Министерство образования и науки Украины
Национальный технический университет
«Харьковский политехнический институт»

Методические указания
к лабораторной работе № 6

«Связь микроконтроллера с датчиком температуры по 1-Wire интерфейсу»

по курсу «*Интерфейсы компьютерных систем управления*»

Харьков 2006

Министерство образования и науки Украины

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ № 6

«СВЯЗЬ МИКРОКОНТРОЛЛЕРА С ДАТЧИКОМ ТЕМПЕРАТУРЫ ПО 1-WIRE ИНТЕРФЕЙСУ»

ПО КУРСУ «ИНТЕРФЕЙСЫ КОМПЬЮТЕРНЫХ СИСТЕМ УПРАВЛЕНИЯ»

для студентов специальности 7.0910002

Утверждено
редакционно-издательским
советом университета,
протокол № 2 от 02.06.04

Харьков НТУ «ХПИ» 2006

Методические указания к лабораторной работе «Связь микроконтроллера с датчиком температуры по 1-Wire интерфейсу» по курсу «Интерфейсы компьютерных систем управления» для студентов специальности 7.0910002 / Сост. Д.А. Лунин, М.В. Гунбин. – Харьков: НТУ «ХПИ» 2006. – 13 с.

Составители: Д.А. Лунин, М.В. Гунбин

Рецензент А.В. Ивашко

Кафедра автоматики и управления в технических системах

Цель работы: изучить принципы разработки программы для микроконтроллера, который осуществляет связь с датчиком температуры по 1-Wire интерфейсу.

Основные сведения.

Измерение температуры распределенных объектов является типичной задачей, оптимальное решение которой может быть достигнуто путем применения низкоскоростных сетей с соответствующими датчиками. Существует весьма практичный способ проектирования соответствующих систем на базе однопроводной сети MicroLAN компании Dallas Semiconductor. Выпускаемая этой же компанией серия цифровых термометров и термостатов, изначально подготовленных к работе в однопроводной сети, позволяет с успехом решать разнообразные задачи измерения и управления температурой в удаленных друг от друга точках.

Рассмотрим протокол, который используется 1-проводным интерфейсом. В любой момент времени на 1-проводной шине можно выделить устройство-мастер, которое в нашем случае является микропроцессор, и подчиненные устройства – это микросхемы термометров. Нам требуется знать протокол передачи байтов, которые могут являться командами или данными.

1. Схемная реализация 1-Wire интерфейса

Для начала рассмотрим принципиальную электрическую схему, реализующую 1-Wire интерфейс. Схема соединения ведущего и ведомого устройств посредством однопроводной шины приведена на рис. 1. На том же рисунке показаны особенности схемной реализации выходных каскадов ведущего и ведомого устройств. Как видно из рисунка, в схеме 1-Wire интерфейса, так же как и в интерфейсе для PC шины, используются выходные каскады с открытым коллектором (стоком) и общей нагрузкой R_H для всех элементов сети. В спецификации для 1-Wire интерфейса специально оговаривается, что резистор R_H должен находиться в непосредственной близости от ведущего устройства.

В режиме ожидания все выходные транзисторы закрыты. На шине присутствует напряжение логической единицы. Информация по шине передается при помощи отрицательных импульсов. Любое устройство, подключенное к шине, как ведущее, так и ведомое, может создавать отрицательные импульсы и тем самым передавать информацию. Однако ведомое устройство начинает процесс передачи только под управлением ведущего.

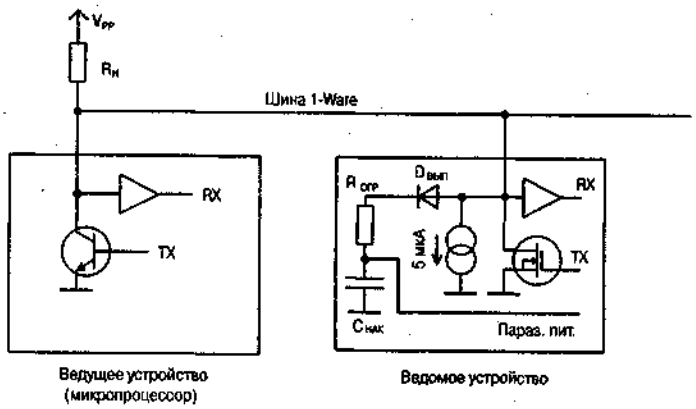


Рис. 1. Электрическая схема сети MicroLAN

Рассмотрим дополнительные элементы в составе ведомого устройства (см. рис. 1). 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 конденсатор ($C_{НАК}$), токоограничивающего резистора (Rorp) и выпрямительного диода (Dвып). 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.

2. Синхронизация и побитная передача информации

Как и в случае PC 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. 1). 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. 2 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. Рассмотрим подробнее два режима работы сети.

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_H на рис. 1). Для того, чтобы записать данные в одно из Slave устройств, Maser начинает формировать отрицательные синхроимпульсы (верхняя диаграмма на рис. 2). 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.

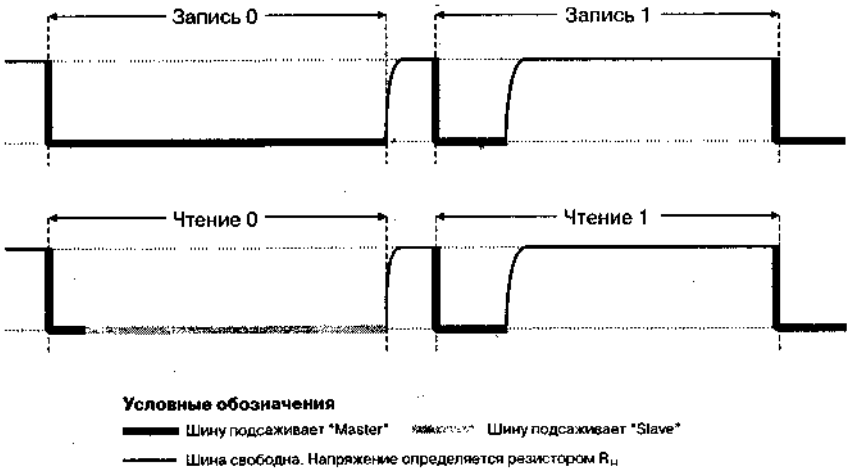


Рис. 2. Упрощенная диаграмма процесса записи и чтения одного бита

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 1-Wire жестко определяет только длительность слота. Интервал между, слотами имеет ограничение только на минимальное свое значение. Максимальное значение интервала между слотами неограниченно. Таким образом, может легко регулироваться скорость передачи данных от своего максимального значения (16,3 Кбит/с) практически до нуля.

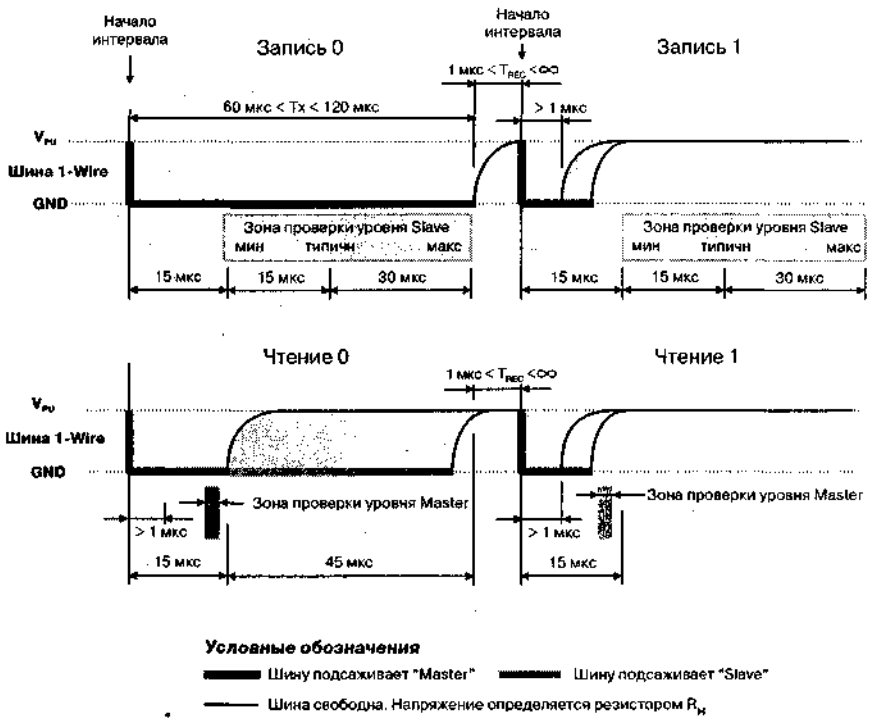


Рис. 3. Временные характеристики протокола 1-Wire

Чтение бита.

Процесс чтения бита (нижняя диаграмма на рис. 3) 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. 3 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.

Временные параметры.

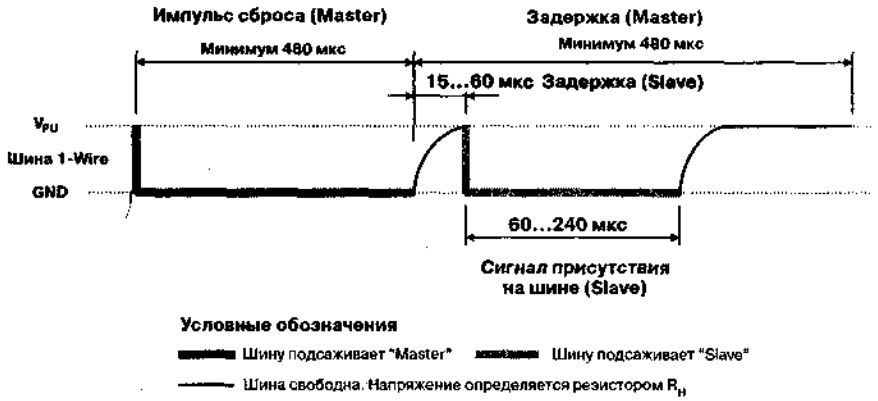
Для надежной работы однопроводного интерфейса необходимо, чтобы в процессе передачи информации всеми элементами сети строго соблюдались временные параметры. Каждая микросхема, подключенная к сети, самостоятельно вырабатывает все необходимые для ее работы интервалы времени. Для ведущего устройства в сети 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. 3 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, неограничена.

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. Сброс и обнаружение присутствия на линии

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 1-Wire, вырабатываемый ведущим устройством.

Временная диаграмма, иллюстрирующая процесс формирования импульса сброса, приведена на рис. 5. 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.

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. 5. 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.).

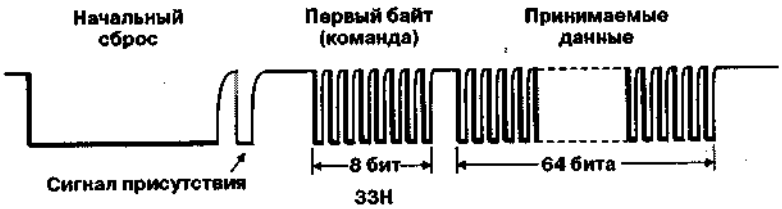


Рис. 5. Временная диаграмма процесса начального сброса

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. Следует заметить, что команда «Чтение ПЗУ», в отличие от всех остальных команд, может корректно выполняться только в том случае, когда на шине имеется одно Master и одно Slave устройство.

Выполнение команды начинается с импульса сброса. Затем Slave устройство вырабатывает, а Master проверяет сигнал присутствия на линии.

Если сигнал на месте, Master выдает на линию код команды «Чтение ПЗУ» (33Н). Код выдается в режиме записи. Получив этот код, Slave микросхема переключается в режим выдачи информации. В этом режиме Master читает из Slave микросхемы 64 бита (8 байт), составляющие ее ID код. Последние восемь битов ID кода – это контрольная сумма. Она вычисляется по специальному алгоритму из его первых 56 битов. Микропроцессор проверяет контрольную сумму и в случае несоответствия повторяет процедуру «Чтение ПЗУ».

Существует целая система команд, позволяющая гибко управлять процессами обмена информации по шине.

3. Система команд протокола 1-Wire

Выше была рассмотрена работа сети MicroLAN на нескольких разных уровнях. Каждый из этих уровней имеет свое название. Первые два уровня определяются 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 1 -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».

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. 1.

Таблица 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
OCC H	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 OCCH. 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 ненайденные 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 свои 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, все микросхемы выдадут код 01 (двоичное). Если первый бит ID 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 1-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».

3. Общие сведения о датчике температуры DS18S20.

Отличительные особенности.

1. Уникальный 1-проводный интерфейс требует только одного вывода порта для подключения;
2. Каждое устройство имеет 64-битный уникальный серийный номер, хранящийся в ПЗУ на кристалле;
3. Возможность многоабонентской работы упрощает создание приложений, осуществляющих распределённое измерение температуры;
4. Не требует внешних компонентов;
5. Микросхема может быть запитана от линии данных. Рабочее напряжение от 3 до 5,5 В;
6. Измеряет температуру от -55 до +125 °C (от -67 до +257 °F);
7. Точность ±0.5 °C в условиях температуры от -10 до +85 °C;
8. Разрешение термометра - 9-бит;
9. Преобразует температуру за 750 мс (максимум);
10. Задаваемые пользователем энергонезависимые параметры настройки температурных сигналов;
11. Команда поиска сигнала распознаёт и адресует те устройства,

температура которых вне запрограммированных границ (условие температурной сигнализации);

12. Применяется в управлении термостатированием, промышленных системах, устройствах широкого потребления, термометрах или любых чувствительных к температуре системах.

Описание

Цифровой термометр DS18S20 обеспечивает 9-12 битные температурные измерения по шкале Цельсия и имеет сигнальную функцию с верхней и нижней точками переключения, программируемыми пользователем и хранящимися в энергонезависимой памяти (EEPROM). Микросхема DS18S20 подключается через 1-проводную шину, которая по определению требует только одной линии данных (а также общей) для взаимодействия с центральным процессором. Она имеет рабочий температурный диапазон от -55 до +125 °C и точность ±0.5 °C в диапазоне от -10 до +85 °C. Также микросхема DS18S20 может запитываться прямо с линии данных («паразитное питание»), устраняя необходимость во внешнем источнике питания.

После завершения преобразования полученное значение сравнивается с величинами, хранящимися в регистрах TH и TL. Если измеренная температура выходит за установленные пределы, устанавливается сигнальный «флаг» (впрочем, его установка производится после каждого измерения). При установленном «флаге» DS1820 отвечает на команду «Поиск сигнала». Это позволяет быстро идентифицировать точку с отклонениями температуры свыше допустимых пределов и сразу считать показания соответствующего термометра. Если команда «Поиск сигнала» не применяется, регистры TH и TL могут быть использованы как регистры общего назначения.

Задание на лабораторную работу.

1. На основании вышеприведенного теоретического материала разработать программу связи по 1-Wire интерфейсу микроконтроллера с датчиком температуры. Для описания протокола обмена микроконтроллера использовать язык ассемблер.

2. Проверить наличие связи микроконтроллера и датчика температуры. Принятые данные от датчика температуры, сохранить в ОЗУ микроконтроллера. Преобразовав измеренные данные, вывести их на индикатор. Все запросы должен генерировать микроконтроллер, на каждый запрос микроконтроллера, датчик температуры должен дать ответ.

3. Измерить температуру воздуха в помещении. Опыт повторить три раза и сравнить полученные значения.

Контрольные вопросы.

1. Для чего предназначен интерфейс 1-Wire?
2. Какие режимы работы интерфейса 1-Wire бывают?
3. Электрические параметры интерфейса 1-Wire.
4. Структура сообщения по интерфейсу 1-Wire.
5. Требования интерфейса 1-Wire.