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.

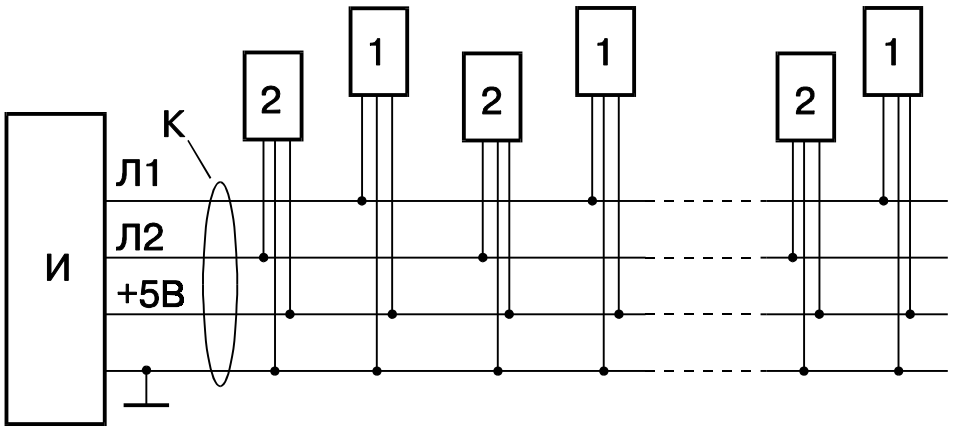


Рис.6.1. Использование свободных проводников для увеличения числа датчиков, подключенных к кабелю (И - 32-входовой интерфейс, используются только два входа - Л1 и Л2; К - четырехжильный кабель; 1 - датчики, подключенные к линии Л1; 2 - датчики, подключенные к линии Л2).

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

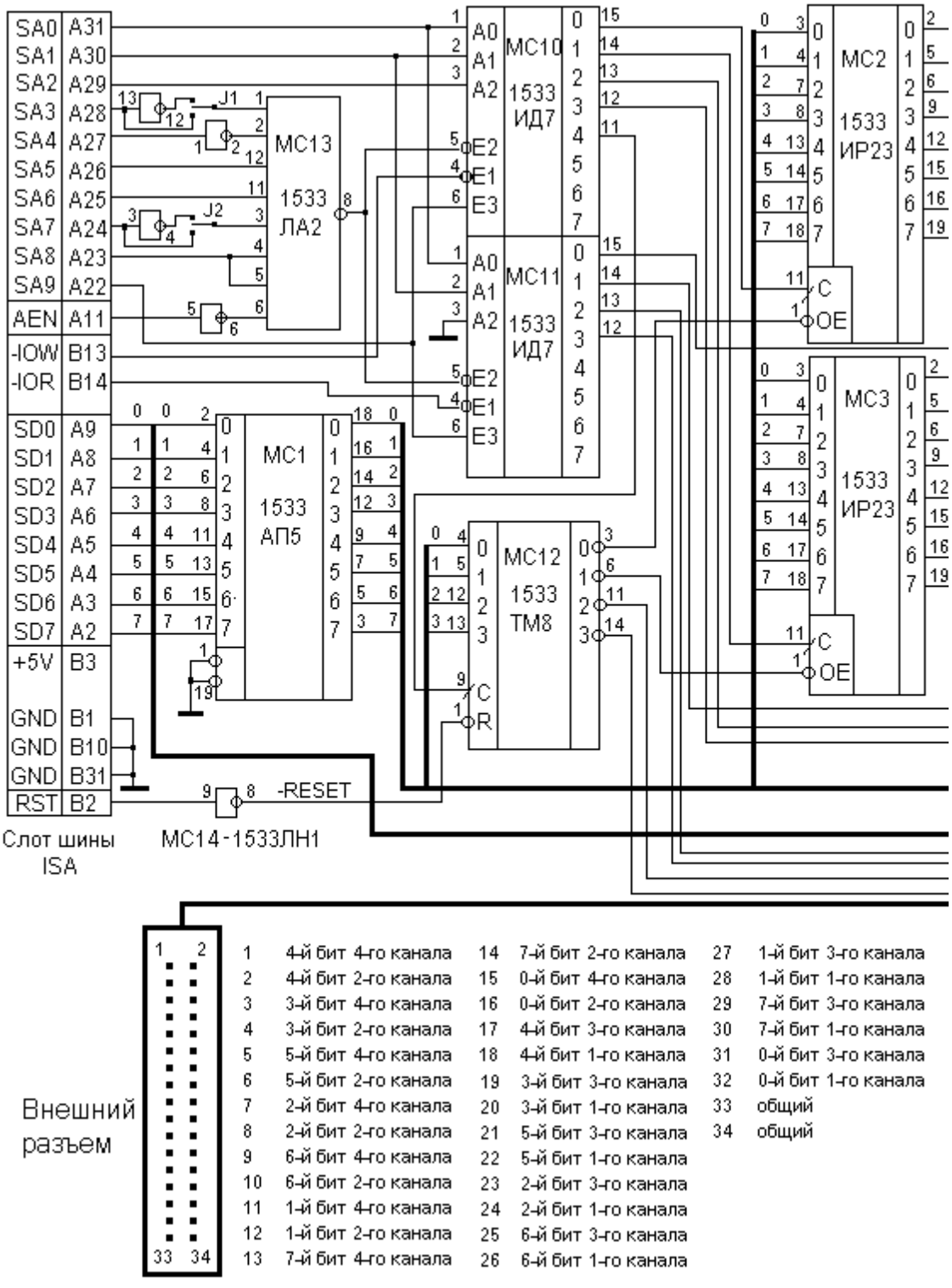
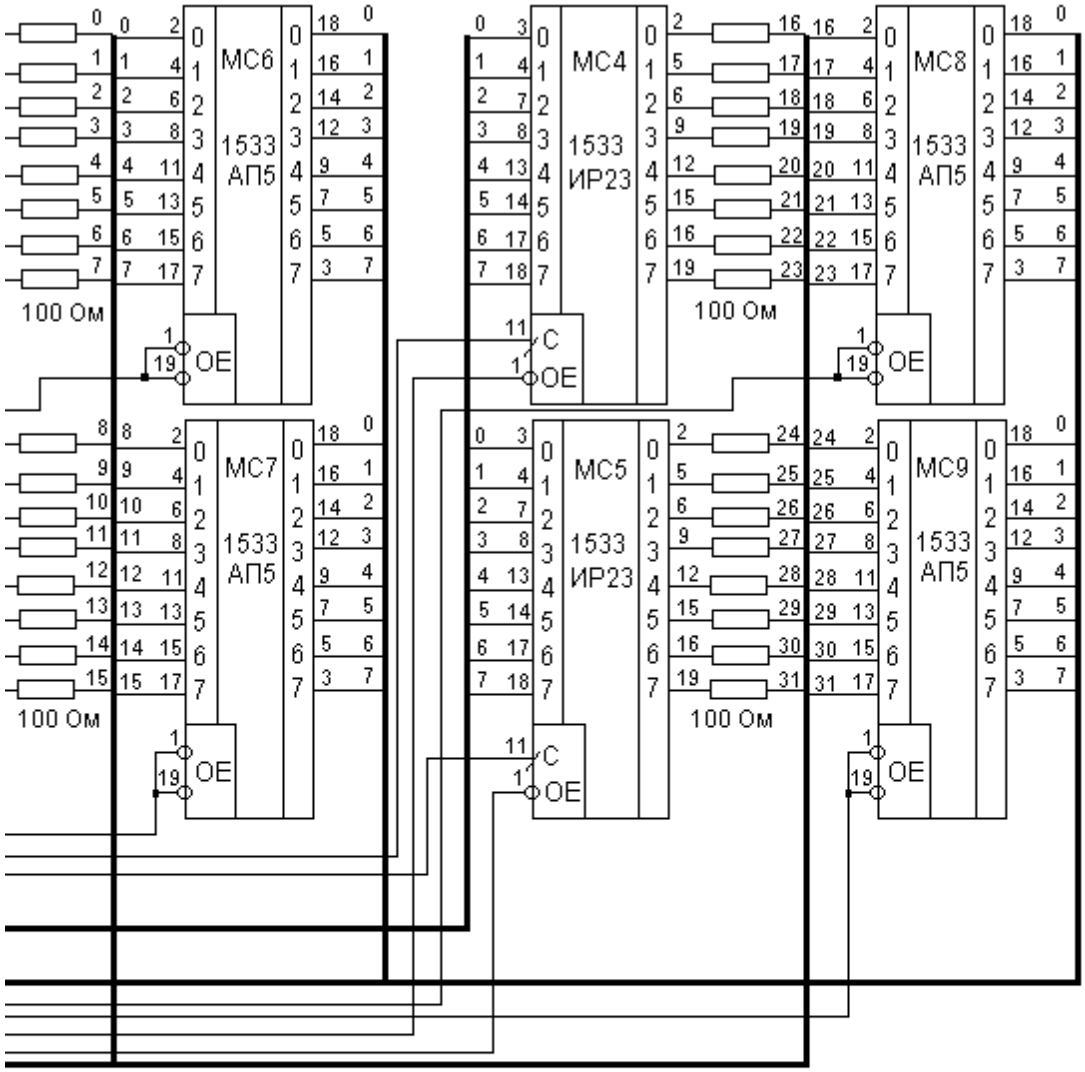


Рис. 6-2. Электрическая принципиальная схема



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-

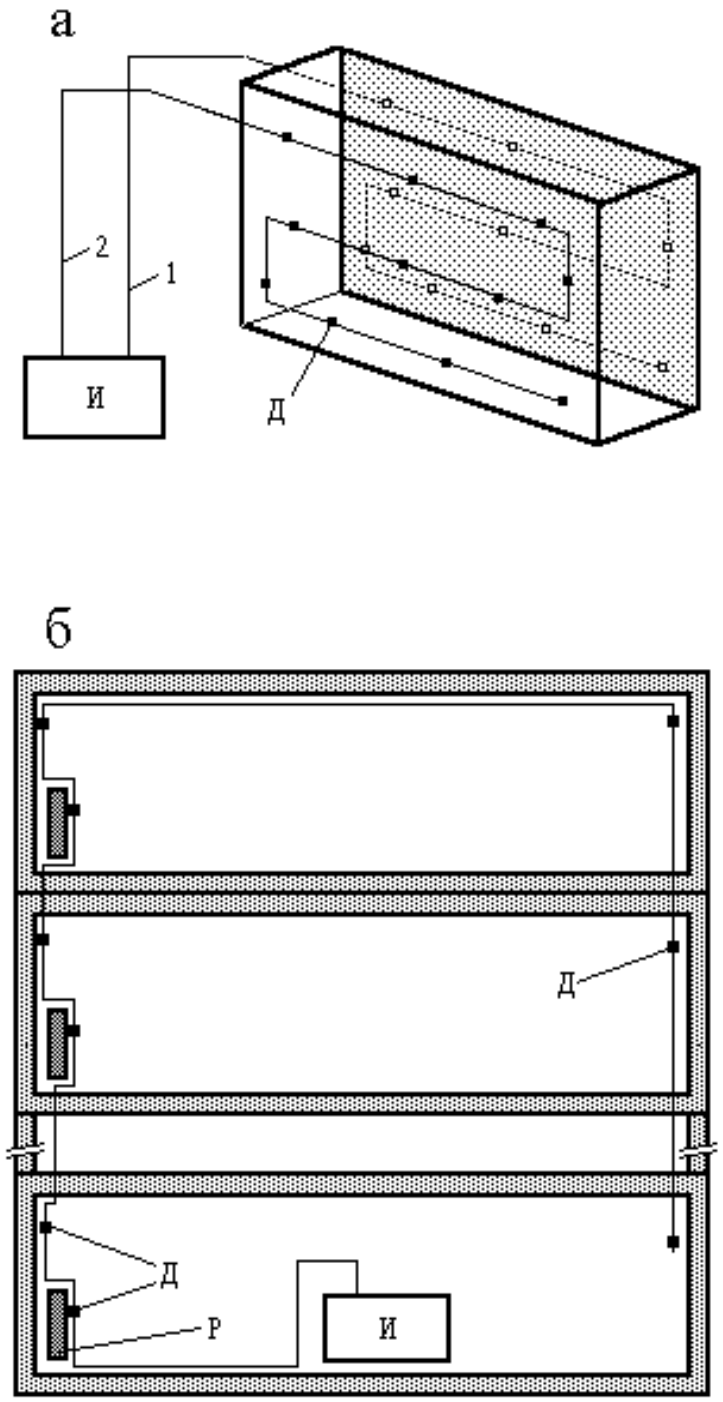


Рис.6.3. Примеры применения микролокальной сети температурного мониторинга: а)- для испытания ограждающих конструкций (И - интерфейс; Д - датчик; 1 - первая линия; 2 - вторая линия). б)- для контроля температурного режима помещений (И - интерфейс; Д - датчик; Р - радиатор отопления).

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.

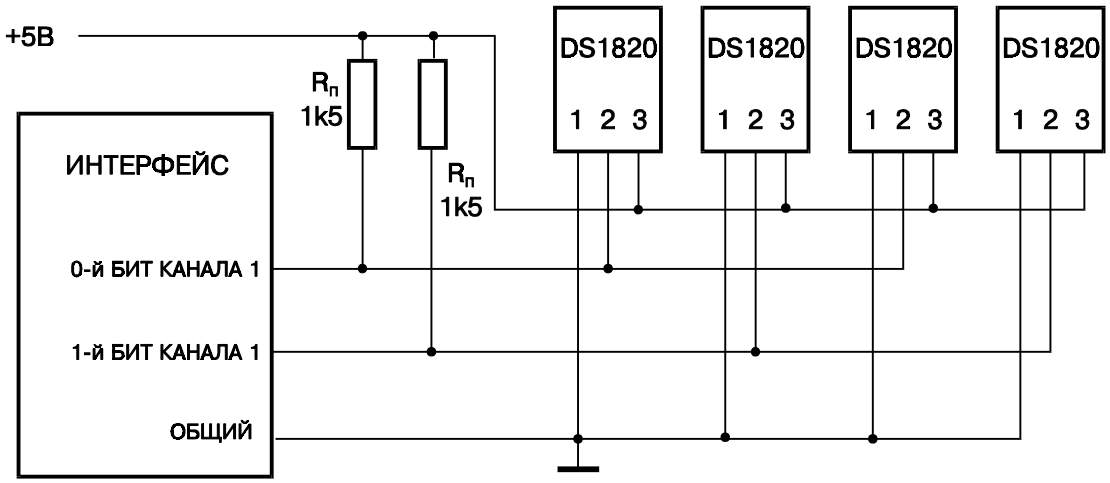


Рис. 6.4. Использование 32-битового интерфейса для управления MicroLAN с двумя линиями связи.

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
```