НАЦИОНАЛЬНАЯ АКАДЕМИЯ НАУК УКРАИНЫ

ДОНЕЦКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
им. А.А.ГАЛКИНА

А.С.Карначёв
В.А.Белошенко
В.И.Титиевский

МИКРОЛОКАЛЬНЫЕ СЕТИ:

* интеллектуальные датчики
* однопроводный интерфейс
* системы сбора информации

Донецк 2000

УДК 62-52 + 62-529 + 681.325.5

Печатается по решению Ученого совета Донецкого физико-технического института им. А.А.Галкина Национальной Академии наук Украины

Рецензенты - д. ф.-м. н. А.Д.Прохоров, к. т. н. Н.Н.Кабдин

Карначёв А.С., Белошенко В.А., Титиевский В.И.
Микролокальные сети: интеллектуальные датчики, однопроводный интерфейс, системы сбора информации. - Донецк: ДонФТИ НАНУ Украины, 2000. - 199с. с ил. - ISBN 966-95187-3-3

Книга посвящена перспективной технологии организации компьютерных микролокальных сетей сбора и обработки технической информации, основанной на устройствах с однопроводным интерфейсом фирмы Dallas Semiconductor. Подробно рассмотрены устройство приборов с однопроводным интерфейсом (интеллектуальных датчиков температуры и адресуемых ключей), особенности протокола обмена. На примере сети температурного мониторинга показаны принципы разработки микролокальных сетей этой технологии: выбор оптимальной топологии, кабелей, состава устройств и т.п. Приведены примеры использования описываемой технологии для организации температурного мониторинга зданий и сооружений в коммунальном хозяйстве. В книге содержатся все необходимые пояснения и технические данные для практического построения микролокальной сети.

Для инженеров и разработчиков компьютерных систем сбора и обработки информации.

Содержание

Принятые сокращения

АСУТП	-	автоматизированная система управления технологическими процессами;
ВИС	-	виртуальные измерительные средства;
МЗР	-	младший значащий разряд;
ОЗУ	-	оперативное запоминающее устройство;
ПЗУ	-	постоянное запоминающее устройство;
ПК	-	персональный компьютер;
ПО	-	программное обеспечение;
СЗР	-	старший значащий разряд;
СОП	-	сверхоперативная память;
УАПП	-	универсальный асинхронный приемо-передатчик;
УПИ	-	универсальный параллельный интерфейс;
ЦИК	-	циклический избыточный код;
ЭППЗУ	-	электрически программируемое постоянное запоминающее устройство;
MicroLAN	-	Miniature Local Area Network (микролокальная сеть);
TMEX	-	Touch Memory Executive (администратор контактной памяти).

Введение

О чем эта книга и для кого она написана? Книга посвящена одной из компьютерных технологий сбора и обработки информации, описание которой еще не появлялось в отечественной литературе. Ее можно назвать технологией однопроводного интерфейса. Она, несомненно, представляет собой перспективную ветвь среди многих других решений компьютерных систем сбора и обработки информации. Что же это за технология и чем она интересна?

Рассматриваемая технология сбора, накопления и передачи информации разработана фирмой Dallas Semiconductor и основывается на перспективных однопроводных каналах связи, специальном однопроводном протоколе обмена и целой серии датчиков, модулей памяти и других устройств, поддерживающих однопроводный интерфейс. Можно назвать следующие привлекательные черты этой системы: 1) наличие большой номенклатуры взаимозаменяемых аппаратных средств (датчиков, электронных ключей, модулей памяти); 2) наличие однопроводного интерфейса и, следовательно, лишь одного провода для коммуникаций; 3) возможность работы без внешнего источника питания, только за счет энергии информационного сигнала; 4) гибкая топология сетей сбора информации; 5) возможность организации беспроводной сети на основе автономных энергонезависимых датчиков-накопителей информации.

На основе этой технологии можно строить самые разнообразные информационные системы, имеющие для пользователя несомненное достоинство - простоту эксплуатации. Все сложности по электрическому сопряжению, программному обеспечению спрятаны внутри. Разработчику информационных систем остается лишь, как из конструктора, собрать, выбирая наиболее подходящие детали, нужную ему конфигурацию.

В этой книге мы рассмотрели однопроводную технологию Dallas Semiconductor на примере организации сети температурного мониторинга. Главное внимание уделено описанию устройства датчиков и адресуемых ключей, описанию однопроводного протокола обмена и принципов организации информационных микролокальных сетей. Прояснив эти вопросы и имея в руках датчики любого типа (не только темпе-

ратурные), разработчик может создать сеть для сбора информации любой другой физической природы.

Описываемая здесь технология хороша еще и тем, что даже неискушенный читатель, мало-мальски знакомый с электроникой и компьютерным "железом", может, используя материал этой книги, создать дешевые и надежные системы температурного мониторинга, скажем, своей квартиры, теплицы и т.п.

Книга рассчитана как на специалистов, так и на широкий круг читателей, интересующихся компьютерными технологиями сбора и обработки информации.

Книга построена следующим образом. В первой главе приводится общее описание построения системы сбора информации, основанной на технологии однопроводного интерфейса, даются перечень и краткая характеристика входящих в нее компонент, определяется используемая терминология. Во второй главе описаны аппаратные и программные аспекты организации однопроводного обмена данными. Третья и четвертая главы посвящены периферийным устройствам, а именно, интеллектуальным температурным датчикам и адресуемым ключам. В пятой главе подробно рассмотрена организация древовидной микролокальной сети сбора информации. В шестой главе описан пример практического использования однопроводной технологии для температурного мониторинга объектов большой протяженности. Наконец, в приложениях приведены справочные данные по периферийным устройствам микролокальных сетей.

Глава 1. Общее описание технологии MicroLAN

1.1. Терминология

Прежде чем перейти к дальнейшему изложению, приведем список часто встречающихся терминов и наше толкование содержания этих терминов.

Микролокальная сеть - информационная сеть небольшой протяженности (до нескольких сот метров), объединяющая некоторое количество периферийных устройств (датчиков, адресуемых ключей, модулей памяти) под единым управлением компьютера или автономного микропроцессора.

Однопроводная шина - провод, соединяющий компьютер (микропроцессор) с периферийными устройствами; везде в дальнейшем подразумевается, что шина включает в себя как сигнальный, так и общий провод (т.е. на самом деле состоит из двух проводов). По шине осуществляется весь обмен данными и командами между компьютером и периферийными устройствами. Часто для краткости мы будем вместо термина "однопроводная шина" использовать термин "шина" или "шина данных", имея всегда в виду именно однопроводную шину. Чтобы избежать сухости изложения, иногда шину мы будем называть линией связи или просто линией.

Однопроводный интерфейс - аппаратные и программные средства согласования компьютера (микропроцессора) с периферийными устройствами микролокальной сети посредством однопроводной шины.

Мастер шины - устройство, управляющее работой шины (в данном случае компьютер или микропроцессор). Иногда для краткости мы будем его называть просто мастером.

Помощник - устройство, подключенное к однопроводной шине (датчик, адресуемый ключ, модуль памяти и т.п.), не являющееся мастером этой шины. Для краткости мы будем применять также термин "устройство".

Интеллектуальный датчик - первичный преобразователь, преобразующий физическую величину в цифровой код и способный выполнять ряд команд мастера шины.

Распределенный температурный мониторинг - контроль температуры во многих точках протяженного объекта либо

большого числа объектов, расположенных на удалении друг от друга.

Временные слоты чтения и записи - ограниченные интервалы времени, необходимые для передачи или приема одного бита информации.

1.2. Что такое микролокальная сеть

Микролокальная сеть (Miniature Local Area Network - MicroLAN) - это сеть, использующая для цифрового обмена однопроводную линию связи. Она обеспечивает очень дешевый обмен информацией между компьютером и включенными в сеть устройствами, поддерживающими однопроводный интерфейс. Основу таких сетей составляют интеллектуальные датчики температуры, представляющие собой специализированные микропроцессоры, измеряющие температуру окружающей их среды и преобразующие ее значение в последовательный двоичный код. Существенным является тот факт, что каждый датчик индивидуально маркирован, т.е. содержит внутри себя идентификационный номер, по которому управляющий компьютер может распознать каждый конкретный датчик в сети и обратиться к нему. Двух датчиков с одинаковыми идентификационными номерами не существует.

MicroLAN имеет древовидную структуру, содержащую “ствол” (магистраль) и много “ветвей” (рис.1.1). Ее “ствол” через специальный контроллер соединяется с COM портом персонального компьютера, обеспечивающим протокол RS-232; в этом случае компьютер выступает как сервер этой микролокальной сети. Этот компьютер называют также мастером шины. Управление сетью осуществляется с помощью специального программного пакета TMEX из-под DOS или WINDOWS [1]. На вершине “ствола” (удаленном его конце) размещается маркер ствола. Вдоль “ствола” располагаются адресуемые ключи, через которые к “стволу” подключаются дополнительные линии - “ветви”. Эти “ветви” на своих удаленных концах также имеют маркеры. “Ветви”, в свою очередь, могут иметь ответвления - “веточки”. От “веточек” могут отходить более мелкие отростки, называемые “черенками”. В любом случае каждое из ветвлений имеет

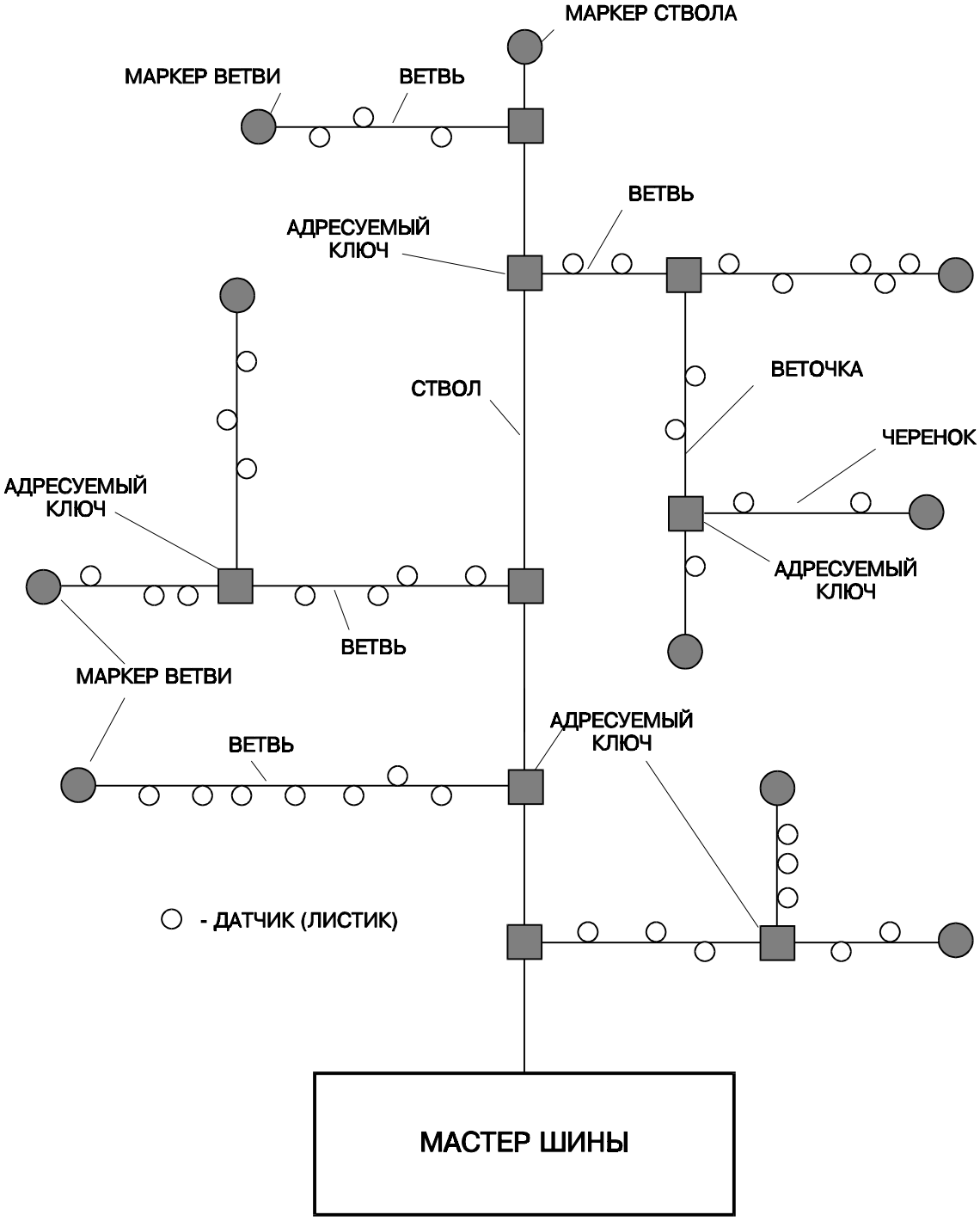


Рис. 1.1. Сеть MicroLAN древовидной структуры.

на конце свой маркер. Назначение маркеров - проверять, надежно ли осуществляется обмен информацией в границах данного ветвления. Мастер шины отличает маркеры от других подсоединенных к сети устройств по содержанию передаваемой ими информации. В общем случае подсоединяемые к сети устройства (модули памяти, датчики, адресуемые ключи)

подключаются не к “стволу”, а к “ветвям”, “веточкам” и “черенкам”, представляя собой как бы “листья” этого “дерева”. Однако, если сеть не имеет ветвлений, то “листья” подсоединяются непосредственно к “стволу”.

Чтобы обменяться информацией, например, с устройством A, подключенным к какой-либо “веточке”, мастер шины должен сначала включить адресуемый ключ, соединяющий “ствол” с той “ветвью”, от которой отходит нужная “веточка”, а затем включить еще один ключ, соединяющий данную “ветвь” с нужной “веточкой”, на которой находится требуемый “листик” - устройство A. Такое программно управляемое избирательное подключение отдельных сегментов сети позволяет избежать перегрузки кабеля, возникающей при одновременном включении всех имеющихся в сети устройств, и, следовательно, позволяет строить довольно протяженные и разветвленные сети. Кроме того, адресуемый ключ несет информацию о физическом расположении подключаемого к диалогу устройства.

Циркулирующая в сети MicroLAN информация передается пакетами. При передаче или приеме пакета информации всегда осуществляется контроль истинности циклическим избыточным кодом (ЦИК). В пакете передаваемых данных содержится 8-битовый или 16-битовый ЦИК. Мастер шины, приняв очередной пакет данных, подсчитывает его ЦИК. Если этот ЦИК совпадает со значением, переданным в пакете, значит, данные приняты верно. В противном случае операция приема повторяется.

Обмен информацией в сети является программно управляемым. Мастер шины опрашивает узлы сети и по результатам опроса предпринимает те или другие действия. Если персональный компьютер оборудован несколькими COM портами, то к каждому из COM портов может быть подсоединена независимая сеть описанной древовидной структуры. Это позволяет увеличить производительность и размер сети.

1.3. Датчики температуры

Интеллектуальные датчики температуры фирмы Dallas Semiconductor (подробно о них см. главу 3) представляют собой специализированные микропроцессоры, осуществля-

ющие по команде мастера шины преобразование температуры в цифровой двоичный код и передачу этого кода в линию связи. Эти датчики имеют следующие особенности:

- обладают уникальным однопроводным интерфейсом, требующим только одной линии связи для коммуникации;
- допускают подключение на одну линию связи нескольких устройств, что позволяет осуществлять распределенный температурный мониторинг;
- не требуют внешних компонент;
- могут питаться от линии данных;
- в режиме ожидания не потребляют энергию;
- допускают определяемые пользователем энергонезависимые установки пороговых температур;
- измеряют температуру в диапазоне от -55°C до +125°C с шагом от 0.03°C до 0.5°C в зависимости от модификации;
- выдают в линию связи значение температуры в виде 9-ти, 10-ти, 11-ти или 12-битового двоичного числа;
- цикл преобразования температуры в код занимает от 200 мс до 750мс, в зависимости от разрешающей способности;
- предусматривают специальный режим поиска аварийных температур, при котором мастер шины по специальной команде может адресоваться только к тем датчикам, температура которых вышла за установленные пользователем пределы.

Все датчики температуры имеют в своем составе три основных компонента: 1) 64-битовое постоянное запоминающее устройство (ПЗУ); 2) собственно датчик температуры и 3) энергонезависимые триггеры нижнего и верхнего порогов температур TL и TH. В ПЗУ каждого датчика содержится информация, идентифицирующая этот датчик и позволяющая мастеру шины вычленить из множества подключенных к шине устройств нужное и вести с ним диалог. При этом остальные устройства этот диалог "слушают", но в нем не участвуют. Общение датчиков с мастером шины осуществляется посредством системы команд. Эти команды разделены на две группы (или функции). В первую группу входят команды, работающие с ПЗУ датчика и исследующие его содержимое. Эти команды позволяют мастеру исследовать шину и определить, сколько и

какие именно устройства подключены к шине; найти нужное устройство и вести с ним диалог; реализовать функцию определения аварийных датчиков, т.е. датчиков, чья температура перешла заданный пользователем порог. Вторую группу команд составляют команды управления и памяти. Например, одна из команд управления инструктирует датчик на выполнение преобразования температуры в цифровой код. Результат помещается в сверхоперативную память (СОП) и может быть в любое время прочитан путем выдачи команды чтения сверхоперативной памяти. Подробнее о командах первой и второй групп см. главы 2 и 3.

1.4. Адресуемые ключи

Адресуемые ключи представляют собой электронные коммутаторы, предназначенные для организации ветвлений в однопроводной сети MicroLAN. Информация, поступающая от мастера шины на вход ключа, может транслироваться в другую ветвь микролокальной сети. И, наоборот, информация из побочной ветви может быть передана через те же выводы в обратном направлении - к мастеру шины. Ключ не требует источника питания. Он берет энергию непосредственно из шины данных. Так же, как и датчики температуры, адресуемые ключи в своем составе содержат ПЗУ, благодаря которому мастер может идентифицировать каждый включенный в шину ключ и обращаться к нему с командами.

1.5. Однопроводная шина

В основу работы MicroLAN положена организация связи процессора с периферийными устройствами через однопроводную шину. Однопроводная шина представляет собой систему, состоящую из одного мастера шины и одного или нескольких помощников (адресуемых ключей, датчиков и т.п.) (рис. 1.2). Однопроводная шина по определению имеет только одну сигнальную линию. Важно, чтобы любое подключенное к ней устройство имело возможность захватить эту шину в некоторый момент времени. Для этого каждое из подсоединенных устройств должно иметь либо выход с тремя со-

стояниями, либо выход с открытым коллектором (стоком).

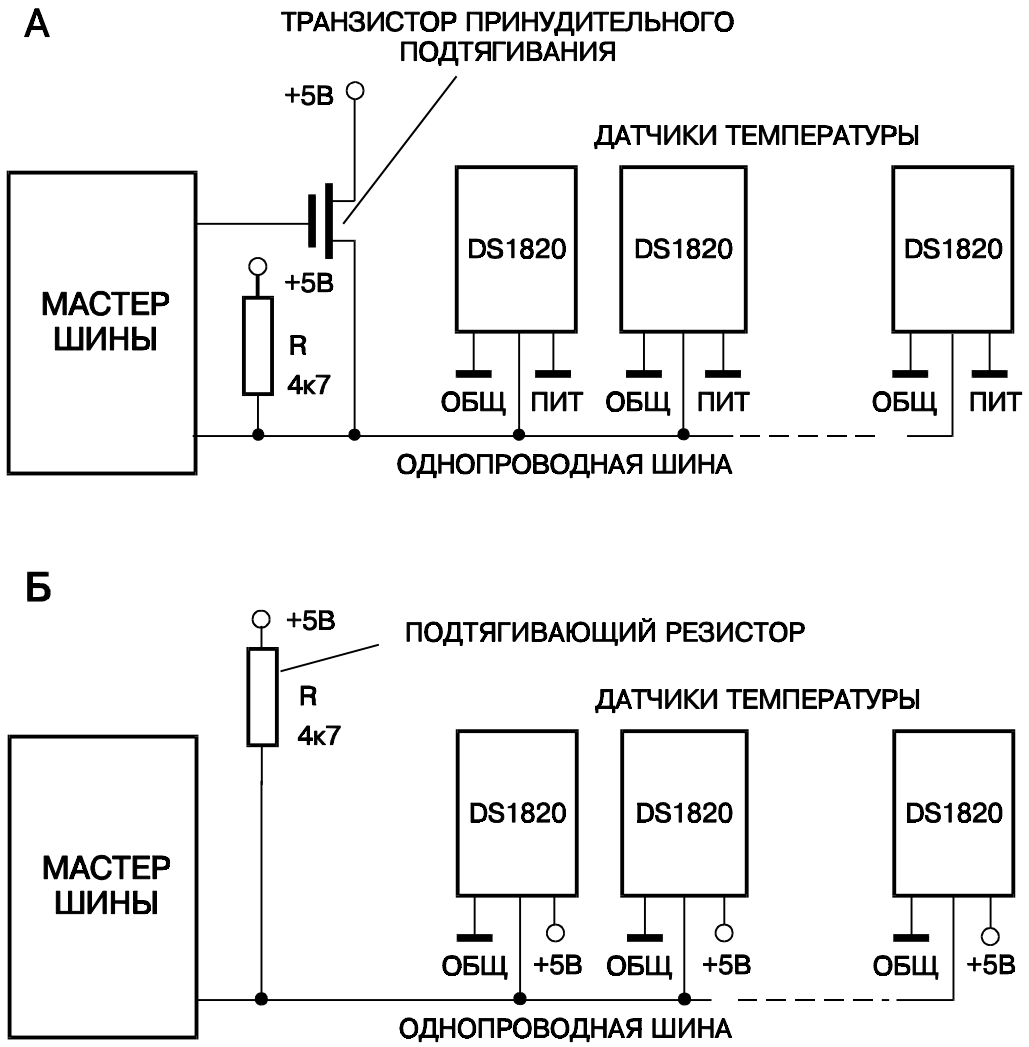


Рис 1.2. Однопроводная шина (А - подключение датчиков в режиме пассивного питания; Б - подключение датчиков в режиме активного питания).

Многоточечная шина состоит из однопроводной линии с подключенным к ней большим количеством помощников. Основное состояние шины - высокое. Оно обеспечивается подтягивающим резистором R, подключающим шину к источнику питания. Всякая передача информации осуществляется путем закорачивания шины мастером или помощниками, т.е. переводом ее в низкое состояние. Как только и мастер, и помощники отпускают шину, на ней устанавливается исходное высокоуровневое состояние.

Если по каким-либо причинам нужно приостановить обмен данными, шина должна удерживаться в высоком состо-

янии, пока обмен не возобновится. Если же шина опустится до нулевого логического уровня и пробудет в этом состоянии более 480 мкс, помощники будут сброшены и данные потеряны. Кроме того, в этом случае адресуемые ключи перейдут в свое исходное (выключенное) состояние.

Получить доступ к какому-либо устройству на шине можно в такой последовательности:

- провести инициализацию;
- выполнить одну из команд функции ПЗУ;
- выполнить команду функций управления или памяти.

Все операции в однопроводной шине начинаются с инициализирующей последовательности. Эта последовательность состоит из импульса сброса, выдаваемого мастером шины, и следующего за ним импульса (или импульсов) присутствия, выдаваемого помощником (помощниками). Импульс присутствия позволяет мастеру узнать, подсоединен ли к шине хоть один помощник. Как только мастер получил импульс присутствия, свидетельствующий о том, что хотя бы один помощник подключен к шине и готов к работе, он может выдать одну из команд первой группы, позволяющих либо исследовать состав подключенных устройств, либо, если состав уже известен, обратиться к конкретному устройству. Команды эти однобайтовые и передаются следующим образом. Как только мастер зафиксировал окончание импульса присутствия, т.е. обнаружил его задний (возрастающий) фронт, он осуществляет побитовый посыл байта команды, начиная с младшего бита. Посыл ведется фиксированными временными интервалами - слотами. Все слоты генерирует мастер, переводя шину в низкоуровневое состояние. Этот перевод шины в низкое состояние синхронизирует процесс обмена. Если мастер передает единицу, то он, переведя шину в низкое состояние, сразу же отпускает ее. Под действием подтягивающего резистора на шине устанавливается высокое состояние, которое сохраняется на все время действия данного временного слота. Затем мастер генерирует следующий временной слот, принудительно переводя шину в низкое состояние. Если он теперь желает передать нуль, то он удерживает шину в низком состоянии в течение длительности слота и в конце слота отпускает ее. Периферийные устройства, получив сигнал синхронизации в виде спадающего фронта, после некоторой задержки, необходимой для установления

нужного уровня сигнала на шине, проверяют ее состояние и, в результате, считывают нуль или единицу. Так происходит передача нулей и единиц кода команды. Передав команду, мастер продолжает генерировать временные слоты, но после очередного перевода шины в низкое состояние он обязательно отпускает ее, чтобы во время действия слота на ней установился уровень, определяемый периферийным устройством. Таким образом мастер осуществляет прием информации от периферийного устройства. Подробнее организацию обмена мы обсудим в следующей главе.

Глава 2. Организация обмена

При всем разнообразии номенклатуры устройств, поддерживающих однопроводный интерфейс, обмен информацией между ними и мастером проходит по единому протоколу, называемому протоколом однопроводной шины, или протоколом однопроводного интерфейса. Этот протокол регламентирует временные, электрические и логические параметры сигналов. В этой главе мы подробно рассмотрим все аппаратные и программные аспекты реализации обмена по однопроводной шине.

2.1. Однопроводная шина

Однопроводная шина - это такая система, которая состоит из одного мастера и одного или нескольких помощников. В этой системе в качестве помощников выступают периферийные устройства, поддерживающие однопроводный интерфейс (датчики, адресуемые ключи, модули памяти и т.п.). Дальнейшее обсуждение этой системы разбито на три темы: аппаратная конфигурация, сигналы однопроводной шины (типы сигналов и их временные параметры), последовательность обработки.

2.1.1. Аппаратная конфигурация

Электрическая эквивалентная схема однопроводной шины с подключенными к ней мастером и одним помощником (по сути, эквивалентная схема простейшей MicroLAN) представлена на рис. 2.1. Со стороны мастера шины сеть оборудована источником напряжения постоянного тока V_P, определяющим потенциал, к которому подтягивается шина в состоянии логической единицы, подтягивающим резистором $R_п$ и управляемым мастером шины транзисторным ключом K, опускающим шину до уровня логического нуля.

Соединительные провода (кабель) на эквивалентной схеме представлены индуктивными $L_{ПР}$, $L_{ОБР}$ и активными $R_{ПР}$, $R_{ОБР}$ сопротивлениями соответственно прямой и обратной линий, а также суммарной емкостью $C_{КАБ}$ кабеля. Суммарная

емкость есть произведение погонной емкости кабеля на его длину. Активное сопротивление прямого провода - произведение длины этого провода на величину его погонного сопротивления. Активное сопротивление обратной линии равно сопротивлению прямой линии плюс несколько Ом, приходящихся на каждый из ключей, расположенных между мастером шины и наиболее удаленной ветвью. Если адресуемых ключей в составе сети нет, то сопротивления прямой и обратной линий равны.

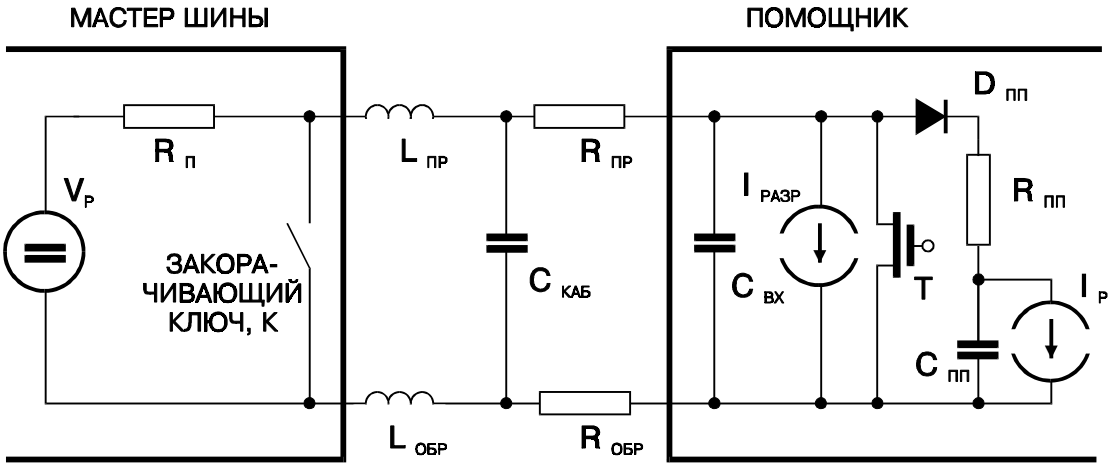


Рис. 2.1. Эквивалентная схема простейшей MicroLAN.

Устройство с однопроводным интерфейсом (помощник) на эквивалентной схеме представлено своей входной емкостью $C_{ВХ}$, постоянным разрядным током $I_{РАЗР}$ (обычно равным 5 мкА), цепью пассивного питания ($D_{ПП}$, $R_{ПП}$, $C_{ПП}$) и рабочим током $I_Р$ (возникающим при обмене и равным 10 мкА). Любой прибор, включенный в сеть, потребляет рабочий ток, даже если он в данный момент не адресуется. Этот ток необходим для синхронизации его однопроводного интерфейса с протоколом обмена. В открытом состоянии импеданс устройства с однопроводным интерфейсом составляет 100 Ом, что приводит к стекающему току 4 мА и падению напряжения на устройстве 0.4В. Если к шине подключено несколько устройств, то значения $C_{ВХ}$, $I_{РАЗР}$, $I_Р$ и $C_{ПП}$ должны быть умножены на число устройств; величину $R_{ПП}$ нужно разделить на число устройств. Полевой транзистор Т в эквивалентной схеме устройства позволяет ему реагировать на команды мастера шины и передавать данные, переводя шину в состояние логического нуля. В каждый момент

времени только один такой транзистор в сети находится в открытом состоянии, опуская шину на короткие периоды временных интервалов чтения. Исключение составляют режим, когда мастер шины проверяет наличие на ней приборов с однопроводным интерфейсом (так называемый цикл опроса присутствия), а также периоды выполнения команд Search ROM (Исследование ПЗУ), Skip ROM (Игнорирование ПЗУ) и Read ROM (Чтение ПЗУ) (об этих командах см. ниже).

Многоточечная шина состоит из однопроводной шины со множеством присоединенных к ней помощников. Однопроводная шина требует подтягивающего резистора примерно в 5кОм. В исходном состоянии на шине присутствует высокий уровень. Если по какой-либо причине обработку нужно прервать, то шина ДОЛЖНА находиться в исходном, т.е. высоком состоянии до тех пор, пока обработка продолжится. Если этого не обеспечить и шина останется в низком состоянии на время, большее 480 мкс, то все компоненты шины будут сброшены. Иными словами, все датчики перейдут в режим ожидания, а все адресуемые ключи окажутся разомкнутыми.

2.1.2. Сигналы однопроводной шины

Для сохранения целостности данных требуется выполнение строгого протокола обмена. Протокол состоит из нескольких типов сигналов в линии: импульса сброса, импульса присутствия, записи нуля, записи единицы, чтения нуля, чтения единицы. Все эти сигналы, за исключением импульса присутствия, генерируются мастером шины. На рис. 2.2 приведена инициализирующая последовательность, необходимая для начала любых обменов в шине. Импульс сброса и следующий за ним импульс присутствия показывают, что помощник готов к передаче или приему данных, инициируемых командами функций ПЗУ и памяти. Мастер шины передает импульс сброса (сигнал низкого уровня длительностью не менее 480 мкс). Затем освобождает линию и переходит в режим приема. Однопроводная шина подтянута к высокому логическому уровню подтягивающим резистором 5кОм. Обнаружив задний (восходящий) фронт имульса сброса, помощник ждет 15-60 мкс и затем передает импульс присутствия (сиг-

нал низкого уровня длительностью 60-240 мкс).

Чтение и запись данных происходят с использованием временных интервалов чтения/записи, разделяющих соседние биты, а также командного слова, определяющего смысл операции.

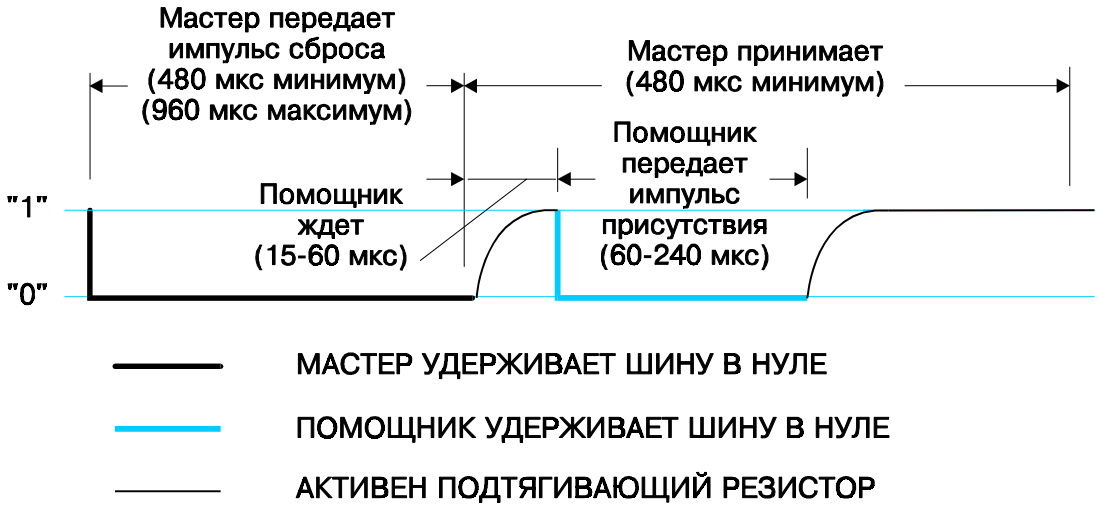


Рис. 2.2. Инициализирующая последовательность в однопроводной шине.

Временные интервалы (слоты) записи. Временной интервал записи инициируется сразу же, как только мастер переведет шину из высокого логического состояния в низкое. Имеется два типа временных интервалов записи: временные интервалы записи единицы и временные интервалы записи нуля. Каждый из временных интервалов записи должен иметь длительность не менее 60 мкс, а между соседними интервалами записи должна быть восстановительная пауза не менее 1 мкс.

Для выдачи одного временного интервала записи единицы мастер должен установить шину в низкое состояние, а затем освободить ее. В течение 15 мкс шина под действием подтягивающего резистора примет высокое логическое состояние. Для выдачи одного временного интервала записи нуля мастер должен установить шину в низкое состояние и удерживать ее в этом состоянии 60 мкс. В пределах временного окна от 15 мкс до 60 мкс после перевода шины в низкое состояние помощник проверяет состояние этой шины. Если в этот период шина находится в высоком состоянии, то помощник воспринимает этот факт как запись в него "1", если в низком, то - запись "0" (см. рис. 2.3).

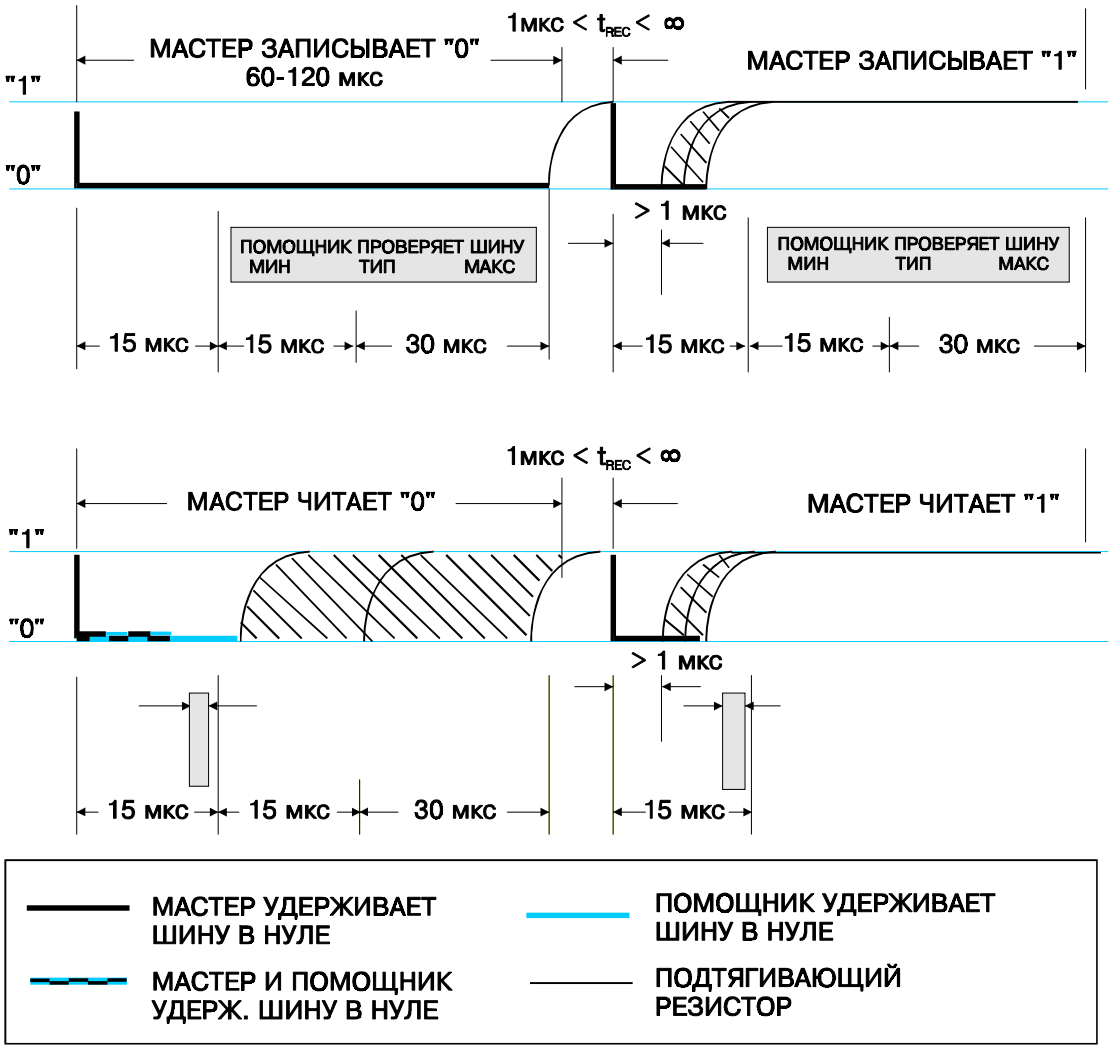


Рис. 2.3. Временные слоты чтения/записи.

Временные интервалы (слоты) чтения. Если нужно считать данные из помощника, мастер генерирует временные интервалы чтения. Временной интервал чтения инициируется сразу же, как только мастер переведет шину из высокого логического состояния в низкое. Шина данных должна оставаться в низком состоянии, как минимум, в течение 1 мкс. Данные от помощника будут истинны в течение 15 мкс после спадающего фронта временного интервала чтения. Значит, мастер через 1 мкс после перевода шины в низкое состояние (т.е. начала генерации временного интервала чтения) должен освободить шину, чтобы иметь возможность в последующие 14 мкс прочитать ее истинное значение, устанавливаемое помощником (см. рис. 2.3). По окончании временного интер-

вала чтения шина под воздействием подтягивающего резистора возвратится в высокое логическое состояние, так как помощник по истечении 15 мкс перейдет в Z-состояние. Каждый из временных интервалов чтения должен иметь длительность не менее 60 мкс, а между соседними интервалами чтения должна быть восстановительная пауза не менее 1 мкс.

На рис. 2.4 показано, что сумма времени инициализации интервала чтения T_{INIT}, времени освобождения шины T_{REC} и времени анализа шины T_{SAPMLE} не должна превышать 15 мкс. Из рис. 2.5 видно, что надежные условия чтения можно обеспечить, сводя к минимуму времена T_{INIT} и T_{REC} и сдвигая окно анализа состояния шины к концу 15-микросекундного интервала.

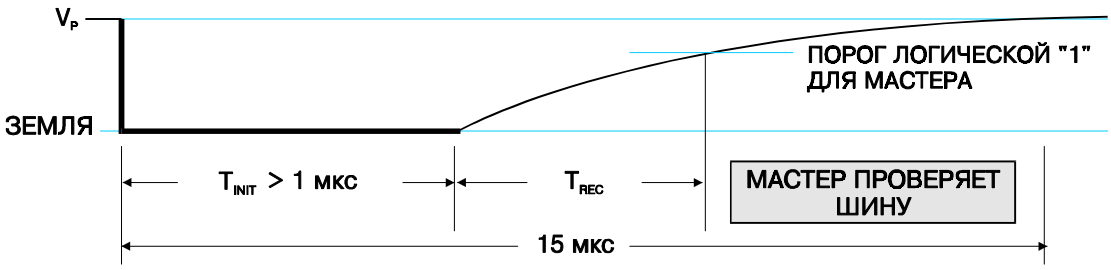


Рис. 2.4. Подробная диаграмма временного слота чтения "1".

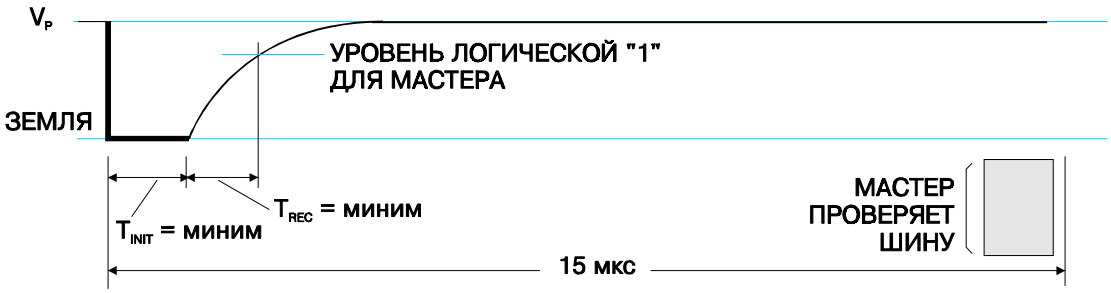


Рис.2.5.Рекомендуемая диаграмма временного слота чтения "1".

2.1.3. Последовательность обработки данных

Протокол доступа к помощнику через однопроводный порт следующий:

Инициализация

Команда функции ПЗУ

Команда функции памяти

Инициализация.

Вся обработка на однопроводной шине начинается с инициализирующей последовательности. Инициализирующая последовательность состоит из импульса сброса, посылаемого мастером шины, за которым следует(ют) импульс(ы) присутствия, посылаемый помощником (помощниками). Импульс присутствия сообщает мастеру, что помощник подключен к линии и готов к работе. Подробнее об этом было сказано выше.

Команды функции ПЗУ

Как только мастер шины определил наличие подсоединенных помощников, он может выдать одну из команд функции ПЗУ. Все команды функции ПЗУ имеют длину 8 бит. Младший бит передается первым. Список этих команд следующий:

Чтение ПЗУ (Read ROM, [33h])

Команда позволяет мастеру шины прочитать 8-битовый код принадлежности к семейству однопроводных устройств из 64-битового ПЗУ помощника, уникальный 48-битовый серийный номер и 8-битовый ЦИК. Её можно использовать, если к шине подключен только один помощник. Если к шине подключено больше помощников, то при использовании этой команды может возникнуть конфликт данных, когда все помощники сразу начнут передачу (открытый сток при этом даст результат, эквивалентный монтажному "И").

Выбор ПЗУ (Match ROM, [55h])

Эта команда, если за ней следует 64-битовый код, позволяет мастеру адресоваться к конкретному устройству на многоточечной шине. Только тот помощник, чей код в точности соответствует посланной мастером 64-битовой последовательности, откликнется на последующую команду функции памяти. Все остальные помощники будут ждать импульса сброса. Команда может использоваться как при единственном подключенном к шине помощнике, так и в случае, когда их много.

Игнорирование ПЗУ (Skip ROM, [CCh])

Команда экономит время в случае, если к шине подключен только один помощник. Она позволяет мастеру обратить-

ся к функциям памяти без предварительного генерирования 64-битового кода ПЗУ. Если на шине присутствует более одного помощника и после команды CCh следует команда чтения, возникнет конфликт данных, так как несколько помощников будут передавать одновременно (открытый сток при этом даст результат, эквивалентный монтажному "И").

Исследование ПЗУ (Search ROM, [F0h])

При первом включении системы мастер шины может не знать, сколько устройств подключено к шине, а также их коды. Данная команда позволяет мастеру методом исключения идентифицировать ПЗУ-коды всех подключенных помощников.

Аварийный опрос (Alarm Search, [ECh])

Эта команда используется только при диалоге с датчиками температуры. Её алгоритм идентичен алгоритму команды "Исследование ПЗУ". Однако датчик будет откликаться на эту команду только в том случае, если при последнем температурном измерении был зафиксирован выход за пределы выбранных порогов. Флаг тревоги остается установленным в датчике до его отключения от питания либо до первого температурного измерения, обнаруживающего, что температура устройства пришла в норму. Если флаг тревоги установлен и при этом изменены установки триггеров TL или TH, то для правильной установки состояния тревоги нужно провести еще одно измерение температуры.

Исследование ПЗУ только активных устройств
(Active-Only Search ROM, [ECh])

Используется только при диалоге с адресуемыми ключами. Она работает точно так же, как и команда "Исследование ПЗУ", за исключением того факта, что в процессе участвуют только те ключи, у которых закорачивающие транзисторы открыты. Иными словами, с помощью этой команды мастер может из общего числа адресуемых ключей выделить те из них, которые находятся в активном состоянии (см. главу 4).

Команды функции памяти

После успешного выполнения команды функции ПЗУ мастер может выдать одну из команд функции памяти. Эта группа

команд используется для связи с датчиками температуры. Их подробное описание приведено в главе 3.

2.2. Проверка истинности данных циклическим избыточным кодом

2.2.1. Общие положения

Семейство устройств с однопроводным интерфейсом фирмы Dallas Semiconductor отличает та особенность, что связь с ними осуществляется через единственный провод посредством специальной системы команд, называемой однопроводным протоколом. Отличительной чертой каждого из этих устройств является встроенное 8-байтное ПЗУ, содержащее уникальный идентификационный код устройства. Содержимое этого 8-байтного кода показано на рис. 2.6.

Рис. 2.6. Содержание 8-байтного ПЗУ. (СЗР - старший значащий разряд; МЗР - младший значащий разряд).

Младший байт содержит код, свидетельствующий о принадлежности к семейству устройств с однопроводным интерфейсом, определяя одновременно функциональное назначение прибора. Например, DS1820 имеет код 10h, DS1990A - код 01h, DS1991 - код 02h и т.д.. Поскольку к одному коммуникационному проводу может быть подключено сразу несколько устройств различного функционального назначения, процессор должен иметь возможность безошибочно разобраться, какие именно устройства подключены. Код принадлежности помогает решить эту задачу. Следующие 6 байт ПЗУ содержат уникальный серийный номер конкретного экземпляра, позволяющий отличить его от других устройств данного семейства. Этот 48-битовый код можно рассматривать как индивидуальный адрес устройства. Старший байт ПЗУ содержит результат проверки истинности данных циклическим избыточным кодом - ЦИК (cyclic redundancy check - CRC). Этот код вычисляется на основе содержимого предыдущих

семи байтов. Когда процессор начинает сеанс связи с устройством, он первым делом читает 8 байт ПЗУ этого устройства, причем младший значащий бит читается первым. Если ЦИК, рассчитанный процессором на основе принятых данных (семи младших байтов), совпадает с содержимым 8-го байта ПЗУ, то считается, что данные приняты верно. Если эти значения не совпадают, то процесс чтения ПЗУ повторяется.

Некоторые из устройств, в дополнение к восьми байтам ПЗУ, имеют до 64кбайт оперативного запоминающего устройства (ОЗУ), доступ к которому осуществляется посредством специальных команд. Данные в эту память записываются процессором обычным образом, затем к ним добавляется рассчитанный процессором ЦИК и также записывается в ПЗУ устройства. При получении (чтении) данных из устройства процесс - обратный. Процессор, приняв данные от устройства, рассчитывает на их основе ЦИК и сравнивает его со значением, хранимым в памяти устройства как ЦИК данных. Если значения ЦИК одинаковы, то данные приняты верно. Чтобы в полной мере воспользоваться преимуществами, даваемыми данной системой проверки истинности, нужно понять, что она собой представляет и как работает. Кроме того, необходимо освоить практический метод расчета значений ЦИК процессором как в программной, так и в аппаратной реализации.

Проверка на истинность последовательных данных может осуществляться различными способами. Распространенным способом является включение в каждый проверяемый пакет данных дополнительного бита, указывающего на наличие или отсутствие ошибки. Например, при передаче 8-битовых пакетов ASCII символов к каждому 8-битовому пакету добавляется один бит (бит паритета), указывающий, есть или нет ошибки при передаче этого пакета. Предположим, что пакет данных состоит из битовой строки 11010001. Девятый бит добавляется с таким расчетом, чтобы суммарное количество битов, содержащих единицу, было нечетным. Таким образом, в приведенный выше битовый пакет должна быть добавлена единица. Теперь он будет выглядеть как 1̲11010001. Подчеркнутый символ представляет собой дополнительный бит, так называемый бит паритета, вводимый для того, чтобы сделать суммарное количество единичных битов в 9-битовом пакете нечетным. Если приемник принял пакет 111010001, то

данные переданы и приняты верно. Если же приемник принял, например, 111010101, где седьмой слева бит принят неверно, то общее количество единиц в пакете уже не будет нечетным, что свидетельствует о наличии ошибки. Обнаружив ошибку, приемник предпримет соответствующие действия. Такая схема называется проверкой на нечетность. Можно договориться о том, чтобы общее число единиц в пакете было четным. Эта схема называется проверкой на четность. Обе схемы, к сожалению, могут обнаружить только нечетное количество битовых ошибок. Если в рассмотренном только что примере пакет будет поврежден сразу в 6-м и 7-м битах, т.е. будет принята последовательность 111011101, то контроль паритета по нечетности не обнаружит ошибки, так как общее количество единичных битов будет нечетным. Указанная ошибка не будет обнаружена и в случае контроля на четность.

2.2.2. Однопроводный ЦИК фирмы Dallas Semiconductor

Схема проверки истинности потока данных, основанная на вычислении циклического избыточного кода, является наиболее эффективной при локализации ошибок и требует минимальных аппаратных затрат. Опишем применяемую Dallas Semiconductor схему определения ошибок, не приводя сложных математических обоснований метода. Работу схемы проще всего пояснить на примере ее аппаратной реализации. Аппаратно схема представляет собой сдвиговый регистр с обратной связью (рис. 2.7).

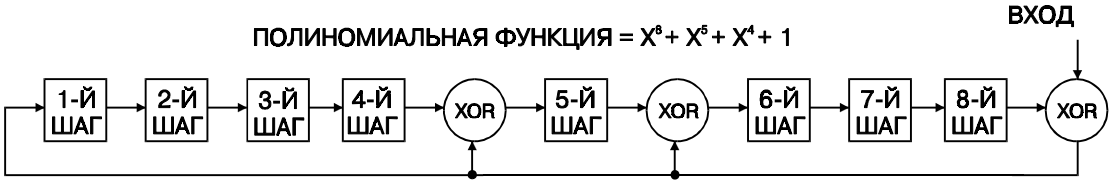


Рис. 2.7. Аппаратная реализация 8-битового алгоритма проверки истинности данных циклическим избыточным кодом, соответствующая полиномиальной функции $X^8 + X^5 + X^4 + 1$.

Процедуру выделения ошибки часто описывают в виде полиномиальной функции формальной переменной X. Показатели степени ненулевых членов этого полинома показывают, на какие разряды сдвигового регистра замыкаются петли

обратной связи. Число разрядов сдвигового регистра (в аппаратной реализации) или степень полинома (при математическом описании) определяют разрядность вычисляемого ЦИК. Обычно при цифровом обмене данными используют 16-разрядные ЦИК. Dallas Semiconductor при чтении 64-битовых ПЗУ своих устройств использует 8-разрядные ЦИК. 64-битовый код ПЗУ состоит из 8-битового кода принадлежности к семейству устройств, поддерживающих однопроводный интерфейс, 48-битового серийного номера устройства и 8-битового ЦИК, рассчитанного на основе первых 56 битов содержимого ПЗУ. ЦИК содержится в старшем байте ПЗУ. Расположение точек входа петель обратной связи, обозначенных на рис. 2.7 логическими элементами “исключающее ИЛИ” (или, что то же самое, степени ненулевых членов полинома), определяет свойства ЦИК и способность алгоритма выявить определенные классы ошибок при передаче данных. Алгоритм, приведенный на рис. 2.7, способен распознать следующие классы ошибок:

- Любое нечетное число ошибок в любом месте 64-битовой последовательности.
- Все двойные ошибки (ошибки в двух смежных битах) в любом месте 64-битовой последовательности.
- Любые кластеры ошибок в пределах 8-битового “окна”, т.е. кластеры, содержащие от 1 до 8 неправильных битов.
- Большинство кластерных ошибок с размерами кластера, превышающими 8 бит.

Входные данные подаются на один из входов двухвходового логического элемента “исключающее ИЛИ” (XOR). Ко второму входу этого элемента подключается выход старшего разряда 8-разрядного сдвигового регистра (рис. 2.7). Математически сдвиговый регистр может рассматриваться как схема деления. Входные данные - это делимое, а сдвиговый регистр с цепями обратной связи служит делителем. Целая часть полученного частного отбрасывается, а остаток представляет собой искомый ЦИК, который окажется в сдвиговом регистре, как только последний бит данных пройдет на вход. Из такого аппаратного описания алгоритма видно, что результат (значение ЦИК) очень сложным образом зависит от предыстории контролируемой последовательности

битов. Поэтому комбинации ошибок, способные пройти незамеченными через такой контроль, очень редки.

В табл. 2.1 показан пошаговый расчет ЦИК исходной 64-битовой последовательности. Каждая строка левого столбца отражает содержимое сдвигового регистра после подачи на вход очередного бита из 64-битовой последовательности пересылаемых данных. Первым на вход подается младший бит младшего байта. В начале процедуры все биты сдвигового регистра сброшены (т.е. представляют собой нули). Расчет начинается подачей младшего бита 64-битовой последовательности. В этом примере младший байт равен 02h и представляет собой код принадлежности к семейству устройств с однопроводным интерфейсом. После того как последний, 56-й бит (код принадлежности + серийный номер) будет введен в сдвиговый регистр, в нем окажется число 10100010 (или A2h в 16-ричном выражении). Это и будет ЦИК для введенной последовательности из 56 битов. Если это число использовать в качестве входной последовательности для следующих восьми шагов, то в результате в сдвиговом регистре останутся одни нули. Иными словами, если вы проверяете полное 64-битовое ПЗУ, хранящее в своем старшем байте ЦИК первых 56 бит, то, не ограничиваясь при вводе 56-ю битами, а вводя все 64, получите очищенный сдвиговый регистр (все нули). Отмеченный факт является свойством описываемого алгоритма. Его можно сформулировать и в более общей форме. Если любое 8-битовое число, находящееся в данный момент в сдвиговом регистре, использовать в качестве входной последовательности для последующих 8 шагов сдвига, то результатом будет очищенный регистр, все разряды которого содержат нуль. Это можно объяснить следующим образом. При описанном выше условии (равенстве содержимого регистра и входной последовательности) выход старшего разряда регистра (8-й шаг) всегда будет равен входному биту данных, а на выходе логического элемента “исключающее ИЛИ” всегда будет нуль. Этот нуль по цепи обратной связи поступает в младший разряд сдвигового регистра (1-й шаг). При поступлении на вход следующего бита нулевой бит младшего разряда регистра сдвинется в следующий разряд, а на его место поступит нулевой же бит с выхода элемента “исключающее ИЛИ”. После восьми таких шагов во всех разрядах сдвигового

регистра окажутся нули. Структура 64-битового ПЗУ в однопроводных устройствах Dallas Semiconductor использует указанное свойство для упрощения аппаратного обеспечения устройств, применяемых для чтения ПЗУ. Сдвиговый регистр читающего устройства очищается, а затем читаются 64 бита данных из ПЗУ контролируемого устройства, включая и байт ЦИК. Если данные прочитаны верно, то регистр читающего устройства будет снова содержать нули, что легко фиксируется. Если же в сдвиговом регистре окажется ненулевое значение, операцию чтения нужно повторить.

Таблица 2.1. Пример расчета ЦИК 64-битовой последовательности.

Содержание 64-битового ПЗУ: A200000001B81C02 Hex

Код семейства:

0	2	Шестнадцитиричный (Hex)
0000	0010	Двоичный (Bin)

Серийный номер:

0	0	0	0	0	0	0	1	B	8	1	C	Hex
0000	0000	0000	0000	0000	0000	0000	0001	1011	1000	0001	1100	Bin

ЗНАЧЕНИЕ ЦИК	ВХОДНОЕ ЗНАЧЕНИЕ	
00000000	0	
00000000	1	
10001100	0	2
01000110	0	
00100011	0	
10011101	0	
11000010	0	0
01100001	0	
10111100	0	
01011110	0	
00101111	1	C
00010111	1	
00001011	1	
00000101	0	
10001110	0	1
01000111	0	
10101111	0	
11011011	0	
11100001	0	8
11111100	1	
11110010	1	
11110101	1	
01111010	0	B
00111101	1	
00011110	1	
10000011	0	

11001101	0	1
11101010	0_____	
01110101	0	
10110110	0	
01011011	0	0
10100001	0_____	
11011100	0	
01101110	0	
00110111	0	0
10010111	0_____	
11000111	0	
11101111	0	
11111011	0	0
11110001	0_____	
11110100	0	
01111010	0	
00111101	0	0
10010010	0_____	
01001001	0	
10101000	0	
01010100	0	0
00101010	0_____	
00010101	0	
10000110	0	
01000111	0	0
10101101	0_____	
11011010	0	
01101101	0	
10111010	0	0
01011101	0_____	

10100010 = A2H = ЦИК для [00000001B81C(Серийный номер) + 02(Код семейства)]

10100010	0	
01010001	1	
00101000	0	2
00010100	0_____	
00001010	0	
00000101	1	
00000010	0	A
00000001	1_____	

00000000 = 00H = ЦИК для [A2(ЦИК) + 00000001B81C(Серийный номер) + 02(Код семейства)].

До сих пор мы обсуждали аппаратную реализацию процесса вычисления ЦИК. Но можно вычислить ЦИК и программным способом. В табл. 2.2 приведен ассемблерный код такой процедуры. Операция XRL над содержимым регистра A и константой 18h соответствует в аппаратной реализации наличию логических элементов “исключающее ИЛИ” на входах четвертого и пятого разрядов сдвигового регистра (см. рис. 2.7).

Таблица 2.2. Ассемблерная процедура для подсчета 8-битового ЦИК.

```
DO_CRC:     PUSH    ACC         ;сохраняем аккумулятор
            PUSH    B           ;сохраняем регистр B
            PUSH    ACC         ;сохраняем биты,
                                ;подлежащие сдвигу
            MOV     B,#8        ;устанавливаем сдвиг = 8 бит

CRC_LOOP:   XRL     A,CRC       ;подсчитываем ЦИК
            RRC     A
            MOV     A,CRC       ;получаем последнее
                                ;значение ЦИК
            JNC     ZERO        ;пропускаем, если данные = 0
            XRL     A,#18H      ;обновляем значение ЦИК

ZERO:       RRC     A           ;позиционируем новыйЦИК
            MOV     CRC,A       ;запоминаем новый ЦИК
            POP     ACC         ;получаем оставшиеся биты
            RR      A           ;позиционируем следующий
                                ;бит
            PUSH    ACC         ;сохраняем оставшиеся биты
            DJNZ    B,CRC_LOOP  ;повторяем для 8-ми битов
            POP     ACC         ;очищаем стек
            POP     B           ;восстанавливаем регистр B
            POP     ACC         ;восстанавливаем
                                ;аккумулятор
            RET
```

Другим программным решением является создание таблицы преобразования, определяющей связь между любым текущим 8-битовым значением регистра ЦИК и любым 8-битовым отрезком входных данных. В простейшем случае, когда текущее значение регистра ЦИК равно 00h, можно рассчитать 256 возможных битовых комбинаций входного байта и разместить их в матрице, где индекс матрицы будет равен значению входного байта (т.е. индекс будет пробегать значения от 0 до 255, и число, размещенное в i-й позиции, будет равно i). Можно показать, что если текущее значение регистра ЦИК отлично от нуля, то для любого заданного значения ЦИК и любого входного байта таблица преобразования будет состоять из тех же 255 значений, но расставленных в таблице в соответствии с формулой:

Новое значение ЦИК = Таблица (i),

где i = (Текущее значение ЦИК) XOR (Входной байт).

Когда текущее значение ЦИК равно 00h, данная формула приводит к описанному выше простейшему случаю. Второе программное решение сокращает время расчета, так как операции производятся не над битами, как в первом варианте, а над байтами. Однако в этом случае требуется 256 байт дополнительной памяти для размещения таблицы преобразования. Для первого варианта программного решения память требуется только для размещения программного кода. Пример программного кода для второго варианта приведен в табл. 2.3. Табл. 2.4 иллюстрирует использование метода таблицы преобразования на уже приводившемся ранее примере. Два свойства алгоритма вычисления ЦИК могут помочь в отладке программного кода. Первое из них уже упоминалось при описании аппаратной реализации алгоритма, а именно: если текущее значение регистра ЦИК использовать в качестве следующего входного байта, то в результате в регистре ЦИК окажутся одни нули. Второе свойство заключается в том, что если в качестве входной последовательности использовать байт, инверсный байту, содержащемуся в данный момент в регистре ЦИК, то через восемь шагов в регистре ЦИК окажется число 35h (53 в десятичном выражении). Эта процедура показана в табл. 2.5.

Таблица 2.3. Программный код, вычисляющий 8-битовый ЦИК по методу таблицы преобразования.

```
Var
  CRC : Byte
Procedure Do_CRC(X: Byte);
{
        Эта процедура вычисляет ЦИК для всех байтов, проходящих через однопроводную шину. Результат накапливается в глобальной
переменной CRC.
}
Const
   Table : Array[0...255] of Byte = (
                0,    94,   188,   226,    97,    63,   221,   131,   194,   156,   126,    32,   163,   253,    31,    65,
              157,   195,    33,   127,   252,   162,    64,    30,    95,     1,   227,   189,    62,    96,   130,   220,
               35,   125,   159,   193,    66,    28,   254,   160,   225,   191,    93,     3,   128,   222,    60,    98,
              190,   224,     2,    92,   223,   129,    99.    61,   124,    34,   192,   158,    29,    67,   161,   255,
               70,    24,   250,   164,    39,   121,   155,   197,   132,   218,    56,   102,   229,   187,    89,     7,
              219,   133,   103,    57,   186,   228,     6,    88,    25,    71,   165,   251,   120,    38,   196,   154,
              101,    59,   217,   135,     4,    90,   184,   230,   167,   249,    27,    69,   198,   152,   122,    36,
              248,   166,    68,    26,   153,   199,    37,   123,    58,   100,   134,   216,    91,     5,   231,   185,
              140,   210,    48,   110,   237,   179,    81,    15,    78,    16,   242,   172,    47,   113,   147,   205,
               17,    79,   173,   243,   112,    46,   204,   146,   211,   141,   111,    49,   178,   236,    14,    80,
              175,   241,    19,    77,   206,   144,   114,    44,   109,    51,   209,   143,    12,    82,   176,   238,
               50,   108,   142,   208,    83,    13,   239,   177,   240,   174,    76,    18,   145,   207,    45,   115,
              202,   148,   118,    40,   171,   245,    23,    73,     8,    86,   180,   234,   105,    55,   213,   139,
               87,     9,   235,   181,    54,   104,   138,   212,   149,   203,    41,   119,   244,   170,    72,    22,
              233,   183,    85,    11,   136,   214,    52,   106,    43,   117,   151,   201,    74,    20,   246,   168,
              116,    42,   200,   150,    21,    75,   169,   247,   182,   232,    10,    84,   215,   137,   107,    53);

Begin
  CRC : = Table[CRC xor X];
End;
```

Таблица 2.4. Использование метода таблицы преобразования для вычисления ЦИК (в примере приведены те же исходные данные, что и в табл. 2.1).

Текущее значение ЦИК (= текущему табличному индексу)	Входные данные	Новый индекс (= результату операции "Текущий ЦИК" XOR "Входные данные")	Новая позиция (индекс) в таблице (= новому значению ЦИК)
00000000 = 00H	00000010 = 02H	(00H XOR 02H) = 02H = 2Dec	Table[2] = 10111100 = BCH = 188Dec
10111100 = BCH	00011100 = 1CH	(BCH XOR 1CH) = A0H = 160Dec	Table[160] = 10101111 = AFH = 175Dec
10101111 = AFH	10111000 = B8H	(AFH XOR B8H) = 17H = 23Dec	Table[23] = 00011110 = 1EH = 30Dec
00011110 = 1EH	00000001 = 01H	(1EH XOR 01H) = 1FH = 31Dec	Table[31] = 11011100 = DCH = 220Dec
11011100 = DCH	00000000 = 00H	(DCH XOR 00H) = DCH = 220Dec	Table[220] = 11110100 = F4H = 244Dec
11110100 = F4H	00000000 = 00H	(F4H XOR 00H) = F4H = 244Dec	Table[244] = 00010101 = 15H = 21Dec
00010101 = 15H	00000000 = 00H	(15H XOR 00H) = 15H = 21Dec	Table[21] = 10100010 = A2H = 162Dec
10100010 = A2H	10100010 = A2H	(A2H XOR A2H) = 00H = 0Dec	Table[0] = 00000000 = 00H = 0 Dec

В таблице: H - шестнадцитиричное число: Dec - десятичное число.

Таблица 2.5. Изменение содержимого сдвигового регистра генератора ЦИК при подаче на его вход байта, комплементарного байту, содержащемуся в нем в данный момент.

Значение регистра ЦИК								Вход
X_0	X_1	X_2	X_3	X_{4*}	X_{5*}	X_6	X_7	X_7^*
1	X_0	X_1	X_2	X_{3*}	X_4	X_{5*}	X_6	X_{6*}
1	1	X_0	X_1	X_{2*}	X_3	X_4	X_{5*}	X_{5*}
1	1	1	X_0	X_1	X_{2*}	X_3	X_4	X_{4*}
0	1	1	1	X_0	X_{1*}	X_{2*}	X_3	X_{3*}
1	0	1	1	0	X_0	X_{1*}	X_{2*}	X_{2*}
1	1	0	1	0	1	X_0	X_{1*}	X_{1*}
0	1	1	0	1	0	1	X_0	X_0
0	0	1	1	0	1	0	1	Z

Z = 35H = 53Dec (конечное значение ЦИК)
X_i^* - значение, комплементарное X_i.

Глава 6. Использование технологии MicroLAN для температурного мониторинга объектов коммунального хозяйства

6.1. Новые подходы к эксплуатации зданий и сооружений

Анализ литературы по строительству и коммунальному хозяйству за последние несколько лет показывает, что в среде специалистов формируется новое понимание проблемы эксплуатации зданий, сооружений, вообще любых технических конструкций. Обращает на себя внимание тот факт, что все предлагаемые новые технологии эксплуатации либо представляют собой средства для получения значений эксплуатационных параметров контролируемого объекта, либо базируются на необходимости иметь подробные и достоверные данные об объекте. Иными словами, все они проникнуты идеей технологического мониторинга эксплуатируемых объектов. Мировая практика эксплуатации технических конструкций и сооружений сейчас претерпевает существенные изменения. Она идет по пути создания технологии управления эксплуатацией конструкций, когда можно будет в любой момент времени знать состояние конструкции и предсказывать ее поведение в будущем. В конечном итоге такой подход должен привести к созданию системы постоянного мониторинга эксплуатационных параметров (температуры, влажности, напряжений в несущих конструкциях, теплопроводности ограждающих конструкций), опрос охранной и пожарной сигнализации и т.д. Такая система призвана своевременно и автоматически обеспечивать персонал всех уровней управления достоверной объективной информацией, необходимой для принятия решений по оперативному управлению зданиями и сооружениями.

Возможно, идея снабдить каждый дом, квартиру комплектом всевозможных датчиков покажется избыточной, дорогой и преждевременной. Но это только на первый взгляд. Вспомните, еще полвека назад большинство жилого фонда в СССР не имело центрального отопления, не говоря уже о горячем водоснабжении. Сейчас уже речь идет об установке поквартирных счетчиков всех видов ресурсов, об индивидуальных ре-

гуляторах микроклимата. Все изменяется, и даже жилье наше становится все более насыщенным инженерным оборудованием. Что касается дороговизны, то расчеты показывают, что доля затрат на инженерное оборудование зданий составляет сейчас не более 9% капитальных затрат. Что же до компьютеров, то стоимость их постоянно снижается. Так, стоимость процессоров с 1975 г. снизилась на три порядка. Предполагается, что к 2010 году она уменьшится еще во столько же раз. Дисковая память дешевеет на 60% в год и составляет сейчас около 5 центов за 1 Мбайт. К 2010 году она предположительно будет равна 5 центам за 1 Гбайт. Уже сейчас компьютер типовой конфигурации с процессором i80486 (вполне достаточный для использования в качестве мастера разветвленной MicroLAN) стоит примерно 150 долларов США. Описанные нами выше датчики температуры фирмы “Dallas Semiconductor” стоят от 3 до 5 долларов за штуку (это при поштучной продаже, оптовая цена существенно ниже). Так что, например, дешевые системы подробного температурного мониторинга можно создавать уже сейчас.

Требование рациональной эксплуатации зданий и сооружений предполагает, что уже недостаточно оценивать их состояние “на глазок”. Нужна четкая объективная оценка. Поэтому системы всеобъемлющего мониторинга являются не только своевременными, но и необходимыми. Не последнюю роль в этом нашем заключении играет и то соображение, что здания строятся для человека и человеку в этих зданиях должно быть комфортно. Обеспечить комфорт дифференцированно, по потребностям, а не нынешний среднестатистический комфорт, могут только такие системы автоматического регулирования, которые обладают развитой обратной связью и способны получать и анализировать информацию о состоянии каждого конкретного помещения.

Говоря здесь о температурном мониторинге, приведем ряд конкретных примеров его необходимости. Известно, что в системе теплоснабжения большие потери тепла происходят на теплотрассах. Большая доля этих потерь санкционирована. И оплачивает их потребитель. Но много потерь тепла происходит и у потребителя, главным образом за счет несовершенных теплозащитных свойств зданий. Поэтому важной становится задача определения структуры потерь, их картографирование. С этой целью необходимо проводить темпера-

турный мониторинг тепловых сетей и зданий. Такие попытки уже предпринимаются. При обследовании состояния тепловых сетей и ограждающих конструкций зданий проводятся как дистанционные тепловизионные, так и контактные теплометрические измерения. Однако для масштабных работ этого направления, а тем более для обеспечения постоянного теплового мониторинга теплосетей и зданий еще не хватает достаточно дешевого приборного обеспечения, да и методики таких обследований не разработаны.

Для экономии энергоресурсов большое значение имеет проведение энергоаудита предприятий или отдельных зданий. По нашему мнению, и здесь система технологического мониторинга должна сыграть положительную роль. Прежде всего потому, что она, по сути, выполняет задачи энергетического аудита, причем выполняет их не разово, а постоянно, непрерывно, а значит, более качественно. Кроме того, она и стоить будет дешевле, поскольку установлена стационарно и выполняет еще и другие функции.

Системы технологического мониторинга, в частности, температурного, необходимы и при осуществлении новой стратегии планирования ремонтов инженерного оборудования жилищно-коммунального хозяйства. Эта стратегия, называемая “обслуживание и ремонт в зависимости от технического состояния”, приходит на смену стратегии регламентных работ в определенные временные интервалы и позволяет сэкономить материальные и энергетические ресурсы в тех случаях, когда срок регламентных работ наступил, а показания технологического мониторинга говорят о хорошем состоянии объекта. В таких случаях регламентные работы можно не проводить. И наоборот, если срок регламентных работ еще не наступил, а мониторинг показывает, что состояние объекта ухудшилось, то ремонт проводится в соответствии с фактическим состоянием объекта.

6.2. Автоматизированные системы контроля объектов теплоснабжения и управления ими

Чтобы лучше понять место микролокальных сетей типа MicroLAN в ряду систем технологического мониторинга, рассмотрим кратко существующие принципы организации авто-

матизированных систем контроля и управления технологическими процессами (АСУТП) для нужд теплоснабжения. Теория и практика АСУТП производства, транспортировки, распределения и потребления тепла хорошо описана в литературе. Каждая из этих систем создается для конкретного объекта, но в основе всех их лежат, как правило, хорошо известные типовые технические решения. Это касается структуры АСУТП, ее аппаратной платформы, набора первичных преобразователей, каналов связи, да и программного обеспечения, если иметь в виду не функциональные особенности, а организацию графического интерфейса пользователя. По выполняемым задачам эти системы можно разделить на три группы:

- системы, осуществляющие учет и контроль (измерительные системы);
- системы, осуществляющие автоматическое управление;
- комбинированные системы, выполняющие обе эти функции.

6.2.1. Структура АСУТП

Современные АСУТП строятся по иерархическому принципу. В наиболее совершенной схеме каждый иерархический уровень представляет собой гибкую подсистему, которую можно перестраивать (конфигурировать) применительно к конкретным обстоятельствам. На практике используются двухуровневые, трехуровневые и четырехуровневые АСУТП. Так, четырехуровневая система состоит из:

- совокупности первичных приборов и датчиков, исполнительных устройств (первый уровень);
- локальных сетей, включающих один или группу измерительно-управляющих контроллеров (второй уровень);
- каналов связи, обеспечивающих двунаправленную связь между локальными сетями и центральным диспетчерским пунктом (третий уровень);
- центрального диспетчерского пункта (четвертый уровень).

На практике каждая конкретная система не обязательно включает все составляющие этой схемы. Она конфигурируется в зависимости от решаемых задач, конкретных условий рабо-

ты, стоимости и т.д.

Принцип построения АСУТП (структура, тип используемого интерфейса, каналов связи) зависит от территориальной рассредоточенности объекта управления. По этому признаку различают сосредоточенные системы (длина каналов связи до 2 м), локальные (до 20 м) и рассредоточенные (более 20 м).

6.2.2. Выбор платформы

До недавнего времени АСУТП базировались на использовании так называемых гибко программируемых контроллеров. Однако в последние годы серьезную конкуренцию таким системам стали оказывать АСУТП на базе персональных компьютеров (PC-based Control). Сейчас нельзя сказать, что какое-либо из этих направлений непременно победит. Скорее всего, будущее за их рациональным сочетанием. Системы на базе персональных компьютеров (ПК) имеют преимущества в тех случаях, когда, кроме собственно задач технологического управления, решаются задачи визуализации технологических процессов, обработки данных. Гибко программируемые контроллеры целесообразно применять на тех участках АСУТП, где решаются узкие задачи сбора данных и управления (например, на втором уровне иерархической структуры по приведенной выше четырехуровневой классификации). В настоящее время практически используются три подхода к проектированию АСУТП: в простейших случаях применяются комплексы на основе микросхем высокого и среднего уровней интеграции; более сложные системы строятся на базе программируемых контроллеров или на базе ПК (как вариант - однокристалльных микро-ЭВМ). Учитывая доступность и сравнительную дешевизну современных IBM-совместимых компьютеров, все чаще и чаще разработчики АСУТП базируются на них.

6.2.3. Аппаратный интерфейс

Для организации связи ЭВМ или контроллера с периферийной аппаратурой в современных АСУТП чаще всего используются открытые стандарты, позволяющие объединять в одной схеме аппаратные и программные средства различных

производителей. К числу наиболее популярных стандартов относятся:

- “Hewlett-Packard Interface Bus” (HP-IB) - система интерфейса для измерительных устройств с байт-последовательным, бит-параллельным обменом информацией; этот интерфейс осуществляет обмен информацией на расстоянии до 20 м со скоростью до 1Мбит/с. У нас он известен как “канал общего пользования”;
- последовательный интерфейс стандарта Ассоциации электронной промышленности (Electronics Industries Assosiation - EIA) - RS-232C. Это однопроводный несимметричный интерфейс, осуществляющий побитовый последовательный обмен данными по специальному протоколу; особую популярность RS-232C придает тот факт, что наиболее распространенные компьютеры - компьютеры семейства IBM - комплектуются этим интерфейсом для связи с внешними устройствами;
- последовательные интерфейсы стандартов RS-422, RS 485. Это симметричные однопроводные интерфейсы, позволяющие к одной линии подключать несколько приемников (RS-422), а также несколько приемников и передатчиков (RS-485). Эти интерфейсы позволяют обмениваться информацией на расстоянии до 1200 м со скоростью до 10 Мбит/с.

Отдельно отметим платы расширения, использующие шинный интерфейс стандартов ISA, EISA, UESA LOCAL BUS, PCI. Эти платы вставляются непосредственно в слот расширения системной платы компьютера и представляют собой устройства самого разного назначения: аналого-цифровые и цифро-аналоговые преобразователи, системы сбора и накопления данных, коммутаторы и усилители сигналов и т.д. Номенклатура этих плат очень широка, как и велико множество фирм, их производящих. При несомненно высоких метрологических параметрах эти платы, тем не менее, имеют и высокую стоимость, что делает их не всегда доступными. Поэтому многие разработчики АСУТП создают собственные платы расширения - недорогие и адаптированные к конкретным нуждам.

6.2.4. Первичные преобразователи (датчики)

В системах автоматического управления теплоснабжением обычно измеряются, обрабатываются и регулируются следующие физические параметры технологического оборудования: температура, давление, расход, уровень, электрический ток, напряжение.

Для проведения постоянного или хотя бы контрольного мониторинга зданий и сооружений нужны прежде всего дешевые и надежные первичные преобразователи (датчики). Накапливающие и обрабатывающие системы - в лице персональных компьютеров - уже есть и они доказали свою надежность. Слабой стороной систем сбора и обработки данных сейчас являются датчики и каналы связи. Практика эксплуатации таких систем показала, что основным их недостатком является невозможность обеспечить надежную передачу данных по каналам связи (из-за влияния индустриальных помех, кратковременных и долговременных отключений питания и т.д.). В результате непрерывный процесс накопления и обработки данных прерывается и целые пакеты данных пропадают. Восстановление их, скажем, путем экстраполяции вносит искажения в реальную картину, порой довольно существенные. С другой стороны, практически все применяемые в настоящее время датчики имеют аналоговый выходной сигнал, который также искажается помехами, действующими как непосредственно на датчик, так и на каналы связи. Поэтому существенно, чтобы датчик имел возможность преобразовать измеренную им аналоговую величину в цифровой код и по каналам связи передать этот код накапливающему и обрабатывающему устройству. То есть измерительный преобразователь должен быть *телеметрическим*.

6.2.5. Программное обеспечение

Если аппаратная часть АСУТП более или менее стандартизована и отдельные ее компоненты выпускаются большими тиражами, то программное обеспечение (ПО) для каждой задачи уникально. Поэтому не представляется возможным даже перечислить все программные продукты, разработанные для всевозможных АСУТП. Но и в мире ПО для измерительных и

управляющих систем просматривается определенная тенденция к универсализации и стандартизации. Прежде всего это касается пользовательского интерфейса, т.е. организации общения оператора с компьютером. Интерактивный диалоговый интерфейс давно уже стал нормой в мире программных продуктов. Кому не известны знаменитые Windows различных версий и модификаций? В последние несколько лет в программном обеспечении измерительных технологий произошла настоящая революция - появились и прочно завоевали рынок виртуальные измерительные средства (ВИС). Под ВИС понимаются средства измерений, построенные на базе ПК, многофункциональных встраиваемых в компьютеры плат расширения, внешних программно-управляемых модулей, объединенных и управляемых специализированными программными оболочками, позволяющими управлять алгоритмами сбора, обработки и визуального представления измерительной информации. Такие измерительные средства называются виртуальными в связи с тем, что:

- с помощью одного и того же аппаратного и программного обеспечения можно сконфигурировать системы, выполняющие различные функции и имеющие разные графические интерфейсы пользователя;
- управление такими системами осуществляется с помощью технологии drag-and-drop через элементы управления виртуальных приборных панелей, высвечиваемых на дисплее компьютера.

6.3. Сети температурного мониторинга на основе технологии MicroLAN

В предыдущих главах мы познакомились с технологией микролокальных сетей сбора информации типа MicroLAN: принципами организации сетей, а также с их аппаратными составляющими. Ниже мы приведем пример конкретного использования этой технологии для целей температурного мониторинга. На основе этой технологии мы постарались создать простую в эксплуатации, дешевую, гибкую и надежную систему сбора и обработки информации о температуре протяженного объекта (здания, цеха, участка тепло-

трассы и т.п.). При этом мы стремились исключить избыточность конфигурации сети, то есть создать сеть максимально простой топологии и с минимальным набором компонент. Говоря о минимальном наборе компонент, мы имеем в виду компоненты служебного характера, такие как адресуемые ключи. Эти компоненты не поставляют информацию о температуре объекта, но, тем не менее, нагружают линию, сокращая количество подключенных к этой линии поставщиков информации - датчиков температуры.

6.3.1.Простейшая топология MicroLAN

Описанная в главе 5 оптимальная топология допускает подключение нескольких тысяч датчиков. Но при этом в сети присутствуют адресуемые ключи, временные параметры протокола ужесточаются, усложняется программное обеспечение. В то же время простая топология - без ветвлений - ограничивает количество датчиков максимальным числом 142 (табл. 5.1).

Имея в виду, что типовые телефонные кабели имеют, как правило, не менее 4-х жил, можно в одном кабеле осуществить несколько коммуникационных линий. И дело здесь не в том, что возможны взаимные помехи. В конце концов, можно применить экранированные витые пары. Задача в том, где взять дополнительные COM-порты для обеспечения каждой из линий своим мастером шины. Операционные системы типовых IBM-совместимых компьютеров могут обслуживать до четырех последовательных портов. На практике таких портов в компьютере всего два. Чаще всего один из них занят мышью. Таким образом, для организации мониторинговой сети остается один порт, т.е. одна линия для последовательного обмена данными. Сетей с большим количеством датчиков на такой базе не построишь. Кроме того, для электрического согласования уровней напряжения COM-порта и датчика необходимо применять специальный адаптер (например, DS2480 [3]). Подключать линии к порту принтера тоже невыгодно по ряду причин. Во-первых, не во всех моделях компьютеров порты принтера устроены одинаково. Во-вторых, они не могут обеспечить обмен информацией с большим количеством датчиков. В-третьих, порт принтера часто бывает

занят собственно принтером (особенно в системах сбора и обработки информации, где информацию нужно оперативно выводить на печать).

Оптимальное решение заключается в том, чтобы создать собственный аппаратный интерфейс, дополняющий типовой набор устройств ввода/вывода компьютера и отвечающий стандартам однопроводного обмена MicroLAN. Авторами разработан такой интерфейс, расширяющий доступ к пространству ввода/вывода компьютера и позволяющий увеличить число линий сети до 32. Для этого необязательно увеличивать и количество кабелей тоже в 32 раза. Как мы уже упоминали, типовые телефонные кабели имеют, как минимум, четыре жилы. Если использовать одну из них для подвода питания, вторую в качестве общего провода, третью в качестве линии данных, то оставшуюся, четвертую, можно использовать как еще одну линию данных. Подключив эти две линии данных к разным входам упомянутого интерфейса, можно в одном кабеле организовать обмен по двум линиям, каждая из которых может нести на себе до 98 датчиков (при максимально допустимой протяженности 286м и питающем напряжении 5В), т. е. количество датчиков на данном кабеле удваивается. Пример такого включения датчиков приведен на рисунке 6.1.

Поскольку в нашем интерфейсе нет временных ограничений, подобных описанным в главе 5 для универсального асинхронного приемо-передатчика, мы рассчитали максимальную емкость кабеля, исходя из того, что шина должна подтягиваться до уровня логической единицы за 15 мкс, а не за 13.02 мкс, как это было принято при учете параметров УАПП COM-порта. Результаты такого расчета приведены в табл. 6.1. Видно, что по сравнению с данными таблицы 5.2 допустимая емкость кабеля, нагружающего линию, увеличилась примерно на 15%. Увеличилась также и максимальная длина линии (табл. 6.2).

Таким образом, используя 32-битовый интерфейс и цифровые датчики температуры фирмы Dallas Semiconductor, можно построить гибкую сеть температурного мониторинга очень простой топологии: центральный процессор и несколько (до 32) линий связи с подключенными к ним интеллектуальными датчиками температуры. Каждая из линий независима от остальных, поэтому в рамках одной сети можно создавать линии разной длины и с разным содержанием дат-

чиков. Авторы провели испытания сети с такой топологией и протяженностью линий до 50м. В качестве линий связи использовался обычный телефонный четырехжильный неэкранированный кабель (не витая пара) с погонной емкостью 44пФ/м. Сеть показала надежную работу, в частности, в условиях производственного здания с высоким уровнем промышленных помех.

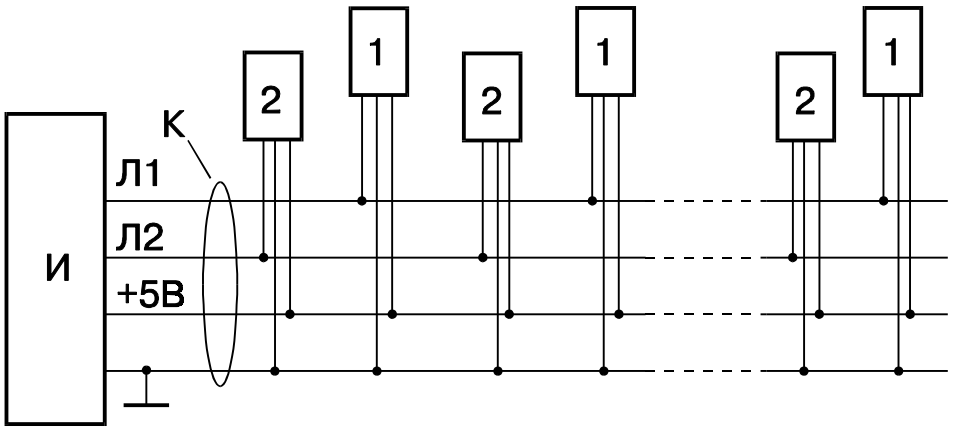


Рис.6.1. Использование свободных проводников для увеличения числа датчиков, подключенных к кабелю (И - 32-входовой интерфейс, используются только два входа - Л1 и Л2; К - четырехжильный кабель; 1 - датчики, подключенные к линии Л1; 2 - датчики, подключенные к линии Л2).

Таблица 6.1. Максимальная емкость кабеля (нФ) при максимальном числе подключенных датчиков (см. табл. 5.1).

Номинал подтягивающего резистора, кОм	Рабочее напряжение, В		
	4.0	***5.0***	***6.0***
1,5	10,94	14,33	17,61
1,8	9,12	11,96	14,65
2,2	7,47	9,76	12,00
2,7	6,06	7,97	9,78
3,3	4,98	6,53	7,99
3,9	4,19	5,50	6,76
4,7	3,49	4,58	5,63

Таблица 6.2. Максимальная длина кабеля (м) при его емкости 50пФ/м

Номинал подтягивающего резистора, кОм	Рабочее напряжение, В		
	4.0	***5.0***	***6.0***
1,5	218	286	352
1,8	182	239	293
2,2	149	195	240
2,7	121	159	195
3,3	99	130	159
3,9	83	110	135
4,7	69	91	112

6.3.2. 32-битовый интерфейс

Интерфейс, обеспечивающий связь компьютера с внешним оборудованием, должен удовлетворять определенным требованиям. Во-первых, он должен обладать достаточным быстродействием, чтобы успевать отрабатывать в течение временных интервалов циклов обмена системной шины компьютера; во-вторых, его приемники должны иметь высокоомные входы, чтобы не перегружать шину, а передатчики должны выдавать достаточный выходной ток, чтобы обеспечить работу внешних устройств; в-третьих, передатчики и приемники должны иметь отключаемый выход.

Практика показывает, что для пользователя наиболее удобно, если интерфейс оформлен в виде стандартной платы расширения, устанавливаемой в слот системной платы компьютера. Имея в виду, что системные платы всех современных IBM-совместимых компьютеров имеют в своем составе слоты стандарта ISA, мы свой интерфейс разрабатывали именно под этот стандарт.

При разработке подобных устройств всегда возникает вопрос о количестве каналов ввода/вывода, обеспечиваемых устройством. Часто количество каналов ограничивается используемым разъемом. Преследуя цель простоты конструкции и доступности комплектующих, мы остановились на использовании в качестве входного/выходного разъема стандартного 34-контактного IDC разъема, применяемого для связи флоп-

пи-дисководов с контроллерами. Этот выбор ограничил число байтовых каналов ввода/вывода четырьмя. Если использовать каждый бит в качестве независимого канала с последовательным протоколом обмена, то число каналов возрастет до 32.

Принципиальная схема интерфейса приведена на рис. 6.2. Он представляет собой четыре независимых двунаправленных восьмибитовых канала ввода/вывода, которые могут быть использованы и как тридцать два однобитовых канала для передачи и приема информации в последовательном коде. На микросхеме МС1 собран буферный усилитель сигналов системной шины, выходы которого подключены ко входам четырех буферных передатчиков (МС2 - МС5), а четырьмя младшими разрядами - также ко входам триггера-формирователя управляющего слова (МС12). Выходы передатчиков через резисторы номиналом 100 Ом соединены с выходным разъемом универсального параллельного интерфейса (УПИ). К разъему же подключены и входы четырех приемников (МС6 - МС9). Резисторы на выходах передатчиков включены для того, чтобы избежать конфликтов при чтении информации в случае, когда на одну линию от передатчика и внешнего устройства приходят сигналы разных логических уровней. При наличии резисторов приемник всегда прочитает информацию, приходящую от внешнего стройства, а не от передатчика УПИ.

На микросхемах МС10, МС11, МС13 и МС14 собран дешифратор адреса. В адресном пространстве ввода/вывода компьютера УПИ может занимать один из четырех диапазонов адресов: 360h-364h; 368h-36Ch; 3E0h-3E4h; 3E8h-3ECh. Конфигурирование на нужный диапазон производится соответствующей установкой перемычек J1 и J2. В каждом диапазоне первые четыре адреса - это адреса четырех каналов ввода-вывода, а последний, старший адрес - адрес управляющего слова. Установка режима работы каждого из каналов (на прием или передачу) производится записью соответствующего кода по адресу управляющего слова. При включении компьютера или подаче сигнала RESET все каналы интерфейса устанавливаются в режим приема информации.

Вместо микросхем серии 1533 можно использовать аналогичные микросхемы серии 555.

Предлагаемый УПИ может быть использован не только для работы с датчиками Dallas Semiconductor, но и с любыми

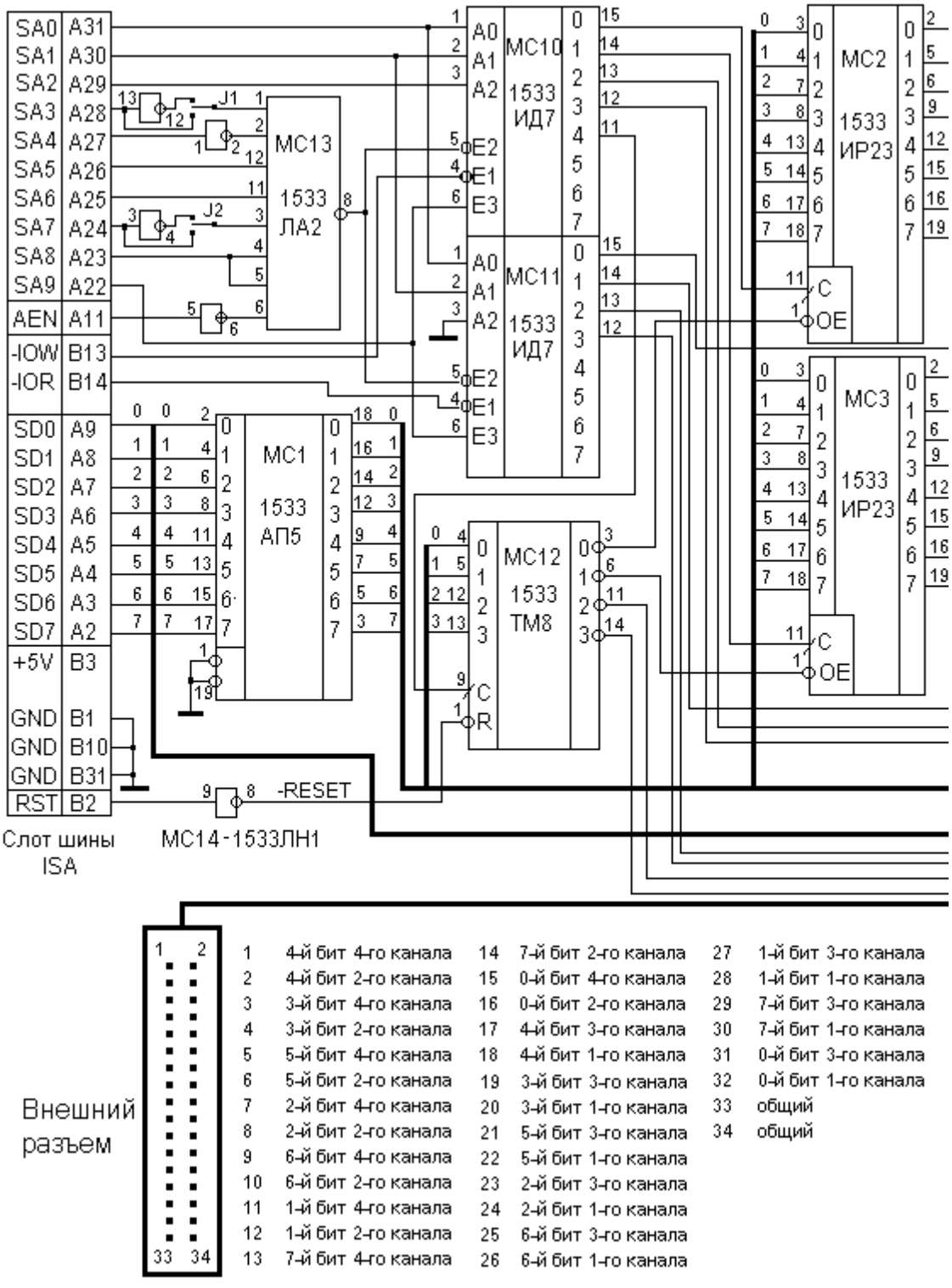
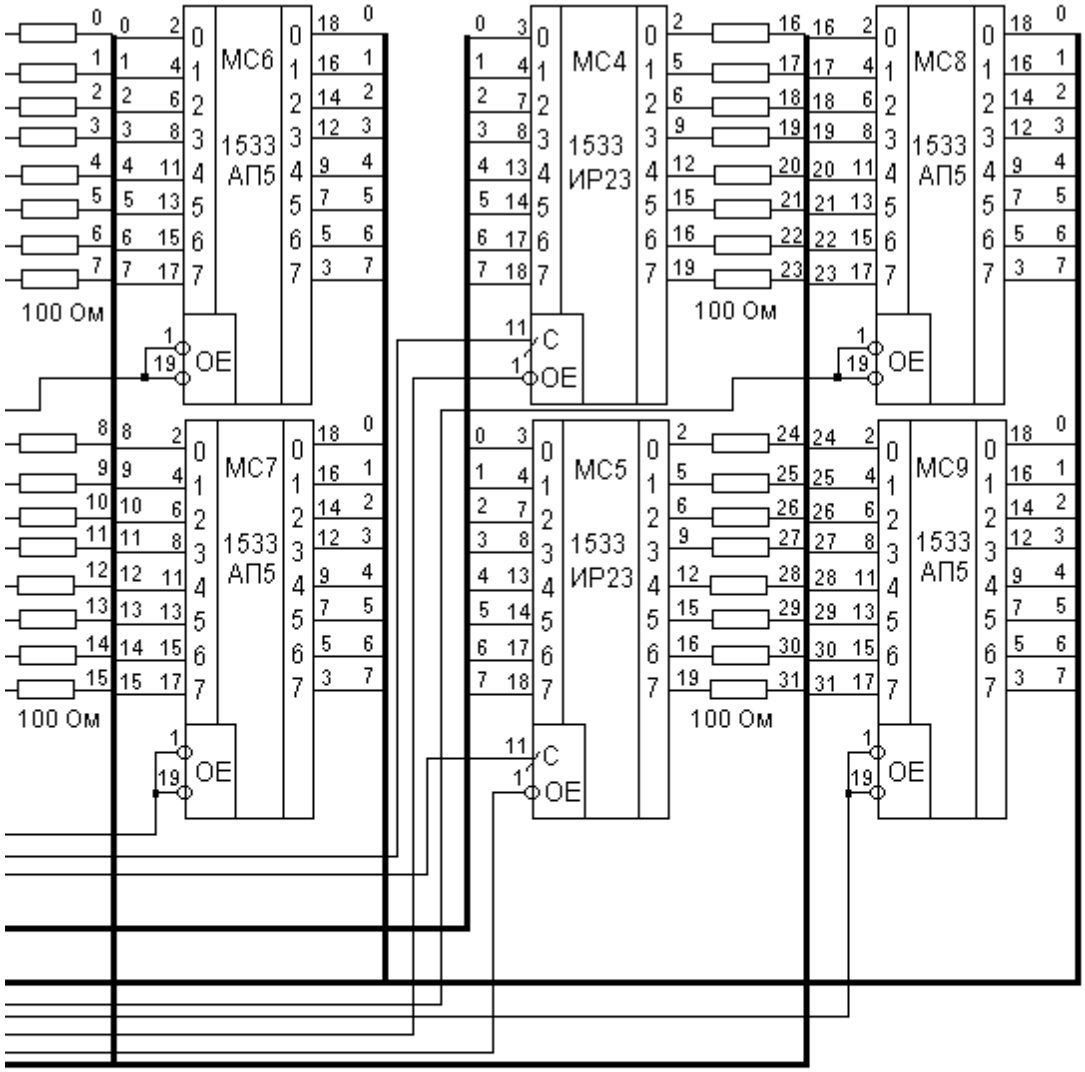


Рис. 6-2. Электрическая принципиальная схема



32-входового универсального интерфейса.

датчиками, имеющими цифровой (кодовый) выход стандарта ТТЛ, а также другими устройствами.

6.3.3. Примеры систем температурного мониторинга с простейшей топологией MicroLAN

Для организации систем температурного мониторинга с описанной выше простейшей топологией необходимы всего четыре вещи: персональный компьютер (или программируемый микроконтроллер), выполняющий роль мастера шины; описанный в предыдущем разделе аппаратный интерфейс (который выполнен в виде платы расширения и вставлен в слот материнской платы компьютера), кабель (или несколько кабелей) с подсоединенными к нему датчиками температуры типа DS18XX и программное обеспечение. При этом "на поверхности" оказываются только два компонента: компьютер и кабели. Согласитесь, что такая конфигурация очень удобна в эксплуатации, здесь не запутаешься в хитросплетении проводов и ничего случайно не зацепишь. Датчики по длине кабеля могут быть размещены произвольно, в зависимости от конкретных целей системы. Важно лишь, чтобы их количество на одном кабеле не превышало максимально допустимого.

Описанная сеть может найти применение в различных приложениях. В частности, на рис. 6.3а показана одна из возможных схем размещения датчиков температуры на наружной и внутренней поверхностях ограждающей конструкции при испытании ее теплоизоляционных свойств. Датчики подключены к двум линиям. Одна линия содержит датчики внешней поверхности, а другая - внутренней. С помощью такой схемы можно, например, измерять коэффициент теплотехнической неоднородности, что является характеристикой качества материала ограждающей конструкции. При расчете мощности отопительной системы и расположения отопительных приборов, с точки зрения обеспечения комфортных условий для человека, полезной может оказаться схема температурного мониторинга, с помощью которой можно собрать исходные данные для такого расчета и контролировать пространственное распределение температуры в течение длительного периода времени. Такая система мониторинга показана на рис. 6.3б. Здесь датчики расположены в контрольных точках на внеш-

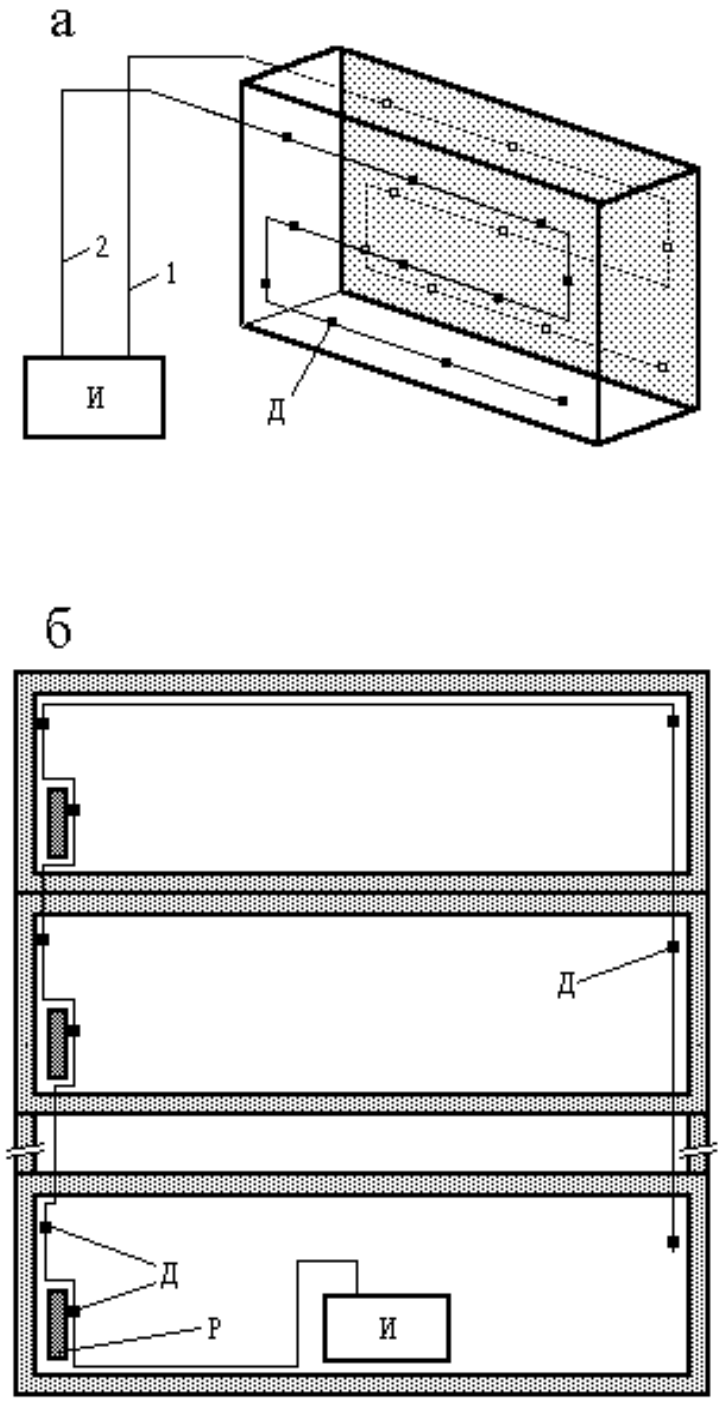


Рис.6.3. Примеры применения микролокальной сети температурного мониторинга: а)- для испытания ограждающих конструкций (И - интерфейс; Д - датчик; 1 - первая линия; 2 - вторая линия). б)- для контроля температурного режима помещений (И - интерфейс; Д - датчик; Р - радиатор отопления).

ней и внутренней стенах и на источнике тепла, что позволяет снимать температурную карту помещения при различных режимах его эксплуатации, при различных температурах наружного воздуха и т.п. Подобные мониторинговые сети могут с успехом применяться, например, для замеров температуры наружных поверхностей котла, исследования свойств термоизоляции трубопроводов и т.п.

6.3.4. Пример использования 32-битового интерфейса для организации простейшей MicroLAN

Для примера построим сеть, изображенную на рис. 6.4. Эта сеть использует четырехпроводный телефонный кабель для организации двух линий связи с активным питанием датчиков. На каждой из линий для простоты расположено по два датчика. Линии связи подключены к двум младшим битам первого восьмибитового канала интерфейса. Питание для сети берется от компьютера.

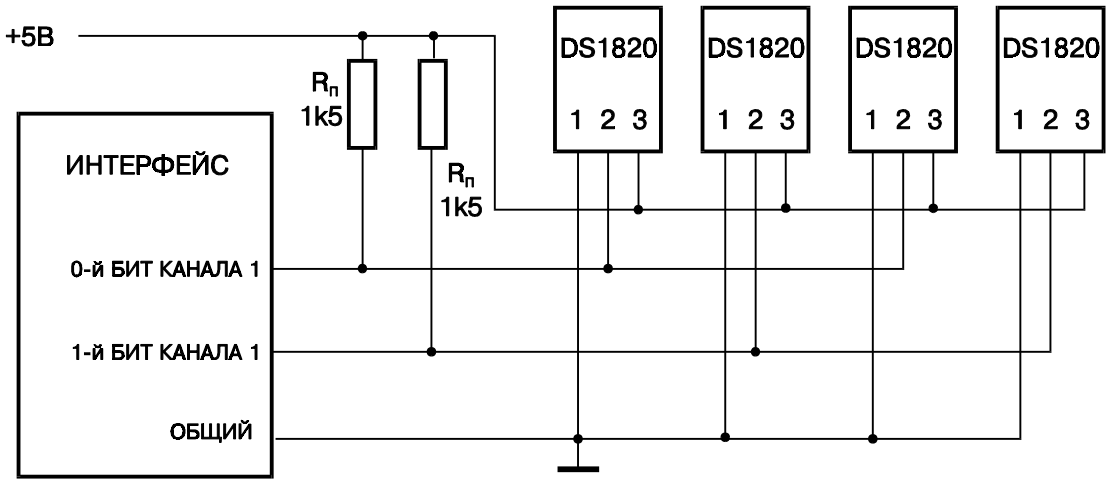


Рис. 6.4. Использование 32-битового интерфейса для управления MicroLAN с двумя линиями связи.

Как уже упоминалось выше, в пространстве ввода/вывода компьютера интерфейс может занимать один из четырех возможных диапазонов адресов. Каждый диапазон состоит из пяти адресов: по одному на каждый восьмибитовый канал связи и один - адрес командного слова, или регистра статуса, определяющего режим работы (прием или передача) для каждого канала. Адрес регистра статуса - последний (самый

старший) в ряду адресов диапазона. Если по этому адресу записан байт, четыре младших бита которого содержат нули, то все каналы интерфейса настроены на прием. Если эти биты содержат единицы, то все каналы настроены на передачу. Иначе говоря, каждый из первых четырех битов регистра статуса ответственен за режим подведомственного ему канала: 0-й бит - за режим первого канала, 1-й бит - за режим второго канала и т.д. Если управляющий бит содержит "1", то соответствующий канал настроен на передачу, если "0", то - на прием.

Предположим, что наш интерфейс настроен перемычками на диапазон адресов 360h - 364h. Тогда первый канал будет иметь адрес 360h, а адрес 364h будет принадлежать регистру статуса. Две коммуникационные линии сети подключены, как уже говорилось, к двум младшим битам (нулевому и первому) первого канала интерфейса. Эти два бита выступают как мастера шины для подключенных к ним однопроводных линий связи. Тогда, чтобы перевести обе линии в исходное высокоуровневое состояние, достаточно биты 0 и 1 первого канала перевести в режим приема. Тогда их приемо-передатчики перейдут в высокоимпедансное состояние, а подтягивающие резисторы $R_п$ (рис. 6.4) подтянут соответствующие шины в высокоуровневое состояние. Так как интерфейс допускает изменение режима приема/передачи только побайтово (поканально), то, записывая в регистр статуса (по адресу 364h), например, "0", мы заведомо переведем в режим приема все четыре канала. Но поскольку все каналы, кроме первого, нас в данном случае не интересуют, то в дальнейшем для перевода мастеров в режим приема будем для простоты записывать в регистр статуса именно 0. Аналогично, для перевода первого канала в режим передачи будем посылать в регистр статуса число 1. При этом только первый канал переходит в режим передачи, остальные три остаются в режиме приема. Переведя нужный канал в режим передачи, можно теперь послать по адресу этого канала (в нашем случае 360h для первого канала) соответствующий код, переводящий биты этого канала в высокоуровневое либо низкоуровневое состояние. Так, если по адресу канала послать число 0, то все биты окажутся в низкоуровневом состоянии. Если же послать число 1, то нулевой бит перейдет в высокоуровневое состояние. Если послать число 255, то все восемь битов первого канала пе-

рейдут в высокоуровневое состояние.

Таким образом, чтобы наши два мастера - биты 0 и 1 первого канала - отпустили свои шины, достаточно в регистр статуса по адресу 364h записать число 0, переводящее эти биты в высокоимпедансное состояние приема. Для того, чтобы соответствующий мастер перевел свою шину в низкоуровневое состояние, нужно записать соответствующий код по адресу 360h и послать в регистр статуса по адресу 364h число 1, переводящее канал 1 в режим передачи. Сразу после этого биты 0 и 1 первого канала перейдут в состояния, определяемые записанным в регистр первого канала кодом. Например, если было записано число 1, то бит 0 окажется в высокоуровневом состоянии, никак не влияя на состояние своей шины, а бит 1 перейдет в низкоуровневое состояние, опуская свою шину в нуль.

В нашей сети будут функционировать четыре датчика температуры. На вид они неразличимы. Их идентификационные коды спрятаны внутри и прочитать их можно лишь программным путем. Можно, конечно, воспользоваться командой "Исследование ПЗУ" для идентификации всех датчиков сети. Но при этом остается неясным, какому именно физическому датчику соответствует данный код. Мы поступим другим способом. Перед установкой датчиков на кабель сети протестируем каждый из них командой "Чтение ПЗУ". Для этого, поочередно подключая датчики к одному из мастеров, прочитаем его ПЗУ с помощью этой команды.

Ниже представлен листинг программы на бейсике, которая выполняет единственную функцию: читает ПЗУ подключенного к шине датчика. В этой программе предполагается, что датчик подключается к биту 0 первого канала интерфейса и что используется внешнее питание датчика.

Листинг 1. Программа чтения ПЗУ отдельного датчика

```
'Объявляем процедуры
DECLARE SUB ReadROM (Address%, Bit%)
DECLARE FUNCTION SearchForPresence% (Address%, Bit%)
DECLARE SUB Delay ()

'Объявляем переменные
DIM SHARED FirstAddress AS INTEGER
DIM SHARED SecondAddress AS INTEGER
```

```
DIM SHARED ThirdAddress AS INTEGER
DIM SHARED FourthAddress AS INTEGER
DIM SHARED StatusAddress AS INTEGER
DIM SHARED ReadROMArray(7)  AS INTEGER 'Массив, содержащий биты
                                        'команды 33h

'Определяем численные значения констант и переменных
FirstAddress = &H360
SecondAddress = FirstAddress + 1
ThirdAddress = FirstAddress + 2
FourthAddress = FirstAddress + 3
StatusAddress = FirstAddress + 4

'Заполняем массив команды "Чтение ПЗУ"
A% = &H33
FOR I% = 7 TO 0 STEP -1
      ReadROMArray(I%) = A% \ 2 ^ I%
      A% = A% MOD 2 ^ I%
NEXT I%

'Запускаем цикл чтения ПЗУ
T! = TIMER
DO UNTIL TIMER - T! > 1 'Устанавливаем время тайм-аута равным 1 сек
      A% = SearchForPresence(FirstAddress, 0) 'Проверяем наличие
                                              'датчика
      SELECT CASE A%
         CASE 0 'Датчик подключен
            CALL ReadROM(FirstAddress, 0) 'Вызываем процедуру чтения
                                          'ПЗУ
         CASE 1 'Датчик не подключен
            PRINT "На линии нет датчиков температуры"
      END SELECT
      IF INKEY$ = CHR$(27) THEN EXIT DO 'Обеспечиваем выход по
                                        'клавише "Esc"
LOOP
END

SUB Delay 'Подпрограмма, формирующая длительность временного
          'слота
      FOR I% = 1 TO 40: NEXT I%
END SUB

SUB ReadROM (Address%, Bit%) 'Подпрограмма чтения ПЗУ DS1820
DO
      'Очищаем строковую переменную для содержимого ПЗУ
      S$ = ""

      'Проводим инициализацию
      DO
```

```
    A% = SearchForPresence(Address%, Bit%)
    SELECT CASE A%
       CASE 0
          EXIT DO
       CASE ELSE
          EXIT SUB
    END SELECT
LOOP

'Выдаем команду "Чтение ПЗУ" (33h)
FOR I% = 0 TO 7
    SELECT CASE ReadROMArray(I%)
       CASE 0                    'Записываем "0"
          OUT Address%, 0
          OUT StatusAddress, 1
          Delay
          OUT StatusAddress, 0
       CASE 1             'Записываем "1"
          OUT Address%, 0
          OUT StatusAddress, 1
          OUT StatusAddress, 0
          Delay
    END SELECT
NEXT I%

'Посылаем 64 временных слота чтения и прочитанные биты
'складываем в
'строковую переменную S$
FOR U% = 0 TO 63
    OUT Address%, 0
    OUT StatusAddress, 1
    OUT StatusAddress, 0
    A% = (INP(Address%) AND 2 ^ Bit%) / 2 ^ Bit%
    S$ = S$ + RTRIM$(LTRIM$(STR$(A%)))
NEXT U%

'Вычисляем ЦИК
'Определяем 8-битовый сдвиговый массив (регистр)
DIM Register(7) AS INTEGER
'Обнуляем этот массив
FOR I% = 0 TO 7: Register(I%) = 0: NEXT I%
'Начиная с младшего разряда, побитово обрабатываем данные с
'помощью функции
'XOR и сдвигового регистра. Проходим все биты, кроме старших
'восьми. В результате в регистре (массиве) окажется ЦИК.
N% = LEN(S$) - 8
FOR I% = 1 TO N%
    C% = VAL(MID$(S$, I%, 1))
    A% = Register(0) XOR C%
```

```
            B% = Register(3) XOR A%
            C% = Register(4) XOR A%
            Register(0) = Register(1)
            Register(1) = Register(2)
            Register(2) = B%
            Register(3) = C%
            Register(4) = Register(5)
            Register(5) = Register(6)
            Register(6) = Register(7)
            Register(7) = A%
        NEXT I%
        'Преобразуем числовое значение регистра в строковое
        G$ = ""
        FOR I% = 0 TO 7
            G$ = G$ + RTRIM$(LTRIM$(STR$(Register(I%))))
        NEXT I%

        'Сравниваем полученную строку со старшими 8-ю битами строки
        'данных. Если эти строки совпадают, то данные приняты верно
        IF G$ = RIGHT$(S$, 8) THEN
            EXIT DO
        ELSE
            EXIT SUB
        END IF
LOOP

'Выводим содержимое ПЗУ на экран ( для удобства - сначала одной
'строкой, а затем в столбик побайтово)
CLS
LOCATE 1, 1: PRINT S$;
LOCATE 3, 1
FOR I% = 0 TO 7
        PRINT MID$(S$, 1 + I% * 8, 8)
NEXT I%
SLEEP
END SUB


FUNCTION SearchForPresence% (Address%, Bit%)
'Функция проверки присутствия на линии хотя бы одного датчика
'температуры

'Ждем пока шина перейдет в высокое состояние
OUT StatusAddress, 0
DO
  A% = INP(Address%) AND 2 ^ Bit%
LOOP WHILE A% = 0
'Сбрасываем нужный бит
OUT Address%, 255 - 2 ^ Bit%
```

```
OUT StatusAddress, 1
'Задержка на длину импульса сброса
FOR I% = 1 TO 10: Delay: NEXT I%
'Освобождаем шину
OUT StatusAddress, 0
'Задержка на восстановление шины
Delay
'Читаем импульс присутствия
SearchForPresence = INP(Address%) AND 2 ^ Bit%
'Ждем пока шина перейдет в высокое состояние
OUT StatusAddress, 0
DO
  A% = INP(Address%) AND 2 ^ Bit%
LOOP WHILE A% = 0
END FUNCTION
```

По поводу этого листинга следует сделать пару замечаний. Во-первых, чтобы не усложнять картину, мы все временные задержки оформили в виде пустых циклов. Это самый простой способ создания в программе временных задержек, но не самый хороший. При использовании пустых циклов длительность задержки целиком зависит от аппаратного обеспечения компьютера. На разных компьютерах вышеприведенная программа будет давать разные задержки. Может сложиться ситуация, когда она вообще не будет работать. Для ее правильного функционирования число пустых циклов для задержек нужно выбирать, исходя из быстродействия конкретного компьютера. На практике это делается путем подбора. Можно, например, ввести в программу процедуру, которая, опираясь на внутренний таймер, определит, сколько времени займет, например, выполнение 1000000 пустых циклов. Разделив затем это число циклов на число микросекунд прошедшего временного интервала, можно получить число пустых циклов, приходящихся на 1 микросекунду. Этим числом можно потом руководствоваться, вводя программные задержки, требуемые однопроводным протоколом обмена. Однако, и здесь никакой гарантии точности отсчета временных интервалов нет. Дело в том, что, в зависимости от конкретных задач, решаемых в данный момент процессором, число пустых циклов в единицу времени будет различным. Тем не менее, наши исследования показали, что такая организация временных интервалов для целей однопроводного протокола дает надежные результаты. Для более грамотного определения временных задержек можно, например, пойти по пути

перепрограммирования микросхемы таймера. Второе замечание по поводу вышеприведенного листинга заключается в том, что в нем не отражены процедуры сохранения полученного значения ПЗУ в файле в виде, удобном для дальнейшего использования. Программа также не оптимизирована по коду и времени работы. Она служит исключительно для иллюстрации процесса чтения ПЗУ.

Наконец, приведем листинг программы, обеспечивающей опрос четырех датчиков нашей сети (рис. 6.4), вывод значений температуры на экран монитора, а также вывод каждые четверть часа значений температуры в файл с привязкой к реальным времени и дате. Иными словами, эта программа позволяет осуществить постоянный мониторинг температуры четырех точек объекта и зафиксировать эти температуры в файле.

Листинг 2. Программа опроса четырех датчиков температуры

```
'Объявляем процедуры
DECLARE SUB ConvertAndReadTemperature (Address%, A$, N%, M%, T$)
DECLARE FUNCTION SearchForPresence% (Address%, Bit%)
DECLARE SUB FillCommandArrays ()
DECLARE SUB Delay ()

'Объявляем переменные
DIM SHARED FirstAddress AS INTEGER
DIM SHARED SecondAddress AS INTEGER
DIM SHARED ThirdAddress AS INTEGER
DIM SHARED FourthAddress AS INTEGER
DIM SHARED StatusAddress AS INTEGER
DIM SHARED ConvertTemperatureArray(7)  AS INTEGER 'Массив,
'содержащий биты команды 44h
DIM SHARED MatchROMArray(7)  AS INTEGER 'Массив, содержащий
биты 'команды 55h
DIM SHARED ReadScratchpadArray(7)  AS INTEGER 'Массив,
содержащий 'биты команды BEh
DIM SHARED UniqueCodesArray(3) AS STRING * 64 'Массив, содержащий
'коды ПЗУ подключенных датчиков
DIM SHARED Temperature(3) AS SINGLE 'Массив для хранения значений
'температур
DIM SHARED ErrorFlag AS INTEGER 'Флаг факта ошибочного чтения

'Подключаем файлы, содержащие объявления функций
'форматирования времени
'$INCLUDE: 'datim.bi'
'$INCLUDE: 'format.bi'
```

```
'Определяем численные значения констант и переменных
CONST TRUE = -1
CONST FALSE = 0
FirstAddress = &H360
SecondAddress = FirstAddress + 1
ThirdAddress = FirstAddress + 2
FourthAddress = FirstAddress + 3
StatusAddress = FirstAddress + 4

'Вызываем подпрограмму заполнения массивов команд
FillCommandArrays

'Заполняем содержимым ПЗУ подключенных датчиков соответствующий
'массив
UniqueCodesArray(0) =
"0000100010001101101000010011100000000000000000000000000010000011"
UniqueCodesArray(1) =
"0000100010011111111011010100111000000000000000000000000010000100"
UniqueCodesArray(2) =
"0000100000110110000000101001110000000000000000000000000011101110"
UniqueCodesArray(3) =
"0000100011001000100110001001110000000000000000000000000010110101"

'Рабочий цикл опроса датчиков и фиксации их показаний
CLS
LOCATE 1, 1: PRINT "Температура первого датчика    = ";
LOCATE 2, 1: PRINT "Температура второго датчика    = ";
LOCATE 3, 1: PRINT "Температура третьего датчика   = ";
LOCATE 4, 1: PRINT "Температура четвертого датчика = ";
DO
      ErrorFlag = 0
      T$ = TIME$
      FOR U% = 0 TO 3
         SELECT CASE U%
            CASE 0, 2
               CALL ConvertAndReadTemperature(FirstAddress,
UniqueCodesArray(U%), U%, 0, T$)
            CASE 1, 3
               CALL ConvertAndReadTemperature(FirstAddress,
UniqueCodesArray(U%), U%, 1, T$)
         END SELECT
      NEXT U%
      IF ErrorFlag = 0 THEN
         LOCATE 1, 34: PRINT USING "+###.#"; Temperature(0);
```

```
      LOCATE 2, 34: PRINT USING "+###.#"; Temperature(1);
      LOCATE 3, 34: PRINT USING "+###.#"; Temperature(2);
      LOCATE 4, 34: PRINT USING "+###.#"; Temperature(3);
   END IF
   SELECT CASE MID$(T$, 4, 5) 'Каждые четверть часа пишем
                              'показания в файл
      CASE "00:00", "15:00", "30:00", "45:00"
         SELECT CASE ErrorFlag
            CASE 0
               A# = Now#
               D$ = FormatD$(A#, "dd-mm-yyyy")
               T$ = FormatD$(A#, "hh:mm:ss")
               X$ = D$ + "_" + T$
               FileNumber = FREEFILE
               FileName$ = MID$(D$, 7, 4) + MID$(D$, 4, 2) + MID$(D$, 1,
2) + ".dat"
               OPEN FileName$ FOR APPEND AS FileNumber
               PRINT #FileNumber, TAB(0); A#; TAB(19); X$;
               FOR I% = 0 TO 3
                  PRINT #FileNumber, TAB(39 + I% * 7); USING "+###.#";
Temperature(I%);
               NEXT I%
               CLOSE FileNumber
               BEEP
               DO UNTIL MID$(TIME$, 7, 2) <> "00": LOOP
            CASE ELSE
         END SELECT
      CASE ELSE
   END SELECT
   IF INKEY$ = CHR$(27) THEN EXIT DO
LOOP
END


SUB ConvertAndReadTemperature (Address%, ROMCode$, Unit%, Bit%,
T$)
'Подпрограмма выдачи заданий датчикам на преобразование
'температуры и съем информации с них

Temperature(Unit%) = 0
'Проводим инициализацию
DO
   A% = SearchForPresence(Address%, Bit%)
   SELECT CASE A%
      CASE 0
         EXIT DO
      CASE ELSE
         EXIT SUB
   END SELECT
```

```
LOOP

'Выдаем команду "Выбор ПЗУ" (55h)
FOR I% = 0 TO 7
      SELECT CASE MatchROMArray(I%)
         CASE 0                  'Записываем "0"
            OUT Address%, 0
            OUT StatusAddress, 1
            Delay
            OUT StatusAddress, 0
         CASE 1              'Записываем "1"
            OUT Address%, 0
            OUT StatusAddress, 1
            OUT StatusAddress, 0
            Delay
      END SELECT
NEXT I%

'Выдаем 64-битовый код нужного датчика
FOR I% = 1 TO 64
      SELECT CASE VAL(MID$(ROMCode$, I%, 1))
         CASE 0                  'Записываем "0"
            OUT Address%, 0
            OUT StatusAddress, 1
            Delay
            OUT StatusAddress, 0
         CASE 1              'Записываем "1"
            OUT Address%, 0
            OUT StatusAddress, 1
            OUT StatusAddress, 0
            Delay
      END SELECT
NEXT I%

'Записываем в DS1820 команду 44h ("Преобразование температуры")
FOR I% = 0 TO 7
      SELECT CASE ConvertTemperatureArray(I%)
         CASE 0                  'Записываем "0"
            OUT Address%, 0
            OUT StatusAddress, 1
            Delay
            OUT StatusAddress, 0
         CASE 1              'Записываем "1"
            OUT Address%, 0
            OUT StatusAddress, 1
            OUT StatusAddress, 0
            Delay
      END SELECT
NEXT I%
```

```
'Ждем 500 мс пока происходит преобразование
SELECT CASE T$
       CASE "24:00:00"
          T! = 0: DO UNTIL TIMER - T! > .5: LOOP
       CASE ELSE
          T! = TIMER: DO UNTIL TIMER - T! > .5: LOOP
END SELECT

'Проводим инициализацию
DO
       A% = SearchForPresence(Address%, Bit%)
       SELECT CASE A%
          CASE 0
             EXIT DO
          CASE ELSE
             EXIT SUB
       END SELECT
LOOP

'Выдаем команду "Выбор ПЗУ" (55h)
FOR I% = 0 TO 7
       SELECT CASE MatchROMArray(I%)
          CASE 0                    'Записываем "0"
             OUT Address%, 0
             OUT StatusAddress, 1
             Delay
             OUT StatusAddress, 0
          CASE 1                'Записываем "1"
             OUT Address%, 0
             OUT StatusAddress, 1
             OUT StatusAddress, 0
             Delay
       END SELECT
NEXT I%

'Выдаем 64-битовый код нужного датчика
FOR I% = 1 TO 64
       SELECT CASE VAL(MID$(ROMCode$, I%, 1))
          CASE 0                    'Записываем "0"
             OUT Address%, 0
             OUT StatusAddress, 1
             Delay
             OUT StatusAddress, 0
          CASE 1                'Записываем "1"
             OUT Address%, 0
             OUT StatusAddress, 1
             OUT StatusAddress, 0
             Delay
```

```
        END SELECT
NEXT I%

'Записываем команду BEh ("Чтение СОП") в DS1820
FOR I% = 0 TO 7
        SELECT CASE ReadScratchpadArray(I%)
           CASE 0                    'Записываем "0"
              OUT Address%, 0
              OUT StatusAddress, 1
              Delay
              OUT StatusAddress, 0
           CASE 1                 'Записываем "1"
              OUT Address%, 0
              OUT StatusAddress, 1
              OUT StatusAddress, 0
              Delay
        END SELECT
NEXT I%

'Читаем девять байтов полной СОП датчика в строковую переменную S$
S$ = ""
FOR U% = 0 TO 71
        OUT Address%, 0
        OUT StatusAddress, 1
        OUT StatusAddress, 0
        A% = (INP(Address%) AND 2 ^ Bit%) / 2 ^ Bit%
        S$ = S$ + RTRIM$(LTRIM$(STR$(A%)))
NEXT U%

'Вычисляем ЦИК
'Определяем 8-битовый сдвиговый массив (регистр)
DIM Register(7) AS INTEGER
'Обнуляем этот массив
FOR I% = 0 TO 7: Register(I%) = 0: NEXT I%
'Начиная с младшего разряда, побитово обрабатываем данные с
'помощью функции XOR и сдвигового регистра. Проходим все биты,
'кроме старших восьми. В результате в регистре (массиве) окажется
'ЦИК.
N% = LEN(S$) - 8
FOR I% = 1 TO N%
        C% = VAL(MID$(S$, I%, 1))
        A% = Register(0) XOR C%
        B% = Register(3) XOR A%
        C% = Register(4) XOR A%
        Register(0) = Register(1)
        Register(1) = Register(2)
        Register(2) = B%
        Register(3) = C%
        Register(4) = Register(5)
```

```
        Register(5) = Register(6)
        Register(6) = Register(7)
        Register(7) = A%
NEXT I%
'Преобразуем числовое значение регистра в строковое
G$ = ""
FOR I% = 0 TO 7
        G$ = G$ + RTRIM$(LTRIM$(STR$(Register(I%))))
NEXT I%

'Сравниваем полученную строку со старшими 8-ю битами строки
'данных. Если эти строки совпадают, то данные приняты верно
IF G$ = RIGHT$(S$, 8) THEN
        A% = TRUE
ELSE
        A% = FALSE
        ErrorFlag = 1
        EXIT SUB
END IF

'Вычисляем значения температуры
SELECT CASE A%
        CASE TRUE
           L$ = MID$(S$, 1, 8)
           H$ = MID$(S$, 9, 8)
           T% = 0
           SELECT CASE INSTR(H$, "1") 'Если в старшем байте "1", то T° < 0
              CASE 0
                 FOR I% = 8 TO 1 STEP -1
                    T% = T% + VAL(MID$(L$, I%, 1)) * 2 ^ (I% - 1)
                 NEXT I%
                 Temperature(Unit%) = T% / 2
              CASE ELSE
                 FOR I% = 8 TO 1 STEP -1
                    T% = T% + VAL(MID$(L$, I%, 1)) * 2 ^ (I% - 1)
                 NEXT I%
                 Temperature(Unit%) = (T% - 256) / 2
           END SELECT
        CASE FALSE
END SELECT
END SUB


SUB Delay  'Подпрограмма временной задержки (задает длительность
           'временного 'слота)
        FOR I% = 1 TO 40: NEXT I%
END SUB
```

```
SUB FillCommandArrays  'Подпрограмма заполнения массивов,
                       'содержащих коды команд

'Заполняем массив команды "Выбор ПЗУ (55h)"
A% = &H55
FOR I% = 7 TO 0 STEP -1
        MatchROMArray(I%) = A% \ 2 ^ I%
        A% = A% MOD 2 ^ I%
NEXT I%

'Заполняем массив команды "Преобразование температуры (44h)"
A% = &H44
FOR I% = 7 TO 0 STEP -1
        ConvertTemperatureArray(I%) = A% \ 2 ^ I%
        A% = A% MOD 2 ^ I%
NEXT I%

'Заполняем массив команды "Чтение СОП (BEh)"
A% = &HBE
FOR I% = 7 TO 0 STEP -1
        ReadScratchpadArray(I%) = A% \ 2 ^ I%
        A% = A% MOD 2 ^ I%
NEXT I%
END SUB


FUNCTION SearchForPresence% (Address%, Bit%)
'Функция проверки присутствия на линии хотя бы одного датчика
'температуры

'Ждем пока шина перейдет в высокое состояние
OUT StatusAddress, 0
DO
  A% = INP(Address%) AND 2 ^ Bit%
LOOP WHILE A% = 0
'Сбрасываем нужный бит
OUT Address%, 255 - 2 ^ Bit%
OUT StatusAddress, 1
'Задержка на длину импульса сброса
FOR I% = 1 TO 10: Delay: NEXT I%
'Освобождаем шину
OUT StatusAddress, 0
'Задержка на восстановление шины
Delay
'Читаем импульс присутствия
SearchForPresence = INP(Address%) AND 2 ^ Bit%
'Ждем пока шина перейдет в высокое состояние
OUT StatusAddress, 0
DO
```

```
  A% = INP(Address%) AND 2 ^ Bit%
LOOP WHILE A% = 0
END FUNCTION
```