на порте А (начиная со старшего RA5) будет «000001» - при отпущенной клавише «4», и «010001» - при нажатой. Эту комбинацию часто называют **кодом** нажатой **клавиши**. Аналогично может быть сформирован код клавиш «1» и «7» соответственно равные «001001» и «100001»;

- на контакте RA1 контроллер формирует уровень «1», на RA0 = RA2 = «0». В этом случае возможно протекание тока через нажатые кнопки второго (среднего) столбца: клавиши «2», «5», «8» и аналогично могут быть определены их коды. Например, код клавиши «8» равен «100010»;

- в заключение сканируется третий столбец путем формирования уровня логической «1» на RA2 и определяются коды нажатия клавиш «3», «6», «9»;

- процесс циклически повторяется со скоростью превышающей скорость возможного нажатия клавиш, чтобы нажатие и отпускание клавиши не произошло в интервал отсутствия «1» на столбце в котором находится эта клавиша, т.е. чтобы контроллер «не пропустил» факт нажатия клавиши.

Фактически, контроллер, выставив «1», на столбце, «ищет» ее на одной из строк, на пересечении которых и находится нажатая клавиша.

Примечание: на практике обычно оговаривается максимально возможное число одновременно нажатых клавиш, которое может корректно обнаружить контроллер. Например, если в момент сканирования первого столбца, при нажатой клавише «4», нажать клавиши «5» и «6», то резисторы 1К второго и третьего столбцов окажутся включены параллельно резистору 10К в этой строке и вызовут снижение уровня на контакте RA4, который может опуститься ниже уровня логической «1».

В любом случае, при нажатии на клавишу, на соответствующем входе порта появляется уровень «1», при ее отпускании – уровень «0» (в некоторых схемах включения - наоборот). В этом смысле матричная клавиатура не отличается от набора «обычных» кнопок, включаемых между шинами +5V и землей и, соответственно, обладает всеми их недостатками. Основным из них является **дребезг контактов**, эффект, связанный с не идеальной поверхностью механического контакта внутри самой клавиши и **приводящий к многократному изменению логического уровня при однократном нажатии клавиши**. Уровни сигналов на контактах контроллера в таких случаях приведены на рис. 95. Здесь показано изменение сигнала на контакте RA4 для идеального контакта, когда одно нажатие приводит к единственному скачку уровня (верхняя диаграмма), и та же осциллограмма для реального механического контакта (нижняя диаграмма). В этом случае, очевидно, что контроллер зафиксирует несколько нажатий и «отпусканий» кнопки, причем не только в момент реального нажатия, но при реальном отпускании клавиши. Очевидно, что этот эффект может нарушить работу устройств, реагирующих на число или последовательность нажатия определенного числа клавиш. Третья диаграмма приведена в режиме временного увеличения и позволяет оценить количество ложных нажатий, которые обнаружит контроллер, считывающий значение логического сигнала с порта RA4 в моменты, показанные стрелками. **Реальные значения длительностей этих процессов таковы: $T_{дребезга}$ = 0,5 миллисекунды, $T_{нажатия}$ = 100 миллисекунд.**

Одним из программных [18] **методов защиты от дребезга** является **задержка опроса** клавиатуры, введенная **после обнаружения первого изменения** уровня **сигнала от клавиши**. Другими словами, контроллер, обнаружив первое изменения сигнала с «0» на «1» (т.е. нажатие) должен перестать сканировать клавиатуру на время, превышающее время дребезга, а затем повторно проверить есть ли уровень «1» на клавише. Если уровень «1» сохранился, значит, клавиша действительно нажата и контроллер формирует ее

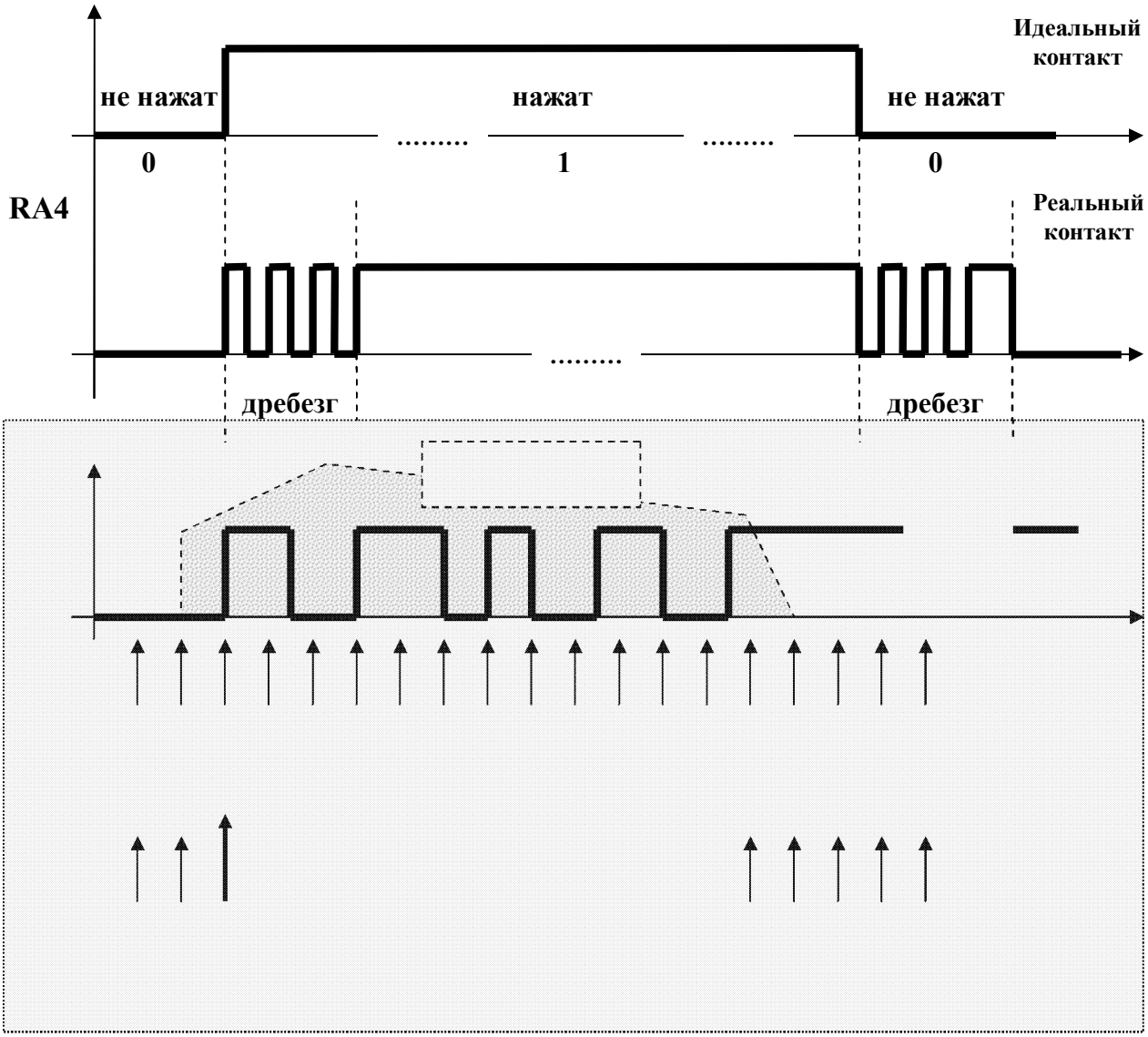


Рис. 95 Эффект дребезга контактов и один из методов борьбы с ним

код. Если же после задержки на клавише обнаруживается «0», то принимается решение об отсутствии нажатия.

Примечание: строго говоря, в этом случае возможны ситуации когда, дребезг еще продолжается, либо кнопку уже отпустили. Эти проблемы решаются выбором длительности задержки.

Для матричной клавиатуры, задержка опроса на 1 мс вводится после обнаружения «1» на каждой из строк, при установленной «1» на каждом столбце.

Все указанные действия выполняет специальное ПО, называемое **драйвером клавиатуры**. Драйвер обычно оформляется в виде вызываемой процедуры и включается в тело основной программы.

13.2 Драйвер матричной клавиатуры, алгоритма работы, исходный код ПО

Драйвер решает следующие задачи:

- при его вызове осуществляет **один цикл опроса текущего состояния всех клавиш** клавиатуры, путем установки уровня «1» поочередно на каждом из столбцов

матрицы и считывания состояния каждой из строк. Это делается для повышения скорости работы и упрощения алгоритма драйвера;

- определяет, **нажата ли клавиша и формирует ее код**, помещая его (либо иное оговоренное число) в некоторый регистр по некоторому алгоритму;

- определяет **число одновременно нажатых клавиш** в этом цикле опроса, с помещением этого числа в некоторый регистр;

- реализует встроенную программную **задержку для защиты от дребезга**.

Таким образом, на драйвер возлагается лишь минимально необходимый набор функций обработки сигналов с клавиатуры, необходимый для решения любой задачи. Все остальные задачи, например, определение событий, отпущена ли нажатая клавиша, есть ли одновременно (и не только в одном цикле опроса) нажатые клавиши, или последовательность в которой нажимались клавиши и т.д. возлагаются на основное ПО, т.к. во многом они определяются особенностями решаемой задачи.

Очевидно, что при использовании драйвера в виде вызываемой процедуры **период ее вызова должен быть меньше, чем время нажатия клавиши**.

На рис. 96 приведена блок-схема алгоритма работы драйвера матричной клавиатуры 3x3 клавиши, оформленной в виде процедуры Keyboard. Рассмотрим ее особенности:

- цикл работы драйвера заключается в **однократном выполнении алгоритма опроса клавиатуры,** изложенного в п. 13.1;

- **результатом** работы **драйвера** является **запись числа 0x01 в регистры Key1 – Key9,** соответственно при нажатых клавишах «1» - «9», а также **запись в регистр NumPressKey числа одновременно нажатых** в этом цикле опроса **клавиш**. Например, при нажатии клавиш «5», «8» в регистры Key5 и Key8 запишутся 1, а в регистр NumPressKey – число 0x02. Если эти клавиши не нажимались в этом цикле опроса - в этих регистрах останутся 0. В данной реализации драйвера в качестве кода нажатой клавиши выступает адрес того из регистров Key1 – Key9, в котором будет записано число 0x01;

- **алгоритм опроса содержит 3 этапа по столбцам**: вначале сканируется 1-ый столбец клавиатуры для определения нажатия клавиш «1» «4» «7», затем второй (клавиши «2» «5» «8»,) и третий - клавиши «3» «6» «9». На рис 96 первый этап изображен полностью, второй и третий – упрощенно в виде эллипсов. Число одновременно нажатых клавиш определяется суммированием их кодов, т.к. при нажатой клавише в ее регистре будет 1;

- определение нажата ли одна из клавиш столбца осуществляется последовательно: вначале ищется 1 на одном из входов RA3-RA5, а затем, при ее наличии - определяется строка, на которой находится нажатая клавиша. При отсутствии 1 на контактах RA3-RA5 переходим к сканированию 2-го столбца и далее третьего;

- **защита от дребезга** осуществляется путем введения **задержки** на 1 миллисекунду **после** того как обнаружен факт **нажатия одной из клавиш** в сканируемом столбце;

- дополнительно вводится задержка по времени в 100 мкс, необходимая для завершения процессов установления напряжения логической «1» на входах контроллера, подключенных к строкам клавиатуры, в случае если нажата клавиша;

- «пропустить» нажатие клавиши процедура не может, т.к. время ее выполнения ≈ 5 мс (при F_{OSC} = 4МГц, см. ниже исходный код) существенно меньше минимально возможного времени нажатия клавиши ≈ 100 мс.

Ниже приведен исходный код процедуры Keyboard с комментариями. Все регистры расположены в нулевом банке, присвоение имен регистрам осуществляется в теле

основного ПО.

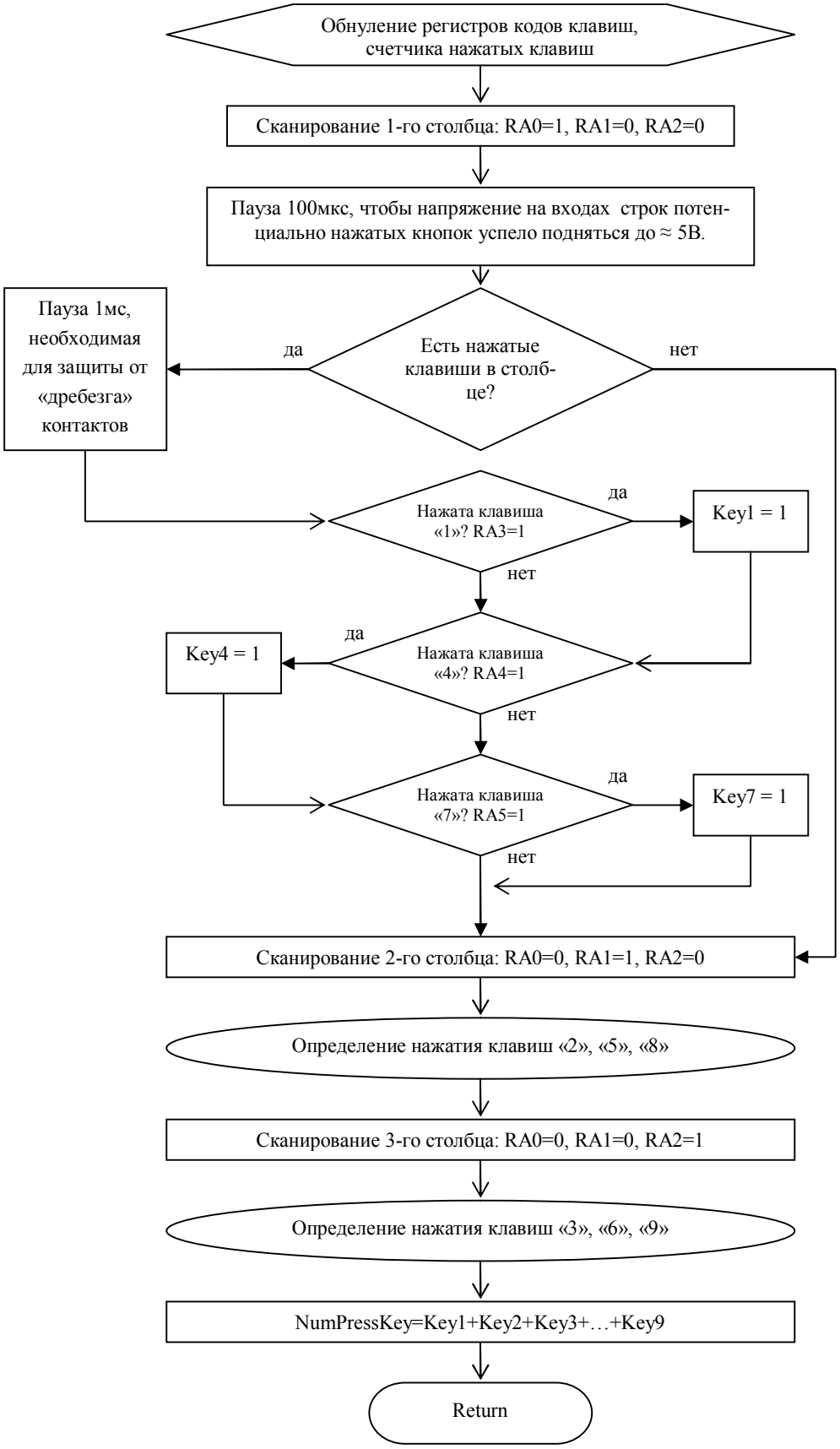


Рис. 96 Блок-схема драйвера клавиатуры (процедура Keyboard)

```
Keyboard                                    ; метка начала процедуры
        bcf STATUS, RP0                     ; переход в нулевой банк, для обес-
        bcf STATUS, RP1                     печения корректности выполнения
                                            процедуры при ее вызове из любого
                                            места основного ПО
        clrf NumPressKey                    ; обнуление регистра, в котором бу-
                                            дет храниться число нажатых кла-
                                            виш, обнаруженных в одном цикле
                                            сканирования клавиатуры, т.е. при
                                            однократном вызове процедуры
                                            Keyboard. Так обеспечивается об-
                                            новление результатов сканирования
                                            в каждом следующем цикле опроса
        clrf Key1 clrf Key2  clrf Key3      ; обнуление регистров, в которых
        clrf Key4 clrf Key5 clrf Key6       хранится состояние клавиш (нажата
        clrf Key7 clrf Key8 clrf Key9       – 1, отпущена - 0)
col1    bsf PORTA,0                         ; подача питания на первый столбец
        bcf PORTA,1                         (колонку) клавиатуры для начала
        bcf PORTA,2                         опроса состояния клавиш «1», «4»,
                                            «7»
        movlw .24                           ; вызов процедуры задержки на ≈
        call small_delay                    100 микросекунд для завершения
                                            переходного процесса установления
                                            +5V на RA0. Это делается путем вы-
                                            зова процедуры  Small delay с пара-
                                            метром w=24, описание которой
                                            приведено в примере исходного ко-
                                            да АЦП в п. 6.3
        movf PORTA,W                        ; проверка нажата ли хотя бы одна
        andlw 0x38                          клавиша в первом столбце. Осуще-
                                            ствляется путем чтения контактов
                                            порта A в W и выполнения поразря-
                                            дного умножения битов порта на
                                            число 0011 1000. В этом случае 5-
                                            ый, 4-ый и 3-ий разряды порта, на
                                            которых «1» будут только при нажа-
                                            тии клавиш соответственно «7»,
                                            «4», «1» умножаются на эти же раз-
                                            ряды числа 0x38. При нажатии хотя
                                            бы одной из них результат логиче-
                                            ского умножения будет не равен 0.
                                            Если же не нажата ни одна из них –
                                            результат операции будет равен 0, о
                                            чем «скажет» Z=1.
        btfsc STATUS,Z                      ; если Z=1 (т.е. не нажата ни одна из
        goto col2                           кнопок) переходим на метку col2
                                            для сканирования второго столбца
                                            клавиатуры
```

```
        movlw .250          ; если Z=0 (т.е. нажата хотя бы одна
        call small_delay    из кнопок) вызываем процедуру за-
                            держки на 1 миллисекунду для за-
                            щиты от дребезга (см. рис. 95) при
                            повторном считывании контактов
                            RA3, RA4, RA5
        btfsc PORTA,3       ; опрос первой строки, а т.к. уровень
        incf Key1,F         «1» подан на первый столбец, это
                            фактически определение состояние
                            клавиши «1». Если в этот момент
                            RA3 = «1», то клавиша «1» - нажата,
                            если RA3 = «0», то – отпущена.
                            Увеличение на 1 обнуленного реги-
                            стра Key1 происходит если RA3 =
                            «1», если нет, то переходим к анали-
                            зу состояния контакта RA4
        btfsc PORTA,4       ; аналогичное определение нажатия
        incf Key4,F         клавиши «4» и запись 1 в регистр
        btfsc PORTA,5       Key4  и клавиши 7 с записью в ре-
        incf Key7,F         гистр Key7
col2                        ; метка начала сканирования 2-го
                            столбца клавиатуры
        bcf PORTA,0         ; подача питания на второй столбец
        bsf PORTA,1         (колонку) клавиатуры для начала
        bcf PORTA,2         опроса состояния клавиш «2», «5»,
        movlw .24           «8» с задержкой сканирования
        call small_delay
        movf PORTA,W        ; аналогичная проверка (см. выше)
        andlw 0x38          нажата ли хотя бы одна из клавиш
        btfsc STATUS,Z      «2», «5», «8». Если да (Z=0), запус-
        goto col3           кается процедура задержки на 1 мс,
        movlw .250          для защиты от дребезга, если нет
        call small_delay    (Z=1) – переходим к сканированию
        btfsc PORTA,3       3-го столбца клавиатуры. После за-
        incf Key2,F         держки повторно сканируем состоя-
        btfsc PORTA,4       ния контактов RA3, RA4, RA5, для
        incf Key5,F         определения действительно ли на-
        btfsc PORTA,5       жата одна из этих клавиш. При об-
        incf Key8,F         наружении 1 на них, в соответст-
                            вующий регистр Key записывается 1
col3                        ; метка начала сканирования 3-го
                            столбца клавиатуры
        bcf PORTA,0         ; подача питания на третий столбец
        bcf PORTA,1         (колонку) клавиатуры для начала
        bsf PORTA,2         опроса состояния клавиш «3», «6»,
        movlw .24           «9» с задержкой сканирования. Ана-
        call small_delay    логичная проверка (см. выше) нажа-
        movf PORTA,W        та ли хотя бы одна из названных
```

```
        andlw 0x38              клавиш. Если да (Z=0), запускается
        btfsc STATUS,Z          процедура задержки на 1 мс, для
        goto cnt                защиты от дребезга и повторного
        movlw .250              определения какие же клавиши на-
        call small_delay        жаты, если нет (Z=1) – переходим на
        btfsc PORTA,3           метку cnt для подсчета общего чис-
        incf Key3,F             ла одновременно нажатых клавиш
        btfsc PORTA,4           из предыдущих столбцов, т.к. ска-
        incf Key6,F             нирование третьего столбца закон-
        btfsc PORTA,5           чено, а, следовательно, просканиро-
        incf Key9,F             вана вся клавиатура
cnt                             ; метка начала подсчета числа одно-
                                временно нажатых клавиш, обнару-
                                женных в этом цикле работы проце-
                                дуры
        btfsc Key1,0            ; анализ проводится путем последо-
        incf NumPressKey,F      вательной проверки значения млад-
        btfsc Key2,0            шего разряда каждого из регистров
        incf NumPressKey,F      Key1 - Key9. Так, если Key1=1, то
        btfsc Key3,0            NumPressKey  увеличиваем на 1, ес-
        incf NumPressKey,F      ли нет – анализируем Key2=1 и
        btfsc Key4,0            опять добавляем 1 в NumPressKey, и
        incf NumPressKey,F      т.д. все девять регистров Key. В ре-
        btfsc Key5,0            зультате в регистре NumPressKey
        incf NumPressKey,F      оказывается записанным число рав-
        btfsc Key6,0            ное числу одновременно нажатых
        incf NumPressKey,F      клавиш
        btfsc Key7,0
        incf NumPressKey,F
        btfsc Key8,0
        incf NumPressKey,F
        btfsc Key9,0
        incf NumPressKey,F
        return                  ; выход из процедуры
```

13.3 Пример использования матричной клавиатуры в блоке управления кодовым замком на PIC 16F877, блок-схема алгоритма работы, исходный код ПО

Постановка задачи: разработать ПО для PIC16F877 в блоке управления кодовым замком с матричной клавиатурой, функционирующее по следующему алгоритму:

- при включении контроллера замок закрыт (RD1=0), аварийная сирена выключена (RC1=0);

- открывание замка (RD1=1) происходит при последовательном нажатии и отпускании клавиш «1», «8» по отпусканию последней клавиши;

- при нажатии любой иной комбинации двух или более клавиш, срабатывает сирена (RC1=1). В обоих случаях после этого ПО останавливает свою работу до сигнала Reset;

- схема подключения замка и сирены к контроллеру приведена на рис. 97.

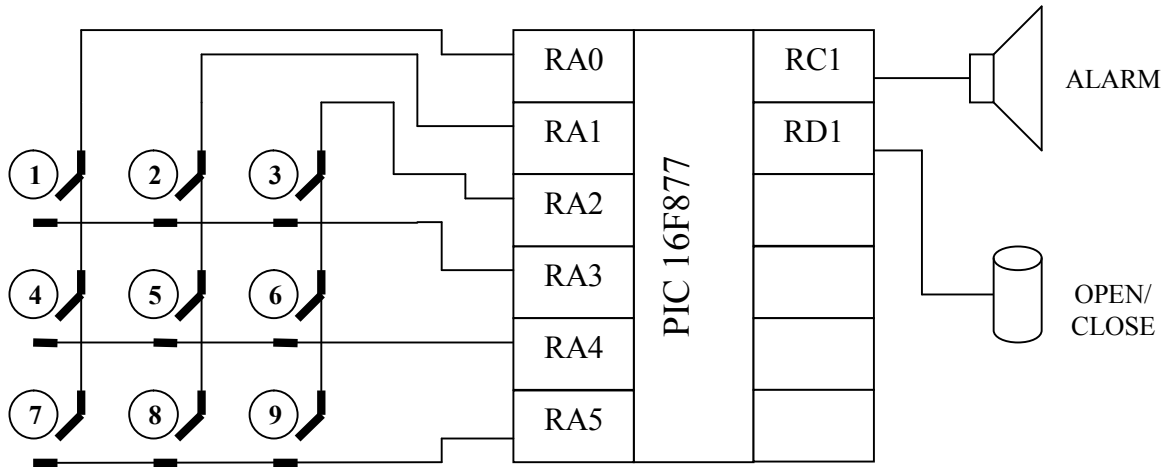


Рис. 97 Схема блока управления кодовым замком
с матричной клавиатурой на PIC 16F877

На рис. 98 приведена блок-схема алгоритма работы блока управления. Рассмотрим ее особенности:

- состоит из двух последовательных идентичных блоков проверки нажатия (и отпускания) клавиш «1» и «8». На блок-схеме первый блок показан полностью, а второй - схематично в виде эллипса;

- для сканирования клавиатуры используется вызываемая процедура драйвера Keyboard;

- факт корректного нажатия и отпускания клавиши проверяется путем вызова драйвера клавиатуры с последующей проверки совпадения двух событий: одна ли клавиша нажата и является ли она клавишей «1». При их совпадении содержимое контрольного регистра Control уменьшается на 1, говоря о том, что нажата и отпущена именно нужная клавиша. Если же нажатых клавиш не было, или же удалось «одновременно» нажать несколько, то продолжаем сканировать клавиатуру, до отпускания последней, но содержимое Control не уменьшается на 1, говоря о том, что была нажата и отпущена неправильная клавиша;

- при двух нажатых и отпущенных правильных клавишах Control становится равным нулю, и замок открывается, в противном случае звучит сирена. В обоих случаях, после этого контроллер входит в бесконечный цикл, выйти из которого может только по сигналу сброса.

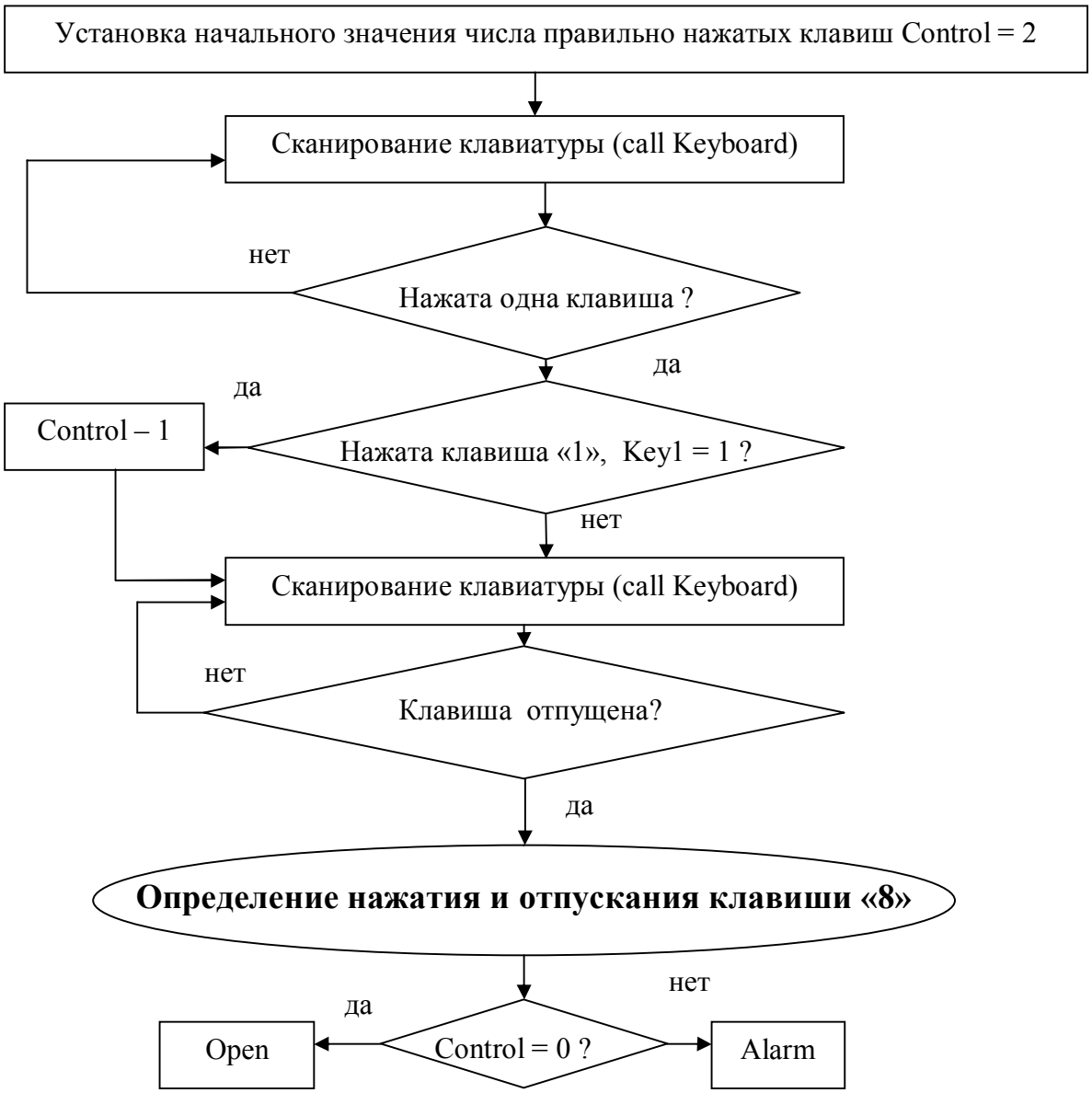


Рис. 98 Блок-схема алгоритма работы блока управления кодовым замком с матричной клавиатурой на PIC 16F877

Ниже приведен исходный код программы с комментариями.

```
                PROCESSOR PIC16F877     ; директива принудительной установки типа микроконтроллера

                #include <P16F877.inc>  ; подключение файла инструкций
fCOUNTER1       equ 0x21                ; введение вспомогательного регистра, используемого в работе процедуры задержки

Key1            equ 0x26                ; регистры Key1 – Key9 это регистры, в который записывается «1» если клавиша нажата, и «0» - если нет.
Key2            equ 0x27                Клавиши соответствуют регистрам:
Key3            equ 0x28                Клавиша 1 соответствует регистру
Key4            equ 0x29
Key5            equ 0x2a
```

```
Key6          equ 0x2b
Key7          equ 0x2c
Key8          equ 0x2d
Key9          equ 0x2e
NumPressKey   equ 0x2f
Control       equ 0x30
              errorlevel-302
              org   0x000
              clrf STATUS
              movlw 0x00
              movwf PCLATH
              goto Start
Start
              bsf STATUS, RP0
              movlw b'00000000'
              movwf TRISC
              movlw b'11111000'
              movwf TRISA
              movlw b'00000000'
              movwf TRISD
              movlw b'00001111'
              movwf OPTION_REG
              clrf ADCON1
              bsf ADCON1,0x01
              bsf ADCON1,0x02
              bcf STATUS, RP0
              clrf Key1 clrf Key2 clrf Key3
              clrf Key4 clrf Key5 clrf Key6
              clrf Key7 clrf Key8 clrf Key9
              clrf PORTC clrf PORTD
              clrf Control clrw
Begin
```

Key1, Клавиша 2 - Key2, Клавиша 3 - Key3, Клавиша 4 - Key4, Клавиша 5 - Key5, Клавиша 6 - Key6, Клавиша 7 - Key7, Клавиша 8 - Key8, Клавиша 9 - Key9

; регистр числа нажатых клавиш в **одном** цикле опроса клавиатуры (при одном вызове процедуры Keyboard)

; регистр числа **правильно нажатых клавиш** (две клавиши, которые должны быть нажаты в определенном порядке)

; управление сообщениями об ошибках

; директива указания адреса расположения первой команды ПО в памяти программ

; установка «нулевых» страниц памяти программ и данных микроконтроллера, принудительный переход к началу выполнения программы

; переход в 1-ый банк для конфигурации контактов

; конфигурирование контактов управления сиреной (RC1 - выход), замком (RD1 - выход), клавиатурой (RA0-RA2 – выходы, RA3-RA5 – входы)

; установка максимального коэффициента предварительного делителя, включенного перед сторожевым таймером WDT

; установка контактов порта A – как цифровые контакты ввода/вывода (см. рис. 41)

; переход в нулевой банк для обнуления регистров:

- Key (так как нажатия клавиш еще не было).
- PORTD – запирание замка
- PORTC – выключение сирены
- Control, W - обнуление для инициализации

; запись в регистр Control требую-

```
        movlw 0x02
        movwf Control
        call Keyboard
```
щегося числа правильно нажатых клавиш. По условию задачи контроллер отслеживает нажатие и отпускание 2-х клавиш: 1-ой и 8-ой. Вызов драйвера для определения состояния клавиатуры

```
        movf NumPressKey,1
        btfss STATUS,Z
```
; проверка нажата ли хоть одна клавиша осуществляется путем записи содержимого NumPressKey в самого себя. Эта команда влияет на бит Z, и в случае NumPressKey=0, установит бит Z в 1. В зависимости от этого, происходит либо продолжение ожидания нажатия (Z=1, переход на метку Begin), либо проверка одна ли клавиша нажата (при Z=0). Строго говоря, данная команда не является обязательной, т.к. далее идет дублирующая проверка, и приведена здесь в методических целях для демонстрации еще одного метода проверки нулевого события

```
        decfsz NumPressKey,w
```
; проверка условия NumPressKey – 1 = 0, с пропуском следующей команды в этом случае, т.к. это означает, что нажата только одна клавиша.

```
        goto Begin
```
; возврат к ожиданию нажатия одной клавиши, осуществляемый, если NumPressKey ≠ 1

```
        btfsc Key1,0
```
; эта команда выполняется, если NumPressKey = 1, и проверяет была ли нажата клавиша «1», проверка осуществляется по младшему биту регистра Key1

```
        decf Control, 1
```
; если это так, то содержимое регистра Control уменьшается на 1, показывая, что нажата первая правильная клавиша кодового замка

```
Down1   call Keyboard
```
; если же нажата не клавиша «1», вновь вызывается драйвер клавиатуры для отслеживания события отпускания клавиши, и ставится метка Down для зацикливания ожидания отпускания клавиши

```
        movf NumPressKey,1
        btfss STATUS,Z
```
; проверка отпустили ли нажатую клавишу, т.к. в этом случае будет NumPressKey = 0

```
        goto Down1
```
; зацикливание ожидания отпуска

```
                                    клавиши
n2          call Keyboard           ; если кнопка отпущена, переходим
                                    к сканированию клавиатуры с целью
                                    определения второй нажатой кла-
                                    виши, для этого вновь вызывается
                                    драйвер
            movf NumPressKey,1      ; аналогично изложенному выше,
            btfss STATUS,Z          определяется нажата ли, и если на-
            decfsz NumPressKey,w    жата, то одна ли клавиша, с зацик-
            goto n2                 ливанием этого ожидания, если на-
                                    жата не одна клавиша
            btfsc Key8,0            ; переход сюда осуществляется, если
            decf Control,1          нажата одна клавиша. После пере-
Down2       call Keyboard           хода, проверяется является ли нажа-
            movf NumPressKey,1      тая клавиша клавишей «8» по со-
            btfss STATUS,Z          стоянию Key8=1. Если да, то ре-
            goto Down2              гистр Control уменьшается на 1 и
                                    становится равным 0, говоря о пра-
                                    вильном наборе двух клавиш, и вы-
                                    зывается драйвер для отслеживания
                                    отпускания клавиши. Если – нет,
                                    драйвер также вызывается, но без
                                    уменьшения регистра. В обоих слу-
                                    чаях после обнаружения, что кла-
                                    виша отпущена, переходим к про-
                                    верке числа в регистре Control
            movf Control,1          ; проверка Control=0, если да то сра-
            btfss STATUS,Z          батывает процедура открывания
                                    замка Open, если – нет, то - вклю-
                                    чения сирены Alarm, свидетельст-
                                    вующая о двух неправильно на-
                                    бранных клавишах
            goto Alarm              ; вызов процедур открывания замка
            goto Open               и срабатывания сирены
Open        bsf PORTD,0x01          ; процедура открывания замка с за-
            clrwdt                  цикливанием
            goto Open
Alarm       bsf PORTC,0x01          ; процедура бесконечного включе-
            clrwdt                  ния сирены
            goto Alarm
small_delay:                        ; процедура задержки длительно-
            movwf fCOUNTER1         стью ( (fCOUNTER1 * 4) + 6 ) мкс
            clrwdt                  для защиты от дребезга. Вызов этой
            decfsz fCOUNTER1,F      процедуры осуществляется в теле
            goto $-2                драйвера
            return

            end                     ; конец программы
```