Министерство образования и науки Украины
Донбасская государственная машиностроительная академия
Кафедра АПП

Е. И. Донченко

ПРОЕКТИРОВАНИЕ ВСТРАИВАЕМЫХ МИКРОКОНТРОЛЛЕРОВ

Конспект лекций

(для студентов специальности 151
дневной и заочной форм обучения)

Краматорск 2018

УДК 681.3.06 (074.8)

Конспект лекций по дисциплине «Проектирование встраиваемых микроконтроллеров» (для студентов специальности 151 дневной и заочной форм обучения) / Сост. Е. И. Донченко. – Краматорск: ДГМА, 2018. – 96 с.

Рассматриваются особенности программирования и применения встраиваемых микроконтроллеров фирмы Philips. Приведены данные для самостоятельного изучения и практического освоения курса «Программирование встраиваемых микроконтроллеров»

Составитель Е. И. Донченко, ст. преп.

Отв. за выпуск Е. И. Донченко, ст. преп.

1 ТРАДИЦИОННЫЕ ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА

1.1 Кнопки и датчики

Типовая схема подключения микроконтроллера AT89C2051 показана на рисунке 1.1.

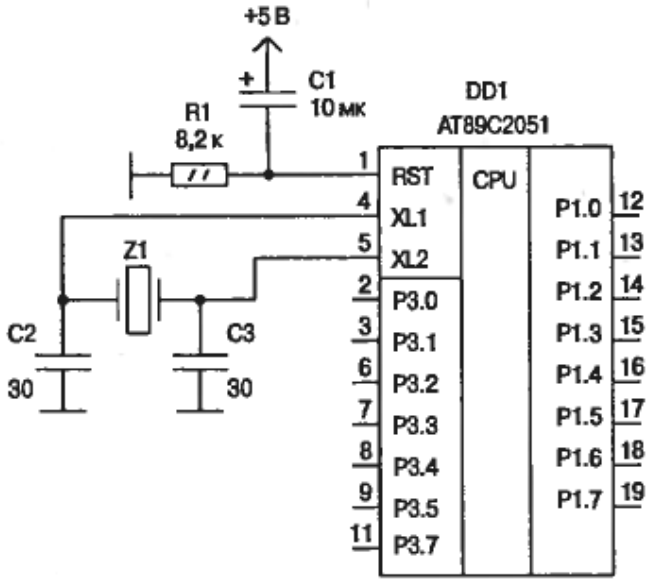


Рисунок 1.1 - Типовая схема подключения микроконтроллера AT89C2051

Практически ни одно микропроцессорное устройство не обходится без кнопок и простейших датчиков на основе обычных контактов. При помощи этого вида периферийных элементов в микропроцессорное устройство поступает различная информация, которая используется для изменения алгоритма работы программы.

Примером может служить датчик поворота. Допустим, наша микропроцессорная система должна управлять поворотом некоего поворотного устройства. Для отслеживания реального угла поворота нам понадобится датчик, при помощи которого микроконтроллер сможет определить этот угол. Самый простой способ построения такого датчика — это механические контакты, связанные с поворачиваемым устройством. Проще всего на валу устройства укрепить постоянный магнит, а на неподвижной ее части расположить геркон. При вращении вала укрепленный на нем магнит будет проходить рядом с герконом и вызывать срабатывание его контактов. Однако

с таким же успехом подобную схему можно подключить и к любой другой линии любого из двух портов микроконтроллера (см. рис. 1.1).

Схема подключения такого датчика к микроконтроллеру приведена на рис. 1.2.

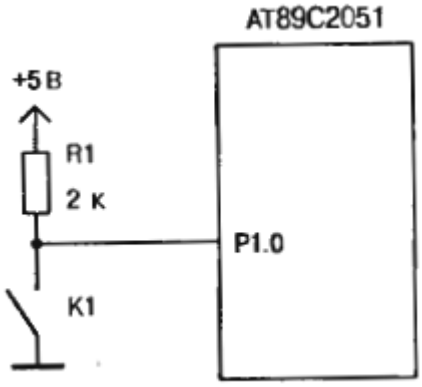


Рисунок 1.2 - Простая схема подключения датчика на основе геркона

Датчик представляет собой свободно разомкнутые контакты геркона. В приведенном примере датчик подключен к линии P1.0 порта PI микроконтроллера. Через этот вход микропроцессор производит считывание состояния датчика. Однако с таким же успехом подобную схему можно подключить и к любой другой линии любого из двух портов микроконтроллера.

Принцип работы схемы, приведенной на рис. 1.2, очень простой. В исходном состоянии контакты датчика разомкнуты. На вход микроконтроллера через резистор Rl подается напряжение от источника питания +5 В. Микросхема воспринимает это напряжение как сигнал логической единицы. При срабатывании датчика контакты замыкаются и соединяют вывод микроконтроллера с общим проводом. В результате напряжение на входе P1.0 падает до нуля. Теперь микросхема воспринимает входной уровень сигнала как логический ноль. Резистор Rl при этом служит токоограничивающим элементом, предотвращая короткое замыкание между шиной питания и общим проводом. Наличие резистора Rl обязательно лишь в том случае, если датчик подключается к линии P1.0 или P1.1. Остальные линии имеют свои внутренние резисторы нагрузки, которые могут с успехом заменить внешний резистор.

Схема, приведенная на рис. 1.2, универсальна и широко применяется как для подключения простейших контактных датчиков, так и для подключения различных кнопок управления. Любое реальное

микропроцессорное устройство редко обходится без кнопок управления. При помощи таких кнопок могут переключаться режимы работы вашего устройства. Например, кнопки «Пуск» и «Стоп» могут быть использованы для запуска и останова любого процесса. Если вам нужно иметь несколько кнопок управления, вы можете подключить их к разным входам микроконтроллера. При этом будут одновременно работать несколько кнопок. Если количество кнопок не слишком велико, то данный способ их включения — самый рациональный. Однако, если требуется
большое количество управляющих кнопок, то вам просто может не хватить имеющихся
выводов. В этом случае не обойтись без матрицы клавиш. На рис. 1.3 приведена схема подключения клавиатуры из 32 клавиш путем составления из них матрицы.

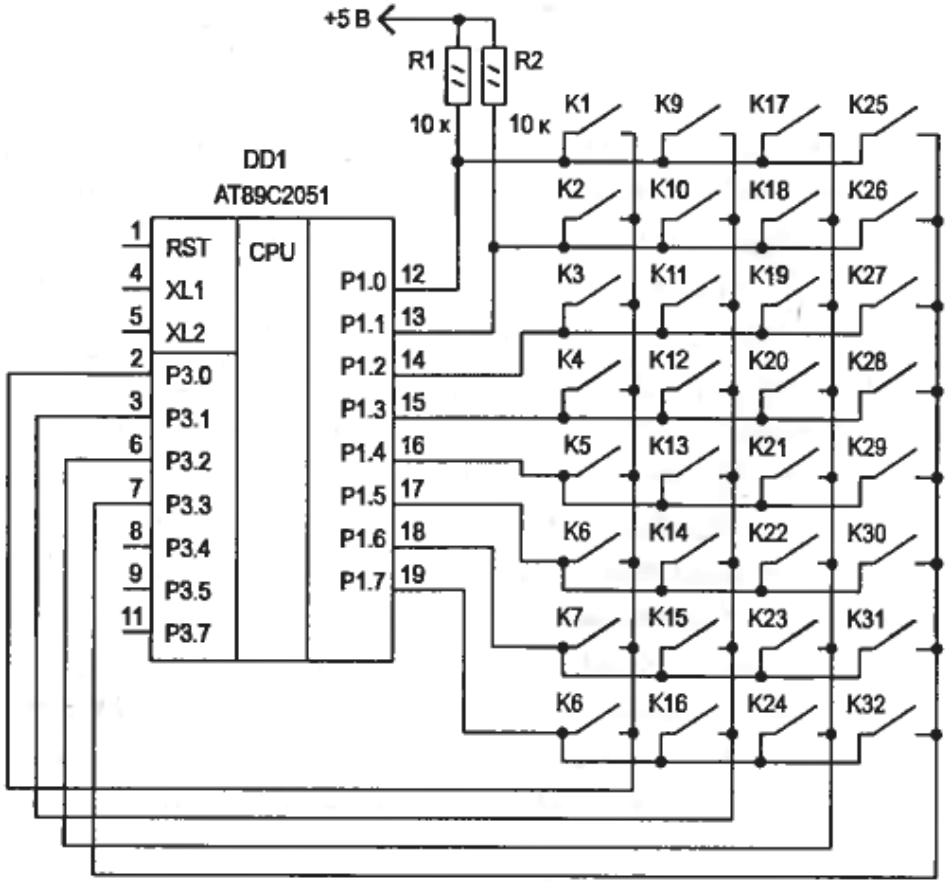


Рисунок 1.3 – Схема подключения клавиатуры в виде матрицы клавиш

Напоминаю, что цепи сброса и цепи, связанные с кварцевым резонатором, на этой схеме, как и в большинстве последующих примеров, условно не показаны. Как видно из схемы, для подключения тридцати двух клавиш используется всего лишь 12 выводов. В данном конкретном случае порт Р1 работает как порт ввода. Четыре младших разряда порта РЗ работают

на вывод. Возможен и обратный вариант (P1 на вывод, P3 на ввод). Но мы выбрали тот вариант, который наиболее рационален с точки зрения удобства составления программы. Давайте рассмотрим подробнее принцип работы схемы.

В исходном состоянии на выводы P3.0...P3.3 подается сигнал логической единицы. На все выводы порта P1 (P1.0...P1.7) также поданы единицы. Но во втором случае единицы поданы для того, чтобы обеспечить возможность работы линий в режиме ввода. Контроллер периодически опрашивает состояние клавиш путем изменения сигналов на выходах P3.0...P3.3 и считывания сигнала из порта P1. В случае обнаружения замыкания контактов одной из клавиш, программа выполняет закрепленные за этой клавишей действия. В случае, если при опросе клавиатуры обнаружится несколько одновременно нажатых клавиш, это считается ошибкой и никаких действий не выполняется. Если ни одна клавиша не нажата, никаких действий также не выполняется. Такой алгоритм используется в большинстве современных клавиатур.

Каким же образом осуществляется опрос клавиш? Процедура опроса клавиш поочередно переводит одну из линий P3.0...P3.3 в нулевое состояние. Сначала в нулевое состояние переводится линия P3.0. Сразу после этого контроллер производит чтение числа из порта P1. Если ни одна из клавиш не нажата, то все разряды считанного числа будут равны единице (считанное число будет равно 0FFH). Если хотя бы одна из клавиш К1...К8 окажется нажатой, то число, прочитанное из порта P1, будет отличаться от значения 0FFH. Предположим, что мы нажали клавишу К1. Тогда сигнал логического нуля с выхода P3.0 поступит на вход P1.0 и младший разряд считанного числа окажется равным нулю. В этом случае процессор из порта P1 прочитает 0FEH. Нажатие любых других клавиш этой вертикали (К2...К8) приведет к обнулению других разрядов считываемого числа. В результате для разных комбинаций клавиш мы получим разные коды.

Аналогичным образом микроконтроллер опрашивает остальные вертикали клавиш. То есть, на следующем этапе, процессор выставляет ноль на линию P3.1, а линию P3.0 возвращает в единичное состояние. После этого считанное из порта P1 число будет определять состояние клавиш К9...К16. Затем ноль выставляется на линию P3.2 и проверяется состояние клавиш К17...К24. И в заключение ноль выставляется на линию P3.3 и проверяется состояние клавиш К25...К32. Анализируя полученные при этом коды можно вычислить номер нажатой клавиши.

Посмотрим, как это делается. Для обслуживания клавиатуры, изображенной на рис. 1.2, была разработана процедура klav, текст которой вы

можете видеть в листинге 1.2. Это всего лишь один из возможных вариантов реализации подобной процедуры. Процедура оформлена в виде отдельной подпрограммы, к которой при необходимости обращается основная программа микроконтроллера. При каждом обращении подпрограмма klav производит однократное сканирование всей клавиатуры и возвращает код нажатой клавиши. Возвращаемый код помещается в аккумулятор (то есть регистр a). Если в момент сканирования клавиатуры ни одна клавиша не была нажата, то возвращаемый код состояния будет равен нулю. Если нажато сразу несколько клавиш, программа также возвращает ноль. И только в том случае, если нажата всего одна клавиша, то возвращаемый подпрограммой код будет равен номеру этой клавиши. Описанный выше алгоритм является стандартным для клавиатуры, состоящей из матрицы клавиш.

Листинг 1.2

```
1 $mod2051
   :---------------------- Определение констант
2  bank0        EQU    00000000H     : Коды банков памяти
3  bank1        EQU    00001000B
4  bank2        EQU    00010000B
5  bank3        EQU    00011000B
    ----------------------Резервирование ячеек памяти
6               DSEG
7               ORG    20H           Начинаем резервирование с адреса  20H
8  p3bcf:       DS     1             Буфер порта  p3
   :----------------------Начало программного кода
9               CSEG
10              ORG    00H           Начинаем программный код с адреса  00H
11 ------------------------------------------------------------------------------------------
   В этом месте вы должны поместить основной текст вашей программы
   : ########################################
   : ##  Подпрограмма опроса клавиатуры   ##
   : ########################################
12 klav:        push   psw           : Сохранение регистров флага в стеке
13              mov    psw, #bank3   : Переключение на банк3  регистров РОН
14              mov    p1, #0FFH     : Перевод  P0.1 – P1.7  в единичное состояние
15              call   clrU          : Перевод  P3.0 – P3.3  в единичное состояние
16              mov    r0, #0        : Очистка буфера кода клавиши
17              mov    r1, #4        : Инициализация счетчика столбцов
18              mov    r2,#0FEH      : Код сканирование первого столбца
19 kl1          call   setU          : Вывод кода в порт P3
20              mov    a, p1         : Считывание состояния клавиатуры
21              cjne   a, #0FFH,kl3  : Если клавиша нажата,перехлдим к  kl3
         :---------Переход к следующему столбцу
22 kl2          mov    a, r2         : Извлечение кода
23              r1     a             : Сдвиг
24              mov    r2, a         : Записать назад в буфер
25              djnZ   r1, kl1       : Команда цикла опроса столбцов
```

```
        :---------Окончание процедуры опроса клавиатуры
26 klfin    call    clrU            : Переход в исхожное состояние порта P3
27          mov     a,r0            : Запись в аккумулятор кода клавиши
28          pop     psw             : Восстановление регистра флагов
29          ret                     : Выход из подпрограммы
        :---------Нахождение номера клавиши в строке
30 kl3      mov     r3, #1          : Инициализация счетчика строк
31 kl4      setb    c               : Обнуление признака переноса
32          rrc     a               : Сдвиг входного кода
33          jnc     kl5             : Если нашли разряд, равный нулю, перейти к  kl5
34          inc     r3              : Итерация счетчика строк
35          jmp     kl4             : Переход к началу цикла счета строк
36 kl5      cjne    a, #0FFH, kl6   : Если есть еще хоть один ноль, перейти к kl6
37          mov     a, r0           : Не найдена ли уже другая нажатая кнопка
38          jnz     kl6             :
        :---------Вычисление номера клавиши
39          mov     a,#4            : Запись константы в аккумулятор
40          clr     c               : Очистка признака переноса
41          subb    a, r1           : Вычисление номера столбца
42          r1      a               : Три оператора сдвига
43          r1      a
44          r1      a
45          add     a, r3           : Прибавление номера строки
46          mov     ro, a           : Записываем в буфер
47          jmp     kl2             : Продолжаем поиск по остальным столбцам
48 kl6      mov     r0, #0          : Возвратить ноль
49          jmp     klfin           : Перейти на конец процедуры
        :---------Сброс разрядов столбцов
50 clrU     mov     a, p3buf        : Считывание содеержимого буфера порта P3
51          orl     a, #0FH         : Перевод разрядов P3.0 – P3.3 в единичное состояние
52          mov     p3buf, a        : Запись результата назад в буфер
53          mov     p3, p3buf       : Вывод содержимого буфера в порт P3
54          ret
        :---------Вывод столбцов
55 setU     a3, p3buf               : Считывание содержимого буфера порта P3
56          orl     a, #0FH         : Перевод разрядов P3.0 – P3.3 в единицу
57          anl     a, r2           : Вывод в P3.0 – P3.3  кода сканирования
58          mov     p3buf, a        : Запись результата назад в буфер
59          mov     p3, p3buf       : Вывод содержимого буфера в порт P3
60          ret
        :-------------------------------------------------------------------------------------
        Сюда вы можете поместить другие подпрограммы
        :-------------------------------------------------------------------------------------
61  end
```

Сразу после команд инициализации начинается главный цикл сканирования. Тело цикла составляют строки 19...25. Цикл начинается с вызова вспомогательной подпрограммы setU. Эта подпрограмма производит вывод четырех младших разрядов регистра r2 в порт P3, не затрагивая при этом

четыре старших разряда. Для выполнения этих действий тоже используется буфер p3buf. Сначала значение буфера извлекается в аккумулятор (строка 55). Затем производится установка старших разрядов в единицу (строка 56). Теперь, когда старое значение этих разрядов сброшено, можно производить операцию объединения. В строке 57 производится операция «И» между содержимым аккумулятора и содержимым регистра r2. В результате таких вычислений мы получаем в аккумуляторе число, четыре младших разряда которого равны младшим разрядам регистра r2, а старшие разряды остались без изменения. Это число мы, во-первых, записываем обратно в буфер (строка 58), а во-вторых, выводим в порт P3 (строка 59). Итак, сигнал опроса ряда установлен. Теперь программа производит считывание порта P1 (строка 20). Если в опрашиваемом столбце ни одна клавиша не нажата, то считанное таким образом число будет равно 0FFH. Если же хотя бы одна клавиша окажется нажатой, то считанное число будет отличаться от 0FFH. В строке 21 происходит проверка считанного числа. Если клавиша нажата, то управление передается по метке И3. Здесь определяется номер строки клавиатуры, где произошло замыкание клавиши. Если замыканий не обнаружено, то эта часть алгоритма пропускается и программа продолжает опрос столбцов. Для перехода к следующему столбцу производится сдвиг кода в регистре r2 (строки 22...24). Оператор djnz в строке 25 — последний оператор в цикле опроса столбцов клавиатуры. Он уменьшает содержимое регистра r1 на единицу и обеспечивает переход в начало цикла (метка kl1), если содержимое регистра не достигло нуля. Таким образом, опрашиваются все четыре столбца. Когда все столбцы опрошены, цикл завершается. Программа выходит на финишную прямую. Младшие разряды порта P3 переводятся в исходное состояние (строка 26). Код нажатой клавиши помещается в аккумулятор (строка 27). Восстанавливается регистр psw (строка 28), и, наконец, происходит выход из подпрограммы klav (строка 29).

В процессе описания подпрограммы мы пропустили одну ее часть. Ту часть, где происходит вычисление кода нажатой клавиши. Вернемся и рассмотрим ее подробнее. Вычисление кода нажатой клавиши начинается с метки W3 и занимает строки с 30 по 49. Как вы помните, в это место программа переходит, если при опросе очередного столбца считанный из порта P1 код не равен 0FFH. Этот код мы будем называть кодом столбца. Если код столбца не равен 0FFH, то тут возможны два варианта. Первый вариант — в опрашиваемом столбце нажата всего одна клавиша.

Второй вариант — в одном столбце нажато сразу несколько клавиш. Если нажатых клавиш несколько, дальнейший перебор клавиш не имеет смысла. Программа должна завершиться досрочно и возвратить ноль в

аккумуляторе. Если на данном этапе обнаружится всего одна нажатая клавиша, то заканчивать перебор еще рано. В этом столбце одна, а в других? Это мы узнаем в процессе дальнейшей проверки. А сейчас нам нужно вычислить номер нажатой клавиши, записать его в буфер кода клавиатуры (регистр r0) и продолжить процесс опроса клавиатуры.

В дальнейшем содержимое регистра r0 можно использовать для проверки факта одновременного нажатия двух клавиш в разных столбцах. Обнаружив нажатую клавишу, нужно сразу проверить r0. Если содержимое этого регистра не равно нулю, то нажатая клавиша не единственная.

Посмотрим теперь как это делается на практике. Для начала рассмотрим процесс определения номера строки, где нажата клавиша. Этот номер определяется путем многократного сдвига кода столбца. Сдвиг выполняется при помощи оператора rrc (строка 32). Сдвиг происходит до тех пор, пока ноль от нажатой клавиши не достигнет ячейки CY. Количество циклов сдвига, которые потребуются для этого и равно номеру строки, где произошло нажатие клавиши. Если в столбце была нажата лишь одна клавиша, то после всех этих циклов сдвига во всех оставшихся разрядах аккумулятора останутся лишь единицы. То есть код в аккумуляторе будет равен 0FFH. Если это не так, то это значит, что в одном столбце нажато сразу несколько клавиш.

Рассмотрим, как происходит весь этот процесс в программе. Перед тем как начинать сдвигать аккумулятор, признак переноса устанавливается в единицу (строка 31). Если там оставить ноль, то при сдвиге он попадет в аккумулятор и сделает невозможным обнаружение лишних нажатий. Тело цикла сдвига занимает строки 31...35. Перед началом цикла устанавливается начальное значение для счетчика строк (строка 30). В качестве счетчика используется регистр r3. После каждого сдвига проверяется значение ячейки CY (строка 33). Как только признак окажется равным нулю, оператор jnc, находящийся в этой строке, передаст управление по метке kl5 и цикл на этом завершится.

Подсчет строк производится в строке 34. После каждого цикла сдвига счетчик строк увеличивается на единицу. В строке 35 расположен оператор, организующий замыкание цикла. Это оператор безусловного перехода. В другом случае применение оператора безусловного перехода могло бы привести к образованию бесконечного цикла и зависанию программы. Но в нашем случае применение этого оператора вполне оправдано. Так как точно известно, что, как минимум, один из разрядов сдвигаемого числа обязательно будет равен нулю. Если это было бы не так, то управление вообще бы не перешло в эту часть программы. Предварительная проверка содержимого

аккумулятора производится в строке 21. После завершения всех циклов сдвига в регистре r3 будет находиться номер строки, где обнаружено нажатие клавиши. Но прежде чем записывать его в буфер (r0), нужно произвести две проверки для выявления двойного нажатия. Сначала проверяется содержимое аккумулятора (строка 36). После сдвига там должно быть 0FFH. Если это не так, то управление передается по метке kl6, где производится досрочное завершение подпрограммы с нулевым результатом.

Следующая проверка — это проверка буфера клавиатуры. Если там не ноль, то это значит, что в одном из предыдущих столбцов уже была обнаружена нажатая клавиша. Для проверки содержимое r0 помещается в аккумулятор (строка 37). Затем, в строке 38, оно проверяется на равенство нулю. Для этого используется оператор jnz. Если аккумулятор не равен нулю, то управление также передается по метке kl6 и подпрограмма завершается с нулевым результатом.

Если подпрограмма провела все описанные выше проверки и определила, что мы имеем дело с единственной нажатой клавишей, она должна вычислить код этой клавиши и записать его в r0. Вычисление кода происходит в строках 39...47. Код вычисляется по простой формуле: номер столбца умножается на восемь и к полученному результату прибавляется номер строки. Номера столбца в чистом виде у нас нет. Однако в регистре r1 содержится счетчик циклов опроса столбцов. Он содержит обратную величину. В самом начале его значение равно четырем. После опроса каждого очередного столбца содержимое счетчика уменьшается на единицу.

Очевидно, что текущий номер столбца легко найти, если содержимое регистра r1 вычесть из четырех. Это арифметическое действие выполняется в строках 39...41. Сначала в аккумулятор помещается значение, равное 4 (строка 39). Для вычитания используется оператор subb (строка 41). Оператор subb производит вычитание с учетом признака переноса. Поэтому в строке 40 признак переноса обнуляется. После выполнения команды subb мы получим номер столбца, который будет находиться в аккумуляторе. Далее номер столбца умножается на 8. Для этого используется метод тройного сдвига (строки 42...44). Каждый сдвиг двоичного числа вправо эквивалентен операции умножения на 2. После трех сдвигов содержимое аккумулятора увеличится в восемь раз.

Далее к полученному числу прибавляется номер строки (строка 45). Вычисленный таким образом номер нажатой клавиши записывается в буфер клавиатуры — регистр r0 (строка 46). На этом завершается процесс вычисления кода нажатой клавиши, но не заканчивается процесс

сканирования клавиатуры. Для продолжения этого процесса управление передается по метке W2 (строка 47).

Описанную только что схему (рис. 1.2) можно усовершенствовать. Добавив всего один дешифратор, мы можем сэкономить две линии порта P3. Схема клавиатуры с дешифратором приведена на рисунке 1.4.

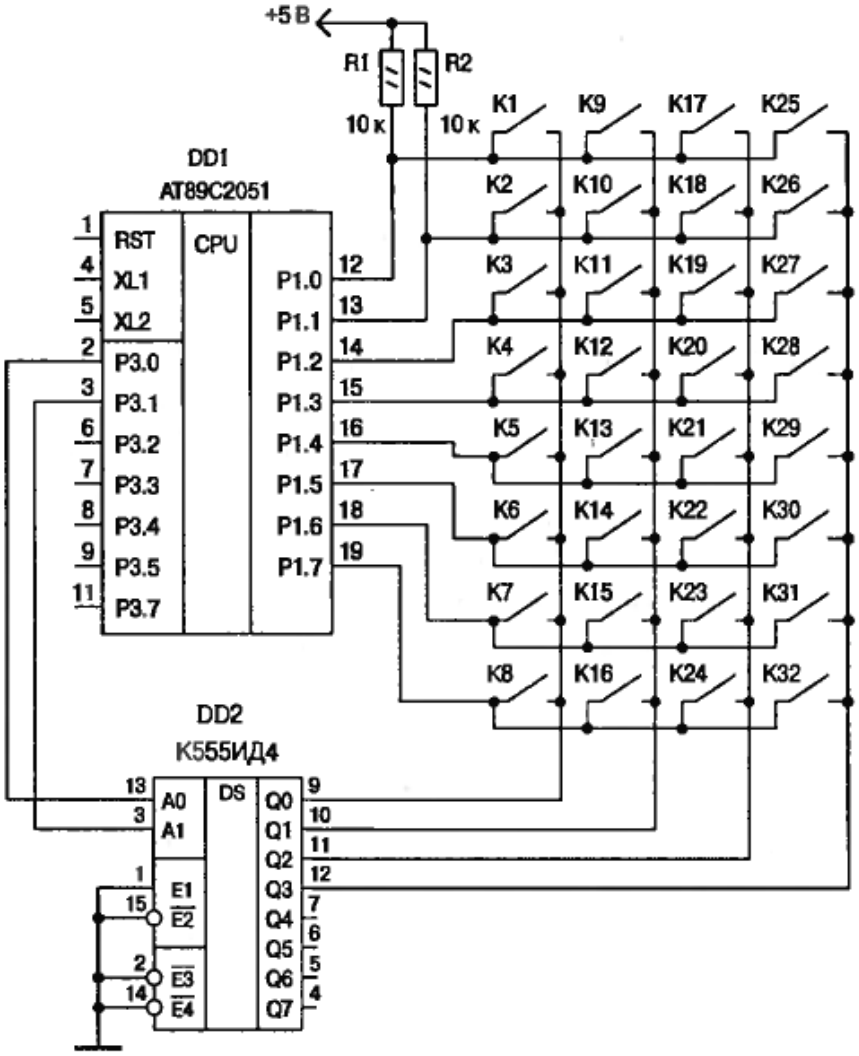


Рисунок 1.4 – Второй вариант подключения матрицы клавиш

В этой схеме для выбора одного из четырех столбцов авиатуры используется дешифратор DD2 типа К555ИД4. В такой схеме для сканирования столбцов микроконтроллер должен одавать на выходы P3.0 и P3.1 двухразрядный двоичный код, авный номеру столбца. Код поступает на входы AO и A1 дешифратора. В результате один из его выходов (тот, номер которого оответствует поступившему коду) примет нулевое значение. На стальных же выходах будет единица. Так, при коде 00B на входе ешифратора выход Q0 (вывод 9) принимает нулевое значение, ри коде 01B — ноль будет на выходе Q1. И так далее. Таким бразом, микроконтроллер может

перебирать все четыре столбца, спользуя всего два разряда. В остальном алгоритм работы новой схемы ничем не отличается от предыдущей (рис. 1.2).

В связи с изменением схемы немного изменится и программа. Изменения коснулись лишь процедуры опроса столбцов. Теперь код будет формироваться другим способом. В остальном новый вариант программы будет почти полностью совпадать с предыдущим вариантом.

1.2 Световые индикаторные устройства

Практически любое микропроцессорное устройство содержит элементы световой индикации. В качестве световых индикаторов в настоящее время чаще всего применяются светодиоды. На рынке имеется огромный выбор светодиодов, самых разных видов и размеров. Легко можно купить светодиоды повышенной яркости, мигающие, двух- и даже трехцветные. Светодиоды выделяются среди других светоизлучательных элементов благодаря особой экономичности и долговечности.

Если нужно подключить индикатор большого размера или повышенной яркости, который потребует большего рабочего тока, придется применить буферный элемент. Одним из вариантов буферного элемента является электронный ключ. Схема подключения светодиода при помощи электронного ключа приведена на рис. 1.6.

Основой электронного ключа служит транзистор VT1. Резистор R1 — токоограничивающий в цепи базы. Резистор R2 служит для надежного запирания ключа при нулевом сигнале на выходе P3.0. Резистор R3 ограничивает рабочий ток светодиода. Его номинал может изменяться в зависимости от типа применяемого светодиода. Если в качестве VTI использовать транзистор КТ315, схема способна обеспечивать рабочий ток светодиода до 250 мА.

Кроме максимально допустимого тока нагрузки, обе приведенные выше схемы отличаются способом управления. Схема, изображенная на рис. 1.5, с инверсным способом управления.

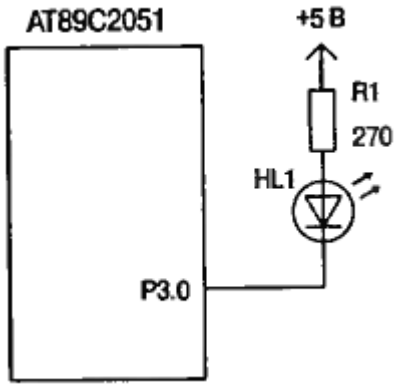


Рисунок 1.5 – Простейшая схема подключения светодиодного индикатора

Для того, чтобы зажечь светодиод, нужно подать на выход P3.0 низкий логический уровень. Схема, изображенная на рис. 1.6, обеспечивает прямое управление. Для того, чтобы зажечь светодиод, включенный по этой схеме, на выход P3.0 нужно подать сигнал логической единицы.

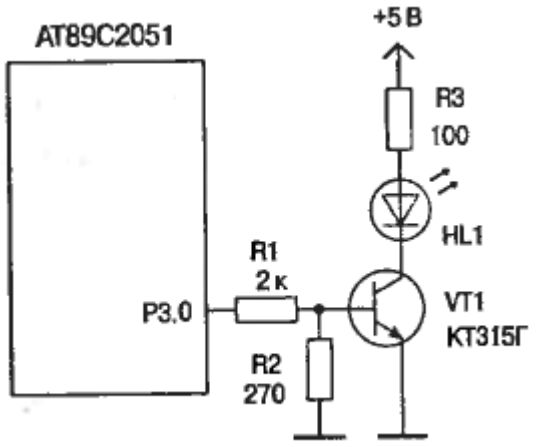


Рисунок 1.6 - Схема подключения светодиода посредством электронного ключа

С программной точки зрения управление светодиодом, включенным по любой из двух вышеприведенных схем, не представляет никаких трудностей. Например, для того, чтобы установить линию P3.0 в единичное состояние, процессор должен просто выполнить команду:

setb P3.0 *// перевод бита P3.0 в единичное состояние*

При этом в первом случае (рис. 1.5) светодиод потухнет, а во втором (рис. 1.6) — загорится.

В микропроцессорных устройствах световые индикаторы могут служить для отображения различных режимов работы: предупреждать о критических ситуациях, отражать ход приема управляющих сигналов. Подключить одиночный светодиодный индикатор к микропроцессору очень просто. На рис. 1.5 приведена схема подключения светодиода непосредственно к выводу порта процессора AT89C2051.

Все выходные каскады микроконтроллера имеют достаточную нагрузочную способность для того, чтобы выдержать подключение одного светодиодного индикатора. Тут имеется в виду светодиод среднего и миниатюрного размера с током потребления в рабочем режиме не больше 20 мА. Если вы желаете подключить по такой схеме несколько светодиодов на несколько разных выходов микроконтроллера, то суммарный ток, проходящий через все эти выходы, не должен превышать 80 мА.

Кроме одиночных световых индикаторов в микропроцессорных устройствах часто применяют знакосинтезируюшис матрицы, которые еще называют цифровыми индикаторами. Простейшим примером цифрового индикатора может служить так называемый ссмисегмснтный индикатор. Вам наверняка хорошо знаком такой тип индикаторов. Он широко применяется в самых разных устройствах цифровой техники: от калькулятора, до электронных часов. Семисегментный индикатор представляет собой матрицу из семи светодиодов продолговатой формы, размешенных таким образом, чтобы, зажигая их в разных сочетаниях, можно было бы отобразить любую десятичную цифру (имеются в виду арабские цифры от 0 до 9). Кроме семи основных сегментов индикатор чаше всего дополняют восьмым маленьким сегментом, который предназначен для отображения десятичной точки (запятой). Расположив в рад несколько таких индикаторов, можно отображать любое десятичное число с плавающей запятой.

На рис. 1.7 изображен внешний вид семисегментного индикатора. В цифровой технике принято каждый сегмент индикатора обозначать буквой латинского алфавита так, как это показано на рисунке.

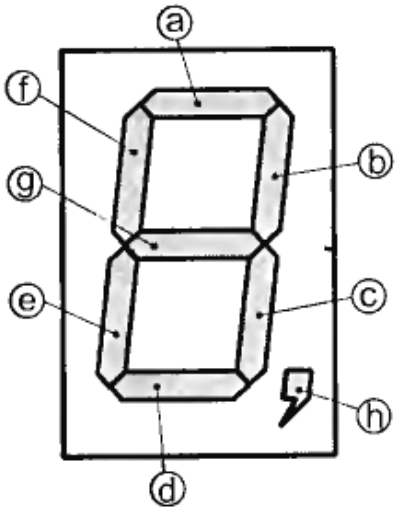


Рисунок 1.7 - Сегментный цифровой индикатор

Такой индикатор обычно выполняется в виде отдельного самостоятельного компонента и имеет 9 выводов. По внутренней схеме ключения семисегментные индикаторы подразделяются на индикаторы с общим анодом и индикаторы с общим катодом.

Схемы обоих видов индикаторов приведены на рис. 1.8 и рис. 1.9, соответственно.

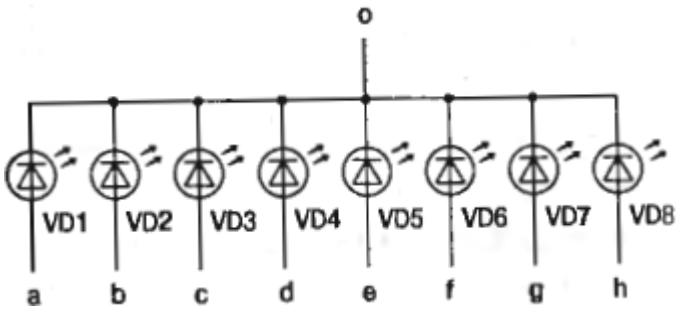


Рисунок 1.8 – Схема индикатора с общим анодом

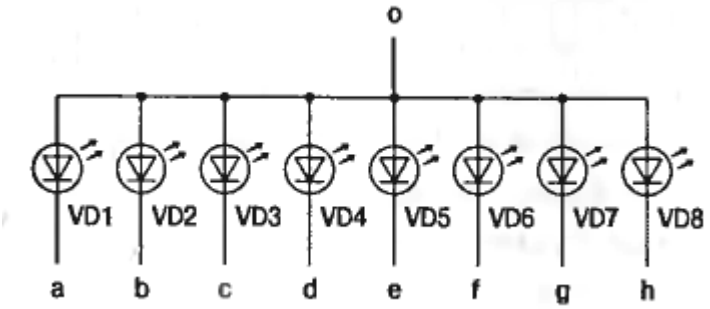


Рисунок 1.9 – Схема индикатора с общим катодом

В первом случае на общий провод подается плюс источника питания, а во втором - минус. Иногда семисегментный индикатор имеет 10 выводов (общий вывод дублируется). Каждый из восьми светодиодов семисегментного индикатора можно подключать к выводам микропроцессора по любой из двух схем, приведенных в начале этого раздела (рис. 1.5 или рис. 1.6). Например, индикатор с общим анодом можно включить так, как показано на рис. 1.10. В этой схеме использовано непосредственное подключение к выходам микропроцессора.

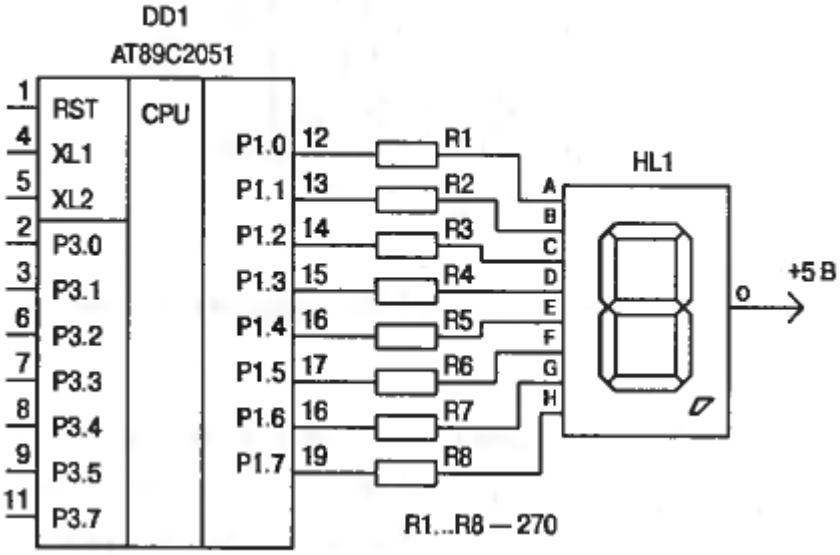


Рисунок 1.10 – Подключение сегментного индикатора

Так же, как и для отдельных светодиодов, существует широкая гамма различных модификаций семисегментных индикаторов. Они отличаются друг от друга размерами, цветом свечения, яркостью, расположением выводов. Существуют двухцветные семисегментные индикаторы, много-разрядные матрицы и так далее. Применение простейших цифровых светодиодных индикаторов — самый недорогой способ заставить ваше микропроцессорное устройство отображать цифры.

Для отображения цифровых данных одного цифрового индикатора обычно недостаточно. В таких случаях к микроконтроллеру подключают сразу несколько таких индикаторов. Однако, из-за отсутствия достаточного количества выводов у процессора приходится применять специальные ухищрения. На рис. 1.11 изображена типичная схема подключения четырех семисегментных индикаторов к микроконтроллеру. Как видно из рисунка, индикаторы, включенные по такой схеме, не могут работать все одновременно. Если их включить одновременно, то все они будут

отображать одно и то же. Схема, изображенная на рис. 1.11, предназначена для работы индикаторов в режиме динамической индикации.

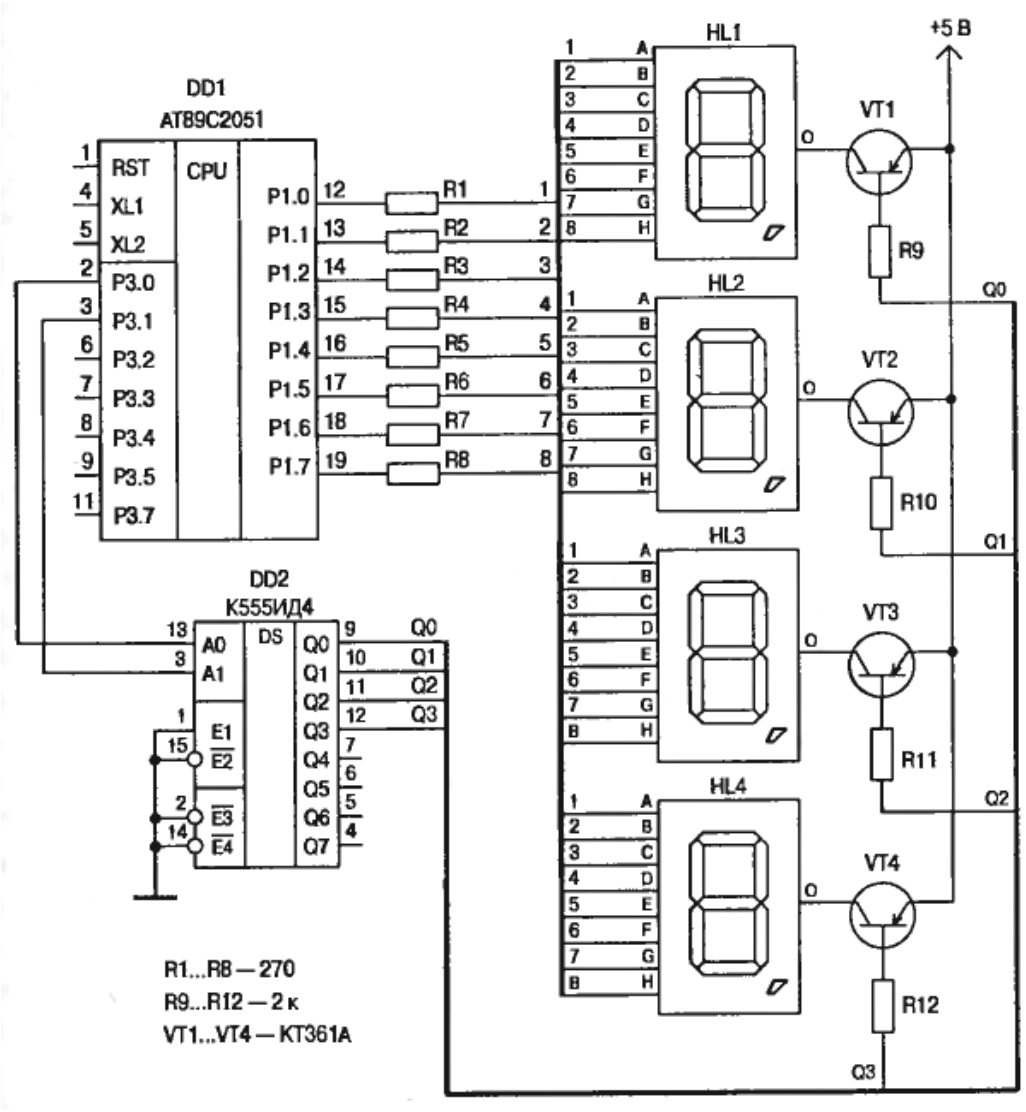


Рисунок 1.11 – Схема включения четырехсегментных индикаторов с динамической индикацией

Режим динамической индикации часто применяется для многоразрядных цифровых индикаторов. Он состоит в том, что разряды индикатора работают не одновременно, а по очереди. Переключение разрядов происходит с большой скоростью. Если скорость переключения разрядов достаточно велика, то человеческий глаз не замечает того, что разряды горят по очереди. Мерцания сливаются в статическую картинку, и человек видит цифры одновременно во всех разрядах. Подобный эффект используется в кино и в телевидении. Там, сменяющиеся с большой скоростью кадры сливаются в одно непрерывное изображение. Ученые давно

определили тот предел скорости смены кадров, при которой глаз человека уже не различает дискретности. Этот предел равен 24 кадрам в секунду. Если кадры на экране менять с этой скоростью, то они сливаются в плавно движущееся изображение. Однако для глаза такое изображение не будет комфортным. Мы не увидим дерганья при смене кадров, но просмотру будет мешать неприятное мерцание. Эффект мерцания возникает от того, что в промежутках между кадрами экран темный, а в момент отображения кадра экран ярко освещен. Глаз способен воспринимать такое изменение общей яркости и это мерцание неприятно для зрения. Для того, чтобы мерцание было незаметно, частота смены яркости экрана должна быть, как минимум, в два раза выше. То есть должна быть не менее 48 Гц. В кино это достигается введением дополнительного затемнения экрана посреди каждого кадра. В телевидении для устранения этого эффекта придумана чересстрочная развертка. Каждый кадр выводится в два приема: сначала нечетные строки, а затем — четные.

Работа цифрового индикатора в режиме динамической индикации очень напоминает смену кадров в кино или телевидении. В каждый момент времени работает только один разряд. И каждый разряд показывает свою цифру. Разряды включаются по очереди, начиная с первого и заканчивая последним. Затем все начинается сначала. Такой способ работы индикатора имеет только одно преимущество: он позволяет экономить выводы микропроцессора и количество управляющих элементов (ключей). Представьте, что мы решили бы подключить четыре семисегментных индикатора в обычном статическом режиме. То есть, каждый из них подключен по схеме, изображенной на рис. 1.10. Тогда нам потребовалось бы 32 линии ввода/вывода или четыре порта. А если понадобится подключить более мощные индикаторы, которые не смогут работать без применения управляющих ключей, то нам понадобится собрать еще и 32 схемы, подобные той, которая изображена на рис. 1.6.

А теперь взгляните на рис. 1.11. Выводы одноименных сегментов всех разрядов индикатора объединены вместе и подключены к порту P1 микроконтроллера. При этом линия P1.0 управляет сегментом «a» каждого индикатора, линия P1.1 — сегментом «b». И так далее. Возможность поочередного выбора сегментов обеспечивается при помощи дешифратора DD2 и четырех транзисторных ключей VT1...VT4. На вход дешифратора поступает управляющие сигналы с выводов P3.0 и P3.1. Подавая на эти выходы код номера разряда (от 00B до 11B), микроконтроллер может включать соответствующий разряд. При этом все остальные разряды окажутся выключены.

Работает схема очень просто. Специальная программа, реализующая работу динамической индикации, постоянно перебирает разряды индикатора и выводит на каждый из индикаторов соответствующий символ. Такая программа должна работать в фоновом режиме, независимо от остальных программ, выполняемых процессором. Это достигается применением режима прерываний по таймеру. При переборе разрядов индикатора микроконтроллер сначала подает номер очередного разряда на выходы P3.0, P3.1. Затем, на всех выходах порта P1 устанавливается код, соответствующий выводимому символу. Символ появляется на соответствующем индикаторе. Затем отрабатывается задержка по времени. Назовем ее периодом смены разрядов (t_{cvip}). По истечении времени t_{cvip} микроконтроллер производит смену индицируемого разряда. Для этого он сначала подает код нового разряда на линии P3.0, P3.1. А затем выводит в порт P1 код, соответствующий символу, который в этом новом разряде должен отобразиться. Время между моментами переключения разрядов должно быть всегда одинаковым и равным t_{cvip}. Иначе свечение разрядов получится неравномерным по яркости.

Для того, чтобы обеспечить работу динамической индикации в фоновом режиме, программа использует систему прерываний. По этой причине программа состоит из двух частей: модуля инициализации и процедуры обработки прерывания dind. Программа инициализации выполняется один раз, сразу после включения питания. Она настраивает один из внутренних таймеров микроконтроллера таким образом, чтобы тот непрерывно, с периодом, равным t_{cup}, вызывал процедуру обработки прерывания (dind). Затем микроконтроллер переходит к выполнению основной программы (в листинге не показана). В то время, когда процессор занят выполнением основной программы, работает запущенный вначале внутренний таймер. Он отсчитывает промежуток времени, равный t_{cvip}. Отсчитав этот промежуток времени, таймер вызывает прерывание основной программы и вызов процедуры dind. Процедура производит смену разрядов индикации и возвращает управление основной программе. При этом происходит перезапуск таймера, и отсчет времени возобновляется. Отсчитав следующий период, таймер снова вызывает процедуру обработки прерывания. И так продолжается постоянно, пока включено питание.

Важный этап при разработке подобной программы — определение коэффициента деления для нашего таймера. Как уже было сказано выше, оптимальной частотой смены «кадра» для динамически изменяющегося изображения является величина 48 Гц. Округлим эту величину и выберем частоту смены изображения для динамической индикации 50 Гц. Одним

«кадром» в нашем случае можно считать поочередное включение всех четырех разрядов индикатора. Поэтому частота переключения разрядов будет равна 50 * 4 = 200 Гц. Наш таймер должен формировать сигналы прерывания именно с этой частотой. Теперь нам необходимо найти коэффициент пересчета таймера. Прежде, чем это сделать, нам нужно разобраться, как он работает. Интегрированные таймеры микросхемы AT89C205I имеют несколько режимов работы. Используем для динамической индикации таймер Т0. Для нашего случая таймер удобно перевести в режим подсчета внутренних тактовых импульсов. Внутренние тактовые импульсы формируются из сигнала тактового генератора микроконтроллера путем деления его частоты на 12. Таймер производит подсчет этих импульсов. Причем подсчет ведется в прямом направлении. Это значит, что при поступлении каждого очередного импульса содержимое регистра Т0 увеличивается на единицу. При достижении максимального значения таймер переполняется. Если система прерываний включена, то сигнал переполнения таймера произведет вызов прерывания. Для того, чтобы таймер сформировал нужную нам задержку по времени, при его запуске в регистр Т0 нужно записать некоторое начальное значение. При этом задержка по времени будет определяться количеством импульсов, которое должно поступить на вход таймера прежде, чем он переполнится. После того как таймер сработает и вызовет процедуру обработки прерывания, эта процедура должна перезапустить таймер, то есть снова записать в регистр Т0 коэффициент пересчета. Итак, коэффициент пересчета таймера напрямую определяется его начальным значением, записываемым в регистр таймера перед началом его работы.

Для того, чтобы найти необходимое нам начальное значение, сначала определим требуемый коэффициент пересчета таймера. Для этого нужно выбрать частоту кварцевого генератора. Напомним, что кварц подключается по схеме, приведенной на рис. 1.1. Микросхема AT89C2051 допускает подключение кварцевого резонатора с частотой резонанса до 24 МГц. В данном случае на частоту кварца нет строгих ограничений. Все же желательно выбирать ее повыше. Для простоты расчетов выберем кварц с частотой 12 МГц. При этом на вход таймера будут поступать импульсы с частотой 12/12 = 1 МГц (10^6 Гц). Учитывая, что частота смены разрядов динамической индикации должна быть равна 200 Гц, коэффициент пересчета нашего таймера должен быть равен $10^6/200 = 5000$. Теперь можно найти начальное значение для записи в регистр таймера. Этот регистр содержит 16 двоичных разрядов, а значит, максимальное значение регистра, при котором происходит его переполнение равно 0FFFFH. В десятичном формате это

число равно 65535. Отсюда находим начальное значение: 65535 - 5000 = 60535. Это значение мы и будем использовать в нашей программе.

Схему, изображенную на рис. 1.11, легко усовершенствовать, добавив туда новые разряды. Небольшое изменение в схеме включения дешифратора DD2 позволяет легко расширить количество разрядов динамической индикации до восьми. Новая схема включения дешифратора приведена на рис. 1.12.

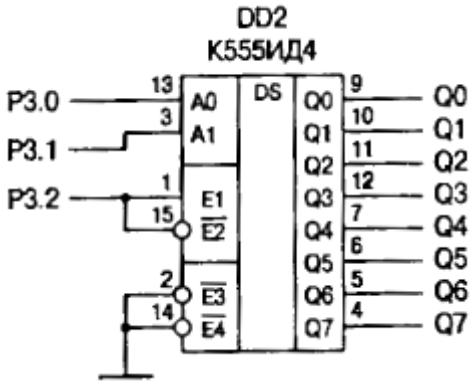


Рисунок 1.12 – Включение дешифратора по полной схеме

В новой схеме используются уже три младших разряда порта P3. В результате мы получим возможность увеличить число индикаторов до восьми. Нужно только предусмотреть еще четыре транзисторных ключа, подключенных к остальным выходам дешифратора. Предлагаю читателю самому разобраться, как будет выглядеть полная схема динамической индикации для восьмиразрядного индикатора и попробовать изменить программу так, чтобы она оказалась пригодной для этого случая.

1.4 Комбинированные устройства

В предыдущих разделах мы рассмотрели основные приемы подключения к микроконтроллеру управляющих клавиш и элементов индикации. На практике чаще всего клавиатура и индикация объединяются в одну общую схему. На рис. 1.13 приведен пример подобной схемы.

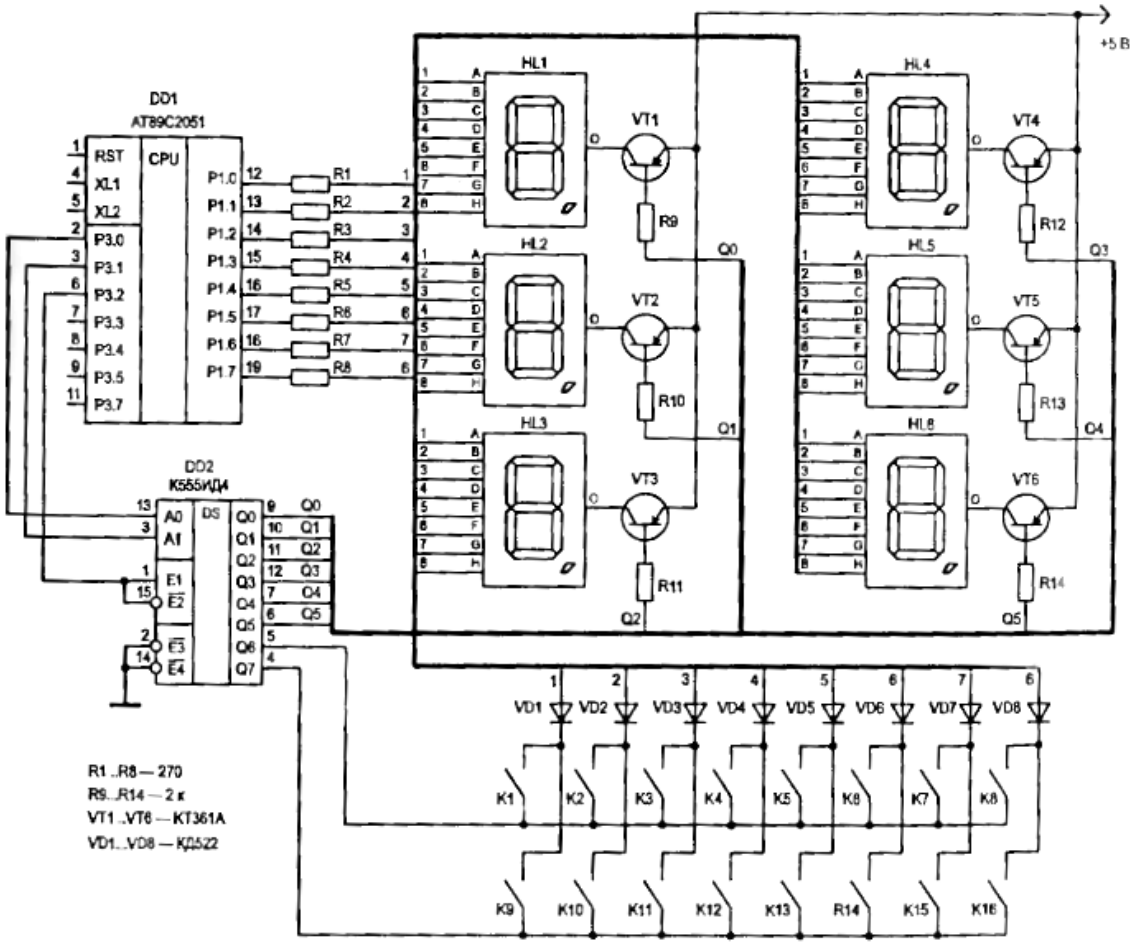


Рисунок 1.13 - Комбинированная схема клавиатуры и индикации

Если вы посмотрите на этот рисунок внимательно, то увидите, что он представляет собой симбиоз рис. 1.4 (матричное включения клавиатуры) и рис. 1.11 (схема с динамической индикацией). Правда, количество клавиш, по сравнению с прототипом, уменьшено вдвое, а количество разрядов индикатора, наоборот, увеличено до шести. Благодаря тому, что клавиатура и индикаторы подключены к разным выходам дешифратора DD2, микроконтроллер может раздельно управлять индикацией и производить опрос клавиатуры.

В связи с тем, что две схемы объединились в одну, пришлось ввести несколько разделяющих диодов: VD1...VD8. Эти диоды предотвращают появление помех для индикации в случае одновременного нажатия двух или более клавиш. Что это за помехи, и как они возникают? Представим себе, что диоды VD1...VD8 отсутствуют и заменены перемычками. Допустим, пользователь нажал и удерживает одновременно клавиши К1 и К2. Это приведет к замыканию сегментов a и b во всех разрядах индикатора. В результате они не будут зажигаться раздельно. Установка диодов устраняет

эту проблему. При одновременном замыкании клавиш К1 и К2 один из двух диодов (VD1 или VD2) всегда будет закрыт.

Как же совместить динамическую индикацию и опрос клавиатуры? Да очень просто! Необходимо выполнять эти два алгоритма в разные моменты времени. Опрос клавиатуры должен происходить в промежутке времени между сменами разрядов в процессе динамической индикации.

2 ДРУГИЕ ВАРИАНТЫ ПОСТРОЕНИЯ СХЕМ ВВОДА-ВЫВОДА

2.1 Пример построения ЦАП

Описываемый ниже цифро-аналоговый преобразователь был разработан как вспомогательный, для применения в качестве составной части многоканального АЦП, схему которого мы рассмотрим в следующем разделе. Однако он вполне может служить самостоятельным устройством для преобразования цифровой информации в аналоговую форму. Точность преобразования этой схемы невелика. Преобразователь позволяет сформировать аналоговый сигнал, имеющий всего 256 градаций уровня. Однако, в некоторых случаях этого может оказаться вполне достаточно. Представьте, например, микропроцессорную систему управления преобразователем напряжения. Подобный ЦАП с успехом может использоваться для формирования синусоидального напряжения, которое затем можно усилить по мощности, подать на вход трансформатора и получить на выходе напряжение 220 В 50 Гц практически идеальной синусоидальной формы.

Схема цифро-аналогового преобразователя приведена на рис. 2.1. Он представляет собой матрицу резисторов R3...R10. Через каждый из этих резисторов на выход преобразователя поступает сигнал с одного из выходов буферного регистра DD2. Номиналы резисторов подобраны по принципу удвоения. Сопротивление резистора R9 в два раза больше, чем сопротивление R10. Сопротивление R8 еще в два раза больше. И так далее. Из стандартного пятипроцентного ряда специально подобраны такие номиналы, чтобы получился ряд значений, в котором каждое последующее в два раза больше предыдущего. В связи с тем, что главным критерием была простота и низкая стоимость, решено было отказаться от прецизионных радиоэлементов и более совершенных схемных решений. Для минимизации влияния нагрузки на точность преобразования используется эмиттерный повторитель (VT1, R11).

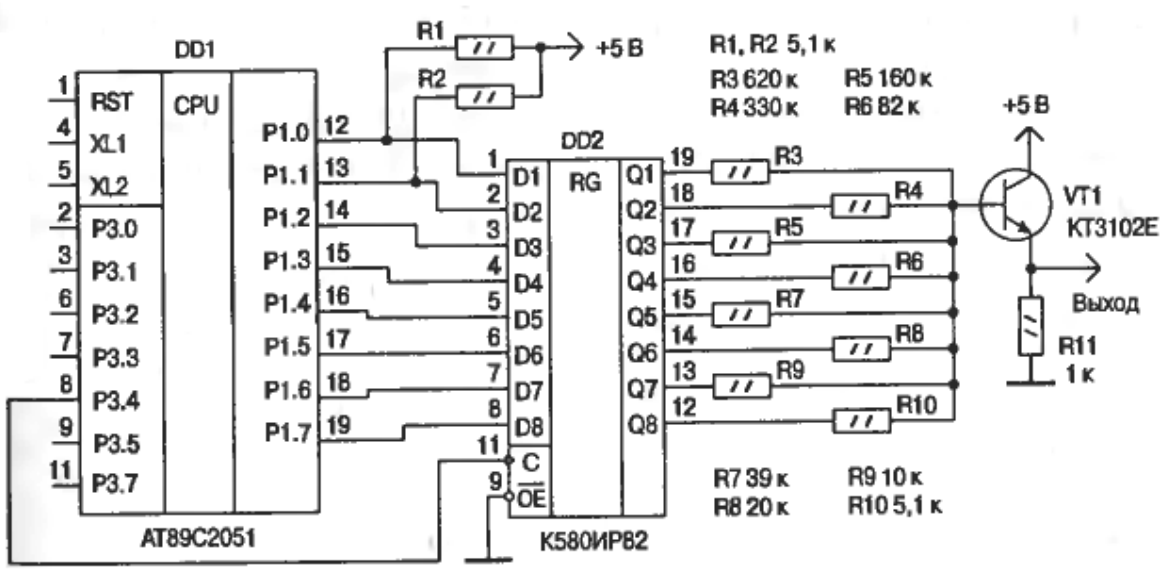


Рис. 2.1 - Схема простейшего ЦАП

Посмотрим, как работает такая схема. Процессор DDI просто записывает число, предназначенное для преобразования в аналоговый сигнал, в буферный регистр DD2. В тот же момент то же число появляется на выходах этого регистра в виде восьми битов двоичного кода. В соответствии со значениями разрядов этого кода, на каждом из выходов DD2 установится одно их двух возможных выходных напряжений. Напряжение на выходе может быть либо равно нулю (низкий логический уровень), либо практически равно напряжению питания (высокий логический уровень). Через соответствующие резисторы матрицы эти напряжения поступят на вход эмиттерного повторителя. Результирующее напряжение сложится из всех этих сигналов. Однако вес каждого из сигналов в общей сумме будет разный. Младший разряд (Q1) будет иметь самый маленький вес. Разряд Q2 будет иметь вес в два раза больший. И так далее. В результате уровень сигнала на выходе схемы будет прямо пропорционален значению двоичного числа, записанного в регистр DD2. При изменении этого числа от OOH до FFH напряжение на выходе преобразователя будет изменяться в пределах от 0 до 5 вольт. Преобразователь способен выдавать 256 дискретных уровней напряжения. Шаг между соседними уровнями будет равен 5/256 = 0,0195 (где-то около 20 мВ). Это в идеале. В реальной схеме диапазон изменения выходного напряжения будет меньше из-за падения напряжения на переходе база-эмиттер. Несколько улучшить параметры такого ЦАП можно путем исключения эмиттерного повторителя, но только в случае, если нагрузка высокоомная.

Для управления регистром DD2 используется порт P1 микроконт-

роллера (передача данных) и линия P3.4 (сигнал записи). Резисторы R1 и R2 установлены для обеспечения нормальной работы выходов P1.0 и P1.1. Эти выходы не имеют внутренних резисторов нагрузки, поэтому они требуют применения внешней нагрузки. Приведенную схему ЦАП можно было бы еше больше упростить. Например, можно обойтись и без микросхемы DD2, подключив матрицу резисторов непосредственно к порту P1. Можно также отказаться от эмиттерного повторителя. Но отказ от эмиттерного повторителя еще более снизит точность преобразования. А отказ от промежуточного регистра приведет к резкому ограничению функциональных возможностей всей схемы. Так как сделает не¬возможным одновременное применение порта P1 для нескольких разных целей. Как еще можно использовать порт P1 мы увидим уже в следующем разделе. Но прежде нам предстоит рассмотреть пример управляющей программы, предназначенной для работы со схемой (рис. 2.1).

Для обслуживания схемы была разработана специальная процедура pDAP (см. листинг 2.1). Для работы этой процедуры предварительно нужно определить только один параметр — описать линию управления внешнего порта. Эта линия получает имя EDS (см. строку 6) и служит для передачи на вход C микросхемы DD2 сигнала записи. Текст процедуры pDAP занимает строки 7...15. Основная программа вызывает процедуру pDAP каждый раз, когда нужно изменить код в регистре DD2. Новый код передается в процедуру через аккумулятор.

Листинг 2.1

Программа обслуживания схемы ЦАП

```
1.          $mod2051
                  :------------------------------ Определение констант
2.          bank0       EQU         00000000B     Коды банков памяти
3.          bank1       EQU         00001000B
4.          bank2       EQU         00010000B
5.          bank3       EQU         00011000B

6.          EDS         BIT         P3.4          Выход включения дешифратора
            :-------------------------------------------------------------------------------------------
            : Сюда вы должны поместить модуль инициализации
            : и текст основной программы
            :-------------------------------------------------------------------------------------------
            :#####################################################
            :##          Подпрограмма ЦАП          ##
            :#####################################################
                  Выводимы код в A
7.          pDAP:       push        psw
```

```
8.                  mov         psw, #bank1     : Выбор банка 1 в памяти

9.                  setb        EDS             : Установка выхода управления

10.                 mov         p1,a            : Вывод кода в регистр ЦАП
11.                 clr         EDS
12.                 setb        EDS
13.                 mov         p1, #0FFH

14.                 pop         psw
15.                 ret

                    :-------------------------------------------------------------------------------
                    :  Сюда вы можете поместить другие подпрограммы
                    :-------------------------------------------------------------------------------
16.                 end
```

2.2 Система аналогового ввода

Итак, мы узнали один из способов, при помощи которого можно преобразовывать код в напряжение. Теперь мы займемся обратным преобразованием. И здесь нам поможет уже знакомая схема ЦАП. Оказывается, любой ЦАП легко превратить в АЦП. Для этого нужен только компаратор напряжения и специальная программа. Блок-схема такого АЦП приведена на рис. 2.2.

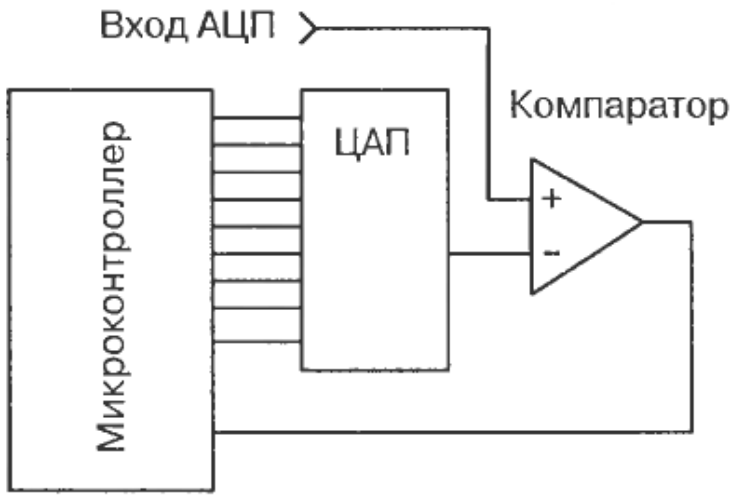


Рисунок 2.2 – Блок-схема построения АЦП

Принцип его работы достаточно прост. Он основан на измерении напряжения на одном из входов компаратора путем подбора напряжения на другом его входе. На этот второй вход компаратора подается напряжение с выхода ЦАП. Выход компаратора тоже подключен к микроконтроллеру. Для того, чтобы измерить величину входного напряжения, нужна специальная измерительная программа. Действуя по этой программе, микропроцессор изменяет напряжение на выходе ЦАП от нуля до максимума, при этом постоянно контролируя уровень сигнала на выходе компаратора. Как только напряжение на выходе ЦАП превысит входное напряжение, сигнал на выходе компаратора переключится с единицы на ноль. Обнаружив перепад, микроконтроллер прекращает перебор кодов. Последнее значение кода, засланное в ЦАП перед тем, как сработал компаратор и есть цифровой эквивалент измеряемого напряжения.

Описанная ниже схема (см. рис. 2.3) работает именно по такому принципу.

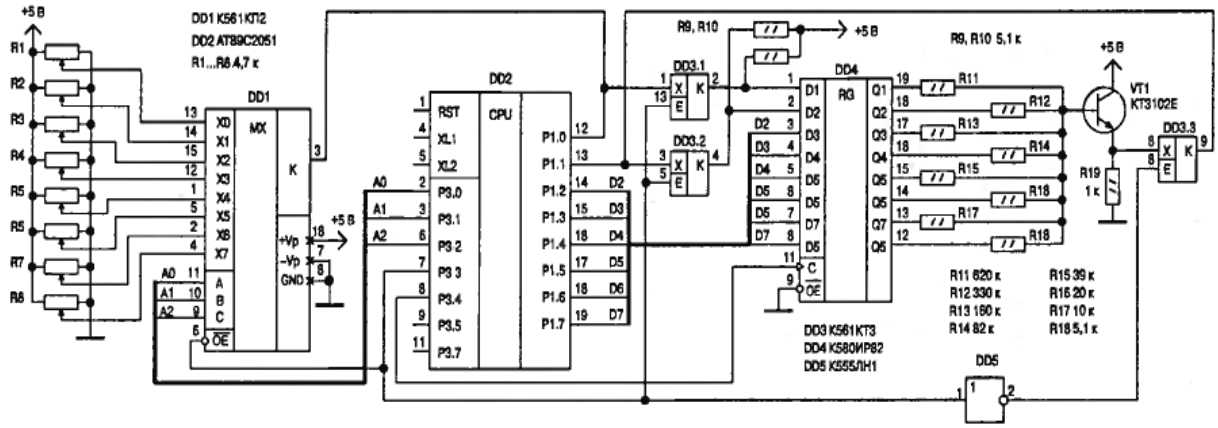


Рисунок 2.3 – Схема многоканального аналогового ввода

Однако это не просто отдельный АЦП. Применение аналогового коммутатора позволяет превратить один преобразователь в целую систему аналогового ввода. Система аналогового ввода позволяет вводить информацию в микропроцессорное устройство путем плавного поворота рукоятки на передней панели. Представьте себе, что вы решили создать систему управления отоплением в индивидуальном жилом доме. Причем температура в каждом помещении дома должна регулироваться индивидуально. Для оперативной регулировки температуры в каждом помещении установлен регулятор. Находясь в комнате, можно повернуть регулятор в ту или иную сторону и сделать теплее или холоднее. В данном

случае регулятор предпочтительнее, чем кнопки «Больше» и «Меньше». Во-первых, крутить ручку удобнее. Во-вторых, отпадает потребность в цифровом индикаторе уровня нагрева. В случае с кнопками без него не обойтись.

Можно придумать множество других применений для системы аналогового ввода. Но вернемся к схеме устройства. Микросхемы AT89C2051 уже содержат встроенный компаратор. Эти упрощает нашу задачу. Очень удобно использовать этот компаратор для построения АЦП. Входы встроенного компаратора совмещены с выводами P1.0, P1.1. Именно по этой причине в выходных схемах разрядов P 1.0 и P 1.1 отсутствуют внутренние резисторы нагрузки. Выход встроенного компаратора подключен к входу P3.6, который не выводится на внешние выводы микросхемы и может работать только с компаратором. Цифро-аналоговый преобразователь, аналогичный приведенному на рис. 2.1, построен на основе регистра DD4, резистивной матрицы R11...RI8 и эмиттерного повторителя VT1, R19 (см. рис. 2.3).

При подключении ЦАП к микроконтроллеру возникает одна небольшая проблема. Два вывода — P1.0 и P1.1 — необходимо использовать сразу двумя способами. Во-первых, они должны работать, как младшие разряды порта P1 при записи кода в регистр DD4. Те же самые выводы должны работать, как входы компаратора в процессе ввода аналогового сигнала. К счастью, эта проблема легко разрешима. Для решения этой задачи применяется метод динамической коммутации выводов. Для коммутации выводов применена микросхема К561КТ3 (DD3 на рис. 2.3). Она содержит четыре управляемых электронных ключа на основе КМОП транзисторов. Такой ключ действует не хуже, чем электромагнитное реле. При подаче управляющего сигнала на вход Е ключа он замыкается, и выводы X и К оказываются замкнутыми между собой. Причем ток по цепи X-К может течь как в прямом, так и в обратном направлении. При снятии управляющего напряжения выводы X и К размыкаются. В описываемой схеме для коммутации выводов P1.0 и P1.1 используются три ключа микросхемы DD3.

Кроме простых аналоговых ключей в схеме используется более сложное переключающее устройство — аналоговый коммутатор. Микросхема аналогового коммутатора DDI (К561КП2) имеет восемь аналоговых входов, обозначенных как XO, XI, ...X7. Любой из аналоговых входов может быть подключен к единственному аналоговому выходу коммутатора, обозначенному буквой К. Для этого микросхема коммутатора содержит восемь аналоговых ключей, аналогичных тем, что применяются в микросхеме К561КТ3. Управление этими ключами осуществляет внутренняя

логика микросхемы, использующая для этого сигналы на входах A, B, C и OE. Одновременно может быть включен только один ключ коммутатора. Для подключения одного из входов (X0...X7) на выход коммутатора на входы A, B и C нужно подать адрес этого входа. Затем на вход OE нужно подать сигнал разрешения. Разрешающим сигналом для инверсного входа OE будет логический ноль. Если на вход OE подать сигнал логической единицы, то все ключи микросхемы будут закрытыми.

При помощи коммутатора DDI к входу нашего АЦП может быть подключен движок любого из переменных резисторов R1...R8. Эти резисторы используются как регулируемые делители напряжения для ввода аналоговой информации. Микроконтроллер опрашивает все резисторы по очереди и считывает величину напряжения на выходе каждого из них. Затем он использует эти величины в процессе реализации основного алгоритма.

Для управления схемой коммутации используются выходы P3.0...P3.2 микроконтроллера. Они служат для подачи адреса на входы A, B, C коммутатора. Выход P3.3 служит для управления всей схемой коммутации.

Рассмотрим работу системы коммугации и всей схемы аналогового ввода в режиме измерения величины входного сигнала. Сначала микропроцессор устанавливает на выходе P3.3 уровень логической единицы. Коммутатор DDI закрывается. Закрывается также и ключ DD3.3 благодаря тому, что элемент D5 инвертирует управляющий сигнал. В то же время ключи DD3.1 и DD3.2 открываются. В результате выводы 12 и 13 микроконтроллера подключаются к выводам 1 и 2 регистра DD4, а аналоговые сигналы от них отключаются. Такой режим работы схемы коммутации назовем цифровым режимом. В цифровом режиме микроконтроллер записывает в регистр DD4 код, предназначенный для цифро-аналогового преобразования. Код запоминается в регистре и соответствующий ему аналоговый сигнал поступает на выход эмиттерного повторителя Закончив работу в цифровом режиме, микроконтроллер устанавливает на выходе P3.3 логический ноль. Ключи DD3.1 и DD3.2 закрываются и отключают выводы 12 и 13 микроконтроллера от цифровых цепей. Одновременно открывается коммутатор DDI и ключ DD3.3. На вход компаратора (вывод 12 микросхемы DD2) поступает аналоговый сигнал с одного из резисторов R1...R8. А на второй вход компаратора (вывод 13 микросхемы DD2) поступает напряжение с эмиттерного повторителя. Этот режим работы схемы мы будем называть аналоговым режимом. В аналоговом режиме два напряжения, поступившие на входы компаратора, сравниваются между собой. Микроконтроллер считывает результат сравнения через линию P3.6. Если микроконтроллер обнаружит на выходе

компаратора логическую единицу, то процесс измерения продолжится. Система снова перейдет в цифровой режим, запишет новое значение кода в регистр ЦАП (DD4), а затем перейдет в аналоговый режим и снова проверит сигнал на выходе компаратора.

Микропроцессор многократно повторяет описанный выше цикл сравнения. При этом он каждый раз увеличивает значение кода, засылаемого в регистр DD4. Так продолжается до тех пор, пока единичный уровень на выходе компаратора не сменится на нулевой.

Два резистора R9 и R10 — это нагрузка для линий P 1.0 и P 1.1. Они подключаются к выводам 12 и 13 микроконтроллера посредством ключей DD3.1 и DD3.2 только в цифровом режиме работы схемы коммутации. В аналоговом режиме ключи DD3.1 и DD3.2 закрываются и резисторы не мешают работе компаратора.

В качестве датчиков сигнала для аналогового ввода наряду с переменными резисторами можно применять и другие источники аналогового сигнала. Например, терморезисторы, аналоговые датчики давления, влажности и т.п. Необходимо лишь согласовать диапазон изменения напряжения на выходе датчика с входным диапазоном нашего АЦП. Для того, чтобы полностью использовать диапазон АЦП, напряжение на его входе должно изменяться в пределах от 0 до +5 В. Для согласования диапазонов не обойтись без специального согласующего устройства. Например, в случае использования терморезистора согласующее устройство должно преобразовывать уровень снимаемого напряжения таким образом, чтобы при изменении температуры в пределах заданного диапазона напряжение на входе АЦП изменялось от нуля до 5 В.

Для работы со схемой аналогового ввода была разработана специальная процедура pADP. Процедура считывает величину напряжения с одного из восьми аналоговых входов и помещает ее в аккумулятор.

2.3 Жидкокристаллический дисплей

В первой главе мы узнали, как построить цифровой дисплей на основе светодиодных семисегментных индикаторов. Однако дисплеи, построенные на основе светодиодных индикаторов, обладают рядом недостатков. Во-первых, это относительно большой уровень потребляемой мощности. А во-вторых, низкая контрастность изображения в условиях прямых солнечных лучей. В настоящее время имеется альтернативный вариант — использование жидкокристаллических индикаторов (ЖКИ). В английской литературе для

таких индикаторов существует свое сокращение — LCD. Производители электронного оборудования предлагают широчайший выбор готовых индикаторных панелей. Начиная с простых однострочных дисплеев и заканчивая большими графическими дисплеями, позволяющими одновременно отображать большое количество информации.

Обычно ЖКИ дисплей имеет встроенный контроллер управления. Миниатюрные жидкокристаллические дисплеи очень удобно применять в микропроцессорных системах. Они информативны, потребляют маленький ток от источника питания. И, наконец, готовые ЖКИ панели промышленного производства очень красиво выглядят на передней панели любого микропроцессорного устройства. Все ЖКИ дисплеи подразделяются на два больших класса: цифровые семисегментные дисплеи и алфавитно-цифровые знакосинтезирующие. Первый класс индикаторов содержит одну или несколько строк, состоящих из семисегментных разрядов. По расположению сегментов такие дисплеи похожи на уже знакомые нам светодиодные. В знакосинтезирующих индикаторах каждый из разрядов представляет собой матрицу точек, в которой могут отображаться не только цифры, но и буквы, а также любые другие символы. Эти индикаторы более дорогие по сравнению с семисегментными, но зато они гораздо более информативны.

В литературе вы уже могли встречать описание различных конструкций с использованием готовых ЖКИ индикаторов. В качестве примера возьмем ЖКИ модуль типа МТ-10Т7-7 российской фирмы МЭЛТ.

Основные характеристики модуля MT-WT7-7:
Тип индикатора ..... цифровой семисегментный;
Количество строк 1;
Количество разрядов 10**;**
Напряжение питания:
- минимальное ….+3 В;
- максимальное ….+5 В;

Ток потребления ..... 30 мкА;
Способ регулировки контрастности - ручной (внешний резистор);
Количество выводов 12;
Габаритные размеры ..... 66x31.5x9.5 мм.

Каждое знакоместо — это семисегментный цифровой индикатор с точкой. Расположение и наименование сегментов в каждом разряде

индикатора аналогично светодиодному индикатору (см. рис. 1.7). Функциональная схема модуля МТ-10Т7-7 показана на рис. 2.4.

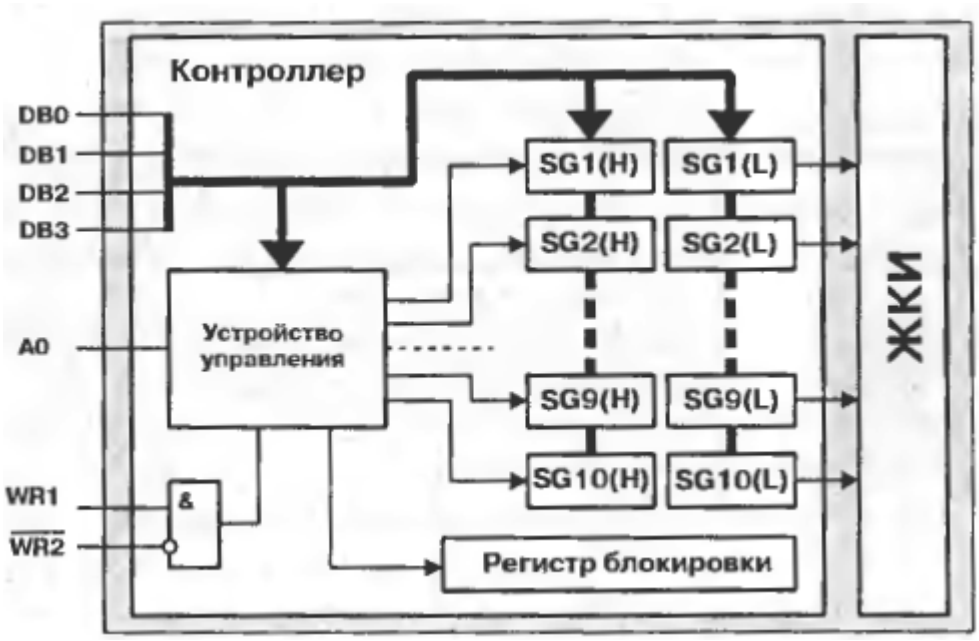


Рисунок 2.4 - Функциональная схема дисплея МТ-10Т7-7

Система управления построена таким образом, что каждый сегмент каждого разряда может быть включен или выключен независимо от всех остальных сегментов. Специализированный контроллер содержит десять восьмиразрядных регистров хранения информации (SG1...SG10). Каждый регистр управляет одним разрядом ЖКИ (см. табл. 2.1). Одиннадцатый регистр — это регистр блокировки. Он используется для хранения признака временной блокировки работы модуля. Каждый из регистров SG1...SG10 разделен на две тетрады. То есть на два полурегистра по 4 разряда в каждом. Они обозначены на функциональной схеме буквами H (старший) и L (младший).

Таблица 2.1 – Адреса регистров ЖКИ модуля

№ знакоместа	1	2	3	4	5	6	7	8	9	10	--
Наименование регистра	SG1	SG2	SG3	SG4	SG5	SG6	SG7	SG8	SG9	SG10	Блокировка
Адрес (HEX)	00	01	02	03	04	05	06	07	08	09	OF

Разделение управляющих регистров на полурегистры необходимо для уменьшения количества управляющих выводов. Модуль имеет всего четыре входа для записи данных (DB0...DB3) и три входа управления. Входы

DB0...DB3 используются как для ввода данных, так и для ввода адреса. Для переключения режимов «Адрес/данные» служит вход АО. Если на этот вход подать низкий логический уровень, то сигналы на входах DB0...DB3 будут интерпретироваться как адрес. Если на входе АО высокий уровень, то входы DB0...DB3 используются для ввода данных. Входы WR1 и WR2 — это входы разрешения записи. Первый из них — прямой, второй — инверсный. Запись данных или адреса происходит только в том случае, если уровень сигнала на входе WRI равен единице, а уровень сигнала на входе WR2 при этом равен нулю. Наличие двух входов управления записью позволяет упростить схему в случае параллельного включения нескольких модулей. Если модуль только один, удобно вывод WR2 напрямую соединить с общим проводом, а процессом записи управлять при помощи входа WRI.

Рассмотрим теперь логику управления ЖКИ модулем МТ-ЮТ7-7. Диаграмма сигналов на входах модуля в процессе записи приведена на рис. 2.6. Этот рисунок будет иллюстрировать описание управляющего алгоритма. Допустим, что нам нужно высветить какой-либо символ в первом (самом левом) разряде индикатора. Для управления этим разрядом служит регистр SGI (см. табл. 2.1). Поэтому нам нужно записать в регистр SGI некий код, который соответствует высвечиваемому символу. Для этого сначала нужно передать в модуль адрес этого регистра.

Процесс передачи адреса происходит следующим образом: прежде всего на входы DB0...DB3 подается значение адреса (для регистра SG1 адрес равен ООН). Затем на входе АО необходимо установить низкий логический уровень. Завершается процесс записи адреса коротким положительным импульсом на входе WRI (при условии, что на входе WR2 ноль). По переднему фронту этого импульса значение адреса записывается во внутренний регистр адреса модуля. Теперь, когда адрес нужного регистра записан, можно приступить к записи в этот регистр кода символа. Код записывается в два приема. Сначала записывается младший полубайт, затем старший. Полубайты записываются один за другим.

Система управления модуля автоматически помещает первый из полученных полубайтов в младшую часть регистра (SGl(L)), а второй полубайт в старшую его часть (SGl(H)). Запись данных происходит точно так же, как и запись адреса. Однако, на входе А0 теперь должен присутствовать сигнал логической единицы. Сначала на входы DB0...DB3 подается младший полубайт. Положительный импульс на входе WR1 записывает информацию в модуль. Затем на входы DB0...DB3 подается старший полубайт. Следующий единичный импульс на входе WR1 записывает и его. После записи обоих полубайтов значение адреса во внутреннем регистре адреса ЖКИ модуля

автоматически увеличивается на единицу. Это позволяет записывать коды символов в несколько регистров подряд, без дополнительных операций по записи адреса. Этот режим, как и режим однократной записи, наглядно показан на рис. 2.5.

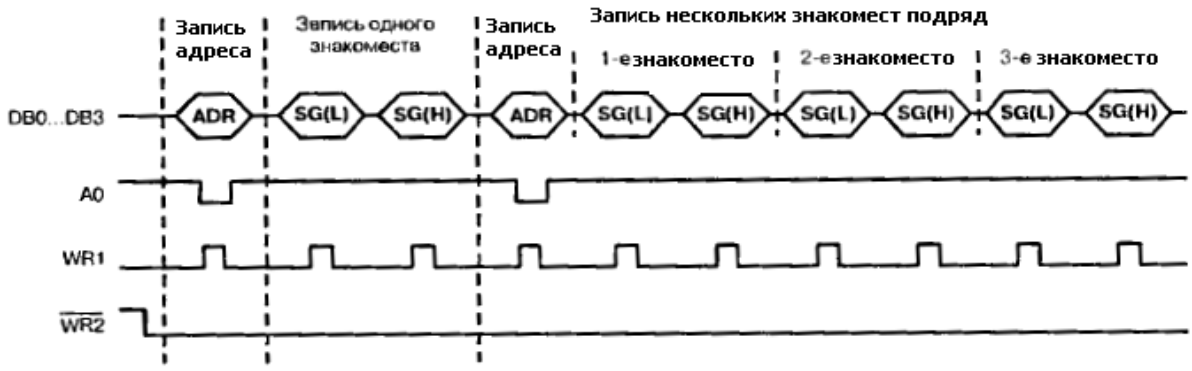
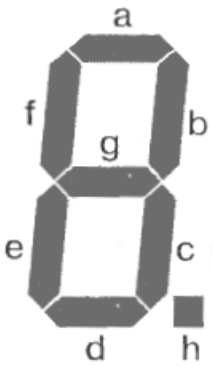


Рисунок 2.5 – Временные характеристики процесса записи управляющих сигналов

Теперь разберемся с тем, какие коды нужно записывать в регистры модуля для того, чтобы высветить на индикаторе тот или иной символ. Как уже говорилось, каждому разряду любого из регистров SGx соответствует свой сегмент семисегментного индикатора. В таблице на рис. 2.6 показано соответствие разрядов регистра и сегментов индикатора. Высокий логический уровень включает соответствующий сегмент, а низкий уровень его выключает. На рис. 2.6 показаны коды, которые нужно записывать в регистр SGx для того, чтобы высветить тот или иной символ.

Разряд	Сегмент	Символы									
		0	1	2	3	4	5	6	7	8	9
DB0(L)	g	0	0	1	1	1	1	1	0	1	1
DB1(L)	e	1	0	1	0	0	0	1	0	1	0
DB2(L)	d	1	0	1	1	0	1	1	0	1	1
DB3(L)	a	1	0	1	1	0	1	1	1	1	1
DB0(H)	h	0	0	0	0	0	0	0	0	0	0
DB1(H)	b	1	1	1	1	1	0	0	1	1	1
DB2(H)	c	1	1	0	1	1	1	1	1	1	1
DB3(H)	f	1	0	0	0	1	1	1	1	1	1



Разряд	Сегмент	Символы										
		A	b	C	d	E	F		°	-	r	o
DB0(L)	g	1	1	0	1	1	1	0	1	0	0	0
DB1(L)	e	1	1	1	1	1	1	0	0	0	0	1
DB2(L)	d	0	1	1	1	1	0	0	1	0	0	0
DB3(L)	a	1	0	1	0	1	1	0	0	0	0	0
DB0(H)	h	0	0	0	0	0	0	0	1	0	0	0
DB1(H)	b	1	0	0	1	0	0	0	0	0	0	1
DB2(H)	c	1	1	0	1	0	0	0	0	0	1	1
DB3(H)	f	1	1	1	0	1	1	0	1	1	1	1

Рисунок 2.6 – Схема соответствия разрядов регистра и сегментов индикатора

Как видно из рисунка, на семисегментном индикаторе можно высвечивать не только символы цифр от «0» до «9», но и другие знаки и даже некоторые буквы. В наш пример дополнительно включены символы латинского алфавита от «A» до «F», символ градуса (« ° »), минус, а также буквы «г» и «о». Символы латинского аафавита используются для отображения на индикаторе чисел в шестнадцатеричной форме. Знак минус и символ градуса можно использовать для индикации температуры. Буквы «г» и «о» используются для вывода на индикатор слова «Еггог» (ошибка). Возможности по отображению букв у семисегментного индикатора ограничены. Так пришлось буквы

«b», «d», «г» и «о» сделать строчными, а остальные буквы заглавными. Это немного затрудняет читаемость при отображении цифр в шестнадцатеричном формате. Но к этому легко привыкнуть. И, наконец, еще один символ, который приведен в таблице на рис. 2.6 — это символ пробела. Пробел — это когда не один из сегментов не светится. Символ пробела

удобно выводить на индикатор в том случае, когда между двумя цифрами на экране должен быть промежуток.

Каким же образом определяются коды, приведенные на рис.2.6 ? Это делается очень просто. Достаточно посмотреть на изображение семисегментного индикатора в правой верхней части рисунка и представить себе, какие из сегментов индикатора должны быть включены, а какие выключены для отображения того или иного символа. Затем нужно проставить в строке, соответствующей каждому из сегментов ноль либо единицу. Ноль, если сегмент должен быть выключен, а единицу, если включен. Сегмент «h» (десятичная точка) будем считать пока выключенным.

Как будут выглядеть описанные выше символы и как кодируется каждый из них видно из табл. 2.2. В графе SG(L) этой таблицы приводится младший полубайт кода символа. В графе SG(H) — старший полубайт. Это те же самые значения, которые мы уже видели на рис. 2.6. В графе HEX представлен уже полный код символа в шестнадцатеричном виде. Полный код получается путем объединения обоих полубайтов (SG(H) и SG(L)).

Таблица 2.2 – Аппаратная кодировка символов для ЖКИ модуля

Изображение	SG(H)	SG(L)	HEX	Изображение	SG(H)	SG(L)	HEX
0	1110	1110	0EEH	A	1110	1011	0EBH
1	0110	0000	060H	b	1100	0111	0C7H
2	0010	1111	02FH	C	1000	1110	08EH
3	0110	1101	06DH	d	0110	0111	067H
4	1110	0001	0E1H	E	1000	1111	08FH
5	1100	1101	0CDH	F	1000	1011	08BH
6	1100	1111	0CFH	o	1010	1001	0A9H
7	0110	1000	068H	-	0000	0001	001H
8	1110	1111	0EFH	r	0000	0011	003H
9	1110	1101	0EDH	o	0100	0111	047H

И последнее, о чем нужно сказать в этом разделе, так это о регистре блокировки. Этот регистр не разделен на два полурегистра. Напротив, он имеет всего один действующий бит. Этот бит соответствует входу DB0. При записи нуля в разряд DB0 регистра блокировки, работа модуля блокируется. Запись во все остальные регистры модуля становится невозможной. Режим блокировки действует на 30 циклов записи. То есть, если в этом режиме пытаться все же записывать информацию, то после 30 попыток режим блокировки автоматически снимается. Если режим блокировки нужно снять раньше, то нужно просто записать единицу в разряд DB0 регистра блокировки.

Для управления входами A0 и WR1 можно было бы использовать любые два из остальных выводов порта. В данной схеме для управления входом A0 используется линия P1.6, а для управления входом WR1 — линия P1.4. Выбор именно этих линий обусловлен исключительно соображениями удобства разводки печатной платы. Удобство при разводке особенно важно в случае применения плоского кабеля, где запутывать провода нежелательно. Резисторы R1 и R2 выполняют роль нагрузки для линий P1.0 и P1.1 микроконтроллера. Резистор R3 служит для регулировки контрастности изображения. Каждый конкретный экземпляр ЖКИ модуля требует подстройки контрастности. Контрастность выставляется таким образом, чтобы включенные сегменты были видны максимально ярко, но при этом сегменты, которые должны быть отключены, на индикаторе не просматривались.

2.4 Подключение ЖКИ дисплея к микроконтроллеру

Подключать ЖКИ модуль MT-10T7-7 к микроконтроллеру очень просто. Учитывая, что вход WR2 мы напрямую подключаем к общему проводу, для управления модулем потребуется всего 6 линий. Одна из возможных схем подключения изображена на рис. 2.7. В этой схеме для управления ЖКИ модулем используется порт P1 микроконтроллера. В качестве шины данных/адреса (DB0...DB3) используются четыре младших разряда порта.

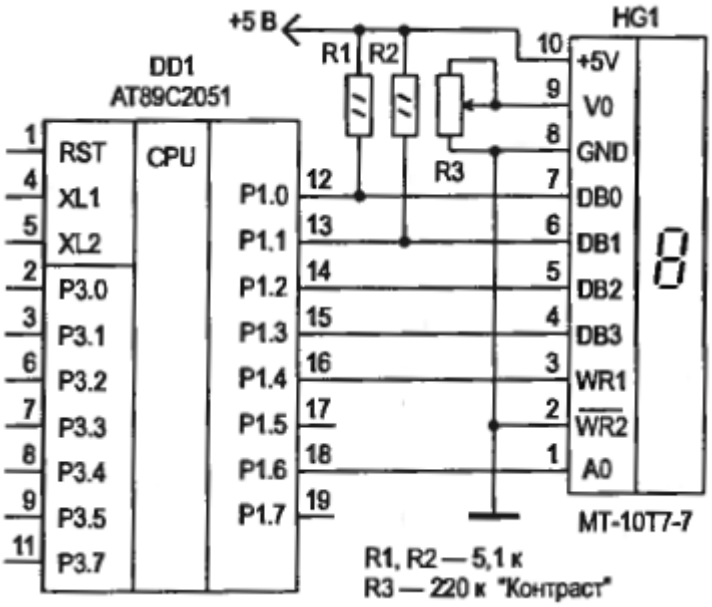


Рисунок 2.7 – Пример схемы подключения ЖКИ модуля к микроконтроллеру

Как и в случае со светодиодным дисплеем, в нашем примере все отображаемые символы кодируются двумя разными способами. С одним видом кодировки мы познакомились при составлении табл. 2.2. В этой таблице приведены коды, которые нужно (в два приема) выдавать на индикатор для того, чтобы высветить необходимый нам символ. Такой набор кодов можно назвать аппаратной кодировкой. От нее никуда не денешься: она заложена в конструкцию ЖКИ модуля.

Однако использовать такой способ кодирования в программе неудобно. Значение кода не связано с функцией высвечиваемого знака. С такими кодами затруднительно производить логические операции и различные вычисления. Учитывая то, что модуль МТ-10Т7-7 в основном ориентирован на вывод чисел, было бы гораздо удобнее, чтобы код символа по возможности совпадал с его значением. Такая кодировка приведена в табл. 2.3. Символу «О» присвоен код ноль (00Н). Символу «I» — код единица (01Н) и так далее. Так как буквы латинского алфавита A, B, C, D, E и F предназначены для отображения шестнадцатеричных чисел, то им присвоены коды с 10 по 15. Для остальных символов выбор кодов может быть любым. Поэтому коды этих символов продолжают начатую последовательность. Такой способ кодировки мы будем называть естественной кодировкой.

Таблица 2.3 – Естественная кодировка символов

Символ	Код	Символ	Код	Символ	Код
0	0	7	7	E	14
1	1	8	8	F	15
2	2	9	9	Пробел	16
3	3	A	10	o	17
4	4	b	11	-	18
5	5	C	12	r	19
6	6	d	13	o	20

В процессе вывода символа на индикатор производится перекодировка. Как и в случае со светодиодным индикатором для перекодировки используется специальная таблица символов.

3 РАБОТА С I^2C ШИНОЙ

3.1 Общие сведения

В предыдущих главах мы рассмотрели основные способы подключения периферийных устройств, из разряда традиционных приемов. Описанные при этом схемы — это системы с параллельной передачей информации. Главный недостаток параллельного способа передачи информации — в них задействовано слишком большое количество выводов микроконтроллера. Можно, конечно, применить другой микроконтроллер. Например, микросхема AT89C51 имеет четыре порта ввода/вывода. Существуют приемы расширения количества линий. Однако все эти приемы усложняют конструкцию. Как следствие, увеличивается время разработки системы, уменьшается ее надежность.

Современная микроэлектроника предлагает новые способы решения этих задач. В распоряжении современного разработчика существуют микросхемы, которые способны обмениваться информацией с любым микроконтроллером при помощи последовательной шины передачи данных. Устройство и логика работы таких шин может быть самая разная. В настоящей книге я хотел бы рассказать о двух самых популярных стандартах последовательных линий связи.

И начать я хочу с детища фирмы Philips, которое называется I^2C шина. Многим, кто имеет дело с микроэлектроникой, хорошо знакомо это оригинальное наименование. Название шины происходит от сокращения ПС, что расшифровывается, как «Inter IC». Сокращение «IC» — это известное сокращение, которое расшифровывается как «интегральная схема». Поэтому «Inter IC bus» расшифровывается примерно так: «межмикросхемная шина». Для большей наглядности сокращение ПС решено было записывать

I^2C шина была разработана около двадцати лет назад, и была задумана как средство передачи сигналов управления на различные микросхемы сложных электронных устройств. Например, современные телевизионные приемники широко используют наборы специализированных микросхем с цифровым управлением. Каждая микросхема служит основой целого блока. Существуют микросхемы радиоканала, декодера цветности, выходных видеоусилителей, системы синхронизации и другие. При этом сами микросхемы — это аналоговые устройства, осуществляющие преобразование

сигналов традиционным способом. Однако, для уменьшения количества выводов и упрощения схемы соединений все регулировки внутренних параметров этих микросхем производятся при помощи цифровой системы управления, основанной на применении 1^2C шины. Управляющие сигналы вырабатываются центральным процессором телевизора и передаются посредством I^2C шины на все остальные элементы схемы. Процессор может изменять в процессе работы любые параметры в любом блоке телевизора. Таким образом, происходит регулировка яркости, контраста, насыщенности, громкости и т.д. Таким же образом происходит переключение и настройка телевизионных каналов, выбор телевизионного стандарта (PAL, SECAM, NTSC) и так далее. Подобная система управления применяется и в других областях электронной техники. Например, в радиоприемниках, магнитолах, системах измерения и в промышленном оборудовании.

Первую спецификацию на I^2C шину (версии 1.0) фирма Philips выпустила в 1992 году. А к 1998 году I^2C шина стала фактическим мировым стандартом, который теперь применяется более чем в 1000 различных видов микросхем и лицензия на ее использование куплена более чем 70 компаниями. Спецификация на шину много раз дорабатывалась. Последний опубликованный стандарт на шину I^2C имеет номер версии 2.1.

Основные преимущества I^2C шины — это простота ее аппаратной реализации. Шина представляет собой двухпроводную линию, на которую параллельно подключаются все управляемые ею микросхемы. При этом нет никаких особенных требований к самим проводникам, составляющим шину. Устройства, подключаемые к шине, должны иметь специальный I^2C интерфейс, который аппаратным путем реализует всю логику работы шины. Существует также возможность программной реализации I^2C интерфейса. В этом случае в качестве элементов, подключаемых к шине, выступают микроконтроллеры.

Технология передачи информации при помощи I^2C шины позволяет решать огромный спектр задач. Однако, исходя из темы нашей книги, нас будет интересовать только следующий аспект: подключение периферийных устройств к микроконтроллеру. Среди 150 видов микросхем производства фирмы Philips, имеющих I^2C интерфейс, и более 1000 видов микросхем других производителей мы можем найти практически все типы знакомых нам периферийных устройств, которые с успехом можно использовать при разработке любой микропроцессорной системы. Вот только краткий перечень этих устройств: аналого-цифровые и цифро-аналоговые преобразователи, контроллеры ЖКИ и светодиодных дисплеев, универсальные порты ввода/вывода, преобразователи температуры и напряжения, микросхемы

энергонезависимой памяти (EEPROM), контроллеры DIP переключателей и многое другое. Нужно только выбрать одну или несколько микросхем с нужными нам функциями и подсоединить к микроконтроллеру посредством I^2C шины. Для присоединения всех этих устройств нам потребуется задействовать всего две линии ввода/вывода микроконтроллера.I

3.2 Основные характеристики I^2C шины

I^2C — это двухпроводная, двунаправленная шина, с Последовательной передачей данных. В стандартном режиме работы (Standard mode) шина может адресовать до 128 подключенных к ней устройств. Длина шины и количество подключаемых к ней устройств ограничены максимально допустимой суммарной электрической емкостью всех ее элементов. Значение этой емкости не должно превышать 400 пФ. Скорость передачи информации в стандартном режиме составляет 100 Кбит/с.

Последняя модификация шины (версия 2.1) предусматривает несколько дополнительных режимов работы:

- ✓ *Ускоренный режим* (Fast mode) позволяет передавать данные со скоростью 400 Кбит/с.
- ✓ *Режим повышенной скорости* (High-speed mode или HS-mode) разработан для современных высокоскоростных систем. Скорость передачи информации (bit rate) составляет 3,4 Мбит/с.
- ✓ *Режим 10-битной адресации* позволяет адресовать до 1024 устройств на I^2C шине.

Все дополнительные режимы совместимы сверху вниз. Это значит, что микросхемы, поддерживающие один или несколько дополнительных режимов, без труда будут работать и в стандартном режиме.

Мы остановимся только на стандартном режиме работы I^2C шины с семибитовой адресацией. Если мы ставим перед собой задачу — просто подключить периферийное устройство к микропроцессору, то стандартного режима работы для этого будет вполне достаточно. Поэтому все, что вы прочтете далее, относится только к стандартному режиму. Итак, рассмотрим электрическую схему шины (см. рис. 3.1). Шипа состоит из линии передачи данных (SDA) и линии синхронизации (SCL). Все 1^2C устройства подключаются к обеим линиям шины параллельно. Кроме двух сигнальных проводников, все подключаемые к шине устройства должны иметь один общий провод (минус источника питания). Кроме того, на каждое из устройств необходимо подать напряжение питания. Источник питания не

обязательно должен быть общим, однако в пределах одной шины все напряжения питания должны быть одинаковыми.

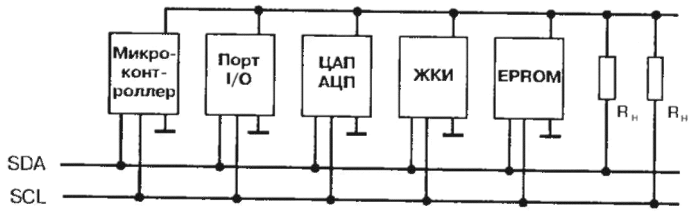


Рисунок 3.1 – Присоединение микросхем к I^2C шине

Устройства, подключаемые к I^2C шине, по своему назначению делятся на ведущие (Master) и ведомые (Slave). Это известная терминология. Буквальная расшифровка терминов Master и Slave означает — «хозяин» и «раб». В русскоязычной литературе пользуются и теми и другими названиями. Для правильной работы, как минимум, одно из подключенных к шине устройств должно быть ведущим. Ведущее устройство берет на себя все управление шиной. Оно вырабатывает тактовые импульсы на шине синхронизации (SCL), а также инициирует сеансы передачи информации. Информация может передаваться как от ведущего устройства к ведомому, так и в обратном направлении. В обоих случаях процесс передачи информации управляется ведущим устройством. В зависимости от направления передачи информации, как Master, так и Slave устройство в разные моменты времени могут выступать в качестве передатчика (Transmitter), либо в качестве приемника (Receiver). Если информация передается от Master к Slave то Master-устройство выступает как передатчик, а Slave — как приемник. При передаче информации от Slave к Master, уже Master станет приемником, а Slave, соответственно, передатчиком.

Каждое Slave-устройства, подключенное к I^2C шине, имеет свой уникальный адрес. Ведущее (Master) устройство имеет возможность обратиться к любому из ведомых, используя для выбора нужного устройства его адрес. В стандартном режиме для адреса используется 7 бит информации.

В качестве ведущего (Maser) устройства чаще всего выступает микроконтроллер. Причем для этой цели подойдет практически любой существующий тип микроконтроллеров. Протокол обмена по I^2C шине реализуется обычно программным путем. Примеры таких программ будут подробно описаны в конце этой главы. Для подключения микроконтроллера к проводникам I^2C шины обычно используются любые две линии любого порта ввода/вывода микроконтроллера. Существуют микроконтроллеры, в

которых 1^2C интерфейс реализован аппаратным путем. Однако они имеют большую стоимость и применяются значительно реже. В качестве ведомых (Slave) устройств обычно используют специализированные микросхемы, имеющие встроенный I^2C интерфейс. Шина I^2C допускает наличие на шине нескольких Master-устройств. Для этого в протоколе шины предусмотрена процедура синхронизации ведущих устройств между собой (Synchronization) и арбитражная процедура (Arbitration). Однако в задачи книги не входит описание подобного режима.

3.3 Схема построения I^2C интерфейса

На рис. 3.2 показана принципиальная электрическая схема шины I^2C. На схеме условно показаны два устройства, подключенные к линиям SDA и SCL шины. Остальные устройства подключаются точно также.

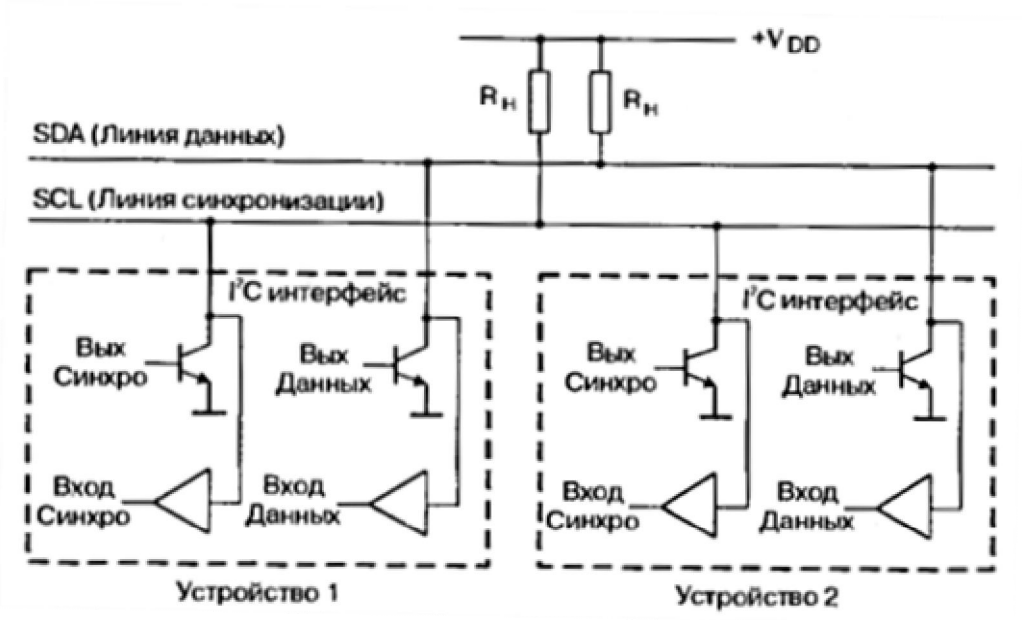


Рисунок 3.2 - Принципиальная электрическая схема шины I^2C

С электрической точки зрения линии SDA и SLC построены одинаково. Причем каждый из двух выводов I^2C интерфейса любой микросхемы должен быть и входом и выходом. Для правильной работы шины все выходные каскады должны быть построены по схеме с открытым коллектором, как это показано на рис. 3.2. Общие нагрузочные резисторы ($R_н$), один для линии SDA и один для SLC, могут быть установлены в любом месте. Но чаще всего

эти резисторы ставятся рядом с ведущим устройством. Через каждый из таких резисторов на линию поступает напряжение питания от источника V_{DD}.

Такое схемное построение шины позволяет не только легко менять направление передачи данных, но и решать задачи захвата/отпускания шины, передачи сигнала подтверждения и т.д. В исходном состоянии на выходах всех микросхем, подключенных к I^2C шине должны быть установлены сигналы логической единицы (транзисторы выходных каскадов всех микросхем закрыты). Это позволяет всем устройствам, подключенным к шине, находиться в режиме чтения

Теперь допустим, что устройство 1 начинает процесс передачи информации. В этом случае устройство 1 выступает в качестве передатчика. Остальные устройства будут выполнять роль приемников. При этом в передатчике происходят следующие процессы.Сигнал данных и сигнал синхронизации поступают на соответствующие выходные каскады I^2C интерфейса. Оба сигнала представляют собой последовательность импульсов. Эти импульсы открывают транзисторы выходных каскадов. Когда транзистор открывается, он «подсаживает» соответствующую линию I^2C шины. Напряжение на «подсаженной» линии близко к нулю. При «отпущенной» линии (транзистор закрыт) напряжение близко к напряжению питания. Переданные таким образом сигналы данных и синхронизации поступают на входы всех остальных микросхем, работающих в режиме приема. Каждый приемник считывает переданный сигнал и оценивает его содержание. Кроме полезных данных в передаваемый сигнал заложен адрес устройства, для которого эти данные предназначены. Если передаваемый адрес совпадет с индивидуальным адресом одной из микросхем-приемников, то эта микросхема примет и обработает передаваемые данные. Остальные микросхемы их проигнорируют. Это общий принцип работы I^2C шины. Теперь рассмотрим, каким образом происходит передача информации.

Информация по I^2C шине передастся побайтно. Каждый байт передается последовательно, бит за битом. В конце байта формируется контрольный бит. При этом передача каждого бита по шине SDA сопровождается передачей синхроимпульса по шине SCL. Приемник побитно читает сигнал с шины SDA, используя сигнал на шине SCL для синхронизации. После передачи восьми битов данных передатчик запрашивает сигнал подтверждения от приемника. Для этого он «отпускает» шину SDA и передает импульс синхронизации по шине SCL. В ответ на этот импульс, приемник должен сам «подсадить» шину SDA. Передатчик считывает сигнал на шине. Если подтверждение получено, он продолжает передачу. Если нет, передача приостанавливается. Таким образом, схема

шины позволяет легко изменять направление передачи информации в процессе одного сеанса обмена.

В заключение этого раздела сделаем небольшое уточнение. На рис. 3.2 показан вариант построения I^2C интерфейса, где в выходных каскадах применены биполярные транзисторы. Это лишь один из возможных вариантов. С таким же успехом можно использовать и полевые транзисторы. Более того, в настоящее время именно так и поступают большинство производителей микросхем с I^2C интерфейсом.

3.4 Протокол I^2C шины

В предыдущем разделе мы рассмотрели электрическую схему I^2C шины. Такой принцип построения схем последовательного канала часто используется в самых различных устройствах последовательной передачи информации. Для того, чтобы приведенная на рис. 3.3 схема могла считаться I^2C шиной, необходимо, чтобы во всех входящих в нее микросхемах была реализована определенная логика работы.

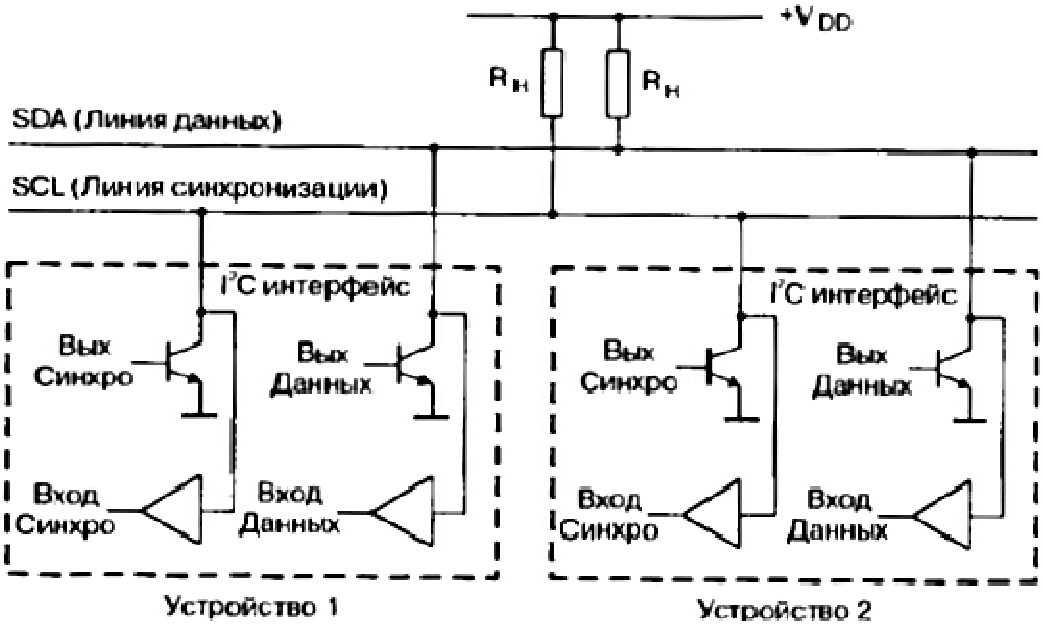


Рисунок 3.3 – Принципиальная электрическая схема I^2C шины

Набор правил, определяющий логику взаимодействия всех элементов I^2C шины, и называется протоколом I^2C шины. В данном разделе мы

подробно рассмотрим все особенности протокола I^2C шины в стандартном режиме работы (Standard Mode).

3.4.1 Взаимодействие на уровне электрических сигналов

Как уже говорилось ранее, I^2C шина рассчитана на передачу данных в виде байтов. Каждый байт передается по шине последовательно, бит за битом. Сигнал данных передается по линии SDA. Передача каждого бита синхронизируется одним импульсом на линии SCL. Этот процесс (см. рис. 3.4) происходит следующим образом. Устройство - передатчик выставляет на шине SDA уровень, соответствующий очередному передаваемому биту (0 или 1). В тот момент, когда завершатся все переходные процессы и уровень сигнала на линии данных примет требуемое значение, на линии синхронизации (SCL) появляется положительный импульс, который и служит для устройства - приемника сигналом, разрешающим чтение бита.

Спецификация сигналов требует, чтобы в процессе передачи битов сигнал на линии SDA изменялся только в тот момент, когда уровень сигнала на линии SCL равен нулю, как это показано на рис. 3.4.

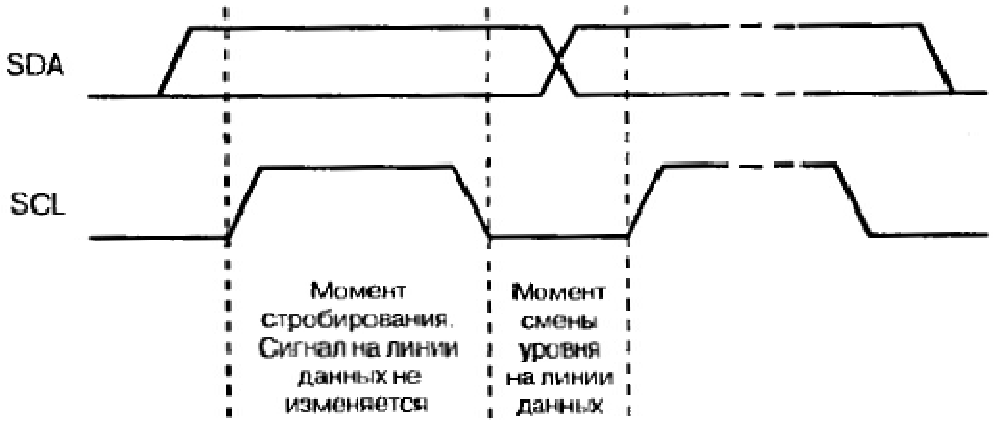


Рисунок 3.4 – Процесс передачи битов

Однако есть два особых случая, когда сигнал на линии SDA изменяется как раз в момент, когда на линии SCL единица. Это происходит при формировании «СТАРТ-условия» (START condition) и «СТОП-условия» (STOP condition). На рис. 3.5 наглядно показано как формируются эти условия.

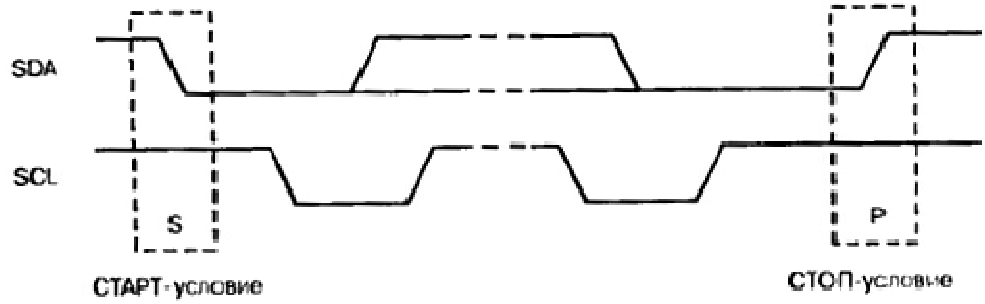


Рисунок 3.5 – Стартовое и стоповое условия

СТАРТ-условие — это отрицательный перепад (с единицы в ноль) на линии SDA при единичном уровне на линии SCL. СТОП-условие — это положительный перепад (от нуля к единице) на линии SDA при единице на линии SCL. В фирменной спецификации I^2C протокола эти условия принято обозначать буквами S и P, как показано на рис. 3.5. СТАРТ и СТОП условия используются для синхронизации пакетов. Передача информации по I^2C шине всегда начинается со СТАРТ-условия и заканчивается СТОП-условием. В середине пакета допускается производить повторный СТАРТ. СТАРТ-условие также используется для правильного разделения передаваемых битов на байты. Сразу после СТАРТ-условия передача байта всегда начинается сначала.

Обратимся к рис. 3.6. На нем показан процесс передачи байта по шине I^2C. Диаграмма, изображенная на рисунке одинаково справедлива как для случая записи данных из ведущего устройства в ведомое, так и для случая чтения данных ведущим устройством из ведомого. Начнем описание этого процесса с первого из вариантов.

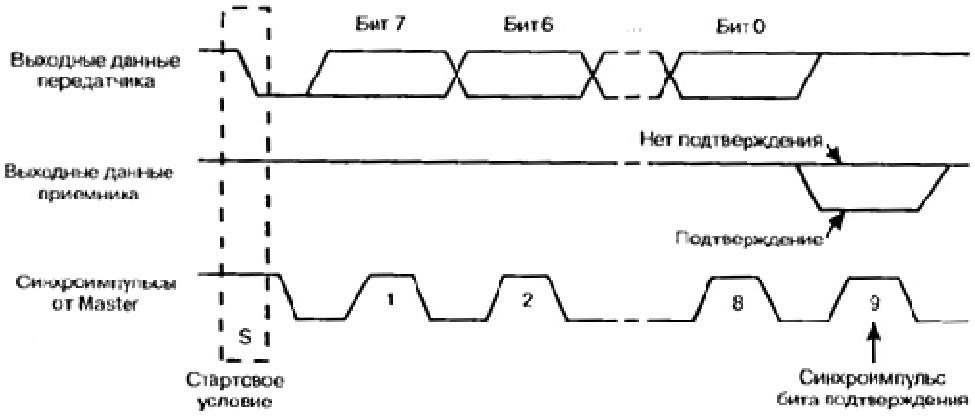


Рисунок 3.6 – Передача данных

Итак, посмотрим, как происходит передача информации от ведущего устройства к ведомому. Очевидно, что при этом ведущее устройство выступает в качестве передатчика, а ведомое в качестве приемника. Для лучшего понимания происходящих процессов сигнал на линии SDA на рис. 3.6 условно изображен в виде двух разных сигналов: выходные данные передатчика и выходные данные приемника. На самом деле эти два сигнала присутствуют на линии одновременно. Они объединены между собой благодаря эффекту схемного «ИЛИ», который образуется при параллельном включении нескольких каскадов с открытым коллектором (стоком). Разделение одного сигнала на два позволяет легче понять, какое из двух устройств вызвало «подсаживание» линии в тот или иной момент времени.Процесс передачи данных всегда начинается со СТАРТ-условия. После чего начинается побитная передача первого байта. Сразу после первого передается второй и так далее. Каждый байт передается за 9 тактов сигнала синхронизации. За восемь первых тактов передаются восемь информационных битов, составляющих байт. Девятый такт используется для сигнала подтверждения. Передача байта производится старшим битом вперед. То есть, вначале передается бит 7, а в конце бит 0. Передача битов производится, таким образом, как показано на рис. 3.4. Затем ведущее устройство «отпускает» линию SDA и формирует на линии SCL еще один импульс синхронизации. Если приемник работает нормально, и успел принять все восемь битов от передатчика, то, получив девятый импульс по линии SCL, он должен сформировать на линии SDA сигнал подтверждения. А именно, он должен перевести линию в ноль (см. рис. 3.6). Приемник проверяет уровень на линии SDA. Если там ноль, то процесс передачи продолжается. Если единица, то программа приступает к процедуре обработки ошибки.

Теперь рассмотрим случай, когда ведущее устройство должно прочитать данные из ведомого. Этот процесс очень похож на предыдущий. Только теперь ведомое устройство является передатчиком, а ведущее — приемником. Несмотря на то, что ведущее устройство выступает в роли приемника, именно оно, как и прежде, занимается формированием СТАРТ и СТОП условий, а также импульсов синхронизации. Процесс считывания данных начинается с того, что ведущее устройство вырабатывает СТАРТ-условие. Сразу после этого ведущее устройство начинает процесс считывания первого байта. Для этого оно вырабатывает импульсы синхронизации на линии SCL.

Ведомое устройство считывает эти импульсы и на каждый из них выдает на линию SDA очередной бит передаваемого байта. Данные

передаются старшим битом вперед. После передачи восьми битов данных ведомое устройство ожидает от ведущего сигнал подтверждения. Ведущее устройство передает сигнал подтверждения в девятом такте.

Если чтение байта прошло нормально, то сигнал подтверждения будет равен нулю. Если произошла ошибка, то сигнал подтверждения равен единице. Ведущее устройство выставляет этот сигнал на шине SDA и затем формирует синхроимпульс на линии синхронизации (SCL). При нормальном завершении процесса считывания байта ведущее устройство сразу же приступает к считыванию следующего байта.

Рассмотрим теперь логику работы PC шины при передаче пакета данных. Процесс передачи пакета показан на рис. 3.7. Как видно из рисунка, процесс передачи цепочки байтов начинается со СТАРТ-условия и заканчивается СТОП-условием. Промежуток между S и P условиями называется посылкой. Кроме того, I^2C протокол допускает применение повторного СТАРТ-условия. Повторное СТАРТ-условие формируется в середине посылки и ничем не отличается от обычного. После повторного СТАРТ-условия мастер может изменить параметры передачи данных, например, может быть назначен другой адрес ведомого устройства или поменяться направление передачи информации. На временных диаграммах повторное СТАРТ-условие обозначается, как Sr.

На рис. 3.7 показана часть посылки между СТАРТ-условием н СТОП-условием, или между двумя СТАРТ-условиями, при передаче информации от ведущего устройства (Master) к ведомому (Slave). Сразу после СТАРТ-условия происходит передача восьми бит информации (первый байт). Затем, на девятом такте, Master получает сигнал подтверждения от Slave.

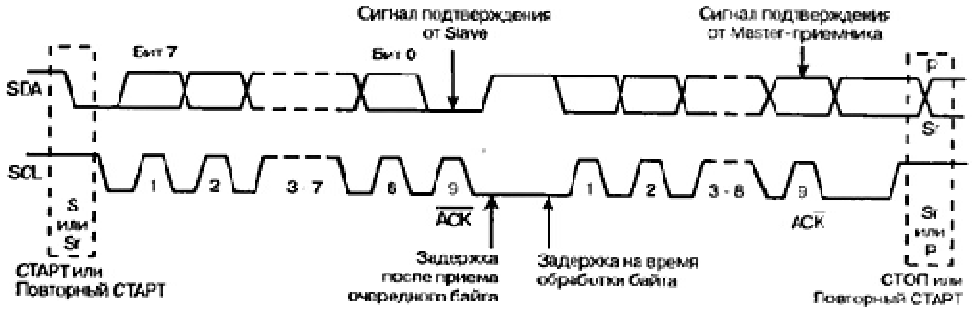


Рис. 3.7 - Структура одной посылки при передаче данных

Реальные устройства после приема байта требуют некоторого времени для обработки полученных данных. Поэтому в I^2C протоколе предусмотрен механизм передачи сигнала готовности. Сигнал готовности передается от

Slave устройства к Master устройству через линию синхронизации (SCL). Получив байт и выдав в линию сигнал подтверждения, Slave устройство «подсаживает» линию SCL и удерживает се в этом состоянии до тех пор, пока не окончит обработку принятого байта. Master устройство ожидает, пока линия SCL будет «отпущена», и все это время не начинает процесс передачи следующего байта. Аналогичным образом передаются и все остальные байты посылки.

Механизм передачи сигнала готовности действует и в случае, когда происходит чтение данных из Slave устройства. Ведущему устройству также может потребоваться время для обработки принятого байта. В этом случае оно просто задерживает очередной синхроимпульс до тех пор, пока обработка байта не закончится.

Таковы основные принципы работы I^2C шины на уровне электрических сигналов. На этом этапе пока не понятно, каким образом происходит выбор режима записи, либо режима чтения. А также не понятно, каким образом происходит передача адреса Slave устройства. Все эти процессы происходят на более высоком уровне протокола — на логическом уровне.

3.4.2 Логический уровень протокола I^2C шины

Принципы работы протокола I^2C шины на логическом уровне иллюстрирует рис. 3.8. Каждый бит, передаваемый по I^2C шине, изображен на рисунке в виде квадратика. Серыми квадратиками изображены биты, передаваемые Master устройством, а светлыми квадратиками — биты, передаваемые Slave устройством. Все условные обозначения приведены в нижней части рисунка.

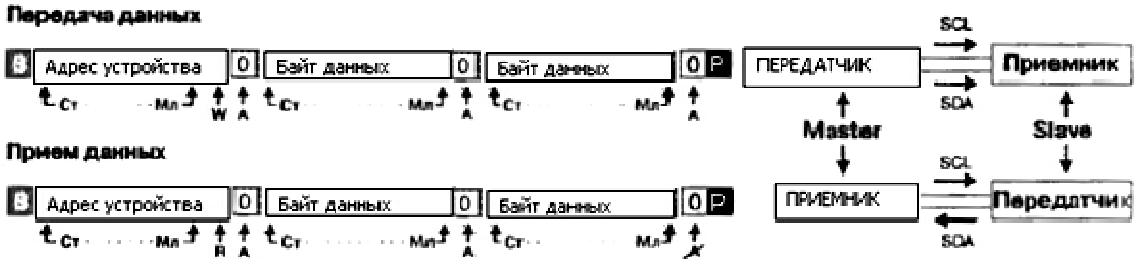


Рис. 3.8 - Диаграмма процессов передачи и приема данных

Верхняя диаграмма изображает процесс передачи данных от Master к Slave (запись данных). Процесс начинается с того, что Master формирует СТАРТ-условие (на рисунке обозначено символом «S»). Затем Master выдает

на шину первый байт (восемь бит информации). Первые семь бит этого байта — это адрес Slave устройства, на которое Master желает передать данные. Восьмой бит называется битом выбора режима и определяет направление передачи информации. Если этот бит равен нулю, то это значит, что Master начинает передачу данных, если единице, то прием.

Как видите, в данном случае бит равен нулю. По этой причине бит выбора режима помечен символом «W» (от английского слова Write — запись). Все Slave устройства, подключенные к шине, получив СТАРТ-условие. принимают первый байт и сравнивают заключенный в нем адрес со своим собственным адресом. Только для одного из Slave устройств условие равенства адресов будет выполнено. Именно это устройство и продолжит дальнейшую работу на шине. Остальные Slave устройства переходят в режим ожидания. Все они будут ждать следующего СТАРТ-условия на шине, после которого они опять выполнят проверку адреса. Устройство, адрес которого совпал, сначала завершает процедуру приема байта. Для этого оно вырабатывает сигнал подтверждения, обозначенное на диаграмме как «A». Получив подтверждение, Master продолжает передачу байтов. Но теперь это уже байты данных, предназначенные для записи в выбранное Slave устройство. Передача каждого байта происходит уже знакомым нам образом. Восемь бит данных, а затем сигнал подтверждения. Таким образом, происходит передача любого количества байтов. Для завершения процесса передачи Master вырабатывает СТОП-условие (обозначено на диаграмме символом «P»).

Нижняя диаграмма на рис. 3.8 иллюстрирует процесс приема данных. Прием данных начинается так же, как и передача. Сначала Master вырабатывает СТАРТ-условие. Затем передает на линию служебный байт. Старшие 7 бит служебного байта, как и в предыдущем случае, означают адрес Slave устройства, из которого Master желает получить данные. Последний (младший) бит теперь равен единице. Это означает, что Master устройство начинает операцию чтения. Поэтому бит выбора режима обозначен на диаграмме символом «R» (от английского слова Read — чтение). Служебный байт, как и в предыдущем случае, принимается всеми устройствами, подключенными к PC шине. И только одно из них останется в активном состоянии.

Выбранное устройство вырабатывает сигнал подтверждения (A), а затем переходит в режим передачи. В этом режиме выбранное устройство один за другим передает байты данных, используя сигналы на линии SCL для синхронизации. В свою очередь, Master устройство принимает эти байты и выдает сигнал подтверждения после каждою из них. В этом режиме работы

сигнал подтверждения используется для выхода из режима передачи. Пока сигнал подтверждения равен нулю, передача продолжается. При получении последнего байта посылки Master устанавливает сигнал подтверждения в единицу. В результате Slave устройство прекращает дальнейшую передачу данных. Завершается весь процесс формированием СТОП-условия. В правой части рис. 3.8 схематически показано направление передачи информации по обеим линиям I^2C шины в разных режимах работы.

Итак, мы теперь получили представление о принципах организации протокола передачи информации посредством двухпроводной I^2C шины. Очевидно, что при создании электронных устройств с применением этой шины необходимо обязательно соблюдать условие, чтобы на одну шину были подключены устройства с разными индивидуальными адресами. Как же определяются адреса для Slave устройств с PC интерфейсом? Иногда I^2C адрес микросхемы просто жестко зашит в нее при изготовлении. В этом случае сам производитель микросхем должен позаботиться, чтобы адреса разных микросхем, предназначенных для применения в одной и той же общей схеме, были бы различными. Если адрес микросхемы жестко зашит в нее при изготовлении, то он обязательно должен быть указан в прилагаемой документации. Например, микросхема TDA8844 (однокристальный телевизор фирмы Philips) имеет PC адрес для записи 10001010 (или 138 в десятичном представлении). Для чтения, соответственно, адрес будет равен 10001011 (139 десятичное). В данном случае под адресом подразумевается значение всего служебного байта, вместе с битом «чтение/запись». В других случаях в микросхему зашита лишь часть адреса. А остальная часть адреса определяется при помощи специальных адресных входов, на которые подаются разные комбинации напряжения. Микросхемы энергонезависимой памяти, которые будут описаны далее в этой главе — пример именно таких устройств, у которых часть адреса фиксирована, а другая часть определяется при помощи адресных входов.

3.5 Микросхемы EEPROM с I^2C интерфейсом

Хорошим примером применения I^2C шины являются микросхемы энергонезависимой памяти. Такие микросхемы имеют и второе название: Flesh-память (флэш-память). Что же такое флэш-память? Задача надежного сохранения информации после выключения электропитания давно стояла перед разработчиками. Микросхемы памяти с момента своего появления делились на оперативные запоминающие устройства (ОЗУ) и постоянные запоминающие устройства (ПЗУ). ОЗУ — это такое устройство, информацию в котором можно оперативно изменять. Однако при выключении питания она не сохраняется. В ПЗУ информация хранится даже после выключения питания.

Однако в первых моделях ПЗУ информация записывалась однократно, в процессе изготовления. Позже появились технологии, позволяющие «перепрошивать» ПЗУ. Но этот процесс был довольно длительным и требовал извлечения микросхемы из устройства, где она использовалась и подключения к специальному программатору.

Долго не удавалось разработать технологию, позволяющую оперативно изменять информацию в ПЗУ. Наконец была изобретена технология ПЗУ с электрическим стиранием информации (ренрограммируемые ПЗУ). В английской транскрипции такие микросхемы называются EEPROM. Именно эти устройства и получили второе название: флэш-память. В современных микросхемах флэш-памяти время стирания информации составляет 5...10 мс. Это намного медленнее, чем процесс записи в ОЗУ, однако для устройств долговременного хранения данных вполне приемлемо. Количество циклов записи/стирания для флэш-памяти ограничено, однако на сегодняшний день эта цифра достигает миллиона.

Микросхемы флэш-памяти широко применяются в микропроцессорной технике. Например, память программ микроконтроллера AT89C205I использует технологию флэш-памяти. Существует также широкая номенклатура различных микросхем автономной флэш-памяти. Такие микросхемы для связи с микроконтроллером могут использовать самые различные виды интерфейса. Есть микросхемы, использующие параллельный способ подключения. Однако нас в данном случае будут интересовать микросхемы флэш-памяти с 1^2C интерфейсом. Такие микросхемы получили широкое применение в современной электронной аппаратуре. Ведущим разработчиком микросхем флэш-памяти по праву можно считать фирму Atmcl. Она выпускает целую серию микросхем энергонезависимой памяти с РС интерфейсом. Эта серия получила название AT24Cxx. Здесь xx — это две

или даже три цифры, которые обозначают номер модели микросхемы. Обычно он соответствует объему памяти микросхемы, выраженному в килобитах. В табл. 3.1 приведены основные характеристики микросхем этой серии. Родоначальником серии можно считать микросхему AT24C01. По сравнению с другими микросхемами, эта микросхема имеет самую простую внутреннюю структуру. Микросхема содержит всего 128 ячеек памяти, каждая из которых может хранить один байт информации. Объем памяти в килобитах будет равен 128 * 8 = 1024, что соответствует 1 Кбит. Для нас привычнее отсчитывать объем памяти в байтах и килобайтах. Поэтому в табл. 3.1 приведены оба варианта. При подключении микросхемы AT24C01 к PC шине она одна выступает в качестве 128 Slave устройств. Каждая ячейка памяти — свое отдельное устройство. По этой причине на одной шине может одновременно присутствовать только одна микросхема типа AT24C01. То есть процессор просто соединяется с микросхемой при помощи двух проводников, к каждому из которых нужно не забыть подключить сопротивление нагрузки. Микроконтроллер при этом, естественно, является Master.

Таблица 3.1 – Семейство микросхем AT24Cxx фирмы Atmel

Тип микросхемы	Объем памяти		Организация памяти		Макс. кол-во на шине	Входы установки адреса	Формат служебного байта								Формат адреса памяти		Максим. адресуемая память (байтов)
	битов	байтов	Страниц	Байт на страницу			7	6	5	4	3	2	1	0	байтов	битов	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15	17	18
AT24C01	1K	128	32	4	1	нет	Адресует ячейку памяти							R/W	нет	-	128
AT24C01A	1K	128	16	8	8	A0, A1, A2	1	0	1	0	A2	A1	A0	R/W	1	7	1K
AT24C02	2K	256	32	8	8	A0, A1, A2	1	0	1	0	A2	A1	A0	R/W	1	8	2K
AT24C04	4K	512	32	16	4	A1, A2	1	0	1	0	A2	A1	P0	R/W	1	8	2K
AT24C08	8K	1K	64	16	2	A2	1	0	1	0	A2	P1	P0	R/W	1	8	2K
AT24C16	16K	2K	128	16	1	нет	1	0	1	0	P2	P1	P0	R/W	1	8	2K
AT24C164	16K	2K	256	8	8	A0, A1, A2	1	A2	A1	A0	P2	P1	P0	R/W	1	8	16K
AT24C32	32K	4K	128	32	8	A0, A1, A2	1	0	1	0	A2	A1	A0	R/W	2	12	32K
AT24C64	64K	8K	256	32	8	A0, A1, A2	1	0	1	0	A2	A1	A0	R/W	2	13	64K
AT24C128	128K	16K	256	64	4	A0, A1	1	0	1	0	0	A1	A0	R/W	2	14	64K
AT24C256	256K	32K	512	64	4	A0, A1	1	0	1	0	0	A1	A0	R/W	2	15	128K
AT24C512	512K	64K	512	128	4	A0, A1	1	0	1	0	0	A1	A0	R/W	2	16	256K
AT24C1024	1M	128K	512	256	2	A1	1	0	1	0	0	A1	P0	R/W	2	16	256K
AT24C21 *	1K	128	128	1	1	нет	1	0	1	0	***	***	***	R/W	1	7	128
AT24CS128 **	128K	16K	256	64	4	A0, A1	1	0	1	0	0	A1	A0	R/W	2	14(15)	64K

Для того, чтобы записать байт данных в ячейку памяти с адресом N, микроконтроллер должен выполнить стандартную процедуру передачи информации по PC шине от Master к Slave. При этом Slave будет иметь адрес N. Передачу данных в этом режиме в точности отображает диаграмма,

приведенная на рис. 3.8. Следует заметить, что запись во флэш-память — это относительно медленный процесс. Для микросхемы AT24C01 время записи может достигать 10 мс.

Для того, чтобы ни задерживать работу шины, микросхема AT24C01 имеет специальную буферную память. В режиме записи информация она сначала попадает в буферное ОЗУ. При этом скорость передачи информации равна 100 Кбит/с. А в некоторых режимах она может быть увеличена до 400 Кбит/с. Сразу после того, как Maser сформирует СТОП-условие, микросхема памяти перейдет в режим переноса информации из буфера непосредственно во флэш-память. До тех пор, пока этот процесс не закончится, микросхема AT24C01 не выдаст сигнал готовности.

Большое время записи информации обусловлено особенностями EEPROM технологии. Прежде чем записать новое значение в любой из ячеек памяти, система управления микросхемы должна произвести операцию по электрическому стиранию предыдущей информации. Это сложный процесс, связанный со структурными изменениями в материале подложки.

Если нужно записать несколько ячеек подряд, не обязательно обращаться к каждой из них отдельно. Для этого в микросхеме предусмотрен страничный способ записи. При страничной записи Master в одном сеансе передает по PC шине не один, а несколько байт. В качестве Slave адреса он указывает адрес первой ячейки памяти, с которой и будет начинаться запись. В результате данные будут записаны последовательно в несколько ячеек. После записи каждого очередного байта текущий адрес, хранящийся в специальном внутреннем регистре микросхемы памяти, автоматически увеличивается на единицу. Однако изменяются только два младших разряда адреса. Пять старших разрядов остаются без изменения. Поэтому длина страницы в режиме страничной записи для микросхемы AT24C01 равна четырем. Это значит, что максимальное количество байтов, которые можно передать за один сеанс — четыре байта.

Вся память микросхемы, таким образом, разделена на 32 страницы по 4 байта в каждой. Программа управляющего микроконтроллера должна учитывать эту особенность. Важно понять, что при страничном способе записи, за один прием можно переписать не более четырех байтов. Причем все они всегда будут находиться в пределах одной страницы. Если вы неправильно составите программу, и ваш микроконтроллер за один раз попытается записать в микросхему памяти цепочку байт, которая выходит за пределы страницы, то, дойдя до конца страницы, запись начнется с ее начала.

Например, для передачи блока из четырех байтов необходимо обязательно начинать запись с первого байта одной из страниц. Если вы все

же решили начинать со второго байта страницы, то безболезненно передать можно будет только три байта. И так далее. Сама микросхема AT24C01 не выполняет никаких проверок для обнаружения неправильного использования страниц памяти.

Вы спросите: зачем же такие сложности? Дело в том, что стирать каждую ячейку в отдельности дело довольно сложное и дорогое. Поэтому производится стирание сразу нескольких ячеек. Эти несколько ячеек и составляют страницу памяти. Даже если вы переписываете не всю страницу, а лишь ее часть, все равно соответствующая часть электрически стираемой памяти очищается. Содержимое всех остальных ячеек (которые не надо переписывать) предварительно сохраняется в буферной памяти. Затем, после окончания процесса стирания, все байты, составляющие страницу, записываются назад. При этом нужные байты получают новое значение, а в остальных остается старое.

Другие микросхемы из серии AT24Cxx также имеют страничную организацию памяти. В табл. 3.1 в графе 3 показаны данные о структуре страниц для каждого типа микросхем.

В режиме чтения страничные ограничения отсутствуют. Процесс чтения для микросхемы AT24C01 полностью описывается соответствующей диаграммой на рис. 3.8. Причем адрес Slave устройства, передаваемый в первом (служебном) байте посылки, соответствует адресу той ячейки памяти, с которой начинается процесс чтения. Микроконтроллер может читать как один байт за сеанс, так и любое количество байтов подряд. Максимальное количество прочитанных байтов равно полному объему используемой памяти. Для микросхемы AT24C01 эта величина будет равна 128 байтам. Если попытаться за один раз прочитать большее количество байтов, то микросхема выдаст все содержимое своей памяти, а затем начнет выдавать его сначала.

Микросхема AT24C01 в настоящее время используется очень редко. Объема памяти в 128 ячеек для современных электронных приборов совершенно недостаточно. Однако используемый способ адресации ячеек не позволяет увеличить их количество. Чтобы не изменять уже сложившийся протокол для 1^2C шины и получить возможность для адресации большого количества ячеек памяти, фирма Atmel разработала несколько вариантов расширения этого протокола. Все эти расширения полностью совместимы с основным стандартом снизу вверх. Расширенных протоколов всего два. Оба они предусматривают введение дополнительных служебных байтов. В первом варианте добавляется только один дополнительный служебный байт. С применением такого протокола появилась возможность адресовать до 2

Кбайт памяти. Фирма Atmel первоначально выпустила целую линейку микросхем, использующих этот протокол: AT24C01A, AT24C02, AT24C04, AT24C08, AT24C16. Основные характеристики этих микросхем вы можете видеть в табл. 3.1. Первая микросхема этой линейки — AT24C01A. Объем ее памяти не отличается от объема памяти микросхемы AT24C01 и тоже равен 128 байтам. Объем памяти каждой последующей модели описываемой линейки в два раза больше предыдущего (см. табл. 3.1). Адресация ячеек памяти происходит совершенно не так как в микросхеме AT24C01. Главное отличие нового способа адресации — возможность подключения нескольких микросхем памяти на одну PC шину. Первый служебный байт I^2C протокола предназначается для выбора одной из нескольких, подключенных к шине микросхем. Для того, чтобы каждому конкретному экземпляру микросхемы AT24C01A можно было бы присвоить свой индивидуальный адрес, она имеет специальные входы адреса: AO, Al, A2. Эти входы предназначены для задания трех младших разрядов адреса микросхемы. Для задания адреса каждый из этих входов нужно подсоединить либо к общему проводу, либо к напряжению питания (V_{pp}). Если разряд должен быть равен нулю, то его подсоединяют к общему проводу. Если единице — к питанию. Четыре старших разряда адреса фиксированы. Их значение всегда равно 1010B. Структура основного служебного байта, в который заложен Slave адрес микросхемы, приведена в табл. 3.1 (столбцы 8... 15). Очевидно, что при таком способе адресации на одной PC шине могут одновременно работать до восьми микросхем памяти.

На рис. 3.9 показана схема подключения микросхем AT24C01A к микроконтроллеру AT89C2051. Суммарный объем памяти при подключении микросхем AT24C01A по такой схеме составляет 1 Кбайт. Это максимально возможный объем памяти при использовании именно этого вида микросхем.

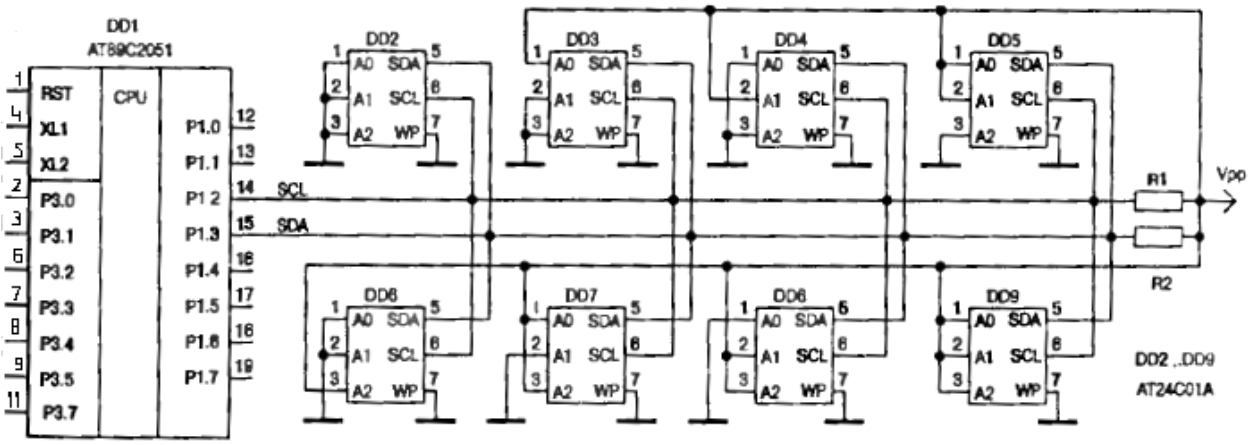


Рисунок 3.9 – Схема подключения EEPROM типа AT24C01A к микропроцессору AT89C2051

Максимально возможный объем памяти для других видов микросхем приведен в столбце 18 табл. 3.1. При помощи перемычек на входах АО, Al, А2 каждой из микросхем памяти присваиваются свой индивидуальный адрес. В табл. 3.2 приведены адреса всех микросхем, изображенных на рис. 3.9. Адрес каждой микросхемы приведен в двоичном и шестнадцатеричном видах.

Таблица 3.2 – Адреса микросхем памяти, включенных по схеме на рис. 3.9

Микросхема		DD2	DD3	DD4	DD5	DD6	DD7	DD8	DD9
Адрес	BIN	1010000B	1010001B	1010010B	1010011B	1010100B	1010101B	1010110B	1010111B
	HEX	50H	51H	52H	53H	54H	55H	56H	57H
Служебный байт в режиме записей	BIN	10100000B	10100010B	10100100B	10100110B	10101000B	10101010B	10101100B	10101110B
	HEX	0A0H	0A2H	0A4H	0A6H	0A8H	0AAH	0ACH	0AEH

Для адресации ячеек памяти внутри каждой микросхемы применяется дополнительный служебный байт. С точки зрения стандартного протокола — это первый байт данных. Однако для микросхемы памяти этот байт имеет смысл адреса ячейки памяти. Диаграмма, поясняющая принципы работы расширенного 1^2C протокола с одним дополнительным байтом, приведена на рис. 3.10. Если вы посмотрите внимательнее на рисунок, вы без труда поймете эти принципы.

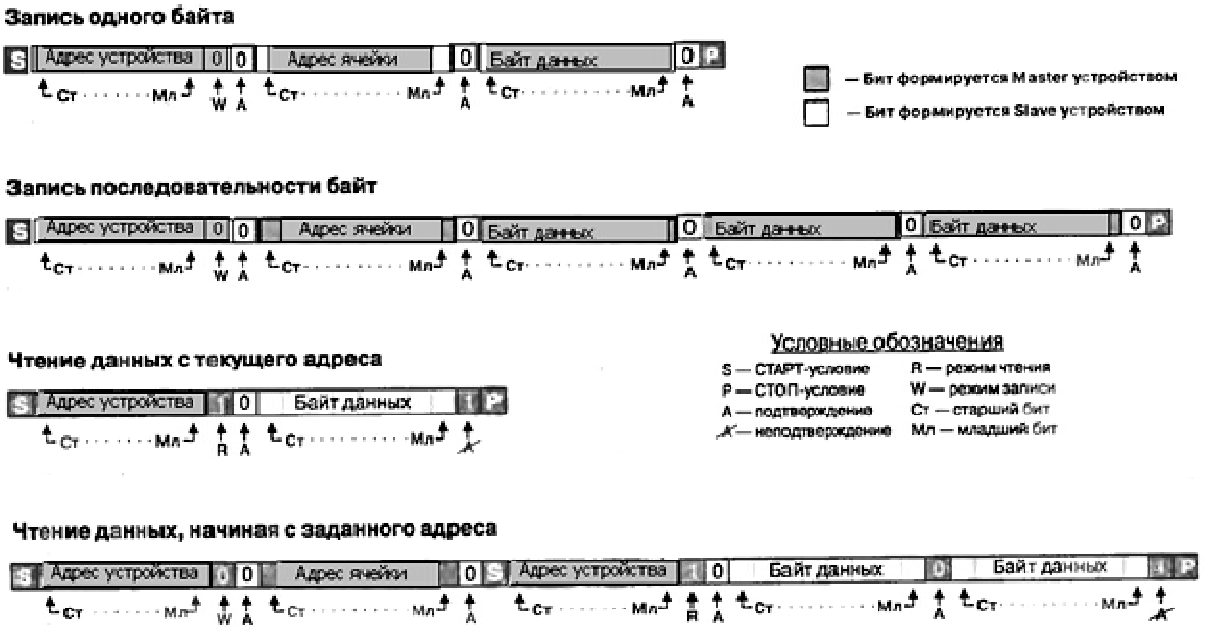


Рисунок 3.10 – Однобайтовый способ адресации ячеек памяти EEPROM

Описываемый протокол используют все микросхемы описываемой линейки. Микросхема AT24C01A содержит всего 128 ячеек памяти. Для их адресации достаточно лишь семи разрядов. Поэтому старший разряд дополнительного служебного байта в этой микросхеме не используется. Микроконтроллер может устанавливать в этом разряде любое значение. На выбор ячейки памяти это не повлияет. В табл. 3.1 в столбцах 16 и 17 приведено количество дополнительных служебных байтов для разных видов микросхем, а также количество задействованных разрядов.

Адрес ячейки, передаваемый в дополнительном служебном байте, запоминается в специальном внутреннем регистре микросхемы памяти. При записи последовательности из нескольких байтов, после каждого очередного байта этот адрес автоматически увеличивается на единицу. Значение адреса после записи последнего байта цепочки сохраняется во внутреннем регистре микросхемы памяти до тех пор, пока он не будет переписан новой командой записи. Этот адрес называется текущим адресом. Сохранение текущего адреса позволяет в любой момент продолжить прерванный процесс чтения памяти. Процесс чтения всегда начинается с текущего адреса. После чтения каждого очередного байта текущий адрес также увеличивается. Если перед началом сеанса чтения требуется изменить текущий адрес, нужно просто произвести запись адреса. Режим записи адреса отличается от обычного режима записи только тем, что на запись передается только один дополнительный служебный байт. Этот процесс изображен на рис. 3.10 на диаграмме «Чтение данных, начиная с заданного адреса».

3.6 Программная реализация I^2C интерфейса

В предыдущем разделе мы подробно рассмотрели протоколы для всех режимов работы микросхем флэш-памяти серии AT24Cxx. В настоящей главе нам предстоит познакомиться с примерами программ, реализующих эти протоколы. Фирма Atmel очень ответственно подходит к вопросу технической поддержки своей продукции. Существует также и набор подпрограмм, специально разработанных для работы с микросхемами AT24Cxx серии. Вполне естественно, что существует несколько вариантов таких программ, написанных для разных типов микроконтроллеров. Мы рассмотрим типовой пакет подпрограмм для работы с микросхемами флэш-памяти, имеющими I^2C интерфейс. Этот вариант программ разработан фирмой Atmel специально для микроконтроллера AT89C2051. В свое время эти программы были скачаны автором с сайта Atmel и адаптированы под ту версию ассемблера, на которой написаны все программные примеры, приведенные в настоящей книге. Применение программ, рекомендованных изготовителем микросхем, гарантирует надежность и качество работы. Приведенные ниже подпрограммы опробованы на практике. Именно эти подпрограммы используются, например, для записи информации во внешнюю флэш-память позиционера спутниковой антенны.

В процессе реализации протокола необходимо отрабатывать различные временные интервалы. В приведенных примерах подпрограмм все временные интервалы формируются программным путем. Чаше всего, для формирования задержки программа выполняет несколько «пустых» команд. Время задержки при этом будет зависеть от количества пустых операторов и от частоты кварцевого резонатора. При разработке подпрограмм предполагалось, что кварцевый резонатор должен иметь резонансную частоту 12 МГц.

4 ШИНА MICROLAN

4.1 Общие сведения

В предыдущей главе мы познакомились с двухпроводной шиной для последовательной передачи данных I^2C. Эта шина хорошо подходит для передачи сигналов управления между микросхемами, расположенными на одной и той же плате. В крайнем случае, I^2C шину можно применить в пределах одного блока. Максимальная протяженность I^2C шины не может превышать 10 м. Однако перед разработчиками микропроцессорных устройств часто ставится более сложная задача: соединить микросхемы, находящиеся на более значительных расстояниях друг от друга.

Шина 1-Wire является основой сетей MicroLAN и разработана в конце 90-х годов фирмой Dallas Semiconductor. В настоящее время фирма Dallas Semiconductor является дочерним предприятием фирмы MAXIM. Микросхемы и комплектующие фирмы MAXIM широко известны разработчикам электронных устройств. Логотип этой фирмы приведен на рис. 4.1.

Рисунок 4.1 – Логотип фирмы Dallas Semiconductor

Создавая шину MicroLAN, фирма Dallas Semiconductor поставила перед собой, казалось бы, неразрешимую задачу. Идея состояла в том, чтобы соединить между собой множество различных микросхем, расположенных на значительном расстоянии друг от друга, используя при этом всего один сигнальный провод. Разумеется, кроме сигнального провода для замыкания цепи обязательно должен быть и обратный так называемый «общий провод». Все микросхемы должны подключаться к такой двухпроводной шине параллельно. И вот по этой линии, состоящей всего из двух проводников от одной микросхемы к другой, должна передаваться информация, как в прямом, так и в обратном направлении. Кроме того, в случае необходимости, по той же однопроводной шине решено было осуществлять питание всех

подключенных к ней микросхем. Сама постановка такой задачи уже была достаточно дерзкой. Фирма Dallas Semiconductor блестяще справилась с этой задачей. Шина 1-Wire и основанные на ней сети MicroLAN давно с успехом применяются в электронной технике.

Ниже приведены основные характеристики этой шины:

- Максимальная протяженность шины до 300 м.
- Скорость передачи информации 16,3 Кбит/с.
- Максимальное количество адресуемых элементов на шине 256.
- Уровни напряжений на шине соответствуют стандартным КМОП/ТТЛ уровням.
- Напряжение питания компонентов сети 2,8...6 В.
- Для соединения элементов сети может применяться обычный телефонный кабель или витая пара.

Существуют и модификации шины 1-Wire. Например, отдельные виды микросхем поддерживают скоростной режим работы шины (Overdrive). В этом режиме скорость передачи информации равна 142 Кбит/с. Однако такие микросхемы могут работать только на шине малой протяженности и при условии, когда уровень внешних электрических помех сведен к минимуму.

Шина MicroLAN, так же как и I^2C шина, построена по технологии Master/Slave. На шине должно быть хотя бы одно ведущее устройство (Master). Все остальные устройства должны быть ведомыми (Slave) Ведущее устройство инициирует все процессы передачи информации в пределах шины. Master может прочитать данные из любого Slave устройства или записать их туда. Передача информации от одного Slave к другому напрямую невозможна. Для того, чтобы Master мог обращаться к любому из ведомых устройств по шине, каждое ведомое устройство содержит в себе индивидуальный код (ID-код). Этот код заносится в специальную область микросхемы при помощи лазера. Фирма-изготовитель гарантирует, что этот код никогда не повторится и все когда-либо изготовленные микросхемы, содержащие l-Wire интерфейс, всегда будут иметь разные коды.

Еще одним полезным свойством шины является возможность автоматического обнаружения факта подключения новых компонентов. Протокол l-Wire включает в себя специальную команду поиска, при помощи которой ведущее устройство (Master) может осуществлять автоматический поиск ведомых устройств. В процессе поиска Master определяет ID коды для всех подключенных к сети микросхем. Поиск происходит путем постепенного отсеивания несуществующих адресов. Поэтому для того, чтобы найти все

устройства, подключенные к шине (их еще называют узлами сети), требуется довольно значительное время. Средняя скорость поиска элементов в сети MicroLAN составляет примерно 75 узлов в секунду.

При разработке протокола 1-Wire большое внимание было уделено надежности работы сети. Изначально было поставлено условие — работа должна происходить в условиях плохих контактов. Кроме того, допускается подключение и отключение ведомых элементов прямо в процессе работы. Для обеспечения надежной и устойчивой работы всех элементов в сети MicroLAN используется механизм двухэтапной записи информации с применением промежуточной, так называемой «блокнотной памяти». А также методы контроля целостности передаваемой информации методом контрольных сумм. В свое время об этом мы поговорим подробнее.

Идеальным применением для сети MicroLAN является система телеметрии и автоматики в производственных или бытовых помещениях. Представьте себе, что в некоем здании по всем помещениям в разных местах установлены миниатюрные датчики температуры, влажности, освещенности и др. Все они соединены между собой одной двухпроводной линией и подключены к единому управляющему устройству (см. рис. 4.2). Этим устройством может быть персональный компьютер, снабженный специальной программой, либо специально разработанный блок на основе микроконтроллера. Кроме того, по всем помещениям рассредоточены миниатюрные исполнительные устройства. Они могут включать и выключать обогреватели, осветительные приборы, вентиляторы, электрические замки на дверях, зуммеры, механизмы открывания штор и любые другие, электрические и неэлектрические приборы, расположенные в разных помещениях вашего дома.

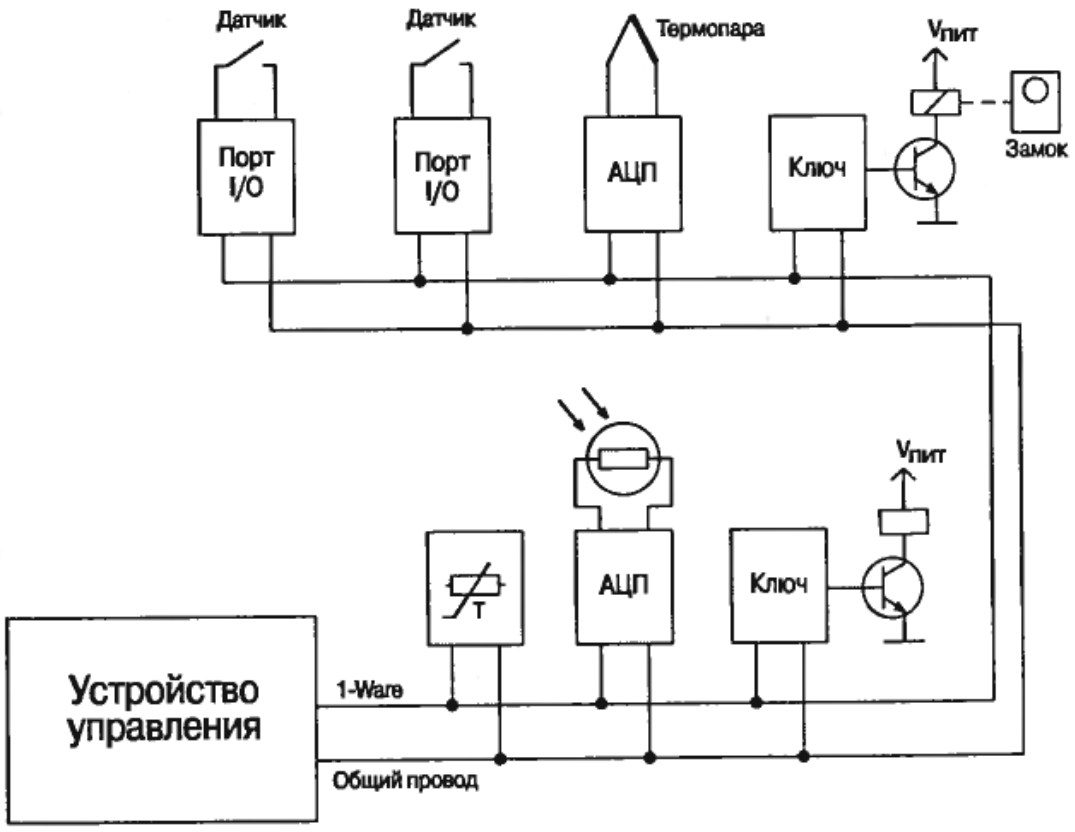


Рисунок 4.2 – Топология сети MicroLAN

Все исполнительные механизмы также подключаются к той же самой двухпроводной шине. Конечно, тс устройства, которые требуют значительной мощности для своего питания, не могут получать его по шине 1-Wire. Они должны иметь свои автономные источники питания. Те же микросхемы, которые потребляют незначительную мощность (например, различные датчики), с успехом могут получать питание от той же двухпроводной линии, через которую происходит обмен информацией. Управляющее устройство объединяет все подключенные к ней элементы. Оно должно содержать программу для централизованного управления всей этой системой. В рамках сети MicroLAN можно организовать глобальную систему управления всеми бытовыми устройствами и механизмами в вашем доме, или системами комфорта и жизнеобеспечения на любом предприятии.

При разработке сетей MicroLAN фирма Dallas Semiconductor позаботилась о том, чтобы обеспечить самый широкий набор вариантов ее построения. Например, в качестве Master устройства может выступать как специализированный контроллер MicroLAN, так и персональный компьютер, а также любой универсальный микроконтроллер, работающий на тактовых частотах превышающих 1,8 МГц. Для согласования шины MicroLAN с

компьютером фирма Dallas Semiconductor разработала специальные микросхемы — преобразователи интерфейса. Такие устройства могут подключаться как к LPT порту компьютера, так и к любому его COM порту и работать в качестве моста, согласующего порты компьютера с шиной 1-Wire. Существует даже вариант реализации 1-Wire интерфейса непосредственно на выводах COM порта, без применения дополнительных элементов. Этот способ возможен только в том случае, если в вашем компьютере для реализации последовательного COM интерфейса используется микросхема Intel 8250. Протокол 1-Wire в этом случае реализуется программным путем.

Иногда при построении сети MicroLAN использование топологии, состоящей всего из одной 1-Wire шины, оказывается недостаточно. Ограничения могут быть вызваны слишком большой суммарной емкостью или большим суммарным током потребления всех подключенных к шине устройств. Специально для таких случаев в стандарте сетей MicroLAN предусмотрена возможность ветвления. Для ветвления шины 1-Wire применяется специальные микросхемы — управляемые разветвители. Разветвитель управляется по той же самой 1-Wire шине, которую он же и коммутирует. На рис. 4.3 показана схема подключения дополнительной ветви сети при помощи микросхемы разветвителя типа DS2409. Встроенный 1-Wire интерфейс принимает команды из сети и управляет электронным коммутатором К и схемой управления (Упр). По команде из ведущего устройства шина «ответвление» может быть подключена или отключена от основной шины. Если ответвление подключено к основной шине, то эти две шины представляют собой одно целое.

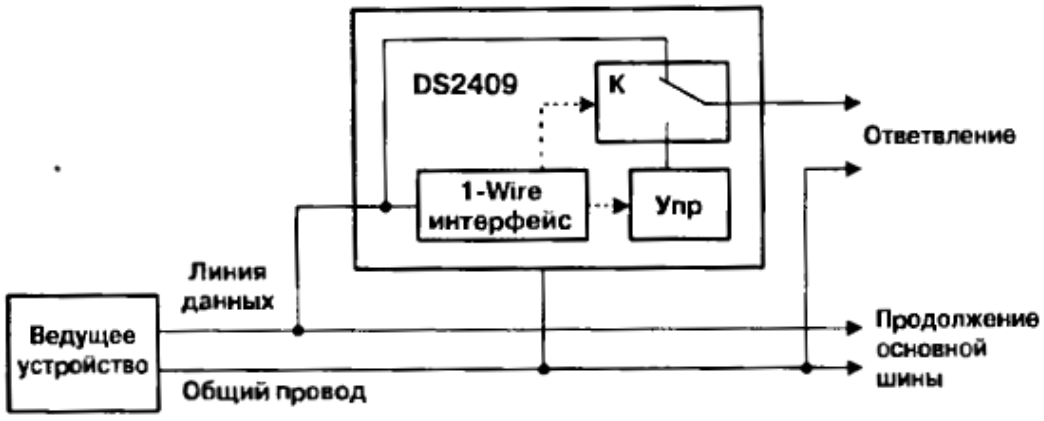


Рисунок 4.3 – Ответвление сети

Команды, посланные ведущим устройством, распространяются на все ведомые устройства, подключенные к обеим ветвям. Когда шина-

ответвление отключается от основной шины, она подключается к специальной управляющей схеме внутри микросхемы DS2409. Управляющая схема способна по команде от ведущего устройства передавать информацию в шину «ответвление», выступая в роли управляемого Master устройства для дополнительной шины. При таком методе управления прямое соединение двух этих шин отсутствует. Однако передача сигналов возможна, хотя алгоритм управления при этом усложняется.

Итак, микросхема-разветвитель позволяет сделать в любом месте однопроводной шины управляемое ответвление. Шина может содержать любое количество ответвлений. При этом структура сети будет напоминать дерево (см. рис. 4.4). Древовидная структура позволяет включать различные ветви сети по мере надобности. Остальные ветви временно отключаются и не создают дополнительной нагрузки.

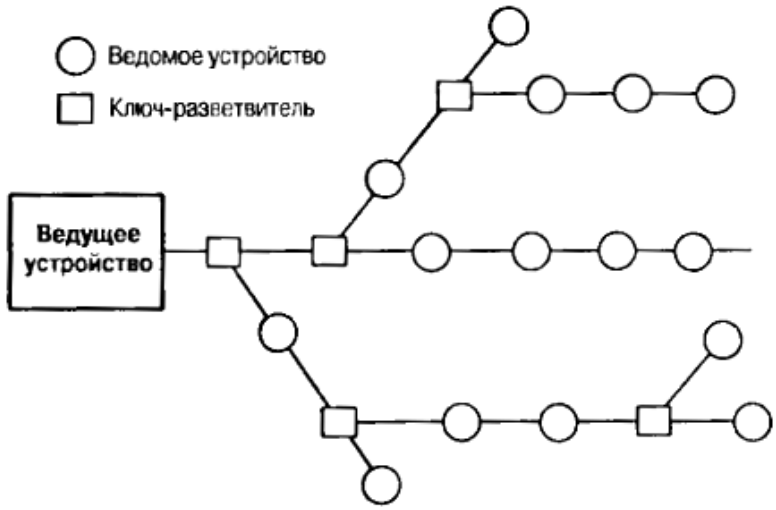


Рисунок 4.4 – Древовидная структура

Как можно было убедиться из всего вышесказанного, шина MicroLAN имеет огромные возможности и большие перспективы. Однако тема данной книги, если помните — подключение периферийных устройств к микроконтроллеру. l-Wire интерфейс прекрасно подходит для этой цели. Микроконтроллер AT89C2051 позволяет легко, программным путем реализовать 1-Wire интерфейс на любой из своих линий ввода/вывода.

Долгое время шина MicroLAN не получала широкого распространения из-за того, что номенклатура микросхем, оснащенных 1-Wire интерфейсом, была не слишком велика. Однако в последнее время фирма Dallas

Semiconductor разработала множество новых типов микросхем. Для того, чтобы вы могли почувствовать весь диапазон возможностей этой технологии, приведем краткий обзор производимых в настоящее время 1-Wire совместимых микросхем. Все микросхемы с 1-Wire интерфейсом можно разделить на две большие группы:

1. Микросхемы, предназначенные для монтажа при помощи пайки.
2. Микросхемы в специальном, мобильном исполнении, получившие название iButton.

4.2 Новый класс микросхем — iButton

Фирма Dallas Semiconductor по праву гордится тем, что является изобретателем нового вида микросхем. Этот вид микросхем даже получил специальное название — iButton, которое также является зарегистрированной торговой маркой. На рис. 4.5 показан логотип торговой марки iButton.

Рис. 4.5 - Торговая марка iButton

Что же представляют собой эти микросхемы? Главная их особенность - новое конструктивное решение. Корпус этих микросхем имеет форму таблетки (см. рис. 4.6). До сих пор в подобных корпусах выпускались только миниатюрные источники питания.

Рисунок 4.6 – Микросхемы iButton

Рисунок 4.7 – Микросхема изнутри

Новый корпус для микросхем получил название MicroCan. Корпус выпускается в двух модификациях, которые отличаются друг от друга по толщине. На рис. 4.8 приведены основные размеры микросхем в корпусе MicroCan.

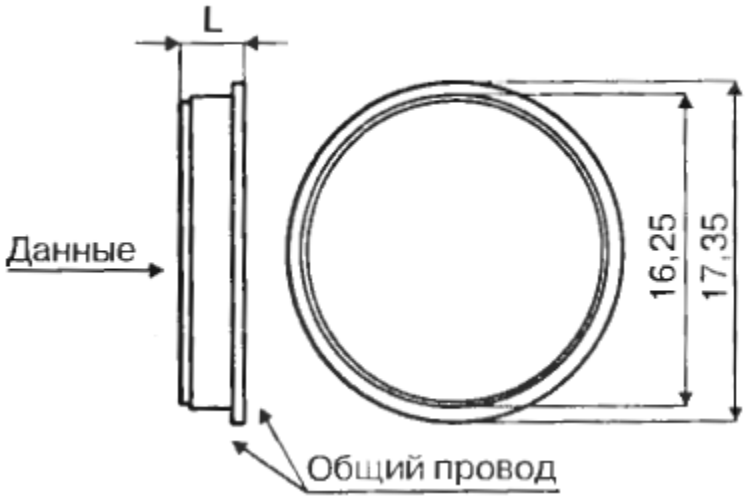


Рисунок 4.8 – Корпус MicroCan

Диаметр корпуса для обеих модификаций одинаковый и равен 16,25 мм. Толщина корпуса L для модификации F3 равна 3,1 мм, а для модификации F5 — 5,8 мм. В одних моделях микросхем в корпусе размещается лишь кремниевая пластина (см. рис. 4.7). В других же, кроме кремниевой пластины, может находиться миниатюрный литиевый источник питания, и даже кварцевый резонатор.

Микросхема, заключенная в такой корпус, всегда имеет только два вывода — это верхняя и нижняя обкладки корпуса. Такая микросхема не предназначена для пайки. Для подключения микросхем в корпусе MicroCan используются специальные адаптеры. Микросхема в любой момент легко

вставляется в адаптер и извлекается из него. Металлический корпус микросхемы делает ее исключительно прочной и надежной в эксплуатации. Микросхема выдерживает удар в 0,5 кг, падение на пол с высоты до 1,5 метра. Допустимый диапазон температур для микросхем iButton: от -40 до +85 С.

Какие же функции могут выполнять микросхемы iButton? Оказывается можно придумать множество применений микросхемам, имеющим всего два вывода. Перечислим все по порядку. Элементы контроля доступа. Такая микросхема не содержит ничего, кроме своего собственного уникального 64-разрядного ID кода. Применяются они как ключ. Разработано множество систем с использованием микросхем iButton в качестве ключа. Существуют дверные замки, имеющие вместо замочной скважины гнездо для подключения микросхемы. Системы защиты компьютерных программ для подключения микросхем iButton используют специальные кабели. Такой кабель подключается к LPT или COM порту компьютера. На втором конце такого кабеля имеется одно или два гнезда для подключения ключевой микросхемы. Специально раз- работанные программы проверяют наличие микросхемы в разъеме и соответствие ее внутреннего кода коду владельца программы. При несовпадении кодов доступ к программе блокируется.

Микросхемы памяти. Существует несколько видов таких микросхем. Это однократно записываемая память (EPROM), многократно записываемая память с электрическим стиранием (EEPROM), а также энергонезависимое ОЗУ со встроенным источником питания (NV RAM). Память может использоваться для хранения и переноса с компьютера на компьютер любой информации: изображений, музыки, текстовой информации. Кроме того, существует специальный вид защищенной памяти, которая используется в качестве электронного кошелька, для мелких сумм. Существует также па-мять, защищенная паролем.

Цифровой термометр. Служит для измерения температуры в цифровом виде. Измеренную информацию можно считывать по l-Wire интерфейсу. К цифровому термометру мы еще вернемся в конце этой главы и рассмотрим этот вопрос очень подробно.

Часы, календари реального времени. Содержат в себе миниатюрный источник питания и микросхему, выполняющую непрерывный подсчет времени. Считывание и установка этого времени возможна только при помощи 1-Wire интерфейса. Кроме задачи подсчета времени в микросхеме реализован календарь, секундомер, измеритель времени наработки подключенного к контактам микросхемы внешнего устройства и др. Кроме

самостоятельных таймеров производятся и комбинированные устройства. Первый такой гибрид — таймер, совмещенный с энергонезависимым ОЗУ. Такое совмещение позволяет ограничить доступ к хранящейся в ОЗУ микросхемы информации в определенные моменты времени. Например, можно реализовать ключ доступа к компьютерной программе, который будет срабатывать только в заранее установленное вами время суток.

Самый интересный вариант применения таймера — это устройство термохрон (Thermochron). Термохрон содержит часы реального времени, энергонезависимое ОЗУ и датчик температуры. Микросхема измеряет температуру через заданные промежутки времени, а результаты измерений записывает в свое энергонезависимое ОЗУ. Параметры процесса измерения могут быть предварительно запрограммированы при помощи 1-Wire интерфейса. Через тот же интерфейс в любой момент могут быть считаны результаты измерений. Таблетки термохрона можно поместить внутри упаковки любого товара, и они будут автоматически контролировать температурный режим хранения товара на протяжении всего времени транспортировки. Получатель может подключить такую таблетку к компьютеру. Специальная программа считает записанную туда информацию и отобразит ее на экране. В случае несоблюдения температурного режима при транспортировке товара фирма-изготовитель может легко предъявить претензии к транспортной компании, осуществляющей перевозку грузов.

Криптопроцессор. Эта микросхема содержит встроенный микроконтроллер, совместимый с системой команд процессора 8051, защищенные часы реального времени, быстродействующий модульный ускоритель возведения в степень больших целых чисел длиной до 1024 бита, ПЗУ (64 Кб) с предварительно зашитой управляющей программой, энергонезависимое ОЗУ (NVRAM) объемом в 6 Кб для хранения важных данных. Криптопроцессор предназначен для шифрования информации. Прибор обеспечивает аппаратное выполнение таких криптографических операций, как быстродействующее шифрование передаваемых данных с открытым 1024 битным ключом и защита сообщений от просмотра.

4.3 1-Wire микросхемы в традиционном исполнении

Наряду с микросхемами iButton в корпусе MicroCan фирма Dallas Semiconductor выпускает большой набор микросхем в корпусах

традиционного вида (см. рис. 4.9). По функциональному назначению среди микросхем в традиционном исполнении можно найти аналоги почти всех микросхем, выполненных по технологии iButton.

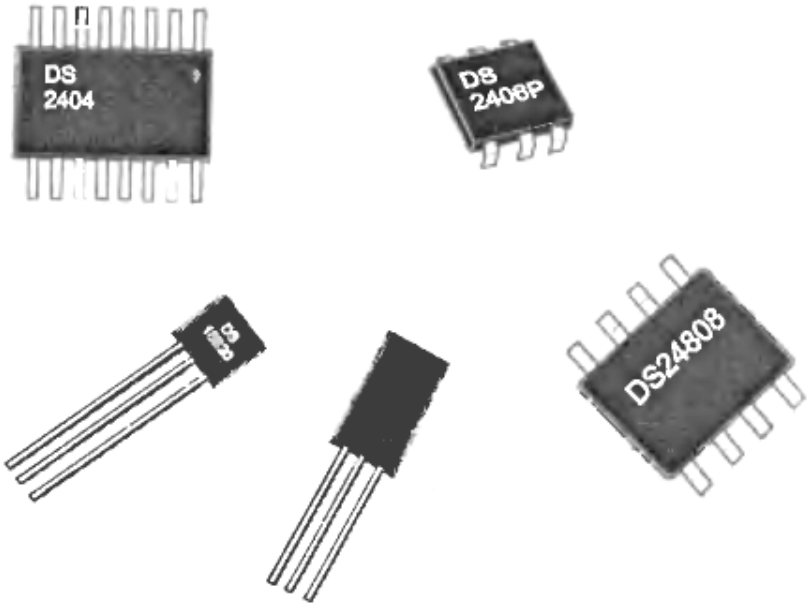


Рисунок 4.9 – Микросхемы для монтажа пайкой

Так в традиционном корпусе можно найти микросхемы памяти (EPROM, EEPROM, NVRAM), часы реального времени, измерители температуры. Однако имеются и другие типы микросхем, которые невозможно выполнить по технологии iButton. Рассмотрим их по порядку.

1) АЦП (аналого-цифровой преобразователь). Типичный представитель - четырехканальный АЦП типа DS2450. Эти микросхемы удобно применять для считывания информации с различных датчиков, выдающих сигнал в аналоговой форме. Например, датчик освещенности, влажности, давления и т.д.

2) Цифровой потенциометр. Это специальный прибор, имитирующий работу обычного потенциометра. Управление «движком» потенциометра осуществляется посредством l-Wire интерфейса. Функциональная схема одного из вариантов прибора изображена на рис. 4.10.

Рисунок 4.10 – Управляемый переменный резистор

Такую микросхему удобно применять для цифрового управления аналоговыми устройствами. Например, такие потенциометры вполне можно было бы применить в качестве программно управляемых регуляторов громкости, баланса и тембра в современном аудиокомплексе. С таким же успехом можно было бы осуществлять все оперативные регулировки в телевизоре. При этом блок управления телевизора можно будет выполнить на микроконтроллере, а управляющие сигналы ко всем элементам будут передаваться по шине l-Wire.

3) Счетчик внешних импульсов**.** Пример такого счетчика — микросхема DS2423. Кроме двухканального счетчика внешних импульсов она имеет встроенную буферную память. Микросхема предназначена для обслуживания различных датчиков, выдают сигнал в импульсном виде. Например, датчик количества оборотов, выходной сигнал с расходомеров-вертушек, емкостные датчики влажности, включенные в задающие цепи управляемых генераторов импульсов, счетчики уровня радиации, преобразователи напряжения в частоту и т.д.

4) Универсальный транзисторный ключ. Содержит не только ключ для переключения внешних цепей, но и имеет возможность контролировать напряжение в переключаемой сети. Например, микросхема DS2407 содержит два таких ключа. Транзисторный ключ может использоваться для включения индикаторов и различных исполнительных механизмов (после усиления по мощности). Благодаря встроенной схеме контроля уровня те же самые микросхемы могут осуществлять считывание сигнала с дискретных датчиков. То есть **с** таких датчиков, которые выдают сигналы вида Да/Нет (датчики положения, прохода, присутствия, пожарной и охранной сигнализации и т.д.).

5) Управляемые разветвители сети. С этим видом устройств мы уже знакомы (см. раздел 4.1). Однако приведенная в этом разделе

функциональная схема разветвителя (см. рис. 4.3) немного упрощена. На самом деле микросхема DS2409 имеет два управляемых электронных коммутатора, позволяющих создать сразу два ответвления. Они называются MAIN (основное) и AUX (дополнительное). Кроме того, микросхема содержит электронный ключ для управления сигнальным светодиодом. Допускается независимое управление посредством l-Wire интерфейса как любым из двух сетевых ответвлений, так и ключом для светодиодного индикатора. Мониторинг элементов питания. Это специализированные микросхемы, осуществляющие непрерывный контроль режимов работы аккумуляторных батарей. Такая микросхема встраивается в аккумулятор и производит непрерывный контроль режима его разрядки. Она измеряет ток разрядки через заданные промежутки времени и записывает результаты измерений в свою внутреннюю память. Когда приходит время заряжать аккумулятор, он подключается к специальному зарядному устройству. Это устройство считывает информацию, записанную во внутреннюю память микросхемы, и использует ее для определения оптимального режима зарядки.

6) Датчики температуры. Датчики температуры, пожалуй, самый применяемый тип микросхем с l-Wire интерфейсом. Выпускается множество модификаций датчиков температуры в корпусах, предназначенных для монтажа пайкой. Например, датчик DS18B20 выпускается в двух модификациях. В корпусе SOIC (8 выводов) и в корпусе TO-92 (см. рис. 4.11). Такие датчики очень удобно использовать для измерения температуры в различных местах помещений, разбросанных на большой площади.

Рисунок 4.11 – Интегральные датчики температуры с 1-Wire интерфейсом

Кроме перечисленных выше приборов, управляемых посредством l-Wire интерфейса, фирма Dallas Semiconductor выпускает большой ассортимент вспомогательных микросхем для облегчения согласования сетей MicroLAN с самыми различными устройствами. Например, микросхема

DS2404 — это двухпортовая статическая память. Считывание и запись памяти в этой микросхеме может осуществляться как по l-Wire интерфейсу, так и со стороны подчиненного последовательного интерфейса, управляемого микропроцессором. Такая микросхема может служить «мостиком» между системами различных типов. Микросхема DS1410E — адаптер параллельного порта. Он позволяет подключать шину MicroLAN к параллельному порту компьютера. Микросхемы DS9097E и DS909-7U — это адаптеры последовательных портов (COM). Микросхема DS2490 — адаптер порта USB.

4.4 Схемная реализация 1 -Wire интерфейса

Для начала рассмотрим принципиальную электрическую схему, реализующую l-Wire интерфейс. Схема соединения ведущего и ведомого устройств посредством однопроводной шины приведена на рис. 4.12. На том же рисунке показаны особенности схемной реализации выходных каскадов ведущего и ведомого устройств. Как видно из рисунка, в схеме l-Wire интерфейса, так же как и винтерфейсе для PC шины, используются выходные каскады с открытым коллектором (стоком) и обшей нагрузкой R,, для всех элементов сети. В спецификации для l-Wire интерфейса специально оговаривается, что резистор R_H должен находиться в непосредственной близости от ведущего устройства. Биполярный транзистор в выходном каскаде ведущего устройства показан условно. С неменьшим (а скорее большим) успехом можно применять микросхемы, у которых выходные каскады построены по КМОП технологии. Ведомые устройства обычно целиком построены на КМОП транзисторах. В режиме ожидания все выходные транзисторы закрыты. На шине присутствует напряжение логической единицы. Информация по шине передается при помощи отрицательных импульсов.

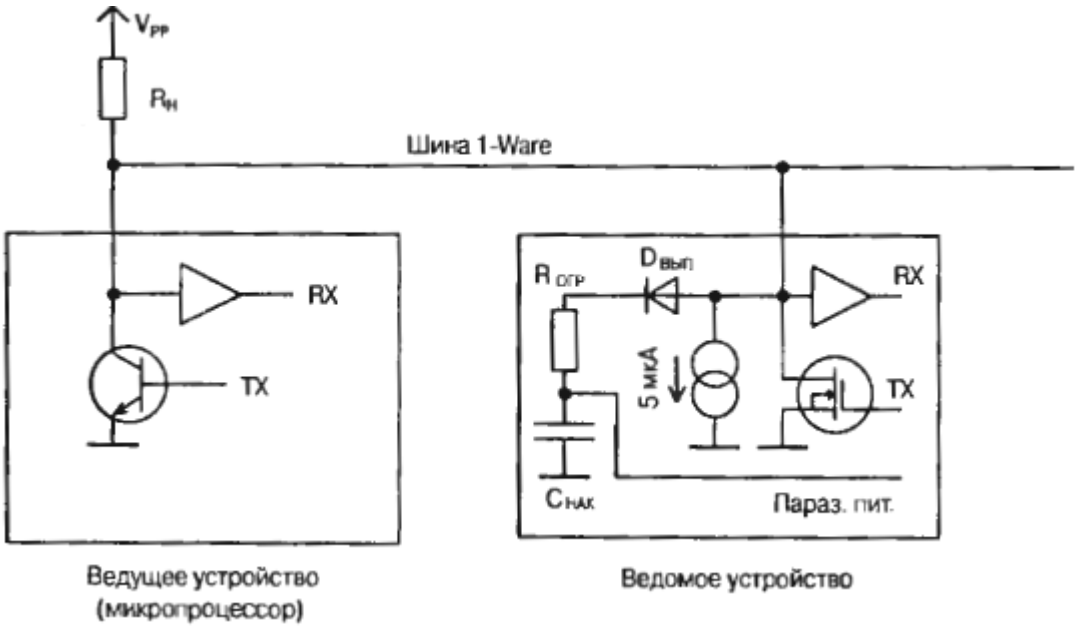


Рисунок 4.12 - Электрическая схема сети MicroLAN

Подробнее этот процесс мы рассмотрим в следующем разделе. А пока обратите внимание на дополнительные элементы в составе ведомого устройства (см. рис. 4.12). Во первых, это встроенный источник тока (обозначенный двумя пересекающимися окружностями). Этот источник создает внутреннюю утечку на входе 1-Wire интерфейса. Смысл этой утечки — создать нулевой уровень сигнала на внутренних элементах ведомого устройства при его отключении от шины 1-Wire. Когда соединение будет восстановлено, внутренняя логика ведомой микросхемы обнаруживает перепад напряжения с нуля на единицу. Сразу после получения такого сигнала ведомая микросхема должна выдать на шину 1-Wire сигнал присутствия. Каким образом выдается этот сигнал, мы увидим чуть позже. Обнаружив сигнал присутствия на линии связи, ведущее устройство может провести процедуру обнаружения новых устройств. Таким образом, шина 1-Wire позволяет легко подключать и отключать различные устройства, не нарушая при этом ее работу. Еще один дополнительный элемент, который присутствует только в схеме интерфейса ведомого устройства — это цепь паразитного питания. Эта цепь состоит из накопительного конденсатора ($C_{нак}$), токоограничивающего резистора ($R_{огр}$) и выпрямительного диода ($D_{вып}$). Все эти элементы используются в режиме паразитного питания. Каждая микросхема, рассчитанная на работу с 1-Wire интерфейсом, имеет два режима питания. Первый режим — обычный. В этом режиме питание подается на специальный вывод микросхемы. Такой вывод обычно имеется у всех микросхем, кроме микросхем серии iButton.

Независимо от наличия вывода питания любая микросхема может питаться непосредственно от информационной шины. Этот способ получения электроэнергии называется «паразитным питанием». В процессе работы на шине всегда присутствует импульсный сигнал. В те моменты, когда напряжение на шине равно единице, выпрямительный диод открывается и накопительный конденсатор заряжается через токоограничивающий резистор. Емкость конденсатора невелика (примерно 800 пФ). Однако и ток потребления КМОП микросхемы тоже очень мал. Поэтому заряда конденсатора хватает для обеспечения питания микросхемы в промежутках между импульсами.

Однако режим паразитного питания применим далеко не во всех случаях. Некоторые операции невозможно выполнить при сверхмалом потреблении энергии. Например, процесс преобразования температуры в код требует полноценного, а не паразитного питания. На этот случай 1-Wire протокол предусматривает третий, специальный режим питания. Такой режим питания используется только совместно с режимом паразитного питания. Основная идея — подавать полноценное питание на ту же самую шину, по которой передается информация в те моменты времени, когда ведомая микросхема выполняет особо энергоемкие операции. При этом, пока на шине присутствует полноценное питание, передача информации невозможна. Подробнее об этом режиме питания мы поговорим, когда будем рассматривать принцип работы микросхемы электронного термометра.

Несколько устройств, включенных по схеме, изображенной на рис. 4.14, это еще не 1-Wire шина. Для того, чтобы она стала таковой, каждое из подключенных устройств должно содержать специальное управляющее устройство, реализующее протокол шины. Именно протокол определяет все правила передачи информации. Поэтому мы приступаем к изучению протокола.

4.5 Синхронизация и побитная передача информации

Как и в случае 1^2C шины, протокол 1-Wire имеет несколько разных уровней. Самый низкий уровень описывает, каким образом передаются отдельные биты. Как уже говорилось выше, 1-Wire протокол предусматривает двухсторонний обмен информации. При этом все операции на шине производятся исключительно под управлением Master устройства. Master

может выполнять операции двух видов: записывать информацию в Slave устройство или считывать информацию из него. Информация передается побайтно, в последовательном виде, бит за битом, начиная с младшего бита. В любом из этих двух случаев, для передачи информации Master устройство вырабатывает на шине тактовые импульсы. Для этого оно периодически «подсаживает» шину при помощи выходного транзистора своего 1-Wire интерфейса (см. рис. 4.12). Полезная информация передается путем изменения длительности этих импульсов. Причем при записи информации длительностью импульсов управляет исключительно Master устройство.

В режиме чтения начинает формирование импульса Master устройство, но Slave устройство может продлевать длительность любого импульса, «подсаживая» в свою очередь сигнал на линии в нужный момент. На рис. 4.13 изображены две временные диаграммы. Верхняя диаграмма иллюстрирует режим записи двух разных битов информации, а нижняя — режим чтения. Участки диаграммы, где линия «отпущена» и уровень сигнала на линии определяется лишь резистором R_H, изображены на диаграмме при помощи тонких линий. Участки, где один из элементов сети «подсаживает» линию, изображены при помощи широких линий.

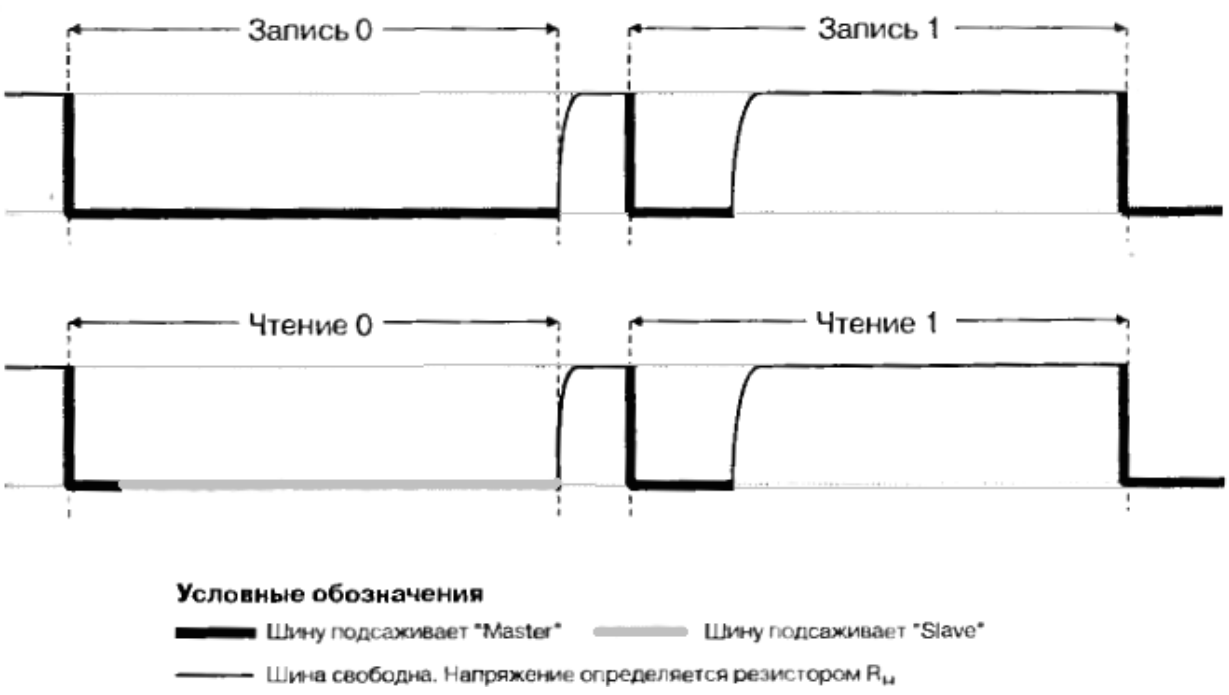


Рисунок 4.13 – Упрощенная диаграмма процесса записи и чтения одного бита

Рассмотрим подробнее два режима работы сети.

1) Запись бита. В исходном состоянии все Slave устройства, подключенные к шине, находятся в режиме ожидания. Линия «отпущена». То есть выходные транзисторы всех элементов шины закрыты, и напряжение на шине определяется резистором нагрузки ($R_н$ на рис. 4.12). Для того, чтобы записать данные в одно из Slave устройств, Maser начинает формировать отрицательные синхроимпульсы (верхняя диаграмма на рис. 4.13). На каждый передаваемый бит формируется один импульс. Импульсы передаются путем «подсаживания» линии до нуля. Для передачи каждого бита выделяется промежуток времени стандартной длительности. Этот промежуток получил название «слот» (Slot). Если значение передаваемого бита равно 0, то Master вырабатывает «длинный» импульс. Его длина равна длительности слота. Для передачи единичного бита Master вырабатывает «короткий» импульс, который, по сути, является чистым синхроимпульсом. Оставшаяся часть слота должна быть заполнена единичным сигналом. Между двумя слотами обязательно должен быть небольшой промежуток, во время которого уровень сигнала на шине тоже равен единице.

Slave устройство в этом режиме только принимает сигнал. Для этого оно постоянно находится в режиме ожидания. Обнаружив начало синхроимпульса, Slave устройство начинает процесс приема бита информации. Передний фронт этого импульса служит Slave устройству началом отсчета. Выдержав паузу, равную длительности синхроимпульса, Slave устройство считывает уровень сигнала на линии. Если в этот момент времени уровень сигнала на линии равен нулю, значит и передаваемый бит равен нулю. Если же сигнал будет равен единице, то и бит равен единице. Протокол шины l-Wire жестко определяет только длительность слота. Интервал между слотами имеет ограничение только на минимальное свое значение. Максимальное значение интервала между слотами неограничено. Таким образом, может легко регулироваться скорость передачи данных от своего максимального значения (16,3 Кбит/с) практически до нуля.

2) Чтение бита. Процесс чтения бита (нижняя диаграмма на рис. 4.13) очень похож на процесс записи. Отличие его в том, что при чтении Master вырабатывает только синхроимпульсы (короткой длительности). Обнаружив синхроимпульс, Slave устройство должно удлинить или не удлинять этот синхроимпульс в пределах слота. Если очередной считываемый бит равен нулю, то синхроимпульс удлиняется. Если единице, удлинения не происходит. Для иллюстрации этого процесса на рис. 4.13 участки временной диаграммы, где линию «подсаживает» Master устройство, изображены

черным цветом. Участки, которые «подсаживает» Slave устройство изображены серым цветом. Master-устройство считывает эту информацию. Для этого оно контролирует уровень сигнала внутри слота сразу после синхроимпульса.

3) Временные параметры. Для надежной работы однопроводного интерфейса необходимо, чтобы в процессе передачи информации всеми элементами сети строго соблюдались временные параметры. Каждая микросхема, подключенная к сети, самостоятельно вырабатывает все необходимые для ее работы интервалы времени. Для ведущего устройства в сети эти требования более жесткие, чем для ведомых. Это связано с тем, что в качестве ведущего устройства обычно выступает микроконтроллер. Любой микроконтроллер способен с высокой точностью отрабатывать любые временные интервалы, благодаря использованию кварцевого резонатора. Ведомые же микросхемы обычно выполнены в микроминиатюрном корпусе. Временные параметры 1-Wire интерфейса таких микросхем формируются обычно параметрическими методами, без помощи кварца.

На рис. 4.14 приведены временные параметры протокола 1-Wire в различных режимах работы. Как видно из рисунка, величина слота для передачи одного бита информации (Tx) должна лежать в пределах от 60 до 120 мкс. Длительность синхроимпульса равна 1 мкс. Ведомое устройство, обнаружив на шине передний фронт синхроимпульса, должно сформировать задержку минимум в 15 мкс, и затем произвести проверку сигнала на шине. Допустимый разброс времени задержки для разных экземпляров микросхем лежит в пределах от 15 до 60 мкс. Этот диапазон показан на рисунке в виде области, обозначенной как «Зона проверки уровня Slave». Минимальная величина интервала между слотами (T_{REC}) равна 1 мкс. Максимальная, как уже говорилось, нсограничена.

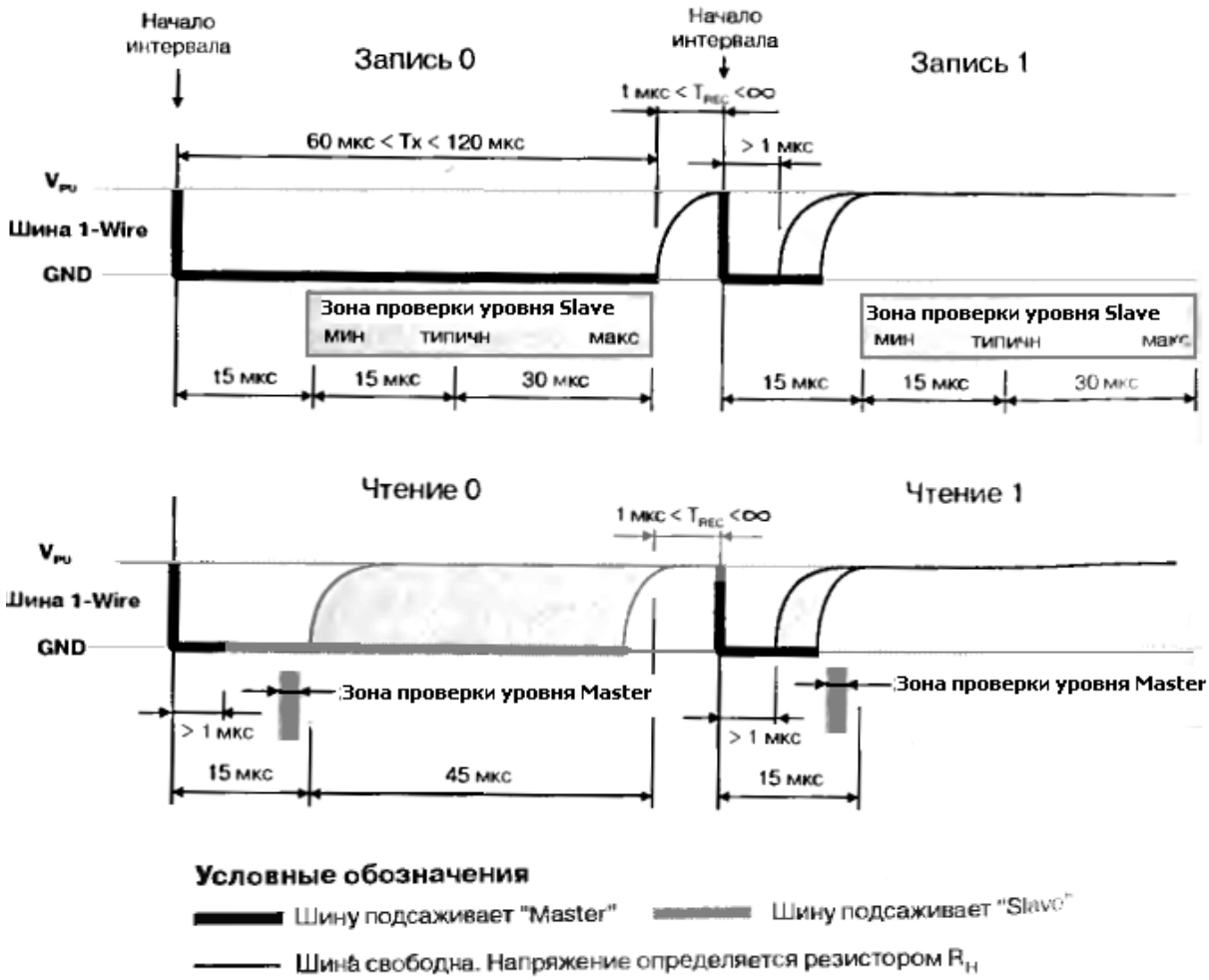


Рисунок 4.14 – Временные характеристики протокола 1-Wire

В режиме записи нулевого бита Master вырабатывает только синхроимпульсы, длительность которых равна 1 мкс. Если читаемый бит равен нулю, Slave устройство продлевает длительность синхроимпульса. Минимальная длительность продленного импульса составляет 15 мкс. Для этого временного интервала тоже допускается довольно значительный разброс. В пределах этого разброса длительность удлиненного импульса может вырасти еще на 45 мкс. Если читаемый бит равен единице, удлинения синхроимпульса не происходит. Для того, чтобы правильно оценить значение читаемого байта, Master устройство должно прочитать уровень сигнала на шине сразу после окончания синхроимпульса, но не позднее, чем через 15 мкс. Зона проверки уровня для Master устройства в режиме чтения значительно уже аналогичной зоны для Slave устройства в режиме записи.

4.6 Сброс и обнаружение присутствия на линии

В предыдущем разделе мы узнали, как происходит чтение и запись отдельных битов. Возникает вопрос, как происходит их разделение на байты. Байты формируются путем подсчета битов. Байты передаются младшим битом вперед. Первые восемь битов — это первый байт. Следующие восемь битов — второй байт. И так далее. Начало же всей этой цепочки определяется сигналом сброса. Любой цикл обмена данными в сети MicroLAN начинается с импульса сброса. Импульс сброса — это сверхдлинный отрицательный импульс на шине l-Wire, вырабатываемый ведущим устройством.

Временная диаграмма, иллюстрирующая процесс формирования импульса сброса, приведена на рис. 4.15. С импульсом сброса тесно связан еще один служебный сигнал — сигнал присутствия на шине. Сигнал присутствия вырабатывает каждое Slave устройство сразу после окончания действия импульса сброса. Master устройство должно проконтролировать наличие этого импульса. Если импульса присутствия нет, то это значит, что на линии нет ни одного Slave устройства.

Рисунок 4.15 – Временная диаграмма процесса начального сброса

Кроме инициации импульсов присутствия импульс сброса переводит в исходное состояние всю систему. Любые незаконченные процессы на линии моментально завершаются, и отсчет битов начинается сначала. Временные характеристики сигнала сброса и сигнала присутствия на линии приведены на рис. 4.15. Длительность импульса сброса должна быть не менее 480 мкс.

Процесс передачи информации по линии может начинаться не раньше, чем через 480 мкс после окончания действия импульса сброса. В этом временном интервале и ожидается появление сигнала присутствия. Для этого после окончания импульса сброса Master «отпускает» линию и ждет сигнала от Slave устройств.

Каждое Slave устройство, обнаружив импульс сброса, выдерживает паузу в 15...60 мкс, а затем «подсаживает» линию. Длительность импульса присутствия составляет 60...240 мкс. Сигналы присутствия от всех Slave устройств сливаются в один общий импульс. Ведущее устройство проверяет наличие нулевого уровня на линии в средине этого интервала. Если сигнал обнаружен, то это значит, что на линии имеется хотя бы одно нормально работающее Slave устройство и Мастер может продолжать работу в сети. Если сигнал не обнаружится, то микропроцессор перейдет к обработке этой ситуации. Обычно в этом случае он выдает сигнал ошибки (зуммер, светодиод, надпись на дисплее и т.д.).

Итак, мы познакомились с самым низким уровнем протокола l-Wire, и знаем, как передается информация на уровне отдельных битов. Теперь перейдем к следующему уровню. Этот уровень определяет, каким же образом Master устройство управляет направлением передачи информации по шине, и как передается информация на уровне байтов. Для этого в сети MicroLAN существует такое понятие, как команды. Master посылает команды, а все Slave устройства их выполняют. Любая операция в сети MicroLAN начинается с команды. Команда представляет собой один байт информации. Каждая команда имеет свой собственный код. Для того, чтобы понять, как работает механизм передачи команд, рассмотрим, как выполняется команда «Чтение ПЗУ» (Read ROM). Под ПЗУ понимается специальная область памяти микросхемы, где при помощи лазера записан ее индивидуальный код (ID-код). Команда «Чтение ПЗУ» позволяет микроконтроллеру (Master устройству) прочитать этот код.

4.7 Система команд протокола 1 –Wire

В предыдущих разделах мы рассмотрели работу сети MicroLAN на нескольких разных уровнях. Каждый из этих уровней имеет свое название.

Первые два уровня мы изучили в предыдущем разделе. Определяются они следующим образом:

- Физический уровень. Это схемное решение шины, уровни напряжений, взаимодействие элементов сети на уровне электрических сигналов.
- Уровень связи. Это правила формирования сигнала сброса, синхронизация и способ передачи битов информации в обоих направлениях.

Эти два уровня определяются физическими особенностями сети MicroLAN. Однако при описании протокола принято выделять еще два уровня, которые, напротив, связаны с логикой работы протокола. Это, так называемые, сетевой и транспортный уровни. Любая шина I -Wire всегда находится на одном из этих уровней. Причем, сразу после сигнала сброса шина переходит на сетевой уровень. И лишь отработав команду сетевого уровня, шина переходит на транспортный. Команды сетевого уровня служат для адресации Slave микросхем. Это команды вида «Выбрать микросхему такую-то». Команды транспортного уровня служат для непосредственной передачи информации. Это команды типа «записать», «прочитать». Рассмотрим оба этих уровня подробнее.

1) Сетевой уровень. На этом уровне происходит адресация элементов на однопроводной шине. Находясь на этом уровне, все Slave устройства ожидают одну из команд адресации. Такая команда должна выбрать одно или несколько устройств, с которыми Master будет работать на транспортном уровне. Все команды сетевого уровня-приведены в табл. 4.1.

Таблица 4.1 – Команды сетевого уровня

Код	Команда	Описание
33H	Чтение ПЗУ	Чтение индивидуального кода ведомой микросхемы из специального ПЗУ (см. рис. 4.16)
0FH	Чтение ПЗУ	То же самое, но для микросхемы DS1990A (в этой микросхеме используется немного другая система команд)
55H	Совпадение ПЗУ	Поиск и активизация элемента с заданным кодом. Остальные элементы переходят в пассивный режим
OCCH	Пропуск ПЗУ	Пропуск команды выбора устройства. Применяется, если нужно работать сразу со всеми устройствами, или оно всего одно
0F0H	Поиск ПЗУ	Команда, позволяющая осуществлять поиск элементов, подключенных к шине. Одновременно с поиском происходит определение их индивидуальных номеров

Как видно из таблицы, коды одной и той же команды для некоторых видов микросхем могут отличаться друг от друга. Так, команда «Чтение ПЗУ» (Read ROM) почти для всех типов микросхем имеет код 33H, а для микросхемы DS1990A этот код равен 0FH. Принцип работы команды «Чтение ПЗУ» мы уже рассмотрели в предыдущем разделе. Master сначала передает код операции, а затем читает 64 разряда ID кода.

Команда «Совпадение ПЗУ» (Match ROM) работает по-другому. Master передает на шину код операции, а затем передает еще 64 разряда, представляющих собой ID код адресуемой микросхемы. Получив всю эту информацию, все Slave микросхемы, подключенные к шине, переходят на транспортный уровень. Причем одна из этих микросхем, у которой коды совпали, останется в активном режиме и будет принимать команды транспортного уровня. Остальные перейдут в пассивный режим и будут ожидать очередного сигнала начального сброса. Очевидно, что перед тем, как применить команду «Совпадение ПЗУ», программист должен знать ID коды всех Slave микросхем, подключенных к однопроводной шине. Единственный способ узнать эти коды — по очереди подключить все эти микросхемы к микроконтроллеру, и при помощи специальной вспомогательной программы прочитать ID код каждой микросхемы, используя команду «Чтение ПЗУ».

Команда «Пропуск ПЗУ» (Skip ROM) позволяет перейти на транспортный уровень и при этом оставить активными все Slave микросхемы. Для выполнения этой команды Master просто передает в шину код ОССН. И сразу после этого все элементы шины перейдут на транспортный уровень. Команда «Пропуск ПЗУ» используется в двух случаях. Либо на линии присутствует всего одно Slave устройство и выбор адреса не требуется. Либо нужно подать команду транспортного уровня сразу на все элементы сети. Такой режим применяется, например, когда на все микросхемы датчиков температуры нужно одновременно подать команду «Начало преобразования температуры в код».

Команда «Поиск ПЗУ» (Search ROM) позволяет, при любом количестве элементов, подключенных к шине, адресовать всего один из них, не зная заранее его ID кода. Алгоритм этой команды составлен таким образом, что в процессе ее выполнения становится известен ID код одной из ведомых микросхем. Повторяя команду «Поиск ПЗУ» несколько раз подряд, можно обнаружить все подключенные к шине Slave устройства и определить их ID коды. Причем, если к шине подключены N разных Slave устройств, то для того чтобы найти их все, достаточно применить команду «Поиск ПЗУ»

только N раз. Причем алгоритм всегда позволяет узнать, есть ли еще не-найденные устройства на шине. Как же работает эта удивительная команда? Рассмотрим подробно все этапы выполнения команды «Поиск ПЗУ».

- Сначала ведущее устройство выдает на шину код операции (0F0H) и переключается в режим приема. Но принимает оно в данном случае всего два бита информации.
- Получив команду, каждое ведомое устройство передает эти два бита. Они представляют собой значение младшего разряда ID кода конкретной микросхемы в прямом и инверсном виде. Сначала передается прямое, а затем инверсное значение разряда. Все микросхемы, подключенные к шине, передают своп биты одновременно. Поэтому ведущее устройство принимает результирующее значение этих двух битов. Представим себе ситуацию, что первый бит ID кодов всех Slave микросхем окажется одинаковым и равным нулю. В этом случае, вес микросхемы выдадут код 01 (двоичное). Если первый бит 1D кода у всех микросхем окажется равным единице, то значение двух принятых битов будет 10 (двоичное). И, наконец, в том случае, если первый бит ID кода хотя бы у одной из микросхем будет отличаться от того же бита у других, на линии возникнет конфликт. Разные микросхемы будут выдавать разные коды. Одни 01, другие 10. Так как однопроводная шина обладает эффектом схемного «И», значение пары битов, принятых в этом случае Master устройством будет равно 00. Таким образом, на этом этапе выполнения команды «Поиск ПЗУ» можно получить следующую информацию:
  1) Мы можем определить, имеет ли младший разряд ID кода одинаковое значение для всех микросхем, подключенных к I-Wire шине или нет.
  2) Если младший разряд для всех микросхем одинаковый, мы можем точно узнать его значение.
- Следующий этап — сепарация. Master выдает на шину всего один бит, при помощи которого он производит отбор Slave микросхем.

  Если Master выдаст на шину единицу, то все элементы, у которых младший разряд кода равен нулю, отключаются. Остаются только те элементы с единицей в младшем разряде. Если Master выдаст ноль, то на шине остаются микросхемы с младшим разрядом, равным нулю. Сепарацию нужно производить таким образом, чтобы на шине всегда оставались активные устройства. Так, если на предыдущем этапе Master получил два бита, значение которых равно 01, то он должен выдать на шину бит равный нулю. Если два полученных бита были равны 10, то ответный бит должен быть единичным. Особый случай — значение битов

00. Здесь алгоритм допускает любые варианты. Передаваемый бит может быть равен как нулю, так и единице. В любом случае часть микросхем отключится, а часть останется в работе. Во всех трех перечисленных выше случаях на шине останутся активными элементы, для которых нам теперь точно известно значение одного из разрядов ID кода. Кроме того, точно известно, что на шине имеется хотя бы одна такая микросхема.

- На следующем этапе вся шина переходит к проверке следующего разряда ID кода. Для этого Master опять читает, а все Slave выдают новую пару битов. Теперь эти биты представляют прямое и инверсное значения нового разряда. После чтения пары битов происходит очередной этап сепарации.
- Так, этап за этапом, происходит оценка всех 64 разрядов ID кода. В случае разницы в кодах, происходит выбор одного из вариантов и отсеивание остальных. Выполнение команды «Поиск ПЗУ» заканчивается после того, как таким образом будут оценены все 64 разряда. В результате на шине останется всего одно активное Slave устройство, а микроконтроллер будет точно «знать» ее ID код. Дальше вся шина переходит на транспортный уровень.

Операцию «Поиск ПЗУ» можно повторять множество раз. При этом поиск новых ID кодов нужно вести с учетом уже найденных. Найденные ранее варианты ID кода в процессе поиска нужно исключать. Таким образом, можно найти ID коды для всех устройств подключенных к вашей шине. Адреса запоминаются в памяти микроконтроллера и в дальнейшем, для обращения к каждому элементу сети, используется команда «Совпадение ПЗУ».

4.7.1 Дополнительные команды сетевого уровня

В некоторых видах микросхем встречается еще несколько дополнительных команд сетевого уровня. Например, микросхема DS1994 имеет встроенный таймер, который способен вырабатывать специальный сигнал прерывания в заданный момент времени. Сигнал прерывания передастся все по той же одной единственной однопроводной шине.

Сигнал прерывания представляет собой отрицательный импульс большой длительности (960...3840 мкс). Если на шине присутствует несколько микросхем DS1994, то, обнаружив сигнал прерывания,

микропроцессор должен еще определить, какая из этих микросхем является его источником. Для этого предусмотрена специальная команда «Поиск прерывания» (Search Interrupt). Код этой команды — 0ECH. Команда «Поиск прерывания» работает точно так же как команда «Поиск ПЗУ», но поиск происходит только среди тех микросхем, которые вызвали прерывание. Остальные микросхемы сразу отключаются. Подобная команда существует и в микросхемах других видов. Например, в серии микросхем DS18S20, DS18B20, DS1822. Эти микросхемы представляют собой электронные измерители температуры. Все они содержат два специальных регистра, при помощи которых можно задавать верхний и нижний пределы изменения температуры. При выходе температуры за установленные пределы микросхема термодатчика не вырабатывает на шине сигнал прерывания, как микросхема DS1994. Однако в составе ее команд также имеется команда «Поиск Прерывания», при помоши которой микроконтроллер всегда может быстро отобрать из всех микросхем только те, которые требуют немедленной обработки.

Транспортный уровень. Все микросхемы сети MicroLAN переходят на этот уровень после любой команды сетевого уровня. Набор команд транспортного уровня для разных микросхем несколько отличается один от другого. Основные команды транспортного уровня — это команды «Запись памяти» и «Чтение памяти». Все микросхемы, использующие l-Wire интерфейс, по внутренней архитектуре представляют собой несколько ячеек памяти. Обычно это набор регистров, через который осуществляется обмен данными. Через одни регистры микропроцессор может записывать в микросхему коды, определяющие режим ее работы. Через другие он может считывать различные величины. Например, температуру, результат цифро-аналогового преобразования и так далее.

В качестве примера команды транспортного уровня рассмотрим команду «Чтение памяти». На рис. 4.17 приведена временная диаграмма выполнения команды «Чтение памяти». Для адресации микросхемы в приведенном примере использована команда сетевого уровня — «Совпадение ПЗУ». Как видно из рисунка, выполнение команды начинается с импульса сброса. Далее, все ведомые устройства вырабатывают сигнал присутствия на линии. Но пока это только сетевой уровень.

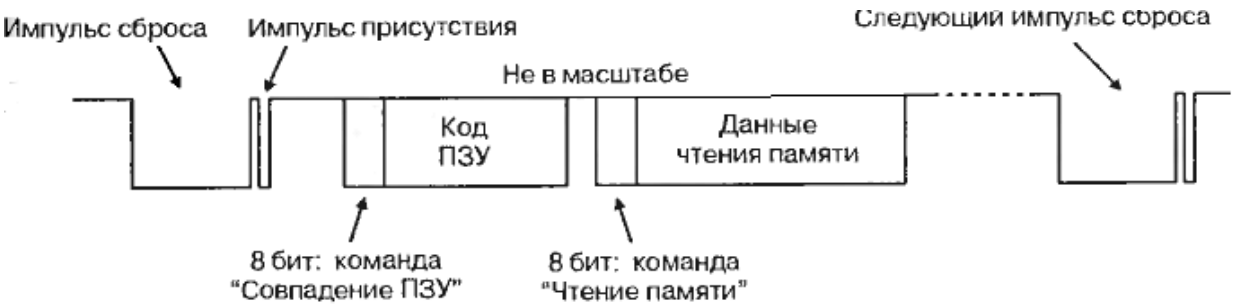


Рисунок 4.17 – Формат команды 1-Wire протокола

Выдержав положенную паузу, ведущее устройство выдает на линию команду «Совпадение ПЗУ». Команда состоит из восьми бит кода операции и 64-битного ID кода. Получив эту команду, все ведомые микросхемы переходят на транспортный уровень. Но только одна из них останется активной. Остальные перейдут в пассивный режим ожидания.

Далее, ведущее устройство передает на шину команду «Чтение памяти» (8 бит). Получив эту команду, выбранная нами ведомая микросхема переходит в режим выдачи данных. Ведущее же устройство читает эти данные из ведомой микросхемы. Объем считываемых данных зависит от типа применяемой микросхемы. Чаще всего он равен 8 байтам (64 бит) и обязательно содержит контрольную сумму в последних восьми битах.

После того, как все данные приняты, выполнение команды можно считать законченным. Для того, чтобы выполнить следующую команду, нужно все начинать с начала. То есть с импульса начального сброса.

Если вы сравните команды сетевого и команды транспортного уровней, то можете заметить, что для команд разного уровня могут применяться одинаковые коды. Однако это не вызывает никаких затруднений, так как каждая из категорий команд действует на своем уровне.

4.8 Структура ID кода

Структура индивидуального кода, записанного в специальное ПЗУ любой микросхемы, поддерживающей однопроводный интерфейс l-Wire, условно изображена на рис. 4.18.

Рисунок 4.18 – Структура индивидуального кода микросхемы 1-Wire

Код состоит из трех основных элементов:

Первые 8 бит, начиная с младшего разряда — это групповой код микросхемы. Групповой код однозначно определяет ее вид. Примеры группового кода для разных видов микросхем приведены в табл. 4.3. Следующие 48 бит — это уникальный серийный номер микросхемы. Последние 8 бит — это контрольная сумма (CRC) младших 56 бит.

Контрольная сумма, или циклически избыточный код (Cyclic Redundancy Check — CRC), формируется при помощи специального полиномного генератора, состоящего из регистра сдвига и трех логических элементов XOR, реализующих функцию «Исключающее ИЛИ». Функциональная схема такого генератора показана на рис. 4.19.

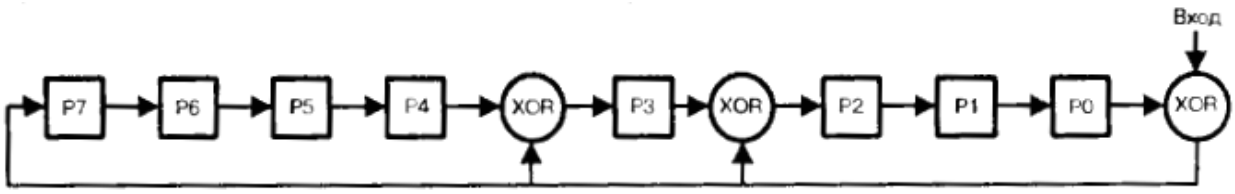


Рисунок 4.19 – Функциональная схема полиномного генератора

Такая схема реализует многочлен $X^8+X^5+X^4+1$. Биты (P7...P0) регистра сдвига перед началом вычисления устанавливаются в ноль. Затем, на вход схемы последовательно подаются 56 младших разрядов ID кода. При подаче каждого очередного разряда информация в сдвиговом регистре полиномного генератора сдвигается на один шаг в направлении, показанном стрелками. При этом в разряд P7 регистра попадет результат операции XOR между значением очередного бита на входе всей схемы и значением разряда P0.

Значения разрядов P3 и P2 также являются результатом операции XOR, которую выполняют остальные два элемента. После 56 циклов такого сдвига в разрядах P7...P0 полиномного генератора будет находиться искомая контрольная сумма. Точно такое значение будут иметь последние 8 бит исходного ID кода. Если содержимое регистра полиномного генератора и последних восьми байтов ID кода совпали, то это значит, что при передаче кода по сети ошибок не произошло.

Таблица 4.3 - Групповые коды некоторых видов микросхем

Вид микросхемы	Описание	Групповой код
DS18S20	Электронный термометр	ЮН
DS18B20	Электронный термометр	28H
DS18B22	Электронный термометр	22H
DS1992	Энергонезависимая память	08H
DS1993	Энергонезависимая память	06H
DS1994	Энергонезависимая память	04H
DS2401	Кремневый серийный номер	01H
DS2417	Таймер реального времени	27H
DS2450	4-канальный АЦП	20H
DS2890	Электронный переменный резистор	2CH

В микросхемах l-Wire периферии полиномный генератор выполнен аппаратным способом. Там действительно имеется сдвиговый регистр и три элемента «Исключающее ИЛИ». Если ведущее устройство выполнено на основе микроконтроллера, то подсчет контрольной суммы организуется программным путем.

4.9 Интегральные датчики температуры

До конца понять принципы работы любого последовательного протокола можно только на конкретном примере. Для I^2C шины в качестве такого примера мы выбрали микросхемы флэш-памяти. Это было абсолютно оправдано, так как для I^2C шины — это самое удачное применение. Теперь нам нужно выбрать столь же удачный пример для однопроводной шины. Безусловно, самое удачное применение шины — подключение к микроконтроллеру микросхем измерителей температуры. Фирма Dallas Semiconductor выпускает целый набор таких микросхем. Самый первый интегральный термодатчик с 1-Wire интерфейсом назывался DS1820. Эта оригинальная микросхема сразу после своего появления приобрела

популярность у разработчиков электронной аппаратуры. Микросхема была помещена в пластмассовый корпус, напоминающий по размерам и форме миниатюрный транзистор. Однако в процессе ее проектирования была допущена одна досадная ошибка. Ошибка приводила к тому, что термодатчик в отдельных случаях давал неверный результат.

Правда эта ошибка легко исправлялась путем несложной дополнительной программной обработки на стороне микроконтроллера. Поэтому, несмотря на ошибку, микросхема продолжала широко применяться. Фирма Dallas Semiconductor быстро исправила досадную ошибку. Однако существующую микросхему переделывать не стала для обеспечения совместимости с массой электронных устройств, разработанных к тому времени на ее основе. Фирма просто выпустила новую микросхему, уже без дефекта, и назвала ее DS18S20. Новая микросхема имеет более миниатюрный корпус и является полным функциональным аналогом DS1820, но ее внутренняя микропрограмма не содержит никаких ошибок.

Термодатчики DS1820 и DS18S20 имеют рабочий диапазон температур от -55°C до +125°C. Максимальное время преобразования температуры в код 750 мс. Результаты измерения считываются по 1-Wire интерфейсу из специальных внутренних регистров микросхемы в виде девятиразрядого двоичного кода. Девять двоичных разрядов представляют собой значение температуры в дополнительном коде, измеренную с шагом в 0.5°C. Точность измерения температуры различна в различных точках рабочего диапазона. На отрезке -10°C...+85°C точность измерения равна ±0.5°C. Чем ближе к краю диапазона, тем точность измерения меньше. На самых краях она составляет всего ±2°C. Преобразователь температуры в код, применяемый в микросхем DS1820 и DSI8S20, способен выдавать значение температуры с большим количеством отсчетов (более мелким шагом). Промежуточные значения просто вычисляются. Для этого в обеих микросхемах предусмотрены два специальных регистра. В первом из них хранится число оставшихся отсчетов, а во втором — число отсчетов на один градус. Используя содержимое этих регистров, можно рассчитать значение температуры с разрешением в 0.01...0.05°C. Тут следует заметить, что повышение дискретности не ведет к увеличению точности измерения температуры, а лишь улучшает плавность регулировочных характеристик.

Более совершенная микросхема DS18B20 не требует никаких дополнительных вычислений. Высокая дискретность достигается увеличением количества разрядов результирующего кода. Причем в микросхеме имеется возможность изменения количества разрядов выходного регистра. По

умолчанию выходной регистр имеет 9 разрядов. Изменяя содержимое регистра конфигурации, микроконтроллер может увеличить количество разрядов до 12. Точность измерения температуры в диапазон -10...+85°С составляет ±0.5°С. На выходе микросхемы DS18B20 мы получаем прямой код, значение которого равно величине измеряемой температуры. В 9-разрядном режиме значение измеряемой температуры выдается с дискретностью в 0.5°С. В двенадцатиразрядном режиме количество отсчетов повышается в восемь раз. Максимальное время преобразования для микросхемы DS18B20 также зависит от выбранного количества разрядов. Для 12-разрядного режима работы оно равно 750 мс.

В настоящее время микросхема DSI8B20— это самый распространенный вид микросхем подобного назначения. Однако фирма Dallas Semiconductor выпускает и другие модификации электронных термометров. Микросхема DS1822 — более дешевый вариант DS18B20. По функциональным возможностям они являются полными аналогами. Однако DS1822, имеет меньшую точность измерения температуры. В диапазоне — 10°С...+85°С точность измерения составляет всего ±2°С. А на краях еще хуже. Зато она дешевле своего аналога.

Выпускаются также варианты всех вышеперечисленных микросхем, предназначенные для работы исключительно в режиме паразитного питания. Они получили название DS18S20-PAR, DS18B20-PAR, DS1822-PAR. Эти микросхемы внешне ничем не отличаются от своих аналогов. Однако вывод питания у них не задействован. Существуют и другие, экзотические модификации датчиков температуры. Например, микросхема DS18B20X — аналог микросхемы DS18B20, выполненный в специальном микроминиатюрном безвыводном корпусе.

Очевидно, что для нашего примера удобнее всего выбрать микросхему DS18B20, так как остальные варианты либо устарели, либо могут считаться ее модификациями. Поэтому рассмотрим подробнее именно эту микросхему.

4.10 Внутренняя архитектура микросхемы DS18B20

Микросхема DS18B20 выпускается в двух модификациях. Они отличаются исключительно конструкцией корпуса. На рис. 4.20 приведен внешний вид обеих модификаций микросхемы. Основной вариант микросхемы выполнен в миниатюрном пластмассовом корпусе типа TO-92.

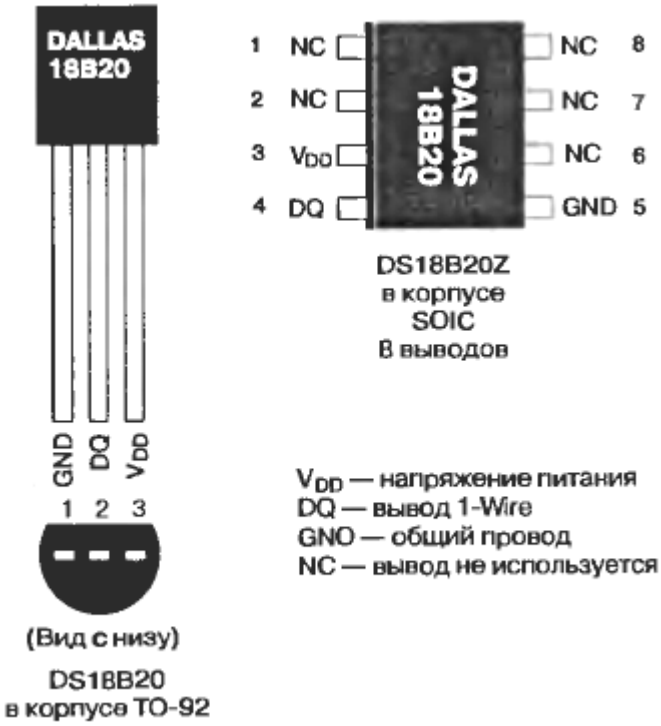


Рисунок 4.20 – Внешний вид микросхем DS18B20 в двух разных исполнениях

Второй вариант заключен в пленарный восьмивыводной, миниатюрный корпус типа SOIC. Для того, чтобы различать эти два варианта исполнения, второй вариант получил обозначение DS18B20Z. Микросхема имеет всего три задействованных вывода: DQ — вход/выход данных l-Wire интерфейса; V_{DD} — вывод внешнего питания; GND — общий провод. Расположение выводов показано на рис. 4.20.

Внутренняя структура микросхемы DS18B20 приведена на рис. 4.21. Сигнал с шины DQ и напряжение с внешнего вывода питания (V_{DD}) прежде всего поступают на схему паразитного питания. Принцип работы паразитного питания мы уже рассматривали в разделе 4.4 настоящей главы. Однако в схеме паразитного питания имеется еще один дополнительный элемент, о котором не говорилось ранее. Это датчик наличия питания. Датчик представляет собой пороговый элемент, на который поступает напряжение питания от внешнего источника. Датчик вырабатывает логический сигнал, поступающий в схему управления. В результате микросхема получает возможность автоматически определять режим своего питания. Микроконтроллер, работающий в качестве Master устройства на той

же шине, имеет возможность запросить у всех подключенных к ней датчиков информацию о режиме питания и соответствующим образом скорректировать алгоритм своей работы.