ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ГОУ ВПО «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ
КОМПЬЮТЕРНЫХ НАУК И ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА
ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМ
И ТЕХНОЛОГИЙ

МИШУНИН ВАДИМ ВАСИЛЬЕВИЧ
ЛИХОЛОБ ПЕТР ГЕОРГИЕВИЧ

МИКРОПРОЦЕССОРЫ И ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Белгород, 2010 г.

УДК 681.325.5
ББК 32.97
М71

Печатается по решению

редакционно-издательского совета

Белгородского государственного университета

Рецензенты:

Александр Владимирович Белоусов, доцент кафедры технической кибернетики Белгородского государственного технологического университета им. В.Г. Шухова, к.т.н., доцент;

Владимир Ервандович Хачатрян, заведующий кафедрой математического и программного обеспечения информационных систем Белгородского государственного университета, д.ф-м.н., доцент

Мишунин, В.В.

М71 Микропроцессоры и цифровая обработка сигналов: Учебно-методическое пособие / В.В. Мишунин, П.Г. Лихолоб. – Белгород: Изд-во БелГУ, 2010. – 210 с.

Учебное пособие по курсу «Микропроцессоры и цифровая обработка сигналов» состоит из лекционного материала и лабораторного практикума. Учебно-методическое пособие составлено в соответствии с требованиями по обязательному минимуму содержания и уровня подготовки специалиста с высшим образованием, предъявляемым Государственным образовательным стандартом высшего профессионального образования по специальности 210405 (201100) «Радиосвязь, радиовещание и телевидение», а также в соответствии с учебной программой.

УДК 681.325.5
ББК 32.97

© Авторы, 2010
© Белгородский
государственный университет, 2010

ОГЛАВЛЕНИЕ

Введение 6
Тема 1. Общие вопросы и основные понятия о микропроцессорных системах 8
1.1. Основные понятия о микропроцессорах и микропроцессорных системах 8
1.1.1. Понятие «микропроцессор» 11
1.1.2. Шинная структура связей 16
1.1.3. Режимы работы микропроцессорной системы 21
1.1.4. Архитектура микропроцессорных систем 25
1.2. Классификация микропроцессоров 29
Вопросы для повторения 34
Тема 2. Организация обмена информацией в микропроцессорной системе 35
2.1. Аппаратная поддержка шинного обмена 35
2.1.1. Организация шинного обмена данными 36
2.1.2. Дешифрация адреса и выборка устройств МС 39
2.1.3. Прохождение сигналов по магистрали 43
2.2. Циклы обмена данными по магистралям МС 45
2.2.1. Мультиплексирование шин и виды обмена данными в МС 47
2.2.2. Циклы программного обмена 49
2.2.3. Циклы обмена по прерываниям 53
2.2.4. Циклы обмена в режиме ПДП 56
Вопросы для повторения 58
Тема 3. Основы функционирования МП 59
3.1. Сегменты памяти и регистры МП 59
3.2. Адресация операндов 67
3.2.1. Методы адресации 69
3.2.2. Сегментная адресация памяти 72
3.3. Система команд микропроцессора 74
3.3.1. Команды пересылки данных 75
3.3.2. Арифметические команды 76
3.3.3. Логические команды 77
3.3.4. Команды переходов 79
Вопросы для повторения 81
Тема 4. Запоминающие устройства и программируемые логические матрицы 83
4.1. Запоминающие устройства 83
4.1.1. Оперативное запоминающее устройство 88
4.1.2. Постоянные запоминающие устройства (ПЗУ). 91
4.2. Программируемые логические матрицы 96
Вопросы для повторения 100
Тема 5. Специализированные процессоры и микроконтроллеры 101
5.1. Архитектура микроконтроллера и ПЛК 101

5.2. Реализация функций реального времени 107
5.3. Процессоры цифровой обработки сигналов 112
Вопросы для повторения 125
Тема 6. Внешние устройства ввода-вывода 126
6.1. Ввод-вывод сигналов 126
6.1.1. Вывод аналоговых сигналов 126
6.1.2. Ввод аналоговых сигналов 129
6.1.3. Параллельные порты и дискретный ввод-вывод 139
6.2. Последовательные интерфейсы и сетевые средства передачи данных 143
6.2.1. Последовательные интерфейсы 143
6.2.2. Телекоммуникационные и сетевые средства передачи данных 148
Вопросы для повторения 151
Практикум (лабораторный) 152
Лабораторная работа №1. Изучение и программирование DSP SPEEDY-33 в задачах цифровой обработки сигналов 152
Требования к содержанию, оформлению и порядку выполнения 152
Общая постановка задачи 153
Описание лабораторного стенда 153
Выполнение работы 164
Пример создания виртуального измерительного прибора на платформе DSP NI SPEEDY-33 167
Контрольные вопросы к защите 172
Способ оценки результатов 172
Лабораторная работа №2. Управление дискретными процессами на базе ПЛК Siemens LOGO! 172
Требования к содержанию, оформлению и порядку выполнения 172
Описание лабораторного стенда 173
Общая постановка задачи 180
Выполнение работы 180
Контрольные вопросы к защите 182
Способ оценки результатов 182
Лабораторная работа №3. Программирование ПЛК серии VIPA CPU216 183
Требования к содержанию, оформлению и порядку выполнения 183
Описание лабораторного стенда 183
Общая постановка задачи 189
Выполнение работы 189
Контрольные вопросы к защите 191
Способ оценки результатов 191
Лабораторная работа №4. Изучение PC-совместимых ПЛК серии ADAM-5510 191
Требования к содержанию, оформлению и порядку выполнения 191

Описание лабораторного стенда 192
Общая постановка задачи 197
Выполнение работы 197
Контрольные вопросы к защите 199
Способ оценки результатов 200
Лабораторная работа №5. Управление микроклиматом помещения АТС на базе ПЛК серии ADAM-5510 200
Требования к содержанию, оформлению и порядку выполнения 200
Описание лабораторного стенда 201
Общая постановка задачи 206
Выполнение работы 206
Контрольные вопросы к защите 208
Способ оценки результатов 208
Список сокращений 208
Список источников* 209

Введение

Специализированные процессоры и микроконтроллеры, обеспечивающие техническую реализацию устройств обработки данных и управления, нашли широкое применение в системах радиосвязи, радиовещания и телевидения. Так, современные сигнальные процессоры способны производить достаточно сложную цифровую обработку аудио и видео сигналов в реальном времени – «на лету».

Аппаратные и программные средства цифровых систем обработки сигналов, должны отвечать требованиям унификации и стандартизации, метрологической, конструктивной, информационной и эксплуатационной совместимости, принципам модульного построения. При выполнении этих требований обеспечивается низкая себестоимость, высокая надежность, гарантированное и высокое качество цифровой обработки сигналов. От аппаратной платформы зависит не только точность и надежность функционирования, но и возможность использования сложных и наукоемких алгоритмов цифровой обработки сигналов.

Введем основные понятия, являющиеся предметом изучения данной дисциплины.

Микропроце́ссор — процессор (устройство, отвечающее за выполнение арифметических, логических и операций управления, записанных в машинном коде), реализованный в виде одной микросхемы или комплекта из нескольких специализированных микросхем. Некоторые авторы относят к микропроцессорам только устройства, реализованные строго на одной микросхеме. Такое определение расходится как с академическими источниками, так и с коммерческой практикой (например, варианты микропроцессоров Intel и AMD в корпусах типа SECC реализуются на нескольких чипах). В настоящее время, в связи с очень незначительным распространением процессоров, не являющихся микропроцессорами, в бытовой лексике термины «микропроцессор» и «процессор» практически равнозначны.

Микропроцессорная система (МС) представляет собой функционально законченное изделие, состоящее из одного или нескольких устройств, главным образом микропроцессорных: микропроцессора и/или микроконтроллера.

Цифрова́я обрабо́тка сигна́лов (ЦОС) – преобразование сигналов, представленных в цифровой форме.

Классический подход ЦОС состоит в том, что любой непрерывный (аналоговый) сигнал может быть подвергнут дискретизации по времени и квантованию по уровню (оцифровке), то есть представлен в цифровой форме.

Если частота дискретизации сигнала не меньше, чем удвоенная наивысшая частота в спектре сигнала, то полученный дискретный сигнал эквивалентен исходному сигналу (согласно теореме Котельникова). Процесс преобразования сигналов называется фильтрацией, а устройство, выполняющее фильтрацию, называется фильтр. Поскольку отсчёты сигналов поступают с постоянной скоростью, фильтр должен успевать обрабатывать текущий отсчет до поступления следующего отсчёта сигнала по времени, то есть обрабатывать сигнал в реальном времени.

Различают методы обработки сигналов во временной (англ. time domain) и в частотной (англ. frequency domain) области. Эквивалентность частотно-временных преобразований однозначно определяется через преобразование Фурье.

В последние годы при обработке сигналов и изображений широко используется новый математический базис представления сигналов с помощью "коротких волночек" – вейвлетов. С его помощью могут обрабатываться нестационарные сигналы, сигналы с разрывами и иными особенностями и сигналы в виде пачек.

Всё это полностью применимо не только к непрерывным сигналам, но и к прерывистым, дискретным (импульсным и релейным), а также к сигналам, записанным на запоминающие устройства в различных форматах.

Для обработки сигналов (фильтрации) в реальном времени применяют специальные вычислительные устройства – цифровые сигнальные процессоры (DSP – англ. digital signal processing), – часто встречается название – процессоры цифровой обработки сигналов (ПЦОС).

С целью изучения математических аспектов обработки сигналов используются пакеты расширения (чаще всего под именем Signal Processing) математических проблемно-ориентированых сред, таких как MATLAB (Signal Processing Toolbox), LabVIEW (DSP Module) и др.

Цифровая обработка сигналов широко используется в современных системах радиосвязи, радиовещания и телевидения. При переходе телевещания на территории РФ в цифровой формат, изучение технических средств и алгоритмов ЦОС становится особенно актуальным.

Тема 1. Общие вопросы и основные понятия о микропроцессорных системах

Цели и задачи изучения темы:

Основной целью изучения данной темы является знакомство с понятием «микропроцессор» (МП) и «микропроцессорная система» (МС), способов построения и архитектуры МС.

Задачи:

1. Получить основные понятия о реализации микропроцессорных системах управления: жесткая логика и программное управление.
2. Ознакомиться с обобщенной структурой МП.
3. Рассмотреть основные режимы работы МП.
4. Рассмотреть различные классы МП.

1.1. Основные понятия о микропроцессорах и микропроцессорных системах

Микропроцессорная система может рассматриваться как частный случай электронной системы, предназначенной для обработки входных сигналов и выдачи выходных сигналов (рис. 1.1). В качестве входных и выходных сигналов при этом могут использоваться аналоговые сигналы, одиночные цифровые сигналы, цифровые коды, последовательности цифровых кодов. Внутри системы может производиться хранение, накопление сигналов (или информации), но суть от этого не меняется. Если система цифровая (а микропроцессорные системы относятся к разряду цифровых), то входные аналоговые сигналы преобразуются в последовательности кодов выборок с помощью АЦП, а выходные аналоговые сигналы формируются из последовательности кодов выборок с помощью ЦАП. Обработка и хранение информации производятся в цифровом виде [1].

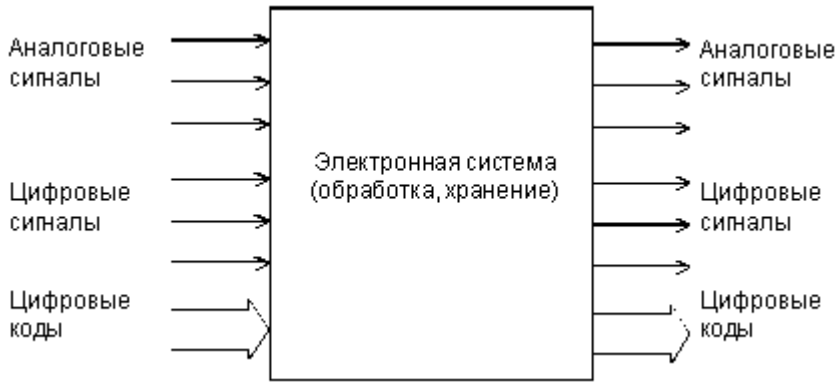


Рис. 1.1. Электронная система управления

Характерная особенность традиционной цифровой системы состоит в том, что алгоритмы обработки и хранения информации в ней жестко связаны

со схемотехникой системы. То есть изменение этих алгоритмов возможно только путем изменения структуры системы, замены электронных узлов, входящих в систему, и/или связей между ними. Например, если нам нужна дополнительная операция суммирования, то необходимо добавить в структуру системы лишний сумматор. Или если нужна дополнительная функция хранения кода в течение одного такта, то мы должны добавить в структуру еще один регистр. Естественно, это практически невозможно сделать в процессе эксплуатации, обязательно нужен новый производственный цикл проектирования, изготовления, отладки всей системы. Именно поэтому традиционная цифровая система часто называется системой на "жесткой логике".

Любая система на "жесткой логике" обязательно представляет собой специализированную систему, настроенную исключительно на одну задачу или (реже) на несколько близких, заранее известных задач. Это имеет свои бесспорные преимущества.

Во-первых, специализированная система (в отличие от универсальной) никогда не имеет аппаратурной избыточности, то есть каждый ее элемент обязательно работает в полную силу (конечно, если эта система грамотно спроектирована).

Во-вторых, именно специализированная система может обеспечить максимально высокое быстродействие, так как скорость выполнения алгоритмов обработки информации определяется в ней только быстродействием отдельных логических элементов и выбранной схемой путей прохождения информации. А именно логические элементы всегда обладают максимальным на данный момент быстродействием.

Но в то же время большим недостатком цифровой системы на "жесткой логике" является то, что для каждой новой задачи ее надо проектировать и изготавливать заново. Это процесс длительный, дорогостоящий, требующий высокой квалификации исполнителей. А если решаемая задача вдруг изменяется, то вся аппаратура должна быть полностью заменена. В нашем быстро меняющемся мире это довольно расточительно.

Путь преодоления этого недостатка довольно очевиден: надо построить такую систему, которая могла бы легко адаптироваться под любую задачу, перестраиваться с одного алгоритма работы на другой без изменения аппаратуры. И задавать тот или иной алгоритм мы тогда могли бы путем ввода в систему некой дополнительной управляющей информации, ***программы*** работы системы (рис. 1.2). Тогда система станет универсальной, или ***программируемой***, не жесткой, а гибкой. Именно это и обеспечивает микропроцессорная система.

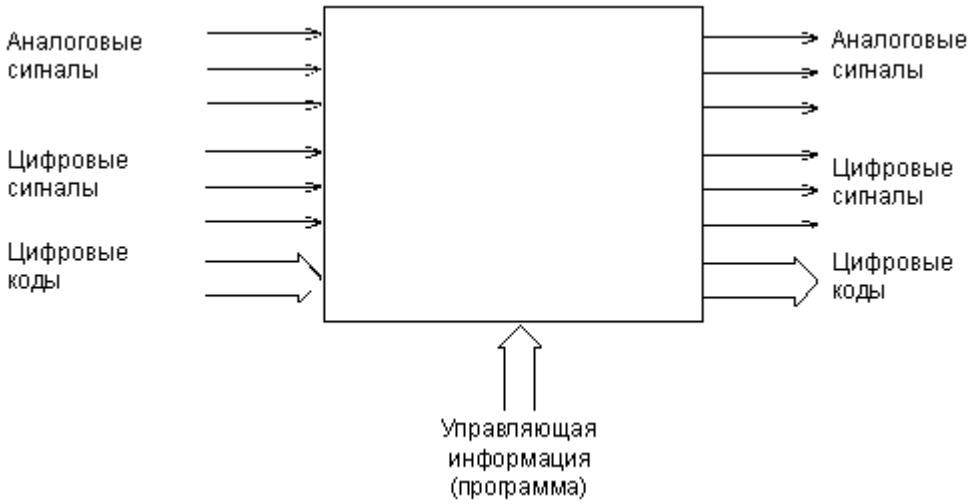


Рис. 1.2. Программируемая электронная система управления.

Но любая универсальность обязательно приводит к избыточности. Ведь решение максимально трудной задачи требует гораздо больше средств, чем решение максимально простой задачи. Поэтому сложность универсальной системы должна быть такой, чтобы обеспечивать решение самой трудной задачи, а при решении простой задачи система будет работать далеко не в полную силу, будет использовать не все свои ресурсы. И чем проще решаемая задача, тем больше избыточность, и тем менее оправданной становится универсальность. Избыточность ведет к увеличению стоимости системы, снижению ее надежности, увеличению потребляемой мощности и т.д.

Кроме того, универсальность, как правило, приводит к существенному снижению быстродействия. Оптимизировать универсальную систему так, чтобы каждая новая задача решалась максимально быстро, попросту невозможно. Общее правило таково: чем больше универсальность, гибкость, тем меньше быстродействие. Более того, для универсальных систем не существует таких задач (пусть даже и самых простых), которые бы они решали с максимально возможным быстродействием. За все приходится платить.

Таким образом, можно сделать следующий вывод. Системы на "жесткой логике" хороши там, где решаемая задача не меняется длительное время, где требуется самое высокое быстродействие, где алгоритмы обработки информации предельно просты. А универсальные, программируемые системы хороши там, где часто меняются решаемые задачи, где высокое быстродействие не слишком важно, где алгоритмы обработки информации сложные. То есть любая система хороша на своем месте.

Однако за последние десятилетия быстродействие универсальных (микропроцессорных) систем сильно выросло (на несколько порядков). К тому же большой объем выпуска микросхем для этих систем привел к резкому снижению их стоимости. В результате область применения систем

на "жесткой логике" резко сузилась. Более того, высокими темпами развиваются сейчас программируемые системы, предназначенные для решения одной задачи или нескольких близких задач. Они удачно совмещают в себе как достоинства систем на "жесткой логике", так и программируемых систем, обеспечивая сочетание достаточно высокого быстродействия и необходимой гибкости. Так что вытеснение "жесткой логики" продолжается.

1.1.1. Понятие «микропроцессор»

Ядром любой микропроцессорной системы является микропроцессор или просто процессор (от английского processor). Перевести на русский язык это слово правильнее всего как "обработчик", так как именно микропроцессор – это тот узел, блок, который производит всю обработку информации внутри микропроцессорной системы. Остальные узлы выполняют всего лишь вспомогательные функции: хранение информации (в том числе и управляющей информации, то есть программы), связи с внешними устройствами, связи с пользователем и т.д. Процессор заменяет практически всю "жесткую логику", которая понадобилась бы в случае традиционной цифровой системы. Он выполняет арифметические функции (сложение, умножение и т.д.), логические функции (сдвиг, сравнение, маскирование кодов и т.д.), временное хранение кодов (во внутренних регистрах), пересылку кодов между узлами микропроцессорной системы и многое другое. Количество таких элементарных операций, выполняемых процессором, может достигать нескольких сотен. Процессор можно сравнить с мозгом системы [1].

Но при этом надо учитывать, что все свои операции процессор выполняет ***последовательно***, то есть одну за другой, по очереди. Конечно, существуют процессоры с параллельным выполнением некоторых операций, встречаются также микропроцессорные системы, в которых несколько процессоров работают над одной задачей параллельно, но это редкие исключения. С одной стороны, последовательное выполнение операций – несомненное достоинство, так как позволяет с помощью всего одного процессора выполнять любые, самые сложные алгоритмы обработки информации. Но, с другой стороны, последовательное выполнение операций приводит к тому, что время выполнения алгоритма зависит от его сложности. Простые алгоритмы выполняются быстрее сложных. То есть микропроцессорная система способна сделать все, но работает она не слишком быстро, ведь все информационные потоки приходится пропускать через один-единственный узел – микропроцессор (рис. 1.3). В традиционной цифровой системе можно легко организовать параллельную обработку всех потоков информации, правда, ценой усложнения схемы.

Рис. 1.3. Информационные потоки в МП

Итак, микропроцессор способен выполнять множество операций. Но откуда он узнает, какую операцию ему надо выполнять в данный момент? Именно это определяется управляющей информацией, программой. Программа представляет собой набор ***команд (инструкций)***, то есть цифровых кодов, расшифровав которые, процессор узнает, что ему надо делать. Программа от начала и до конца составляется человеком, программистом, а процессор выступает в роли послушного исполнителя этой программы, никакой инициативы он не проявляет (если, конечно, исправен). Поэтому сравнение процессора с мозгом не слишком корректно. Он всего лишь исполнитель того алгоритма, который заранее составил для него человек. Любое отклонение от этого алгоритма может быть вызвано только неисправностью процессора или каких-нибудь других узлов микропроцессорной системы.

Все команды, выполняемые процессором, образуют ***систему команд*** процессора. Структура и объем системы команд процессора определяют его быстродействие, гибкость, удобство использования. Всего команд у процессора может быть от нескольких десятков до нескольких сотен. Система команд может быть рассчитана на узкий круг решаемых задач (у специализированных процессоров) или на максимально широкий круг задач (у универсальных процессоров). Коды команд могут иметь различное количество разрядов (занимать от одного до нескольких байт). Каждая команда имеет свое время выполнения, поэтому время выполнения всей программы зависит не только от количества команд в программе, но и от того, какие именно команды используются.

Для выполнения команд в структуру процессора входят внутренние регистры, арифметико-логическое устройство (АЛУ, ALU – Arithmetic Logic Unit), мультиплексоры, буферы, регистры и другие узлы. Работа всех узлов синхронизируется общим внешним тактовым сигналом процессора. То есть процессор представляет собой довольно сложное цифровое устройство (рис. 1.4).

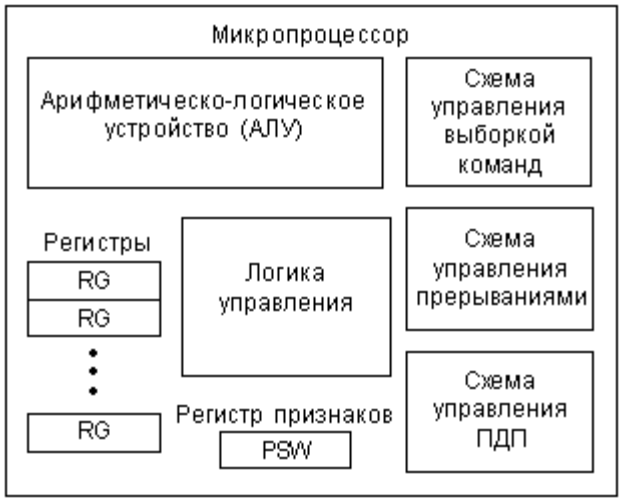


Рис. 1.4. Пример структуры простейшего МП

Основные функции показанных узлов следующие.

Схема управления выборкой команд выполняет чтение команд из памяти и их дешифрацию. В первых микропроцессорах было невозможно одновременное выполнение предыдущей команды и выборка следующей команды, так как процессор не мог совмещать эти операции. Но уже в 16-разрядных процессорах появляется так называемый конвейер (очередь) команд, позволяющий выбирать несколько следующих команд, пока выполняется предыдущая. Два процесса идут параллельно, что ускоряет работу процессора. Конвейер представляет собой небольшую внутреннюю память процессора, в которую при малейшей возможности (при освобождении внешней шины) записывается несколько команд, следующих за исполняемой. Читаются эти команды процессором в том же порядке, что и записываются в конвейер (это память типа FIFO, First In – First Out, первый вошел – первый вышел). Правда, если выполняемая команда предполагает переход не на следующую ячейку памяти, а на удаленную (с меньшим или большим адресом), конвейер не помогает, и его приходится сбрасывать. Но такие команды встречаются в программах сравнительно редко.

Развитием идеи конвейера стало использование внутренней кэш-памяти процессора, которая заполняется командами, пока процессор занят выполнением предыдущих команд. Чем больше объем кэш-памяти, тем меньше вероятность того, что ее содержимое придется сбросить при команде перехода. Понятно, что обрабатывать команды, находящиеся во внутренней памяти, процессор может гораздо быстрее, чем те, которые расположены во внешней памяти. В кэш-памяти могут храниться и данные, которые обрабатываются в данный момент, это также ускоряет работу. Для большего ускорения выборки команд в современных процессорах применяют совмещение выборки и дешифрации, одновременную дешифрацию нескольких команд, несколько параллельных конвейеров команд, предсказание команд переходов и некоторые другие методы.

Арифметико-логическое устройство предназначено для обработки информации в соответствии с полученной процессором командой. Примерами обработки могут служить логические операции (типа логического "И", "ИЛИ", "Исключающего ИЛИ" и т.д.) то есть побитные операции над операндами, а также арифметические операции (типа сложения, вычитания, умножения, деления и т.д.). Над какими кодами производится операция, куда помещается ее результат – определяется выполняемой командой. Если команда сводится всего лишь к пересылке данных без их обработки, то АЛУ не участвует в ее выполнении.

Рассмотрим порядок исполнения инструкций обработки данных – выполнения арифметических или логических функций. Во многих случаях инструкция работает с парой операндов – операндом назначения dest (destination) и операндом-источником src (source). Традиционная схема действия инструкции: dest = F (dest, src), где F - некоторая функция от двух переменных. Это означает, что при выполнении инструкции процессор извлекает из указанных в инструкции мест (регистр, память, константа в самой инструкции) пару двоичных чисел, и результат действия над ними записывает на место одного из них (dest). Для выполнения той же функции над следующей парой чисел требуется повторное исполнение инструкции, но уже с другой парой операндов.

Быстродействие АЛУ во многом определяет производительность процессора. Причем важна не только частота тактового сигнала, которым тактируется АЛУ, но и количество тактов, необходимое для выполнения той или иной команды. Для повышения производительности разработчики стремятся довести время выполнения команды до одного такта, а также обеспечить работу АЛУ на возможно более высокой частоте. Один из путей решения этой задачи состоит в уменьшении количества выполняемых АЛУ команд, создание процессоров с уменьшенным набором команд (так называемые RISC-процессоры). Другой путь повышения производительности процессора – использование нескольких параллельно работающих АЛУ.

Что касается операций над числами с плавающей точкой и других специальных сложных операций, то в системах на базе первых процессоров их реализовали последовательностью более простых команд, специальными подпрограммами, однако затем были разработаны специальные вычислители – математические сопроцессоры, которые заменяли основной процессор на время выполнения таких команд. В современных микропроцессорах математические сопроцессоры входят в структуру как составная часть.

Регистры процессора представляют собой по сути ячейки очень быстрой памяти и служат для временного хранения различных кодов: данных, адресов, служебных кодов. Операции с этими кодами выполняются предельно быстро, поэтому, в общем случае, чем больше внутренних регистров, тем лучше. Кроме того, на быстродействие процессора сильно влияет разрядность регистров. Именно разрядность регистров и АЛУ

называется внутренней разрядностью процессора, которая может не совпадать с внешней разрядностью.

По отношению к назначению внутренних регистров существует два основных подхода. Первого придерживается, например, компания Intel, которая каждому регистру отводит строго определенную функцию. С одной стороны, это упрощает организацию процессора и уменьшает время выполнения команды, но с другой – снижает гибкость, а иногда и замедляет работу программы. Например, некоторые арифметические операции и обмен с устройствами ввода/вывода проводятся только через один регистр – аккумулятор, в результате чего при выполнении некоторых процедур может потребоваться несколько дополнительных пересылок между регистрами. Второй подход состоит в том, чтобы все (или почти все) регистры сделать равноправными, как, например, в 16-разрядных процессорах Т-11 фирмы DEC. При этом достигается высокая гибкость, но необходимо усложнение структуры процессора. Существуют и промежуточные решения, в частности, в процессоре MC68000 фирмы Motorola половина регистров использовалась для данных, и они были взаимозаменяемы, а другая половина – для адресов, и они также взаимозаменяемы.

Регистр признаков (регистр состояния) занимает особое место, хотя он также является внутренним регистром процессора. Содержащаяся в нем информация – это не данные, не адрес, а слово состояния процессора (ССП, PSW – Processor Status Word). Каждый бит этого слова (флаг) содержит информацию о результате предыдущей команды. Например, есть бит нулевого результата, который устанавливается в том случае, когда результат выполнения предыдущей команды – нуль, и очищается в том случае, когда результат выполнения команды отличен от нуля. Эти биты (флаги) используются командами условных переходов, например, командой перехода в случае нулевого результата. В этом же регистре иногда содержатся флаги управления, определяющие режим выполнения некоторых команд.

Схема управления прерываниями обрабатывает поступающий на процессор запрос прерывания, определяет адрес начала программы обработки прерывания (адрес вектора прерывания), обеспечивает переход к этой программе после выполнения текущей команды и сохранения в памяти (в стеке) текущего состояния регистров процессора. По окончании программы обработки прерывания процессор возвращается к прерванной программе с восстановленными из памяти (из стека) значениями внутренних регистров. Подробнее о стеке будет рассказано в следующем разделе.

Схема управления прямым доступом к памяти служит для временного отключения процессора от внешних шин и приостановки работы процессора на время предоставления прямого доступа запросившему его устройству.

Логика управления организует взаимодействие всех узлов процессора, перенаправляет данные, синхронизирует работу процессора с внешними сигналами, а также реализует процедуры ввода и вывода информации. Эта

часть микропроцессора технические реализовывается по принципу «Жесткой логики».

Таким образом, в ходе работы процессора схема выборки команд выбирает последовательно команды из памяти, затем эти команды выполняются, причем в случае необходимости обработки данных подключается АЛУ. На входы АЛУ могут подаваться обрабатываемые данные из памяти или из внутренних регистров. Во внутренних регистрах хранятся также коды адресов обрабатываемых данных, расположенных в памяти. Результат обработки в АЛУ изменяет состояние регистра признаков и записывается во внутренний регистр или в память (как источник, так и приемник данных указывается в составе кода команды). При необходимости информация может переписываться из памяти (или из устройства ввода/вывода) во внутренний регистр или из внутреннего регистра в память (или в устройство ввода/вывода).

Впрочем, для разработчика микропроцессорных систем информация о тонкостях внутренней структуры процессора не слишком важна. Разработчик должен рассматривать процессор как "черный ящик", который в ответ на входные и управляющие коды производит ту или иную операцию и выдает выходные сигналы. Разработчику необходимо знать систему команд, режимы работы процессора, а также правила взаимодействия процессора с внешним миром или, как их еще называют, протоколы обмена информацией. О внутренней структуре процессора надо знать только то, что необходимо для выбора той или иной команды, того или иного режима работы

1.1.2. Шинная структура связей

Для достижения максимальной универсальности и упрощения протоколов обмена информацией в микропроцессорных системах применяется так называемая шинная структура связей между отдельными устройствами, входящими в систему. Суть шинной структуры связей сводится к следующему [1].

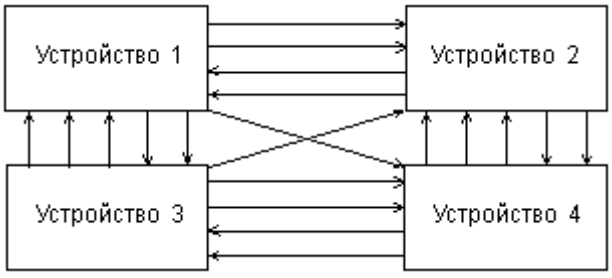


Рис. 1.5. Классическая структура связей

При классической структуре связей (рис. 1.5) все сигналы и коды между устройствами передаются по отдельным линиям связи. Каждое устройство, входящее в систему, передает свои сигналы и коды независимо от других устройств. При этом в системе получается очень много линий связи (число линий связи определяется по формуле $N*(N–1)/2$, где N – число устройств в системе) и разных протоколов обмена информацией.

При шинной структуре связей (рис. 1.6) все сигналы между устройствами передаются по одним и тем же линиям связи, но в разное время (это называется мультиплексированной передачей). Причем передача по всем линиям связи может осуществляться в обоих направлениях (так называемая двунаправленная передача). В результате количество линий связи существенно сокращается, а правила обмена (протоколы) унифицируются и упрощаются. Группа линий связи, по которым передаются сигналы или коды как раз и называется шиной (англ. bus).

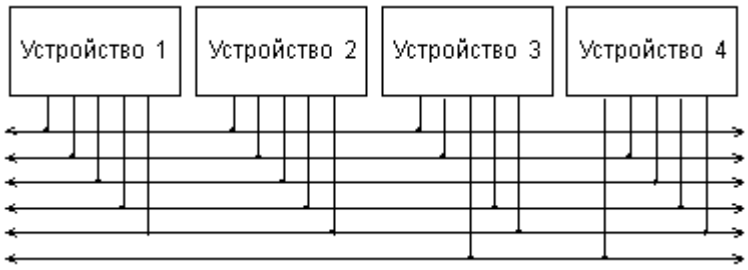


Рис. 1.6. Шинная структура связей

Понятно, что при шинной структуре связей легко осуществляется пересылка всех информационных потоков в нужном направлении, например, их можно пропустить через один процессор, что очень важно для микропроцессорной системы. Однако при шинной структуре связей вся информация передается по линиям связи последовательно во времени, по очереди, что снижает быстродействие системы по сравнению с классической структурой связей.

Большое достоинство шинной структуры связей состоит в том, что все устройства, подключенные к шине, должны принимать и передавать информацию по одним и тем же правилам (протоколам обмена информацией по шине). Соответственно, все узлы, отвечающие за обмен с шиной в этих устройствах, должны быть единообразны, унифицированы.

Существенный недостаток шинной структуры связан с тем, что все устройства подключаются к каждой линии связи параллельно. Поэтому любая неисправность любого устройства может вывести из строя всю систему, если она нарушает работу хотя бы одной линии связи. По этой же причине отладка системы с шинной структурой связей довольно сложна и обычно требует специального оборудования.

В системах с шинной структурой связей применяют все три существующие разновидности выходных каскадов цифровых микросхем:

- стандартный выход или выход с двумя состояниями (обозначается 2С, 2S, реже ТТЛ, TTL);
- выход с открытым коллектором (обозначается ОК);
- выход с тремя состояниями или (что то же самое) с возможностью отключения (обозначается 3С, 3S, с z-состоянием).

Упрощенно эти три типа выходных каскадов могут быть представлены в виде схем на рис. 1.7.

У выхода 2С два ключа замыкаются по очереди, что соответствует уровням логической единицы (верхний ключ замкнут) и логического нуля (нижний ключ замкнут). У выхода ОК замкнутый ключ формирует уровень логического нуля, разомкнутый – логической единицы. У выхода 3С ключи могут замыкаться по очереди (как в случае 2С), а могут размыкаться одновременно, образуя третье, высокоимпедансное, состояние. Переход в третье состояние (Z-состояние) управляется сигналом на специальном входе EZ.

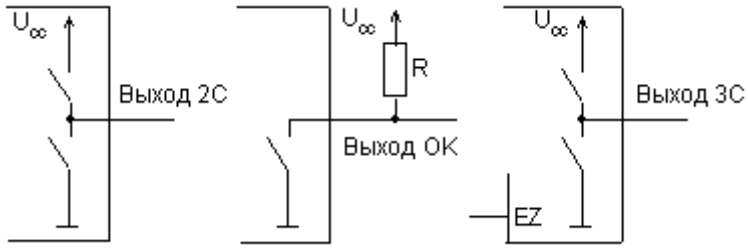


Рис. 1.7. Три типа выходов цифровых микросхем

Выходные каскады типов 3С и ОК позволяют объединять несколько выходов микросхем для получения мультиплексированных (рис. 1.8) или двунаправленных (рис. 1.9) линий.

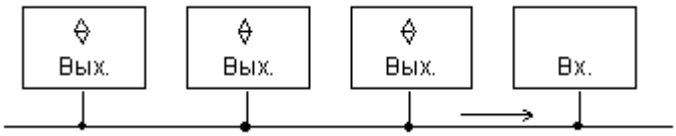


Рис. В.8. Мультиплексированная линия

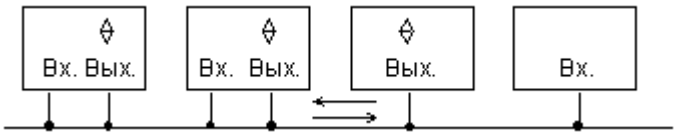


Рис. 1.9. Двунаправленная лини

При этом в случае выходов 3С необходимо обеспечить, чтобы на линии всегда работал только один активный выход, а все остальные выходы находились бы в это время в третьем состоянии, иначе возможны конфликты. Объединенные выходы ОК могут работать все одновременно, без всяких конфликтов.

Типичная структура микропроцессорной системы приведена на рис. 1.10. Она включает в себя три основных типа устройств:

- процессор;
- память, включающую оперативную память (ОЗУ, RAM – Random Access Memory) и постоянную память (ПЗУ, ROM –Read Only Memory), которая служит для хранения данных и программ;
- устройства ввода/вывода (УВВ, I/O – Input/Output Devices), служащие для связи микропроцессорной системы с внешними устройствами, для

приема (ввода, чтения, Read) входных сигналов и выдачи (вывода, записи, Write) выходных сигналов.

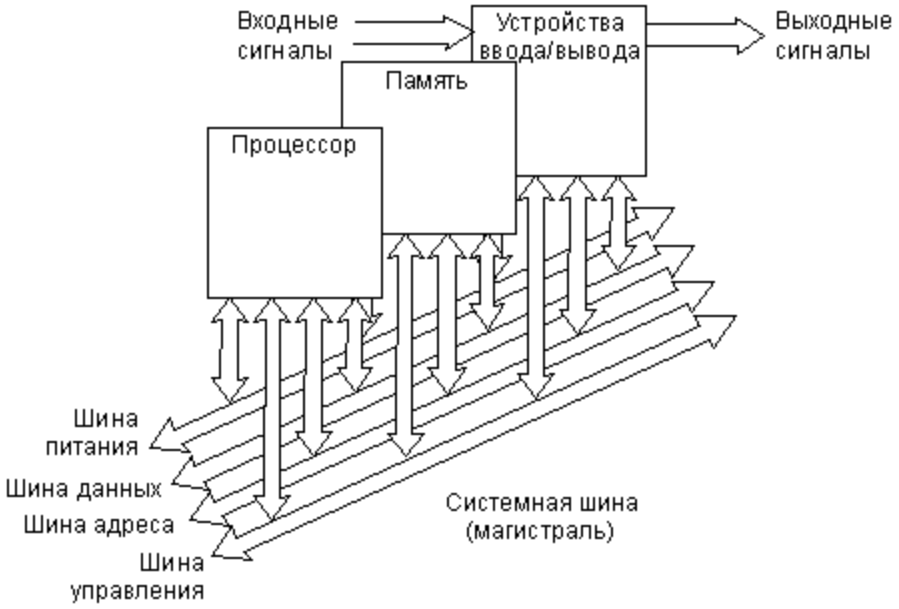


Рис. 1.10. Структура микропроцессорной системы

Все устройства микропроцессорной системы объединяются общей системной шиной (она же называется еще ***системной магистралью*** или ***каналом***). Системная магистраль включает в себя четыре основные шины нижнего уровня:

- шина адреса (Address Bus);
- шина данных (Data Bus);
- шина управления (Control Bus);
- шина питания (Power Bus).

Шина адреса служит для определения адреса (номера) устройства, с которым процессор обменивается информацией в данный момент. Каждому устройству (кроме процессора), каждой ячейке памяти в микропроцессорной системе присваивается собственный адрес. Когда код какого-то адреса выставляется процессором на шине адреса, устройство, которому этот адрес приписан, понимает, что ему предстоит обмен информацией. Шина адреса может быть однонаправленной или двунаправленной.

Шина данных – это основная шина, которая используется для передачи информационных кодов между всеми устройствами микропроцессорной системы. Обычно в пересылке информации участвует процессор, который передает код данных в какое-то устройство или в ячейку памяти или же принимает код данных из какого-то устройства или из ячейки памяти. Но возможна также и передача информации между устройствами без участия процессора. Шина данных всегда двунаправленная.

Шина управления в отличие от шины адреса и шины данных состоит из отдельных управляющих сигналов. Каждый из этих сигналов во время обмена информацией имеет свою функцию. Некоторые сигналы служат для стробирования передаваемых или принимаемых данных (то есть определяют

моменты времени, когда информационный код выставлен на шину данных или адреса). Другие управляющие сигналы могут использоваться для подтверждения приема данных, для сброса всех устройств в исходное состояние, для тактирования всех устройств и т.д. Линии шины управления могут быть однонаправленными или двунаправленными.

Наконец, шина питания предназначена не для пересылки информационных сигналов, а для питания системы. Она состоит из линий питания и общего провода. В микропроцессорной системе может быть один источник питания (чаще +5 В) или несколько источников питания (обычно еще –5 В, +12 В и –12 В). Каждому напряжению питания соответствует своя линия связи. Все устройства подключены к этим линиям параллельно.

Если в микропроцессорную систему надо ввести входной код (или входной сигнал), то процессор по шине адреса обращается к нужному устройству ввода/вывода и принимает по шине данных входную информацию. Если из микропроцессорной системы надо вывести выходной код (или выходной сигнал), то процессор обращается по шине адреса к нужному устройству ввода/вывода и передает ему по шине данных выходную информацию.

Если информация должна пройти сложную многоступенчатую обработку, то процессор может хранить промежуточные результаты в системной оперативной памяти. Для обращения к любой ячейке памяти процессор выставляет ее адрес на шину адреса и передает в нее информационный код по шине данных или же принимает из нее информационный код по шине данных. В памяти (оперативной и постоянной) находятся также и управляющие коды (команды выполняемой процессором программы), которые процессор также читает по шине данных с адресацией по шине адреса. Постоянная память используется в основном для хранения программы начального пуска микропроцессорной системы, которая выполняется каждый раз после включения питания. Информация в нее заносится изготовителем раз и навсегда.

Таким образом, в микропроцессорной системе все информационные коды и коды команд передаются по шинам последовательно, по очереди. Это определяет сравнительно невысокое быстродействие микропроцессорной системы. Оно ограничено обычно даже не быстродействием процессора (которое тоже очень важно) и не скоростью обмена по системной шине (магистрали), а именно последовательным характером передачи информации по системной шине (магистрали).

Важно учитывать, что устройства ввода/вывода чаще всего представляют собой устройства на "жесткой логике". На них может быть возложена часть функций, выполняемых микропроцессорной системой. Поэтому у разработчика всегда имеется возможность перераспределять функции системы между аппаратной и программной реализациями оптимальным образом. Аппаратная реализация ускоряет выполнение функции, но имеет недостаточную гибкость. Программная реализация

значительно медленнее, но обеспечивает высокую гибкость. Аппаратная реализация функций увеличивает стоимость системы и ее энергопотребление, программная – не увеличивает. Чаще всего применяется комбинирование аппаратных и программных функций.

Иногда устройства ввода/вывода имеют в своем составе процессор, то есть представляют собой небольшую специализированную микропроцессорную систему. Это позволяет переложить часть программных функций на устройства ввода/вывода, разгрузив центральный процессор системы

1.1.3. Режимы работы микропроцессорной системы

Как уже отмечалось, микропроцессорная система обеспечивает большую гибкость работы, она способна настраиваться на любую задачу. Гибкость эта обусловлена прежде всего тем, что функции, выполняемые системой, определяются программой (программным обеспечением, software), которую выполняет процессор. Аппаратура (аппаратное обеспечение, hardware) остается неизменной при любой задаче. Записывая в память системы программу, можно заставить микропроцессорную систему выполнять любую задачу, поддерживаемую данной аппаратурой. К тому же шинная организация связей микропроцессорной системы позволяет довольно легко заменять аппаратные модули, например, заменять память на новую большего объема или более высокого быстродействия, добавлять или модернизировать устройства ввода/вывода, наконец, заменять процессор на более мощный. Это также позволяет увеличить гибкость системы, продлить ее жизнь при любом изменении требований к ней [1].

Но гибкость микропроцессорной системы определяется не только этим. Настраиваться на задачу помогает еще и выбор режима работы системы, то есть режима обмена информацией по системной магистрали (шине).

Практически любая развитая микропроцессорная система (в том числе и компьютер) поддерживает три основных режима обмена по магистрали:

- программный обмен информацией;
- обмен с использованием прерываний (Interrupts);
- обмен с использованием прямого доступа к памяти (ПДП, DMA – Direct Memory Access).

Программный обмен информацией является основным в любой микропроцессорной системе. Он предусмотрен всегда, без него невозможны другие режимы обмена. В этом режиме процессор является единоличным хозяином (или задатчиком, Master) системной магистрали. Все операции (циклы) обмена информацией в данном случае инициируются только процессором, все они выполняются строго в порядке, предписанном исполняемой программой.

Процессор читает (выбирает) из памяти коды команд и исполняет их, читая данные из памяти или из устройства ввода/вывода, обрабатывая их, записывая данные в память или передавая их в устройство ввода/вывода.

Путь процессора по программе может быть линейным, циклическим, может содержать переходы (прыжки), но он всегда непрерывен и полностью находится под контролем процессора. Ни на какие внешние события, не связанные с программой, процессор не реагирует (рис. 1.11). Все сигналы на магистрали в данном случае контролируются процессором.

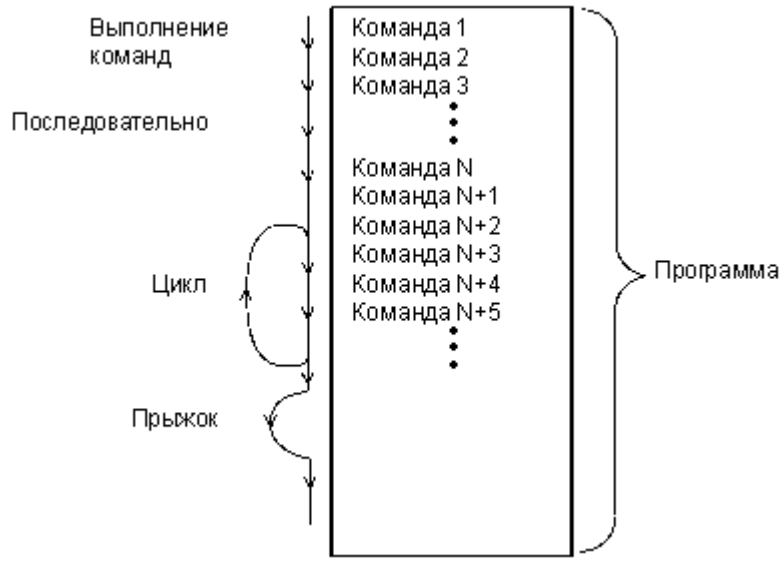


Рис. 1.11. Программный обмен информацией

Обмен по прерываниям используется тогда, когда необходима реакция микропроцессорной системы на какое-то внешнее событие, на приход внешнего сигнала. В случае компьютера внешним событием может быть, например, нажатие на клавишу клавиатуры или приход по локальной сети пакета данных. Компьютер должен реагировать на это, соответственно, выводом символа на экран или же чтением и обработкой принятого по сети пакета.

В общем случае организовать реакцию на внешнее событие можно тремя различными путями:

- с помощью постоянного программного контроля факта наступления события (так называемый метод опроса флага или polling);
- с помощью прерывания, то есть насильственного перевода процессора с выполнения текущей программы на выполнение экстренно необходимой программы;
- с помощью прямого доступа к памяти, то есть без участия процессора при его отключении от системной магистрали.

Проиллюстрировать эти три способа можно следующим простым примером. Допустим, вы готовите себе завтрак, поставив на плиту кипятиться молоко. Естественно, на закипание молока надо реагировать, причем срочно. Как это организовать? Первый путь – постоянно следить за молоком, но тогда вы ничего другого не сможете делать. Правильнее будет регулярно поглядывать на молоко, делая одновременно что-то другое. Это программный режим с опросом флага. Второй путь – установить на кастрюлю с молоком датчик, который подаст звуковой сигнал при закипании

молока, и спокойно заниматься другими делами. Услышав сигнал, вы выключите молоко. Правда, возможно, вам придется сначала закончить то, что вы начали делать, так что ваша реакция будет медленнее, чем в первом случае. Наконец, третий путь состоит в том, чтобы соединить датчик на кастрюле с управлением плитой так, чтобы при закипании молока горелка была выключена без вашего участия (правда, аналогия с ПДП здесь не очень точная, так как в данном случае на момент выполнения действия вас не отвлекают от работы).

Первый случай с опросом флага реализуется в микропроцессорной системе постоянным чтением информации процессором из устройства ввода/вывода, связанного с тем внешним устройством, на поведение которого необходимо срочно реагировать.

Во втором случае в режиме прерывания процессор, получив запрос прерывания от внешнего устройства (часто называемый IRQ – Interrupt ReQuest), заканчивает выполнение текущей команды и переходит к программе обработки прерывания. Закончив выполнение программы обработки прерывания, он возвращается к прерванной программе с той точки, где его прервали (рис. 1.12).

Здесь важно то, что вся работа, как и в случае программного режима, осуществляется самим процессором, внешнее событие просто временно отвлекает его. Реакция на внешнее событие по прерыванию в общем случае медленнее, чем при программном режиме. Как и в случае программного обмена, здесь все сигналы на магистрали выставляются процессором, то есть он полностью контролирует магистраль. Для обслуживания прерываний в систему иногда вводится специальный модуль контроллера прерываний, но он в обмене информацией не участвует. Его задача состоит в том, чтобы упростить работу процессора с внешними запросами прерываний. Этот контроллер обычно программно управляется процессором по системной магистрали.

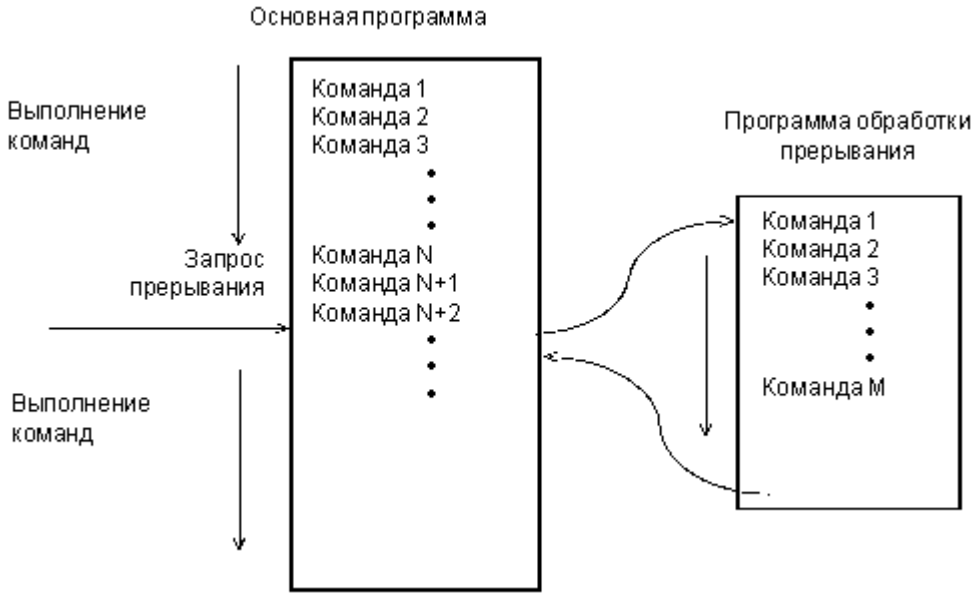


Рис. 1.12. Обслуживание прерывания

Естественно, никакого ускорения работы системы прерывание не дает. Его применение позволяет только отказаться от постоянного опроса флага внешнего события и временно, до наступления внешнего события, занять процессор выполнением каких-то других задач.

Прямой доступ к памяти (ПДП, DMA) – это режим, принципиально отличающийся от двух ранее рассмотренных режимов тем, что обмен по системной шине идет без участия процессора. Внешнее устройство, требующее обслуживания, сигнализирует процессору, что режим ПДП необходим, в ответ на это процессор заканчивает выполнение текущей команды и отключается от всех шин, сигнализируя запросившему устройству, что обмен в режиме ПДП можно начинать.

Операция ПДП сводится к пересылке информации из устройства ввода/вывода в память или же из памяти в устройство ввода/вывода. Когда пересылка информации будет закончена, процессор вновь возвращается к прерванной программе, продолжая ее с той точки, где его прервали (рис. 1.13). Это похоже на режим обслуживания прерываний, но в данном случае процессор не участвует в обмене. Как и в случае прерываний, реакция на внешнее событие при ПДП существенно медленнее, чем при программном режиме.

Понятно, что в этом случае требуется введение в систему дополнительного устройства (контроллера ПДП), которое будет осуществлять полноценный обмен по системной магистрали без всякого участия процессора. Причем процессор предварительно должен сообщить этому контроллеру ПДП, откуда ему следует брать информацию и/или куда ее следует помещать. Контроллер ПДП может считаться специализированным процессором, который отличается тем, что сам не участвует в обмене, не принимает в себя информацию и не выдает ее (рис. 1.14).

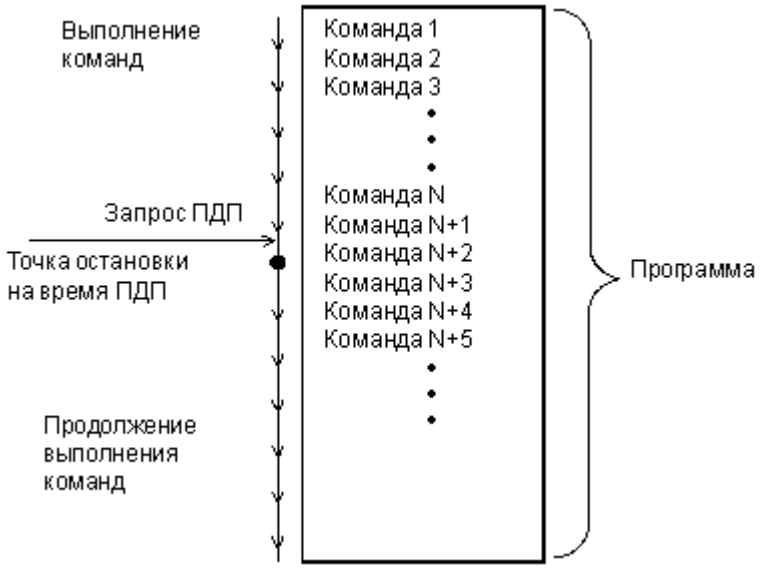


Рис. 1.13. Обслуживание ПДП

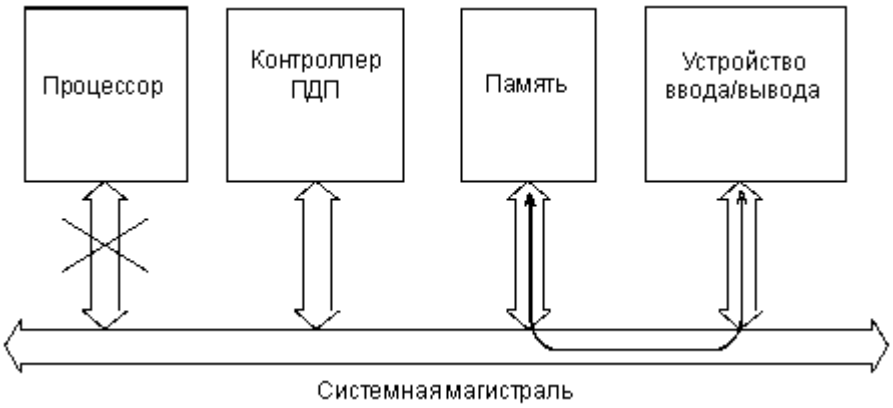


Рис. 1.14. Информационные потоки в режиме ПДП

В принципе контроллер ПДП может входить в состав устройства ввода/вывода, которому необходим режим ПДП или даже в состав нескольких устройств ввода/вывода. Теоретически обмен с помощью прямого доступа к памяти может обеспечить более высокую скорость передачи информации, чем программный обмен, так как процессор передает данные медленнее, чем специализированный контроллер ПДП. Однако на практике это преимущество реализуется далеко не всегда. Скорость обмена в режиме ПДП обычно ограничена возможностями магистрали. К тому же необходимость программного задания режимов контроллера ПДП может свести на нет выигрыш от более высокой скорости пересылки данных в режиме ПДП. Поэтому режим ПДП применяется редко.

Если в системе уже имеется самостоятельный контроллер ПДП, то это может в ряде случаев существенно упростить аппаратуру устройств ввода/вывода, работающих в режиме ПДП. В этом, пожалуй, состоит единственное бесспорное преимущество режима ПДП

1.1.4. Архитектура микропроцессорных систем

Архитектура микропроцессорной системы отражает принципы построения, логическую организацию, программное управление и взаимодействие основных узлов:

- структура памяти;
- способы доступа к памяти и внешним устройствам;
- возможность изменения конфигурации МС;
- система команд центрального МП;
- форматы данных;
- организация интерфейса.

Таким образом, архитектура – это программно-аппаратная модель МС. Под микроархитектурой понимается внутренняя реализация этой модели. Для одной и той же архитектуры разными фирмами и в разных поколениях применяются существенно различные микроархитектурные реализации, при этом, естественно, стремятся к максимальному повышению производительности (скорости исполнения программ).

Чаще всего, в подавляющем большинстве современных компьютеров, включая ПК и сложные многопроцессорные комплексы, реализована архитектура с общей, единой шиной для данных и команд (одношинная, или принстонская, ***фон-неймановская архитектура***) [1]. Эта архитектура была представлена американским математиком Джоном фон-Нейманом еще в 1945 году и имеет следующие основные признаки. Машина состоит из процессора (блока управления и арифметико-логического устройства (АЛУ)), памяти и устройств ввода/вывода (рис. 1.15). В основе архитектуры лежит представление алгоритма решения любой задачи в виде программы вычислений, состоящей из последовательности управляющих слов – команд. Один из важнейших принципов конструирования МС предложенный фон Нейманом – принцип хранимой программы был впервые реализован в Англии в 1949 году в машине EDSAC и используется и в современных компьютерах. Этот принцип требует, чтобы программа вводилась в память компьютера так же, как в нее вводятся данные, т.е. и команды и данные хранятся в одной и той же памяти.

Рис. 1.15. Архитектура с общей шиной данных и команд

Выполняемые действия определяются процессором, который выбирает и исполняет команды из памяти последовательно, адрес очередной команды задается "счетчиком команд" в блоке управления. Этот принцип исполнения называется последовательной передачей управления. Данные, с которыми работает программа, могут включать переменные – именованные области памяти, в которых сохраняются значения с целью дальнейшего использования в программе.

Для доступа к программам, командам и данным используются их адреса. В качестве адресов выступают номера ячеек памяти ЭВМ. Каждая команда содержит указания на конкретную выполняемую операцию, местонахождение (адреса) данных (операндов) и ряд служебных признаков. Операнды – переменные, значения которых участвуют в операциях преобразования данных. Список (массив) всех переменных (входных данных, промежуточных значений и результатов вычислений) является еще одним неотъемлемым элементом любой программы.

Фон-неймановская архитектура – не единственный вариант построения МС, есть и другие, которые не соответствуют указанным принципам (например, управляющие однокристальные микроконтроллеры, потоковые машины). Альтернативный тип архитектуры микропроцессорной системы – это архитектура с раздельными шинами данных и команд (двухшинная, или

гарвардская архитектура). Данная архитектура получила свое название в связи с работами, проведенными в Гарвардском университете под руководством Ховарда Айкена. Архитектура предполагает наличие в системе отдельной памяти для данных и отдельной памяти для команд (рис. 1.16). Обмен процессора с каждым из двух типов памяти происходит по своей шине.

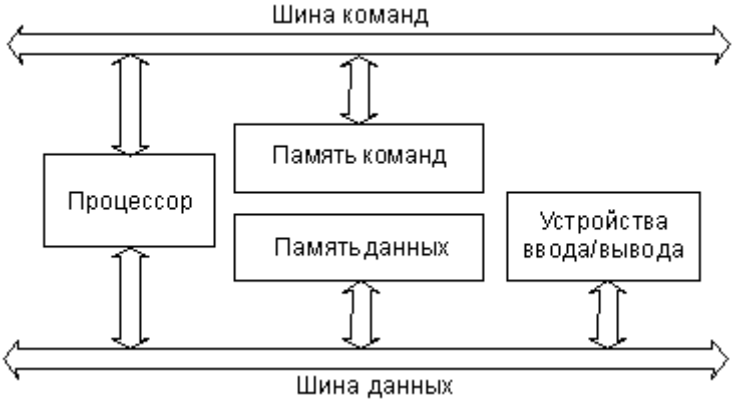


Рис. 1.16. Архитектура с раздельными шинами данных и команд

Рассмотрим некоторые достоинства и недостатки обоих архитектурных решений.

Архитектура с общей шиной (принстонская, фон-неймановская) проще, она не требует от процессора одновременного обслуживания двух шин, контроля обмена по двум шинам сразу. Наличие единой памяти данных и команд позволяет гибко распределять ее объем между кодами данных и команд. Например, в некоторых случаях нужна большая и сложная программа, а данных в памяти надо хранить не слишком много. В других случаях, наоборот, программа требуется простая, но необходимы большие объемы хранимых данных. Перераспределение памяти не вызывает никаких проблем, главное – чтобы программа и данные вместе помещались в памяти системы. Как правило, в системах с такой архитектурой память бывает довольно большого объема (до десятков и сотен мегабайт). Это позволяет решать самые сложные задачи.

Архитектура с раздельными шинами данных и команд сложнее, она заставляет процессор работать одновременно с двумя потоками кодов, обслуживать обмен по двум шинам одновременно. Программа может размещаться только в памяти команд, данные – только в памяти данных. Такая узкая специализация ограничивает круг задач, решаемых системой, так как не дает возможности гибкого перераспределения памяти. Память данных и память команд в этом случае имеют не слишком большой объем, поэтому применение систем с данной архитектурой ограничивается обычно не слишком сложными задачами.

В чем же преимущество архитектуры с двумя шинами (гарвардской)? В первую очередь, в быстродействии.

Дело в том, что при единственной шине команд и данных процессор вынужден по одной этой шине принимать данные (из памяти или устройства

ввода/вывода) и передавать данные (в память или в устройство ввода/вывода), а также читать команды из памяти. Естественно, одновременно эти пересылки кодов по магистрали происходить не могут, они должны производиться по очереди. Современные процессоры способны совместить во времени выполнение команд и проведение циклов обмена по системной шине. Использование конвейерных технологий и быстрой кэш-памяти позволяет им ускорить процесс взаимодействия со сравнительно медленной системной памятью. Повышение тактовой частоты и совершенствование структуры процессоров дают возможность сократить время выполнения команд. Но дальнейшее увеличение быстродействия системы возможно только при совмещении пересылки данных и чтения команд, то есть при переходе к архитектуре с двумя шинами.

В случае двухшинной архитектуры обмен по обеим шинам может быть независимым, параллельным во времени. Соответственно, структуры шин (количество разрядов кода адреса и кода данных, порядок и скорость обмена информацией и т.д.) могут быть выбраны оптимально для той задачи, которая решается каждой шиной. Поэтому при прочих равных условиях переход на двухшинную архитектуру ускоряет работу микропроцессорной системы, хотя и требует дополнительных затрат на аппаратуру, усложнения структуры процессора. Память данных в этом случае имеет свое распределение адресов, а память команд – свое.

Проще всего преимущества двухшинной архитектуры реализуются внутри одной микросхемы. В этом случае можно также существенно уменьшить влияние недостатков этой архитектуры. Поэтому основное ее применение – в микроконтроллерах, от которых не требуется решения слишком сложных задач, но зато необходимо максимальное быстродействие при заданной тактовой частоте.

Модифицированная гарвардская архитектура допускает обмен содержимым между памятью программ и памятью данных, что расширяет возможности устройства. Один из вариантов такого обмена – разделяемая память, т. е. память команд (ПК) и память данных (ПД) имеют некоторое общее пространство адресов, для выполнения программ, отражающих архитектуру фон-Неймана, что дает определенную гибкость при разработке прикладного программного обеспечения (рис. 1.17).

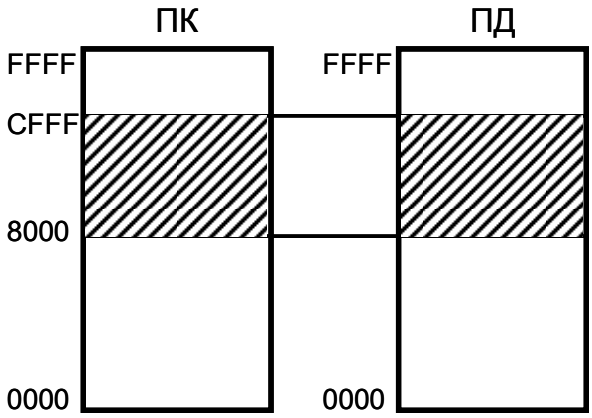


Рис. 1.17. Объединение ПК и ПД

Различают следующие способы организации вычислительного процесса:

- один поток команд – один поток данных (Simple Instruction - Simple Data, SISD) – характерно для традиционной фон-неймановской архитектуры (иногда вместо Simple пишут Single);
- один поток команд – множественный поток данных (Simple Instruction - Multiple Data, SIMD);
- множественный поток команд – один поток данных (Multiple Instruction - Simple Data, MISD);
- множественный поток команд – множественный поток данных (Multiple Instruction - Multiple Data, MIMD).

1.2. Классификация микропроцессоров

Микропроцессор (МП) является сложным устройством универсальной электронной системы, реализующим принцип программного управления. Микропроцессор представляет собой процессор в понимании фон Неймана в однокристальном исполнении с высокой степенью интеграции, что существенно снижает стоимость и габариты, повышает надежность системы управления в целом.

Бурное развитие техники микропроцессорных систем привело к расширению типов МП и требований, предъявляемых к ним. Сформировалось несколько типов микропроцессорных систем и микропроцессоров, различающихся назначением, мощностью, универсальностью, быстродействием и структурными отличиями.

По назначению МП можно разделить на *универсальные* и *специализированные*.

Универсальные МП характеризуются алгоритмическим универсальным набором команд, с помощью которого можно реализовать практически любой алгоритм, и используются для решения широкого круга задач. Производительность таких МП ввиду избыточности набора команд и универсальности не высока и слабо зависит от специфики решаемой задачи,

однако, цена МП из-за огромной популярности и высоких тиражей производства достаточно быстро снижается.

Специализированные МП предназначены для решения какой-то отдельной задачи или группы близких задач, определенного круга задач, их особенность – малая потребляемая мощность, компактность, быстродействие, надежность. Система команд таких МП имеет минимально необходимый набор, позволяющий эффективно решать поставленные задачи. Это обеспечивает более простую архитектуру и высокое быстродействие. Область применения – управляющие микроконтроллеры для работы во встроенных системах реального времени, специализированные устройства для сложной обработки видео-, аудио- информации, математической обработки и т.п.

Четкую границу в оценке быстродействия между универсальными и специализированным МП иногда провести довольно сложно. Быстродействие всех типов микропроцессоров постоянно растет, и нередки ситуации, когда новый универсальный оказывается быстрее, например, устаревшего специализированного МП.

Приведем несколько примеров специализированных процессоров.

Коммуникационные процессоры – имеют специальную поддержку аппаратных интерфейсов и протоколов коммуникационных систем [2]:

- сетевые – поддерживают распространенные сетевые и современные периферийные интерфейсы: Ethernet, HDLC, X.25, T1, ATM, USB, High Speed UART и др. Выпускаются фирмами Motorola (MC683xx, MPC8xx), AMD (Am186CC);
- модемные – поддерживают протоколы серий V2x, V3x передачи данных по синхронным и асинхронным модемным каналам.

Ярким примером специализированного МП может служить (микро)процессор для цифровой обработки сигналов (ПЦОС) – ***DSP*** (Digital Signal Processor) [3]. Особенностью работы DSP является поточный характер обработки больших объемов данных в реальном масштабе времени и, как правило, с интенсивным обменом данных с внешними устройствами. DSP реализуется на основе так называемой базовой архитектуры (DSP Basic Architecture), совмещающей ограниченный набор инструкций и высокую производительность, что позволяет им с наибольшей эффективностью осуществлять алгоритмы рекуррентной обработки данных. В то же время, можно выделить ряд типовых, наиболее распространенных задач, решаемых с применением ПЦОС:

- фильтрация сигналов;
- цифровая связь и кодирование;
- свертка двух сигналов (смешение сигналов);
- вычисление значений авто и кросс- корреляционной функции двух сигналов;
- масштабирование, нормализация или преобразование сигналов;
- прямое и обратное Фурье-преобразование, в т. ч. БПФ.

DSP построены, как правило, по Гарвардской архитектуре, имеющей отдельные шины адреса/данных для памяти команд и памяти данных, что позволяет им с помощью одной команды осуществлять доступ к различным видам памяти и производить несколько операций над данными. Основной особенностью DSP является то, что кроме обыкновенного АЛУ, которое присутствует во всех процессорах, они имеют еще несколько вычислительных устройств [4]. К числу таких устройств в первую очередь относится т. н. “умножитель-аккумулятор” MAU (Multiple-Accumulator Unit), способный с помощью одной команды умножить два многоразрядных числа и сложить результат удвоенной разрядности с результатом предыдущей команды. Подобная операция “умножения-сложения” используется во всех рекуррентных алгоритмах. Наличие MAU в сочетании с вышеуказанными особенностями организации шин процессора позволяет DSP за одну команду полностью выполнить один шаг рекуррентного алгоритма и подготовить исходные данные для следующего шага. Другим дополнительным вычислительным устройством является “многоразрядный регистр сдвига” S (Shifter), способный выполнять операции сдвига над числами, разрядность которых превышает разрядность АЛУ. Совместная работа этих вычислительных устройств позволяет достичь на выполнении рекуррентных алгоритмов вычислительной производительности, несравнимой с любыми другими процессорами.

Основные производители DSP – Texas Instruments (семейство TMS320, 32-разрядные с плавающей точкой, производительность до 60 MFLOPS), Analog Devices (семейство ADSP21xx, 16-разрядные DSP с фиксированной точкой, производительность до 30 MIPS), Motorolla (семейства DSP56xxx, DSP 96xxx).

Медийные процессоры – ориентированные на обработку видео- и звуковой информации с поддержкой мультимедийной обработки (медиапроцессоры) – имеют аппаратные блоки и развитую систему команд для обработки и передачи аудио и графических данных, видеоизображений. Примеры: медиапроцессор систем связи компании MicroUtility, универсальные медиапроцессоры Trimedia (Philips), MediaGX (Ciryx).

По сложности архитектуры и набору команд можно разделить МП с CISC- и RISC-архитектурой.

RISC – Reduced (Restricted) Instruction Set Computer - процессоры (компьютеры) с сокращенной системой команд. Эти процессоры обычно имеют набор однородных регистров универсального назначения, причем их число может быть большим. Система команд отличается относительной простотой, коды инструкций имеют четкую структуру, как правило, с фиксированной длиной. В результате аппаратная реализация такой архитектуры позволяет с небольшими затратами декодировать и выполнять эту инструкцию за минимальное (в пределе 1) число тактов синхронизации. Определенные преимущества дает и унификация регистров. Характерным

примером является RISC-микропроцессор в составе микроконтроллера фирмы Microchip (семейство PICxx).

Микроконтроллеры семейств PIC (Peripheral Interface Controller) компании Microchip объединяют все передовые технологии: электрически программируемые пользователем ППЗУ, минимальное энергопотребление, высокую производительность, хорошо развитую RISC-архитектуру, функциональную законченность и минимальные размеры. Первые микроконтроллеры компании Microchip PIC16C5x появились в конце 1980-х годов и благодаря своей высокой производительности и низкой стоимости составили серьезную конкуренцию производившимся в то время 8-разрядным МК с CISC-архитектурой.

Высокая скорость выполнения команд в PIC-контроллерах достигается за счет использования двухшинной гарвардской архитектуры вместо традиционной одношинной фон-неймановской. Гарвардская архитектура основывается на наборе регистров с разделенными шинами и адресными пространствами для команд и данных. Высокая производительность обусловлена и ограниченным числом команд: в зависимости от семейства PIC-контроллеров количество обрабатываемых команд колеблется от 33 до 75. Все ресурсы микроконтроллера, такие как порты ввода/вывода, ячейки памяти и таймер, стековая память представляют собой физически реализованные аппаратные регистры.

CISC – Complete Instruction Set Computer - процессоры (компьютеры) с полным набором инструкций, к которым относится и семейство x86. Состав и назначение их регистров существенно неоднородны, широкий набор команд усложняет декодирование инструкций, на что расходуются аппаратные ресурсы. Возрастает число тактов, необходимое для выполнения инструкций.

Процессоры x86 имеют самую сложную в мире систему команд. Хорошо ли это, вопрос спорный, но груз совместимости с программным обеспечением для IBM PC, имеющим уже 20-летнюю историю, не позволяет расставаться с этим "наследием тяжелого прошлого". В процессорах семейства x86, начиная с 486, применяется ***комбинированная архитектура*** - CISC-процессор имеет RISC-ядро.

По количеству кристаллов (СБИС) МП могут быть выполнены на одном кристалле и многокристальные.

Однокристальные МП в своей структуре могут содержать одно ядро или несколько ядер, в последнем случае появляется возможность распараллелить вычислительный процесс и организовать конвейеризацию при обработке данных, что повышает производительность МП.

Многокристальные МП реализуются на базе на 2-х и более СБИС. Этот подход целесообразен при требовании к определенному числу разрядов обрабатываемых данных в МС, не кратной степени двойки, тогда из определенного количества секций МП набирается нужная разрядность МС.

Либо же разделение может быть по функциональному назначению СБИС, тогда строится система с заданным набором функций.

По способу управления МП могут иметь аппаратно-реализованное устройство управления (жесткая логика) и микропрограммное управление. Первые – имеют неизменяемую систему команд, как правило, это универсальные однокристальные МП.

МП с ***микропрограммным управлением*** позволяют задавать собственный уникальный набор команд, оптимизированный для решения конкретной задачи. Чаще всего устройство управления такого МП содержит в своей структуре программируемую логическую матрицу, являющуюся своего рода памятью микрокоманд МП.

По возможности прерывания МП могут не иметь прерываний, иметь одно единственное прерывание, или может быть реализовано многоуровневое прерывание. Многоуровневые прерывания используются в системах реального времени, прерывания разделяются по приоритету и допустимо вложение прерываний.

Кроме вышеперечисленных признаков классификации МП существуют и другие: - по виду архитектуры; - по производительности; - по разрядности; - по наличию и объему КЭШ-памяти и др.

Микропроцессорные системы, ядром которых является МП, в своей структуре имеют различное количество БИС и СБИС, которые выполняют функции ввода-вывода, контроля шин, контроля прерываний и ПДП, тактирования, питания и т.д., должны иметь согласованные характеристики по быстродействию, задержкам распространения сигнала, электрическим параметрам, эксплуатационным характеристикам. Набор БИС и СБИС, предназначенный для совместного использования при построении МС называется ***микропроцессорный комплект***. Микропроцессорный комплект как правило выполнен на единой технологической базе, в одном конструктивном исполнении, с едиными параметрами источников питания и, чаще всего, выпускается под одной серией.

Классические микропроцессорные системы на базе микросхем процессоров и микропроцессорных комплектов выпускаются сейчас довольно редко, в первую очередь, из-за сложности процесса разработки и отладки этих систем. Данный тип микропроцессорных систем выбирают в основном тогда, когда микроконтроллеры не могут обеспечить требуемых характеристик.

Широкое распространение в современных микропроцессорных системах в последнее время получили ***микроконтроллеры*** (МК), представляющие собой универсальные устройства на одном кристалле, которые практически всегда используются не сами по себе, а в составе более сложных устройств управления, например программируемых логических контроллеров (ПЛК). Архитектура МК схожа с архитектурой микроЭВМ в интегральном исполнении со своим центральным процессором, памятью, таймерами, интерфейсами ввода-вывода и др., однако, системная шина МК

чаще всего недоступна пользователю, так как скрыта внутри микросхемы. Устройства на микроконтроллерах обычно предназначены для решения одной задачи либо группы близких задач, их структура проста и оптимизирована под максимальное быстродействие. Возможности подключения к микроконтроллеру дополнительных узлов и внешних устройств (например, большой памяти, средств ввода/вывода) ограничены. В большинстве случаев выполняемые программы хранятся в постоянной памяти и не меняются. Конструктивно контроллеры выпускаются в одноплатном варианте.

Вопросы для повторения

1. Что подразумевается под «жесткой логикой» при реализации МС?
2. Что такое программное управление МС?
3. Какие блоки входят в обобщенную структуру МП?
4. Что предполагает шинная организация МС?
5. Какие существуют режимы работы МП?
6. Охарактеризовать режим программного обмена информацией.
7. Охарактеризовать режим обработки прерываний и ПДП.
8. Указать достоинства и недостатки фон-неймановской и гарвардской архитектуры МС.
9. Указать особенности универсальных и специализированных МП.
10. Привести пример применения специализированного МП.
11. Что такое микропрограммное управление МП?
12. В чем отличие CISC и RISC-архитектуры МП?
13. Чем отличается «многокристальный» и «многоядерный» МП?
14. Что такое микропроцессорный комплект?
15. В чем состоят особенности микроконтроллера?

Тема 2. Организация обмена информацией в микропроцессорной системе

Цели и задачи изучения темы

Основной целью изучения данной темы является знакомство с принципами обмена информацией в МС, особенности аппаратной реализации обмена.

Задачи:

1. Ознакомиться со схемотехническими решениями двунаправленного обмена данными по шинам, адресации памяти в внешних устройств.
2. Изучить циклы программного обмена для «быстрых» и «медленных» устройств МС.
3. Изучить циклы обмена с внешними устройствами в режиме прерываний и ПДП.

2.1. Аппаратная поддержка шинного обмена

В системную магистраль (системную шину) микропроцессорной системы входит три основные информационные шины: адреса, данных и управления [1].

Шина данных – это основная шина, ради которой и создается вся система. Количество ее разрядов (линий связи) определяет скорость и эффективность информационного обмена, а также максимально возможное количество команд.

Шина данных всегда двунаправленная, так как предполагает передачу информации в обоих направлениях. Наиболее часто встречающийся тип выходного каскада для линий этой шины – выход с тремя состояниями.

Обычно шина данных имеет 8, 16, 32 или 64 разряда. Понятно, что за один цикл обмена по 64-разрядной шине может передаваться 8 байт информации, а по 8-разрядной – только один байт. Разрядность шины данных определяет и разрядность всей магистрали. Например, когда говорят о 32-разрядной системной магистрали, подразумевается, что она имеет 32-разрядную шину данных.

Шина адреса – вторая по важности шина, которая определяет максимально возможную сложность микропроцессорной системы, то есть допустимый объем памяти и, следовательно, максимально возможный размер программы и максимально возможный объем запоминаемых данных. Количество адресов, обеспечиваемых шиной адреса, определяется как 2^n, где n – количество разрядов. Например, 16-разрядная шина адреса обеспечивает 65 536 адресов. Разрядность шины адреса обычно кратна 4 и может достигать 32 и даже 64. Шина адреса может быть однонаправленной (когда магистралью всегда управляет только процессор) или двунаправленной (когда процессор может временно передавать управление магистралью

другому устройству, например контроллеру ПДП). Наиболее часто используются типы выходных каскадов с тремя состояниями или обычные ТТЛ (с двумя состояниями).

Как в шине данных, так и в шине адреса может использоваться положительная логика или отрицательная логика. При положительной логике высокий уровень напряжения электрического тока соответствует логической единице на соответствующей линии связи, низкий – логическому нулю. При отрицательной логике – наоборот. В большинстве случаев уровни сигналов на шинах – ТТЛ.

2.1.1. Организация шинного обмена данными

Шина данных должна обеспечивать обмен информацией между микропроцессором, памятью и различными устройствами ввода-вывода, причем, процесс обмена должен быть полностью управляемым для исключения неоднозначных ситуаций и коллизий. Рассмотрим, каким образом аппаратно обеспечивается работа шины данных при обмене информацией между несколькими устройствами.

Возможность совместного использования шины данных многочисленными устройствами обеспечивает шинный формирователь с тремя состояниями [5]. Поясним его работу сначала на примере простой однопроводной шины, поскольку принципы ее организации остаются неизменными при любом количестве проводов в шине (типичная шина данных состоит из восьми линий).

Шина с тремя состояниями похожа по своему назначению на групповую телефонную линию, к которой подсоединяются несколько абонентов. С ней могут быть связаны многочисленные источники и приемники сообщений. Для примера на рис. 2.1 показана шина цифровой схемы с тремя источниками (их роль выполняют формирователи с тремя состояниями – 3С) и двумя приемниками (обычные вентили).

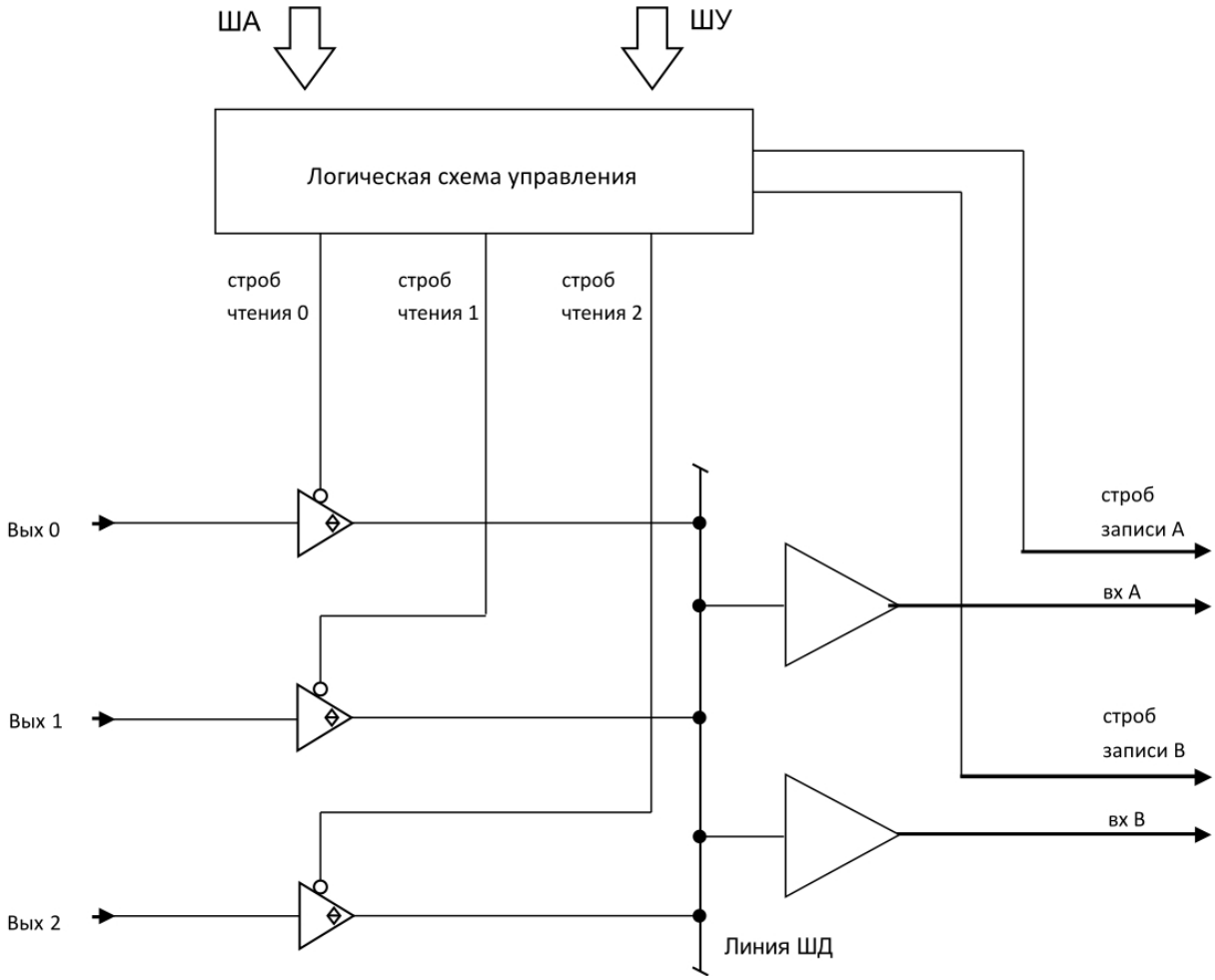


Рис. 2.1. Однопроводная шина с 3С и подключенные к ней 3 источника и 2 приемника

Логическая схема управления в каждый момент времени инициирует работу только одного формирователя (источника). Если бы в этот момент были разблокированы сразу несколько источников, информация, передаваемая по шине, потеряла бы смысл. Как только какой-то из формирователей разблокируется, данные с его выхода попадают на информационную шину. Все остальные аналогичные устройства в это время заблокированы. Их выходы находятся в состоянии, характеризуемом высоким полным сопротивлением (z-состоянии), и поэтому они не оказывают никакого влияния на логическое состояние шины. Логическая схема управления в соответствии с адресом устройства (ША) и сигналами ШУ выбирает устройство, которое должно участвовать в передаче данных.

К шине может подсоединяться большое количество приемников, а так как единственная их функция это прием данных, в один и тот же момент времени может быть разблокирован не один, а несколько приемников сообщений, хотя, как правило, данные, передаваемые по шине, предназначаются конкретно лишь одному из них. Поэтому логическое управляющее устройство формирует специальные сигналы (стробы данных), сообщающие выбранным приемникам, что данные на шине предназначаются

именно им. Строб данных может быть использован, например, для синхронизации операций пересылки данных с шины в триггер. К входам логического устройства управления подключаются адресная и управляющая шины, идущие от микропроцессора. Рассмотренные устройства характеризуются мультиплексированием и однонаправленным действием: это либо передатчики, либо приемники, но ни одно из устройств не может работать и на прием, и на передачу.

Устройства двунаправленного действия, подключенные к однопроводной шине, показаны на рис. 2.2. Каждый источник/приемник получает два управляющих сигнала: разрешающий для отпирания выхода формирователя с тремя состояниями (строб чтения) и строб данных для открывания входа (строб записи). Дешифратор адреса и логическая схема, собранная на логических элементах ИЛИ, формируют стробы записи и чтения, которые поступают на устройства двунаправленного действия «А» и «В».

Микропроцессор формирует два основных управляющих сигнала READ (строб чтения) и WRITE (строб записи), которые входят в состав ШУ. Если сигнал READ имеет низкий уровень, это означает, что выполняется операция считывания и микропроцессор предписывает адресуемому устройству вывести данные на шину данных. Низкий уровень сигнала WRITE означает, что выполняется операция записи, а данные на ту же шину пересылает микропроцессор, предписывающий адресуемому устройству принять эти данные.

Главное отличие управляющей шины от адресной шины и шины данных заключается в том, что каждая ее линия выполняет свою собственную единственную функцию. Что же касается адресной шины и шины данных, то в них по любой из линий передается один и тот же тип информации (1 бит адреса или данных).

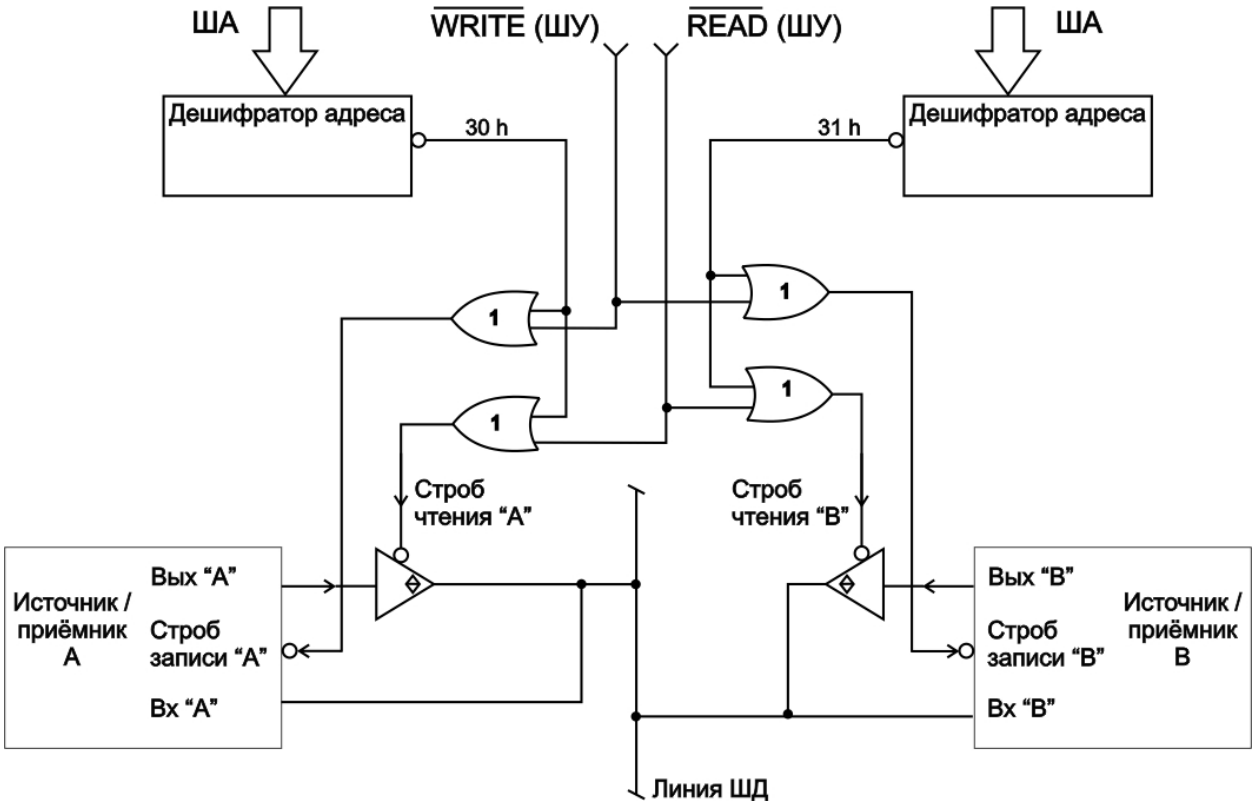


Рис. 2.2. Источники/приемники двунаправленного действия, подключенные к однопроводной шине

Чтобы понять, как протекает этот процесс, предположим, что устройство «А» на рис. 2.2 должно передать данные микропроцессору, а, затем, устройство «В» должно принять данные от микропроцессора. Логическая схема управления согласно выставленному микропроцессором адресу на ША и сигналу READ шины управления, устанавливает разрешающий сигнал – строб чтения – на выходе устройства «А» в состояние «истина» (разрешение), а аналогичный сигнал на выходе устройства «В» – в состояние «ложь» (запрет). Данные от устройства «А» попадают на ШД, откуда производится считывание микропроцессором. Затем, после обработки данных в микропроцессоре, они выставляются на ШД для записи в устройство «В». Спустя определенный промежуток времени, достаточный для того, чтобы данные поступили на вход устройства «В», посылается импульс в его линию строба данных – строб записи «В» – на основе заранее выставленного микропроцессором адреса на ША и сигнала WRITE шины управления. По стробу записи устройство «В» считывает данные с ШД. К шине может быть подключено множество других устройств, но до тех пор, пока разрешающие сигналы на их выходах будут находиться в состоянии «ложь», они не будут участвовать в процессе передачи данных.

2.1.2. Дешифрация адреса и выборка устройств МС

Выше было показано, как шина данных используется совместно многими устройствами в процессе информационного обмена. Для его организации необходимо реализовать выбор микропроцессором конкретного

устройства для работы с шиной данных. Эта функция выбора обеспечивается совместно адресной и управляющей шинами [5].

Поскольку адресная шина однонаправленная, принцип ее функционирования проще, чем соответствующий механизм шины данных. При использовании адресной шины каждая ячейка памяти (и порт ввода/вывода) имеет свой собственный однозначный адрес. Поэтому перед началом любой пересылки информации с использованием шины данных микропроцессор должен указать конкретный адрес пересылки. Этот адрес точно определяет ячейку памяти (или порт ввода/вывода), к которой микропроцессору необходимо получить доступ. Таким способом микропроцессор может выбирать любой блок системы, с которым он должен взаимодействовать. Например, если адресная шина микропроцессора состоит из 16 линий и обеспечивает прямую адресацию 2^{16} = 65 536 ячеек памяти и портов ввода/вывода. Эти линии имеют обозначения A_0, A_1, A_2, … A_{15}, причем A0 соответствует самому младшему разряду.

Частью логического устройства управления является дешифратор адреса. Он формирует сигналы выборки устройства, когда на адресной шине появляется определенный адрес (или область адресов). Для примера на рис. 2.3 показан дешифратор адреса, который в шестнадцатеричной системе представляется числом 30 *h*, а в двоичной – 0011 0000 *b*. Выход этого дешифратора находится в состоянии «истина» (логический нуль) только тогда, когда именно данный адрес поступает на адресную шину. Этот выходной сигнал используется затем для разблокирования порта, которому присвоен адрес 30 *h*.

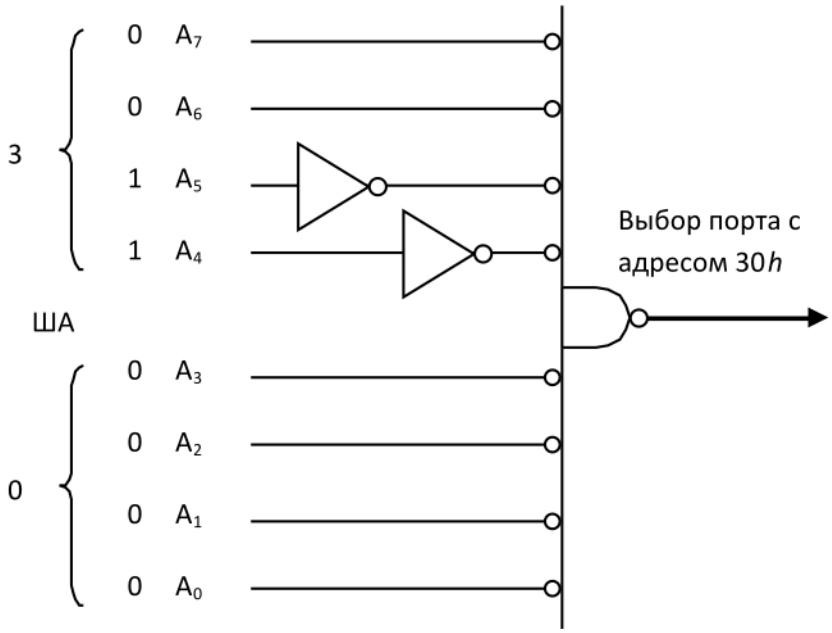


Рис. 2.3. Дешифратор адреса, настроенный на управление портом с адресом 30 *h*

Предположим, что дешифратор адреса нужен для управления не одним, а несколькими портами одного устройства ввода-вывода. Для этого могли бы быть использованы отдельные дешифраторы адреса, например

такие, как на рис. 2.3, однако существует и более простой способ декодирования. На рис. 2.4 показан дешифратор адреса, который формирует сигналы выборки для адресов 31, 31, 32, 33 (шестнадцатиричное). Для этих адресов в 8-разрядном адресе изменяются только два младших разряда: A_0 и A_1. Поэтому 6 других разрядов могут быть декодированы общей схемой, подобно той, которая представлена на рис. 2.3. Выход этой схемы передается на вход мультиплексора, а, затем с инверсией на тот его выход, номер которого закодирован в двух младших адресных разрядах. Мультиплексор в этой схеме выступает в роли дешифратора, так как активный уровень (логический «0»), свидетельствующий о выборе нужного адреса, будет появляться на одном из выходов микросхемы, номер которого определяется сигналами A_0 и A_1, только тогда, когда на ее входе «D» присутствует логическая «1».

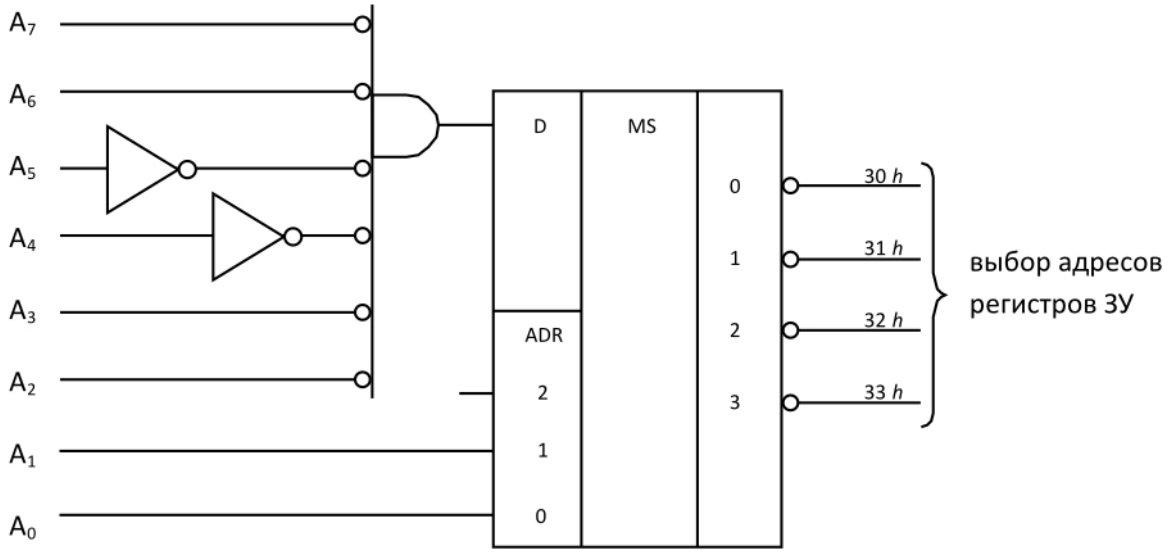


Рис. 2.4. Дешифратор адреса, настроенный на управление портами с адресами 30…33 *h*

В большинстве микропроцессорных систем ЗУ реализуется на нескольких микросхемах ЗУ. В этом случае потребуется более сложный дешифратор адреса. Предположим, например, что две микросхемы статического ОЗУ емкостью по 256 байт имеют области адресов 3000…31FF *h* (рис. 2.5). Адресные линии должны в этом случае указывать выбираемую микросхему ОЗУ и адресуемое в ней слово. Восемь младших разрядов определяют ячейку внутри микросхемы ОЗУ, а восемь старших – адресуемую микросхему. Дешифратор адреса такой же, как и на рис. 2.4, только подключен он не к младшим 8-ми разрядам ША, а к ее старшим разрядам. Таким образом, выходы дешифратора выбирают не одну ячейку (байт) памяти, а сразу 256 ячеек (байт): 30xx *h* (адреса с 3000 *h* по 30FF *h*) и 31xx *h* (адреса с 3100 *h* по 31FF *h*). Если микросхема ОЗУ не выбрана, т.е. на ее входе CS – неактивный уровень (логическая «1»), то выходы ее линий данных D I/O отключаются от ШД микропроцессорной системы – переводятся в z-состояние.

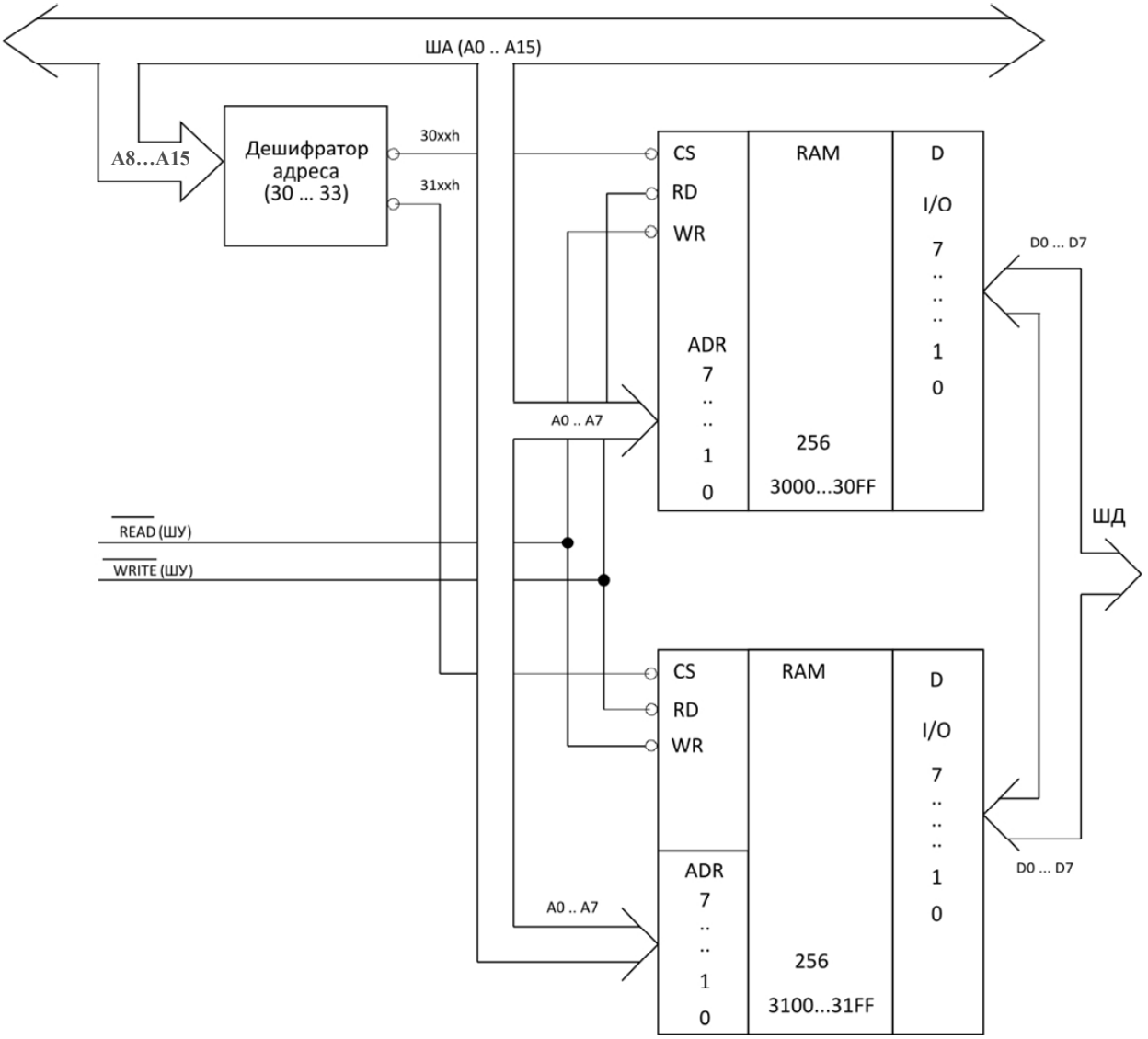


Рис. 2.5. Схема декодирования адресов ОЗУ

Обратите внимание, что при декодировании только разряд A_8 определяет отличие одной выбираемой микросхемы от другой. Причина, по которой изменяется только этот разряд адреса, заключается в том, что для декодирования 512 адресов (2^9= 512) требуется ровно 9 разрядов, а разряд A_8 как раз и является девятым по счету. Поэтому значения бит этого разряда («0» или «1») и определяют один из двух блоков, содержащих по 256 адресов каждый (в соответствии с емкостью каждого ОЗУ).

Восемь младших разрядов адреса A_0…A_7 поступают непосредственно в адресные линии обоих микросхем ОЗУ, поскольку эти разряды определяют конкретную ячейку внутри микросхемы. Затем дешифратор адреса анализирует восемь старших разрядов адреса и формирует сигнал выбора микросхемы. Два самых младших разряда (из той половины кода адреса, которая содержит старшие разряды) A_8 и A_9 служит двоичными входами дешифратора. Остальные старшие разряды адреса используются для разблокирования дешифратора только тогда, когда все соответствующие им сигналы имеют низкий уровень.

Сигналы READ и WRITE управляют режимом записи и считывания информации ОЗУ. В режиме записи линии данных D I/O находятся в состояние ввода информации в ОЗУ, при этом выходные усилители этих линий переводятся в высокоимпедансное состояние (z-состояние). В режиме чтения выходные усилители линий данных D I/O подключаются к ШД и происходит передача данных из ОЗУ.

На рис. 2.6. приведен пример подключения микропроцессорных устройств к системной магистрали: ОЗУ, ПЗУ и дискретных портов ввода-вывода, причем младшие ячейки памяти адресного пространства принадлежат ПЗУ, так как при перезагрузке МП первая выполняемая команда берется из нулевого адреса, а программа начальной загрузки МС всегда размещается в ПЗУ.

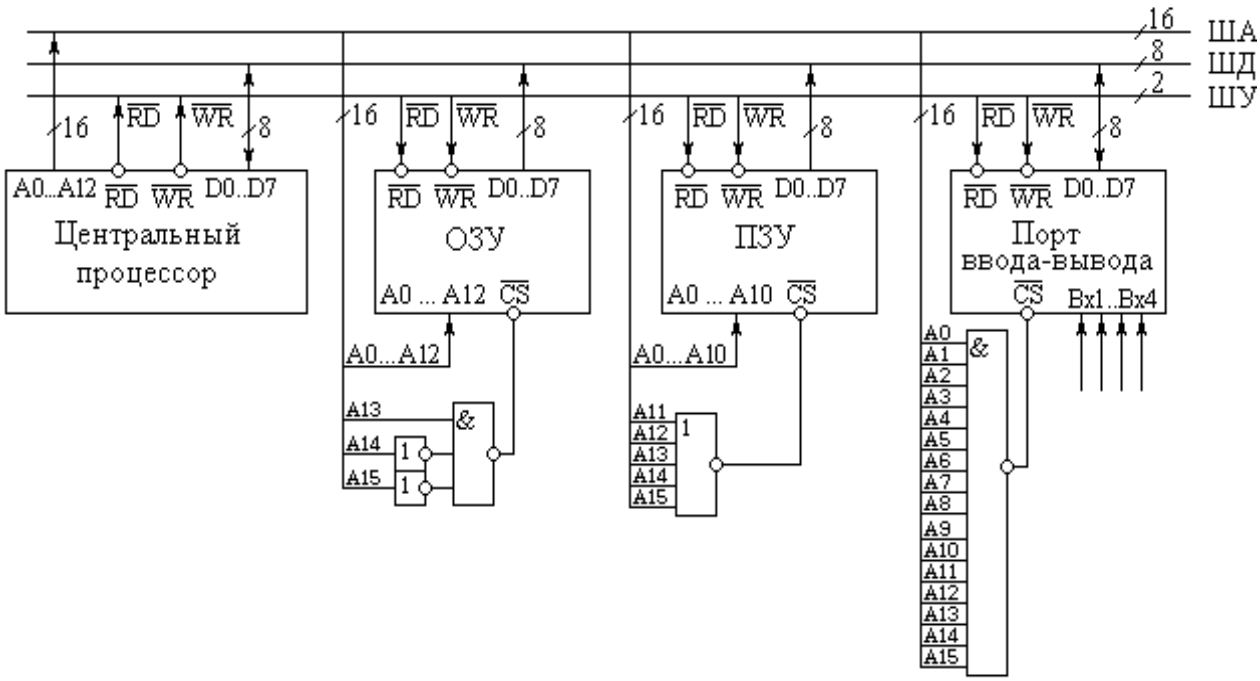


Рис. 2.6. Пример подключения микропроцессорных устройств к системной шине

Более старшие адреса памяти реализуются в ОЗУ, и, наконец, самые верхние адреса, как правило, отводятся под устройства ввода-вывода.

2.1.3. Прохождение сигналов по магистрали

При организации обмена по магистралям и шинам необходимо учитывать несколько важных моментов, связанных как с особенностью распространения сигналов по шинам, так и с самой природой шин [1]. В противном случае микропроцессорная система может попросту не работать или работать неустойчиво, хотя вся логика цифровых устройств, входящих в систему, будет спроектирована безошибочно.

В случае, когда системная шина (магистраль) микропроцессорной системы является внешней, а не скрыта внутри микросхемы, необходимо учитывать особенности распространения сигналов по длинным линиям. Хотя

в большинстве случаев длина магистрали не слишком велика, не превышает 1–2 десятков сантиметров, это все равно оказывает большое влияние на синхронизацию обмена.

На прохождение сигналов по магистрали влияют следующие факторы:

- конечная величина задержки распространения сигналов по линиям магистрали;
- различие задержек распространения сигналов по разным линиям шины;
-
- неодновременное выставление сигналов на линии шины;
- искажение фронтов сигналов, проходящих по линиям магистрали;
- отражение сигналов от концов линий связи (рис. 2.7).

Рис. 2.7. Прохождение сигналов по шине

Для учета всех этих факторов разработчики стандартных магистралей обмена и стандартных протоколов обмена всегда закладывают необходимые задержки между сигналами, участвующими в обмене. Кроме того, задержки между сигналами выбираются таким образом, чтобы устройство, которому адресован тот или иной сигнал, имело достаточно времени для его обработки. Если разрабатывается новая магистраль, все это тоже надо учитывать. Поэтому пытаться "модернизировать" какой-то стандартный протокол и ускорять обмен по магистрали путем уменьшения задержек, предусмотренных стандартом, очень опасно. Точно так же опасно, не изменяя протокола обмена, пытаться увеличить длину магистрали, увеличивая тем самым задержки распространения сигналов по линиям и шинам. Особенно чувствительны к такого рода "модернизациям" синхронные магистрали, в которых не предусмотрено обязательное подтверждение выполнения каждой операции.

Например, длительность фазы адреса в цикле обмена выбирается таким образом. В течение адресной фазы все сигналы всех разрядов кода адреса, пусть даже и сформированные процессором не одновременно, должны дойти до устройства-исполнителя по своим проводам шины. А устройство-исполнитель должно этот код адреса принять и обработать (то есть отличить свой адрес от чужого). Естественно, для гарантии в длительность адресной фазы еще добавляется небольшая дополнительная задержка.

Точно так же длительность фазы данных в цикле чтения должна выбираться такой, чтобы устройство-исполнитель успело получить строб чтения и выдать код читаемых данных на шину данных. Затем этот код должен успеть дойти до процессора и процессор должен успеть его прочитать. После чего процессор снимает сигнал строба чтения, этот задний фронт сигнала доходит с задержкой до устройства-исполнителя, которое также с задержкой снимает свой код данных. Аналогично и в цикле записи.

Для улучшения формы сигналов, распространяющихся по магистрали, иногда применяют оконечные согласователи (терминаторы) на концах линий магистрали. Особенно важно их применение в случае, когда допустимая длина магистрали превышает несколько метров. Например, в случае магистрали Q-bus применяются два типа согласователей: 120-омный и 250-омный (рис. 2.8). Подобные 120-омные согласователи или терминаторы используются и в магистральном последовательном интерфейсе RS-485, включаемые в цепь цифровой линии передачи данных на ее окончаниях, т.е. по краям линии – на крайних приемо-передатчиках (предельная длина передачи данных по RS-485 – 1200 м).

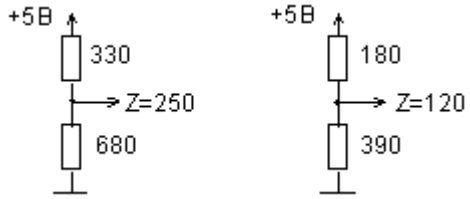


Рис. 2.8. Оконечные согласователи на магистрали Q-bus

Включение согласователей предъявляет дополнительные требования к нагрузочной способности передатчиков, работающих на линии магистрали. В магистрали ISA подобные согласователи не используются, хотя к некоторым линиям присоединены резисторы, соединенные другим своим выводом с шиной питания (прежде всего это линии, тип выходного каскада для которых - ОК).

В любом случае выходные каскады передатчиков, работающих на линии магистрали, должны обеспечивать высокие выходные токи, так как к магистрали может подключаться несколько устройств, каждое из которых потребляет входной ток. Типичные величины требуемых выходных токов магистральных передатчиков находятся в пределах 20–30 мА. В то же время входные токи магистральных приемников должны быть малыми, чтобы не перегружать передатчики. Типичные величины допустимых входных токов магистральных приемников лежат в пределах 0,2–0,8 мА.

2.2. Циклы обмена данными по магистралям МС

За более чем 30 лет, прошедших с момента появления первых микропроцессоров, были выработаны определенные правила обмена, которым следуют и разработчики новых микропроцессорных систем [1]. Правила эти не слишком сложны, но твердо знать и неукоснительно

соблюдать их для успешной работы необходимо. Как показала практика, принципы организации обмена по шинам гораздо важнее, чем особенности конкретных микропроцессоров. Стандартные системные магистрали живут гораздо дольше, чем тот или иной процессор. Разработчики новых процессоров ориентируются на уже существующие стандарты магистрали. Более того, некоторые системы на основе совершенно разных процессоров используют одну и ту же системную магистраль. То есть магистраль оказывается самым главным системообразующим фактором в микропроцессорных системах.

Обмен информацией в микропроцессорных системах происходит в циклах обмена информацией. Под циклом обмена информацией понимается временной интервал, в течение которого происходит выполнение одной элементарной операции обмена по шине. Например, пересылка кода данных из процессора в память или же пересылка кода данных из устройства ввода/вывода в процессор. В пределах одного цикла также может передаваться и несколько кодов данных, даже целый массив данных, но это встречается реже.

Циклы обмена информацией делятся на два основных типа:

- цикл записи (вывода), в котором процессор записывает (выводит) информацию;
- цикл чтения (ввода), в котором процессор читает (вводит) информацию.

В некоторых микропроцессорных системах существует также цикл "чтение-модификация-запись" или же "ввод-пауза-вывод". В этих циклах процессор сначала читает информацию из памяти или устройства ввода/вывода, затем как-то преобразует ее и снова записывает по тому же адресу. Например, процессор может прочитать код из ячейки памяти, увеличить его на единицу и снова записать в эту же ячейку памяти. Наличие или отсутствие данного типа цикла связано с особенностями используемого процессора.

Особое место занимают циклы прямого доступа к памяти (если режим ПДП в системе предусмотрен) и циклы запроса и предоставления прерывания (если прерывания в системе есть).

Во время каждого цикла устройства, участвующие в обмене информацией, передают друг другу информационные и управляющие сигналы в строго установленном порядке или, как еще говорят, в соответствии с принятым протоколом обмена информацией.

Длительность цикла обмена может быть постоянной или переменной, но она всегда включает в себя несколько периодов сигнала тактовой частоты системы. То есть даже в идеальном случае частота чтения информации процессором и частота записи информации оказываются в несколько раз меньше тактовой частоты системы.

Чтение кодов команд из памяти системы также производится с помощью циклов чтения. Поэтому в случае одношинной архитектуры на

системной магистрали чередуются циклы чтения команд и циклы пересылки (чтения и записи) данных, но протоколы обмена остаются неизменными независимо от того, что передается – данные или команды. В случае двухшинной архитектуры циклы чтения команд и записи или чтения данных разделяются по разным шинам и могут выполняться одновременно.

Таким образом, основные функции любой микропроцессорной системы следующие:

- выборка (чтение) выполняемых команд;
- ввод (чтение) данных из памяти или устройства ввода/вывода;
- вывод (запись) данных в память или в устройство ввода/вывода;
- обработка данных (операндов), в том числе арифметические операции над ними;
- адресация памяти, то есть задание адреса памяти, с которым будет производиться обмен;
- обработка прерываний и режима прямого доступа.

2.2.1. Мультиплексирование шин и виды обмена данными в МС

Мультиплексирование шин адреса и данных часто применяется для снижения общего количества линий связи магистрали [1]. То есть одни и те же линии связи используются в разные моменты времени для передачи как адреса, так и данных (в начале цикла – адрес, в конце цикла – данные). Для фиксации этих моментов (стробирования) служат специальные сигналы на шине управления. Понятно, что мультиплексированная шина адреса/данных обеспечивает меньшую скорость обмена, требует более длительного цикла обмена (рис. 2.9). По типу шины адреса и шины данных все магистрали также делятся на ***мультиплексированные*** и ***немультиплексированные***.

Рис. 2.9. Мультиплексирование шин адреса и данных

В некоторых мультиплексированных магистралях после одного кода адреса передается несколько кодов данных (массив данных). Это позволяет существенно повысить быстродействие магистрали. Иногда в магистралях применяется частичное мультиплексирование, то есть часть разрядов данных передается по немультиплексированным линиям, а другая часть – по мультиплексированным линиям. Так, например, по мультиплексированной шине адреса могут передаваться в начале старшая часть адреса, а затем, по некоторым линиям шины – передаваться младшая часть составного адреса.

Шина управления – это вспомогательная шина, управляющие сигналы на которой определяют тип текущего цикла и фиксируют моменты времени, соответствующие разным частям или стадиям цикла. Кроме того,

управляющие сигналы обеспечивают согласование работы процессора (или другого хозяина магистрали, задатчика, master) с работой памяти или устройства ввода/вывода (устройства-исполнителя, slave). Управляющие сигналы также обслуживают запрос и предоставление прерываний, запрос и предоставление прямого доступа.

Сигналы шины управления могут передаваться как в положительной логике (реже), так и в отрицательной логике (чаще). Линии шины управления могут быть как однонаправленными, так и двунаправленными. Типы выходных каскадов могут быть самыми разными: с двумя состояниями (для однонаправленных линий), с тремя состояниями (для двунаправленных линий), с открытым коллектором (для двунаправленных и мультиплексированных линий).

Самые главные управляющие сигналы – это стробы обмена, то есть сигналы, формируемые процессором и определяющие моменты времени, в которые производится пересылка данных по шине данных, обмен данными. Чаще всего в магистрали используются два различных строба обмена:

- ***строб записи*** (вывода), который определяет момент времени, когда устройство-исполнитель может принимать данные, выставленные процессором на шину данных;
- ***строб чтения*** (ввода), который определяет момент времени, когда устройство-исполнитель должно выдать на шину данных код данных, который будет прочитан процессором.

При этом большое значение имеет то, как процессор заканчивает обмен в пределах цикла, в какой момент он снимает свой строб обмена. Возможны два пути решения (рис. 2.10):

- при ***синхронном обмене*** процессор заканчивает обмен данными самостоятельно, через раз и навсегда установленный временной интервал выдержки ($t_{выд}$), то есть без учета состояния устройства-исполнителя;
- при ***асинхронном обмене*** процессор заканчивает обмен только тогда, когда устройство-исполнитель подтверждает выполнение операции специальным сигналом (так называемый режим handshake – рукопожатие).

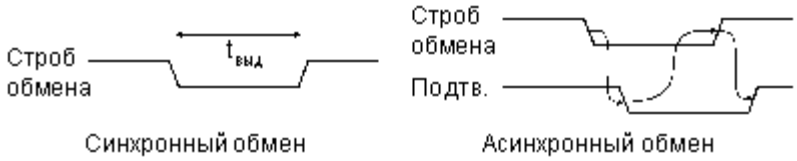


Рис. 2.10. Синхронный асинхронный обмен

Достоинства синхронного обмена – более простой протокол обмена, меньшее количество управляющих сигналов. Недостатки – отсутствие гарантии, что исполнитель выполнил требуемую операцию, а также высокие требования к быстродействию исполнителя.

Достоинства асинхронного обмена – более надежная пересылка данных, возможность работы с самыми разными по быстродействию исполнителями. Недостаток – необходимость формирования сигнала подтверждения всеми исполнителями, то есть дополнительные аппаратурные затраты.

Какой тип обмена быстрее, синхронный или асинхронный? Ответ на этот вопрос неоднозначен. С одной стороны, при асинхронном обмене требуется какое-то время на выработку, передачу дополнительного сигнала и на его обработку процессором. С другой стороны, при синхронном обмене приходится искусственно увеличивать длительность строба обмена для соответствия требованиям большего числа исполнителей, чтобы они успевали обмениваться информацией в темпе процессора. Поэтому иногда в магистрали предусматривают возможность как синхронного, так и асинхронного обмена, причем синхронный обмен является основным и довольно быстрым, а асинхронный применяется только для медленных исполнителей.

По используемому типу обмена магистрали микропроцессорных систем также делятся на синхронные и асинхронные.

2.2.2. Циклы программного обмена

Рассмотрим для примера два довольно типичных случая программного обмена по магистрали микропроцессорной системы [1]. С целью облегчения понимания принципов обмена информацией ниже будут приведены упрощенные диаграммы основных циклов.

Первый пример – это обмен по мультиплексированной асинхронной магистрали Q-bus, предложенной фирмой DEC и широко применявшейся в микрокомпьютерах и промышленных контроллерах. Упрощенные временные диаграммы циклов чтения (ввода) и записи (вывода) по этой магистрали приведены на рис. 2.11 и 2.12.

Отметим, что в дальнейшем тексте знак "минус" перед названием сигнала говорит о том, что активный уровень сигнала низкий, пассивный – высокий, то есть сигнал отрицательный. Если минуса перед названием сигнала нет, то сигнал положительный, его низкий уровень пассивный, а высокий – активный.

На шине адреса/данных (AD) в начале цикла обмена (в фазе адреса) процессор (задатчик) выставляет код адреса. На этой шине используется отрицательная логика. Средний уровень сигналов на шине AD обозначает, что состояния сигналов на шине в данные временные интервалы не важны. Для стробирования адреса используется отрицательный синхросигнал - SYNC, выставляемый также процессором. Его передний (отрицательный) фронт соответствует действительности кода адреса на шине AD. Фаза адреса одинакова в обоих циклах записи и чтения.

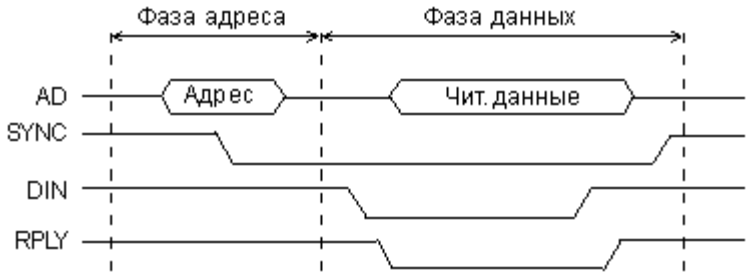


Рис. 2.11. Цикл чтения на магистрали Q-bus

Получив (распознав) свой код адреса, устройство ввода/вывода или память (исполнитель) готовится к проведению обмена. Через некоторое время после начала (отрицательного фронта) сигнала -SYNC процессор снимает адрес и начинает фазу данных.

В фазе данных цикла чтения (рис. 2.11) процессор выставляет сигнал строба чтения данных -DIN, в ответ на который устройство, к которому обращается процессор (исполнитель), должно выставить свой код данных (читаемые данные). Одновременно это устройство должно подтвердить выполнение операции сигналом подтверждения обмена -RPLY.

Для сигнала -RPLY используется тип выходного каскада ОК, чтобы не было конфликтов между устройствами-исполнителями. Процессор, получив сигнал -RPLY, заканчивает цикл обмена. Для этого он снимает сигнал -DIN и сигнал -SYNC. Устройство-исполнитель в ответ на снятие сигнала -DIN должно снять код данных с шины AD и закончить сигнал подтверждения -RPLY. После этого процессор снимает сигнал -SYNC.

Рис. 2.12. Цикл записи на магистрали Q-bus

В фазе данных цикла записи (рис. 2.12) процессор выставляет на шину AD код записываемых данных и сопровождает его отрицательным сигналом строба записи данных -DOUT. Устройство-исполнитель должно по этому сигналу принять данные от процессора и сформировать сигнал подтверждения обмена -RPLY. Процессор, получив сигнал -RPLY, заканчивает цикл обмена. Для этого он снимает код данных с шины AD и сигнал -DOUT. Устройство-исполнитель в ответ на снятие сигнала -DOUT должно закончить сигнал подтверждения -RPLY. После этого процессор снимает сигнал -SYNC.

На рассматриваемой магистрали адрес передается синхронно (без подтверждения его получения исполнителем), а данные передаются

асинхронно, с обязательным подтверждением их выдачи или приема исполнителем. Отсутствие сигнала подтверждения -RPLY в течение заданного времени воспринимается процессором как аварийная ситуация. В принципе возможна и асинхронная передача адреса, что увеличивает надежность обмена, хотя может снижать его скорость.

Помимо циклов чтения и записи на магистрали Q-bus используются также и циклы типа "ввод-пауза-вывод" ("чтение-модификация-запись"). Упрощенная временная диаграмма этого цикла представлена на рис. 2.13.

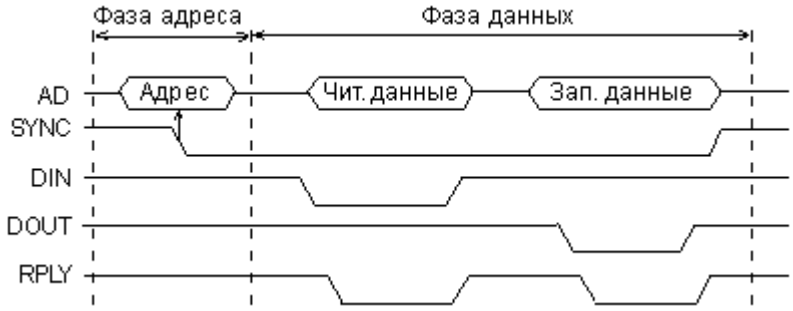


Рис. 2.13. Цикл "ввод-пауза-вывод" на магистрали Q-bus

В этом цикле адресная фаза производится точно так же, как и в циклах чтения (ввода) и записи (вывода). Но в фазе данных процессор производит сначала чтение из заданного в адресной фазе адреса, а потом запись в тот же самый адрес. Для чтения используется строб чтения -DIN, а для записи – строб записи -DOUT. В ответ на сигнал -DIN устройство-исполнитель выдает свои данные на шину AD, а по сигналу -DOUT – принимает данные с шины AD. Как и в циклах чтения и записи, устройство-исполнитель подтверждает выполнение каждой операции сигналом подтверждения -RPLY. Понятно, что цикл "ввод-пауза-вывод" требует больше времени, чем каждый из циклов чтения или записи, но меньше времени, чем два последовательно произведенных цикла чтения и записи (так как для него нужна только одна адресная фаза). Сигнал -SYNC вырабатывается процессором в начале цикла "ввод-пауза-вывод" и держится до окончания всего цикла.

В качестве второго примера рассмотрим циклы обмена на синхронной немультиплексированной магистрали ISA (Industrial Standard Architecture), предложенной фирмой IBM и широко используемой в персональных компьютерах. Упрощенные циклы записи в устройство ввода/вывода и чтения из устройства ввода/вывода приведены на рис. 2.14 и 2.15.

Оба цикла начинаются с выставления процессором (задатчиком) кода адреса на шину адреса SA (логика на этой шине положительная). Адрес остается на шине SA до конца цикла. Фаза адреса, одинаковая для обоих циклов, заканчивается с началом строба обмена данными -IOR или -IOW. В течение фазы адреса устройство-исполнитель должно принять код адреса и распознать или не распознать его. Если адрес распознан, исполнитель готовится к обмену.

В фазе данных цикла чтения (рис. 2.14) процессор выставляет отрицательный сигнал чтения данных из устройства ввода/вывода -IOR. В

ответ на него устройство-исполнитель должно выдать на шину данных SD свой код данных (читаемые данные). Логика на шине данных положительная. Через установленное время строб обмена -IOR снимается процессором, после чего снимается также и код адреса с шины SA. Цикл заканчивается без учета быстродействия исполнителя.

Рис. 2.14. Цикл чтения из УВВ на магистрали ISA

Но так происходит только в случае основного, синхронного обмена. Кроме него на магистрали ISA также предусмотрена возможность асинхронного обмена. Для этого применяется сигнал готовности канала (магистрали) I/O CH RDY. Тип выходного каскада для данного сигнала – ОК, для предотвращения конфликтов между устройствами-исполнителями. При синхронном обмене сигнал I/O CH RDY всегда положительный. Но медленное устройство-исполнитель, не успевающее работать в темпе процессора, может этот сигнал снять, то есть сделать нулевым сразу после начала строба обмена. Тогда процессор до того момента, пока сигнал I/O CH RDY не станет снова положительным, приостанавливает завершение цикла, продлевает строб обмена. Конечно, слишком большая длительность этого сигнала рассматривается как аварийная ситуация. Для простоты понимания можно считать, что устройство-исполнитель формирует в данном случае отрицательный сигнал неготовности завершить обмен. На время этого сигнала обмен на магистрали приостанавливается.

Принципиальное отличие асинхронного обмена по магистрали ISA от асинхронного обмена по магистрали Q-bus состоит в следующем. Если в случае Q-bus сигнал подтверждения обязателен, и его должен формировать каждый исполнитель, то в случае ISA сигнал о неготовности исполнитель может не формировать, если он успевает работать в темпе процессора. Зато в случае Q-bus к концу цикла обмена процессор всегда уверен, что устройство-исполнитель выполнило требуемую операцию, а в случае ISA такой уверенности нет.

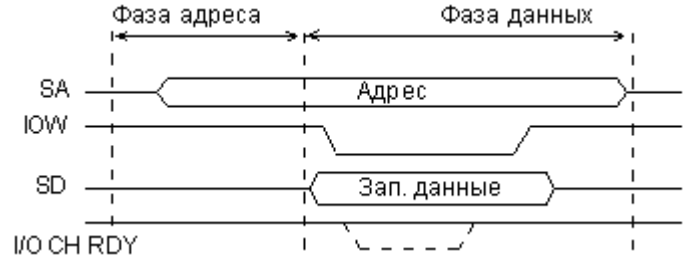


Рис. 2.15. Цикл записи в УВВ на магистрали ISA

В фазе данных цикла записи по магистрали ISA (рис. 2.15) процессор выставляет на шину данных SD код записываемых данных и сопровождает их стробом записи данных в устройство ввода/вывода -IOW. Получив этот сигнал, устройство-исполнитель должно принять с шины SD код записываемых данных. Если оно не успевает сделать это в темпе процессора, то оно может снять на нужное время сигнал I/O CH RDY после получения переднего фронта сигнала -IOW. Тогда процессор приостановит окончание цикла записи.

Рассмотренные примеры, конечно, не раскрывают всех тонкостей обмена по упомянутым магистралям. Они всего лишь иллюстрируют главные принципы обмена по ним.

2.2.3. Циклы обмена по прерываниям

Циклы обмена в режиме прерываний строятся по тем же принципам, что и циклы программного обмена, но имеют ряд специфических особенностей [1]. Прерывания в микропроцессорных системах бывают двух основных типов:

- векторные прерывания, которые требуют проведения цикла чтения по магистрали;
- радиальные прерывания, которые не требуют никакого цикла обмена по магистрали.

Дело в том, что прерываний в микропроцессорной системе обычно бывает много. Поэтому процессору необходима информация о номере (или, как еще говорят, об адресе вектора) конкретного прерывания. Эта информация может быть передана процессору двумя путями.

При векторном прерывании код номера прерывания передается процессору тем устройством ввода/вывода, которое данное прерывание запросило. Для этого процессор проводит цикл чтения по магистрали, и по шине данных получает код номера прерывания. Шина адреса в данном цикле обычно не используется, так как устройство, запросившее прерывание, и так знает, что процессор будет обращаться именно к нему. В этом случае в магистрали достаточно всего одной линии запроса прерывания для всех устройств ввода/вывода. Так организованы прерывания, например, в магистрали Q-bus.

Схема распространения сигналов, участвующих в прерываниях на магистрали Q-bus, показана на рис. 2.16. Упрощенная временная диаграмма цикла запроса и предоставления магистрали представлена на рис. 2.17.

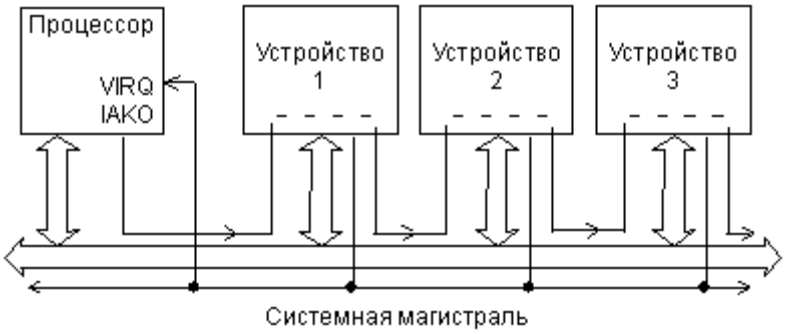


Рис. 2.16. Сигналы запроса и предоставления прерывания в магистрали Q-bus

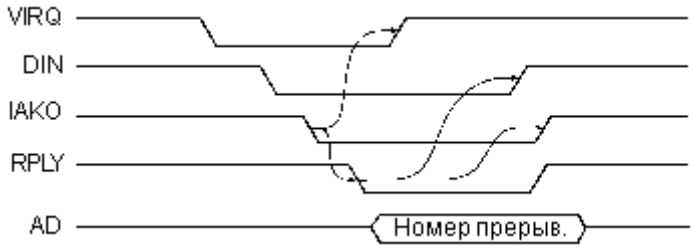


Рис. 2.17. Цикл запроса/предоставления векторного прерывания на магистрали Q-bus

Запрос прерывания осуществляется отрицательным сигналом -VIRQ, который может формироваться каждым из устройств, запрашивающих прерывание. Тип выходного каскада для этого сигнала – ОК, чтобы избежать конфликтов между запрашивающими прерывания устройствами. Получив сигнал -VIRQ, процессор предоставляет прерывание (закончив предварительно выполнение текущей команды). Для этого он выставляет сигнал чтения данных -DIN и сигнал предоставления прерывания IAKO. Этот сигнал IAKO последовательно проходит через все устройства, которые могут запрашивать прерывания. Если устройство запросило прерывание, то оно не пропускает через себя этот сигнал. В результате получается, что если прерывания одновременно запросили два или более устройств, то сигнал предоставления прерывания получит только одно устройство, а именно то, которое ближе к процессору. Такой механизм разрешения конфликтов называется иногда географическим приоритетом (или цепочечным приоритетом, Daisy Chain). Получив сигнал IAKO, устройство, запросившее прерывание, должно снять свой сигнал -VIRQ.

Затем процессор проводит цикл безадресного чтения номера прерывания. В ответ на полученные сигналы -DIN и IAKO устройство, которому предоставлено прерывание, должно выдать на шину адреса/данных AD код номера прерывания (адреса вектора прерывания) и выставить сигнал подтверждения -RPLY. Процессор читает код номера прерывания и заканчивает цикл безадресного чтения снятием сигналов -DIN и IAKO.

При радиальном прерывании в магистрали имеется столько линий запроса прерывания, сколько всего может быть разных прерываний. То есть

каждое устройство ввода/вывода, желающее использовать прерывание, подает сигнал запроса прерывания по своей отдельной линии. Процессор узнает о номере прерывания по номеру линии, по которой пришел сигнал запроса прерывания. Никаких циклов обмена по магистрали при этом не требуется. В случае радиальных прерываний в систему обычно включается дополнительная микросхема контроллера прерываний, обрабатывающая сигналы запросов прерываний. Именно так организованы прерывания, например, в магистрали ISA.

Упрощенная структура связей между устройствами, участвующими в обмене по прерываниям, на магистрали ISA показана на рис. 2.18. Процессор общается с контроллером прерываний как по магистрали (чтобы задать ему режимы работы), так и вне магистрали (при обработке запросов на прерывание). Сигналы запросов прерываний IRQ распределяются между всеми устройствами магистрали. На каждую линию IRQ приходится одно устройство. Тип выходного каскада для этих линий – 2С, так как конфликты здесь не предусмотрены. Запросом прерывания является передний, положительный фронт сигнала IRQ. При одновременном поступлении сигналов IRQ от нескольких устройств порядок их обслуживания определяется контроллером прерываний.

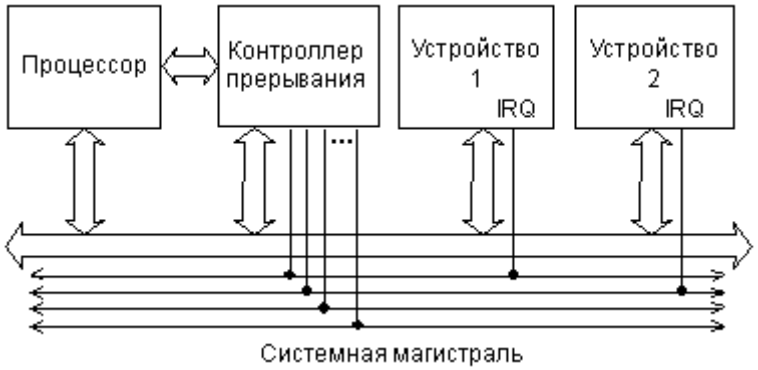


Рис. 2.18. Структура связей для организации радиальных прерываний на магистрали ISA

Какой тип прерываний лучше – векторный или радиальный? Векторные прерывания обеспечивают системе большую гибкость, в системе их может быть очень много. Но зато они требуют дополнительных аппаратурных узлов во всех устройствах, запрашивающих прерывания, для обслуживания циклов безадресного чтения.

Радиальных прерываний в системе обычно не очень много (от 1 до 16). При этом типе прерываний, как правило, требуется введение в систему специального контроллера прерываний. Каждое радиальное прерывание требует введения дополнительной линии в шину управления системной магистрали. Но работать с радиальными прерываниями проще, так как все сводится только к выработке единственного сигнала IRQ, и никаких циклов обмена по магистрали не требуется.

2.2.4. Циклы обмена в режиме ПДП

Циклы обмена в режиме прямого доступа к памяти выполняются по тем же правилам, что и циклы программного обмена, и циклы предоставления прерываний [1].

Прежде чем начать обмен в режиме ПДП, устройство, которому необходим ПДП, должно запросить ПДП и получить его. Процедура запроса и предоставления ПДП очень похожа на процедуру запроса и предоставления прерывания. В обоих случаях устройство, требующее обслуживания, посылает сигнал запроса процессору. Однако в случае ПДП процессор обязательно должен предоставить ПДП запросившему устройству с помощью специальных сигналов, так как на время ПДП процессор отключается от магистрали.

На магистрали Q-bus запрос и предоставление ПДП организуются подобно запросу и предоставлению прерывания. Упрощенная структура связей устройств, участвующих в ПДП, показана на рис. 2.19. Временная диаграмма запроса/предоставления ПДП очень близка к временной диаграмме запроса/предоставления прерывания (см. рис. 2.17).

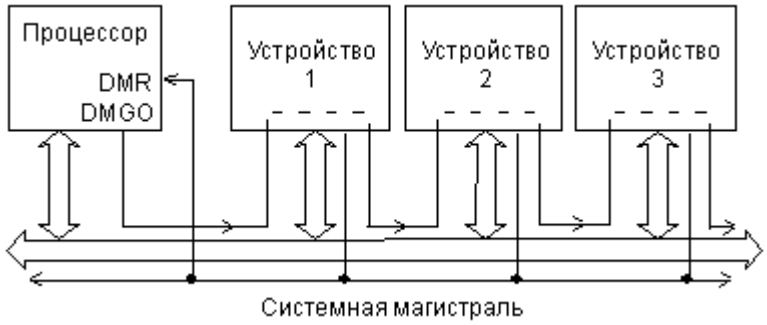


Рис. 2.19. Структура связей запроса/предоставления ПДП на магистрали Q-bus

Сигнал запроса ПДП, называемый -DMR, передается всеми устройствами, нуждающимися в ПДП, по одной линии магистрали. Тип выходного каскада на этой линии – ОК. Процессор, получив сигнал -DMR, выдает сигнал предоставления ПДП DMGO, аналогичный сигналу IAKO, и отключается от магистралей, контролируя при этом только сигнал -DMR. Сигнал DMGO также проходит через все устройства последовательно, в результате чего ПДП получает только то устройство, которое находится ближе к процессору (географический приоритет). А затем устройство, получившее ПДП, проводит циклы обмена по магистрали, аналогично циклам программного обмена. В циклах ПДП информация читается из памяти или записывается в память непосредственно устройством ввода/вывода. После окончания обмена в режиме прямого доступа устройство снимает сигнал –DMR и процессор восстанавливает контроль над магистралями.

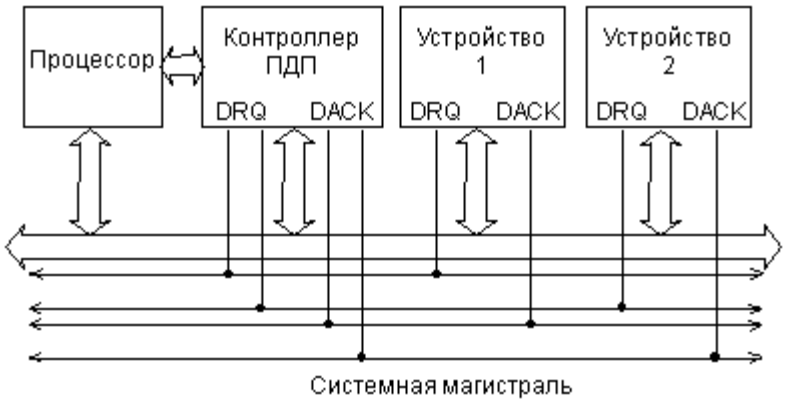


Рис. 2.20. Структура связей запроса/предоставления ПДП на магистрали ISA

На магистрали ISA запрос/предоставление ПДП очень напоминает организацию радиальных прерываний (рис. 2.20). Точно так же в системе существует контроллер ПДП, к которому сходятся сигналы запроса ПДП, называемые DRQ, и от которого расходятся сигналы предоставления ПДП, называемые -DACK. К каждому каналу ПДП (пара сигналов DRQ и -DACK) подключается только одно устройство, запрашивающее ПДП. Тип выходных каскадов для этих сигналов –2С. Устройство, нуждающееся в ПДП, посылает сигнал запроса DRQ и получает в ответ сигнал предоставления -DACK. После этого контроллер ПДП проводит циклы обмена по магистрали между устройством ввода/вывода и памятью.

Упрощенная временная диаграмма циклов ПДП на магистрали ISA показана на рис. 2.21.

Рис. 2.21. Цикл ПДП на магистрали ISA

На магистрали ISA используются раздельные стробы записи в память (-MEMW) и записи в устройства ввода/вывода (-IOW), а также раздельные стробы чтения из памяти (-MEMR) и чтения из устройств ввода/вывода (-IOR). Это позволяет за один цикл обмена ПДП читать информацию из памяти и записывать ее в устройство ввода/вывода или же читать информацию из устройства ввода/вывода и записывать ее в память. При этом на шине адреса выставляется адрес памяти, а адрес устройства ввода/вывода заменяется одним-единственным сигналом AEN. Естественно, в цикле обмена в режиме ПДП участвует только то устройство ввода/вывода, которое

предварительно запросило ПДП и которому ПДП было предоставлено. Поэтому никаких конфликтов между устройствами ввода/вывода из-за такой упрощенной адресации не возникает.

Вопросы для повторения

1. Каковы особенности аппаратной реализации двунаправленной шины?
2. Какие сигналы управления формируются в МС для обмена с УВВ на прием и передачу?
3. Как в МС строится дешифратор адреса для фиксированного адреса и для группы адресов?
4. Перечислить факторы, влияющие на прохождение сигналов по магистрали.
5. Зачем устанавливаются на магистраль терминаторы?
6. Какие основные циклы обмена данными используются в МС?
7. Чем отличается мультиплексированная шина от немультиплексированной?
8. Как организован синхронный и асинхронный обмен по магистрали?
9. Привести пример цикла программного асинхронного обмена.
10. Привести пример цикла программного синхронного обмена.
11. Как организован обмен при обработке по вектору прерываний и при радиальных прерываниях?
12. Указать особенности цикла обмена в режиме ПДП.

Тема 3. Основы функционирования МП

Цели и задачи изучения темы

Основной целью изучения данной темы является знакомство с основами функционирования ядра МС – микропроцессора, структурой памяти МС, методами адресации памяти, системой команд МП.

Задачи:

1. Ознакомиться со структурой памяти, делением на сегменты, назначением стека, специализированных регистров МП.
2. Изучить методы адресации, особенности сегментной адресации.
3. Рассмотреть классификацию команд микропроцессора, выделить общности системы команд различных типов МП.

3.1. Сегменты памяти и регистры МП

В пространстве памяти микропроцессорной системы обычно выделяются несколько особых областей, которые выполняют специальные функции [1].

Память ***программы начального запуска*** всегда выполняется на ПЗУ или флэш-памяти. Именно с этой области процессор начинает работу после включения питания и после сброса его с помощью сигнала RESET. Как правило, аппаратная реализации начального запуска в большинстве МС осуществляется с нулевого адреса.

Еще одна специальная область памяти микропроцессорной системы – это ***память устройств*** (устройств ввода/вывода), подключенных к системной шине. В составе МС, как правило, выделяются три специальные группы устройств ввода/вывода:

- устройства интерфейса пользователя (ввода информации пользователем и вывода информации для пользователя);
- устройства ввода/вывода для длительного хранения информации;
- таймерные устройства.

Микропроцессор имеет возможность обращаться к внутренней памяти устройств ввода/вывода или каких-то еще подключенных к системной шине устройств, как к своей собственной системной памяти. Обычно окно в пространстве памяти, выделяемое для этого, не слишком большое.

Существует два основных подхода к разделения адресов памяти и адресов устройств ввода/вывода:

- выделение в общем адресном пространстве системы специальной области адресов для устройств ввода/вывода;
- полное разделение адресных пространств памяти и устройств ввода/вывода.

Первый подход хорош тем, что при обращении к устройствам ввода/вывода процессор может использовать те же команды, которые служат для взаимодействия с памятью. Но адресное пространство памяти должно

быть уменьшено на величину адресного пространства устройств ввода/вывода. Например, при 16-разрядной шине адреса всего может быть 64К адресов. Из них 56К адресов отводится под адресное пространство памяти, а 8К старших адресов – под адресное пространство устройств ввода/вывода.

Преимущество второго подхода состоит в том, что память занимает все адресное пространство микропроцессорной системы. Для общения с устройствами ввода/вывода применяются специальные команды и специальные стробы обмена (записи в ВУ и чтения из ВУ) на магистрали – шине управления. Именно так сделано, например, в персональных компьютерах. Но возможности взаимодействия с устройствами ввода/вывода в данном случае существенно ограничены по сравнению с возможностями общения с памятью.

В остальной части памяти могут располагаться как данные, так и программы, конечно, в случае одношинной фон-неймановской архитектуры, в случае же гарвардской архитектуры память команд (или память программ) CSEG (Code Segment) и память данных DSEG (Data Segment) разделены и имеют свои собственные адресные пространства и способы доступа к ним. В некоторых МС это пространство памяти используется как единое целое, без всяких границ.

При построении МС на базе микроконтроллеров чаще пространство памяти делится на ***сегменты*** с программно изменяемым адресом начала сегмента и с установленным размером сегмента. Так, например, для семейства микроконтроллеров MSC-51 фирмы Intel сегмент DSEG в свою очередь делится на XDATA – область памяти с относительной-регистровой адресацией, IDATA – область с индексной адресацией и др. (методы адресации изложены в следующем разделе)

Оба подхода имеют свои плюсы и минусы. Например, использование сегментов позволяет защитить область программ или данных, но зато границы сегментов могут затруднять размещение больших программ и массивов данных.

Практически во всех современных микропроцессорах выделяют отдельную область памяти под так называемый “***стек***” (Stack), используемый, в общем случае, для передачи параметров процедурам и сохранения адресов возврата из них. Стек может быть расположен внутри микропроцессора в специально области памяти (в этом случае говорят о т.н. “аппаратном стеке”) или вне его – в оперативной памяти. В последнем случае стек занимает часть адресного пространства DSEG.

Стек предназначен для временного хранения данных в режиме LIFO (Last In – First Out). Особенность стека по сравнению с другой оперативной памятью – это заданный и неизменяемый способ адресации. При записи любого числа (кода) в стек число записывается по адресу, определяемому как содержимое регистра указателя стека, предварительно уменьшенное (декрементированное) на единицу (или на два, если 16-разрядные слова

расположены в памяти по четным адресам). При чтении из стека число читается из адреса, определяемого содержимым указателя стека, после чего это содержимое указателя стека увеличивается (инкрементируется) на единицу (или на два). В результате получается, что число, записанное последним, будет прочитано первым, а число, записанное первым, будет прочитано последним. Это и есть способ организации памяти – LIFO или памятью «патронного» типа (например, в магазине автомата патрон, установленный последним, будет извлечен первым).

Принцип действия стека показан на рис. 3.1 (адреса ячеек памяти выбраны условно).

Пусть, например, текущее состояние указателя стека 1000008, и в него надо записать два числа (слова). Первое слово будет записано по адресу 1000006 (перед записью указатель стека уменьшится на два). Второе – по адресу 1000004. После записи содержимое указателя стека станет – 1000004. Если затем прочитать из стека два слова, то первым будет прочитано слово из адреса 1000004, а после чтения указатель стека станет равным 1000006. Вторым будет прочитано слово из адреса 1000006, а указатель стека станет равным 1000008. Все вернулось к исходному состоянию. Первое записанное слово читается вторым, а второе – первым.

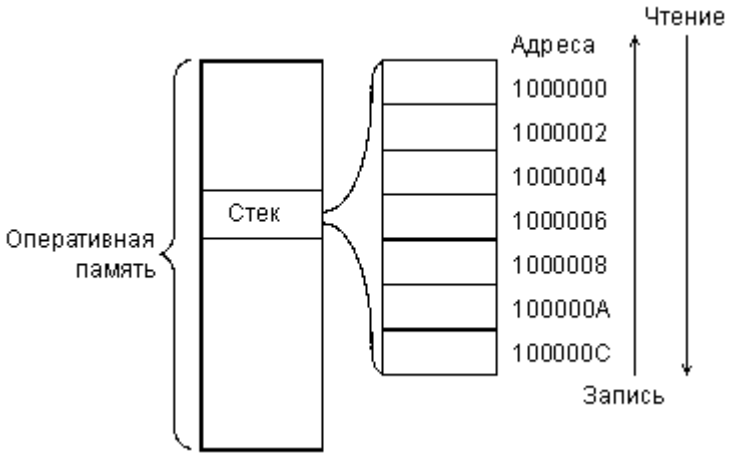


Рис. 3.1. Принцип работы стека

Необходимость такой адресации становится очевидной в случае многократно вложенных подпрограмм. Пусть, например, выполняется основная программа, и из нее вызывается подпрограмма 1. Если нам надо сохранить значения данных и внутренних регистров основной программы на время выполнения подпрограммы, мы перед вызовом подпрограммы сохраним их в стеке (запишем в стек), а после ее окончания извлечем (прочитаем) их из стека. Если же из подпрограммы 1 вызывается подпрограмма 2, то ту же самую операцию мы проделаем с данными и содержимым внутренних регистров подпрограммы 1. Понятно, что внутри подпрограммы 2 крайними в стеке (читаемыми в первую очередь) будут данные из подпрограммы 1, а данные из основной программы будут глубже. При этом в случае чтения из стека автоматически будет соблюдаться нужный

порядок читаемой информации. То же самое будет и в случае, когда таких уровней вложения подпрограмм гораздо больше. То есть то, что надо хранить подольше, прячется поглубже, а то, что скоро может потребоваться – с краю. Стек используется и для обработки многоуровневых прерываний при реализации систем реального времени.

Существует т.н. "***стековой архитектуре***", заключающаяся в передаче функций аккумулятора вершине стека. Стековая организация дает возможность использовать безадресные команды, код которых имеет наименьшую длину. Безадресные команды оперируют данными, находящимися на вершине стека и непосредственно под ней. При выполнении операции исходные операнды извлекаются из стека, а результат передается не вершину стека. Стековая архитектура обладает высокой вычислительной эффективностью. Существует специальный язык высокого уровня FORTH, построенный на основе безадресных команд. Такая архитектура используется в специализированных процессорах высокой производительности и, в частности в RISC-процессорах.

Следующая специальная область памяти DSEG – это ***таблица векторов прерываний***.

Вообще, понятие прерывания довольно многозначно. Под прерыванием в общем случае понимается не только обслуживание запроса внешнего устройства, но и любое нарушение последовательной работы процессора. Например, может быть предусмотрено прерывание по факту некорректного выполнения арифметической операции типа деления на ноль. Или же прерывание может быть программным, когда в программе используется команда перехода на какую-то подпрограмму, из которой затем последует возврат в основную программу. В последнем случае общее с истинным прерыванием только то, как осуществляется переход на подпрограмму и возврат из нее.

Любое прерывание обрабатывается через таблицу ***векторов (указателей) прерывани***й. В этой таблице в простейшем случае находятся адреса начала программ обработки прерываний, которые и называются векторами. Длина таблицы может быть довольно большой (до нескольких сот элементов). Обычно таблица векторов прерываний располагается в начале пространства памяти (в ячейках памяти с малыми адресами). Адрес каждого вектора (или адрес начального элемента каждого вектора) представляет собой номер прерывания.

В случае аппаратных прерываний номер прерывания или задается устройством, запросившим прерывание (при векторных прерываниях), или же задается номером линии запроса прерываний (при радиальных прерываниях). Процессор, получив аппаратное прерывание, заканчивает выполнение текущей команды и обращается к памяти в область таблицы векторов прерываний, в ту ее строку, которая определяется номером запрошенного прерывания. Затем процессор читает содержимое этой строки (код вектора прерывания) и переходит в адрес памяти, задаваемый этим

вектором. Начиная с этого адреса, в памяти должна располагаться программа обработки прерывания с данным номером. В конце программы обработки прерываний обязательно должна располагаться команда выхода из прерывания, выполнив которую, процессор возвращается к выполнению прерванной основной программы. Параметры процессора на время выполнения программы обработки прерывания сохраняются в стеке.

Пусть, например, процессор (рис. 3.2) выполнял основную программу и команду, находящуюся в адресе памяти 5000 (условно). В этот момент он получил запрос прерывания с номером (адресом вектора) 4. Процессор заканчивает выполнение команды из адреса 5000. Затем он сохраняет в стеке текущее значение счетчика команд (5001) и текущее значение регистра «слово состояния программы» PSW (будет рассмотрен ниже). После этого процессор читает из адреса 4 памяти код вектора прерывания. Пусть этот код равен 6000. Процессор переходит в адрес памяти 6000 и приступает к выполнению программы обработки прерывания, начинающейся с этого адреса. Пусть эта программа заканчивается в адресе 6100. Дойдя до этого адреса, процессор возвращается к выполнению прерванной программы. Для этого он извлекает из стека значение адреса (5001), на котором его прервали, и бывшее в тот момент PSW. Затем процессор читает команду из адреса 5001 и дальше последовательно выполняет команды основной программы.

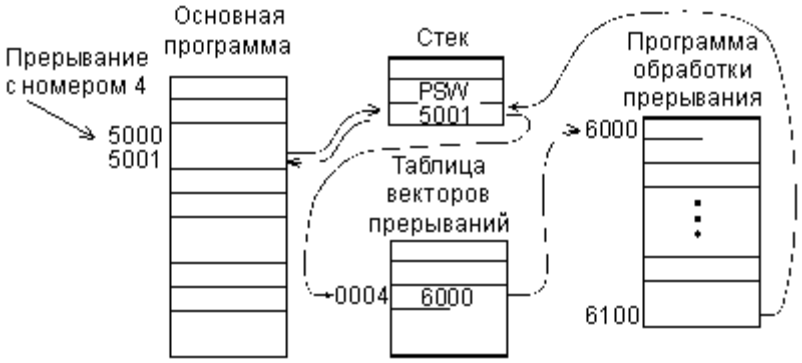


Рис. 3.2. Упрощенный алгоритм обработки прерывания

Прерывание в случае аварийной ситуации обрабатывается точно так же, только адрес вектора прерывания (номер строки в таблице векторов) жестко привязан к данному типу аварийной ситуации.

Программное прерывание тоже обслуживается через таблицу векторов прерываний, но номер прерывания указывается в составе команды, вызывающей прерывание. Такая сложная, на первый взгляд, организация прерываний позволяет программисту легко менять программы обработки прерываний, располагать их в любой области памяти, делать их любого размера и любой сложности.

Во время выполнения программы обработки прерывания может поступить новый запрос на прерывание. В этом случае он обрабатывается точно так же, как описано, но основной программой считается прерванная программа обработки предыдущего прерывания. Это называется

многократным вложением прерываний. Механизм стека позволяет без проблем обслуживать это многократное вложение, так как первым из стека извлекается тот код, который был сохранен последним, то есть возврат из обработки данного прерывания происходит в программу обработки предыдущего прерывания.

Отметим, что в более сложных случаях в таблице векторов прерываний могут находиться не адреса начала программ обработки прерываний, а так называемые дескрипторы (описатели) прерываний. Но конечным результатом обработки этого дескриптора все равно будет адрес начала программы обработки прерываний.

Кроме CSEG и DSEG практически все современные микропроцессоры имеют специально выделенное пространство данных небольшого объема, называемое набором ***программно-доступных регистров*** RSEG (Register Segment) [4]. В отличие от CSEG и DSEG регистры RSEG располагаются внутри ЦП или микропроцессора в непосредственной близости от его АЛУ, что обеспечивает быстрый физический доступ к информации, хранящейся в них. В них, как правило, хранятся промежуточные результаты вычислений, часто используемые ЦП. Область RSEG может быть полностью изолирована от пространства данных DSEG, может частично пересекаться с ней, и может адресно являться подчастью DSEG. Внутренняя логическая организация RSEG очень разнообразна и играет важную роль при классификации архитектур микропроцессоров.

Регистры микропроцессора функционально неоднородны: одни служат для хранения данных или адресной информации, другие - для управления работой ЦП. В соответствии с этим все регистры можно разделить на регистры данных, указатели и регистры специального назначения. Регистры данных участвуют в арифметических и логических операциях в качестве источников операндов и приемников результата, адресные регистры или указатели используются для вычисления адресов данных и команд, расположенных в основной памяти. ***Специальные регистры*** служат для индикации текущего состояния ЦП и управления работой его составных частей. Возможна архитектура, при которой одни и те же регистры используются для хранения как данных, так и адресной информации. Такие регистры называются ***регистрами общего назначения*** (РОН). Способы использования того или иного вида регистров определяют конкретные особенности архитектуры микропроцессора.

Среди регистров данных часто выделяют один регистр, называемый ***аккумулятором*** A (Accumulator), с которым связывают большинство команд арифметической и логической обработки данных. Это означает, что арифметические и логические команды используют в качестве одного из своих операндов содержимое аккумулятора и сохраняют в нем результат операции. Ссылка на него производится неявно с помощью кода операции. При этом нет необходимости в коде команды выделять специальную область для адресов операнда и результата. Такой тип архитектуры микропроцессора

называется аккумуляторным. К недостаткам такой архитектуры можно отнести относительно низкое быстродействие, объясняемое тем, что аккумулятор является ”узким местом”, в которое каждый раз необходимо сначала занести операнд перед выполнением операции. Примером такой архитектуры могут служить микроконтроллеры семейства MCS-51 фирмы Intel.

Другим примером организации регистров данных являются т.н. “***рабочие регистры***” R0, R1, и т.д. В этом случае операнды и результаты арифметических и логических операций могут храниться не в одном, а в нескольких регистрах, что расширяет возможности по манипуляции данными. В отличие от аккумулятора рабочие регистры адресуются явно в коде команды. Такой тип архитектуры микропроцессора называется регистровым. Примером такой организации могут служить микропроцессоры семейства x86 фирмы Intel. В ряде микропроцессоров, предназначенных для работы в реальном масштабе времени, предусмотрены не один, а несколько наборов рабочих регистров. В каждый момент времени доступен лишь один из наборов регистров, выбор которого осуществляется записью соответствующей информации в определенный служебный регистр. Примером таких устройств могут служить микроконтроллеры семейства MCS-48 фирмы Intel.

Архитектура, при которой процессор способен использовать в качестве адресов операндов и результатов операции ячейки основной памяти, называется архитектурой типа “***память - память***”. При этом исключаются временные затраты на перепись содержимого рабочих регистров при переходе от одной процедуры к другой. Однако при этом теряется быстрый доступ к промежуточным данным, т.к. они хранятся не во внутренних регистрах, а в DSEG. Решением этой проблемы может служить размещение части DSEG на одном кристалле с ЦП и использование в качестве рабочих областей этого внутреннего сегмента ОЗУ. Примером такой организации могут служить микроконтроллеры семейства MCS-96 фирмы Intel.

К специальным регистрам относятся и ***служебные регистры***, расположенные внутри микропроцессора и предназначенные для различных функций управления его работой и индикации состояния его составных частей. Их состав и организация зависят от конкретной архитектуры процессора и различаются в каждом конкретном случае. Наиболее часто встречающимися регистрами специальных функций являются адресные регистры (или указатели) – PC и SP – и слово состояния программы PSW.

Адресные регистры или указатели используются для реализации тех или иных методов адресации операндов, используемых в конкретных командах микропроцессора. Их конкретный набор и функции зависят от того, какие методы адресации реализованы в данной модели микропроцессора. Они обязательно выполняют две служебные функции:

- определяют адрес в памяти, где находится выполняемая в данный момент команда (функция ***счетчика команд*** или указателя команд).

Счетчик команд PC (Program Counter) в каждый конкретный момент времени содержит адрес команды, следующей в CSEG за той, которая в данный момент выполняется;

- определяют текущий адрес стека (функция ***указателя стека***). Указатель стека SP (Stack Pointer) хранит текущий адрес вершины стека.

В разных процессорах для каждой из этих функций может отводиться один или два внутренних регистра. Эти два указателя отличаются от других не только своим специфическим, служебным, системным назначением, но и особым способом изменения содержимого. Их содержимое программы могут менять только в случае крайней необходимости, так как любая ошибка при этом грозит нарушением работы компьютера, зависанием и порчей содержимого памяти.

Содержимое указателя (счетчика) команд PC изменяется следующим образом. В начале работы системы (при включении питания) в него заносится раз и навсегда установленное значение. Это первый адрес программы начального запуска, как правило, равный нулю. Затем после выборки из памяти каждой следующей команды значение указателя команд автоматически увеличивается (инкрементируется) на единицу (или на два в зависимости от формата команд и типа процессора). То есть следующая команда будет выбираться из следующего по порядку адреса памяти. При выполнении команд перехода, нарушающих последовательный перебор адресов памяти, в указатель команд принудительно записывается новое значение – новый адрес в памяти, начиная с которого адреса команд опять же будут перебираться последовательно. Такая же смена содержимого указателя команд производится при вызове подпрограммы и возврате из нее или при начале обработки прерывания и после его окончания.

Слово состояния программы (иногда называют: регистр состояния процессора) PSW (Program Status Word) содержит набор текущих признаков результата выполнения операции [4]. С каждым признаком результата связывается одноразрядная переменная-флажок, соответствующая определенному биту PSW. Рассмотрим типовые флажки-признаки на примере семейства МП x86 компании Intel (рис. 3.3):

- CF (Carry Flag) – флажок переноса из старшего разряда АЛУ. Равен 1, если в результате выполнения арифметической операции или операции сдвига произошел перенос из старшего разряда результата;

- PF (Parity Flag) – флажок четности. Равен 1, если число 1 в результате операции нечетно и наоборот;

- AF (Auxilinary Carry Flag) – флажок дополнительного переноса. Равен 1, если в результате выполнения арифметической операции или операции сдвига произошел перенос из младшей тетрады результата в старшую. Часто используется в двоично-десятичной арифметике;

- ZF (Zero Flag) – флажок признака нуля. Равен 1, если результат операции равен 0;

- SF (Sign Flag) – флажок знака результата. Дублирует знаковый разряд результата операции;

- TF (Trap Flag) – флажок пошагового режима (используется при отладке);

- IF (Interrupt Flag) – флажок разрешения прерывания. Индицирует, разрешены ли прерывания в системе;

- DF (Direction Flag) – флажок направления при строковых операциях;

- OF (Owerfow Flag) – флажок переполнения. Равен 1, если в результате выполнения арифметической операции произошло переполнение разрядной сетки результата.

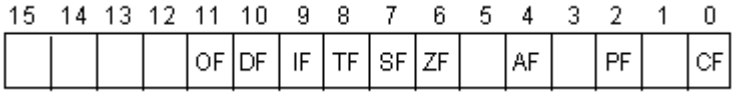


Рис. 3.3. Регистр состояния процессора x86

Биты регистра состояния устанавливаются или очищаются в зависимости от результата исполнения предыдущей команды и используются некоторыми командами процессора. Конкретные флаги используются программой для анализа результата предшествующей команды и принятия решения о дальнейшем ходе выполнения программы (реализация базовой структуры программирования – условия или «развилки»). Биты регистра состояния могут также устанавливаться и очищаться специальными командами процессора (о системе команд процессора будет рассказано в следующем разделе).

3.2. Адресация операндов

Система команд, выполняемых процессором, представляет собой нечто подобное таблице истинности логических элементов или таблице режимов работы более сложных логических микросхем [1]. То есть она определяет логику работы процессора и его реакцию на те или иные комбинации внешних событий.

Написание программ для микропроцессорной системы – важнейший и часто наиболее трудоемкий этап разработки такой системы. А для создания эффективных программ необходимо иметь хотя бы самое общее представление о системе команд используемого процессора. Компактные и быстрые программы и подпрограммы создаются на языке Ассемблер, использование которого без знания системы команд абсолютно невозможно, ведь язык Ассемблер представляет собой символьную запись цифровых кодов машинного языка, кодов команд процессора. Конечно, для разработки программного обеспечения существуют всевозможные программные средства. Пользоваться ими обычно можно и без знания системы команд процессора. Чаще всего применяются языки программирования высокого уровня, такие как Паскаль и Си. Однако знание системы команд и языка Ассемблер позволяет в несколько раз повысить эффективность некоторых наиболее важных частей программного обеспечения любой

микропроцессорной системы – от микроконтроллера до персонального компьютера.

Каждая команда, выбираемая (читаемая) из памяти процессором, определяет алгоритм поведения процессора на ближайшие несколько тактов. Код команды говорит о том, какую операцию предстоит выполнить процессору и с какими операндами (то есть кодами данных), где взять исходную информацию для выполнения команды и куда поместить результат (если необходимо). Кманда может занимать от одного до нескольких байт, причем процессор узнает о том, сколько байт команды ему надо читать, из первого прочитанного им байта или слова. В процессоре код команды расшифровывается и преобразуется в набор микроопераций, выполняемых отдельными узлами процессора. Но разработчику микропроцессорных систем это знание не слишком важно, ему важен только результат выполнения той или иной команды.

Большая часть команд процессора работает с кодами данных (операндами). Одни команды требуют входных операндов (одного или двух), другие выдают выходные операнды (чаще один операнд). Входные операнды называются еще операндами-источниками, а выходные называются операндами-приемниками. Все эти коды операндов (входные и выходные) должны где-то располагаться. Они могут находиться во внутренних регистрах процессора (наиболее удобный и быстрый вариант). Они могут располагаться в системной памяти (самый распространенный вариант). Наконец, они могут находиться в устройствах ввода/вывода (наиболее редкий случай). Определение места положения операндов производится кодом команды. Причем существуют разные методы, с помощью которых код команды может определить, откуда брать входной операнд и куда помещать выходной операнд. Эти методы называются методами адресации. Эффективность выбранных методов адресации во многом определяет эффективность работы всего процессора в целом.

Адресные регистры или указатели используются для реализации тех или иных методов адресации операндов, используемых в конкретных командах микропроцессора [4]. Их конкретный набор и функции зависят от того, какие методы адресации реализованы в данной модели микропроцессора.

Под методом адресации понимается метод кодирования адреса операнда или результата операции в коде команды.

В общем случае код команды микропроцессора можно представить в следующем виде:

КОП	$АОП_1$	$АОП_2$	...	АР

где, КОП – код операции; $АОП_1$ – поле адреса первого операнда; $АОП_2$ – поле адреса второго операнда; АР – поле адреса результата.

Наличие отдельных полей, кроме КОП, определяется конкретной командой и типом микропроцессора. Информация в полях АОП и АР

определяется конкретным методом адресации, используемым в данной команде.

Наиболее распространенными методами адресации, используемыми в современных моделях микропроцессоров, являются:

- регистровая адресация;
- прямая адресация;
- непосредственная адресация;
- косвенная регистровая адресация;
- косвенная автоинкрементная/автодекрементная адресация;
- индексная адресация по базе;
- сегментная адресация.

В зависимости от того, какие методы адресации реализованы в конкретном процессоре, в нем имеются те или иные адресные регистры. Более сложные методы адресации требуют большего времени для вычисления адреса операнда. Одно из современных направлений развития архитектуры микропроцессоров основано на том, чтобы путем сокращения числа возможных команд и методов адресации добиться того, чтобы любая команда выполнялась за один машинный цикл. Эта идея реализована в RISC-процессорах (Reduced Instruction Set Computer).

3.2.1. Методы адресации

Количество методов адресации в различных процессорах может быть от 4 до 16. Рассмотрим несколько типичных методов адресации операндов, используемых сейчас в большинстве микропроцессоров [1].

Непосредственная адресация (рис. 3.4) предполагает, что операнд (входной) находится в памяти непосредственно за кодом команды. Операнд обычно представляет собой константу, которую надо куда-то переслать, к чему-то прибавить и т.д. Например, команда может состоять в том, чтобы прибавить число 6 к содержимому какого-то внутреннего регистра процессора. Это число 6 будет располагаться в памяти, внутри программы в адресе, следующем за кодом данной команды сложения.

Рис. 3.4. Непосредственная адресация

Например, команда сложения значения регистра аккумулятора с конкретным числом на языке Ассемблер-51 будет выглядеть так:

```
ADD     A, #6
```

Прямая (она же абсолютная) адресация (рис. 3.5) предполагает, что операнд (входной или выходной) находится в памяти по адресу, код которого находится внутри программы сразу же за кодом команды. Например, команда может состоять в том, чтобы очистить (сделать нулевым) содержимое ячейки памяти с адресом 1000000. Код этого адреса 1000000 будет располагаться в памяти, внутри программы в следующем адресе за кодом данной команды очистки. Например, команда сложения значения регистра аккумулятора с числом, имеющим размерность один байт, значение которого хранится в ячейке памяти по адресу MEM (в сегменте XDATA – шестнадцатиричный адрес C3F0), на языке Ассемблер-51 будет выглядеть так:

```
ADD          A, MEM
…
…
MEM: XDATA C3F0h
```

Рис. 3.5. Прямая адресация

Регистровая адресация (рис. 3.6) предполагает, что операнд (входной или выходной) находится во внутреннем регистре процессора. Например, команда может состоять в том, чтобы переслать число из нулевого регистра в первый. Номера обоих регистров (0 и 1) будут определяться кодом команды пересылки. Адреса регистров зашифрованы в коде операции. Поле адреса в команде отсутствует.

Рис. 3.6. Регистровая адресация

Например, команда пересылки значения регистра R1 в регистр аккумулятор на языке Ассемблер-51 будет выглядеть так:

```
ADD          A, R1
```

Косвенно-регистровая (она же косвенная) адресация предполагает, что во внутреннем регистре процессора находится не сам операнд, а его адрес в памяти (рис. 3.7). Физический адрес операнда расположен в регистре

косвенного адреса DP (Data Pointer). Адрес регистра включен в код операции. Поле адреса в команде отсутствует. В качестве DP может выступать РОН или специальный адресный регистр. Например, команда может состоять в том, чтобы очистить ячейку памяти с адресом, находящимся в нулевом регистре. Номер этого регистра (0) будет определяться кодом команды очистки.

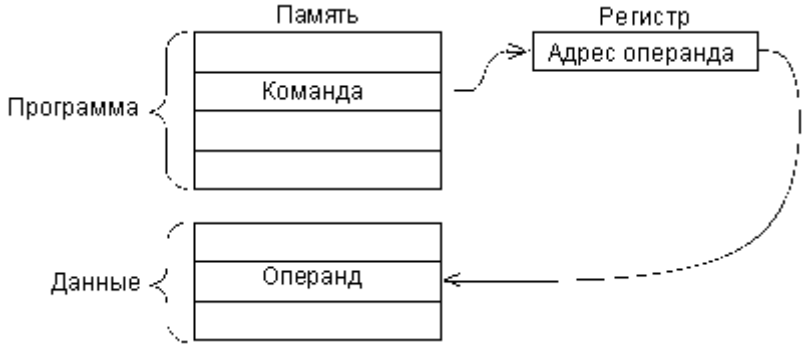


Рис. 3.7. Косвенная адресация

Например, очистка двух байт памяти по адресу MEM (в сегменте XDATA – шестнадцатиричный адрес C3F0), используя косвенную адресацию регистра DP, на языке Ассемблер-51 будет выглядеть так:

```
MOV     DP, #MEM
CLR     a
MOVX    @DP, a
INC     DP
MOVX    @DP, a
…
…
MEM: XDATA C3F0h
```

Автоинкрементная адресация очень близка к косвенной адресации, но отличается от нее тем, что после выполнения команды содержимое используемого регистра увеличивается на единицу или на два. Физический адрес операнда расположен в регистре косвенного адреса DP. Адрес регистра включен в код операции. Поле адреса в команде отсутствует. После выполнения операции содержимое DP автоматически инкрементируется, чтобы указывать на следующий элемент таблицы. Этот метод адресации очень удобен, например, при последовательной обработке кодов из массива данных, находящегося в памяти. После обработки какого-то кода адрес в регистре будет указывать уже на следующий код из массива. При использовании косвенной адресации в данном случае пришлось бы увеличивать содержимое этого регистра отдельной командой.

Автодекрементная адресация работает похоже на автоинкрементную, но только содержимое выбранного регистра уменьшается на единицу или на два перед выполнением команды. Эта адресация также удобна при обработке массивов данных. Совместное использование автоинкрементной и автодекрементной адресаций позволяет организовать память стекового типа.

Из других распространенных методов адресации можно упомянуть об ***индексных методах***, которые предполагают для вычисления адреса операнда прибавление к базовому адресу значения смещения (индекса). Существует три варианта адресации базового адреса и индекса:

- базовый адрес операнда расположен в соответствующем поле адреса, следующего за кодом команды Смещение адреса операнда относительно базового адреса расположено в индексном регистре X (Index). В качестве X может выступать РОН или специальный адресный регистр. Адрес индексного регистра включен в код операции;
- базовый адрес операнда расположен в регистре базы BP (Base Pointer). Адрес регистра включен в код операции. Смещение адреса операнда относительно базового адреса расположено в соответствующем поле адреса непосредственно за кодом команды. В качестве BP может выступать РОН или специальный адресный регистр;
- базовый адрес операнда расположен в регистре базы BP , смещение адреса операнда относительно базового адреса расположено в индексном регистре X. В качестве X и BP могут выступать РОН или специальные адресные регистры. Адреса регистров включены в код операции. Поле адреса в команде отсутствует.

Отметим, что выбор того или иного метода адресации в значительной степени определяет время выполнения команды. Самая быстрая адресация – это регистровая, так как она не требует дополнительных циклов обмена по магистрали. Если же адресация требует обращения к памяти, то время выполнения команды будет увеличиваться за счет длительности необходимых циклов обращения к памяти. Понятно, что чем больше внутренних регистров у процессора, тем чаще и свободнее можно применять регистровую адресацию, и тем быстрее будет работать система в целом.

3.2.2. Сегментная адресация памяти

Сегментная (страничная) адресация применяется в некоторых процессорах, например в процессорах IBM PC-совместимых персональных компьютеров [1]. Вся память разбита на сегменты (страницы) определенного объема. Адрес сегмента хранится в сегментном регистре SR (Segment Register), смещение адреса относительно начала сегмента расположено в соответствующем поле адреса либо в индексном регистре X. В качестве X может выступать РОН или специальный адресный регистр.

В процессоре Intel x86 вся память системы представляется не в виде непрерывного пространства, а в виде нескольких страниц – сегментов заданного размера (по 64 Кбайта), положение которых в пространстве памяти можно изменять программным путем. Для хранения кодов адресов памяти используются не отдельные регистры, а пары регистров:

- сегментный регистр определяет адрес начала сегмента (то есть положение сегмента в памяти);
- регистр указателя (регистр смещения) SR определяет положение рабочего адреса внутри сегмента.

При этом физический 20-разрядный адрес памяти, выставляемый на внешнюю шину адреса, образуется так, как показано на рис. 3.8, то есть путем сложения смещения и адреса сегмента со сдвигом на 4 бита. Положение этого адреса в памяти показано на рис. 3.9.

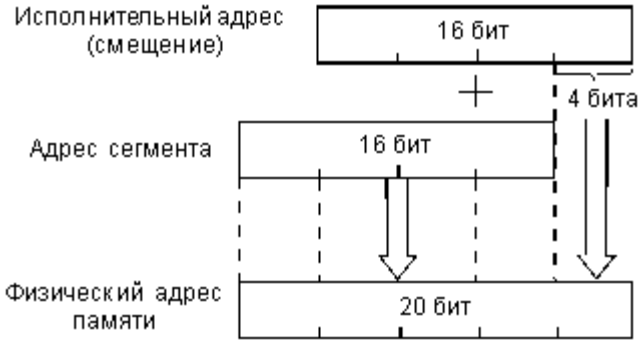


Рис. 3.8. Формирование физического адреса памяти из адреса сегмента и смещения

Сегмент может начинаться только на 16-байтной границе памяти (так как адрес начала сегмента, по сути, имеет четыре младших нулевых разряда), то есть с адреса, кратного 16. Эти допустимые границы сегментов называются границами параграфов.

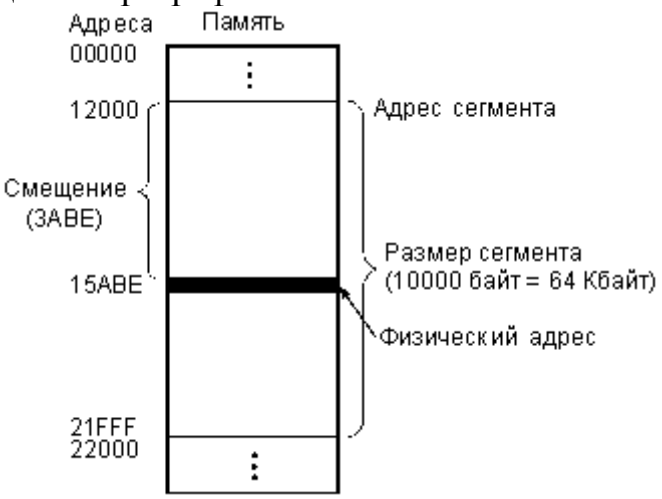


Рис. 3.9. Физический адрес в сегменте (все коды – шестнадцатеричные)

Отметим, что введение сегментирования, прежде всего, связано с тем, что внутренние регистры процессора 16-разрядные, а физический адрес памяти 20-разрядный (16-разрядный адрес позволяет использовать память только в 64 Кбайт, что явно недостаточно).

В любом случае сегментирование позволяет выделить в памяти один или несколько сегментов для данных и один или несколько сегментов для программ. Переход от одного сегмента к другому сводится всего лишь к

изменению содержимого сегментного регистра. Иногда это бывает очень удобно. Но для программиста работать с сегментированной памятью обычно сложнее, чем с непрерывной, несегментированной памятью, так как приходится следить за границами сегментов, за их описанием, переключением и т.д.

3.3. Система команд микропроцессора

В общем случае система команд процессора включает в себя следующие четыре основные группы команд [1]:

- команды пересылки данных;
- арифметические команды;
- логические команды;
- команды переходов.

Команды пересылки данных не требуют выполнения никаких операций над операндами. Операнды просто пересылаются (точнее, копируются) из источника (Source) в приемник (Destination). Источником и приемником могут быть внутренние регистры процессора, ячейки памяти или устройства ввода/вывода. АЛУ в данном случае не используется.

Арифметические команды выполняют операции сложения, вычитания, умножения, деления, увеличения на единицу (инкрементирования), уменьшения на единицу (декрементирования) и т.д. Этим командам требуется один или два входных операнда. Формируют команды один выходной операнд.

Логические команды производят над операндами логические операции, например, логическое И, логическое ИЛИ, исключающее ИЛИ, очистку, инверсию, разнообразные сдвиги (вправо, влево, арифметический сдвиг, циклический сдвиг). Этим командам, как и арифметическим, требуется один или два входных операнда, и формируют они один выходной операнд.

Наконец, команды переходов предназначены для изменения обычного порядка последовательного выполнения команд. С их помощью организуются переходы на подпрограммы и возвраты из них, всевозможные циклы, ветвления программ, пропуски фрагментов программ и т.д. Команды переходов всегда меняют содержимое счетчика команд. Переходы могут быть условными и безусловными. Именно эти команды позволяют строить сложные алгоритмы обработки информации.

В соответствии с результатом каждой выполненной команды устанавливаются или очищаются биты регистра состояния процессора (PSW). Но надо помнить, что не все команды изменяют все имеющиеся в PSW флаги. Это определяется особенностями каждого конкретного процессора.

У разных процессоров системы команд существенно различаются, но в основе своей они очень похожи. Количество команд у процессоров также различно. Например, у упоминавшегося уже процессора MC68000 всего 61 команда, а у процессора 8086 – 133 команды. У современных мощных

процессоров количество команд достигает нескольких сотен. В то же время существуют процессоры с сокращенным набором команд (RISC-процессоры), в которых за счет максимального сокращения количества команд достигается увеличение эффективности и скорости их выполнения.

3.3.1. Команды пересылки данных

Команды пересылки данных занимают очень важное место в системе команд любого процессора [1]. Они выполняют следующие важнейшие функции:

- загрузка (запись) содержимого во внутренние регистры процессора;
- сохранение в памяти содержимого внутренних регистров процессора;
- копирование содержимого из одной области памяти в другую;
- запись в устройства ввода/вывода и чтение из устройств ввода/вывода.

В некоторых процессорах (например, Т-11) все эти функции выполняются одной единственной командой MOV (для байтовых пересылок – MOVB) но с различными методами адресации операндов.

В других процессорах помимо команды MOV имеется еще несколько команд для выполнения перечисленных функций. Например, для загрузки регистров могут использоваться команды загрузки, причем для разных регистров – разные команды (их обозначения обычно строятся с использованием слова LOAD – загрузка).

Часто выделяются специальные команды для сохранения в стеке и для извлечения из стека (PUSH – сохранить в стеке, POP – извлечь из стека). Эти команды выполняют пересылку с автоинкрементной и с автодекрементной адресацией (даже если эти режимы адресации не предусмотрены в процессоре в явном виде).

Иногда в систему команд вводится специальная команда MOVS для строчной (или цепочечной) пересылки данных (например, в процессоре 8086). Эта команда пересылает не одно слово или байт, а заданное количество слов или байтов (MOVSB), то есть инициирует не один цикл обмена по магистрали, а несколько. При этом адрес памяти, с которым происходит взаимодействие, увеличивается на 1 или на 2 после каждого обращения или же уменьшается на 1 или на 2 после каждого обращения. То есть в неявном виде применяется автоинкрементная или автодекрементная адресация.

В некоторых процессорах (например, в процессоре 8086) специально выделяются функции обмена с устройствами ввода/вывода. Команда IN используется для ввода (чтения) информации из устройства ввода/вывода, а команда OUT используется для вывода (записи) в устройство ввода/вывода. Обмен информацией в этом случае производится между регистром-аккумулятором и устройством ввода/вывода. В более поздних процессорах этого же семейства (начиная с процессора 80286) добавлены команды

строчного (цепочечного) ввода (команда INS) и строчного вывода (команда OUTS). Эти команды позволяют пересылать целый массив (строку) данных из памяти в устройство ввода/вывода (OUTS) или из устройства ввода/вывода в память (INS). Адрес памяти после каждого обращения увеличивается или уменьшается (как и в случае с командой MOVS).

Также к командам пересылки данных относятся команды обмена информацией (их обозначение строится на основе слова Exchange). Может быть предусмотрен обмен информацией между внутренними регистрами, между двумя половинами одного регистра (SWAP) или между регистром и ячейкой памяти.

3.3.2. Арифметические команды

Арифметические команды рассматривают коды операндов как числовые двоичные или двоично-десятичные коды [1]. Эти команды могут быть разделены на пять основных групп:

- команды операций с фиксированной запятой (сложение, вычитание, умножение, деление);
- команды операций с плавающей запятой (сложение, вычитание, умножение, деление);
- команды очистки;
- команды инкремента и декремента;
- команда сравнения.

Команды операций с фиксированной запятой работают с кодами в регистрах процессора или в памяти как с обычными двоичными кодами. Команда сложения (ADD) вычисляет сумму двух кодов. Команда вычитания (SUB) вычисляет разность двух кодов. Команда умножения (MUL) вычисляет произведение двух кодов (разрядность результата вдвое больше разрядности сомножителей). Команда деления (DIV) вычисляет частное от деления одного кода на другой. Причем все эти команды могут работать как с числами со знаком, так и с числами без знака.

Команды операций с плавающей запятой (точкой) используют формат представления чисел с порядком и мантиссой (обычно эти числа занимают две последовательные ячейки памяти). В современных мощных процессорах набор команд с плавающей запятой не ограничивается только четырьмя арифметическими действиями, а содержит и множество других более сложных команд, например, вычисление тригонометрических функций, логарифмических функций, а также сложных функций, необходимых при обработке звука и изображения.

Команды очистки (CLR) предназначены для записи нулевого кода в регистр или ячейку памяти. Эти команды могут быть заменены командами пересылки нулевого кода, но специальные команды очистки обычно выполняются быстрее, чем команды пересылки. Команды очистки иногда относят к группе логических команд, но суть их от этого не меняется.

Команды инкремента (увеличения на единицу, INC) и декремента (уменьшения на единицу, DEC) также бывают очень удобны. Их можно в принципе заменить командами суммирования с единицей или вычитания единицы, но инкремент и декремент выполняются быстрее, чем суммирование и вычитание. Эти команды требуют одного входного операнда, который одновременно является и выходным операндом.

Наконец, команда сравнения (обозначается CMP) предназначена для сравнения двух входных операндов. По сути, она вычисляет разность этих двух операндов, но выходного операнда не формирует, а всего лишь изменяет биты в регистре состояния процессора (PSW) по результату этого вычитания. Следующая за командой сравнения команда (обычно это команда перехода) будет анализировать биты в регистре состояния процессора и выполнять действия в зависимости от их значений. В некоторых процессорах предусмотрены команды цепочечного сравнения двух последовательностей операндов, находящихся в памяти (например, в процессоре x86 и совместимых с ним).

3.3.3. Логические команды

Логические команды выполняют над операндами логические (побитовые) операции, то есть они рассматривают коды операндов не как единое число, а как набор отдельных битов. Этим они отличаются от арифметических команд. Логические команды выполняют следующие основные операции:

- логическое И, логическое ИЛИ, сложение по модулю 2 (Исключающее ИЛИ);
- логические, арифметические и циклические сдвиги;
- проверка битов и операндов;
- установка и очистка битов (флагов) регистра состояния процессора (PSW).

Команды логических операций позволяют побитно вычислять основные логические функции от двух входных операндов. Кроме того, операция И (AND) используется для принудительной очистки заданных битов (в качестве одного из операндов при этом используется код маски, в котором разряды, требующие очистки, установлены в нуль). Операция ИЛИ (OR) применяется для принудительной установки заданных битов (в качестве одного из операндов при этом используется код маски, в котором разряды, требующие установки в единицу, равны единице). Операция "Исключающее ИЛИ" (XOR) используется для инверсии заданных битов (в качестве одного из операндов при этом применяется код маски, в котором биты, подлежащие инверсии, установлены в единицу). Команды требуют двух входных операндов и формируют один выходной операнд.

Команды сдвигов позволяют побитно сдвигать код операнда вправо (в сторону младших разрядов) или влево (в сторону старших разрядов). Тип сдвига (логический, арифметический или циклический) определяет, каково

будет новое значение старшего бита (при сдвиге вправо) или младшего бита (при сдвиге влево), а также определяет, будет ли где-то сохранено прежнее значение старшего бита (при сдвиге влево) или младшего бита (при сдвиге вправо). Например, при логическом сдвиге вправо в старшем разряде кода операнда устанавливается нуль, а младший разряд записывается в качестве флага переноса в регистр состояния процессора. А при арифметическом сдвиге вправо значение старшего разряда сохраняется прежним (нулем или единицей), младший разряд также записывается в качестве флага переноса.

Циклические сдвиги позволяют сдвигать биты кода операнда по кругу (по часовой стрелке при сдвиге вправо или против часовой стрелки при сдвиге влево). При этом в кольцо сдвига может входить или не входить флаг переноса. В бит флага переноса (если он используется) записывается значение старшего бита при циклическом сдвиге влево и младшего бита при циклическом сдвиге вправо. Соответственно, значение бита флага переноса будет переписываться в младший разряд при циклическом сдвиге влево и в старший разряд при циклическом сдвиге вправо.

Для примера на рис. 3.10 показаны действия, выполняемые командами сдвигов вправо.

Команды проверки битов и операндов предназначены для установки или очистки битов регистра состояния процессора в зависимости от значения выбранных битов или всего операнда в целом. Выходного операнда команды не формируют. Команда проверки операнда (TST) проверяет весь код операнда в целом на равенство нулю и на знак (на значение старшего бита), она требует только одного входного операнда. Команда проверки бита (BIT) проверяет только отдельные биты, для выбора которых в качестве второго операнда используется код маски. В коде маски проверяемым битам основного операнда должны соответствовать единичные разряды.

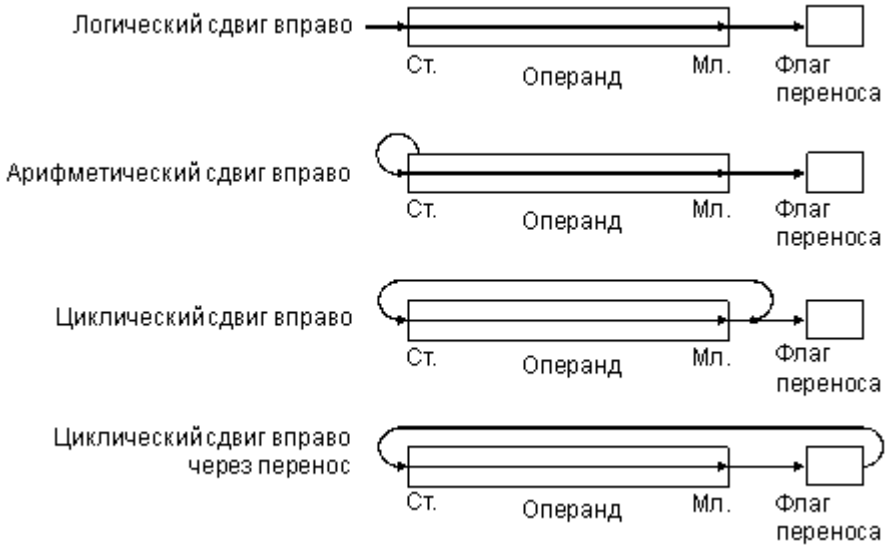


Рис. 3.10. Команды сдвигов вправо

Наконец, команды установки и очистки битов регистра состояния процессора (то есть флагов) позволяют установить или очистить любой флаг,

что бывает очень удобно. Каждому флагу обычно соответствуют две команды, одна из которых устанавливает его в единицу, а другая сбрасывает в нуль. Например, флагу переноса C (от Carry) будут соответствовать команды CLC (очистка) и SEC или STC (установка).

3.3.4. Команды переходов

Команды переходов предназначены для организации всевозможных циклов, ветвлений, вызовов подпрограмм и т.д., то есть они нарушают последовательный ход выполнения программы. Эти команды записывают в регистр-счетчик команд новое значение и тем самым вызывают переход процессора не к следующей по порядку команде, а к любой другой команде в памяти программ. Некоторые команды переходов предусматривают в дальнейшем возврат назад, в точку, из которой был сделан переход, другие не предусматривают этого. Если возврат предусмотрен, то текущие параметры процессора сохраняются в стеке. Если возврат не предусмотрен, то текущие параметры процессора не сохраняются.

Команды переходов без возврата делятся на две группы:

- команды безусловных переходов;
- команды условных переходов.

В обозначениях этих команд используются слова Branch (ветвление) и Jump (прыжок).

Команды безусловных переходов вызывают переход в новый адрес независимо ни от чего. Они могут вызывать переход на указанную величину смещения (вперед или назад) или же на указанный адрес памяти. Величина смещения или новое значение адреса указываются в качестве входного операнда.

Команды условных переходов вызывают переход не всегда, а только при выполнении заданных условий. В качестве таких условий обычно выступают значения флагов в регистре состояния процессора (PSW). То есть условием перехода является результат предыдущей операции, меняющей значения флагов. Всего таких условий перехода может быть от 4 до 16. Несколько примеров команд условных переходов:

- переход, если равно нулю;
- переход, если не равно нулю;
- переход, если есть переполнение;
- переход, если нет переполнения;
- переход, если больше нуля;
- переход, если меньше или равно нулю.

Если условие перехода выполняется, то производится загрузка в регистр-счетчик команд нового значения. Если же условие перехода не выполняется, счетчик команд просто наращивается, и процессор выбирает и выполняет следующую по порядку команду.

Специально для проверки условий перехода применяется команда сравнения (CMP), предшествующая команде условного перехода (или даже

нескольким командам условных переходов). Но флаги могут устанавливаться и любой другой командой, например командой пересылки данных, любой арифметической или логической командой. Отметим, что сами команды переходов флаги не меняют, что как раз и позволяет ставить несколько команд переходов одну за другой.

Совместное использование нескольких команд условных и безусловных переходов позволяет процессору выполнять разветвленные алгоритмы любой сложности. Для примера на рис. 3.11 показано разветвление программы на две ветки с последующим соединением, а на рис. 3.12 – разветвление на три ветки с последующим соединением.

Команды переходов с дальнейшим возвратом в точку, из которой был произведен переход, применяются для выполнения подпрограмм, то есть вспомогательных программ. Эти команды называются также командами вызова подпрограмм (распространенное название – CALL). Использование подпрограмм позволяет упростить структуру основной программы, сделать ее более логичной, гибкой, легкой для написания и отладки. В то же время надо учитывать, что широкое использование подпрограмм, как правило, увеличивает время выполнения программы.

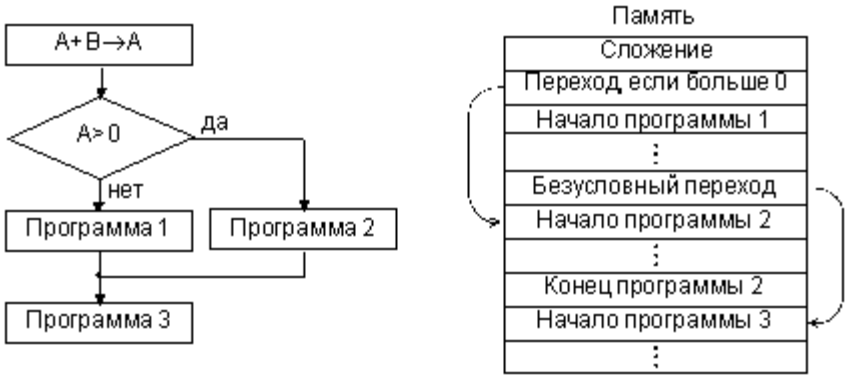


Рис. 3.11. Реализация разветвления на две ветки

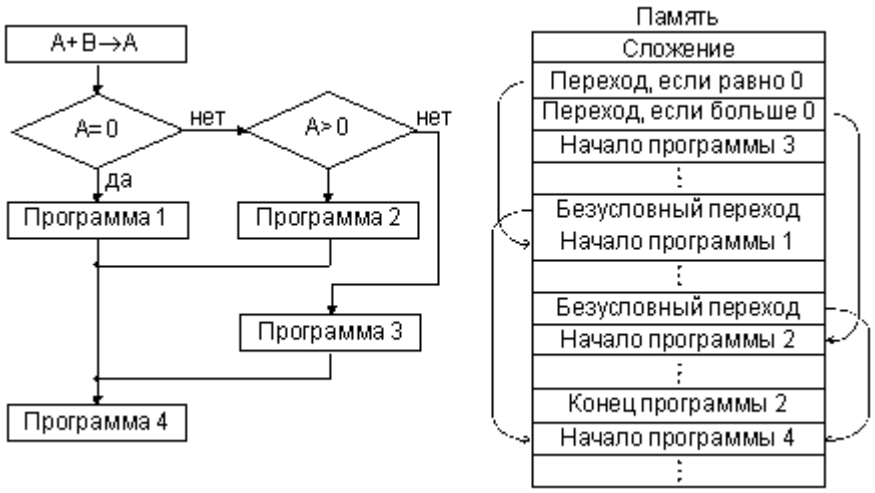


Рис. 3.12. Реализация разветвления на три ветки

Все команды переходов с возвратом предполагают безусловный переход (они не проверяют никаких флагов). При этом они требуют одного входного операнда, который может указывать как абсолютное значение нового адреса, так и смещение, складываемое с текущим значением адреса. Текущее значение счетчика команд (текущий адрес) сохраняется перед выполнением перехода в стеке.

Для обратного возврата в точку вызова подпрограммы (точку перехода) используется специальная команда возврата (RET или RTS). Эта команда извлекает из стека значение адреса команды перехода и записывает его в регистр-счетчик команд.

Особое место среди команд перехода с возвратом занимают команды прерываний (распространенное название – INT). Эти команды в качестве входного операнда требуют номер прерывания (адрес вектора). Обслуживание таких переходов осуществляется точно так же, как и аппаратных прерываний. То есть для выполнения данного перехода процессор обращается к таблице векторов прерываний и получает из нее по номеру прерывания адрес памяти, в который ему необходимо перейти. Адрес вызова прерывания и содержимое регистра состояния процессора (PSW) сохраняются в стеке. Сохранение PSW – важное отличие команд прерывания от команд переходов с возвратом.

Команды прерываний во многих случаях оказываются удобнее, чем обычные команды переходов с возвратом. Сформировать таблицу векторов прерываний можно один раз, а потом уже обращаться к ней по мере необходимости. Номер прерывания соответствует номеру подпрограммы, то есть номеру функции, выполняемой подпрограммой. Поэтому команды прерывания гораздо чаще включаются в системы команд процессоров, чем обычные команды переходов с возвратом.

Для возврата из подпрограммы, вызванной командой прерывания, используется команда возврата из прерывания (IRET или RTI). Эта команда извлекает из стека сохраненное там значение счетчика команд и регистра состояния процессора (PSW).

Отметим, что у некоторых процессоров предусмотрены также команды условных прерываний, например, команда прерывания при переполнении.

Конечно, в данном разделе мы рассмотрели только основные команды, наиболее часто встречающиеся в процессорах. У конкретных процессоров могут быть и многие другие команды, не относящиеся к перечисленным группам команд. Но изучать их надо уже после того, как выбран тип процессора, подходящий для задачи, решаемой данной микропроцессорной системой.

Вопросы для повторения

1. Перечислить сегменты памяти МС, различающиеся по функциональному назначению.
2. В чем особенность памяти начального запуска?

3. Охарактеризовать память, предназначенную для УВВ.
4. Указать особенности построения сегмента памяти типа стек.
5. Изложить принцип работы стека.
6. Что такое вектор прерываний?
7. Что такое таблица векторов прерываний?
8. Пояснить алгоритм обработки прерываний.
9. В какой области памяти физически располагаются программно-доступные регистры?
10. Как делятся по назначению программно-доступные регистры?
11. В чем состоят достоинства и недостатки аккумуляторной, регистровой архитектуры и типа «память-память»?
12. Информация какого характера хранится в счетчике команд?
13. На что указывает в сегменте стека указатель стека?
14. Каково назначения регистра слово состояния процессора?
15. Перечислите флаги в регистре PSW.
16. В каких сегментах памяти могут находиться операнды инструкций МП?
17. Приведите примеры команд на языке Ассемблер для непосредственной, абсолютной и регистровой адресаций.
18. В чем особенность косвенной адресации?
19. Укажите отличия автоинкрементной и автодекрементной адресации.
20. Какие индексные методы адресации используются в МС?
21. С какой целью применяется сегментная (страничная) адресация памяти?
22. Чем отличается система команд RISC- и CISC-процессора?
23. Какие команды пересылки данных Вы знаете?
24. Приведите примеры арифметических команд.
25. Какой арифметической операции соответствуют логические команды сдвигов?
26. Охарактеризуйте команды безусловного и условного перехода.
27. В чем особенность перехода при реализации подпрограмм и обработки прерываний?

Тема 4. Запоминающие устройства и программируемые логические матрицы

Цели и задачи изучения темы

Основной целью изучения данной темы является знакомство с видами, устройством и принципом действия и особенностями построения полупроводниковых запоминающих устройств и программируемых логических матриц.

Задачи:

1. Рассмотреть классификацию ЗУ, способы организации массива-накопителя и структуру ЗУ.
2. Изучить принципы построения статических и динамических ОЗУ.
3. Разновидности и особенности построения постоянных запоминающих устройств.
4. Ознакомится с программируемыми логическими матрицами (ПЛИС) без и с обратной связью.

4.1. Запоминающие устройства

Небольшие объемы информации можно хранить в регистрах процессора, однако при необходимости хранить уже десятки байт применение регистров приводит к неоправданно большим аппаратурным затратам. Для хранения больших массивов данных строят запоминающие устройства (ЗУ) с использованием специальных микросхем. ЗУ микропроцессорной системы представляет собой совокупность электронных средств, обеспечивающих запись, хранение и воспроизведение (считывание) информации под управлением МП или ВУ.

Основными операциями в памяти в общем случае являются занесение информации в память ("запись") и выборка информации из памяти ("чтение"). Обе эти операции осуществляют ("обращение к памяти"). При обращении к памяти производится чтение или запись некоторой единицы данных – различной для устройств разного типа. Такой единицей может быть, например, байт, машинное слово или совокупность машинных слов (зона, сектор, блок) для внешних ЗУ.

Производительность и вычислительные возможности МС в значительной степени определяются составом и характеристиками ее ЗУ. В составе ЭВМ используется одновременно несколько типов ЗУ, отличающихся принципом действия, характеристиками и назначением. Ниже приведены важнейшие количественные характеристики ЗУ [4]:

- ***информационная емкость*** – максимально возможное количество байтов хранимой информации;
- ширина выборки (***разрядность***) – количество разрядов информации, записываемых в ЗУ или извлекаемых из него за одно обращение;

- ***быстродействие*** – характеризуется временем цикла обращения к ЗУ, а именно: временем выборки $t_в$, представляющим собой интервал времени между моментом подачи сигнала выборки и появлением считанных данных на выходе, и циклом записи $t_{цз}$, определяемым минимально допустимым временем между моментом подачи сигнала выборки при записи и моментом, когда допустимо последующее обращение к памяти.

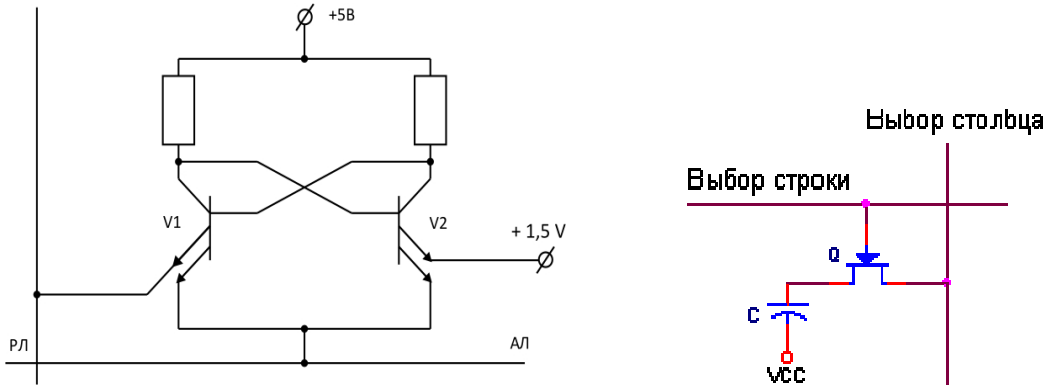
Запоминающие устройства можно разделить по следующим важнейшим классификационным признакам.

- По физическим принципам работы ЗУ делятся на магнитные, электронные, оптические, механические, криогенные.
- По способу организации доступа к памяти различают ЗУ с произвольным доступом (***адресные***) и ЗУ ***с последовательным доступом*** (безадресные).
- По способности сохранять информацию при отключенном питание. Делятся на ЗУ, не способные сохранять информацию при отключении питания, и ***энергонезависимые*** ЗУ;
- По способу хранения информации различают ***статические*** и ***динамические*** ЗУ. В статических ЗУ физическое состояние носителя информации в процессе хранения не изменяется. В динамических ЗУ состояние носителя постоянно изменяется.
- По функциональному назначению различают ***сверхоперативные*** ЗУ (СОЗУ), ***оперативные*** ЗУ (ОЗУ), ***постоянные*** ЗУ (ПЗУ), ***перепрограммируемые*** постоянные ЗУ (ППЗУ), стековые ЗУ.

Каждая микросхема ЗУ кроме времени обращения и емкости характеризуется потребляемой мощностью, набором питающих напряжений, типом корпуса (числом выводов). Микросхемы ППЗУ дополнительно характеризуются временем хранения записанной в них информации (по истечении которого хранящаяся в ячейках информация может самопроизвольно измениться), допустимым количеством циклов перезаписи (после чего микросхема считается негодной для использования).

Рассмотрим основные понятия и принципы построения запоминающих устройств.

Один бит информации может быть сохранен в элементарной запоминающей ячейке (ЗЯ), построение которой может базироваться на полупроводниковой схеме типа «триггер» или на емкостном элементе. Примеры реализации элементарных ЗЯ приведены на рис. 4.1, а – схема простейшего триггера; б – схема с емкостным элементом.



а) *б)*

Рис. 4.1. Элементарная ЗЯ: схема триггера (а); на емкостном элементе (б)

Первая схема (рис. 4.1-а) применяется при построении ОЗУ статического типа и предназначена для реализации по ТТЛ-технологии. Выбор ЗЯ для записи или чтения осуществляется путем подачи сигнала низкого уровня на адресную линию (АЛ).

Запись логической «1» в ЗЯ производится при подачи на разрядную линию (РЛ) активного уровня сигнала (логического «0»), при этом триггер устанавливается в устойчивое состояние: V1 открывается, а V2 закрывается. Это устойчивое состояние сохраняется в триггере пока не отключится его питание или не будет записан в ЗЯ другой уровень – логический «0».

При подаче на РЛ сигнала высокого уровня триггер переключится в другое устойчивое состояние: V1 – закрыт, V2 – открыт, – и произойдет запись логического «0» в ЗЯ. Считывание информации производится при высоком входном сопротивлении РЛ, тогда на эмиттере V1, так же как и на эмиттере V2, устанавливается напряжение примерно 1,5 В, по величине тока в РЛ, ток низкого уровня соответствует логическому «0», а ток высокого уровня – логической «1». Недостатком такой схемы является сравнительная сложность и невысокая плотность ячеек памяти на кристалле.

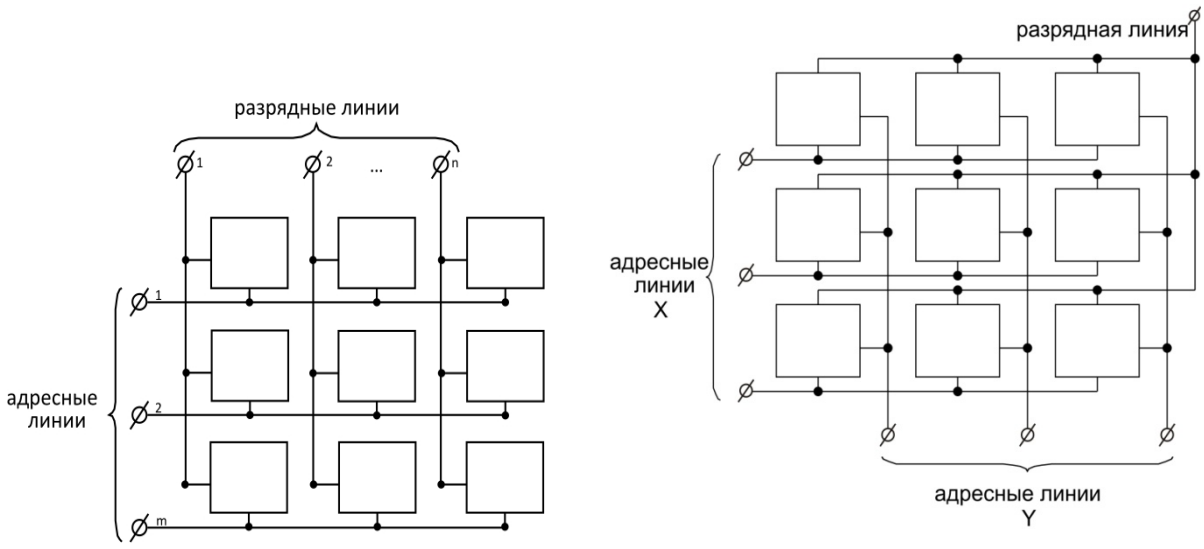
Типовая схема элементарной запоминающей ячейки, представленной на рис. 4.1-б, реализуется при построении ОЗУ динамического типа, в такой ячейки информация запоминается на конденсаторе С. Считывание и запись ЗЯ, т.е. ее выбор, производится при открывании полупроводникового вентиля Q, управляемого от линии «выбор строки», а, собственно, считывание и запись бита информации производится через линию «выбор столбца». В зависимость от величины заряда конденсатора С ячейка хранит либо «0» либо «1». Однако, из-за утечек тока через изоляцию обкладок конденсатора с течением времени происходит саморазряд конденсатора, и как следствие – потеря хранимой информации. В этом основной недостаток таких ЗУ. Для исключения этого недостатка используют достаточно сложные схемы регенерации ОЗУ, работа которых заключается в периодическом построчном сканировании матрицы ЗЯ и одновременной подзарядки конденсаторов ячеек памяти.

Для хранения байта информации или многоразрядного слова потребуется несколько ЗЯ, соединенных определенным образом. При наличии большого числа хранимых слов необходимо устанавливать определенный порядок выбора ячеек памяти. Массив ЗЯ часто называют накопителем (НК). Он представляет собой матрицу, составленную из элементарных ячеек памяти, расположенных вдоль строк и столбцов. Элемент памяти может хранить 1 бит информации (логический «0» либо логическую «1»). Кроме того, он снабжен управляющими цепями для установки элемента в любом из трех режимов: режиме хранения, в котором он отключается от входа и выхода микросхемы, режиме чтения, в котором содержащаяся в ЗЯ информация выдается на выход микросхемы, режиме записи, в котором в ЗЯ записывается новая поступающая со входа микросхемы информация.

В настоящее время используется главным образом адресный принцип хранения информации, предусматривающий наличие у каждой ячейки памяти определенного номера – адреса. Запоминающее устройство содержит некоторое число N ячеек слов, в каждой из которых может храниться слово с определенным числом разрядов n, число элементарных ЗЯ $M=N*n$. Ячейки последовательно нумеруются двоичными числами. Номер ячейки называется адресом.

Для поиска требуемой ЗЯ указывается строка и столбец, соответствующие положению ЗЯ в накопителе. Адрес ЗЯ в виде двоичного числа принимается по шине адреса в регистр адреса. Если для представления адресов используются комбинации m-разрядного двоичного кода, то емкость накопителя – число ячеек в ЗУ – составит $M = 2^m$. Число строк и число столбцов накопителя выбираются равными целой степени двух. И если число строк $M_{стр}=2^{m1}$, где $m1$ – число разрядов адреса, поступающих на дешифратор строк, и число столбцов $M_{ст}=2^{m2}$, где $m2$ – число разрядов адреса, поступающих на дешифратор столбцов, то общее число ЗЯ (емкость накопителя): $M=M_{стр}*M_{ст}=2^{m1}*2^{m2}=2^{(m1+m2)}$, где $m=m1+m2$ – число разрядов адреса, принимаемого в регистр адреса. Например, при емкости $M=2^{10}=1024$ число разрядов адреса $m=10$, при этом выбирается $m_1=m_2=m/2=5$, в этом случае число имеется $2^5=32$ строк и столько же столбцов в накопителе, всего ячеек – 32*32=1024.

В зависимости от того, каким образом соединяются и выбираются ячейки памяти в ЗУ, можно выделить два вида организации НК: словарную и матричную. На рис. 4.2 изображены словарный (а) и матричный (б) способы соединения элементарных запоминающих ячеек накопителя.



а) *б)*

Рис. 4.2. Организации НК: словарная (а); матричная (б)

Запоминающие устройства строятся из набора однотипных микросхем ЗУ с определенным их соединением. В первом случае (рис. 4.2-а) микросхемы памяти собираются по принципу наращивания объема, характерный пример – статические ОЗУ и ПЗУ, а во втором случае (рис. 4.2-б) – по принципу наращивания разрядности, что свойственно для ОЗУ динамического типа. Пример структуры ОЗУ с матричной организацией НК приведен на рис. 4.3.

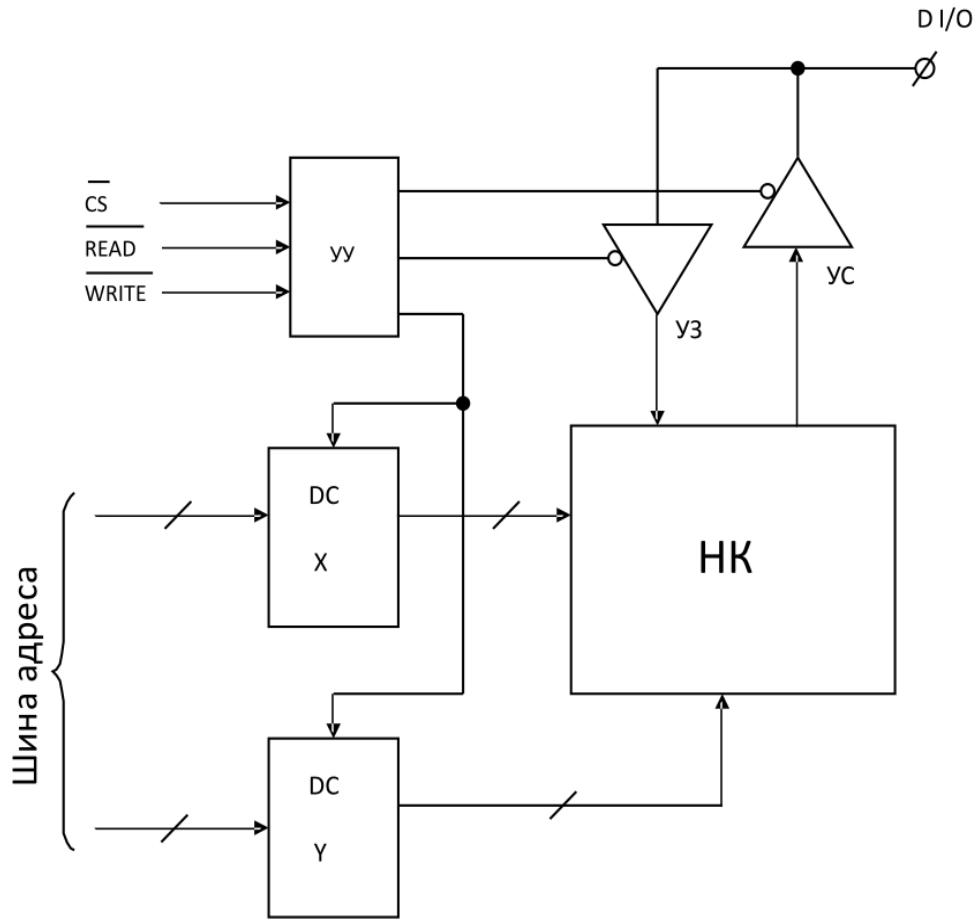


Рис. 4.3. Структура ЗУ с матричной организацией НК

В этой схеме используются отдельные дешифраторы адреса накопителя по строкам (координата X) и по столбцам (координата Y), использующие различные адресные линии ША. УУ управляет работой дешифраторов и в зависимости от режима чтения или записи активизирует или блокирует усилители записи и считывания (УЗ и УС), таким образом, реализуется однопроводная двунаправленная ШД (D I/O). Для организации запоминания *n*-разрядных слов информации необходимо соединить *n* таких одноразрядных микросхем памяти.

4.1.1. Оперативное запоминающее устройство

Оперативное ЗУ (международная аббревиатура – RAM) используется в условиях, когда необходимо выбирать и обновлять хранимую информацию в высоком темпе работы процессора микропроцессорной системы. Вследствие этого в ОЗУ предусматриваются три режима работы: режим хранения при отсутствии обращения к ЗУ, режим чтения хранимых данных и режим записи. При этом в режимах чтения и записи ОЗУ должно функционировать с высоким быстродействием (обычно время чтения или записи слова в ОЗУ составляет доли микросекунды). В цифровых устройствах ОЗУ используются для хранения данных (исходных данных, промежуточных и конечных результатов обработки данных) и программ.

Как было указано выше ОЗУ разделяют на динамические и статические. Элементарная ячейка динамического ОЗУ базируется на конденсаторе (см. рис. 4.1-б) [4].

Подобные запоминающие ячейки расположены в виде матрицы на пересечении линий связи, называемых соответственно столбцами и строками. Во время цикла записи конденсатор заряжается до “0” или “1” от шины выбора столбца при активном состоянии сигнала выбора ряда. Во время цикла чтения заряд конденсатора передается через открытый транзистор и шину выбора столбца на выходные усилители, и дальше, на выход. Утечки емкости, а также ее разряд во время цикла чтения обуславливают необходимость регенерации информации, которая заключается в последовательном переборе адресов строк. Пример динамического ОЗУ емкостью 256 Кбит (разрядность 1 бит) с отдельными линиями данных на считывание DO и на запись DI приведен на рис. 4.4.

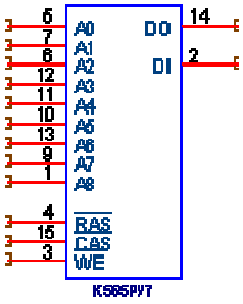


Рис. 4.4. Динамическое ОЗУ

Диаграмма цикла записи данных представлена на рис. 4.5.

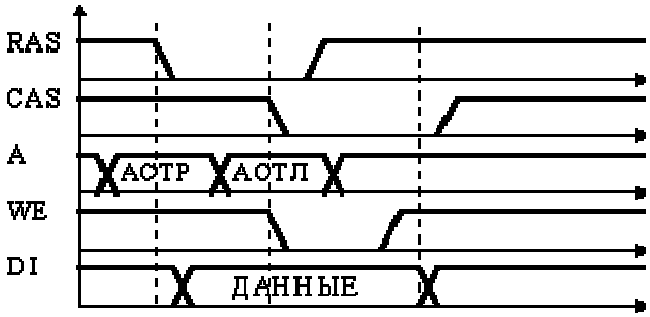


Рис. 4.5. Диаграмма цикла записи динамического ОЗУ

Сначала на адресные входы подается код адреса необходимой строки, который фиксируется внутри микросхемы по отрицательному фронту сигнала RAS. Затем, аналогичным образом, по отрицательному фронту сигнала CAS фиксируется код адреса необходимого столбца. После этого, по отрицательному фронту сигнала WE данные, находящиеся на входе DI, записываются в выбранную запоминающую ячейку.

Диаграмма цикла чтения приведена на рис. 4.6.

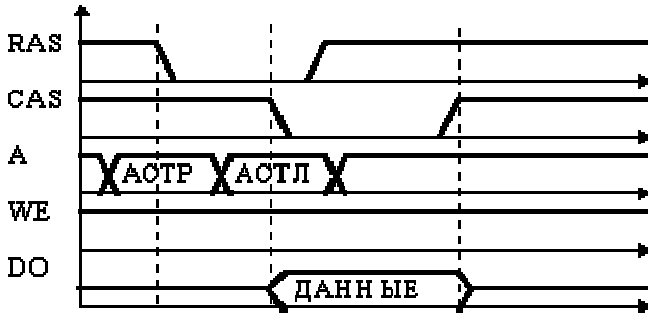


Рис. 4.6. Диаграмма цикла чтения динамического ОЗУ

Адрес требуемой запоминающей ячейки фиксируется аналогично циклу записи. При нулевом уровне сигнала на входе CAS и единичном на входе WE выход DO выходит из высокоимпедансного состояния и на нем появляются данные из требуемой запоминающей ячейки.

Регенерация информации осуществляется обращением к каждой из 512 строк не реже, чем один раз каждые 8 мс. При этом регенерируется сразу вся строка.

Элементарной запоминающей ячейкой статического ОЗУ является триггер (см. рис. 4.1-а). Триггер, как схема с двумя устойчивыми состояниями не требует регенерации информации. Он способен хранить информацию как угодно долго при включенном питании. Пример статического ОЗУ емкостью 2048 байт (16348 бит, разрядность 8 бит) приведен на рис. 4.7.

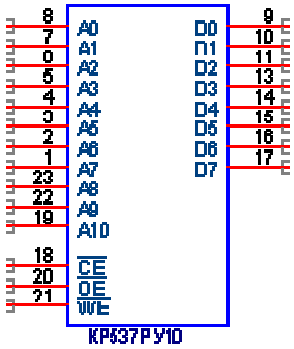
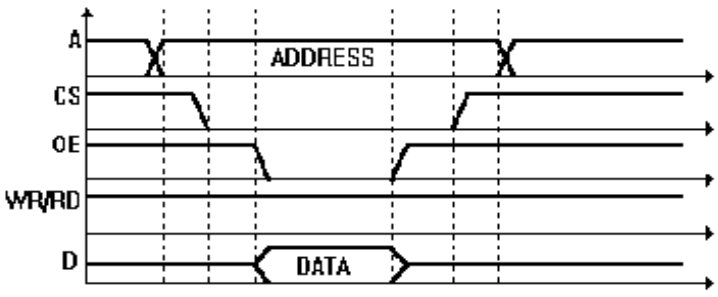
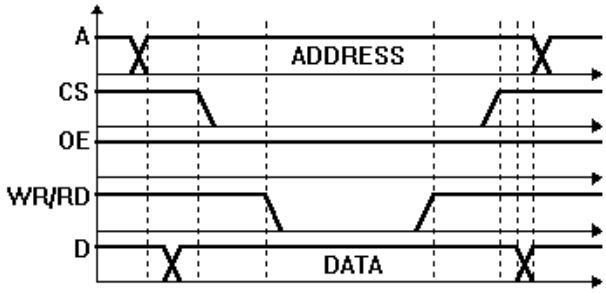


Рис. 4.7. Статическое ОЗУ

Диаграммы циклов чтения и записи представлены на рис. 4.8.



а)



б)

Рис. 4.8. Диаграмма циклов чтения (а) и записи (б) статического ОЗУ

На адресные входы подается код полного адреса требуемой запоминающей ячейки (а не отдельные адреса строки и столбца как в предыдущем случае) и удерживается на протяжении всего цикла. Вход CS является входом выбора кристалла. Микросхема осуществляет цикл записи или чтения только при нулевом уровне сигнала на этом входе. Вход OE является входом разрешения выхода. При подаче на этот вход сигнала нулевого уровня выходы данных микросхемы выходят из высокоимпедансного состояния и на них появляются данные из запоминающей ячейки, адрес которой в данный момент присутствует на адресных входах. Запись данных в микросхему осуществляется по положительному перепаду сигнала на входе WR/RD при единичном уровне сигнала на входе OE и нулевом – на входе CS.

Статические ОЗУ, как правило, быстрее, чем динамические и не требуют регенерации информации в процессе работы. Однако, они имеют более низкую удельную емкость (число ЗЯ на кристалле) и более высокую стоимость производства, в сравнении с динамическими ОЗУ. Поэтому статические ОЗУ имеют меньший объем и большую стоимость чем динамические ОЗУ.

4.1.2. Постоянные запоминающие устройства (ПЗУ).

Постоянное ЗУ (международная аббревиатура – ROM) предназначено для хранения некоторой однажды записанной в него информации, не нарушаемой и при отключении источников питания. В ПЗУ предусматриваются два режима работы: режим хранения и режим чтения с высоким быстродействием. Режим записи не предусматривается. Используются ПЗУ для хранения программ в таких специализированных МС, которые, функционируя длительное время, многократно выполняют действия по одному и тому же алгоритму при различных исходных данных.

Постоянное запоминающее устройство в составе МС работает только в режиме чтения, его содержимое не может быть изменено микропроцессором в процессе работы и сохраняется при выключении питания. По способу программирования делятся на масочные (собственно, ПЗУ) и перепрограммируемые (ППЗУ или PROM) [4].

ПЗУ – постоянное запоминающее устройство масочного типа, информация в которое записывается при изготовлении ИМС на предприятии с помощью шаблона маски на завершающем этапе технологического устройства [6]. Масочные ПЗУ реализуются на базе диодных или транзисторных матриц.

Элементами связи в масочных ЗУ могут быть диоды и транзисторы. Программирование памяти заключается в том или ином расположении элементов связи между горизонтальными и вертикальными шинами связи запоминающих элементов (рис. 4.9).

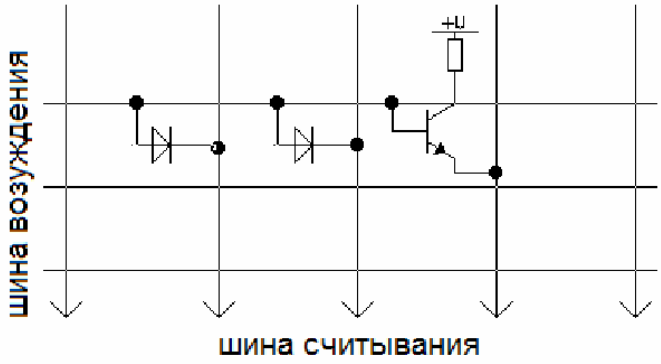


Рис. 4.9. Структура ПЗУ

В матрице диодного ПЗУ горизонтальные линии являются – линии выбора слов, а вертикальные – линии считывания. Читаемое слово определяется расположением диодов в узлах сетки. При наличии диода высокий потенциал выбранной горизонтальной линии передается на соответствующую вертикальную линию в данном разряде слова имеем «1». Запись информации осуществляется путем выжигания ненужных диодов лазерным/электронным лучем или электрическим током на заводе в процессе изготовления микросхемы. В дальнейшем записанная информация изменена быть не может. ЗУ с матричным программированием отличается высоким уровнем интеграции. В качестве элементов связи в масочных ЗУ могут быть диоды, МОП- или биполярные транзисторы. Более быстрое ПЗУ реализуется на биполярных и МОП-транзисторах. Информация записывается путем нанесения или отсутствия металлизации стока затвора. Элемент масочного ЗУ, $R3 \approx R2 >> R1$ (рис. 4.10).

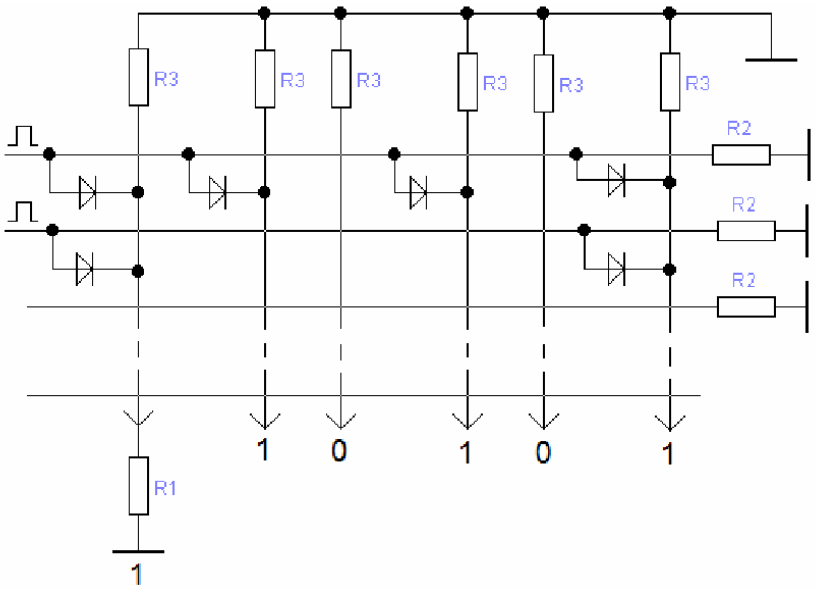
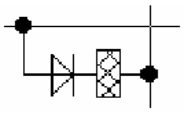


Рис. 4.10. Пример реализации масочного ЗУ на диодной матрице

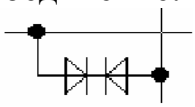
Серийно выпускаются также однократно программируемые ЗУ. Они делятся на два вида:

- настраиваемое на единицу;
- настраиваемое на нуль.

В ПЗУ микросхемы программируются путем устранения или создания специальных перемычек. В исходной заготовке имеются или отсутствуют все перемычки, после программирования остаются или возникают необходимые. Fuse PROM – прожигание, antifuse PROM – восстановление. Fuse PROM может быть металлический сплав титан-вольфрам или поликристаллический кремний. В исходном состоянии логический элемент хранит уровни логических единиц, а логический ноль записывается расплавлением перемычки. Иногда перемычки создаются в виде тонких непроводимых диэлектрических слоев – конденсаторов, которые при программировании пробиваются:



Иногда используют следующее соединение:



на стадии изготовления – ток в цепи отсутствует, при программировании один из диодов прожигается.

В качестве примера рассмотрим ПЗУ на плавких перемычках в многоэмиттерном транзисторе (рис. 4.11). Программирование осуществляется пользователем в специальном программаторе только один раз – перед началом эксплуатации (рис. 4.12).

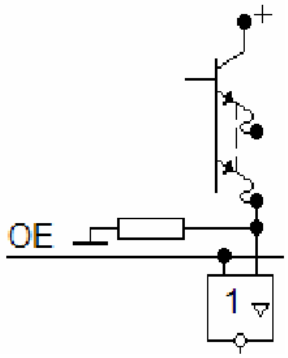


Рис. 4.11. Работа ПЗУ – режим считывания – с плавкими перемысками

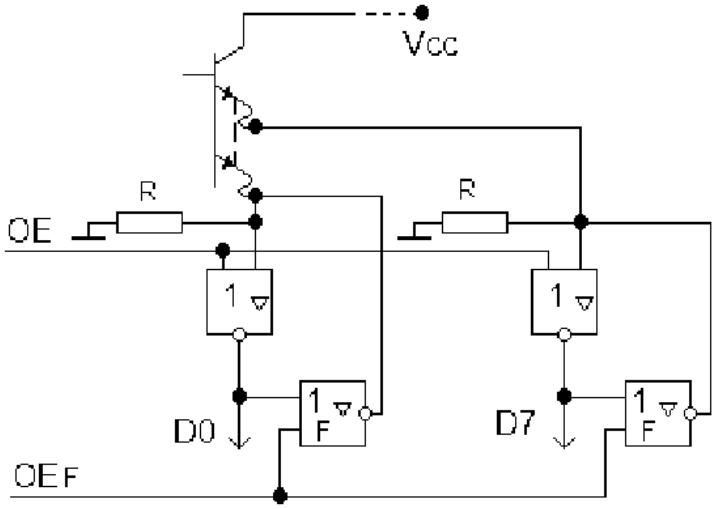


Рис. 4.12. Программирование пользователем ПЗУ с плавкими перемычками

Сигнал F вырабатывается программатором. Для пережигания перемычки на них подают токи около 10 мА в виде серии импульсов.

При больших объемах производства масочное программирование предпочтительнее. При малой тиражности ЗУ затраты на ЗУ велики и не рентабельны. Отсюда и следует применение масочных ЗУ в качестве хранителя стандартной информации, имеющей широкий круг потребителя. Пример масочных ЗУ – прошивка кодов букв алфавитов, телефонной клавиатуры, таблиц типов функций (калькулятор, стандартное программное обеспечение).

Перепрограммируемое постоянное запоминающее устройство (ППЗУ) в процессе функционирования МС используется как ПЗУ. Оно отличается от ПЗУ тем, что допускает обновление однажды занесенной информации, т.е. в нем предусматривается режим записи. Однако в отличие от ОЗУ запись информации требует отключения ППЗУ от цифрового устройства, производится с использованием специальных предназначенных для записи устройств (программаторов) и занимает длительное время, достигающее единиц и десятков минут.

В настоящее время существуют такие ППЗУ, которые допускают перепрограммирование прямо в схеме МС и не требуют специального программатора. ППЗУ дороже ПЗУ, и их применяют в процессе отладки

программы, после чего их можно заменить более дешевым ПЗУ при производстве больших серий МС.

Уровень интеграции ППЗУ ниже по сравнению с ПЗУ, но простота программирования пользователем и невысокая стоимость определили широкое распространение ППЗУ. Перепрограммируемые ПЗУ бывают двух типов:

- электрически стираемые (EPROM);
- с ультрафиолетовым стиранием (RPROM).

Электрически стираемые изготавливаются по МОП-технологии. При подаче на затвор транзистора большого управляющего импульса на транзисторе фиксируется уровень заряда, который сохраняется при снятии программирующего импульса. Амплитуда программирующего импульса в несколько раз превышает амплитуду обычных логических уровней сигналов. При подаче на затвор транзистора импульса обратной полярности его заряд стекает и транзистор вновь переходит в непроводящее состояние. Таким образом осуществляется перепрограммирование.

ППЗУ с ультрафиолетовым стиранием строятся на МОП ячейках с плавающим затвором. Режим программирования осуществляется аналогично электрически стираемым ППЗУ, однако перепрограммирование нельзя осуществить подачей импульса обратной полярности. Для стирания информации матрица облучается ультрафиолетовым излучением определенного спектра в течение нескольких десятков минут, которое переводит все транзисторы обратно в непроводящее состояние.

Работа в режиме считывания аналогична статическому ОЗУ, изложенному выше в п. 4.1.1. Программирование запоминающей ячейки осуществляется при единичном уровне сигнала на входе OE и нулевом уровне сигнала на входе CS путем подачи напряжения программирования 20 В на специальный вход UPR и нулевого импульса программирования длительностью 50 мс на специальный вход PR. Режим программирования существенно отличается от режима чтения и в общем случае не может быть реализован в работающей системе, так как для большинства микросхем требуется подача напряжения на вход «режим программирования» достаточно высокого уровня. Программирование подобных микросхем осуществляется в специальных устройствах, называемых программаторами ППЗУ.

При построении универсальных сложных МС могут применятся также ППЗУ с электрическим способом перезаписи, которые могут перепрограммироваться прямо в схеме МС – в процессе ее работы, такие микросхемы применяются в современных ПК для перезаписи программы начальной инициализации и базовых подпрограмм ввода/вывода BIOS.

Следующей ступенью интеграции применения ППЗУ является так называемый «твердотельный жесткий диск» – Flash-память. Это сложное полнофункциональное устройство ввода-вывода, основанное на электрически перепрограммируемом ПЗУ, которое может быть

перепрограммировано непосредственно в работающей системе [4]. Flash-технология была анонсирована фирмой Intel в 1989 году. Flash-память имеет внутри себя встроенный управляющий микроконтроллер. В обычном режиме чтения, в который Flash-память переходит сразу после включения питания, может производиться чтение как отдельных ячеек памяти, подобно обычному ПЗУ, так и чтение целых массивов – блоков памяти. Для того, чтобы очистить блок памяти или записать байт необходимо сначала задать последовательность команд внутреннему контроллеру. Процедуры очистки и записи внутренний контроллер осуществляет самостоятельно.

4.2. Программируемые логические матрицы

Программируемые логические матрицы (ПЛИС, PLD – programmable logic device) – это особый вид интегральных микросхем, предназначеных для преобразования комбинаций входного кода в определенную комбинацию выходного кода в соответствии с заданными булевыми функциями. Типовая структурная схема ПЛИС, в которой в качестве аргументов булевых функций участвуют только входы микросхемы приведена на рис. 4.13.

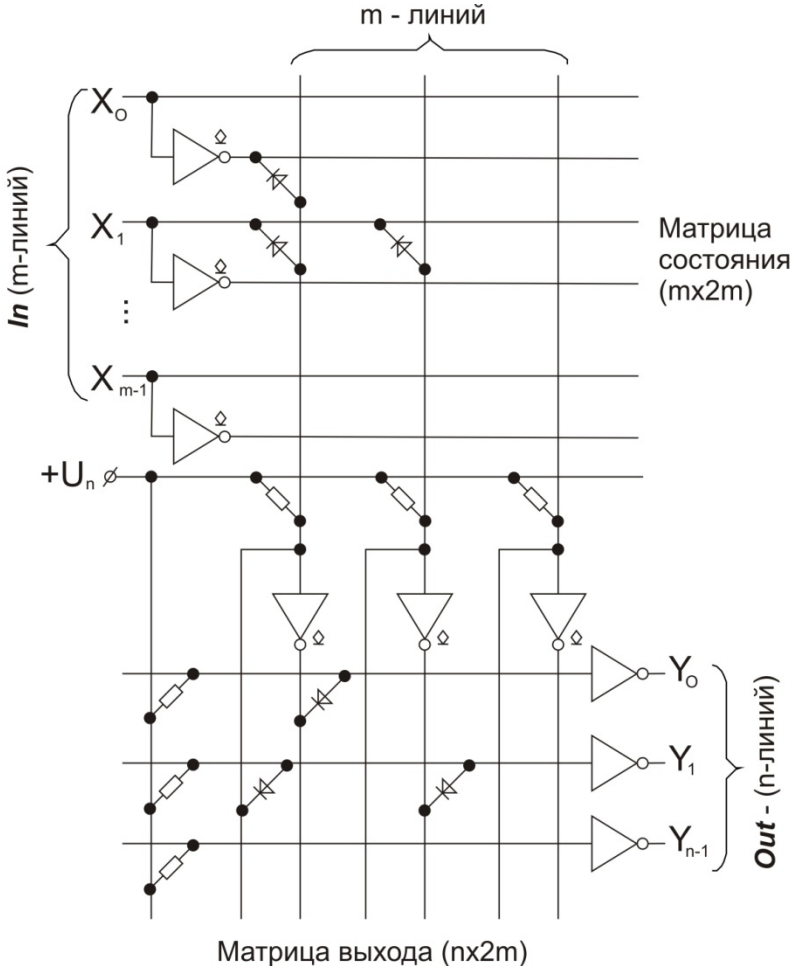


Рис. 4.13. Структура ПЛИС

ПЛИС имеет m входов X и n выходов Y и состоит из двух диодных матриц пересечения, таких же, как и в ПЗУ: матрицы состояния размером (m x $2m$) и

матрицы выхода размером (n x $2m$). Точки пересечения линий матрицы, тех линий, где восстановлены (прожжены) диоды, определяются конкретной минимизированной системой булевых функций и задаются в процессе программирования ПЛИС, осуществляемой в специализированном программаторе вне электронной схемы, в которой матрица в дальнейшем будет функционировать. Диоды, объединяющие две и более входные линии, реализуют функцию ИЛИ (конъюнкцию). Наличие инверторов позволяет заменить операцию И (дизъюнкцию) по правилу де Моргана операцией конъюнкции. Так на рис. 4.13 диодами задана следующая булева функция: $y_1 = \overline{\overline{x_0} + x_1} = x_0 \cdot \overline{x_1}$.

Программируемые логические матрицы применяются в качестве конечных автоматов, устройств управления с аппаратной реализацией, логических преобразователей адресов, управляющей памяти и др. В последнем примере, действительно, работа ПЛИС аналогична работе ПЗУ, тогда в качестве входов используется ША, а в качестве выходов – ШД. Однако, для большинства задач, для которых и предназначены ПЛИС, число возможных логических состояний соизмеримо с числом аргументов логических функций (входов ПЛИС). Для этих задач использование ПЗУ не эффективно, так как емкость ЗУ – число его ячеек разрядностью n – соответствует $N=2^m$, что многократно превышает требуемое число логических состояний. Тогда, как у ПЛИС, число узлов обоих матриц равно ($2m$ x m)+(n x $2m$)=(m+n) x $2m$, что много меньше, чем 2^m.

Например, требуется сформировать 8 выходов, состояние которых описывается 8-ми булевыми уравнениями от 16-ти аргументов (16 входов). При реализации на ПЗУ потребуется 2^{16}=65 536 8-ми разрядных ячеек памяти, т.е. 64 Кбайт памяти, а при реализации на ПЛИС потребуется всего 16*32+8*32=768 ячеек. Таким образом, большинство ячеек памяти ПЗУ не будут использованы, тогда как все ресурсы ПЛИС (для этого примера) используются в полном объеме.

Часто, для реализации сложных алгоритмов функционирования, которые основаны не только на комбинационных, но и на последовательностных принципах, необходимо, чтобы аргументами булевых функций ПЛИС были как входы так и выходы микросхемы. Для этого в структуру ПЛИС добавляется еще одна матрица – матрица обратной связи размером (m x $2n$) (рис. 4.14).

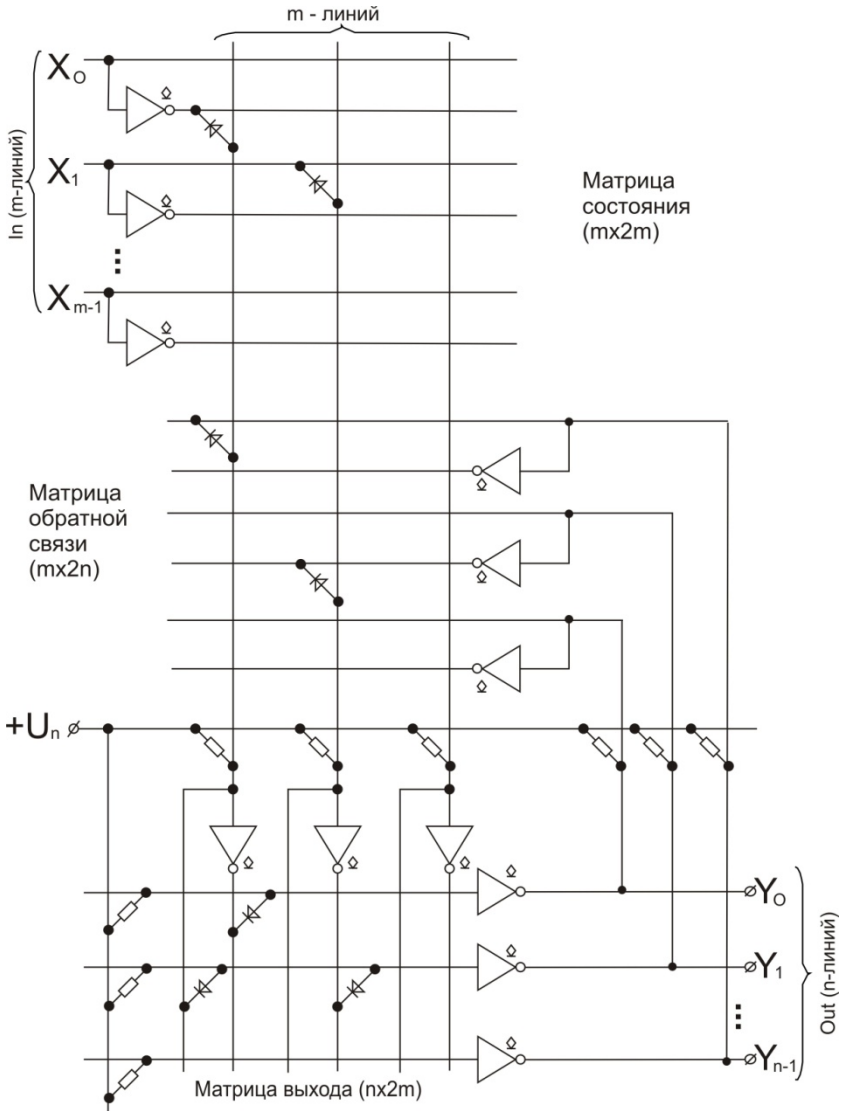


Рис. 4.14. Структура ПЛИС с ОС

Благодаря введению дополнительной матрицы можно реализовать функцию триггера, т.е. последовательностное устройство. Например, при поступлении на вход короткого импульса (динамический вход) можно установить выход в нужное состояние на длительное время (статический выход), а сброс этого выхода можно организовать от другого события: $!Y_1=(!X_1+!Y_1)\ \&\ X_0$, где «!» – инверсия (логическое отрицание), «+» – логическое ИЛИ, «&» – логическое И. В приведенном примере активный уровень выхода Y_1 – логически «0» – устанавливается при поступлении хотя бы короткого импульса (по его отрицательному фронту) на вход X_1, а снимается активный уровень выхода Y_1, уже независимо от состояния входа X_1, только после снятия активного уровня сигнала X_0. Это ни что иное, как реализация функции простейшего триггера.

Ниже приведен пример листинга прошивки для решения задачи управления весовым дозатором дискретного действия. Используемая ПЛИС – микросхема ATF16v8 фирмы Atmel с ОС, допускающая многократное перепрограммирование пользователем и имеющая логическую емкость 8

выходов (они же могут быть и входами) и 10 входов, активный уровень у всех сигналов – «0». Вначале в листинге описываются все пины – входы и выходы – микросхемы (pin 10 и 20 не описываются – это питание микросхемы), а затем, собственно, и записываются логические уравнения для каждого из выходов. Далее, полученный текстовый файл подвергается трансляции, в результате которой логические уравнения преобразуются в минимизированную систему булевых функций и создается двоичный файл прошивки ПЛИС. Двоичный файл должен отвечать формату используемого программатора, поддерживающего прошивку ПЛИС ATF16v8.

```
% CHIP  ATF16v8

% INPUTS
pin    1     meandr      %меандр для синхронизации работы
pin    2     cikl        %цикл дозирования
pin    3     stabil      %стабильность показаний веса
pin    4     lev1        %вес достиг уставки 1 – бункер пуст
pin    5     lev2        %вес достиг уставки 2 – бункер полон
pin    6     start_in    %внешний сигнал –"пуск дозатора"
pin    7     stop_in     %внешний сигнал –"стоп дозатора"
pin    8     run         %разрешение работы дозатора
pin    9     in          %не задействован
pin    11    in          %не задействован

% OUTPUTS
pin    12     stop       %команда "стоп цикла дозирования"
pin    13     start      %команда "старт цикла дозирования"
pin    14     pusk       %команда "пуск дозатора"
pin    15     doza       %внешний выход "доза набрана"
pin    16     load       % внешний выход "загрузка бункера"
pin    17     unload     %внешний выход "разгрузка бункера"
pin    18     out        %не задействован
pin    19     out        %не задействован

% EQUATIONS
!stop := !doza  &  stabil  &  load  &  cikl  &  run  &  meandr;
!start := lev1  &  stabil  &  unload  &  !cikl  &  run  &  meandr;
!pusk := ( !start_in  +  !pusk)  &  stop_in  &  ( !cikl  +  !load)  &  run;
!doza := ( !lev2  +  !doza )  &  cikl  &  run;
!load := lev2  &  doza  &  cikl  &  unload  &  run;
!unload := !lev1  &  !cikl  &  load  &  run;
```

ПЛИС функционирует совместно с дозирующим контроллером, производящим взвешивание бункера и учет материала. Алгоритм работы

дозатора следующий. Дозатор запускается «!pusk» в автоматическом режиме при поступлении внешнего сигнала «!start_in» (кратковременное нажатие кнопки или срабатывание внешнего датчика). Цикл дозирования запускается «!start», когда поступил сигнал со стороны дозирующего контроллера «!cikl» при условии того, что бункер пуст «lev1» (вес ниже уставки 1) и окончена разгрузка бункера «unload» по предыдущему циклу. Формируется внешний выходной сигнал на загрузку бункера «!load». После заполнения бункера срабатывает входной сигнал «!lev2» (вес выше уставки 2) и загрузка материала прекращается «load». Затем, после успокоения показаний «!stabil», цикл дозирования заканчивается «!stop», производится учет материала в бункере (дозирующий контроллер снимает сигнал – «cikl») и осуществляется выгрузка бункера «!unload». Далее, вновь производится загрузка и все повторяется в цикле до тех пор, пока не поступит внешний сигнал «!stop_in», тогда заканчивается текущий цикл дозирования и работа автоматического дозатора останавливается. В этом примере матрица ОС ПЛИС используется для формирования сигналов «!pusk» и вспомогательного сигнала «!doza», необходимого для фиксации набранной дозы даже при кратковременном достижении уставки 2 «!lev2». Это связано с тем, что после прекращения загрузки «load» вес в бункере может колебаться вокруг заданного значения (уставка 2).

Вопросы для повторения

1. Указать основные характеристики ЗУ.
2. Привести классификационные признаки, разделяющие ЗУ по видам.
3. Дать сравнительную оценку построения ЗЯ по триггерной схеме и на основе емкостного элемента.
4. Указать особенности словарной и матричной организации памяти.
5. Какие функциональные блоки входят в структуру ЗУ с матричной организацией?
6. Какие виды ОЗУ используются в МС?
7. Что такое масочное ПЗУ?
8. Что такое fuse- и antifuse-PROM?
9. Какие виды ППЗУ Вы знаете?
10. Каково назначение ПЛИС?
11. Из каких матриц пересечения состоит ПЛИС?
12. Обосновать, почему для построения логических УУ более эффективно использовать ПЛИС, а не ПЗУ?
13. Зачем используется матрица ОС в ПЛИС?
14. Из каких разделов состоит листинг прошивки ПЛИС? Приведите пример.

Тема 5. Специализированные процессоры и микроконтроллеры

Цели и задачи изучения темы

Основной целью изучения данной темы является знакомство с видами специализированных процессоров и микроконтроллеров, с принципами построения программируемых логических контроллеров (ПЛК) проектирования распределенных информационно-управляющих систем, с подходами при реализации функций реального времени, устройством и особенностям программирования цифровых сигнальных процессоров.

Задачи:

1. Рассмотреть устройство и архитектуру специализированных микроконтроллеров в задачах управления в реальном времени;
2. Ознакомится с видами и устройством программируемого логического контроллера и модулей ввода вывода;
3. Изучить основные подходы при построении таймеров и реализации функций реального времени;
4. Рассмотреть устройство и принципы построения процессоров цифровой обработки сигналов.

5.1. Архитектура микроконтроллера и ПЛК

Основной особенностью современного этапа развития МС является завершение перехода от систем, выполненных на основе нескольких БИС, к однокристальным МК, которые объединяют в одном кристалле все основные элементы МС: центральный процессор (ЦП), постоянное запоминающее устройство (ПЗУ), оперативное запоминающее устройство (ОЗУ), порты ввода/вывода, таймеры [1].

Наиболее распространенным представителем семейства МК являются 8-разрядные приборы, широко используемые в промышленности, электрической и связи и коммутации, бытовой и компьютерной технике. Они прошли в своем развитии путь от простейших приборов с относительно слаборазвитой периферией до современных многофункциональных контроллеров, обеспечивающих реализацию сложных алгоритмов управления в реальном масштабе времени. Причиной жизнеспособности 8-разрядных МК является использование их для управления реальными объектами, где применяются, в основном, алгоритмы с преобладанием логических операций, скорость обработки которых практически не зависит от разрядности процессора.

Росту популярности 8-разрядных МК способствует постоянное расширение номенклатуры изделий, выпускаемых такими известными фирмами, как Motorola, Microchip, Intel, Atmel и многими другими.

Современные 8-разрядные МК обладают модульной организацией, при которой на базе одного процессорного ядра (центрального процессора)

проектируется ряд (линейка) МК, различающихся объемом и типом памяти программ, объемом памяти данных, числом портов ввода-вывода, частотой синхронизации и т. п.

Внутренняя архитектура МК как правило скрыта от пользователя и характеризуется отсутствием линий магистралей адреса и данных на выводах корпуса МК. Таким образом, МК представляет собой законченную систему обработки данных, наращивание возможностей которой с использованием параллельных магистралей адреса и данных не предполагается.

Микроконтроллер, как было указано выше, является полноценным узлом обработки данных в интегральном исполнении – БИС. Однако, для его использования в составе управляющих систем необходимо иметь законченное устройство, имеющее определенный вид исполнения и эксплуатационные характеристики, которые задаются областью применения (например, промышленное исполнение). Таким изделием на базе МК является программируемый логический контроллер (ПЛК, PLC – Programming Logical Controoller), который предназначен для ввода внешних аналоговых и дискретных сигналов, их обработке согласно прикладной программе пользователя и выдачи выходных и управляющих сигналов (рис. 5.1).

Рис. 5.1. Программируемый логический контроллер

Первые логические контроллеры появились в виде шкафов с набором соединённых между собой реле и контактов. Эта схема задавалась жёстко на этапе проектирования и не могла быть изменена далее. В первых ПЛК, пришедших на замену обычным логическим контроллерам, логика соединений программировалась схемой соединений (LD – Ladder logic Diagram). То есть устройство имело тот же принцип работы, но реле и контакты (кроме входных и выходных) были виртуальными, то есть существовали в виде программы микроконтроллера.

В зависимости от выполняемых функций ПЛК в прикладной программе преобладают либо логические, либо числовые операции. В первом случае наиболее часто используются ПЛК с акцентом именно на логические свои возможности. Такие ПЛК имеют, как правило, встроенный специализированный язык программирования, ориентированный на программно-логическое управление, и преобладающее количество дискретных входов-выходов над аналоговыми.

В системах, алгоритмы управления которых содержат большое количество числовых операций, в особенности операций с плавающей

точкой, широкое применение нашли PC-совместимые контроллеры, позволяющие производить достаточно сложные вычисления и программирование на алгоритмических языках (Си, Паскаль). Однако, в большинстве приложений преобладают все же логические операции, поэтому за всеми этими устройствами остаётся название ПЛК.

В отличие от универсальных микропроцессоров и построенных на их основе компьютеров, в которых обычно доступна только байтовая или словарная адресация, в ПЛК – обеспечивается доступ к отдельным битам памяти.

ПЛК не имеют развитых средств интерфейса типа клавиатуры с полным набором клавиш и SVGA-дисплея, они устанавливаются в шкафы управления, их программирование, диагностика и обслуживание производятся по средствам компьютера со специальным программным обеспечением, подключенного по сети либо через нестандартный или стандартный интерфейс, например COM-порт или USB. В системах управления ПЛК взаимодействуют с системами человеко-машинного интерфейса: операторскими панелями и АРМ операторов на базе компьютеров.

ПЛК делятся на централизованные и периферийные (локальные). Централизованный ПЛК монтируются в шкаф управления, расположенный в центральной пультовой, а вся периферия подключаются к нему через устройства сопряжения с объектом отдельными проводами. Это устаревший подход, так как получается слишком сложный монтаж и много проводов, а также низкая надежность и помехоустойчивость.

Современный подход заключается в распределенной структуре управления. Централизованный ПЛК выполняет роль интеллектуального контроллера либо контроллера среднего звена управления, который не связан непосредственно с периферийными устройствами. Централизованный ПЛК взаимодействует с нижним звеном управления – локальными контроллерами и модулями ввода-вывода – через полевую сеть (FieldBus), например, PROFIBUS с протоколом DP. Так как локальные контроллеры и модули ввода-вывода соединяются с источниками входных сигналов и приемниками выходных сигналов достаточно короткими проводниками, то обеспечивается высокая помехоустойчивость и надежность такой системы.

ПЛК содержат процессорное ядро, одинаковое для всех МК данного семейства, и изменяемый функциональный блок, который отличает ПЛК разных моделей.

Процессорное ядро включает в себя центральный процессор, внутреннюю контроллерную магистраль (ВКМ) в составе шин адреса, данных и управления, схему синхронизации, схему управления режимами работы МК, включая поддержку режимов пониженного энергопотребления, начального запуска (сброса).

Изменяемый функциональный блок включает в себя модули памяти различного типа и объема, порты ввода/вывода и др. В относительно простых

ПЛК схемы обработки прерываний могут входит в состав процессорного ядра. В более сложных ПЛК обработка прерываний может производиться в отдельном модуле с развитыми возможностями. В состав изменяемого функционального блока могут входить и такие дополнительные схемы как компараторы напряжения, аналого-цифровые преобразователи (АЦП) и др. Каждый элемент в составе ПЛК проектируется с учетом протокола ВКМ микроконтроллера. Данный подход позволяет создавать разнообразные по структуре ПЛК в пределах одного семейства.

Структура ПЛК приведена на рис. 5.2.

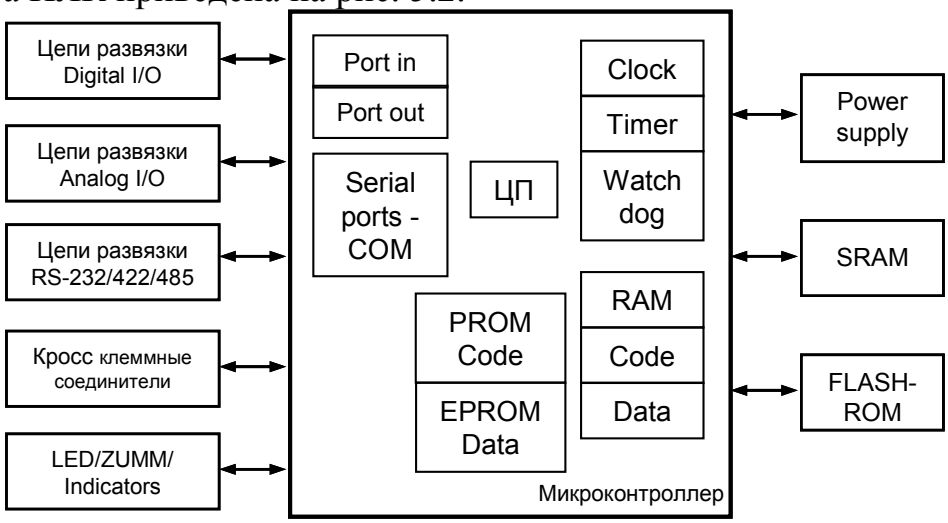


Рис. 5.2. Структура ПЛК

В структуру микроконтроллера ПЛК входят:

- ✓ ЦП – центральный процессор микроконтроллера, может иметь RISC-архитектуру (например, МК – PIC 16) или CISC-архитектуру (например, МК – AMD 80188-40, имеет систему команд, аналогичную x86);
- ✓ RAM – оперативная память (ОЗУ) для гарвардской архитектуры разделена на память программ Code Segment и память данных Data Segment;
- ✓ PROM Code – постоянное запоминающее устройство (ПЗУ) для хранения программ начальной инициализации и библиотек ввода-вывода;
- ✓ EPROM Data – энергонезависимое электрически перезаписываемое ПЗУ в сегменте памяти данных Data Segment для сохранения параметров настройки и констант системы управления;
- ✓ Clock – внутренние часы, счетчики времени и таймеры (Timer), а также таймер специального назначения WatchDog;
- ✓ Port in/out – параллельные порты ввода и вывода;
- ✓ Serial ports – последовательные порты – COM-порты.

Внешние схемы обвязки микроконтроллера, входящие в состав ПЛК, делают его законченным устройством и определяют его конфигурацию в соответствие с конкретными функциональными задачами:

- ✓ цепи развязки Digital I/O – дискретные порты ввода-вывода с цепями гальванической развязки;

- ✓ цепи развязки Analog I/O – аналоговые входы и выходы с цепями защиты от переполюсовки, перенапряжения и цепи искрогашения. Для ввода и вывода аналоговых сигналов в структуре ПЛК дополнительно должны присутствовать АЦП и ЦАП;
- ✓ цепи развязки RS-232/422/485 – последовательные COM-порты, обеспечивающие электрические параметры интерфейсов RS-232, RS-422 или RS-485;
- ✓ LED/ZUMM/Indicators – доступные для программиста органы сигнализации и индикации: светодиоды (LED), цифровые сегментные индикаторы (Indicators), зуммер (ZUMM);
- ✓ SRAM – внешнее по отношению к МК ОЗУ статического типа – расширение для хранения больших по объему программ или данных;
- ✓ FLASH-ROM – внешний твердотельный Flash-накопитель (сменная Flash-карточка), для хранения файлов и папок.

Кроме основных функциональных блоков в ПЛК имеется собственный преобразователь напряжения питания (Power supply), для снабжения электропитанием всех внутренний цепей, возможно с несколькими значениями напряжения, от одного внешнего источника питания, например =24 В или ~220 В. Также ПЛК снабжен внешними клеммными соединителями и стандартными разъемами, обеспечивающими надежный монтаж в шкафах управления.

В зависимости от конфигурации распределенной системы управления дискретный и аналоговый ввод-вывод (Digital I/O, Analog I/O) может быть вынесен из состава ПЛК на уровень модулей ввода-вывода, которые соединяются с ПЛК по полевой шине (fieldbus). Такая конфигурация обеспечивает лучшую распределенность и надежность системы.

Число последовательных портов определяется решаемой задачей. Обычно один из последовательных портов RS-232 отводится для программирования ПЛК, к нему подключается ПК через свой стандартный COM-порт по средствам установленного ПО, которое предназначено для подготовки и внедрения прикладных программ в ПЛК. В качестве такого ПО может быть использована специальная программа, выпускаемая производителем ПЛК, либо проблемно-ориентированная программная среда, взаимодействующая с конкретным ПЛК через его OPC-сервер, поставляемый производителем ПЛК. Остальные последовательные интерфейсы могут иметь любой тип, в том числе и порт Ethernet и USB. Часто, порты типа RS-232, RS-422 и RS-485 могут быть выбраны пользователем путем установления механических перемычек на печатной плате ПЛК при его инсталляции.

Производители обычно выпускают не один универсальный ПЛК, а целую серию – ряд ПЛК, в которой контроллеры отличаются числом и типом последовательных портов, количеством дискретных и аналоговых входов и выходов, объемом внешнего ОЗУ и Flash-накопителя, наличием светодиодов, *n*-разрядного индикатора и зуммера. Таким образом, под конкретную задачу

проектировщик системы управления можно подобрать ПЛК из нужной серии, оптимальный по отношению цена/возможности.

Построение встроенных (управляющих) систем на базе ПЛК предполагает использование типовых функциональных периферийных модулей (таймеров, процессоров событий, контроллеров последовательных интерфейсов, аналого-цифровых преобразователей и др.). Производители ПЛК выпускают в составе каждой серии, которая предназначена для решения определенного круга задач управления, на ряду с линейкой ПЛК (отличающихся набором портов ввода-вывода и объемом памяти) и различные модули ввода-вывода, позволяющие проектировщику подобрать полный комплект устройств для решения конкретной прикладной задачи. Причем, для одного семейства ПЛК модули ввода-вывода могут выпускаться различными производителями, так как многие модули являются унифицированными по электрическим, информационным, эксплуатационным, метрологическим и конструктивным характеристикам.

Модули ввода-вывода осуществляют сбор и первичную обработку данных, формируют выходные сигналы. Связь между ПЛК и модулями ввода-вывода осуществляется через полевую сеть (FieldBus) с интерфейсом PROFIBUS, BITBAS, CANBUS, LonWorks, MODBUS и т.п.

Требования, предъявляемые к модулям ввода-вывода:

- надежность и помехоустойчивость;
- гальваническая развязка цепей связи с объектом;
- буферизация данных, выборка и хранение;
- запоминание состояний выходов при аварийном отключении;
- поддержка унифицированных сигналов и стандартных интерфейсов.

Модули ввода данных по функциональному назначению делятся на:

- ✓ модули дискетного ввода Digital In (DI), предназначены для ввода потенциальных сигналов типа «вкл-выкл» и имеют гальваническую развязку. Выпускаются также модули дискретного ввода типа «сухой контакт», когда входом является электромеханический контакт, имеющий два состояния: замкнуто и разомкнуто;
- ✓ модули аналогового ввода, содержащие в своей структуре одноканальные или многоканальные АЦП. Работают модули либо с унифицированным сигналом электрического напряжения или тока (например, диапазон 0…10 В или 4…20 мА), либо с определенным типом датчика, тогда в составе модуля имеется еще и преобразователь (например, специальные модули для работы с тензометрическими датчиками веса 2,0 мВ/В или с термометрами сопротивления). Параметры преобразователя настраиваются программно или с помощью перемычек на печатной плате;
- ✓ модули частотного ввода осуществляют подсчет числа внешних входных импульсов за определенный промежуток времени, либо их частоту следования. Эти модули могут применяться, в том числе, и для

ввода аналоговых сигналов, если аналоговый сигнал предварительно преобразован с помощью ЧИМ-модуляции в последовательность импульсов.

Модули вывода данных по функциональному назначению делятся на:

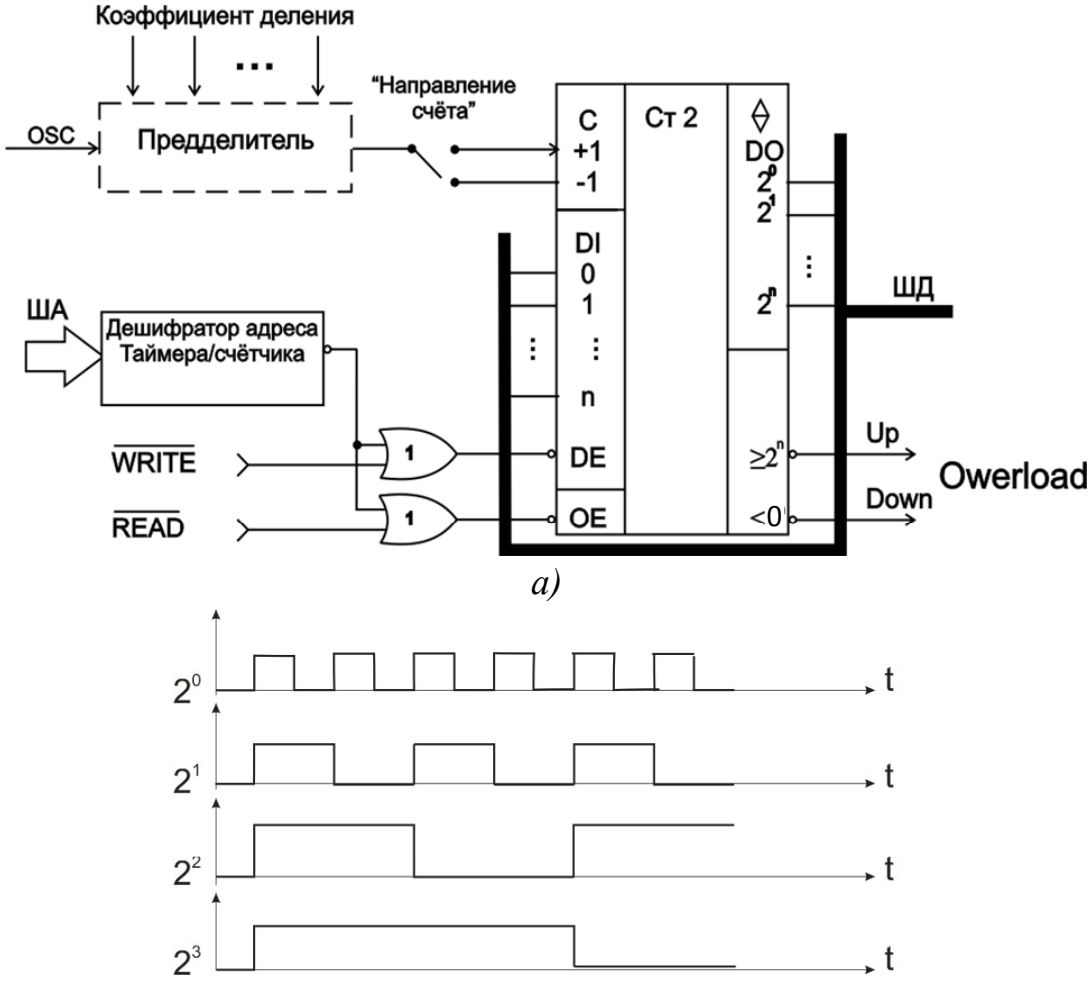
- ✓ модули дискетного вывода Digital Out (DO), предназначены для вывода сигналов типа «вкл-выкл». Применяются выходные схемы типа ОК (общий коллектор транзистора) либо типа «сухой контакт», когда выходом является контакт электромагнитного реле, имеющий два состояния: замкнуто и разомкнуто;
- ✓ модули аналогового вывода, содержащие в своей структуре ЦАП. На выходе модуля формируется унифицированный электрический сигнал напряжения или тока. Вид унифицированного сигнала преобразователя настраивается программно или с помощью перемычек на печатной плате.

Кроме вышеперечисленных модулей существуют также и специализированные модули, которые предназначены для работы с конкретным видом внешнего устройства, например модуль управления шаговым двигателем или специализированные многоканальные модули ШИМ, в частности для управления приводом (80C196MC Intel).

5.2. Реализация функций реального времени

Функции времени в ПЛК, обеспечивающие режим реального времени, реализованы на таймерах. Таймеры предназначены для быстрой генерации события на своем выходе, либо быстрой реакции на внешнее событие, поступающее на вход таймера, без участия процессора. Обычно в структуре ПЛК имеется несколько таймеров: таймер-часы реального времени, таймеры общего назначения, используемые программистом для организации взаимодействия задач реального времени, сторожевой таймер WatchDog и др.

Структурная схема типового таймера основана на работе двоичного счетчика СТ 2 (рис. 5.3-а), на вход которого поступает последовательность импульсов от встроенного тактового генератора – сигнал OSC. Коэффициент деления предделителя частоты доступен программисту и может задаваться в прикладной программе.



а)

б)

Рис. 5.3. Таймер-счетчик: а – структурная схема;
б – эпюры напряжений на выходах двоичного счетчика

Контроль реального времени осуществляется несколькими способами. Первый способ основан на использовании выходных сигналов переполнения (Owerload): переполнение сверху (Up) при прямом направлении счета; переполнение снизу (Down) при реверсном (обратном) счете. В счетчике может программно задаваться направление счета «+1» или «–1». Чаще, для этого способа используется обратный счет до нуля. Начальное значение счета, относительно которого должно фиксироваться переполнение, записывается в счетчик по шине данных во входной регистр счетчика DI (Data In) при наличии активного уровня на его входе DE (Data Enable). Сигнал переполнения инициализирует аппаратное прерывание микропроцессора, свидетельствующее о наступлении события типа «временной интервал». Этот способ отслеживания временного интервала называется «OWERFLOW INTERRUPT».

Второй способ заключается в считывании с выходного регистра счетчика DO (Data Out) параллельного кода (рис. 5.3-б), соответствующего прошедшему времени с момента начала счета. Как правило, счет ведется от

нуля в прямом направлении. Выходной код счетчика выставляется на шину данных при наличии активного уровня на его входе OE (Output Enable). Выходной код таймера может быть использован в двух режимах.

Первый режим реализует операцию аппаратного сравнения цифровых кодов и называется «COMPARE INTERRUPT» (рис. 5.4) [4].

Рис. 5.4. Таймер-счетчик в режиме сравнения

В регистр времени события предварительно записывается по команде МП определенный код, соответствующий ожидаемому интервалу времени. Компаратор сравнивает каждый момент времени содержимое таймера и содержимое регистра времени события. В момент, когда их содержимое становится равным друг другу, выставляется запрос на прерывание типа «временной интервал». Таким образом, можно запрограммировать определённое выходное событие в заданное время и осуществить его без участия процессора.

Второй режим позволяет без участия процессора отслеживать входной сигнал, свидетельствующий о наступлении некоторого внешнего события, он называется «CAPTURE INTERRUPT» (рис. 5.5).

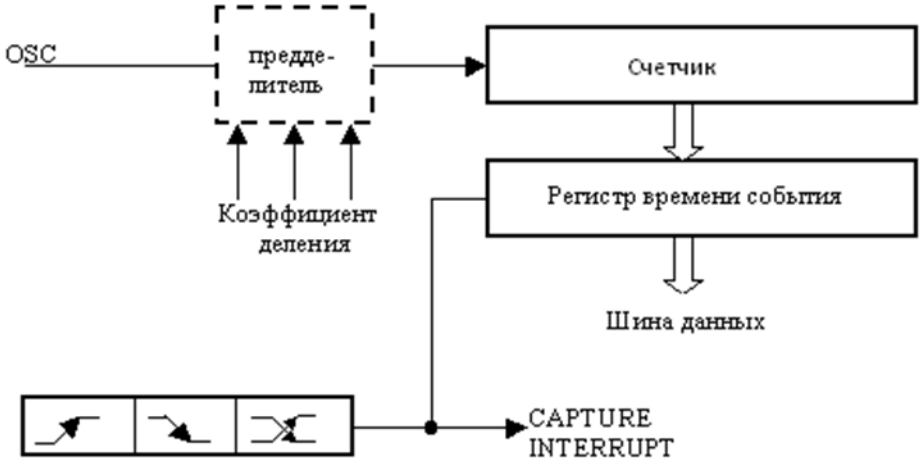


Рис. 5.5. Таймер-счетчик в режиме захвата внешнего события

Таймер отсчитывает синхроимпульсы с момента начала ожидания некоторого внешнего события. В момент, когда происходит заданное событие (поступает сигнал на определенный порт ввода микроконтроллера), генерируется аппаратное прерывание, при этом содержимое таймера фиксируется в регистре времени события. Таким образом, запоминается время события, которое затем может быть считано процессором по соответствующей команде в подпрограмме обработки прерывания. Этот

режим используется в частности для измерения частоты следования входных импульсов или длительности входного импульса – так называемый «***счетный вход***» микроконтроллера.

Важным устройством с точки зрения формирования управляющих воздействий на выходе ПЛК является ШИМ, осуществляющий преобразование цифрового кода в широтно-модулированную последовательность импульсов. Очень часто для этого применяется таймер, работающий в режиме «COMPARE INTERRUPT» (рис. 5.6).

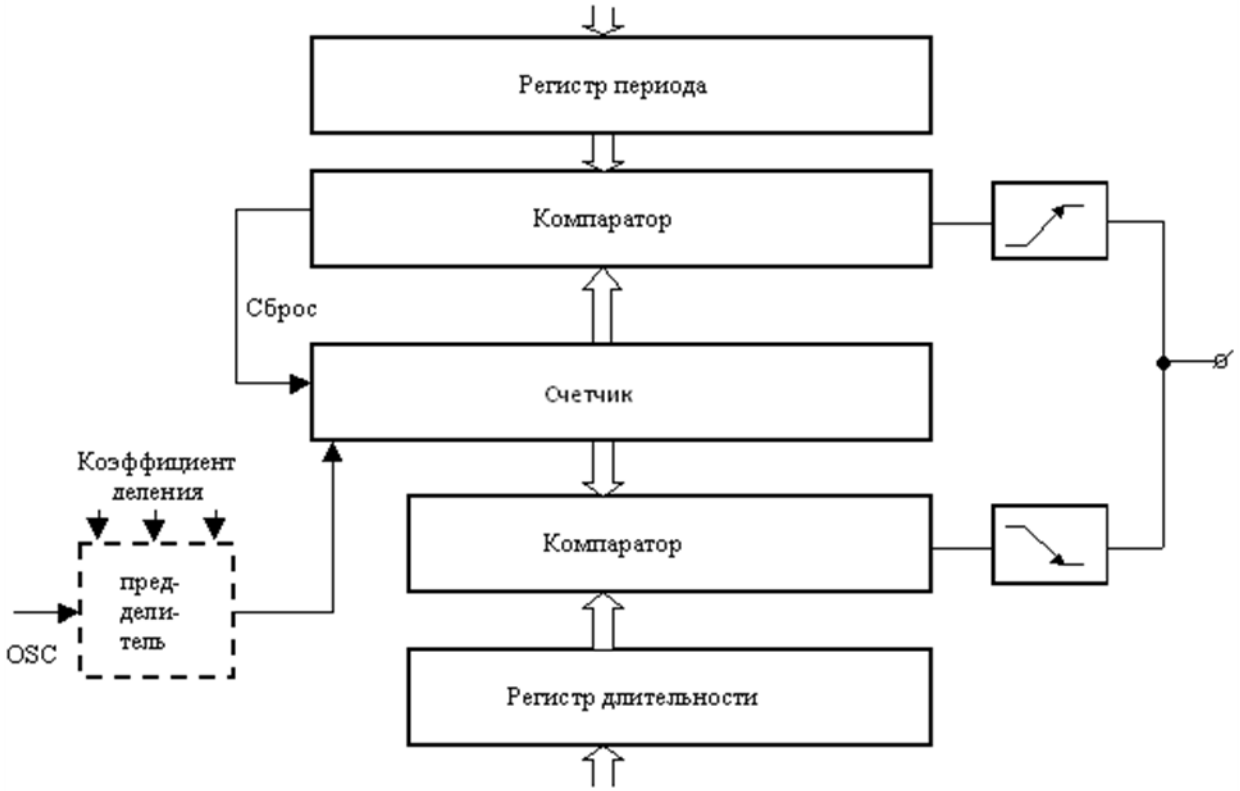


Рис. 5.6. Таймер-счетчик в режиме генерации ШИМ

В регистр периода процессор записывает число, соответствующее периоду ШИМ. При совпадении содержимого этого регистра и таймера, последний сбрасывается в нулевое состояние и на соответствующем выходе формируется положительный перепад. В регистре длительности процессор записывает число, соответствующее длительности импульса. При совпадении его содержимого с содержимым таймера, на выходе микросхемы формируется отрицательный перепад. Таким образом, на выходе таймера формируется последовательность прямоугольных импульсов, имеющих период, равный отношению числа из регистра периода к частоте входных синхроимпульсов OSC. Длительность импульсов равна отношению числа, записанного в регистре длительности, к частоте входных синхроимпульсов OSC. Скважность импульсов равняется отношению длительности импульсов к их периоду следования $\gamma = \frac{t_{имп}}{T}$, $(\gamma = 0 \ldots 1)$. Максимальный

период/длительность следования импульсов зависит от частоты OSC, ее коэффициента деления и разрядности регистра периода/длительности.

Таймер специального назначения – ***сторожевой таймер WatchDog*** служит для предотвращения «зависания» ПЛК, так как «зависание» может привести к нарушению режима реального времени и, как следствие, к отказу системы или даже аварийным ситуациям и выходу из строя оборудования. Если сторожевой таймер активизирован, то он отсчитывает периоды реального времени, соответствующие переполнению счетчика ($\geq 2^n$). Если в течение одного такого периода к сторожевому таймеру поступает сигнал сброса от выполняемой прикладной программы, таймер начинает отсчет нового периода (рис. 5.7).

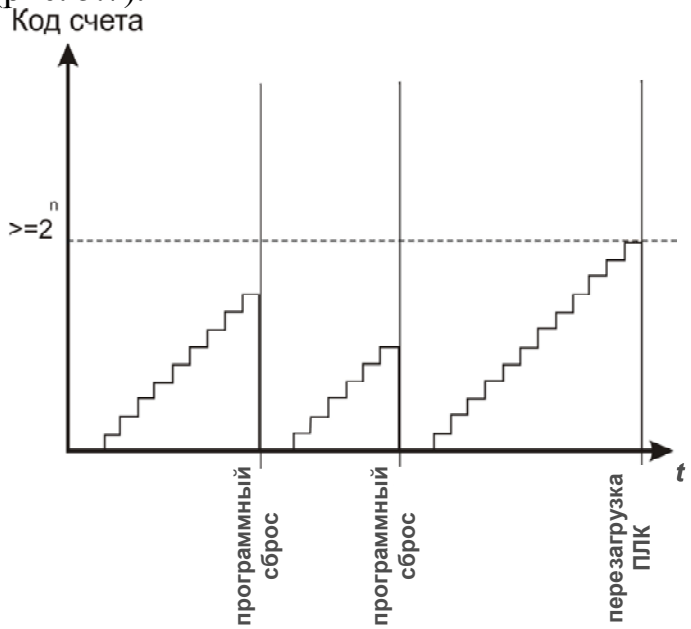


Рис. 5.7. Работа сторожевого таймера WatchDog

Если же по каким-то причинам (либо аппаратного сбоя, либо при некорректной работе прикладной программы), произошло «зависание», то до момента истечения периода таймера сигнал программного сброса не поступит, и WatchDog произведет аппаратную перезагрузку всего ПЛК. Для реализации контроля времени выполнения прикладных программ в их алгоритмах должны быть предусмотрены контрольные точки, в которых выдается сигнал сброса сторожевого таймера WatchDog. При перезагрузке автоматически запускается прикладная программа вновь, а так как параметры процесса управления сохраняются в энергонезависимой памяти, то после перезагрузки продолжается штатное функционирование системы и сбой практически не заметен. Обычно период счета сторожевого таймера выбирается в пределах десятков миллисекунд.

5.3. Процессоры цифровой обработки сигналов

Процессоры цифровой обработки сигналов (ПЦОС) находят широкое применение в самых различных областях, так, как они способны

обеспечивать работу в реальном масштабе времени как существующих, так и принципиально новых устройств [3].

Процессор цифровой обработки сигналов (***DSP*** – Digital Signal Processor) или просто ***сигнальный процессор*** – это специализированный микропроцессор, особенностью работы которого является поточный характер обработки больших объемов данных в реальном масштабе времени и, как правило, с интенсивным обменом данных с внешними устройствами. ПЦОС реализуется на основе так называемой базовой архитектуры (DSP Basic Architecture). Мы будем относить ПЦОС именно к специализированному процессору, а не контроллеру, так как его основная функция связана не управлением, а с обработкой данных, хотя по степени интеграции микросхема DSP схожа именно с микроконтроллером: кроме центрального процессора имеет память, порты ввода-вывода, таймеры и т. п. Если же DSP используется для управления, например электроприводам, то его смело можно назвать контроллером.

Первый ПЦОС – TMS320C10 был выпущен компанией Texas Instruments Inc. в 1982 году и благодаря целому ряду удачных технических решений сразу получил широкое распространение. Первоначально он был задуман как контроллер, способный эффективно управлять работой модема, а также для оборонных нужд.

Одним из основоположников ПЦОС стал ученый из США Джек Килби. За достижения в технологии полупроводников, совместно с российским ученым Ж. Алферовым, Д. Килби получил нобелевскую премию в области физики в 2000 году.

Анализ сложившегося рынка показывает, что доминирующие позиции в ближайшем будущем будут занимать крупные компании-производители ПЦОС, такие как Texas Instruments Inc., Analog Devices, Motorola и ряд других, которые способны создавать не только ПЦОС низкой себестоимости, но и ежегодно инвестировать новые разработки и ноу-хау, создавать принципиально новые модели и платформы.

В числе наиболее распространенных ПЦОС можно назвать изделия следующих компаний — Motorola (56002, 96002), Intel (i960), Texas Instruments Inc. (TMS320) и Analog Devices (21xx, 210xx). Выбор процессоров той или иной компании для реализации конкретного проекта – многокритериальная задача, и сформулировать более или менее четкую методику выбора практически невозможно.

Так, доля компании Texas Instruments Inc. на рынке микросхем для цифровой обработки сигналов вплотную приблизилась к 50%. Этому способствовало также состоявшееся в 2001 году слияние с компанией Burr-Brown, специализирующейся на выпуске микросхем АЦП и ЦАП для профессиональной обработки звука.

Цифровая обработка сигнала (Digital Signal Processing) – это арифметическая обработка в реальном масштабе времени

последовательности значений амплитуды сигнала, определяемых через равные временные промежутки.

Аналоговая обработка сигнала, традиционно применяемая в различных устройствах, является во многих случаях более практичным и дешевым способом достижения требуемого результата. Однако тогда, когда требуется высокая точность обработки данных, реализация устройства в весьма компактном и миниатюрном виде, достижение высокой стабильности характеристик устройства в различных температурных условиях функционирования, цифровая обработка оказывается единственно приемлемым решением.

Базовая архитектура ПЦОС – это совокупность характерных особенностей процессора, направленная на повышение его производительности и отличающая ПЦОС от микросхем других типов. Она обусловлена:

- широким использованием конвейерного режима работы;
- наличием специализированного устройства умножения;
- наличием специальных команд для цифровой обработки сигналов;
- реализацией короткого командного цикла;
- применением модифицированной гарвардской архитектуры.

Конвейер, как правило, применяется в целях повышения производительности ПЦОС. Так, в четырехкаскадном конвейере (например, в TMS320C2xx) ПЦОС может обрабатывать одновременно четыре команды, причем все команды находятся на разных стадиях выполнения. В четырехкаскадном конвейере предварительную выборку команды, дешифрирование, выборку операнда и исполнение команд можно осуществлять независимым образом. Пока производится предварительная выборка команды N, предыдущая команда N─1 дешифрируется, команда N─2 выбирает операнд, а N─3 исполняется.

Для ПЦОС характерным является наличие ***аппаратного умножителя***, позволяющего выполнять умножение двух чисел за один командный такт. В универсальных же процессорах умножение обычно реализуется за несколько тактов, как последовательность операций сдвига и сложения.

Другой особенностью ПЦОС является включение в систему ***специальных команд*** как, например, умножение с накоплением (MAC – Multiply And Accumulate): $C = A \cdot B + C$, с указанным в команде числом выполнений в цикле и с правилом изменения индексов используемых элементов массивов A и B; инверсия битов адреса, разнообразные битовые операции. В ПЦОС реализуется аппаратная поддержка программных циклов и кольцевых буферов, когда один или несколько операндов извлекаются из памяти в цикле исполнения команды.

В ПЦОС широко используются методы сокращения длительности командного цикла, характерные, в том числе, для универсальных RISC-процессоров.

ПЦОС различных компаний-производителей образуют два класса, существенно различающихся по цене: более дешевые ПЦОС обработки данных в формате с фиксированной точкой и более дорогие ПЦОС, аппаратно поддерживающие операции над данными в формате с плавающей точкой.

Использование данных в формате ***с плавающей точкой*** обусловлено несколькими причинами. Для многих задач, связанных с выполнением интегральных и дифференциальных преобразований, особую значимость имеет точность вычислений, обеспечить которую позволяет экспоненциальный формат представления данных. Алгоритмы компрессии, декомпрессии, адаптивной фильтрации в цифровой обработке сигналов связаны с определением логарифмических зависимостей и весьма чувствительны к точности представления данных в широком динамическом диапазоне.

Хотя использование DSP с плавающей точкой автоматически устраняет большинство проблем, связанных с точностью и переполнением, процессоры ***с фиксированной точкой*** остаются очень популярными для многих приложений, и поэтому при их использовании нужно обращать достаточное внимание на возможное переполнение, потерю результатов (выход результата операции за пределы разрядной сетки) и масштабирование операндов.

От DSP требуется высокая точность представления результата в регистре-аккумуляторе. Например, когда перемножаются два 16-битных слова, результат представляется 32-битным словом. Ядро некоторых сигнальных процессоров с фиксированной точкой имеет встроенный 40-битный аккумулятор, который обеспечивает большой запас суммирования без переполнения.

Работа с данными в формате с плавающей точкой существенно упрощает и ускоряет обработку, повышает надежность программы, поскольку не требует выполнения операций округления и нормализации данных, отслеживания ситуаций потери значимости и переполнения. Платой за дополнительные «комфорт и скорость» является высокая сложность функциональных устройств, выполняющих обработку данных в формате с плавающей точкой, необходимость использования более сложных технологий производства микросхем и, как следствие, дороговизна ПЦОС.

Модифицированной гарвардская архитектура допускает обмен содержимым между памятью программ и памятью данных, что расширяет возможности устройства. В DSP фирмы Analog Devices Inc. и команды и данные могут храниться в памяти программ. Например, в случае с цифровым фильтром коэффициенты могут храниться в памяти программы, а значения данных – в памяти данных. Поэтому коэффициент и значение данных могут выбираться в одном машинном цикле.

Помимо выборки коэффициента из памяти программ и значения отсчета из памяти данных, естественно, должна быть выбрана и сама команда

из памяти программы. Цифровые сигнальные процессоры компании Analog Devices решают эту задачу двумя способами. В первом случае к памяти программ обращаются дважды за время цикла. Во втором случае используется кэш-память программ (рис. 5.8).

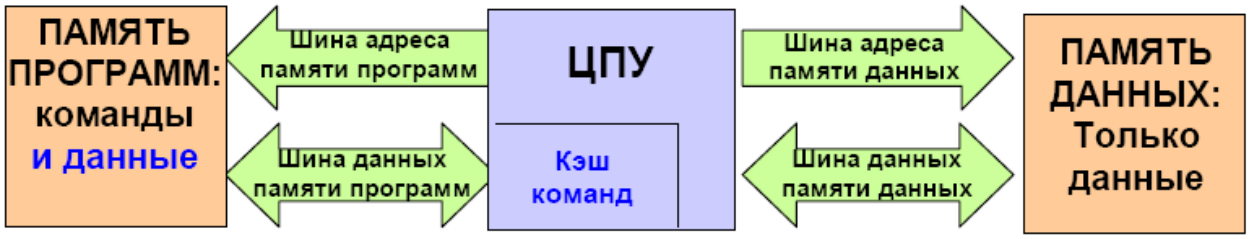


Рис. 5.8. Модифицированная гарвардская архитектура

В алгоритмах, требующих одновременной выборки двух операндов, программист один массив помещает в память программ, а другой – в память данных. Когда процессор в первый раз выполняет команду, требующую двойного обращения к памяти программ, используется один дополнительный машинный цикл, т. к. необходимо выбрать и команду, и коэффициент с помощью шины данных памяти программ. Однако когда возникает такая проблема, процессор помещает команду в кэш, и в следующий раз, когда требуется эта команда, извлекает ее из кэш-памяти, в то время как коэффициент поступает по шине данных памяти программ.

Целью быстрой одновременной выборки двух операндов является необходимость непрерывной загрузки накапливающего умножителя (MAC). При описании MAC было показано, что быстродействие DSP в основном определяется скоростью MAC. Если мы считаем, что MAC выполняется за приемлемое время, то очевидно, что для каждой операции требуется с той же скоростью подавать на MAC два операнда. Увеличение времени выборки операндов из памяти соответствующим образом отразится на скорости работы MAC. В идеале обращение происходит одновременно с выполнением операции в MAC в одном и том же машинном цикле.

Рассмотрим архитектуру ПЦОС с фиксированной точкой на примере микросхем семейства ADSP-21. При изучении архитектуры сигнального процессора целесообразно вначале рассмотреть архитектуру его ядра, построенную по модифицированному гарвардскому принципу, а затем – непосредственно архитектуру самого ПЦОС, включающую встроенную периферию. На рис. 5.9 приведена архитектура ядра процессора семейства ADSP-21.

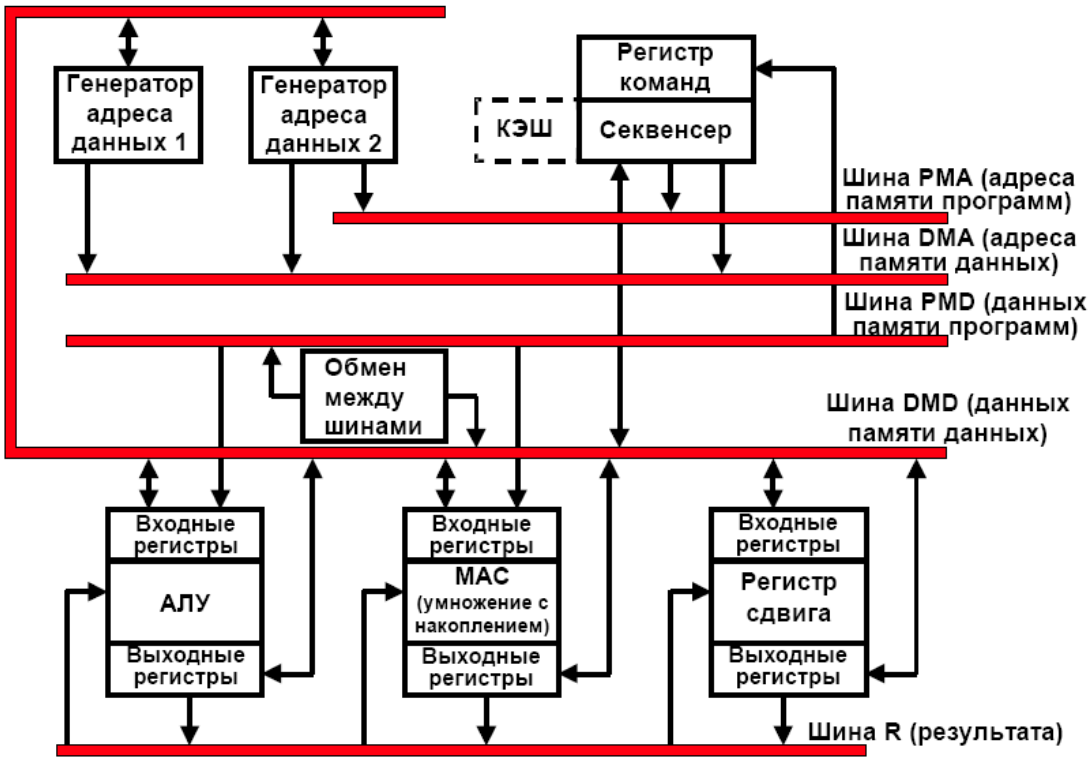


Рис. 5.9. Архитектура ядра процессора ADSP-21xx

Шины. Процессоры семейства ADSP-21 имеют пять внутренних шин для повышения эффективности передачи данных. Шины адреса памяти программы (PMA) и адреса памяти данных (DMA) используются одновременно для адресации в пределах адресных пространств памяти программ и памяти данных. Шины данных памяти программ (PMD) и шина данных памяти данных (DMD) используются для передачи данных из соответствующих областей памяти. При выводе шин на корпус за пределы кристалла они объединяются в одну внешнюю шину адреса и в одну внешнюю шину данных; области памяти выбираются соответствующими сигналами управления. Шина результата (R) используется для пересылки промежуточных результатов напрямую между различными вычислительными блоками. 14-разрядная шина PMA позволяет получить прямой доступ к 16К слов памяти программ. Шина DMD является 16-разрядной. Она обеспечивает доступ к содержимому любого регистра процессора для передачи его в любой другой регистр или любую ячейку памяти данных за один цикл. Адрес памяти данных может поступать от двух источников: абсолютное значение, содержится в коде команды (прямая адресация) или на выходе адресного генератора (косвенная адресация). Для выборки данных из памяти программ используется только косвенная адресация.

Шина данных памяти программы (PMD) является 8-разрядной и может также использоваться для передачи данных между вычислительными блоками напрямую или через блок обмена между шинами PMD-DMD. Блок обмена между шинами PMD-DMD позволяет передавать данные от одной

шины к другой. Он содержит технические средства, которые позволяют, когда это необходимо, преодолевать разницу в 8 бит между двумя шинами.

В памяти программ могут храниться как команды, так и данные, позволяя DSP семейства ADSP-21xx одновременно осуществлять выборку двух операндов в одном цикле, один из памяти программы, а другой из памяти данных. Команды поступают либо прямо из памяти программ путем осуществления двойного доступа в одном машинном цикле (процессоры серии ADSP-218x), либо из кэш-памяти программ (в процессоре серии ADSP-219x и SHARC).

Вычислительные блоки (АЛУ, MAC, регистр сдвига). Процессор содержит три независимых вычислительных блока: арифметико-логическое устройство (АЛУ), умножитель с накоплением (MAC) и устройство сдвига. Вычислительные блоки способны обрабатывать 16-разрядные данные и могут поддерживать вычисления с повышенной точностью. АЛУ обладает флагом переноса CI, который позволяет поддерживать 32-разрядные арифметические действия. АЛУ обеспечивает стандартный набор арифметических и логических функций: сложение, вычитание, смену арифметического знака, инкремент, декремент, получение абсолютного значения, логическое И, ИЛИ, исключающее ИЛИ и инверсию. Также поддерживаются примитивы деления.

MAC способен выполнить за один машинный цикл операцию умножения, умножения/сложения или умножения/вычитания. Он также содержит 40-разрядный аккумулятор, который обеспечивает дополнительные 8 разрядов для накопления результата без потери информации; данное решение допускает возникновение 256-ти переполнений, прежде чем произойдет потеря данных. Специальные команды обеспечивают поддержку блочной плавающей точки, при которой одна экспонента приписывается некоторому набору данных. Для ускорения обработки прерываний в MAC также может использоваться набор вспомогательных регистров. Если после окончания процедуры обработки сигнала установлен флаг MV, это означает, что регистр результата содержит слово, длина которого больше 32 разрядов. В этом случае значение, содержащееся в регистре, может быть заменено максимально или минимально возможной величиной, представимой в пределах 32-х разрядной сетки, в зависимости от природы переполнения.

Устройство сдвига производит операции логического и арифметического сдвига, нормализации и денормализации, а также вычисления блочной экспоненты (блочной плавающей точки). Нормализация – преобразование из формата с фиксированной точкой в формат с плавающей точкой, денормализация – обратное преобразование. Устройство сдвига имеет вспомогательные регистры для быстрого переключения контекста. Устройство может быть использовано для эффективного управления численными форматами, включая представление чисел в формате с плавающей точкой с повышенной точностью.

Вычислительные блоки располагаются параллельно, а не последовательно, так что результат работы любого блока может использоваться как исходное данное для другого блока в следующем машинном цикле. Для обеспечения такой возможности используется шина промежуточных результатов (R).

Адресные генераторы. Адресные генераторы (DAG) вычисляют адреса при перемещении данных из памяти в регистры и обратно. Каждый генератор DAG обладает четырьмя регистрами-указателями. Всякий раз, когда указатель используется для адресации данных (косвенная адресация), он модифицируется значением, содержащимся в специализированном регистре-модификаторе. Для реализации автоматической циклической буферизации каждому регистру-указателю приписывается регистр, хранящий длину циклического буфера. При использовании двух независимых адресных генераторов DAG процессор может генерировать одновременно два адреса для обеспечения одновременной выборки двух операндов. DAG1 может осуществлять адресацию только в памяти данных. DAG2 может осуществлять адресацию и в памяти данных, и в памяти программ. Когда в конфигурационном регистре (MSTAT) установлен соответствующий управляющий бит, адресный генератор DAG1 осуществляет бит-реверсивную адресацию. Бит-реверсивная адресация существенно упрощает реализацию алгоритма БПФ по основанию 2.

Устройство управления последовательностью выполнения команд (секвенсер). Программный секвенсер осуществляет вычисление адреса инструкции, выборка которой должна осуществляться по мере исполнения программы. Ключевым компонентом устройства является регистр команд, который хранит информацию о выполняемой в текущее время команде. Регистр команд образует одноуровневый конвейер в потоке обрабатываемых команд. Команды выбираются и загружаются в регистр команд в одном цикле шины процессора и выполняются в следующем цикле, в то время как осуществляется выборка следующей команды. Чтобы уменьшить количество дополнительных циклов, устройство поддерживает выполнение переходов по условию, вызовов подпрограмм и возвращений к выполнению главной программы за один машинный цикл. Используя внутренний счетчик цикла и аппаратный стек цикла, процессор может выполнять программу цикла с автоматической проверкой условия завершения, без дополнительных затрат процессорного времени на организацию цикла. Таким образом, чтобы организовать цикл, команды явного перехода не требуются.

Необходимость поддержки циклов с автоматической проверкой условий завершения вызвана циклическим характером алгоритмов ЦОС. Функция умножения с накоплением и выборка данных повторяются N раз при каждом вычислении типового алгоритма. В традиционных микропроцессорах организация цикла предполагает наличие в заголовке цикла команд для проверки условия окончания цикла. Архитектура DSP-процессоров обеспечивает аппаратную поддержку программных циклов без

необходимости программной проверки условия продолжения или завершения в теле цикла. Для типичной DSP-архитектуры различие в производительности при аппаратной поддержке цикла с автоматической проверкой условия завершения и при программной проверке условия завершения цикла может превышать 20% времени выполнения цикла.

Устройство также способно с минимальной задержкой реагировать на прерывания, поступающие от контроллера прерываний. Появление прерывания вызывает переход к определенной ячейке памяти, где хранится подпрограмма обработки прерывания. Короткая подпрограмма обработки прерывания может располагаться непосредственно в таблице векторов прерываний, где для этих целей зарезервировано четыре ячейки памяти. Для выполнения более сложной подпрограммы обработки прерывания приходится осуществлять переход с помощью команды JUMP в область памяти программ, где может быть размещена более длинная программа.

Архитектура ядра процессора является одинаковой для всех представителей семейства ADSP-21xx. Напротив, встроенные периферийные устройства имеют различную конфигурацию и разные возможности в зависимости от конкретного исполнения представителя этого семейства. Архитектура процессора ADSP-218x показана на рис. 5.10.

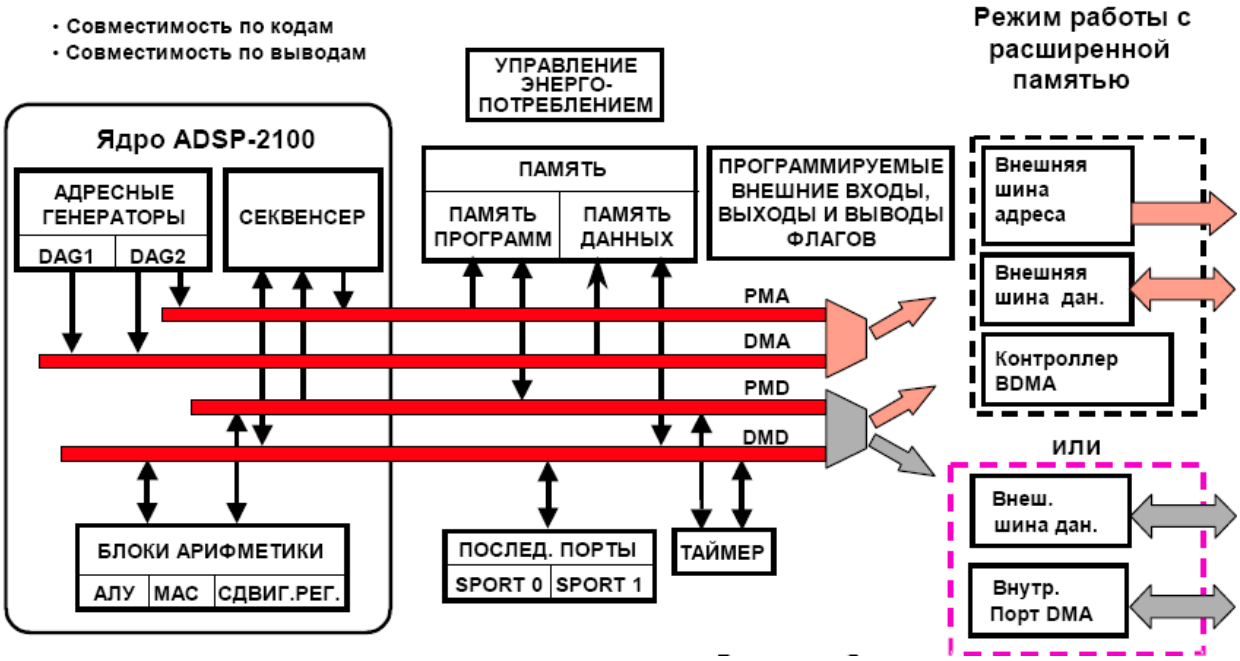


Рис. 5.10. Архитектура процессора ADSP-218x

Встроенная периферия DSP. Семейство 21xx имеет множество различных вариантов процессоров с разным объемом встроенной памяти; в более новом семействе 218x имеются представители, включающие до 48K слов памяти программ и 56K слов памяти данных. Интерфейс внешней памяти поддерживает как быструю, так и медленную память с программируемыми состояниями ожидания.

Процессоры семейства ADSP-218x также поддерживают отдельное адресное пространство портов ввода-вывода. Почти все представители семейства 21xx имеют два последовательных порта с двойной буферизацией (SPORT) для приема и передачи последовательных данных. Каждый SPORT является двунаправленным и имеет свой собственный программируемый генератор битовой и фреймовой синхронизации. Длина слова SPORT может изменяться от 3 до 16 разрядов. Данные могут передаваться с использованием фреймовой синхронизации или без нее. Каждый SPORT способен генерировать прерывания и поддерживает логарифмическое сжатие данных по различным законам.

IDMA-порт процессоров семейства ADSP-218xx поддерживает возможность начальной загрузки процессора от хост-компьютера и возможность доступа со стороны хост-компьютера во внутреннюю память DSP для чтения и записи «на лету», когда DSP занимается выполнением своей программы. Порт IDMA позволяет главному процессору осуществлять доступ ко всей внутренней памяти DSP без использования почтовых регистров. Порт IDMA поддерживает передачу 16- и 24- разрядных слов, при этом передача 24-разрядных слов происходит за два машинных цикла.

Процессоры семейства ADSP-218xx обладают также интерфейсом для взаимодействия с памятью, которая имеет байтовую организацию. Данный интерфейс может использоваться для начальной загрузки процессора и для передачи данных из внутренней памяти и во внутреннюю память «на лету». Максимальный размер адресуемой внешней восьмибитовой памяти составляет 4MB. Данное адресное пространство играет роль загрузочной области (boot memory), характерной для представителей семейства 21xx. Байтовая память имеет организацию 256 страниц по 16К x 8 бит. Передача данных в байтовую память и из нее может осуществляться с упаковкой или распаковкой 24-разрядного, 16-разрядного и 8-разрядного (с выравниванием по младшему или старшему байту) форматов. При доступе к внутренней памяти DSP контроллер DMA занимает один машинный цикл, во время которого ядро не может осуществлять доступ в память.

Процессоры ADSP-21xx имеют специальный режим работы с низкой потребляемой мощностью, который позволяет достигнуть потребления меньше 1 мВт. Переход в данный режим может осуществляться аппаратно или программно. Это особенно важно для устройств, работающих от автономных источников питания. В некоторых режимах работы с низкой потребляемой мощностью отключается внутренний тактовый сигнал, но содержимое памяти и регистров при этом сохраняется.

Циклические буферы. Если мы более внимательно исследуем самую распространенную при цифровой обработке сигналов операцию, то преимущества использования циклических буферов в DSP станут очевидными. Возьмем для примера фильтр с конечной импульсной характеристикой. Во-первых, набор коэффициентов фильтра по своей

природе имеет периодический характер. Во-вторых, при каждом вычислении значения отсчета выходного сигнала фильтр использует новый отсчет входного сигнала и отбрасывает самый старый отсчет. При последовательных вычислениях произведений коэффициентов фильтра на отсчеты сигнала доступ к N коэффициентам фильтра осуществляется последовательно от h(0) до h(N-1). Набор отсчетов входного сигнала циркулирует в памяти следующим образом: новый отсчет входного сигнала сохраняется в памяти вместо старого отсчета всякий раз, когда вычисляется выходное значение фильтра. Для такого циркулирующего буфера может использоваться фиксированная область в ОЗУ. Самое раннее значение в памяти заменяется новым после каждого вычисления операции свертки. При этом информация об N последних отсчетах сохраняется в ОЗУ.

В виде циклического буфера в ОЗУ DSP-процессора может быть реализована задержка, если новые значения записываются в память на место старых. Для упрощения адресации памяти старые значения считываются из памяти, начиная со значения, расположенного сразу после того, которое было только что записано. Например, в фильтре с четырьмя коэффициентами новый отсчет h(4) записывается в ячейку памяти с адресом 0. Далее чтение данных осуществляется из ячеек с адресами 1,2,3 и 0 в указанном порядке. Этот способ применяется при любом числе звеньев фильтра. При такой адресации ячеек памяти генератор адреса должен выдавать лишь последовательные значения адресов, вне зависимости от того, какая операция с памятью – чтение или запись – осуществляется в настоящий момент. Буфер такого типа называется циклическим, потому что когда при записи достигается последняя ячейка, указатель памяти устанавливается на начало буфера.

Выборка коэффициентов из памяти программ осуществляется одновременно с выборкой данных из памяти данных. При рассмотренной схеме адресации самые старые отсчеты извлекаются из памяти первыми. Поэтому последний из коэффициентов должен выбираться из памяти первым. Коэффициенты могут заноситься в памяти в обратном порядке: h(N-1) – в первую ячейку, а h(0) – в последнюю, и генератор адреса в этом случае должен генерировать последовательно возрастающие адреса. И наоборот, коэффициенты могут быть записаны в памяти в нормальном порядке, но доступ к ним при этом должен осуществляться, начиная с конца буфера, а генератор адреса должен генерировать последовательно убывающие адреса.

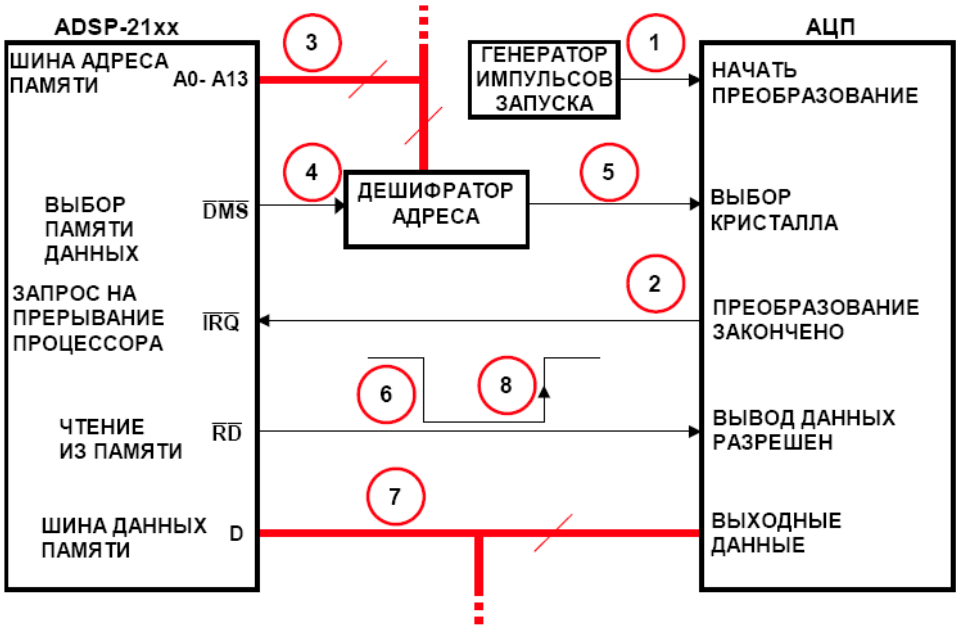
Описанные выше механизмы позволяют реализовать задержку, требуемую при реализации фильтра, без каких-либо дополнительных затрат процессорного времени. Использование циклических буферов является специфическим для цифровой обработки сигналов и для достижения максимальной эффективности циклические буферы должны поддерживаться аппаратно. Аппаратная реализация циклических буферов позволяет установить параметры буфера (такие как адрес начала буфера, длина и т.д.) в

программе вне тела цикла, непосредственно вычисляющего алгоритм. Это позволяет избежать включения дополнительных команд в тело цикла. Отсутствие аппаратной реализации циклических буферов может существенным образом ухудшить возможности DSP-процессора по реализации алгоритмов цифровой обработки сигналов.

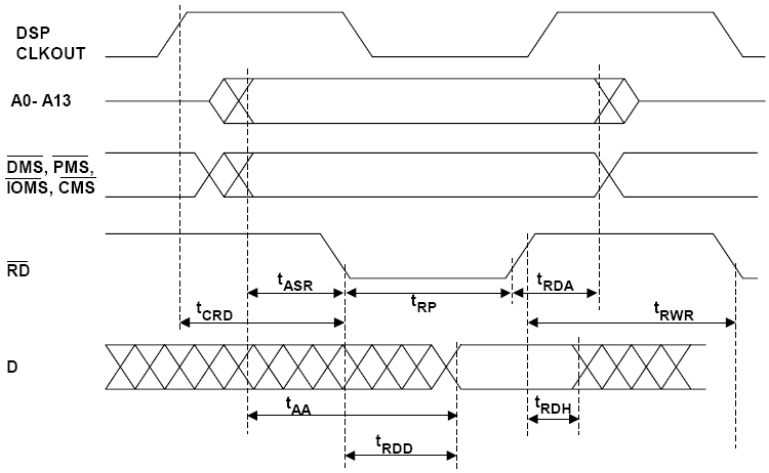
Организация интерфейса с DSP. В связи с быстрым развитием технологии смешанной аналогово-цифровой обработки сигналов устройства на базе DSP с высокой степенью интеграции, появляющиеся на рынке в настоящее время (например ADSP-21ESP202), имеют помимо DSP-ядра интегрированные АЦП/ЦАП, что снимает проблему организации интерфейса между отдельными компонентами.

Большинство DSP не имеют встроенных АЦП и ЦАП. Современные АЦП и ЦАП оснащаются стандартными цифровыми интерфейсами, специально предназначенными для связи с DSP, и тем самым минимизируют или устраняют необходимость внешней поддержки интерфейса или применения интерфейсной логики. Высокопроизводительные сигма-дельта-АЦП и ЦАП в настоящее время выпускаются в одном корпусе (такое комбинированные решение называется КОДЕК или КОдер/ДЕКодер), например, AD73311 и AD73322. Данные устройства также разработаны с учетом минимальных требований к интерфейсной логике при работе с наиболее распространенными DSP-процессорами.

Возможна организация сопряжения как по параллельному, так и по последовательному каналам. На рис. 5.11 приведена схема подключения (а) аналого-цифрового преобразователя AD7854/AD7854L к сигнальному процессору ADSP-2189M и временная диаграмма (б) обмена данными между ними по параллельному интерфейсу. Микросхемы AD7854/AD7854L – это 12-разрядные АЦП, работающие с частотой отсчетов 100 или 200 кГц, которые имеют параллельный интерфейс.



а) структурная схема подключения



б) циклограмма обмена данными

Рис. 5.11. Подключение АЦП к ADSP-21xx через параллельный интерфейс

Основные требования при подключении АЦП через параллельный интерфейс.

- ✓ шина данных периферийного устройства должна поддерживать высокоимпедансное z-состояние;
- ✓ время декодирования адреса и время включения периферийного устройства не должно превышать время t_{ASR} установки процессором адреса и сигнала выбора памяти (0,325 нс – для процессора ADSP-2189M);
- ✓ для того, чтобы осуществить доступ без режима ожидания, время от спадающего (переднего) фронта сигнала чтения RD до момента достоверного установления данных не должно превышать t_{RDD} (1,65 нс – для процессора ADSP-2189M при работе на частоте 75 МГц), иначе необходимо программно обеспечить режим ожидания или снизить частоту работы процессора;
- ✓ на выходе АЦП должны поддерживаться достоверные данные в течение времени t_{RDH} после восходящего (заднего) фронта сигнала чтения RD (время t_{RDH} равно нулю для процессора ADSP-2189M);
- ✓ периферийное устройство должно работать при как можно меньшей длительности строба t_{RP} (3,65 нс – для процессора ADSP-2189M при работе на частоте 75 МГц), иначе необходимо программно обеспечить режим ожидания или снизить частоту работы процессора.

Процессор ADSP-2189M способен эффективно взаимодействовать с медленными периферийными устройствами при помощи имеющихся средств программирования длительности состояния ожидания. Имеется три специальных регистра для управления процессом ожидания: для памяти начальной загрузки, для памяти программ и для памяти данных и пространства ввода-вывода. Программист может задать от 0 до 15 тактов ожидания для каждого параллельного интерфейса памяти. Каждый такт

ожидания увеличивает время доступа к внешней памяти на величину, равную по длительности одному такту генератора тактовых импульсов процессора (13,3 нс для процессора ADSP-2189M, работающего на тактовой частоте 75 МГц). В рассматриваемом примере сигналы адрес памяти данных, DMS и RD удерживаются неизменными в течение дополнительного времени, определяемого продолжительностью тактов ожидания.

DSP-процессоры, имеющие последовательные порты (например, семейство ADSP-21xx), позволяют организовать простой интерфейс с такими периферийными устройствами, как АЦП и ЦАП. Наличие последовательного порта устраняет необходимость использования больших параллельных шин для подключения АЦП и ЦАП к DSP-процессорам.

Вопросы для повторения

15. В чем особенность архитектуру микроконтроллера по сравнению с классической МС?
16. В чем отличие специализируемого сигнального процессора и контроллера?
17. Какие функционально-необходимые элементы содержит микроконтроллер?
18. Какие периферийные модули ввода-вывода используются в распределенной микроконтроллерной системе управления?
19. Назовите способы контроля реального времени на таймере.
20. Что такое режим захвата и режим сравнения таймера?
21. Как организовать ШИМ на базе таймера?
22. Какое назначение таймера Watch Dog, его принцип функционирования?
23. Какие производители сигнальных процессоров наиболее известны на рынке?
24. Назовите особенности базовой архитектуры ПЦОС.
25. Какие функции выполняет устройство MAC сигнального процессора?
26. Какие внутренние шины использует ядро сигнального процессора ADSP-21xx?
27. Укажите особенности встроенной периферии DSP/
28. Особенности реализации циклических буферов в сигнальных процессорах.
29. Каким образом могут быть подключены устройства АЦП/ЦАП к ПЦОС?

Тема 6. Внешние устройства ввода-вывода

Цели и задачи изучения темы

Основной целью изучения данной темы является знакомство с видами, устройством и принципом действия устройств ввода-вывода аналоговых, дискретных и цифровых сигналов, протоколами и сетевыми средствами передачи данных.

Задачи:

1. Рассмотреть устройство и принцип действия ЦАП и АЦП, схемы ввода и вывода аналоговых сигналов.
2. Ознакомится с особенностями построения портов дискретных сигналов одностороннего и двухстороннего действия
3. Изучить основные унифицированные последовательные интерфейсы и сетевые средства передачи данных.

6.1. Ввод-вывод сигналов

Входные и выходные сигналы МС и, в частности, микроконтроллеров можно разделить на аналоговые и дискретные. Внешние аналоговые сигналы могут быть введены в МК по средствам аналого-цифрового преобразователя (АЦП). Обратное преобразование, позволяющее сформировать на выходе МК аналоговый сигнал, называется цифро-аналоговое преобразование (ЦАП).

Микроконтроллер как правило имеет некоторое количество внешних линий ввода-вывода, подключенных к внешним выводам микросхемы и называемых внешними портами. Одиночные (одноразрядные, состоящие из одной линии) порты ввода-вывода объединяются в группы, обычно, по 4, 8 или 16 линий, которые называются параллельными портами [2]. Каждый разряд параллельного порта выступает в качестве отдельного дискретного сигнала. Через порты МК взаимодействует с различными внешними устройствами – считывает значения входных дискретных сигналов и устанавливает значения выходных дискретных сигналов.

Порт ввода и порт вывода могут иметь как разные адреса доступа, так и один адрес, тогда говорят о двунаправленном порте ввода-вывода. Во время цикла чтения МП двунаправленный порт переходит в режим «ввода», а во время цикла записи – в режим «вывода».

6.1.1. Вывод аналоговых сигналов

Вывод аналогового сигнала осуществляется с помощью цифро-аналогового преобразователя, который предназначен для генерации выходного напряжения (или тока) с уровнем, соответствующим заданному цифровому коду. На практике ЦАП применяется для управления различными исполнительными устройствами и системами, например, источниками питания с управляемым напряжением, различными индикаторами. С

помощью ЦАП можно восстанавливать оцифрованные аудио и видео сигналы, синтезировать аналоговые сигналы различной формы.

Существуют различные подходы при построении ЦАП, основанные как на классическом многоразрядном преобразовании, так и на различных методах модуляции. Для построения многоразрядных ЦАП могут использоваться различные принципы. Наиболее часто используется принцип формирования токов, пропорциональных весовым коэффициентам разрядов двоичного кода с последующим их суммированием в разрядах кода, содержащих логическую единицу [6]. Формирование указанных токов обычно производится с помощью ***резисторной матрицы R-2R***. Схема такого формирователя токов приведена на рис. 6.1 [2].

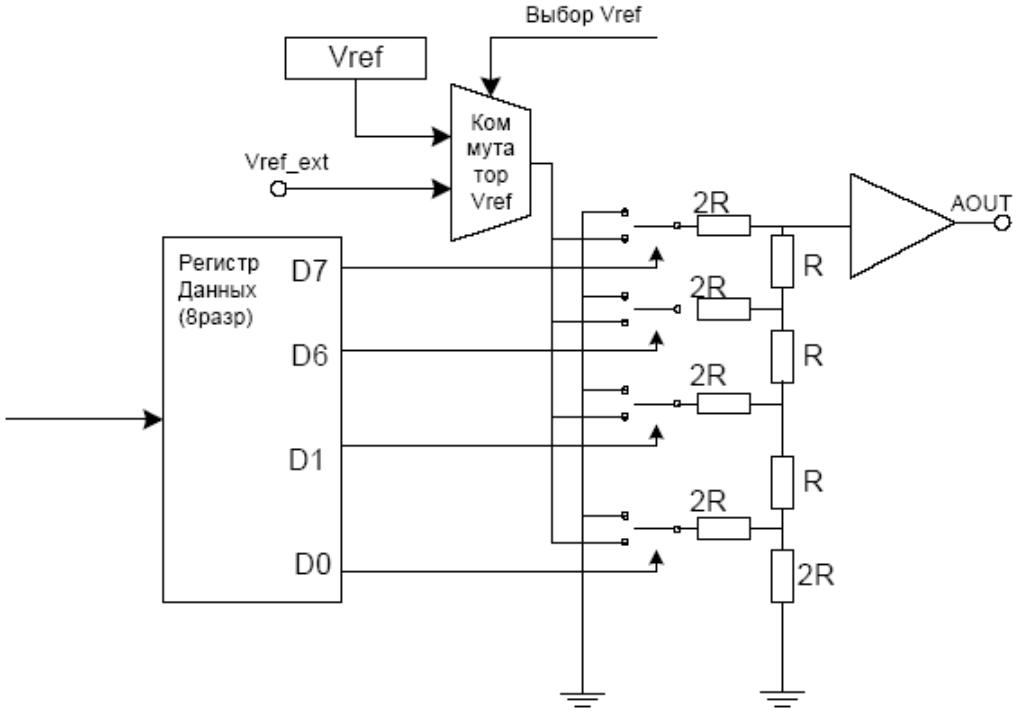


Рис. 6.1. Цифро-аналоговое преобразование с помощью матрицы R-2R

Регистр данных определяет разрядность ЦАП, в него записывается цифровой код выходного сигнала. Матрица R-2R – самый распространенный метод цифро-аналогового преобразования. Матрица работает по принципу деления входного напряжения на входах. Матрица имеет число входов по числу разрядов регистра данных. На каждый вход через ключ может быть подано опорное напряжение Vref или 0 В. Ключи управляются разрядами регистра данных: «1» – на матрицу подается Vref, «0» – подается 0 В. Коммутатор опорного напряжения Vref позволяет выбрать внешний или встроенный источник опорного напряжения. В ИМС используют полевые транзисторы в роли ключей, на затвор которых подается необходимая информация, позволяющая направлять необходимый ток на «землю» или на ОУ. Выходное напряжение, снимаемое с выхода ОУ, будет пропорционально току, поступающему на его вход.

Одним из наиболее простых и распространенных способов преобразования выходной информации ПЛК в аналоговый вид является ЦАП, основанный на широтно-импульсной модуляции. Преобразование цифрового кода в последовательность импульсов осуществляется чаще всего на основе таймера, работающего в режиме «COMPARE INTERRUPT» (см. рис. 5.6), хотя возможны и другие схемы преобразования цифрового кода во временной интервал и генерации широтно-модулированной импульсной последовательности.

Преобразование выходной последовательности ШИМ в непрерывный аналоговый сигнал производиться с помощью интегрирующей цепи или фильтра низких частот (ФНЧ), имеющего постоянную времени много большую, чем период ШИМ. Восстановление аналогового сигнала на выходе ФНЧ иллюстрируется на рис. 6.2, где *x* – выходной непрерывный сигнал, *u* – последовательность импульсов на выходе ШИМ.

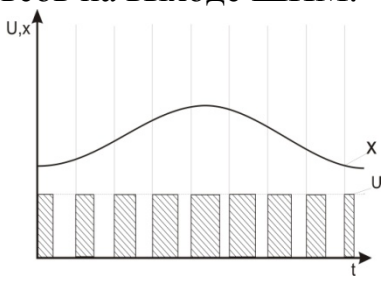


Рис. 6.2. Восстановление непрерывного аналогового сигнала (x) из последовательности импульсов (u) широтно-импульсного модулятора

Описанное устройство, осуществляющее цифро-аналоговое преобразование, основанное на ШИМ, часто называют ***однобитным*** ЦАП (рис. 6.3).

Рис. 6.3. Структурная схема однобитного ЦАП

При этом уровень выходного напряжения однобитного ЦАП прямо пропорционален скважности импульсов ШИМ (γ) :

$$U = U_{имп} \cdot \gamma ,$$

где $U_{имп}$ – амплитуда импульсов, В. Разрешающая способность однобитного ЦАП определяется дискретностью изменения скважности импульсов, т.е. разрядностью регистров таймера в режиме ШИМ.

В однобитном ЦАП используется один дискретный выход – один бит информации. Этот подход технически достаточно просто реализовать, так как требуется всего лишь один вентиль или ключ, формирующий последовательность импульсов на выходе ПЛК (Ф), и фильтр низких частот. Ключ может находиться только в двух состояниях: выключено – ключ

закрыт и включено – ключ открыт. Благодаря импульсному характеру переключения режимов ключа потеря энергии в момент самого переключения минимальна, поэтому мы имеем более высокий КПД по сравнению с аналоговыми усилителями, и, кроме того, техническое решение с одним вентилем имеет более низкую стоимость в сравнении с использованием многобитных систем цифрового вывода.

В качестве примера приведем задачу вывода звукового (речевого) сигнала с помощью однобитного ЦАП MASH. Пусть сигнал был представлен в PCM-формате (импульсно-кодовая модуляция – ИКМ) с 2^{16} квантами сигнала и периодом квантования 44,1 КГц. Однобитный ЦАП MASH по средствам одного единственного ключа преобразует сигнал ИКМ в ШИМ сигнал с частотой модуляции свыше 2 МГц, что более, чем в 16 раз превышает частоту исходного квантования ИКМ – 44,1 КГц. Так как информация идет не по 16 линиям, а только по одной, но плотнее более, чем в 16 раз, то потери информации не происходит. Далее последовательность импульсов поступает на ФНЧ, где отфильтровывается частота модуляции и выделяется аналоговый сигнал.

6.1.2. Ввод аналоговых сигналов

Рассмотрим принципы построения аналого-цифровых преобразователей (АЦП).

АЦП последовательных поразрядных приближений. На рис. 6.4 приведена функциональная схема такого АЦП [6].

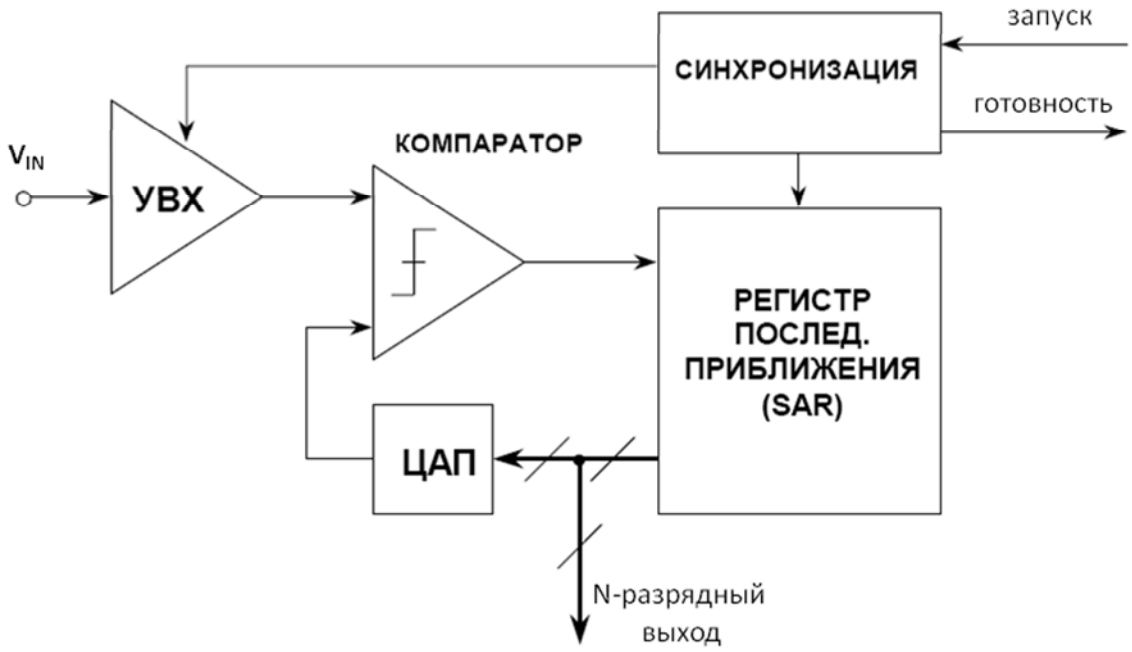


Рис. 6.4. Функциональная схема АЦП последовательных поразрядных приближений.

В преобразовании предусматривается N-разрядный регистр, в котором последовательно, разряд за разрядом, начиная со старшего разряда, формируется двоичное число, соответствующее цифровой форме представления поданного на вход микросхемы аналогового напряжения V_{IN}. Процесс формирования этого числа состоит в следующем.

В начале преобразования регистр сдвига SAR сбрасывается в нуль. Затем запускается цикл аналого-цифрового преобразования с записи единицы в выходной триггер старшего разряда регистра. Получающееся в регистре число с помощью ЦАП преобразуется в аналоговое напряжение, которое на аналоговом компараторе сравнивается с входным напряжением V_{IN}, мгновенное значение которого запоминается на устройстве выборки-хранения УВХ на всем протяжении цикла преобразования АЦП. При выполнении неравенства $V_{IN} \geq V_{ЦАП}$ в триггере старшего разряда регистра сохраняется «1». При обратном неравенстве триггер старшего разряда сбрасывается в «0». Далее производится запись «1» в выходной триггер (N–1)-го разряда регистра, и вновь сравнивается V_{IN} и $V_{ЦАП}$. Так производится опробование во всех N разрядах регистра. По окончанию цикла преобразования, т.е. прохождения всех N-разрядов, на выходе регистра формируется двоичный код, соответствующий входному напряжению, и формируется импульс готовности, по которому код записывается в буферный регистр шины данных.

Метод последовательного приближения требует для своей реализации определенного интервала времени, зависящего от разрядности АЦП и частоты генератора тактовых импульсов. Поэтому, обмен данными между МП и АЦП должен выполняться с обязательным подтверждением готовности УВВ либо в режиме ожидания, либо в режиме прерывания.

Более высоким быстродействием (меньшим временем преобразования) обладают ***АЦП параллельного действия***, так называемые Flash-ADC. Такие АЦП включают источник опорного напряжения, резистивный делитель, компараторы напряжения, преобразователь кодов, выходной буферный регистр и генератор тактовых импульсов [6]. Для преобразования аналогового сигнала в N-разрядный код требуется 2^N–1 компараторов и 2^N резисторов, размещенных, как это показано на рис. 6.5.

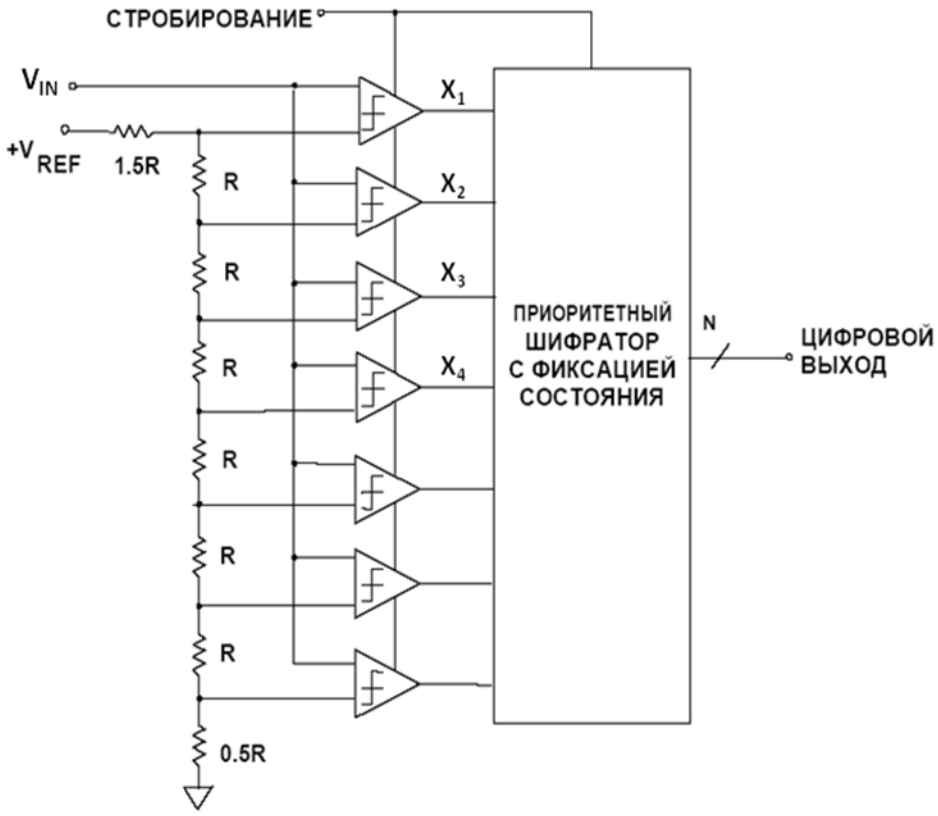


Рис. 6.5. Структурная схема АЦП параллельного действия

На каждый компаратор подается опорное напряжение, значение которого для соседних точек отличается на величину, соответствующую одному младшему значащему разряду (LSB) (более старшие разряды — в верхних по схеме элементах). На один вход каждого компаратора подается напряжение с соответствующей точки делителя, а на второй вход всех компараторов одновременно поступает аналоговый сигнал V_{IN}. Таким образом, на входе каждого компаратора постоянно присутствует заданный уровень напряжения, с которым осуществляется сравнение аналогового сигнала.

Если величина аналогового сигнала V_{IN} превышает уровень напряжения делителя, то на его выходе вырабатывается логическая «1». Если V_{IN} меньше – то на выходе будет уровень «0», т. е. результат получается в унитарном коде ($x_1x_2x_3\ldots$). Таким образом все компараторы, размещенные на схеме ниже некоторой точки, имеют входное напряжение выше опорного напряжения. На их логическом выходе присутствует «1». У всех же компараторов выше этой точки опорное напряжение больше входного, и их логический выход установлен в «0». Поэтому $2^N–1$ выходов компаратора ведут себя аналогично ртутному термометру, и выходной код такого АЦП иногда называют «кодом термометра».

Максимальное значение величины аналогового сигнала V_{IN} определяется соотношением: $(V_{IN})_{max}=(1–½^N)\cdot V_{REF}$, где V_{REF} – опорное напряжение. Эти сигналы поступают на вход преобразователя кодов – шифратора, который преобразует унитарный код в двоичный *n*-разрядный код. Затем этот код записывается в буферном регистре.

Управление работой АЦП параллельного действия осуществляется тактовыми импульсами стробирования. Выборка аналогового сигнала производится после появления положительного фронта импульса на управляющем входе компаратора. Преобразование унитарного кода, поступающего со всех компараторов, осуществляется с момента поступления на входы управления логических схем преобразователя кодов отрицательного фронта импульса. Результат преобразования, т.е. N-разрядный код, записывается в выходной буферный регистр во время появления положительного фронта следующего импульса. Таким образом, полный цикл преобразования занимает один период тактовых импульсов.

АЦП параллельного действия обладают самым высоким быстродействием (до 1…10 нс), что позволяет вести обработку сигналов в реальном масштабе времени. Их максимальная частота дискретизации может достигать 1 ГГц. Однако, из-за большого количества аналоговых компараторов реализовать многоразрядное АЦП весьма проблематично из-за соображений цены и энергопотребления. Обычное количество разрядов серийно выпускаемых АЦП параллельного действия редко превышает 8, при этом требуемое число компараторов $N = 2^8 - 1 = 255$.

Более широкие возможности для работы с аналоговыми сигналами дает ***многоканальный модуль АЦП*** (рис. 6.6) [2, 4].

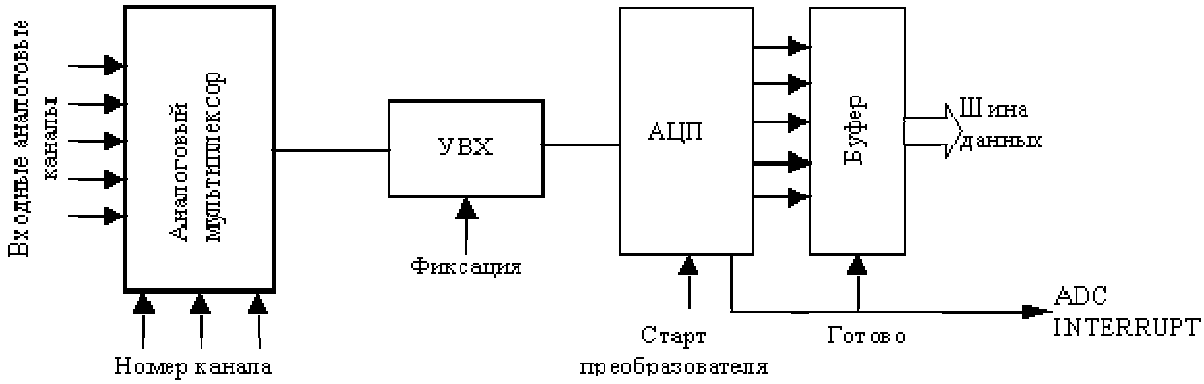


Рис. 6.6. Структурная схема многоканального модуля АЦП

Многоканальный аналоговый мультиплексор (коммутатор) служит для подключения одного из источников аналоговых сигналов ко входу АЦП. Преобразования начинаются выбором требуемого канала, который осуществляется записью номера канала в специальный служебный регистр. При этом сигнал выбранного канала пропускается на выход коммутатора. После этого подается сигнал фиксации на устройство выборки и хранения (УВХ). Оно фиксирует уровень выбранного аналогового сигнала на весь период преобразования. Затем подается сигнал начала преобразования. АЦП обычно строятся по схеме последовательного приближения.

Время преобразования обычно составляет несколько десятков микросекунд, в зависимости от частоты тактирования АЦП. После окончания преобразования АЦП выдает сигнал готовности, по которому его выходная

информация записывается в буферный регистр. Если прерывания от модуля АЦП разрешены, то генерируется запрос на прерывания «ADC INTERRUPT».

Полученный цифровой код может быть считан процессором по соответствующей команде. Некоторые микроконтроллеры имеют блоки АЦП, способные работать в режиме автоматического сканирования, опрашивая несколько каналов последовательно без участи процессора (Intel 8051GB, Intel 80C166, Fujitsu MB90). Программисту будет достаточно считывать данные из регистра, соответствующего требуемому каналу. При этом код выбора канала параллельно подается на адресные входы блока регистров данных

Входы АЦП рекомендуется развязывать от выходных цепей предыдущих каскадов (если они имеют высокое выходное сопротивление) во избежание влияния входного тока АЦП на точность преобразования. Это можно осуществить с помощью повторителя (рис. 6.7) [4].

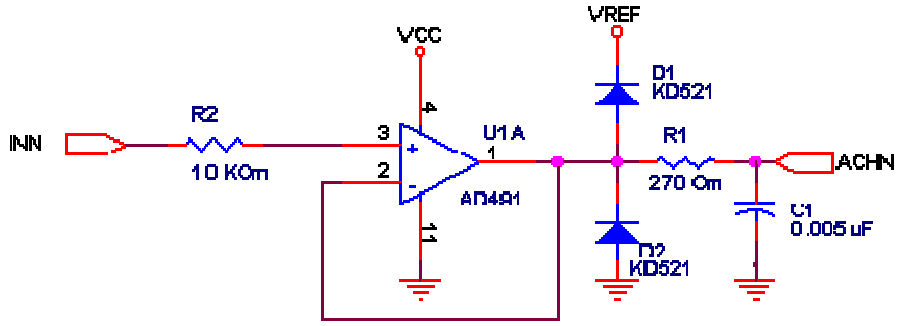


Рис. 6.7. Цепи развязки аналоговых сигналов

Диоды D1 и D2 осуществляют защиту входа АЦП от превышения допустимого диапазона напряжений. Цепь R1, C1 является фильтром низких частот и гасит паразитные высокочастотные колебания на входе АЦП. Вывод земли аналоговой части необходимо подсоединить к общему выводу микросхемы ПЛК в непосредственной близости от нее во избежание паразитных наводок цифровой части схемы на аналоговую.

Ввод аналоговых сигналов может производиться и без применения многоразрядных АЦП. Если преобразование основано на импульсной модуляции аналогового сигнала, то такое устройство называют ***однобитным АЦП.***

Рассмотрим два примера АЦП, работа которых основана на ШИМ и ЧИМ видах модуляции. В первом случае входной аналоговый сигнал пропускается через ШИМ-модулятор. Выход ШИМ подключается к одному из дискретных портов ввода микроконтроллера, способных генерировать сигнал запроса прерывания (схема захвата). С помощью таймера в режиме «CAPTURE INTERRUPT» для ввода аналогового сигнала достаточно

определить длительность импульса, если период следования импульсов постоянен.

Второй пример АЦП используется для ввода аналоговых медленно меняющихся сигналов. Аналоговый сигнал преобразуется в частотно-модулированную последовательность импульсов (ЧИМ) с помощью устройства ГУН (генератор, управляемый напряжением) и формирователя прямоугольных импульсов Ф (рис. 6.8).

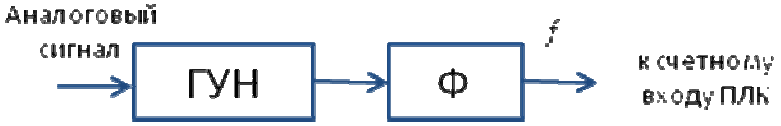


Рис. 6.8. Структурная схема однобитного АЦП

При этом задействован «***счетный вход***» микроконтроллера и таймер, работающий в режиме «CAPTURE INTERRUPT» (см. рис. 5.5). Так как частота следования импульсов f пропорциональна напряжению входного аналогового сигнала, то достаточно программно подсчитать количество импульсов за единицу времени.

Особенность описанного устройства состоит в том, что чем меньше период времени, в течение которого ведется подсчет количества импульсов, тем ниже разрешающая способность АЦП. Как правило, максимальная частота импульсного сигнала, которую может обработать таймер, не превышает сотни кГц. Например, если максимальная частота равна 65 536 Гц, то время измерения, для которого однобитный АЦП соответствует 16-ти разрядному обыкновенному АЦП, должно равняться 1 сек (так как $2^{16} = 65\,536$ импульсов в секунду). С другой стороны, большое время измерения – сглаживает зашумленный аналоговый сигнал, что положительно сказывается на погрешности измерения. Все это обусловило область применения однобитных АЦП – ввод очень медленно меняющихся аналоговых сигналов.

В отличие от рассмотренных выше примеров АЦП, работа которых основана на ШИМ и ЧИМ, в последнее время широкое распространение получили ***сигма-дельта*** АЦП, работа которого основана на смешанном виде модуляции – ШИМ и ЧИМ. Сигма-дельта (ΣΔ – Sigma-delta) АЦП известны почти тридцать лет, но только недавно появилась технология (цифровые микросхемы с очень высокой степенью интеграции, VLSI) для их производства в виде недорогих монолитных интегральных схем. В настоящее время они используются во многих приложениях, где требуется недорогой, узкополосный, экономичный АЦП с высоким разрешением.

ΣΔ АЦП содержит очень простую аналоговую электронику (компаратор, источник опорного напряжения, коммутатор и один или большее количество интеграторов и аналоговых сумматоров) и весьма сложную цифровую вычислительную схему. Эта схема состоит из дискретного фильтра низких частот (или низкочастотного полосового

фильтра), который реализован либо с помощью аппаратной схемы, либо программно с использованием цифрового сигнального процессора (DSP).

Для понимания того, как работает ΣΔ АЦП, важно познакомиться с концепциями избыточной дискретизации, формирования формы кривой распределения шума квантования, цифровой фильтрации и децимации.

Рассмотрим методику избыточной дискретизации с анализом в частотной области. Там, где преобразование постоянного напряжения имеет ошибку квантования до ½ младшего разряда (LSB), дискретная система, работающая с переменным напряжением или током, обладает шумом квантования. Идеальный классический N-разрядный АЦП имеет среднеквадратичное значение шума квантования, равное $q/\sqrt{12}$. Шум квантования равномерно распределен в пределах полосы Найквиста от 0 до $f_S/2$ (где q – значение младшего значащего бита и f_S – частота дискретизации), как показано на рис. 6.9-а.

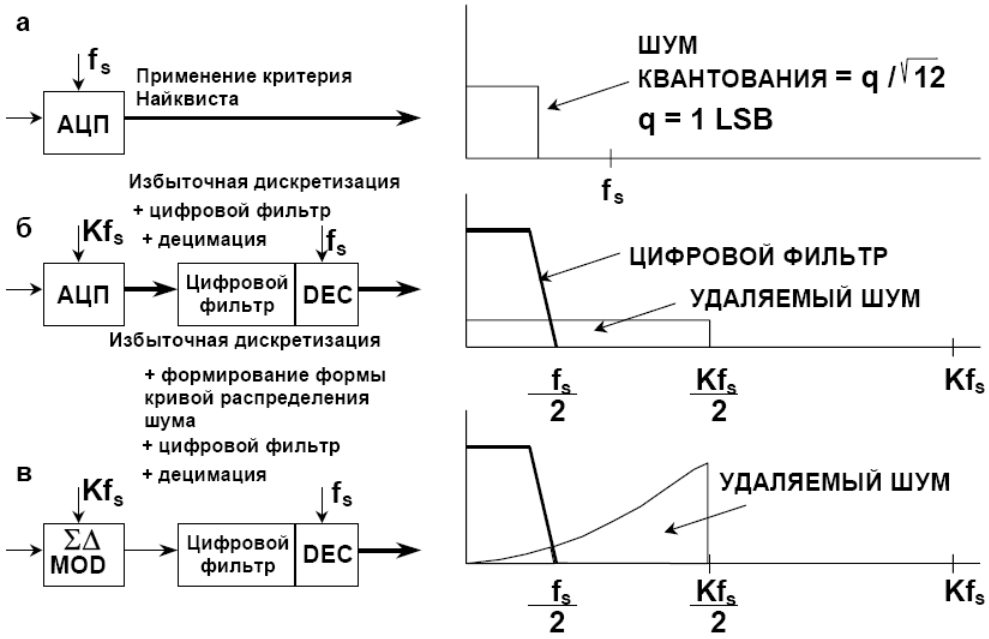


Рис. 6.9. Иллюстрация метода уменьшения шума квантования

Если выбрать более высокую частоту дискретизации $K \cdot f_S$ (рис. 6.9-б), то среднеквадратичное значение шума квантования остается $q/\sqrt{12}$, но шум теперь распределен по более широкой полосе от 0 до $K \cdot f_S/2$. Если затем использовать на выходе цифровой низкочастотный фильтр, то значительно уменьшится шум квантования, но сохранится полезный сигнал, улучшая таким способом эффективное число разрядов. Таким образом, реализуется аналого-цифровое преобразование с высоким разрешением, используя аналого-цифровой преобразователь с низкой разрешающей способностью. Коэффициент K здесь упоминается, как коэффициент избыточной дискретизации. При этом необходимо отметить, что избыточная дискретизация дополнительно выгодна еще и тем, что она понижает требования к аналоговому ФНЧ.

Так как ширина полосы пропускания уменьшена выходным цифровым фильтром, скорость выдачи выходных данных может быть ниже, чем первоначальная частота дискретизации ($K \cdot f_S$), и при этом все же удовлетворять критерию Найквиста. Это достигается посредством передачи на выход каждого M-го результата и отбрасывания остальных результатов. Такой процесс называют прореживанием или децимацией с коэффициентом M. Несмотря на происхождение термина (decem по-латыни – десять), M может принимать любое целое значение, при условии, что частота выходных данных больше, чем удвоенная ширина полосы сигнала. Прореживание не вызывает никакой потери информации.

ΣΔ-преобразователь не только ограничивает полосу пропускания сигнала, но также задает форму кривой распределения шума квантования таким образом, что большая ее часть выходит за пределы этой полосы пропускания, как это показано на рис. 6.9-в.

Если взять одноразрядный АЦП (известный как компаратор), подать на его вход сигнал от интегратора, а на интегратор – входной сигнал, суммированный с выходом одноразрядного ЦАП, на вход которого сигнал поступает с выхода АЦП, получится ΣΔ-модулятор первого порядка, показанный на рис. 6.10.

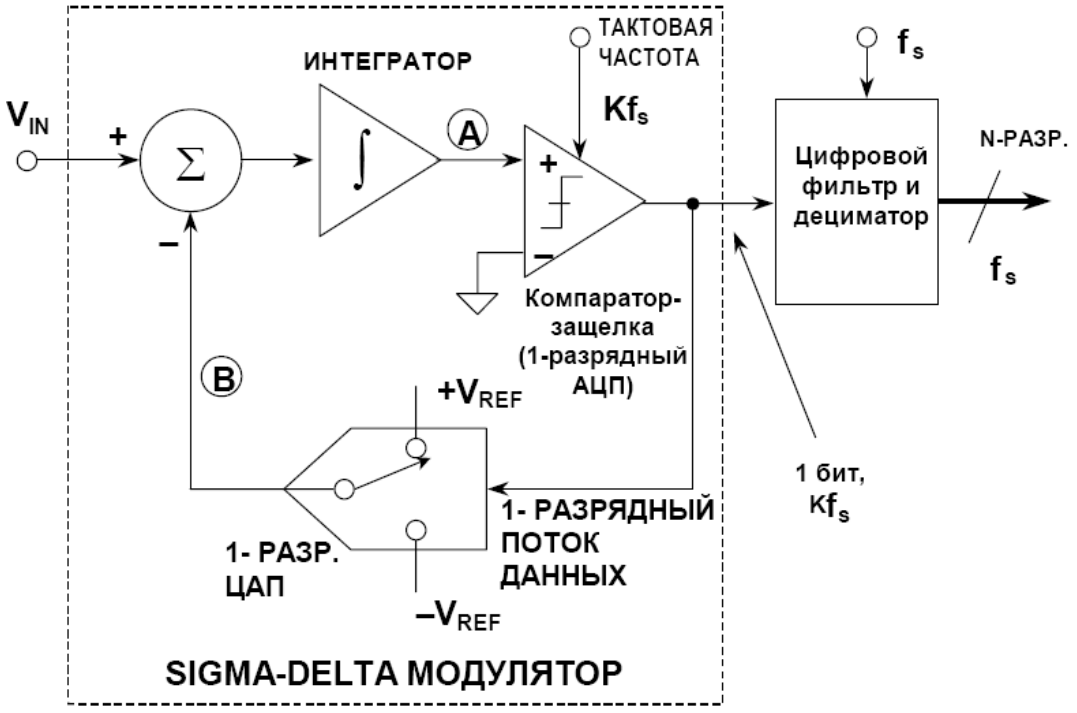


Рис. 6.10. Сигма-дельта АЦП первого порядка

Добавив цифровой низкочастотный фильтр и дециматор на цифровой выход, получим ΣΔ АЦП: ΣΔ-модулятор формирует такую кривую распределения шума квантования, при которой большая часть шума располагается выше полосы пропускания цифрового выходного фильтра и, следовательно, эффективное число разрядов намного больше, чем ожидается только от коэффициента избыточной дискретизации (рис. 6.9-в).

Не вдаваясь в детали, работу ΣΔ АЦП можно описать следующим образом. Представим, что постоянное напряжение подается на вход V_{IN}. Сигнал на выходе интегратора в точке А при этом постоянно нарастает или убывает. С выхода компаратора сигнал подается обратно через одноразрядный ЦАП на суммирующий вход в точке В. Благодаря отрицательной обратной связи, соединяющей выход компаратора через одноразрядный ЦАП с точкой суммирования, среднее значение постоянного напряжения в точке В стабилизируется на уровне V_{IN}. Вследствие этого, среднее выходное напряжение ЦАП равняется входному напряжению V_{IN}. В свою очередь, среднее выходное напряжение ЦАП определяется плотностью потока единиц в одноразрядном потоке данных, следующего с выхода компаратора. Когда значение входного сигнала увеличивается в направлении к $+V_{REF}$, число единиц в последовательном потоке данных увеличивается, а число нулей уменьшается. Точно так же, когда значение сигнала приближается к отрицательному значению $– V_{REF}$, число единиц в последовательном потоке данных уменьшается, а число нулей увеличивается. Попросту говоря, в последовательном потоке разрядов на выходе компаратора содержится среднее значение входного напряжения. Цифровой фильтр и дециматор обрабатывают последовательный поток битов и выдают окончательные выходные данные.

Принцип формирования кривой распределения шума квантования в частотной области объясняется на простой модели ΣΔ-модулятора, показанной на рис. 6.11

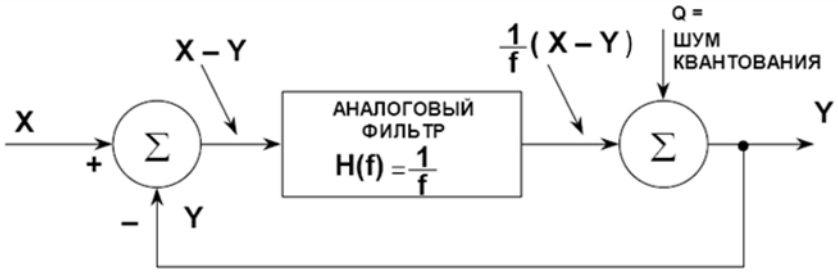


Рис. 6.11. Упрощенная линеаризованная модель сигма-дельта модулятора в частотной области

Интегратор в ΣΔ-модуляторе представлен в виде аналогового ФНЧ с передаточной функцией $H(f) = 1/f$. Эта передаточная функция имеет затухающую амплитудную характеристику со скоростью 20 дБ на каждое увеличение частоты в 10 раз. Одноразрядный источник импульсов генерирует шум квантования Q, который добавляется к выходному сигналу суммирующего блока. Если считать входной сигнал равным X, а выходной – равным Y, то сигнал на выходе входного сумматора должен быть $X – Y$. Эта величина умножается на передаточную функцию фильтра $1/f$, и результат подается на один из входов выходного сумматора. В итоге получается выражение для выходного напряжения Y в виде:

$$Y = \frac{1}{f}(X - Y) + Q$$

Это выражение может быть легко решено относительно Y с аргументами X, f и Q:

$$Y = \frac{X}{f+1} + \frac{Q \cdot f}{f+1}$$

Обратите внимание, что, когда частота f приближается к нулю, значение выходного напряжения Y стремится к X, а шумовая составляющая устремляется к нулю. На более высоких частотах амплитуда сигнальной составляющей стремится к нулю, а шумовая составляющая приближается к Q. В сущности, аналоговый фильтр представляет собой ФНЧ для сигнала и ФВЧ для шума квантования. Иными словами, аналоговый фильтр выполняет функцию формирования кривой распределения шума квантования в модели ΣΔ-модулятора.

В задачах управления на базе ПЛК часто требуется контролировать аналоговые сигналы на предмет выхода их за некоторые предельно допустимые рамки. В этом случае не обязательно вводить эти аналоговые сигналы, а достаточно применить устройство, называемое ***аналоговым компаратором.*** Оно используется для сравнения напряжения двух внешних аналоговых сигналов или для сравнения напряжения внешнего аналогового сигнала с образцовым напряжением, вырабатываемым ЦАП микроконтроллера (рис. 6.12) [2].

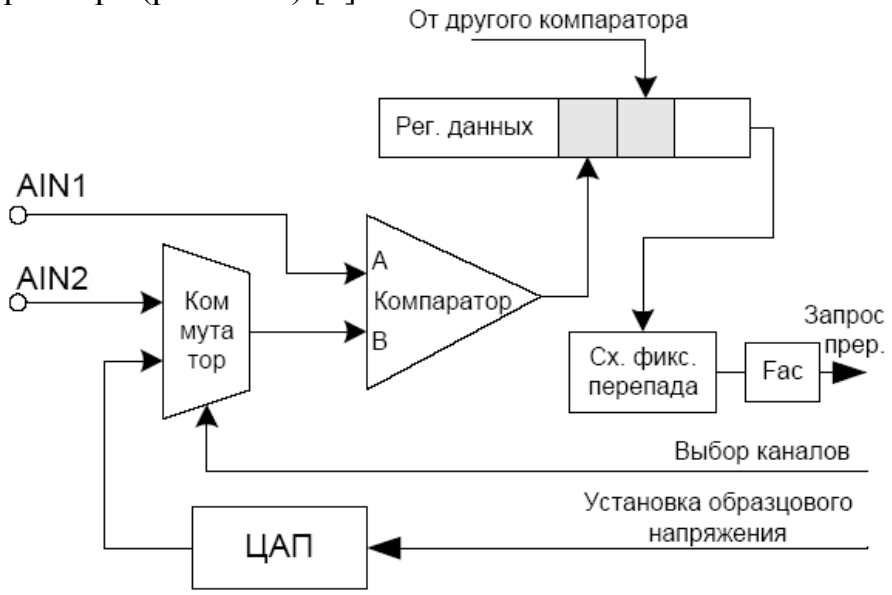


Рис. 6.12. Аналоговый компаратор

Различные уровни образцового напряжения могут быть запрограммированы путем выдачи соответствующего кода на ЦАП. Результат сравнения кодируется битом в регистре специального назначения, например, «1» – вход А больше или равно чем В, «0» – вход А меньше чем В. Аналоговый коммутатор входов – выбирает аналоговые сигналы для сравнения. Один сигнал берется с внешнего входа AIN1, в качестве второго берется или

сигнал с внешнего входа AIN2 или образцовое внутреннее напряжение. Компаратор – сравнивает аналоговые сигналы. Регистр данных – программно доступный регистр, в битах которого сохраняются результаты сравнения одного или нескольких компараторов. Схема фиксации перепада – определяет изменение одного из бит в регистре данных (выхода одного из компараторов) и вырабатывает по этому событию запрос прерывания.

Область применения аналогового компаратора:

- контроль превышения допустимых значений переменных управляемого процесса;
- обнаружение (формирование) фронтов внешних сигналов;
- встроенные схемы контроля напряжения питания системы и др.

6.1.3. Параллельные порты и дискретный ввод-вывод

Микроконтроллер имеет некоторое количество одиночных линий ввода/вывода, которые объединены в многоразрядные (4, 8 или 16 линий разрядов) параллельные порты ввода/вывода [1]. Разрядность параллельных портов может быть нестандартной, например, 5-разрядный порт у микроконтроллера PIC16F84. Через порты процессорное ядро взаимодействует с различными внешними устройствами – считывает значения входных сигналов и устанавливает значения выходных сигналов.

В памяти МК каждому порту ввода/вывода соответствует свой адрес регистра данных. Обращение к регистру данных порта ввода/вывода производится так же, как и обращение к памяти данных. Параллельные порты могут использоваться для ввода-вывода многоразрядного кода данных для быстрого обмена с находящимися в непосредственной близости от МК периферийных устройств. Кроме того, во многих МК линии параллельных портов могут служить для ввода-вывода отдельных дискретных сигналов, для них предусмотрены битовые команды процессора, позволяющие опросить или установить отдельные разряды портов.

По алгоритму обмена различают параллельные порты следующих типов [2]:

1. С программно-управляемым (программным) вводом-выводом – установка и считывание данных определяется только ходом вычислительного процесса. Нет защиты от повторного считывания-записи одного и того же (не изменившегося) значения на выводе и считывания-записи во время переходного процесса на выводе.
2. Со стробированием – каждая операция ввода-вывода подтверждается импульсом синхронизации (стробом) со стороны источника сигнала (при выводе – процессор, при вводе – внешнее устройство). Считывание информации приемником происходит только по стробу, что позволяет защититься от приема данных во время переходного процесса входного сигнала. Пример: порт PSP (Parallel slave port) в ОКМЭВМ PICmicro.

3. С полным квитированием. Данный режим чаще всего используется для обмена данными с другой вычислительной системой по параллельной шине. Кроме сигналов синхронизации со стороны передатчика используются сигналы подтверждения (готовности к следующему обмену) со стороны приемника. Это позволяет управлять интенсивностью обмена обоим взаимодействующим сторонам и предотвращает потерю данных, когда одна из них перегружена. Пример порта с квитированием, порт LPT персонального компьютера.

В задачах управления очень важным является ввод и вывод отдельных дискретных сигналов, производимый либо непосредственно в ПЛК, либо через специализированные модули ввода-вывода. Дискретные входные сигналы отражают логическое состояние управляемого объекта или процесса, источниками которых являются датчики с дискретными выходами или устройства ввода управляющей информации от оператора. Выходные дискретные сигналы являются управляющими воздействиями, реализуемыми посредством исполнительных устройств со входом типа «вкл-выкл» и подаваемыми на объект управления с целью изменения его состояния в желаемом направлении.

В зависимости от реализуемых функций различают следующие типы параллельных портов:

- однонаправленные порты, предназначенные только для ввода или только для вывода информации;
- двунаправленные порты, направление передачи которых (ввод или вывод) определяется циклом чтения или записи МС;

Однонаправленные порты ввода-вывода могут иметь либо однотактную выходную схему (с открытым коллектором или стоком), либо двухтактную выходной схемой (с комплементарной парой).

Порты вывода с двухтактной выходной схемой являются самыми распространенными и реализованы, например, в семействах Atmel AVR, Microchip PICmicro, AMD AM186, Motorola HC08, HC11. Они дают возможность отключаться от линии при отсутствии обмена, т. е. обеспечивают z-состояние (рис 6.13).

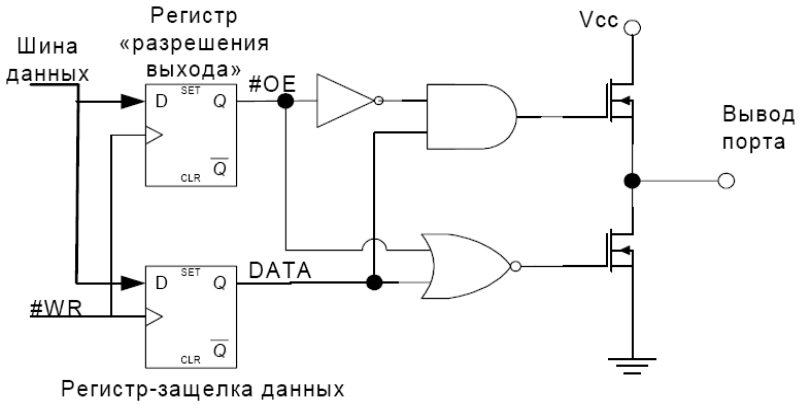


Рис. 6.13. Двухтактная схема порта

Выходные данные записываются в регистр-защелку (тиргтер) данных по внутреннему сигналу записи WR и через простейшую логическую схему управляют выходными транзисторами. Если в регистр записано значение «1», то открыт верхний по схеме транзистор, а нижний закрыт: на выходе Vcc, то есть «1». Если в регистр записано значение «0», то открыт нижний по схеме транзистор, а верхний закрыт: выход соединен с минусовой шиной питания, то есть там установлен «0». Верхний по схеме регистр управляет сигналом OE (Out Enable). Если в регистр записан «0», то схема работает, как было описано выше. Если записана «1», то оба транзистора закрываются и схема переводится в z-состояние.

Порты вывода с однотактной выходной схемой применяются, например, в семействе MCS-51, они имеют более простую внутреннюю схему (рис. 6.14).

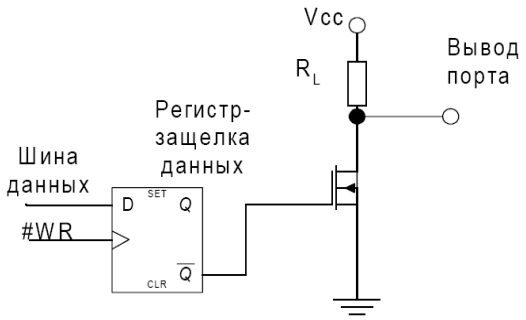


Рис. 6.14. Однотактная схема порта

Когда в регистр-защелку (тиргтер) записано значение «1», транзистор закрыт и на выходе через резистор R_L устанавливается Vcc – логическая «1». Когда же в регистр-защелку записан «0», открывается транзистор и соединяет выход с минусовой шиной питания, то есть там устанавливается «0». При этом резистор R_L оказывается подключенным между шинами питания.

Если резистор R_L вынести из схемы порта, то мы получим выход по схеме с открытым коллектором или стоком. Такая схема применяются во многих семействах микропроцессоров, например, AMD Am186, PICmicro.

Двунаправленный порт дискретного ввода-вывода имеет один и тот же адрес для ввода и для вывода, его режим работы (приема или передачи) задается внутренними сигналами МК – стробами чтения RD и записи WR (рис. 6.15).

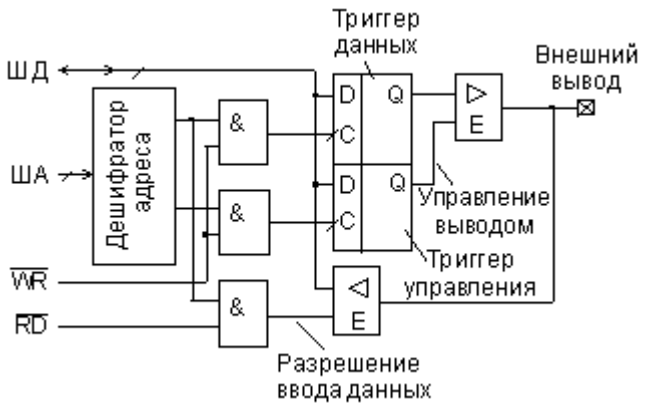


Рис. 6.15. Структурная схема двунаправленного порта ввода-вывода

Триггер управления разрешает вывод данных на внешний вывод. В современных МК, как правило, обеспечивается индивидуальный доступ к триггерам данных и управления, что позволяет использовать каждую линию независимо в режиме ввода или вывода.

Необходимо обратить особое внимание на то, что при вводе данных считывается значение сигнала, поступающее на внешний вывод, а не содержимое триггера данных. Если к внешнему выводу МК подключены выходы других устройств, то они могут установить свой уровень выходного сигнала, который и будет считан вместо ожидаемого значения триггера данных.

Порты дискретного ввода-вывода микроконтроллера должны иметь цепи защиты и развязку от внешних линий датчиков и исполнительных устройств для обеспечения требуемого уровня помехозащищенности, а также схемы формирования стандартного логического сигнала. На рис. 6.16 приведена схема подключения внешней линии к порту дискретного ввода.

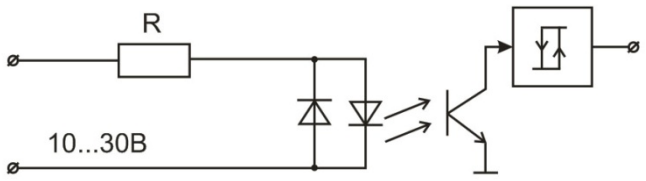


Рис. 6.16. Цепь развязки дискретного порта ввода

Цепи защиты в зависимости от условий эксплуатации должны обеспечивать защиту от перенапряжения, от переполюсовки, молниезащиту и т. п. Цепи гальванической или кондуктометрической развязки призваны исключить проблемы наводки паразитного сигнала, несогласованности «земель» различных источников и др.

Кроме того входной дискретный сигнал должен отвечать стандартным параметрам уровня логических «0» и «1», характеристиками перехода из одного логического состояния в другое. Как правило, входной сигнал пропускается через триггер Шмитта или схему защиты от дребезга. Далее,

информация по внутреннему стробу фиксируются в регистре данных, с выхода которого, в свою очередь, данные считываются процессором. Триггер Шмидта имеет гистерезис по уровню входного напряжения и предотвращает многократное переключение входных схем при пологом фронте сигнала или при наличии помех помехах. Схема защиты от дребезга вводит инерционность переключения и отсекает реакцию на короткие по длительности импульсы. К входу порта также могут подключаться так называемые «резисторы поддержки» логической «1» (Pullup) или логического «0» (Pulldown). Эти резисторы предназначены для переведения входов в устойчивое состояние «0» или «1» и предотвращения произвольных переключений от помех в моменты, когда на них (входы) не подается внешний сигнал, например, неиспользуемых и неподключенных к внешним схемам входов («открытых входов») [2].

6.2. Последовательные интерфейсы и сетевые средства передачи данных

Построение распределенных информационно-управляющих систем требует применения различных коммуникационных устройств, обеспечивающих обмен данными в вычислительных и телекоммуникационных сетях. Обмен информацией на больших расстояниях, в том числе в информационных сетях предполагает последовательный принцип передачи данных, что значительно упрощает приемо-передающую аппаратуру и снижает количество кабелей (оптических или электрических) или требования к каналу (например, радиоканалу).

Последовательные порты называются COM-портами. Программируемый логический контроллер, как правило, имеет несколько COM-портов, режимы которых задаются программно для различных целей. Например, порт программирования ПЛК для ввода управляющей программы, порт для подключения периферийных модулей ввода-вывода, порт для связи с верхним уровнем управления и т. д. Рассмотрим подробнее виды и режимы работы COM-портов.

6.2.1. Последовательные интерфейсы

Последовательный ввод-вывод используется для обмена данными между микроконтроллером и удаленными периферийными устройствами ввода-вывода. Данные передаются в последовательном коде, то есть биты передаются во временной последовательности друг за другом по одному каналу связи. Это даёт возможность использовать один канал связи вместо нескольких в случае параллельного кода. Однако при этом снижается быстродействие канала связи.

Различают полудуплексные и дуплексные каналы последовательной связи. В случае полудуплексного канала данные передаются по одному и тому же каналу в обе стороны, но в каждый конкретный момент времени только в одну сторону. В случае дуплексного канала данные передаются по

двум каналам (в одну сторону – по одному, в другую – по другому). При этом появляется возможность передавать информацию в обе стороны одновременно.

По методу синхронизации различают синхронные и асинхронные каналы связи. При синхронном методе передачи каждый передаваемый бит данных синхронизируется импульсом в канале синхронизации (рис. 6.17).

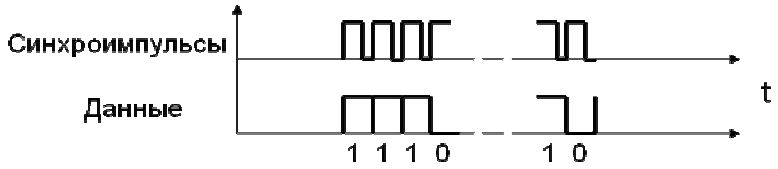


Рис. 6.17. Синхронный способ передачи данных

При этом по каналу данных передаются только собственно данные. Такой метод передачи позволяет обеспечить скорость передачи порядка 1 Мбит/сек. Однако, для его организации необходим дополнительный канал передачи синхроимпульсов.

При асинхронном методе передачи канал синхронизации отсутствует. Это вызывает необходимость передавать по каналу данных дополнительную информацию для синхронизации приемной стороны (рис. 6.18).

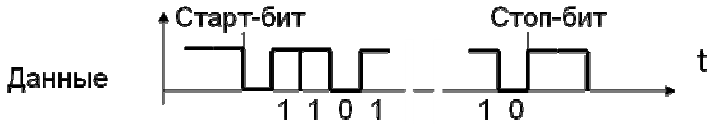


Рис. 6.18. Асинхронный способ передачи данных

Первым в посылке передается «Старт-бит» нулевого уровня. Приняв его, приемная сторона подстраивает фазу своих синхросигналов в соответствии с моментом прихода «Старт-бита». После него один за другим следуют биты данных (младшими битами вперед). В конце цепочки данных может следовать бит четности для проверки правильности приема. Заканчивается посылка «Стоп-битом» единичного уровня. Частота передачи, то есть время передачи одного бита заранее должны быть согласованы на передающей и приемной сторонах. Такой метод передачи позволяет обеспечить скорость передачи до 100 Кбит/сек.

В асинхронном режиме COM-порту требуется всего две информационные линии для обмена данными: передача с вывода TxD, а прием – на вывод RxD. Синхронизация бит и байт осуществляется с помощью двух встроенных таймеров МК – T1 и T2. Скорость передачи определяется их частотой тактирования, в момент завершения передачи очередного байта таймером генерируется сигнал запроса прерывания.

Микроконтроллер может содержать либо отдельные синхронные и асинхронные порты ввода-вывода, либо универсальный порт, способный работать в обоих режимах.

Одним из наиболее широко используемых стандартов при реализации последовательного интерфейса является стандарт RS-232. Стандарт предусматривает двухточечную и многоточечную коммуникацию в полудуплексном и дуплексном режимах на расстояния до 15 м. При скорости до 20 Кбод. При передаче используются уровни сигналов -12В и +12В.

Кроме передаваемых и принимаемых данных стандарт предписывает следующий набор управляющих сигналов:

- ✓ RTS – запрос передатчика, генерируется передатчиком, когда он должен передать данные по линии, активное состояние сохраняется до конца передачи;
- ✓ CTS – сброс передатчика, Используется приемником для информирования передатчика относительно того, готов ли приемник к приему передаваемых данных;
- ✓ DSR – готовность модема, когда модем включен, этот вывод находится в активном состоянии;
- ✓ DCD – флаг обнаружения несущей частоты модема, этот сигнал используется для информирования передатчика о том, что каналом можно пользоваться, и обычно активизируется в тех случаях, когда уже выдан сигнал RTS;
- ✓ DTR – готовность терминала, в активном состоянии указывает на то, что терминал находится в режиме взаимодействия с системой и, следовательно, связь возможна;
- ✓ RI – индикатор вызова, активизируется модемом, когда последний обнаружил поступивший по телефонной линии вызов.

Указанные сигналы управления используются для передачи последовательных данных через модем. Передача же данных в двоичном формате предполагает использование только двух информационных линий: TxD и RxD, – так называемый «нуль-модемный режим».

Микросхемы, реализующие интерфейс RS-232: MAX202, MAX232 фирмы Maxim, ADM232 фирмы Analog Devices и др. На рис. 6.19 представлена конфигурация RS-232 с минимальным количеством соединений между узлами (микросхемы MAX202, MAX232, ADM232 и др.)

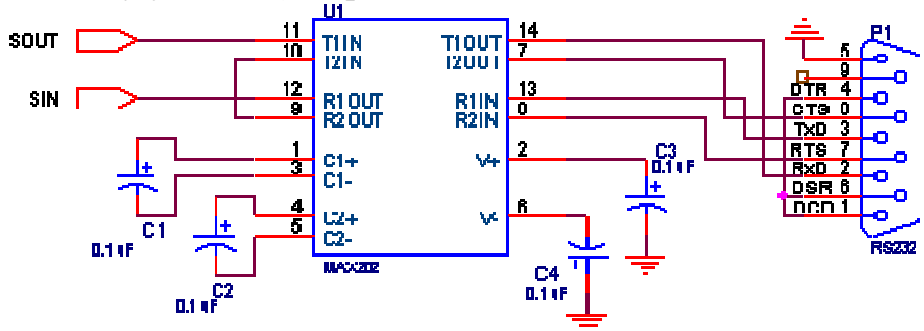


Рис. 6.19. Интерфейс RS-232 в «нуль-модемном» режиме

В условиях повышенных помех либо при большой длине кабеля надежность интерфейса RS-232 может оказаться недостаточной для

приемлемого качества коммуникаций. В этих случаях используется симметричный дифференциальный интерфейс RS-422 (рис. 6.20).

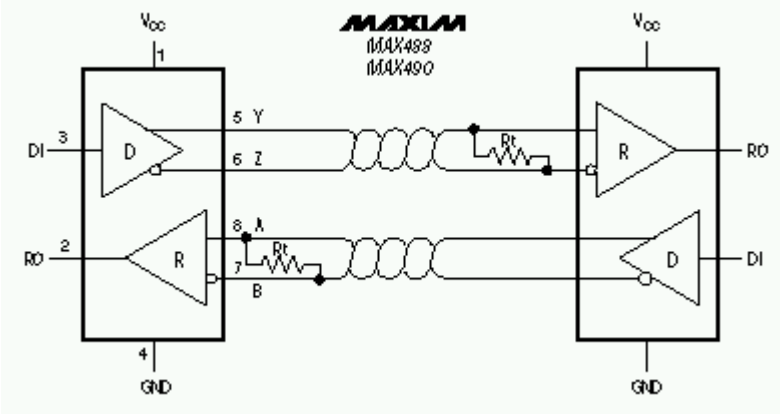


Рис. 6.20. Интерфейс RS-422

Симметричный дифференциальный формирователь, такой как MAX490 фирмы Maxim имеет два выхода. Один из выходов представляет собой буферизованный эквивалент входа формирователя, тогда как другой является его дополнением. Витая пара соединяет два этих вывода с двумя входами приемника (дифференциальный вход). Поскольку оба передаваемых сигнала в одинаковой степени подвергаются воздействию помех (синфазные помехи), то последние устраняются благодаря использованию дифференциального входа в приемнике. Этот интерфейс может эффективно функционировать в присутствии синфазных помех с амплитудой до 3В. При этом уровни передаваемых сигналов 0В и 5В. Допускается длина линии до 1200 м. При скорости передачи до 10 Мбод.

Интерфейс RS-485 является полудуплексным аналогом интерфейса RS-422, в нем одни и те же линии связи используются для приема и передачи сообщений (рис. 6.21).

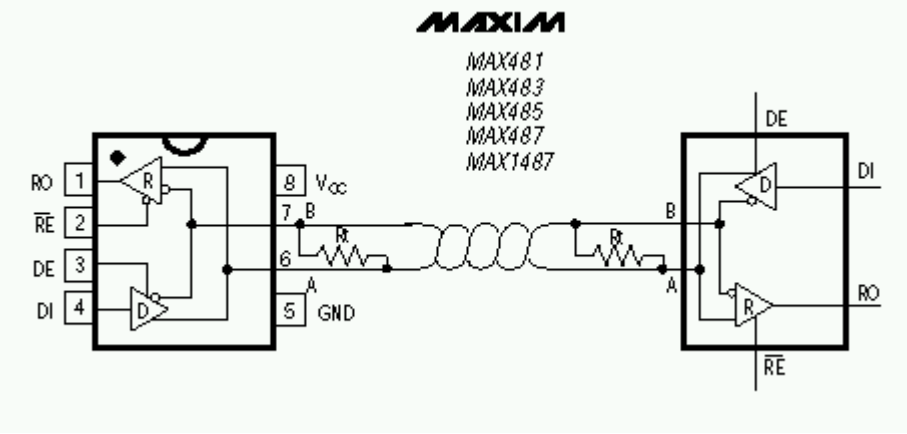


Рис. 6.21. Интерфейс RS-485

Таким образом, число линий связи в RS-485 такое же как и в RS-232 в «нуль-модемном» режиме. Примером микросхем, реализующих интерфейс RS-485 могут служить MAX485, ADM485 и др. Выводы DE и $\overline{RE}$ при активных

уровнях сигнала на них разрешают микросхеме работать на передачу или прием соответственно.

Интерфейсы RS-232, RS-422, RS-485 позволяют создавать многопользовательские топологии каналов связи. RS-422 – радиальный интерфейс (точка-точка) аналогичен по канальному протоколу стандартному COM-порту (RS-232), но отличается от него электрическими характеристиками, т. е. физическим уровнем модели OSI. Интерфейсы RS-422 и RS-485 имеют единые электрические параметры и обеспечивают дальность соединения до 1200 м.

RS-485 – магистральный интерфейс, обеспечивающий соединения «точка – много точек», по нему можно подключать множество внешних устройств: до 128 или до 256. Адресация узлов сети RS-485 поддерживается на прикладном уровне модели OSI. Каждый узел сети – модуль ввода-вывода или ПЛК – имеет свой уникальный адрес от 00 до FF, задаваемый при инсталляции системы путем установления микропереключателей (перемычек) на печатной плате модулей или сохраняется в энергонезависимой памяти при программировании. В каждый момент времени только один узел может быть инициатором общения, т. е. являться ведущим узлом (Master), остальные же узлы являются ведомыми (Slave). Ведущий узел – как правило, ПЛК – посылает сообщение-запрос для конкретного узла, обращаясь к нему по адресу. Модуль, принявший запрос, формирует сообщение-ответ для ведущего узла, в котором выдает требуемую информацию или подтверждает получение информации.

Пример сети RS-485 изображен на рис. 6.22.

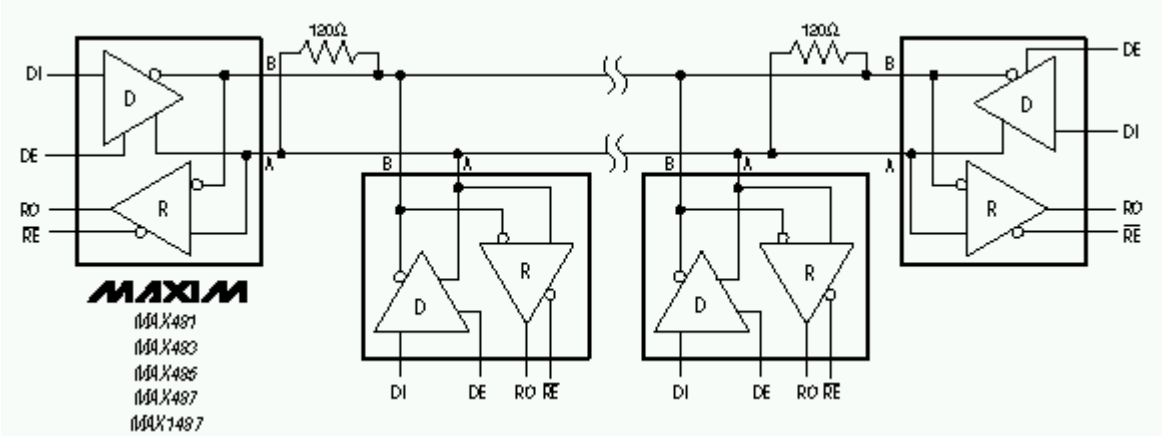


Рис. 6.22. Магистральная сеть RS-485

Магистральный интерфейс RS-485 служит для развертывания полевых сетей (FieldBus), предназначенных для взаимодействия с различными группами оборудования и модулями ввода-вывода. Если ПЛК взаимодействует с несколькими группами оборудования, например от разных производителей, то для каждой группы желательно выделить отдельную сеть RS-485, в которой действует свой протокол обмена. В сети у каждого устройства должен быть установлен свой уникальный адрес.

По полевым сетям могут передаваться как двоичные форматы данных, так и ASCII-форматы, в которых каждый байт трактуется как один из символов таблицы ASCII. Пример формата протокола при обмене сообщениями с модулями I-7000 фирмы ICP DAS (Корея) через последовательный интерфейс RS-485 приведен на рис. 6.23.

Рис. 6.23. Пример формата протокола сообщений

Надежность передачи данных по последовательному интерфейсу выше для символьного формата в сравнении с двоичным форматом, так как специальными символами (на рис. 6.23 – это $ и !) можно четко обозначить начало и конец сообщения. Биты данных могут передаваться только символами ‘0’…’9’, ‘A’…’F’, что также повышает вероятность обнаружения ошибок. Кроме того, надежность передачи обеспечивается проверкой правильности подсчета контрольной суммы, дисциплиной общения на прикладном уровне – «запрос-ответ» и др. механизмами.

6.2.2. Телекоммуникационные и сетевые средства передачи данных

Телекоммуникационные и сетевые устройства (модули) служат для преобразования и передачи информации на расстояние, а так же организации взаимодействия полевых подсетей. Можно выделить несколько групп коммуникационных модулей:

- преобразователи/адаптеры интерфейса: Ethernet/RS-485, USP/RS-485, RS-232/RS-485 и др. (рис. 6.24).

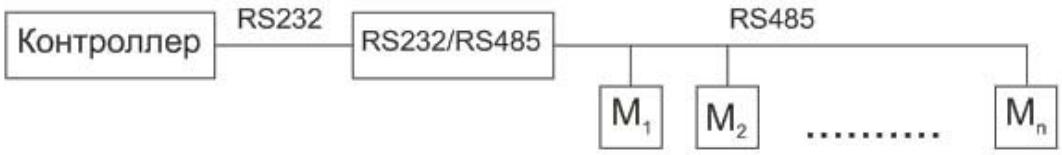


Рис. 6.24. Применение преобразователя интерфейса RS232/RS485

- повторители (repiter) предназначены для увеличения протяженности линий связи, ограниченной конкретным интерфейсом или, например, организация ответвлений в линейной магистрали RS-485;
- модемы служат для передачи цифровой информации на большие расстояния, модулированной определенным образом. Серийно

выпускаются модемы для электропроводных линий передачи, оптико-волоконных линий и беспроводных сетей.

Тональные модемы служат для передачи информации по существующим телефонным или другим проводным линиям связи большой протяженности. В промышленных сетях модемы могут использоваться как удлинители проводных линий. Сигнал из цифровой формы преобразуется в модулированную тональную форму, основанную на частотной модуляции. Такой сигнал при сравнительно низкой плотности передачи имеет очень высокую помехоустойчивость, что и обуславливает их широкое применения для решения задач телекоммуникаций.

Одной из разновидности модемов является радиомодем. По средствам радиомодемов строятся беспроводные радиосети в тех случаях, когда объекты распределены, и имеются трудности в прокладке проводных или оптоволоконных линий из-за их дороговизны или особых условий. Структура радиомодемного модуля Спектр-433 приведена на рис. 6.25, где введены следующие обозначения:

- COM – последовательный интерфейс RS232/RS485;
- Бс – буфер сообщений последовательного интерфейса;
- МК – встроенный микроконтроллер для управления радиомодемом;
- Бп – буфер пакетов, передаваемых в эфир;
- М-Д-М – модулятор-демодулятор;
- п/п – приемопередатчик, передающий сигнал в эфир на несущей частоте 433 МГц, либо принимающий радиосигнал.

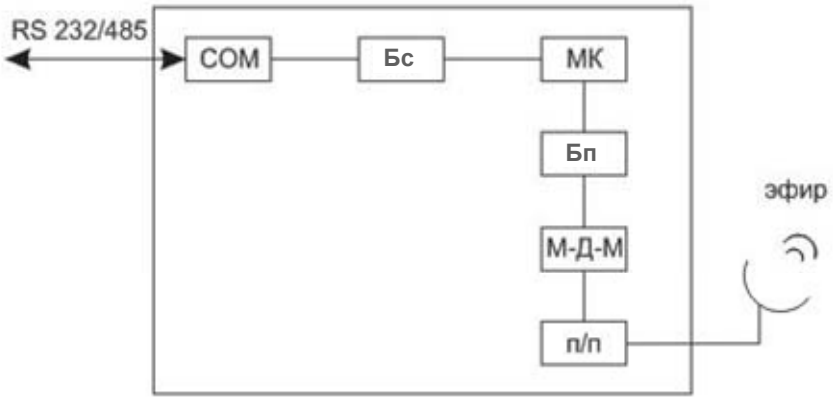


Рис. 6.25. Структура радиомодема Спектр-433

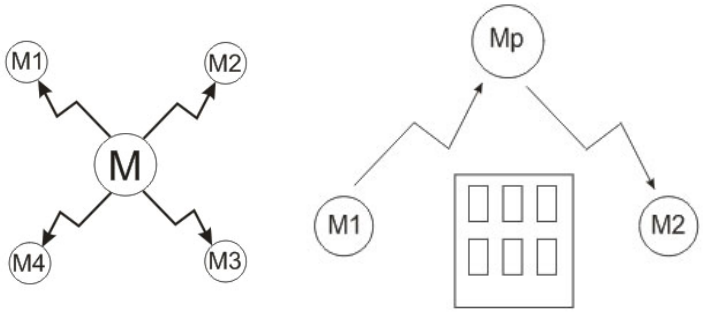
С целью повышения надежности передачи данных сообщения разбиваются на пакеты небольшой длины (не более 256 байт). Эта функция реализована микроконтроллером модема программно. Кроме этой функции МК производит буферизацию данных, входящих по COM-порту, и буферизацию пакетов, приходящих по радиоэфиру, осуществляет сборку этих пакетов в исходные сообщения (этот процесс называется конкатенацией) и помехоустойчивое кодирование, не допускает коллизий в радиоэфире, для этого перед отправкой очередного пакета модем проверяет

свободен ли эфир, обеспечивает различные режимы функционирования модема:

1. Режим «прозрачный». Канал связи скрыт от внешнего оборудования и программного обеспечения. Все, что поступило на COM-порт, передается в эфир, а все, что принимается из эфира, отправляется на последовательный порт. Этот режим существует для удлинения существующих каналов. Адресация и логическое взаимодействие объектов в системе обеспечивается на более высоком прикладном уровне внешним оборудованием и программным обеспечением.
2. Режим «пакетный». Логическое взаимодействие и адресация объектов обеспечивается на уровне радиосети. Для этого внешние оборудование и ПО должны поддерживать внутренний протокол обмена данными модема. В этом режиме пакеты адресуются конкретному модему в радиосети, который подтверждает безошибочное получение пакета.
3. Режим «командный». В этом режиме выполняется настройка и конфигурирование модемов, тестирование и выполнение различных команд.
4. Режим прямого доступа. В этом режиме модем предоставляет внешнему оборудованию прямой доступ к эфиру, точнее к модулятору и демодулятору встроенного приемопередатчика. Для обмена данными внешнее оборудование может использовать собственные протоколы, адресацию, способы помехоустойчивого кодирования, т. е. минуется обработка данных микроконтроллером модема.

Для организации работы модемов в радиосети используется несколько режимов:

- ✓ широковещательный режим, когда модем передает в эфир пакеты сразу для всех модемов, в этом режиме невозможно гарантировать доставку пакетов (рис. 6.26-а);
- ✓ режим «точка-точка», когда пакет передается адресно, конкретному модему, который обязан вернуть пакет подтверждения приема и достоверности данных, при этом гарантируется доставка пакетов;
- ✓ режим ретрансляции, используется в модеме M_P (рис. 6.26-б) в случае отсутствия прямой видимости между модемами M_1 и M_2. В COM-порт модема M_P в режиме ретрансляции информация не транслируется.



а) *б)*

Рис. 6.26. Организации работы радиосети:
а – широковещательный режим; б – режим ретрансляции

Методы повышения надежности передаваемых данных: уменьшение длины пакетов, дублирование пакетов, снижение скорости передачи, подтверждение приема с проверкой контрольных суммарных пакетов, применение помехоустойчивого кодирования (коды Рида-Соломона и Хемминга), алгоритмов перемежения, рандомизации (скремблирования), подтверждение достоверности обмена на прикладном уровне.

Снижение скорости обмена данными в телекоммуникационных модулях может привести к срыву режима реального времени, например, из-за помех в радиоэфире потребуется повторная передача пакета данных радиомодемом. Поэтому, необходимо учитывать возможные временные задержки не только при реализации алгоритмов управления в ПЛК, но и при передаче управляющей информации по телекоммуникационным каналам связи.

Вопросы для повторения

1. Укажите виды цифро-аналоговых преобразователей.
2. В чем особенность реализации однобитного ЦАП?
3. Укажите виды аналого-цифровых преобразователей.
4. Приведите примеры использования однобитных АЦП.
5. Как производится многоканальный ввод аналоговых сигналов?
6. Какие цепи развязки используются для ввода-вывода аналоговых сигналов?
7. Для решения каких задач применяется аналоговый компаратор? Укажите его принцип построения.
8. Как различаются параллельные порты ввода-вывода по алгоритму обмена?
9. Особенности реализации однотактной и двухтактной схем порта вывода.
10. Принцип построения дискретного двунаправленного порта ввода-вывода.
11. Особенности цепей развязки дискретного ввода-вывода.
12. Какие стандартные последовательные интерфейсы Вы знаете? В чем их особенности?
13. Зачем используется старт-бит и стоп-бит в последовательном интерфейсе?
14. Какие сигналы имеет интерфейс RS-232 в нуль-модемном режиме?
15. Приведите пример протокола обмена сообщениями по последовательному интерфейсу.
16. Укажите назначение коммуникационных модулей.
17. Что такое модем? Укажите его область применения.

18. В каких режимах функционирования может работать радиомодем?
19. Какие режимы работы радиомодема используются для организации работы в сети?

Практикум (лабораторный)

Лабораторная работа №1. Изучение и программирование DSP SPEEDY-33 в задачах цифровой обработки сигналов

Требования к содержанию, оформлению и порядку выполнения

Перед выполнением лабораторной работы студент должен изучить теоретическую часть курса. Студент, выполняющий лабораторную работу в составе с реальным лабораторным стендом, обязан получить к ней допуск путем собеседования с преподавателем, который заключается в подтверждении готовности студента к ее выполнению, а именно, знание основ теории по данной теме, порядка выполнения лабораторной работы, особенностей и правил работы с лабораторным стендом. Преподаватель определяет вариант задания для выполнения лабораторной работы конкретным студентом.

Выполнение лабораторной работы производится в 2 этапа:

1. Разработка программы и отладка ее на программном эмуляторе того продукта, который предназначен для данного стенда.
2. Демонстрация преподавателю работы отлаженной программы, которая должна функционировать в составе с реальным лабораторным стендом в строгом соответствии с полученным заданием (только для студентов очной формы обучения).
3. Сохранение созданного проекта, содержащего разработанную и отлаженную программу, в вышеуказанную папку.

После успешного выполнения лабораторной работы студент должен оформить отчет в соответствии с предъявляемыми требованиями и осуществить ее защиту, проводимую в форме собеседования с преподавателем. Цель защиты – подтверждение факта самостоятельного выполнения лабораторной работы и определение степени усвоения студентом теоретического материала по данной теме.

Отчет о выполнении лабораторной работы должен быть выполнен на бумаге формата А4 и содержать следующие разделы:

- титульный лист;
- задание;
- краткие теоретические сведения – не более 1 стр.;
- выполнение – в этом разделе должны быть представлены основные результаты лабораторной работы;

- выводы – отразить особенности применения функциональных элементов, включённых в схему, и их влияние на качество и скорость обработки аудио сигналов, оценить величину запаздывания.

Основные результаты выполнения лабораторной работы:

1) итоговая структурная схема проекта;
2) общая постановка задачи, разбитая на подзадачи, в соответствии с индивидуальным заданием преподавателя;
3) снимок экрана Front Panel on SPEEDY-33 с описанием назначения элементов органов управления и индикации;
4) описание выполняемых функций элементов органов управления и индикации, представленное в виде таблицы;
5) снимок экрана Block Diagram on SPEEDY-33 с описанием назначения элементов и выполняемых ими функций в реализуемой схеме. Описание можно представить в виде блок схемы выполняемой функции;
6) графики сигнала и его спектра, соответствующие порядку выполнения работы, при реализации индивидуального задания. К каждому графику должны быть пояснения. Масштаб графика спектра должен быть выбран из соображений наглядности отражения свойств сигнала;
7) результаты вычислений и выводы.

Громоздкие схемы, графики, снимки экранов и рисунки можно вынести в приложение отчета.

Общая постановка задачи

Основной целью выполнения лабораторной работы является изучение и получение навыков графического программирования цифрового сигнального процессора (DSP) NI SPEEDY-33 при решении задач цифровой обработки аудио сигналов в режиме реального времени.

Описание лабораторного стенда

Типовые Характеристики DSP

Класс DSP (Digital Signal Processor, цифровой сигнальный процессор) предназначается для обработки разнообразных потоков информации (звуковых, видео – или других видов сигналов). Типичные характеристики DSP таковы:

• широкая системная шина (16 или 32 бита),

• порты цифрового ввода-вывода,

• многоканальные АЦП/ЦАП с хорошим разрешением и скоростью,

• аппаратной реализацией основных функций анализа данных,

• прогрессивной системой команд ассемблера, ориентированной на области

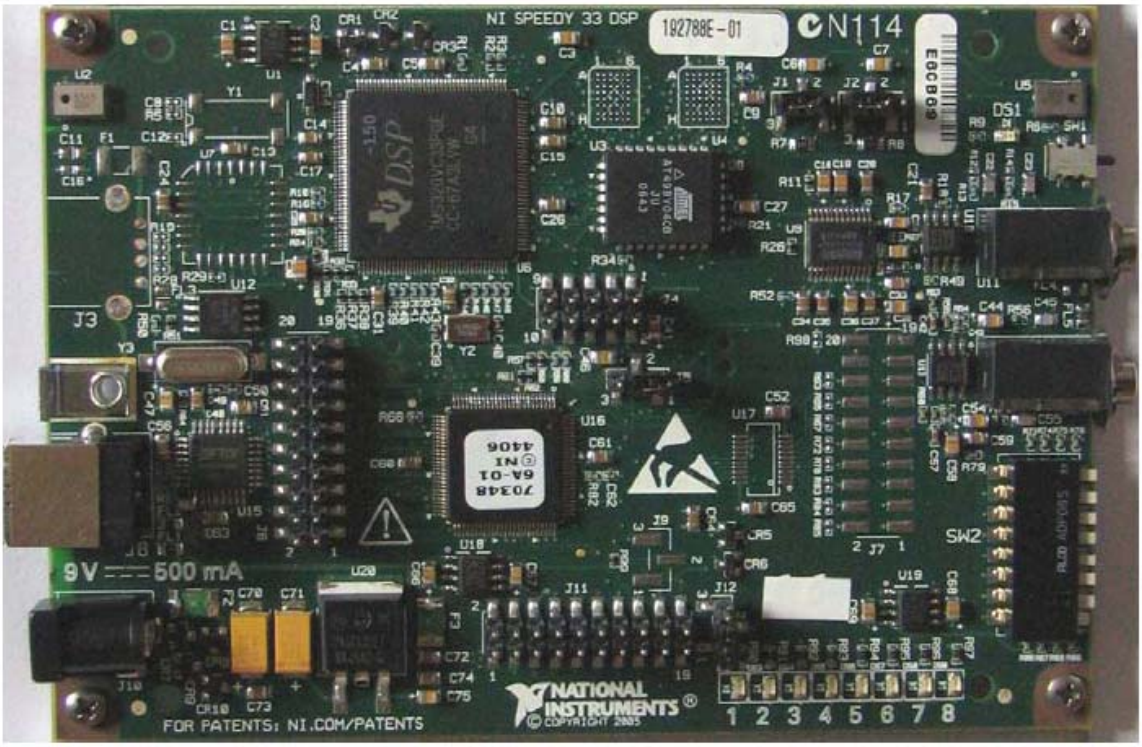
применения процессора и языков высокого уровня (как правило, C),

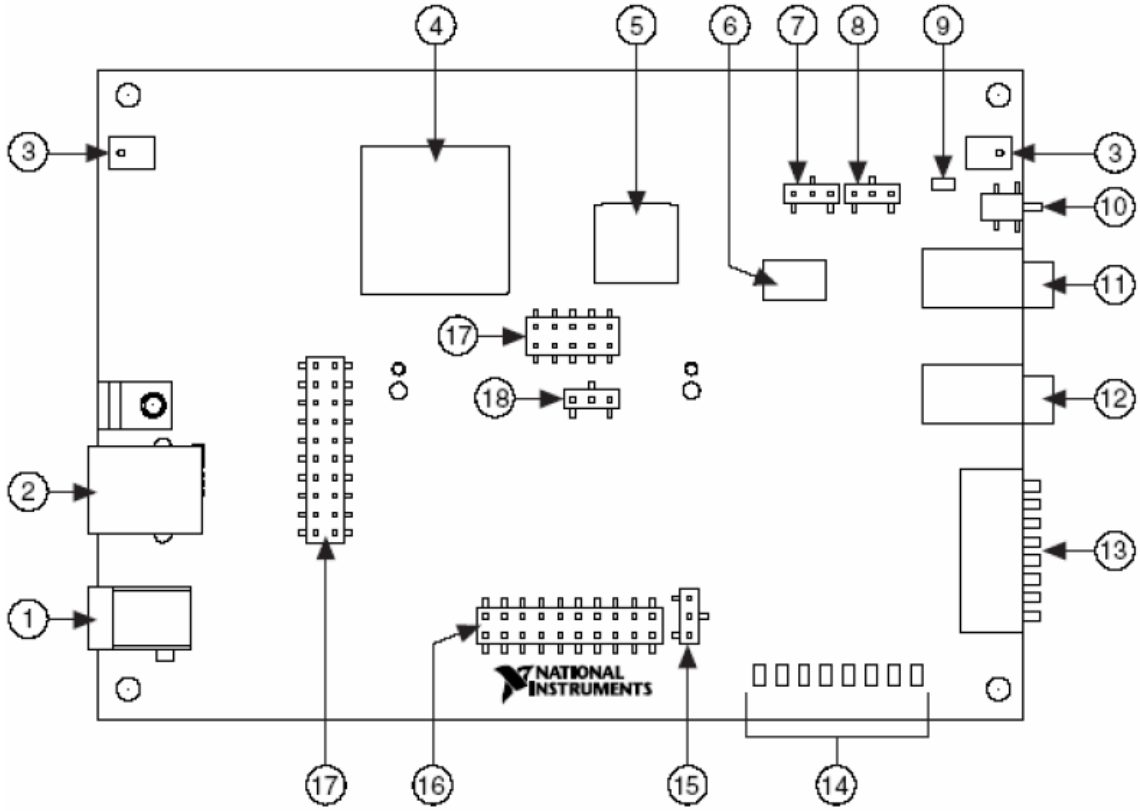
• высокая тактовая частота (от 40 до 1000 MHz),

• DMA-шина (Direct Memory Access) для подключения дополнительной периферии (оперативная память, аудиокодеки, контроллеры интерфейсов, и т.д.).

“Plug & Play” платы на основе DSP

Построение любого устройства, основанного на однокристальных ЭВМ (а именно, в этот класс продуктов входит DSP), начинается с проектирования топологии прибора. Далее следует разработка принципиальной электрической схемы, изготовление макетов печатных плат и их производство. Разработка программного обеспечения для используемых программируемых компонентов проходит параллельно с этим, однако настройка под конкретную конфигурацию возможна только на завершающей стадии разработки прибора. Существует способ избежать задержек и лишних затрат на проектирование материальной части прибора, если воспользоваться готовыми платами. Такие платы разрабатываются для решения определённого круга задач, и поэтому на них устанавливается всё необходимое для разработки конечного продукта. Например, существуют платы на базе DSP, предназначенные для обработки звука или видео. Типичным представителем такого решения является плата SPEEDY-33 компании National Instruments (NI).





Данная плата обладает следующими характеристиками:

- процессор: 150 MHz VC33 TI DSP;
- память программ: 34 kБайт основной и 252 kБайт расширенной;
- Flash-память: 512 kБайт;
- панель светодиодов и DIP-переключателей;
- 8 цифровых линий ввода-вывода;
- АЦП/ЦАП: 2 канала 16 бит, 48 kHz;
- возможность ввода информации как со встроенных микрофонов, так и с LineIn;
- интерфейс: USB 1.1 (FullSpeed, 12 Мбит).

Набор элементов, присутствующих на SPEEDY-33:

1 – автономное питание (при отладке и создании проекта, питание платы осуществляется через USB порт, а в случае автономной работы (без связи с компьютером) необходим внешний источник питания) (J10);

2 – USB порт (J8);

3 – встроенные внешние микрофоны (аналоговый ввод) (U2,U5);

4 – DSP (U6);

5 – Flash-память (U8);

6 – стерео A/D, D/A (U9);

7,8 – джамперы J1, J2;

9 – светодиод, показывающий включена плата или нет (DS1);

10 – кнопка Reset (SW1);

11, 12 – аудио порты ввода/вывода (U11, U14);

13 – DIP переключатели (SW2);

14 – линейка светодиодов (DS2-DS9);

15 – джампер J12;

16 – расширенный разъем ввода/вывода цифрового сигнала (J11);

17 – расширенный разъем ввода/вывода (аналоговый сигнал). В частности для подключения дочерних A/D и D/A модулей (J4, J6);

18 – джампер J5.

Описание джамперов (перемычек):

Переключатель	Microphone Level	Line Level
JP1 (Left Channel)	JP1-2 to JP1-1	JP1-2 to JP1-3
JP2 (Right Channel)	JP2-2 to JP2-1	JP2-2 to JP2-3

JP1, JP2 – служат для переключения внутреннего усиления (при использовании линейных входов необходим Line Level)

Переключатель	PCM3003 Stereo CODEC	Expansion A/D
JP3 (DR Selection)	JP3-2 to JP3-1	JP3-2 to JP3-3

JP3 – осуществляет выбор режима преобразования входного сигнала

Переключатель	Enable FLASH Boot	Disable FLASH Boot
JP4 (FLASH Boot)	JP4-1 to JP4-2 (Shorted)	(Open)

JP4 – определяет будет ли из FLASH памяти загружаться программа при подключении питания или нет

Переключатель	Enable FLASH Write	Disable FLASH Write
JP5 (FLASH Write Enable)	JP5-1 to JP5-2 (Shorted)	(Open)

JP5 – защита записи во FLASH память

Принципы программирования DSP

Написание программного обеспечения для DSP можно производить на ассемблере используемого кристалла. В этом случае необходимы глубокие знания архитектуры и системы команд процессора. Однако, переходя на другое семейство или поколение процессоров, придётся в значительной мере изучать всё заново – изменение топологии процессора сразу же потребует изменений кода. Кроме этого, код на ассемблере чрезвычайно громоздок и сложен для чтения «непосвящённому» человеку.

Второй способ создания программного обеспечения заключается в использовании языков высокого уровня. Для получения бинарного кода программы используются компиляторы, ориентированные на семейства используемых процессоров. В этом случае при изменении аппаратной платформы потребуется намного меньше исправлений и времени для перестройки проекта, поскольку конечный ассемблерный код, напрямую

зависящий от платформы, создаётся без явного участия разработчика. Однако и такой способ программирования не лишён недостатков. Хотя код написанный, например, на С читается намного легче ассемблера, но при расширении проекта могут возникать сложности с многообразием файлов исходных кодов и функций, в них расположенных.

LabVIEW DSP Module как средство разработки программного обеспечения для плат на основе TI DSP

Основанное на LabVIEW, данный пакет представляет собой среду для графического программирования встраиваемых (embedded) систем. Для этого в среду добавлен специальный компилятор, позволяющий переводить код на языке LabVIEW в ассемблерный код DSP. В настоящий момент LabVIEW DSP Module поддерживает 3 платы на основе Texas Instruments DSP:

- SPEEDY-33, National Instruments,
- SPEEDY-33, Hyperception,
- DSK6713, Spectrum Digital,
- DSK6711, Texas Instruments.

Работа с этими платами схожа с работой с DAQMX-платами: все необходимые замеры выполняет плата, а LabVIEW служит лишь для отображения полученных данных. Однако существует очень важное отличие – если в LabVIEW все операции (обработка, анализ и т.п.) над данными исполнялись на процессоре компьютера, то в данном случае вся нагрузка исполнения кода ложится на DSP.

Гибкость LabVIEW DSP Module достигается за счёт использования специальных конфигурационных файлов, описывающих особенности каждой из трёх плат. В этих фалах описаны наборы доступных функций, типы входных и выходных данных.

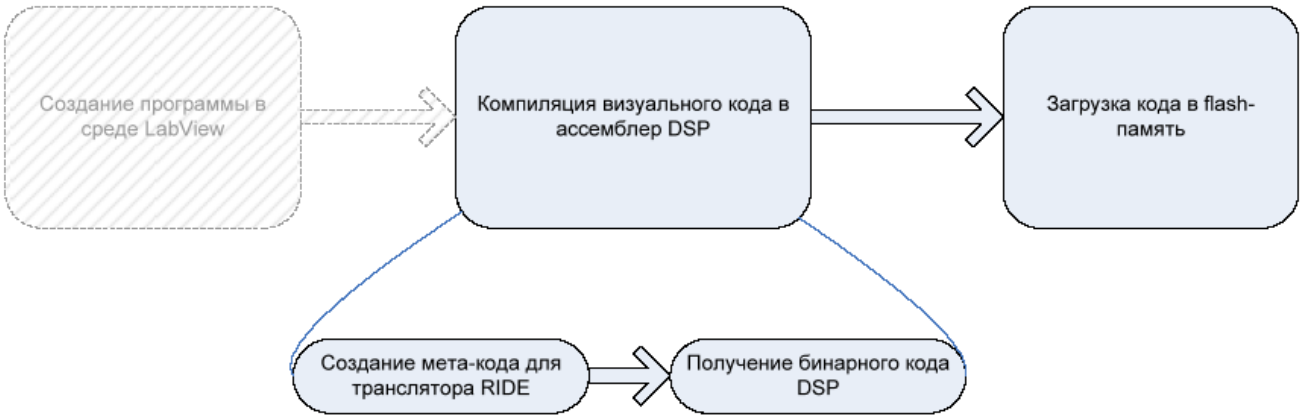
Программисту доступны все основные средства разработки виртуальных приборов:

- элементы управления потоком выполнения программы,
- арифметические, логические, элементы сравнения,
- работа с массивами данных,
- генерация основных функций,
- временные и частотные окна,
- преобразования Фурье, Гильберта и Хартли, временной анализ.

Кроме этого существует специальное меню примитивного обмена, позволяющее общаться с линиями ввода-вывода платы.

Запуск программы. Режим автономной работы

Процесс запуска программы можно разделить на 3 этапа.

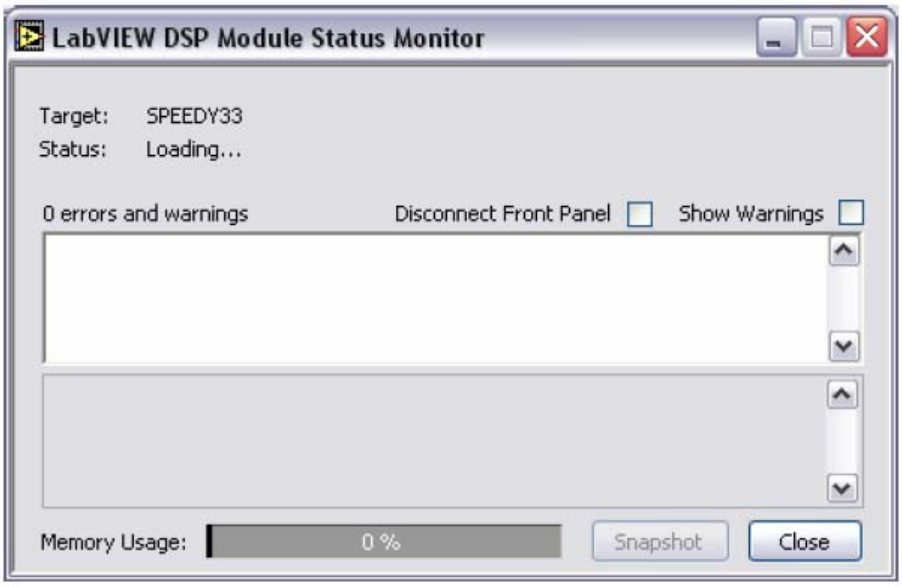


На первом этапе сборки проекта код LabVIEW собирается в код для RIDE, сторонней утилиты от NI или Hyperception. После этого полученный код передаётся в RIDE, где и происходит создание бинарного кода. Далее этот код загружается во flash-память процессора и подаётся команда на запуск. В этом случае работа может происходить в двух режимах:

- с подключённой лицевой панелью,
- с отключённой лицевой панелью.

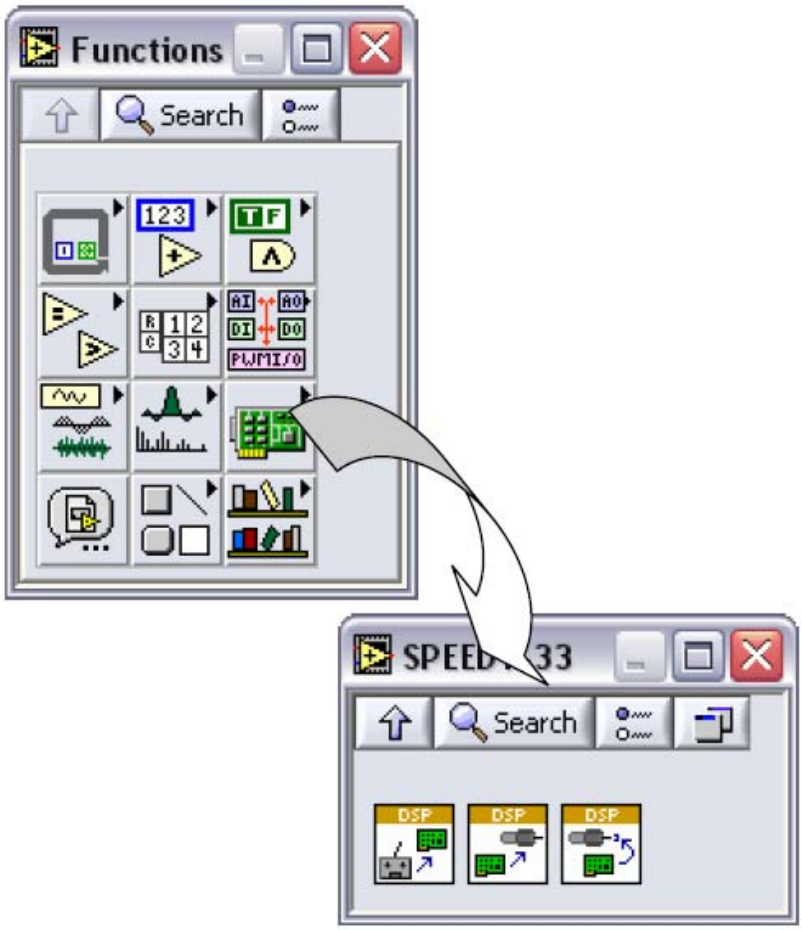
В первом режиме ежесекундно данные лицевой панели посылаются на DSP, а с него приходят данные для отображения. Поскольку в данном режиме DSP приходится «отвлекаться» от работы основной программы и тратить время на создание потока данных для компьютера, этот режим рекомендуется использовать только при необходимости изменения параметров программы. Сразу после перенастройки рекомендуется отключить лицевую панель. Данная настройка находится в специальном окне DSP Module Status Monitor, в котором отображаются все процессы взаимодействия с платой.

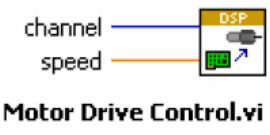
Описанные режимы работы требуют постоянного подключения к компьютеру для работы, что не является предметом нашего внимания, поскольку в этом случае можно применить решения NI для PC. Отличительной особенностью LabVIEW DSP Module является возможность запрограммировать плату так, чтобы ей не требовалось подключение к компьютеру вообще – программа сохраняется во встроенной flash-памяти платы. При этом в качестве значений всех элементов управления берутся их значения на момент компиляции. После переноса программы на плату работа с лицевой панелью невозможна, поскольку в бинарном коде были удалены все процедуры обмена с LabVIEW. Доступными для управления остаются элементы, расположенные на самой плате (DSP-панель переключателей, LED-линейка, порты цифрового IO). В итоге получается прибор, абсолютно независимый от компьютера.



Работа с PWM

В связи с тем, что DSP Module может работать с электровращательными приборами, в его панели функций, в отличие от функций LabVIEW, присутствует опция SPEEDY-33, осуществляющая регулировку и контроль за сервоприводами, двигателями моторами, и так далее.





Компонента управления вращательными электроприборами – Motor Drive Control – используется как для генерации сигнала управления скоростью, так и сигнала регулировки направления вращения.

При многократном использование этой компоненты на блок-схеме, список работающих устройств может быть расширен до 4 двигателей.

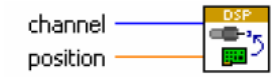
Управляющие сигналы создаются непосредственно самим SPEEDY-33 и могут быть считаны с выходных каналов цифрового порта J1 (см SPEEDY-33 Board Layout).

Наиболее ценной особенность Motor Drive Control является отсутствие необходимости использования элемента радиоуправления для предоставления информации о скорости и направлении вращения. Еще одной особенностью является обеспечение автономного функционирования каждого из каналов.

Motor Drive Control не имеет никаких каналов ввода-вывода. Поэтому все параметры элемента изменяются подобно тому, как меняется скорость и направление вращения оси вращательного электроприбора при использовании VAB (Visual Application Builder) параметров связующей логики. Этот элемент и служит основой в DSP SPEEDY-33 для создания цифровых выходов на портах J1 и J2 (см SPEEDY-33 Board Layout). Цифровые пары (разъема J1) ввода-вывода Out0/Out1, Out2/Out3, Out4/Out5, Out6/Out7 используются для генерации скорости/ направления соответственно и отвечают одному из 4 выбранных R/C каналов. Соответствие цифровых входов с R/C каналами:

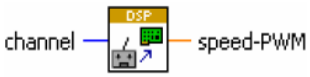
- In0 – RC1,
- In1 – RC2,
- In2 – RC3,
- In3 – RC4.

В связи с типичным применением первых двух элементов, задача второго сводится лишь к принятию информации в виде радиосигнала и передача его напрямую к первому элементу. Стоит отметить, что в среде RIDE можно прервать эту линию взаимодействия и поместить (не эту линию) заданную логическую схему для регулировки, посылаемой на двигатель или сервомотор информации о скорости и направлении вращения



Servo Drive Control.vi

Подобно многократному использованию Motor Drive Control, можно тоже самое проделать с RC Speed Control vi. Но всегда необходимо отслеживать, а не дублируют ли каналы друг друга. Использовать же можно 4 канала одновременно



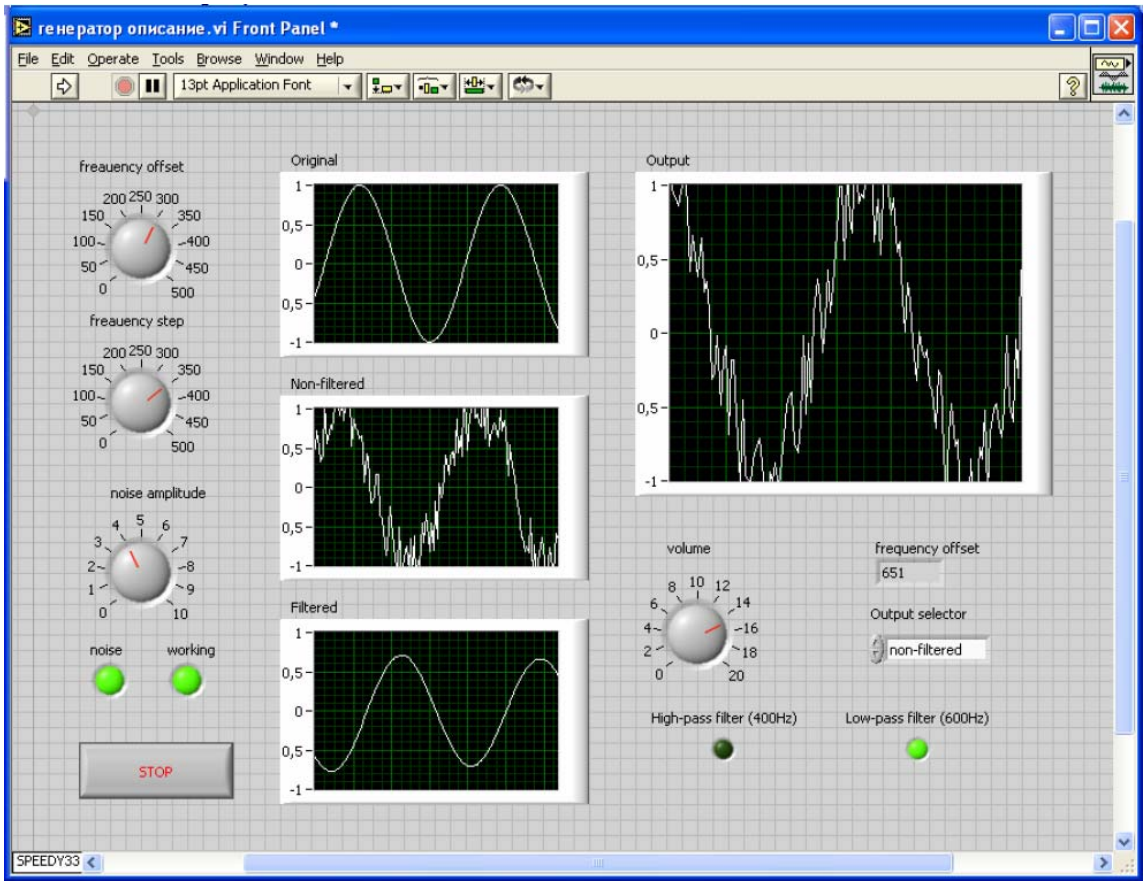
RC Speed Control.vi

Servo Drive Control обладает схожей характеристикой с Motor Drive Control. Но здесь уже осуществляется генерация информации о позиционировании сервопривода для управления положением его оси (установка в определенное угловое положение). Одновременно могут работать 4 независимых каналов, чьи выходы размещены на разъеме J2

Как видно, с помощью этого набора функций можно автоматизировать множество процессов на производстве: упаковка, фасовка, высокоточное перемещение разнообразных частей и деталей на сборочных линиях, перемещение рабочих роботов и манипуляторов, процессы, требующие быстрого и точного позиционирования, поддержание заданного момента, направления и скорости вращения.

Создание генератора сигналов

Создадим на лицевой панели набор элементов управления и отображения:



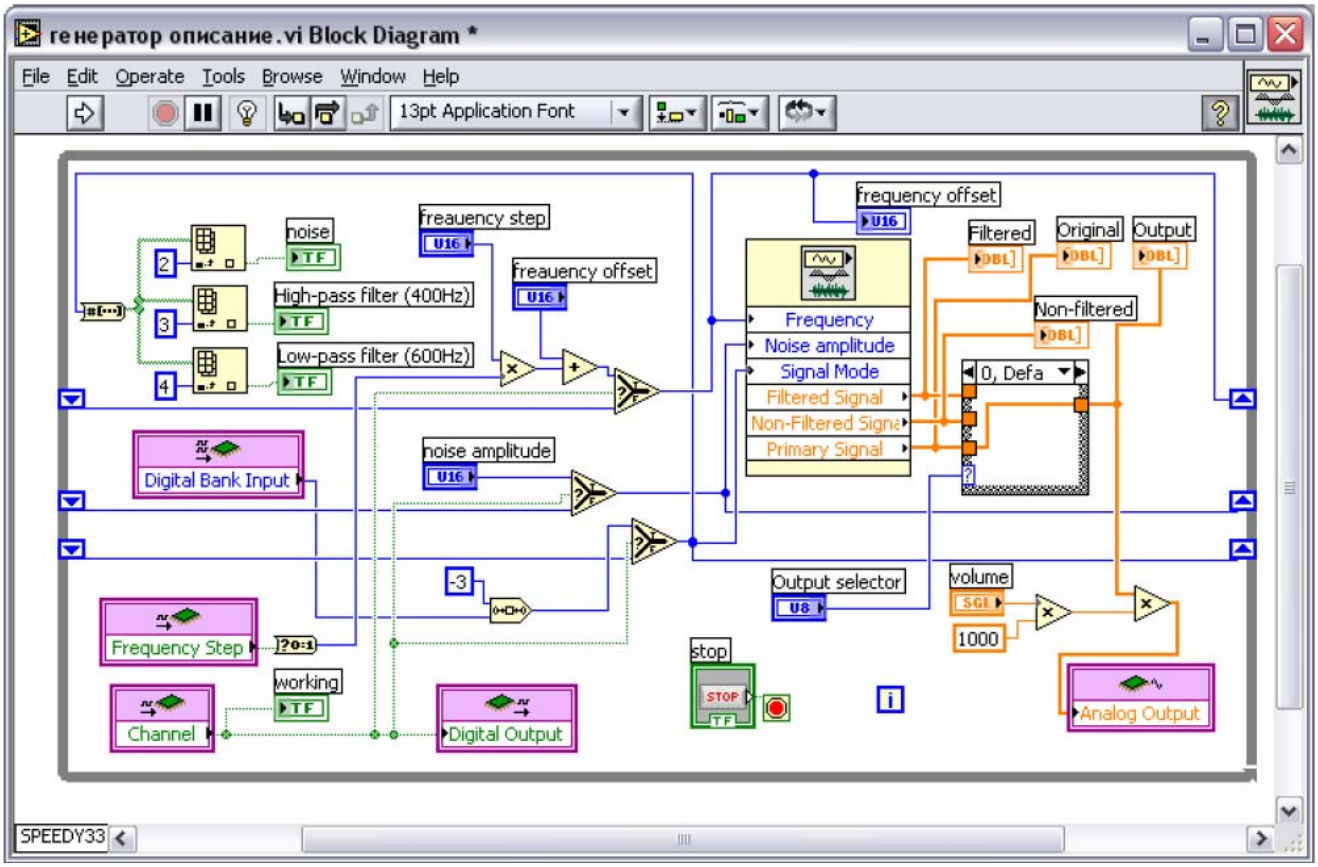
Перейдем на блок-диаграмму для работы с этими элементами, первоначально разместив на блок-диаграмме структуру While Loop. Вся схема работы генератора будет заключена в цикл, цикл, а вместе с ним и работа генератора будет останавливаться по нажатию кнопки Stop на лицевой панели (при отображении работы непосредственно на компьютере).

Необходимо сразу определить какие из DIP-переключателей за что отвечают. Пусть за начало работы генератора будет отвечать 1-й из переключателей, 3 – за сдвиг по частоте (frequency step), с помощью 4 и 5 можно будет задавать форму сигнала:

(4)	(5)	
0	0	Sine
1	0	Square
0	1	Sawtooth
1	1	Triangle

6 – отвечает за наличие шума в исходном сигнале; 7 – включает и выключает high-pass filter (400Hz); 8 – low-pass filter (600Hz).

Далее, необходимо соединить элементы следующим образом:



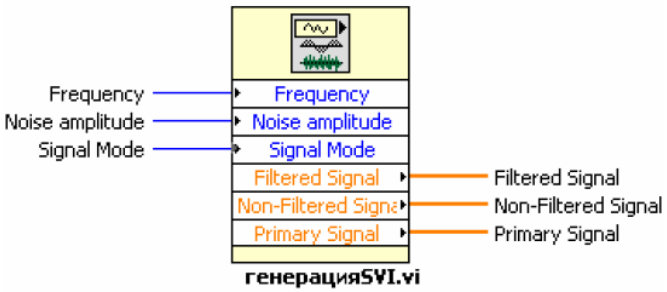
Рассмотрим подробнее назначение отдельных элементов. Для элемента I/O Digital Bank Input в конфигурации изменено Default DBI на On-board DIP switch. Что позволит с помощью 8 DIP-переключателей задать вышеописанные команды. После чего Digital Bank Input соединен с Logical Shift для того, чтобы при задании первых двух команд, было привычно работать с остальными функциями. Так как в последствии, при создании Case Structure, может возникнуть путаница с нумерацией. При использовании сдвига, путаницы не возникает.

При помощи нехитрой схемы, находящейся в левом верхнем углу блок-диаграммы, реализуется возможность отображения на лицевой панели работающего режима (noise, highpass filter, low-pass filter). Для этого на элементы Index Array подается сдвинутый на 3 единицы массив уже Boolean переменных с Digital Bank Input. Если тот или иной из переключателей приведен в нижнее положение (т.е. нажат), то индикатор, соответствующий то или иной команде, будет загораться на лицевой панели.

Для Analog Output в конфигурациях надо в опции Resource установить 1 channel multiple samples (цель этого действия приведена выше, в описании работы с портами ввода- вывода).

Присутствующие здесь три сдвиговых регистров нужны лишь для того, что при переключении первого DIP-переключателя в верхнее положение ранее заданные значения частоты, шума и прочих не обнулялись. Фактически, не будет реакции на изменение параметров либо на плате, либо на лицевой панели, зато продолжается работа с уже заданными значениями.

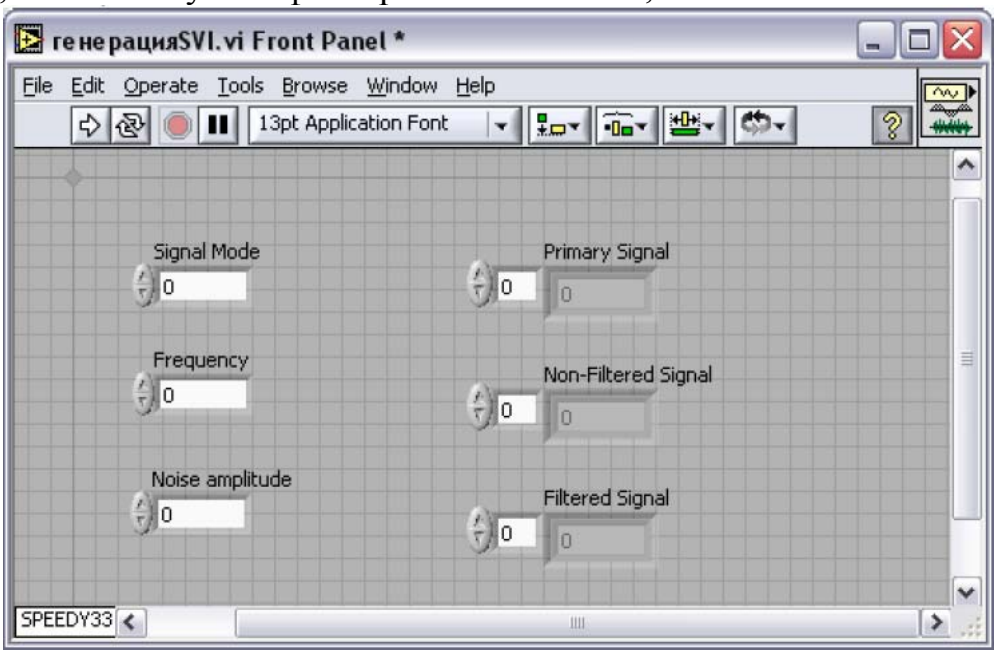
На график Output выводится сигнал в соответствии с установленным с Ring Properties режимом (чисто сгенерированный сигнал, сигнал + шум, отфильтрованный «сигнал + шум»). Это осуществляется при помощи при помощи все той же Case Structure, на вход которой подается значение с Ring Properties.

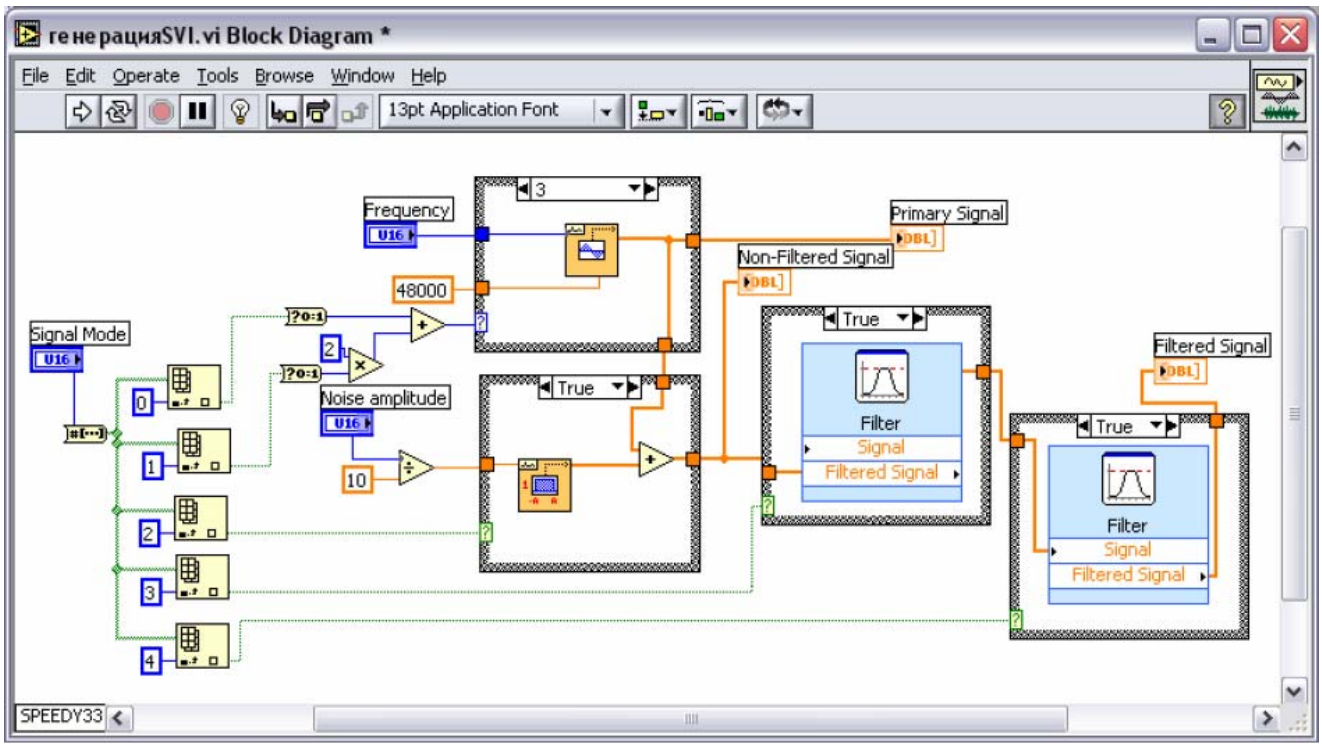


Наиболее важный элемент, реализующий генерацию. Это подпрограмма – определенный набор элементов объединенных в одно целое.

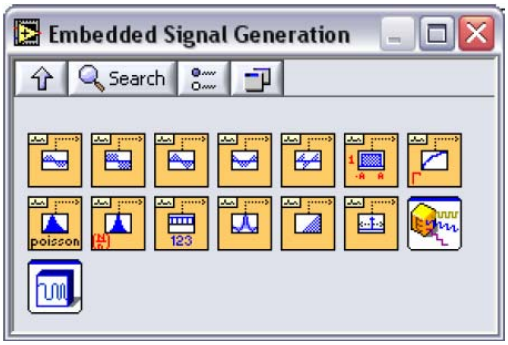
Лицевая панель данной подпрограммы предельно проста. Содержит всего лишь три элемента управления и три элемента отображения.

Здесь значение Signal Mode, которое определяется положение DIP-переключателей, определяет в двух Case Structures форму генерируемого сигнала, наличие шума и фильтра либо высокой, либо низкой частоты.

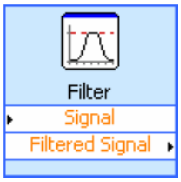




Для генерации сигналов (в данном случае 4 вида) используются Embedded Signal Generation.



Данная палитра содержит 15 элементов. Все они связны с непосредственной генерацией сигналов. Это Sine, Square, Cosine, Sawtooth, различные шумы: uniform white noise, Gamma noise, Poisson noise. Различные генераторы: импульсный генератор (Impulse Generator), генератор частотной развертки (Frequency Sweep Generator). И такой Express VI, как Simulate Signal, который сам по себе может генерировать и все вышеперечисленные сигналы с различными параметрами, и одновременно задавая параметры шума. У данных компонент либо по умолчанию заданы необходимые параметры, либо можно корректировать нужные из них.



В данной подпрограмме так же применяются Express VI, связанные с фильтрами. Например, в данном случае, используется обычный Filter, находящийся в Function T Signal Processing T Filters. При двойном клике мышью по данному Express VI появляется панель, полностью описывающая возможности данного фильтра. Именно на этой панели и устанавливается тип фильтра: высокочастотный, низкочастотный – параметры типа импульсных характеристик: конечные и бесконечные – определяется топология фильтра: ФНЧ Чебышева, фильтр Баттерфорда, фильтр Бесселя и другие.

После создания вышеуказанной блок-схемы необходимо сохранить ее как подпрограмму основной программы.

Выполнение работы

Задание №1

Создать и настроить визуальную программу для DSP NI SPEEDY-33, выполняющую фильтрацию для аудио-сигнала подаваемого в виде суммы тестового сигнала и шума (в виде шума также может выступать эхо-сигнал). При этом тестовый сигнал должен иметь возможность заменяться входным сигналом с внешнего источника или встроенного микрофона.

Решение задачи:

Выполнить суммирование (Add) аудиосигнала (Analog input) и его задержанной копии (задержка регулируется Numeric constant), произвести накопление сигнала (Convert to Dynamic DATA) с последующей фильтрацией (Filter) от эхо-сигнала. На всех этапах необходимо осуществлять контроль временных отсчетов сигнала (Waveform Graph 1, 2, 3, 4). Включение отключение фильтрации осуществляется при помощи логического переключателя (Vertical Toggle Switch использующего логическую структуру – Case structure).

Используемые компоненты

- Analog input – аудио-вход;
- Numeric constant – константа, определяющая временную задержку сигнала;
- Add – суммирование входных сигналов;
- Convert to Dynamic DATA – накопление цифрового сигнала;
- Filter – фильтр сигнала;

- Case structure – логическое управление системой;
- Vertical Toggle Switch – переключатель между генератором синусоиды и внешним аудио-сигналом;

- Waveform Graph 1 – выходной график сигнала;
- Waveform Graph 2 – график входного исходного сигнала;
- Waveform Graph 3 – график входного сигнала выступающего в виде помехи;
- Waveform Graph 4 – график суммированного входного сигнала выступающего в виде помех и исходного сигнала.

Задание №2

Создать и настроить визуальную программу фильтрацию и спектрального анализа в режиме реального времени для аудио-сигналов в DSP NI SPEEDY-33.

Решение задачи:

Получить спектр (Amplitude and Phase Spectrum 1) и вывести спектр на экран (Waveform Chart 2) исходного сигнала (Analog input). После,

сигнал, пришедший с внешнего источника (Analog input), отфильтровать при помощи фильтра (Eliptic Filter) с параметрами (Numeric Control 1 – «Type» тип элептического фильтра, Numeric Control 2 – «Fd» вход дискретизированного сигнала, Numeric Control 3 – «Low» нижняя обрабатываемая частота, Numeric Control 4 – «High» верхняя обрабатываемая частота). Необходимо иметь возможность включения/выключения фильтрации при помощи логического переключателя (Vertical Toggle Switch, использующего логическую структуру – Case structure), получить в этой структуре отключаемую функцию спектра (Amplitude and Phase Spectrum 2) отфильтрованного сигнала и вывести спектр отфильтрованного сигнала на экран (Waveform Chart 1).

Используемые компоненты

- Sine wave – источник синусоидального сигнала;
- Analog input – источник внешнего сигнала;

- Numeric Control 1 – «Type» тип элептического фильтра;
- Numeric Control 2 – «Fd» вход дискретизированного сигнала;
- Numeric Control 3 – «Low» нижняя обрабатываемая частота;
- Numeric Control 4 – «High» верхняя обрабатываемая частота;
- Numeric Control 4 – «dB» конечное усиление;

- Amplitude and Phase Spectrum 1 – вычисление спектра для входного сигнала;
- Amplitude and Phase Spectrum 2 – вычисление спектра для отфильтрованного сигнала;
- Case structure – логическое управление системой;
- Vertical Toggle Switch – переключатель между генератором синусоиды и внешним аудио-сигналом;
- Eliptic Filter – фильтр;

- Waveform Chart 1 – «Filtered Signal's Spectrum» спектр для входного отфильтрованного сигнала;
- Waveform Chart 2 – «Signal's Spectrum» спектр для входного сигнала;

Задание №3

Реализовать на базе DSP NI SPEEDY-33 функцию скрытия аудиосигнала при использовании маскирующего широкополосного сигнала в диапазоне частот всей группы полезных сигналов.

Решение задачи:

Сформировать широкополосный сигнал ПСП в структуре (Case structure 1), имеющей возможность переключения (Vertical Toggle

Switch 2) для использования внутреннего ПСП генератора (Simulate signal). Параметры самостоятельно полученного ПСП сигнала, реализованного на линейном конгруэнтном генераторе (Floor, Divide 1, Multiply, Add, Numeric constant 1, Numeric constant 2, Numeric constant 3, Numeric constant 4). В структуре (Case structure 1) получить возможность усиления (Slide, Divide 2, Multiply 2) входного сигнала (Analog input), отображаемого на экране 2 (Waveform Chart 2) с широкополосным ПСП сигналом. Вывод результата необходимо осуществить при помощи Analog output и на экран 1 (Waveform Chart 1).

Используемые компоненты

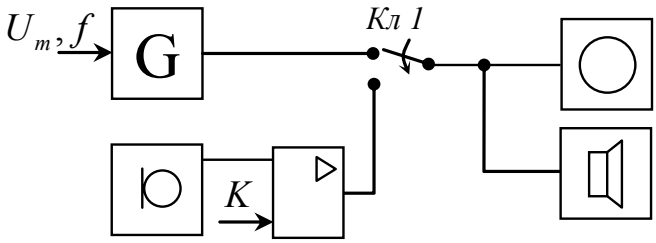
- Numeric constant 1, 2, 3, 4 – константы для линейного конгруэнтном генератора широкополосного ПСП сигнала;
- Analog input – источник внешнего сигнала;
- Analog output – выход обработанного аудио сигнала;

- Slide – регулятор усиления аудио сигнала;
- Multiply 1 – умножение входящее в формулу линейного конгруэнтного генератора широкополосного ПСП сигнала;
- Divide 1 –деление входящее в формулу линейного конгруэнтного генератора широкополосного ПСП сигнала;
- Add – суммирование входящее в формулу линейного конгруэнтного генератора широкополосного ПСП сигнала;
- Divide 2 – деление сигнала реализующую функцию усиления;
- Multiply 2 – умножение сигнала реализующую функцию усиления;
- Case structure 1 – структура в которой происходит создание у правление линейным конгруэнтным генератором широкополосного ПСП сигнала;
- Case structure 2 – структура в которой происходит переключение между линейным конгруэнтным генератором широкополосного ПСП сигнала и внутренним ПСП генератором;
- Vertical Toggle Switch 1 – переключатель между линейным конгруэнтным генератором широкополосного ПСП сигнала и внутренним ПСП генератором;
- Vertical Toggle Switch 2 – переключатель линейный конгруэнтный генератор широкополосного ПСП сигнала;

- Waveform Chart 1 – экран отображающий выходной сигнал после преобразований;
- Waveform Chart 2 – экран отображающий линейный конгруэнтный генератор широкополосного ПСП сигнала.

Пример создания виртуального измерительного прибора на платформе DSP NI SPEEDY-33

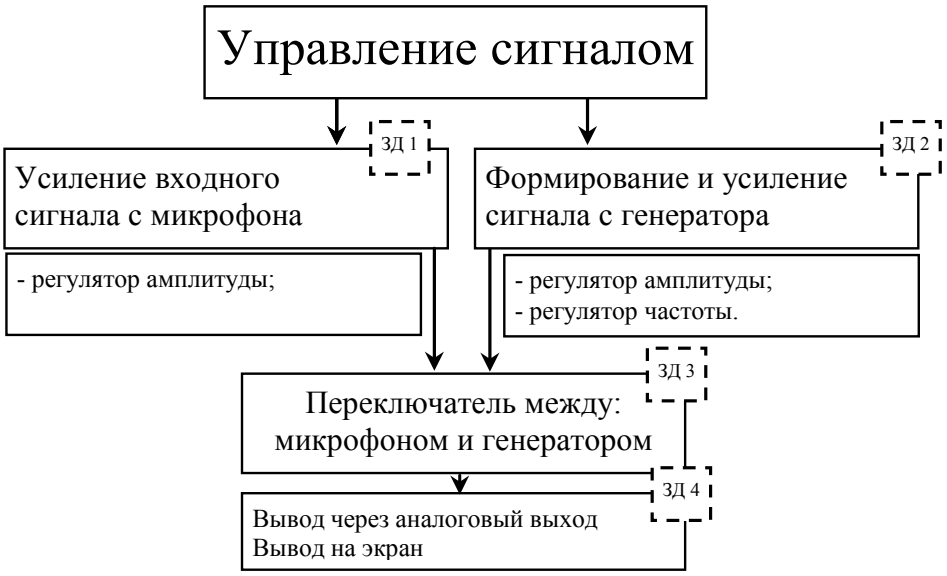
Дана схема визуального программирования на лабораторном стенде:



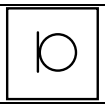
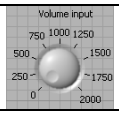
Кл 1 – переключатель между источниками сигнала: генератором и встроенным микрофоном;
K – коэффициент усиления микрофона;
U_m, f – амплитуда и частота гармонического сигнала;

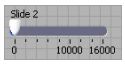
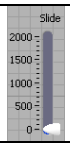
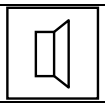
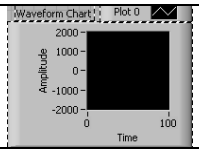
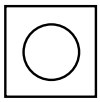
Требуется решить типовую задачу ввода-обработки-вывода звукового сигнала с возможностью выбора источника сигнала. Задача может быть разложена на 5 подзадач:

- ✓ усиление входного сигнала с микрофона;
- ✓ формирование и усиление сигнала с генератора;
- ✓ выбор источника сигнала;
- ✓ вывод сигнала на аналоговый выход стенда и на виртуальный экран осциллографа.



В решении данной задачи будут задействованы следующие интерактивные элементы среды визуального программирования LabVIEW.

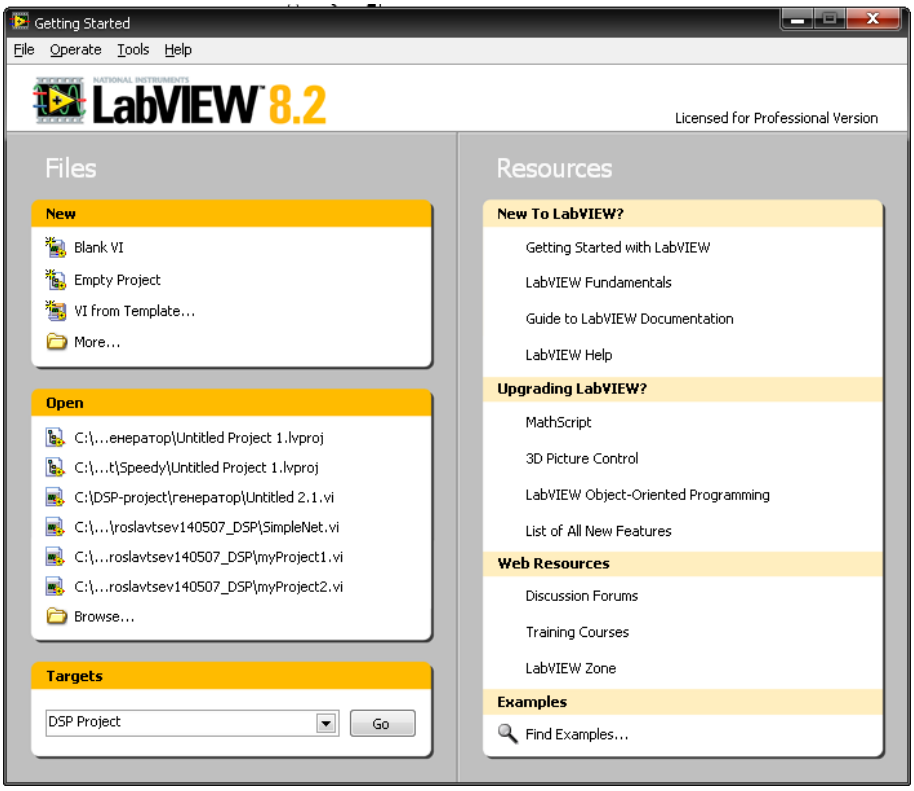
Наименование	**Описание**	**Block Diagram**	**Front Panel**	**Относиться к элементу схемы**
Analog Input (AI)	Аналоговый вход (микрофон)		-	
Knob	Регулятор громкости			

Vertical Toggle Switch	Амплитуда сигнала			G
Horizontal Toggle Switch	Частота сигнала			
Vertical Toggle Switch	Переключатель между внешним входом и генератором			*Кл 1*
Analog Output (AO)	Аналоговый выход		-	
Waveform Chart	Отображение входного сигнала			

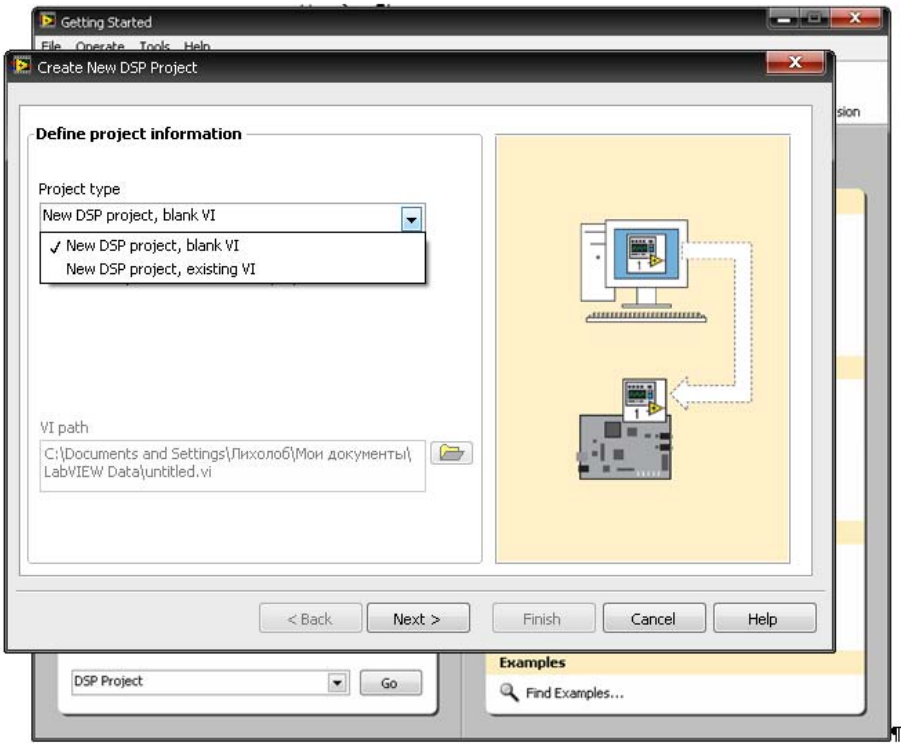
Пошаговое решение задачи графического программирования на базе DSP NI SPEEDY-33 в LabVIEW 8.2 выглядит следующим образом:

1. Выполним запуск программы LabVIEW 8.2 , часто называемой виртуальными инструментами (ВИ).

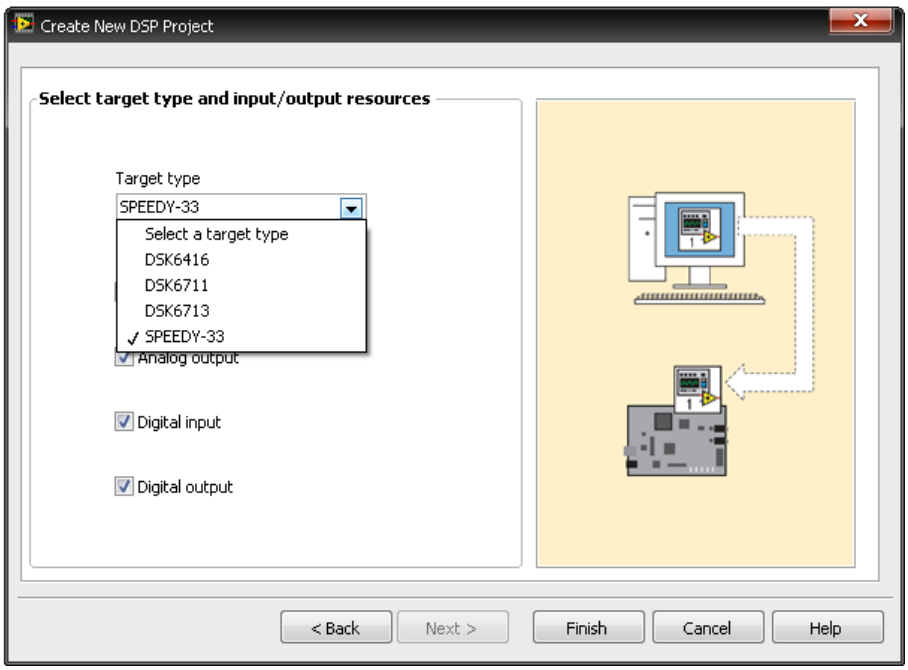
2. В стартовом окне во вкладке «Targets» → выбрать «DSP Project» → нажать «Go»:



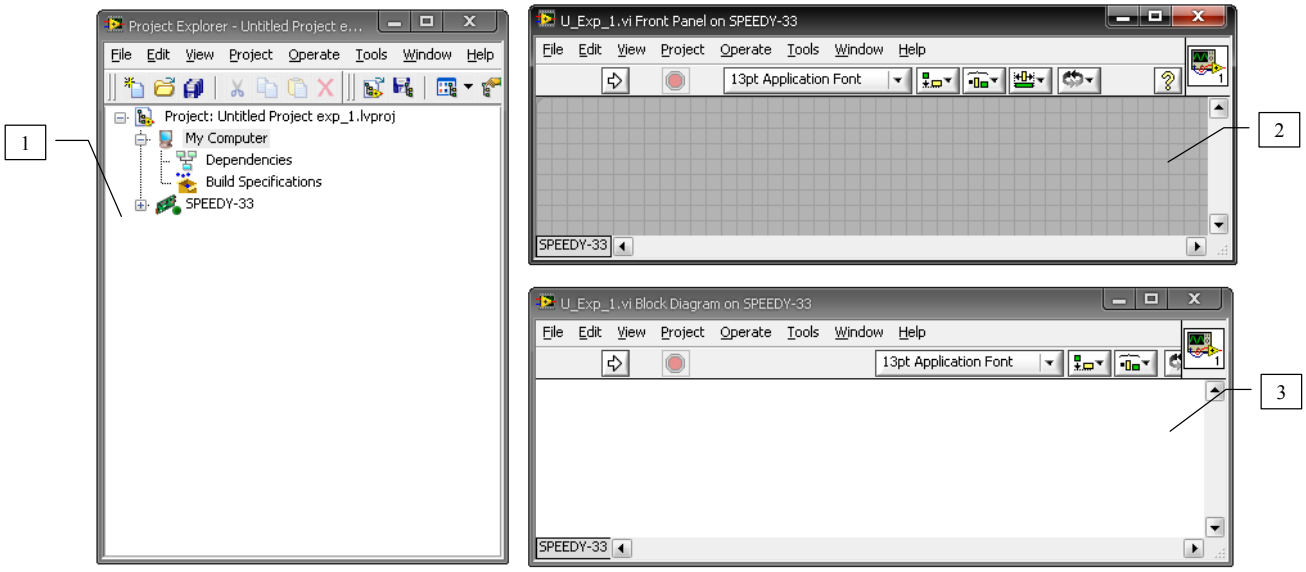
3. Выбрать новый проект New DSP project, blank VI → нажать «Next»:



4. Выбрать тип программирования процессора SPEEDY-33 и выбрать входы и выходы, которые будут использоваться → нажать «Next»:

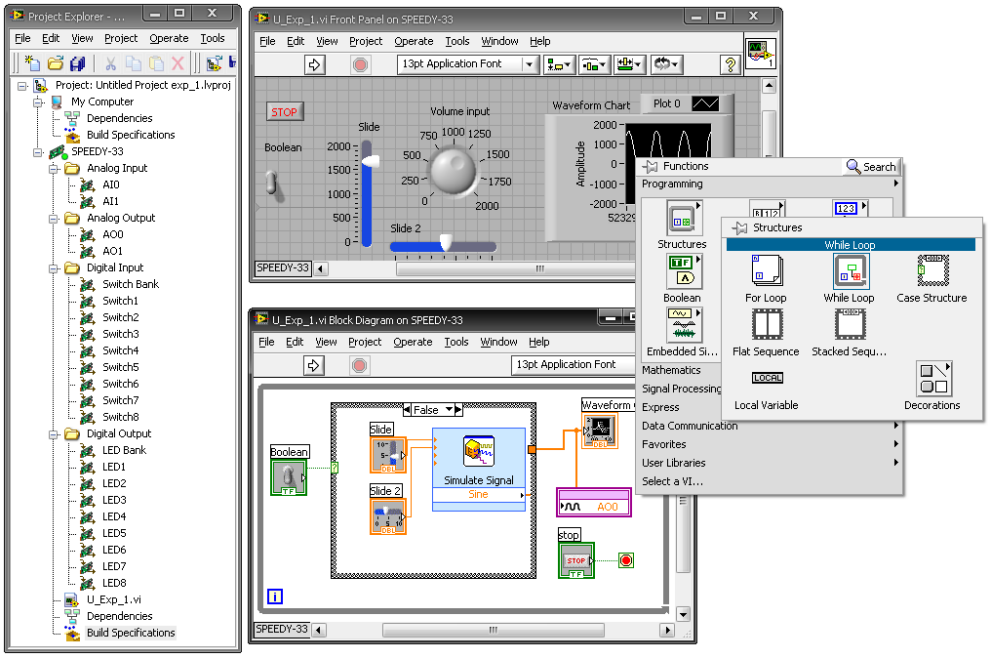


5. Проект состоит из трех рабочих областей: панель проекта; панель визуальных приборов; панель визуального программирования:

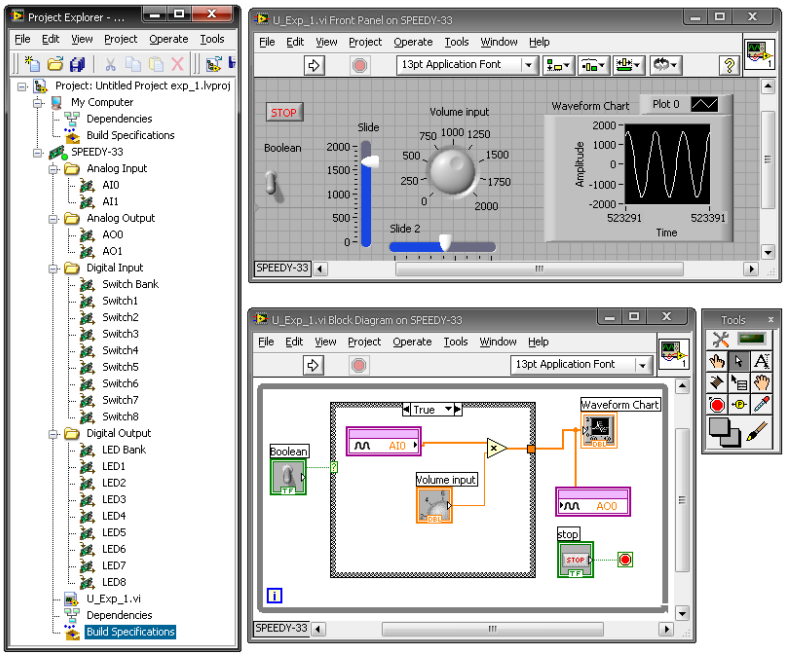


Проект решения задачи в среде LabVIEW будет выглядеть следующим образом:

- при выборе генератора сигналов:



- при выборе микрофона:



Контрольные вопросы к защите

1. Какие принципы построения кристалла заложены в цифровые сигнальные процессоры (DSP) и укажите их отличие от универсальных процессоров?
2. В чем заключается опция SPEEDY-33?
3. Какие функции возложены на LabVIEW а какие на DSP, при обработке сигналов?
4. Опишите реализацию функции программно-аппаратного эхо-подавления, применяемой в Ip-телефонии и реализованной на DSP?
5. Опишите реализацию функции скрытия аудиосигнала при использовании маскирующего широкополосного сигнала в диапазоне частот всей группы полезных сигналов.

Способ оценки результатов

По лабораторной работе предусмотрена защита. Лабораторная работа будет зачтена, если преподавателю будут продемонстрирована программа, отрабатывающая полученное задание, а также, если студент ответит на заданные преподавателем вопросы. Папка с созданным проектом передается преподавателю.

Лабораторная работа №2. Управление дискретными процессами на базе ПЛК Siemens LOGO!

Требования к содержанию, оформлению и порядку выполнения

Перед выполнением лабораторной работы студент должен изучить теоретическую часть курса. Студент, выполняющий лабораторную работу в составе с реальным лабораторным стендом, обязан получить к ней допуск путем собеседования с преподавателем, который заключается в

подтверждении готовности студента к ее выполнению, а именно, знание основ теории по данной теме, порядка выполнения лабораторной работы, особенностей и правил работы с лабораторным стендом. Преподаватель определяет вариант задания для выполнения лабораторной работы конкретным студентом.

Студент при работе с персональным компьютером, на котором установлено прикладное и коммуникационное программное обеспечения для работы с данным лабораторным стендом, должен создать новую папку под именем «Ваша фамилия, Лабораторная работа №_» (например: «Симонов А.М. Лабораторная работа №2»).

Выполнение лабораторной работы производится в 2 этапа:

1. Разработка программы и отладка ее на программном эмуляторе того продукта, который предназначен для данного стенда.
2. Демонстрация преподавателю работы отлаженной программы, которая должна функционировать в составе с лабораторным стендом в строгом соответствии с полученным заданием (только для студентов очной формы обучения).
3. Сохранение созданного проекта, содержащего разработанную и отлаженную программу, в вышеуказанную папку.

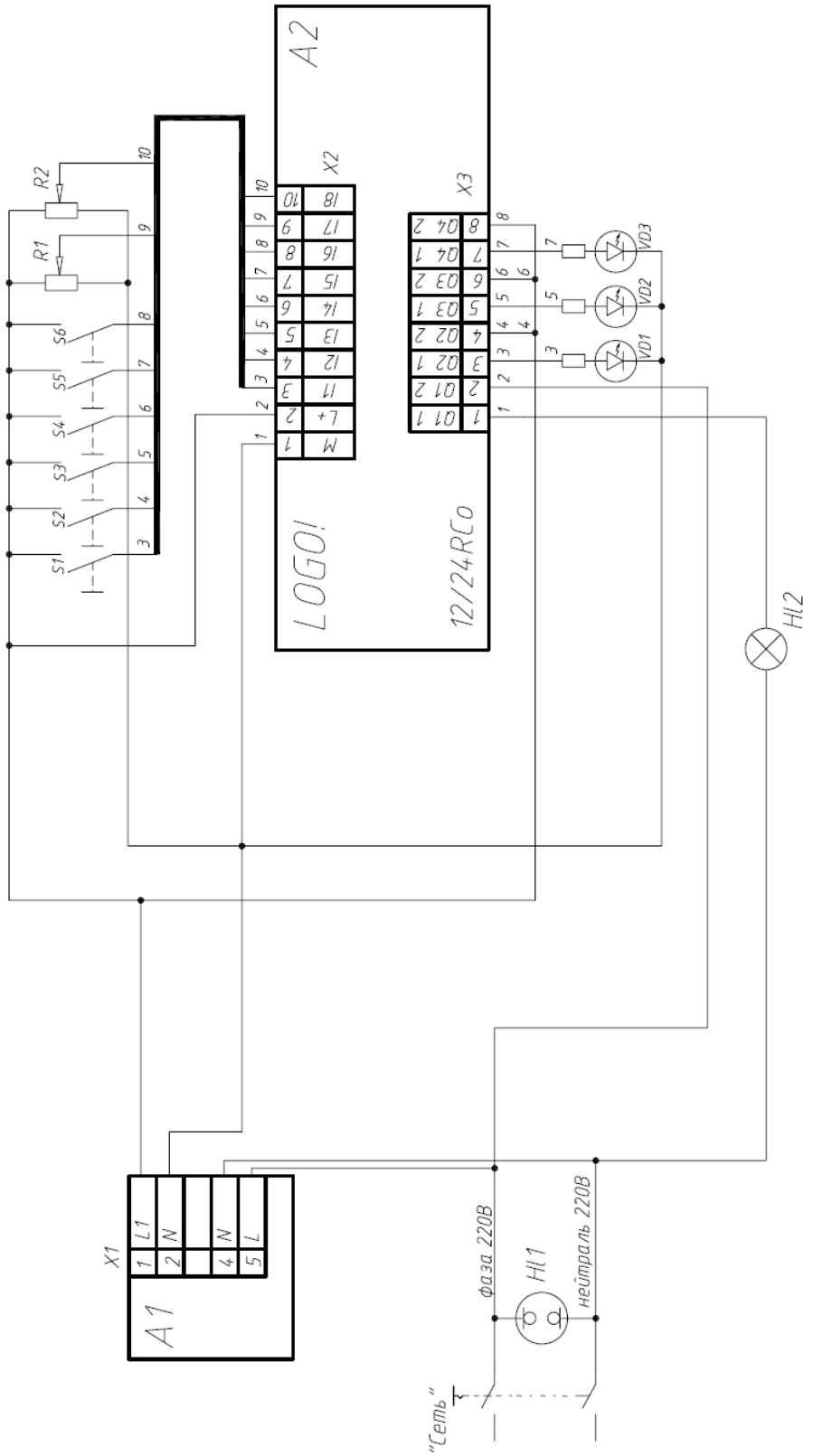
После успешного выполнения лабораторной работы студент должен оформить отчет в соответствии с предъявляемыми требованиями и осуществить ее защиту, проводимую в форме собеседования с преподавателем. Цель защиты – подтверждение факта самостоятельного выполнения лабораторной работы и определение степени усвоения студентом теоретического материала по данной теме.

Описание лабораторного стенда

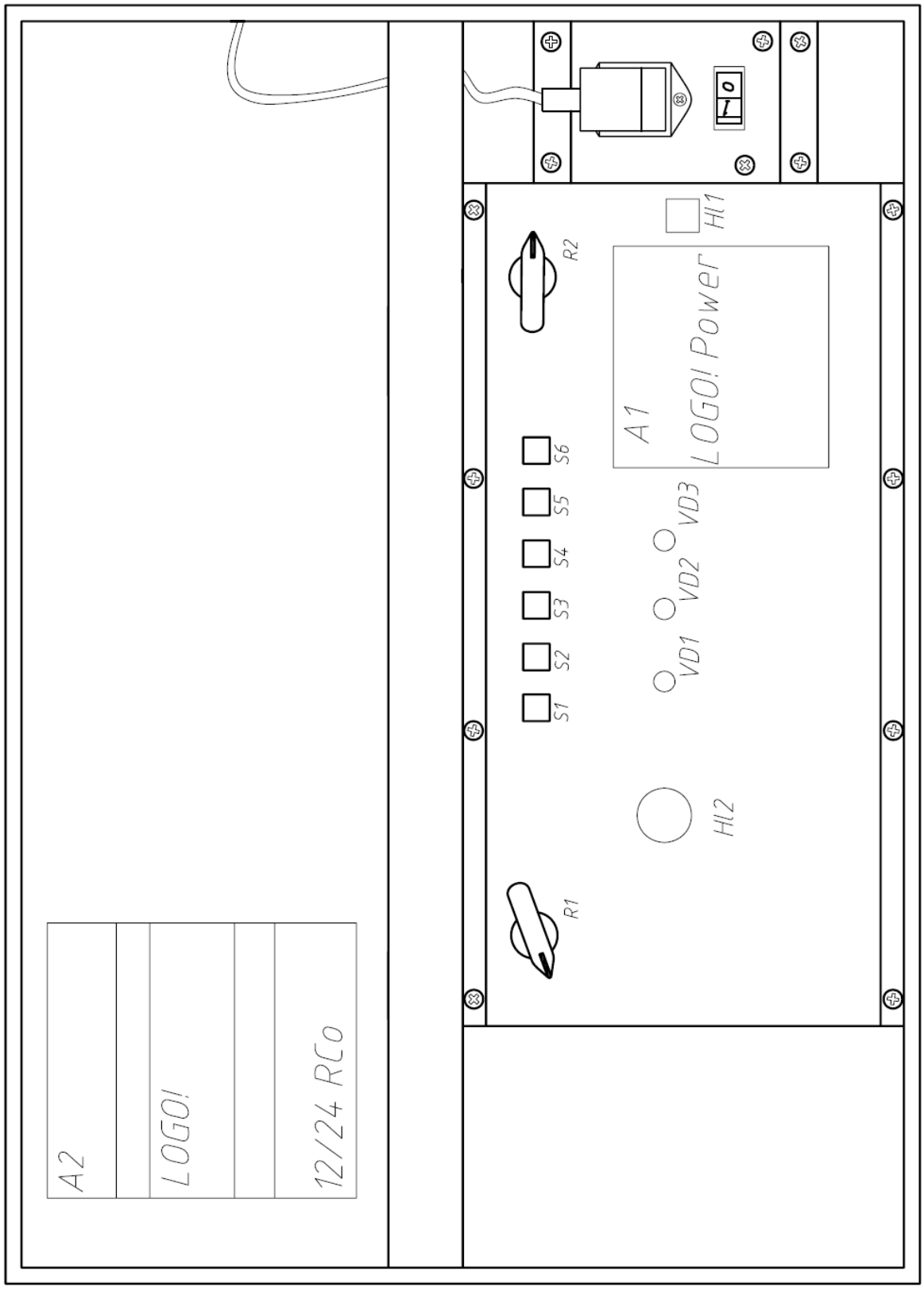
Лабораторный стенд выполнен в переносном корпусе чемоданного типа, он состоит из верхней откидной панели, и нижнего основания. В нижней части смонтированы: блок питания, промышленный контроллер, устройства коммутации, органы управления и индикации. Лабораторный стенд на базе ПЛК LOGO! исполнен в двух вариантах: модель 1 и модель 2. Выбор конкретной модели производится преподавателем после получения допуска студентом. Ниже приведены электрические принципиальные схемы и схемы монтажные для двух вариантов исполнения лабораторного стенда. На схемах введены обозначения:

- A1 – блок питания;
- A2 – ПЛК Siemens LOGO!;
- VD1-VD3 – светодиод (средства индикации);
- HL1-HL2 – лампа (средства индикации);
- S1-S6 – переключатели (органы управления);
- R1-R2 – потенциометр (органы управления);
- X1-X3 – клеммные соединители и разъемы – средства коммутации.

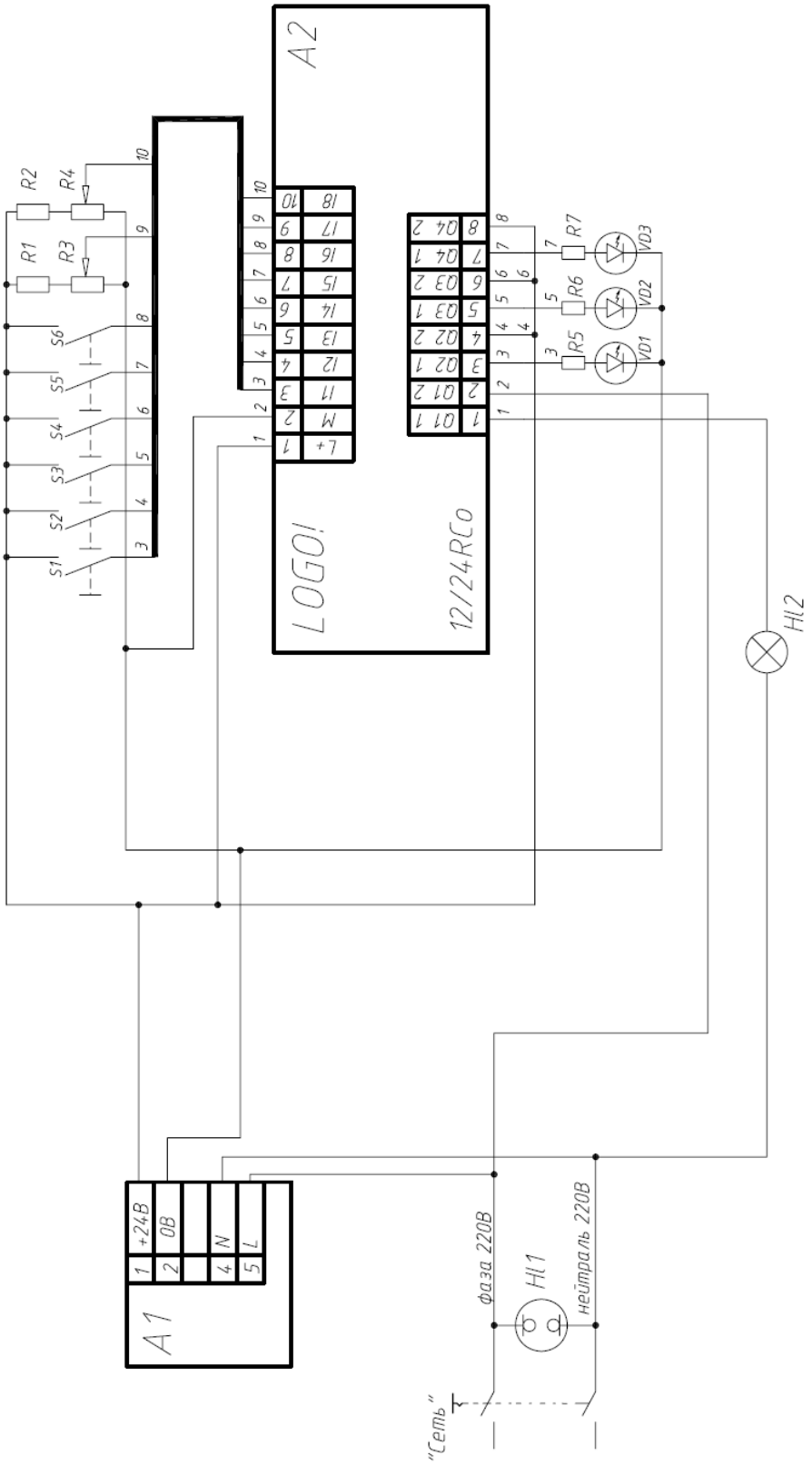
Подключение к сети питания 220В, 50 Гц и к персональному компьютеру осуществляется специальными кабелями под контролем преподавателя.



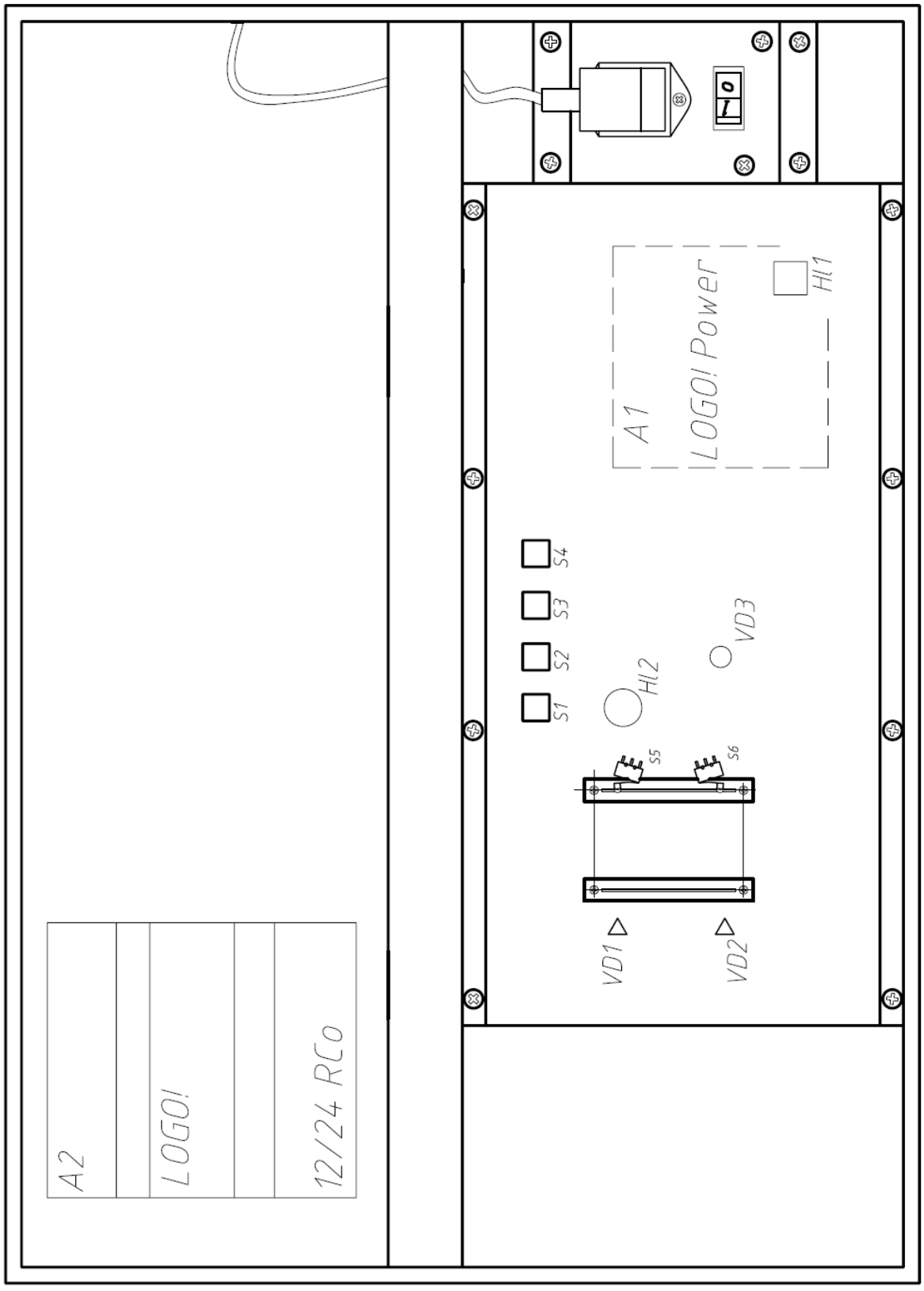
Изм.	Лист	N° докум.	Подп.	Дата	Мобильный лабораторный стенд для изучения контроллера LOGO!. Модель 1. Схема электрическая принципиальная.	Лист



Изм.	Лист	№ докум.	Подп.	Дата	Мобильный лабораторный стенд для изучения контроллера LOGO!. Модель 1 Нижняя панель. Схема монтажная.	Лист



Изм.	Лист	N° докум.	Подп.	Дата	Мобильный лабораторный стенд для изучения контроллера LOGO!. Модель 2. Схема электрическая принципиальная.	Лист



Изм.	Лист	№ докум.	Подп.	Дата	Мобильный лабораторный стенд для изучения контроллера LOGO!. Модель 2 Нижняя панель. Схема монтажная.	Лист

Общая постановка задачи

В ходе выполнения лабораторной работы студент должен получить навыки работы с контроллерами фирмы Siemens модификации LOGO!, а также с его программным продуктом Logo! SoftComfort v.2.5. В процессе выполнения лабораторной работы студент должен правильно подобрать необходимую для решения поставленной задачи конфигурацию промышленного контроллера, определить величины входных сигналов и их типы, создать исполнительную программу на базе стандартных функциональных блоков, входящих в программное обеспечение Logo! SoftComfort v.2.5.

Выполнение работы

Задание №1 «Управление жалюзи на базе ПЛК SIEMENS LOGO!»

С помощью ПЛК LOGO! необходимо управлять жалюзи жилого дома. С помощью селекторного переключателя может быть выбран ручной режим или автоматическое управление. В зависимости от времени, темноты и дня недели жалюзи автоматически закрываются или открываются.

Ручной режим:

С помощью выключателей на DI2 (ОТКРЫТЬ) и DI3 (ЗАКРЫТЬ) жалюзи можно открывать и закрывать вручную. Ручной режим задается переключателем DI4.

Автоматический режим:

Для перехода в автоматический режим селекторный переключатель (DI4) должен находиться в положении «Автоматика». Когда срабатывает сумеречное реле на DI1, жалюзи закрываются на период с 18:00 до 7:00. Открываются они в течение дня между 7:00 и 18:00. В отчете и показываемой на эмуляторе программе необходимо пользоваться именно указанными таймерами.

Эмуляция времени на стенде выполняется с помощью изменения уровня аналогового сигнала на AI1. Через конечные выключатели на DI5 и DI6 производится опрос, открыты жалюзи или закрыты.

Используемые компоненты:

- LOGO!
- DI1 Сумеречное реле (замыкающий контакт)
- DI2 Выключатель для открытия в ручном режиме (замыкающий контакт)
- DI3 Выключатель для закрытия в ручном режиме (замыкающий контакт)
- DI4 Выключатель «Автоматика» (замыкающий контакт)
- DI5 Конечный выключатель «Жалюзи открыты» (замыкающий контакт)
- DI6 Конечный выключатель «Жалюзи закрыты» (замыкающий контакт)
- Q1 Автоматика включена (DO1)
- Q2 Открыть жалюзи (DO2)
- Q3 Закрыть жалюзи (DO3)

Задание№2 «Управление поливом растений в оранжерее на базе ПЛК LOGO!»

В оранжерее необходимо управлять орошением растений с помощью ПЛК LOGO! При этом различают три типа растений. В случае растений типа 1 речь идет о водорослях в бассейне, уровень воды в котором должен поддерживаться в определенном диапазоне.

Растения второго типа должны орошаться утром и вечером по 3 минуты, а растения третьего типа – каждый второй вечер в течение 2 минут.

Должно быть предусмотрено выключение режима автоматического орошения для растений 1-го типа.

Решение с помощью LOGO!

Орошение растений - Тип 1:

Через поплавковые выключатели для максимального и минимального значения (на I1 и I2) уровень воды в бассейне всегда поддерживается в этом заданном диапазоне. При превышении верхнего уровня происходит отключение работы подающего насоса, при принижении нижнего уровня снова включается.

Орошение растений - Тип 2:

Орошение включается через реле времени утром с 6:00 до 6:03 и вечером с 20:00 до 20:03 каждый раз на 3 минуты (ежедневно).

Орошение растений - Тип 3:

С помощью созданного студентом генератора импульсов растения орошаются только каждый второй день вечером в течение 2 минут, когда срабатывает сумеречный выключатель на I3.

Используемые компоненты

- LOGO!
- I1 Поплавковый выключатель для максимального значения (размыкающий контакт)
- I2 Поплавковый выключатель для минимального значения (замыкающий контакт)
- I3 Сумеречный выключатель (замыкающий контакт)
- I4 Выключатель для автоматически управляемого орошения (замыкающий контакт)
- I5 Ручное управление включением подачи воды (замыкающий контакт)
- I6 Ручное управление выключением подачи воды (замыкающий контакт)
- Q1 Электромагнитный клапан для орошения растений типа 1
- Q2 Электромагнитный клапан для орошения растений типа 2
- Q3 Электромагнитный клапан для орошения растений типа 3

Задание№3 «Освещение витрины на базе ПЛК LOGO!»

С помощью ПЛК LOGO! должны автоматически освещаться товары, выставленные в витрине. При этом различают 4 группы источников света. Первая для освещения днем, вторая для дополнительного освещения вечером

и третья для минимального освещения ночью и четвертая для создания световых пятен, которые должны отдельно освещать размещенные предметы.

Решение с помощью LOGO!

Витрина должна освещаться с понедельника по пятницу с 8:00 до 22:00, в субботу с 8:00 до 24:00 и в воскресенье с 12:00 до 20:00. В течение этих интервалов через реле времени включается группа источников света 1 на Q1.

Кроме того, по вечерам подключается группа источников света 2, когда срабатывает сумеречное реле на входе I1. Вне вышеназванных интервалов времени группа источников света 3 на Q3 берет на себя минимальное освещение после деблокировки сумеречного реле.

Через сигнализатор перемещений на I4 в течение всего времени включаются или выключаются световые пятна (группа источников света 4 на Q4).

С помощью тестовой кнопки на I3 можно на 1 минуту включить все группы источников света, чтобы, например, проверить правильность их функционирования.

Используемые компоненты

- LOGO! 230RC
- I1 Сумеречное реле (замыкающий контакт)
- I2 Выключатель ВКЛ (замыкающий контакт)
- I3 Тестовый выключатель (замыкающий контакт)
- I4 Сигнализатор перемещений (замыкающий контакт)
- Q1 Группа источников света 1
- Q2 Группа источников света 2
- Q3 Группа источников света 3(минимальное освещение)
- Q4 Группа источников света 4(световые пятна)

Контрольные вопросы к защите

1. Какие сигналы подаются на аналоговый вход контроллера?
2. В чем отличие классических и PC-совместимых контроллеров?
3. Какие блоки необходимо использовать для отделение одного импульса от длительного сигнала высокого и низкого уровня?
4. Какие типы контроллеров LOGO! вы знаете и в чем их отличие?
5. Какой функциональный блок отвечает за задание таймеров определенного типа?

Способ оценки результатов

По лабораторной работе предусмотрена защита. Лабораторная работа будет зачтена, если преподавателю будут продемонстрирована программа, отрабатывающая полученное задание, а также, если студент ответит на заданные преподавателем вопросы. Папка с созданным проектом передается преподавателю.

Лабораторная работа №3. Программирование ПЛК серии VIPA CPU216

Требования к содержанию, оформлению и порядку выполнения

Перед выполнением лабораторной работы студент должен изучить теоретическую часть курса. Студент, выполняющий лабораторную работу в составе с реальным лабораторным стендом, обязан получить к ней допуск путем собеседования с преподавателем, который заключается в подтверждении готовности студента к ее выполнению, а именно, знание основ теории по данной теме, порядка выполнения лабораторной работы, особенностей и правил работы с лабораторным стендом. Преподаватель определяет вариант задания для выполнения лабораторной работы конкретным студентом.

Студент при работе с персональным компьютером, на котором установлено прикладное и коммуникационное программное обеспечения для работы с данным лабораторным стендом, должен создать новую папку под именем «Ваша фамилия, Лабораторная работа №_» (например: «Симонов А.М. Лабораторная работа №3»).

Выполнение лабораторной работы производится в 2 этапа:

1. Разработка программы и отладка ее на программном эмуляторе того продукта, который предназначен для данного стенда.
2. Демонстрация преподавателю работы отлаженной программы, которая должна функционировать в составе с реальным лабораторным стендом в строгом соответствии с полученным заданием (только для студентов очной формы обучения).
3. Сохранение созданного проекта, содержащего разработанную и отлаженную программу, в вышеуказанную папку.

После успешного выполнения лабораторной работы студент должен оформить отчет в соответствии с предъявляемыми требованиями и осуществить ее защиту, проводимую в форме собеседования с преподавателем. Цель защиты – подтверждение факта самостоятельного выполнения лабораторной работы и определение степени усвоения студентом теоретического материала по данной теме.

Описание лабораторного стенда

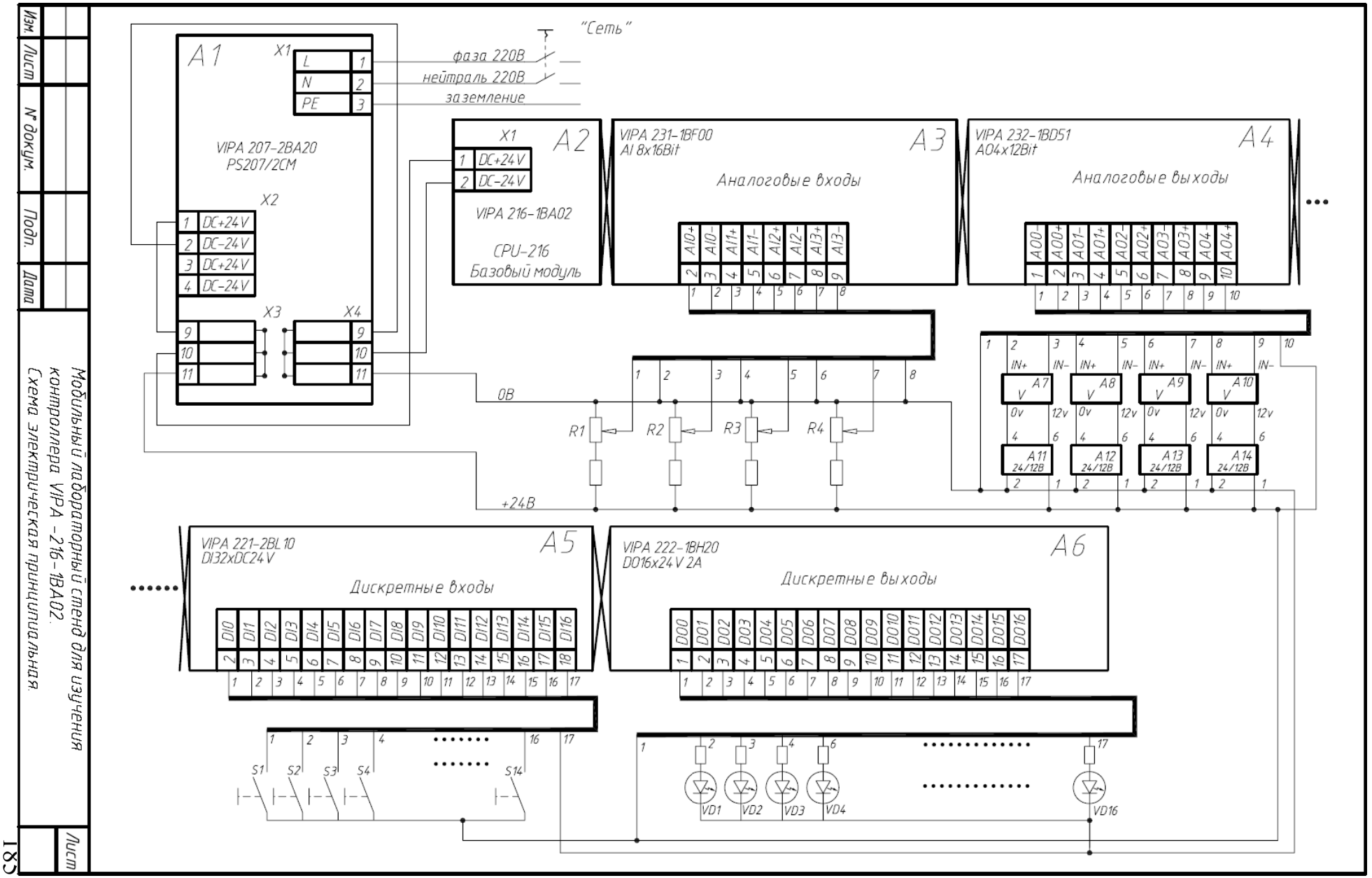
Лабораторный стенд выполнен в переносном корпусе чемоданного типа, он состоит из верхней откидной панели, и нижнего основания. В нижней части смонтированы: блок питания, промышленный контроллер, модули ввода-вывода, устройства коммутации, органы управления и индикации. В верхней части стенда смонтированы органы индикации и цифровые измерительные модули аналоговых сигналов. Ниже приведены электрические принципиальные схемы и схемы монтажные исполнения лабораторного стенда. На схемах введены обозначения:

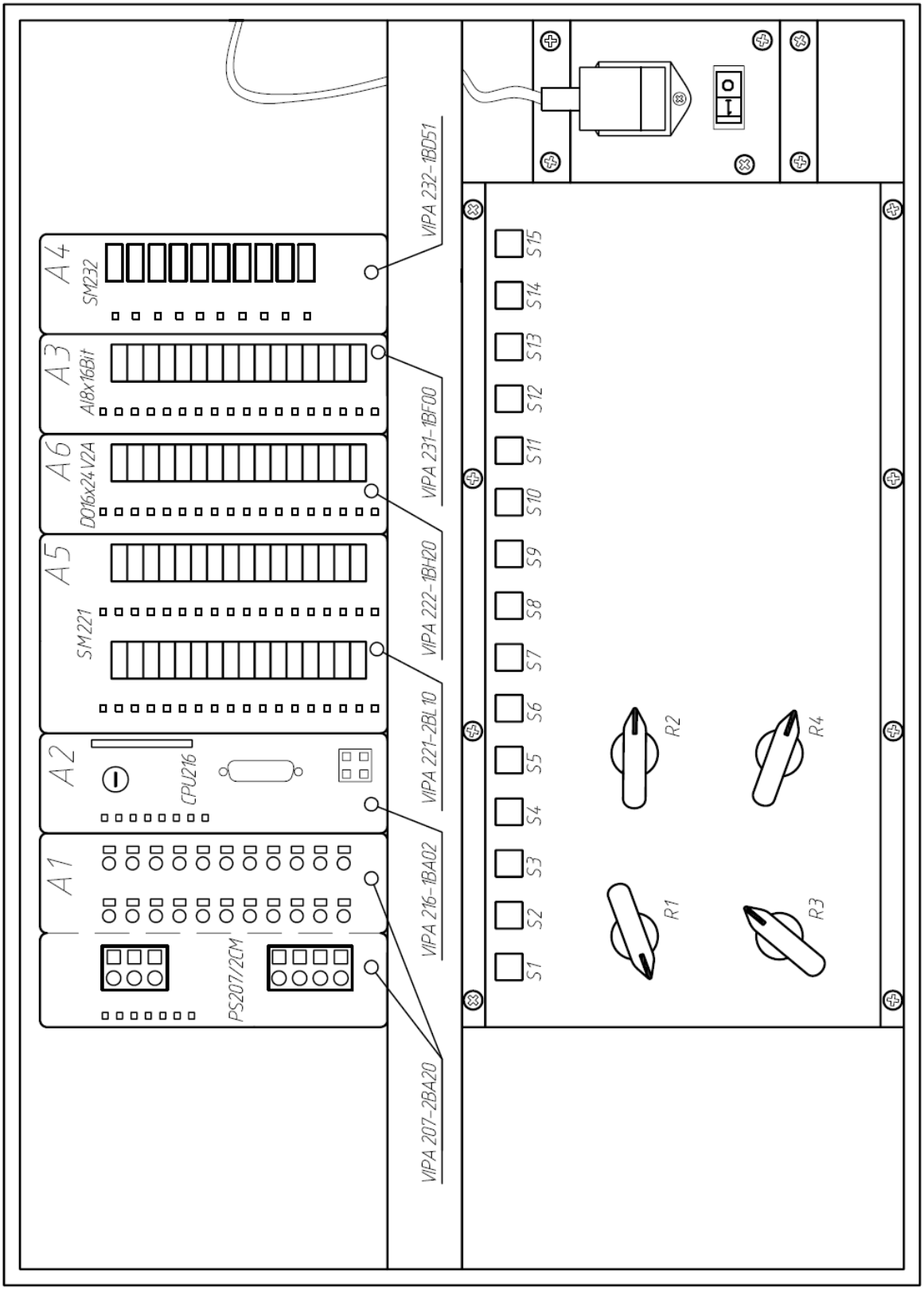
- A1 – контроллер VIPA 207, платформа (корзина с блоком питания);
- A2 – контроллер VIPA 216, базовый модуль;
- A3 – модуль аналогового ввода VIPA 231;
- A4 – модуль аналогового вывода VIPA 232;
- A5 – модуль дискретного ввода VIPA 221;
- A6 – модуль дискретного вывода VIPA 222;
- A7-A10 – вольтметр цифровой;
- A11-A14 – преобразователь напряжения 24/12 VDC;
- VD1-VD16 – светодиод (средства индикации);
- S1-S14 – переключатели (органы управления);
- R1-R4 – потенциометр (органы управления);
- X1-X4 – клеммные соединители и разъемы – средства коммутации.

Модули имеют следующие обозначения входов и выходов:

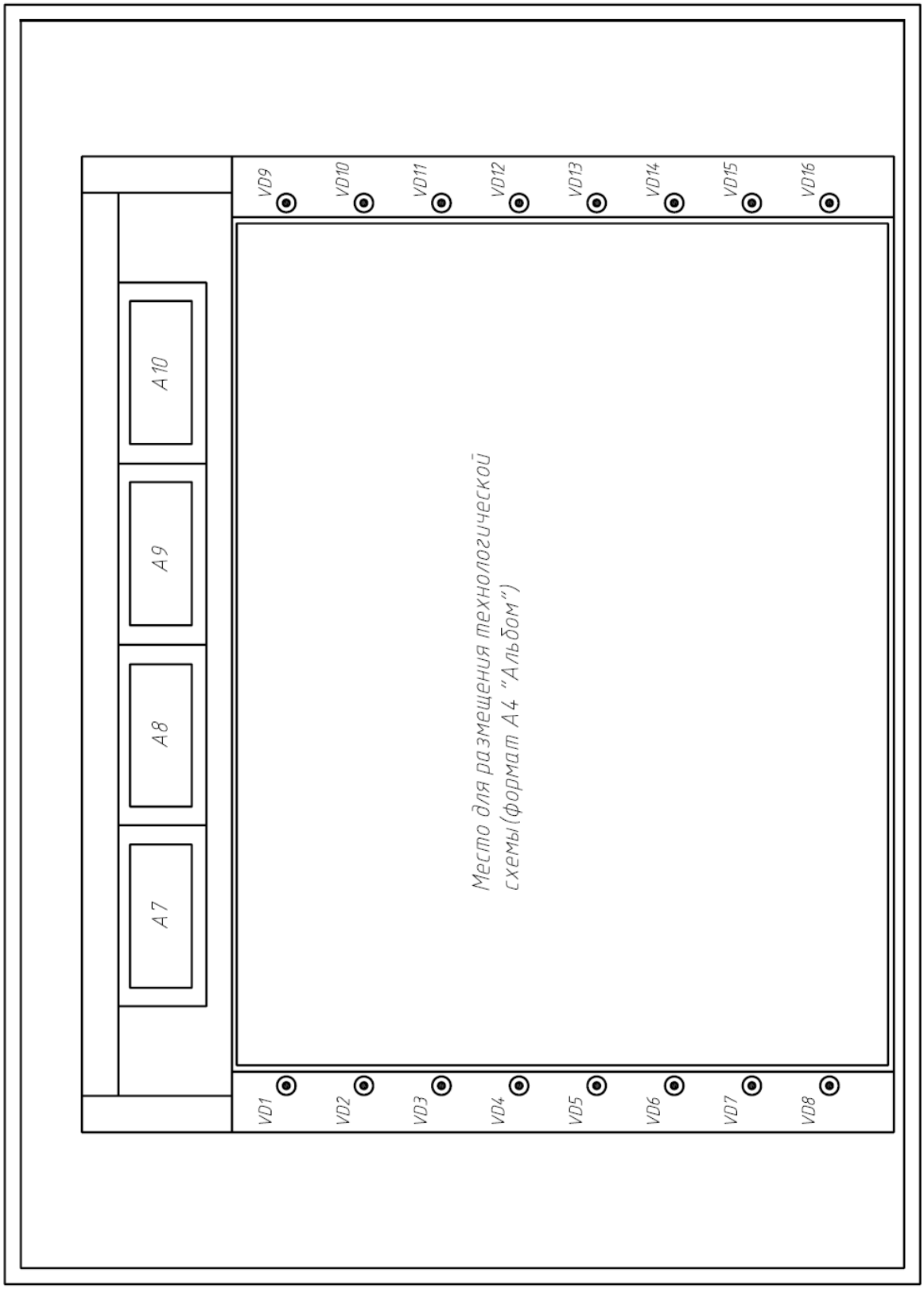
- дискретный ввод – DI;
- дискретный вывод – DO;
- аналоговый ввод – AI;
- аналоговый вывод – AO.

Подключение к сети питания 220В, 50 Гц и к персональному компьютеру осуществляется специальными кабелями под контролем преподавателя.


Изм
Лист
N° докум.
Подп.
Дата
Мобильный лабораторный стенд для изучения контроллера VIPA -216-1BA02.
Схема электрическая принципиальная.
Лист
"Сеть"
фаза 220В
нейтраль 220В
заземление
A1
X1
L
N
PE
VIPA 207-2BA20
PS207/2CM
X2
DC+24V
DC-24V
DC+24V
DC-24V
X3
X4
A2
VIPA 216-1BA02
CPU-216
Базовый модуль
A3
VIPA 231-1BF00
AI 8x16Bit
Аналоговые входы
A4
VIPA 232-1BD51
AO4x12Bit
Аналоговые выходы
0В
+24В
R1
R2
R3
R4
A5
VIPA 221-2BL10
DI32xDC24V
Дискретные входы
A6
VIPA 222-1BH20
DO16x24V 2A
Дискретные выходы
S1
S2
S3
S4
S14
VD1
VD2
VD3
VD4
VD16



Изм.	Лист	N° докум.	Подп.	Дата	Мобильный лабораторный стенд для изучения контроллера VIPA -216-1BA02. Нижняя панель. Схема монтажная.	Лист



Изм.	Лист	№ докум.	Подп.	Дата	Мобильный лабораторный стенд для изучения контроллера VIPA -216-1BA02. Верхняя панель. Схема монтажная.	Лист

Общая постановка задачи

В ходе выполнения лабораторной работы студент должен получить навыки работы с контроллерами фирмы Siemens модификации VIPA, а также с его программным продуктом WinPLC7. В процессе выполнения лабораторной работы студент должен правильно подобрать необходимую для решения поставленной задачи конфигурацию промышленного контроллера с установленным наборами модулей, выполнить программную конфигурацию при помощи отдельного модуля конфигуратора, определить величины входных сигналов и их типы, создать исполнительную программу на базе стандартных функциональных блоков, входящих в программное обеспечение WinPLC7.

Выполнение работы

Задание №1 «Управление автомобильным движением на базе ПЛК VIPA CPU216»

Необходимо разработать и отладить программу для контроллера VIPA CPU216, управляющую движением автомобильного транспорта на перекрестке при помощи светофора.

Описание алгоритма работы:

Управление работой светофора производится при помощи заранее введенных временных задержек на основе программных мультивибраторов. Один из них управляет работой счетчика и определяет направление счета. Так при нарастании значений счетчика в положительном направлении (от красного к зеленому свету) работает один из светофоров, тогда как другой работает в обратном направлении. При снижении значения счетчика производится обратная операция.

Каждый из цветов светофора загорается при достижении определенного значения счетчика. При достижении значением счетчика порога для зеленого света запускается импульсный элемент, который отрабатывает заданное время. Оставшееся время до переключения светофора на желтый свет работает другой мультивибратор, который заставляет светофор мигать зеленым светом.

Используемые компоненты:

- зеленый свет 1-го и 2-го светофора DO1,DO9 соответственно;
- желтый свет 1-го и 2-го светофора DO2,DO10 соответственно;
- красный свет 1-го и 2-го светофора DO3,DO11 соответственно.

Задание№2 «Управление пешеходным движением на базе ПЛК VIPA CPU216»

Необходимо разработать программу для контроллера VIPA, позволяющую при нажатии пешеходом кнопки уменьшить время работы красного света в нужном направлении, но только до определенного значения времени. Аналогично для кнопки для противоположного направления.

Решение на базе VIPA

При нажатии на кнопку пешехода должны загораться индикаторы DO6,DO14 соответственно. При срабатывании зеленого света в нужном направлении индикаторы должны погаснуть.

Используемые компоненты

- кнопка для пешехода с одной стороны DI1;
- кнопка для пешехода с другой стороны DI2;
- зеленый свет 1-го и 2-го светофора DO1,DO9 соответственно;
- красный свет 1-го и 2-го светофора DO2,DO10 соответственно;
- индикаторы нажатия кнопки пешехода с разных сторон индикаторы DO6,DO14 соответственно.

Задание№3 «Управление работой резервуара для смешивания на базе ПЛК VIPA»

Необходимо разработать программу для контроллера VIPA, которая на основе данных по уровню и о наличии составляющей смеси осуществляет заполнение резервуара, смешивание в нем раствора и выгрузку из него получившейся смеси.

Решение с помощью VIPA

В резервуар подается компонент №1 для приготовления смеси. По нажатию кнопки 1 происходит включение накачивающего насоса. Перед включением двигателя проверяется наличие жидкости в трубопроводе. В случае её отсутствия выдается сигнал аварии. В резервуаре контролируются три дискретных уровня (нижний H0, верхний для первого элемента H1 и верхний общий H2). По достижению H1 подача первого элемента прекращается. При этом через второй трубопровод начинается загрузка компонента №2. По достижению уровня H2 прекращается подача компонента №2 и запускается двигатель мешалки. Мешалка должна отработать 1 минуту. После этого мешалка выключается и включается откачка смеси. По достижению уровня H1 процесс повторяется.

Используемые компоненты

- запуск входного насоса – DI1
- трубопровод полон – DI2
- уровень H1 – DI13 – DO6
- уровень H2 – DI14 – DO7
- уровень H3 – DI15 – DO8
- авария – DO9
- первый насос запущен – DO1
- второй насос запущен – DO2
- мешалка запущена – DO3
- выходной насос – DO4

Контрольные вопросы к защите

1. Каковы отличия в контроллерах VIPA различных моделей (VIPA 100, VIPA 200)?
2. Что определяется разрядностью при аналоговом вводе?
3. Какие действия необходимо выполнить чтобы поменять местами модули на базе одного ПЛК?
4. Почему нельзя использовать аналоговые модули ввода/вывода без предварительной конфигурации ПЛК?
5. Всегда ли после редактирования программы в различных языках можно сделать переключение между ними и почему?
6. Какое максимальное количество модулей ввода/вывода поддерживается контроллером VIPA?

Способ оценки результатов

По лабораторной работе предусмотрена защита. Лабораторная работа будет зачтена, если преподавателю будут продемонстрирована программа, отрабатывающая полученное задание, а также, если студент ответит на заданные преподавателем вопросы. Папка с созданным проектом передается преподавателю.

Лабораторная работа №4. Изучение PC-совместимых ПЛК серии ADAM-5510

Требования к содержанию, оформлению и порядку выполнения

Перед выполнением лабораторной работы студент должен изучить теоретическую часть курса. Студент, выполняющий лабораторную работу в составе с реальным лабораторным стендом, обязан получить к ней допуск путем собеседования с преподавателем, который заключается в подтверждении готовности студента к ее выполнению, а именно, знание основ теории по данной теме, порядка выполнения лабораторной работы, особенностей и правил работы с лабораторным стендом. Преподаватель определяет вариант задания для выполнения лабораторной работы конкретным студентом.

Студент при работе с персональным компьютером, на котором установлено прикладное и коммуникационное программное обеспечения для работы с данным лабораторным стендом, должен создать новую папку под именем «Ваша фамилия, Лабораторная работа №_» (например: «Симонов А.М. Лабораторная работа №4»).

Выполнение лабораторной работы производится в 2 этапа:

1. Разработка программы и отладка ее на программном эмуляторе того продукта, который предназначен для данного стенда.
2. Демонстрация преподавателю работы отлаженной программы, которая должна функционировать в составе с реальным

лабораторным стендом в строгом соответствии с полученным заданием (только для студентов очной формы обучения).

3. Сохранение созданного проекта, содержащего разработанную и отлаженную программу, в вышеуказанную папку.

После успешного выполнения лабораторной работы студент должен оформить отчет в соответствии с предъявляемыми требованиями и осуществить ее защиту, проводимую в форме собеседования с преподавателем. Цель защиты – подтверждение факта самостоятельного выполнения лабораторной работы и определение степени усвоения студентом теоретического материала по данной теме.

Описание лабораторного стенда

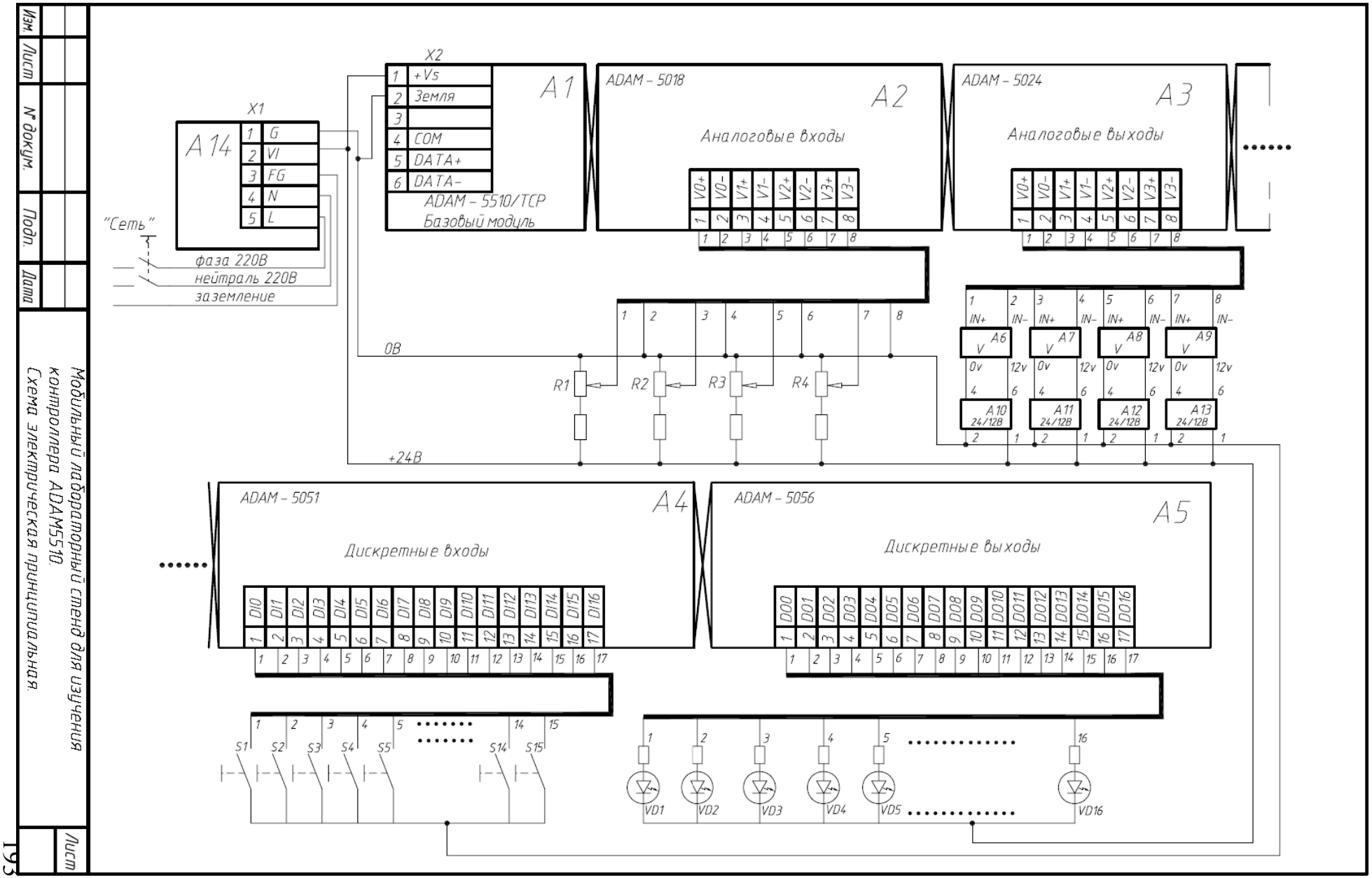
Лабораторный стенд выполнен в переносном корпусе чемоданного типа, он состоит из верхней откидной панели, и нижнего основания. В нижней части смонтированы: блок питания, промышленный контроллер, модули ввода-вывода, устройства коммутации, органы управления и индикации. В верхней части стенда смонтированы органы индикации и цифровые измерительные модули аналоговых сигналов. Ниже приведены электрические принципиальные схемы и схемы монтажные исполнения лабораторного стенда. На схемах введены обозначения:

- A1 – ПЛК ADAM-5510, базовый модуль;
- A2 – модуль аналогового ввода ADAM-5018;
- A3 – модуль аналогового вывода ADAM-5024;
- A4 – модуль дискретного ввода ADAM-5051;
- A5 – модуль дискретного вывода ADAM-5056;
- A6-A9 – вольтметр цифровой;
- A10-A13 – преобразователь напряжения 24/12 VDC;
- A14 – блок питания 24VDC;
- VD1-VD16 – светодиод (средства индикации);
- S1-S15 – переключатели (органы управления);
- R1-R4 – потенциометр (органы управления);
- X1-X2 – клеммные соединители и разъемы – средства коммутации.

Модули имеют следующие обозначения входов и выходов:

- дискретный ввод – DI;
- дискретный вывод – DO;
- аналоговый ввод/вывод – V;

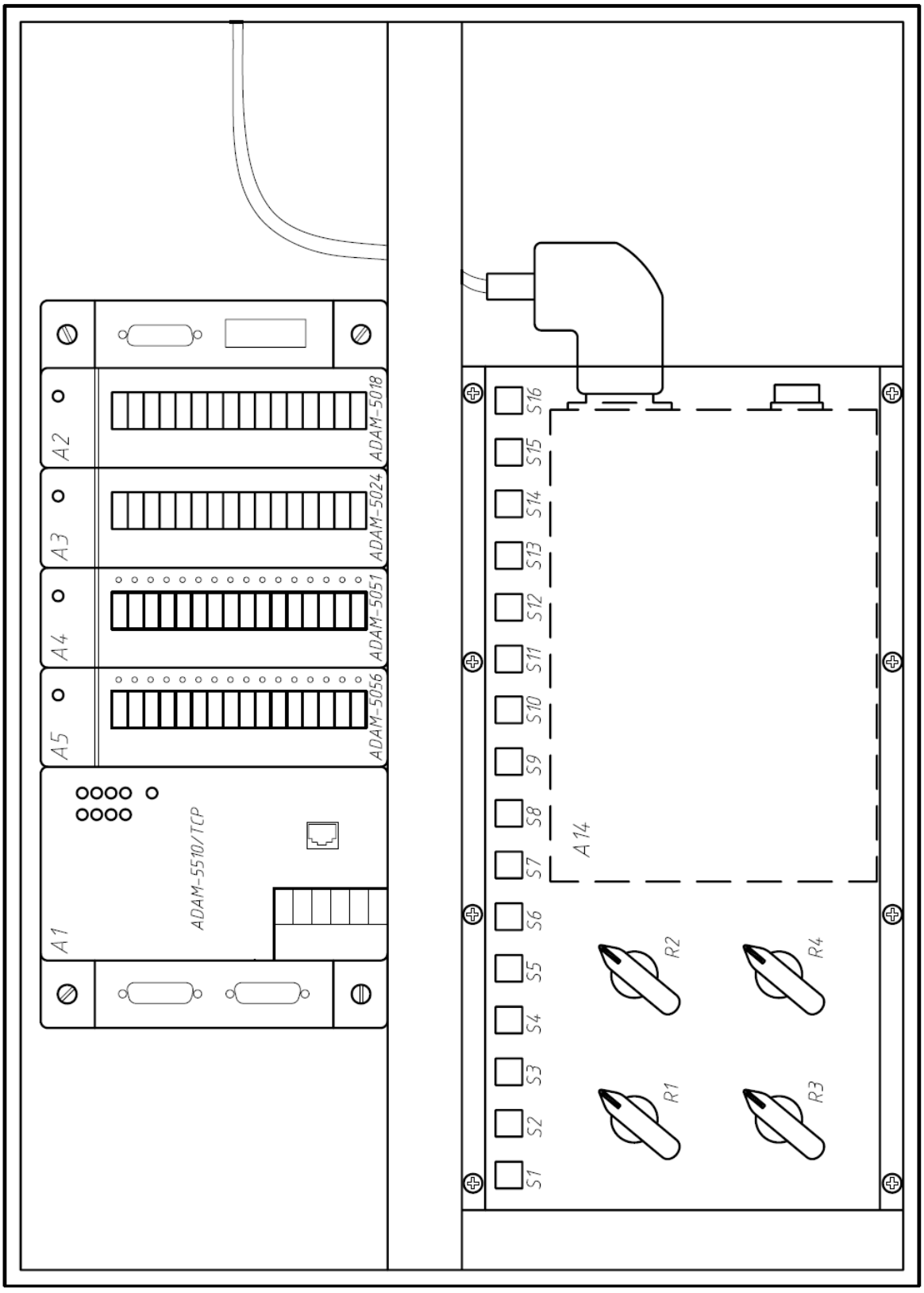
Подключение к сети питания 220В, 50 Гц и к персональному компьютеру осуществляется специальными кабелями под контролем преподавателя.



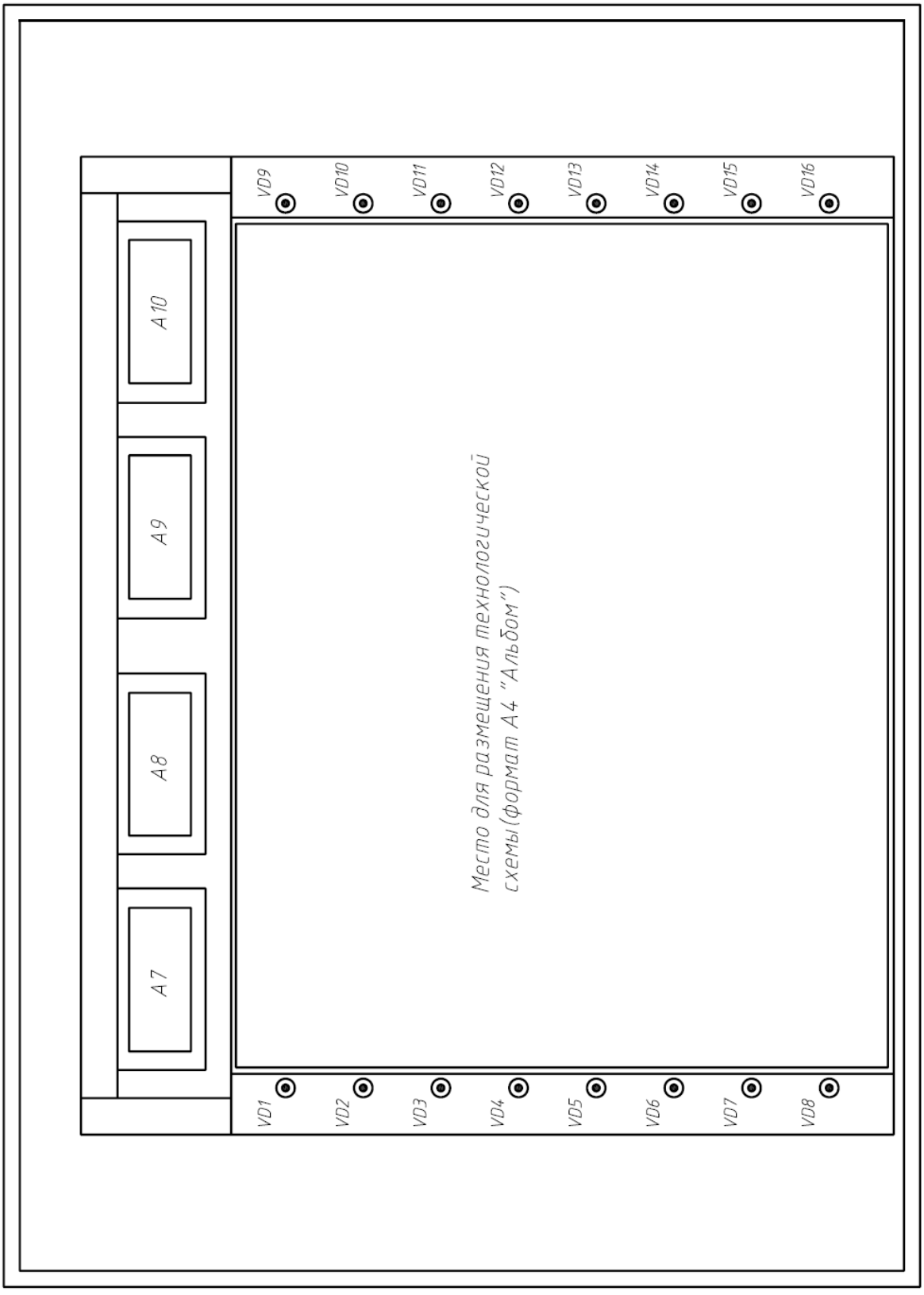
Изм. Лист № докум. Подп. Дата

Мобильный лабораторный стенд для изучения контроллера ADAM5510.
Схема электрическая принципиальная.

Лист



Изм.	Лист	№ докум.	Подп.	Дата	Мобильный лабораторный стенд для изучения контроллера ADAM5510. Нижняя панель. Схема монтажная.	Лист



Изм.	Лист	N° докум.	Подп.	Дата	Мобильный лабораторный стенд для изучения контроллера ADAM5510. Верхняя панель. Схема монтажная.	Лист

Общая постановка задачи

В ходе выполнения лабораторной работы студент должен получить навыки работы с контроллерами фирмы Advantech модификации ADAM-5510 и программного обеспечения SCADA-системы MasterSCADA v.2.5, С-ядро которой прошито в ПЛК. В процессе выполнения лабораторной работы студент должен правильно подобрать необходимую для решения поставленной задачи конфигурацию промышленного контроллера с установленным наборами модулей, произвести прошивку ПЛК как программой написанной на языке программирования Си в среде Borland, так и на основе SCADA-системы, определить величины входных сигналов и их типы, создать исполнительную программу на базе стандартных функциональных блоков, входящих в программное обеспечение MasterSCADA. При работе с программным продуктом MasterSCADA необходимо также получить навыки создания визуальных объектов и мнемосхем, в том числе интерактивных и анимированных.

Выполнение работы

Задание №1 «Управление воротами и освещением гаража на базе ПЛК ADAM»

На базе контроллера ADAM-5510 необходимо разработать систему управления воротами и освещением гаража.

Описание алгоритма работы:

Нажатие на кнопку DI1 включает освещение в гараже (DO3) и включает двигатель, открывающий ворота гаража (DO1). Двигатель выключается по срабатыванию концевого выключателя (DI3). Кнопка DI2 включает двигатель закрытия ворот (DO2). Отключение двигателя закрытия ворот выполняет концевой выключатель DI4. Свет в гараже гаснет через 5 секунд после закрытия ворот гаража.

Используемые компоненты:

- ADAM-5510;
- DI1 кнопка «открыть ворота гаража» (замыкающий контакт);
- DI2 кнопка «закрыть ворота гаража» (замыкающий контакт);
- DI3 концевой выключатель двигателя, открывающего ворота;
- DI4 концевой выключатель двигателя, закрывающего ворота;
- DO1 двигатель, открывающий ворота;
- DO2 двигатель, закрывающий ворота;
- DO3 освещение гаража.

Задание№2 «Управление подъемной платформой на базе ПЛК ADAM»

На базе контроллера ADAM-5510 необходимо реализовать управление подъемной платформой. Для контроля зоны перемещения подъемной платформы установлено несколько параллельно включенных датчиков (например, ультразвуковых).

Описание алгоритма работы:

Подъемная платформа с помощью кнопок может перемещаться вверх или вниз. Для этого к DI1 подключена кнопка «Вверх», а к D3 –кнопка «Вниз». Соответствующее конечное положение распознается конечным выключателем. Конечный выключатель на DI2 – для верхнего положения платформы, конечный выключатель на DI4 – для нижнего положения.

Если конечное положение достигнуто, то движение можно только в противоположном направлении. Направление перемещения задается с помощью кнопок DI1 и DI3. Останов платформы осуществляется посредствам кнопки «Стоп» (DI7).

Ультразвуковые датчики для контроля зоны перемещения платформы подключены к DI5. Если датчиками распознается препятствие, то платформа останавливается, в этом случае платформу можно перемещать в ручном режиме, если клавиша направления нажата дольше 2 секунд.

Однако, если нажата кнопка аварийного останова на DI7, то платформа останавливается немедленно и не может больше перемещаться с помощью кнопок управления направлением, пока не отпущена кнопка аварийного останова.

Для лучшего распознавания того, что платформа движется, активизируется предупредительное сигнальное устройство на DO3. Если платформа движется вверх или вниз, то мигает предупредительный световой сигнал на DO3.

Используемые компоненты

- DI1 Кнопка «Вверх» (замыкающий контакт)
- DI2 Верхний конечный выключатель (замыкающий контакт)
- DI3 Кнопка «Вниз» (замыкающий контакт)
- DI4 Нижний конечный выключатель (замыкающий контакт)
- DI5 Датчики (размыкающие контакты)
- DI6 Кнопка «Стоп» (замыкающий контакт)
- DI7 Кнопка аварийного останова (замыкающий контакт)
- DO1 Платформа едет вверх
- DO2 Платформа едет вниз
- DO3 Предупреждающий световой сигнал

Задание№3 «Управление и контроль загрузочных устройств бункеров на базе ПЛК ADAM»

С помощью контроллера ADAM-5510 необходимо реализовать управление и контроль загрузочных устройств бункеров. Бункеры заполняются через загрузочный шланг грузового автомобиля сыпучим материалом.

Описание алгоритма работы:

Процесс загрузки может быть начат только тогда, когда включен разблокирующий выключатель на DI1 и загрузочный шланг надлежащим образом подключен. Язычковый контакт на загрузочном штуцере

сигнализирует, правильно ли загрузочный шланг соединен с бункером. Это сигнал считывается в ADAM через вход DI2. Затем открывается запорный клапан на Q2, после чего в бункер закачивается сыпучий материал. Одновременно вводится в действие выпускной фильтр на DO1. Он должен быть включен в течение всего процесса заполнения.

Если бункер полон, то это сигнализируется через предельный выключатель уровня заполнения на DI3. Звуковой сигнал сообщает оператору, что до автоматического завершения процесса еще осталось 99 секунд. В течение этого времени должен быть закрыт клапан на грузовом автомобиле, чтобы еще освобождался от содержимого загрузочный шланг. Звуковой сигнал может быть досрочно отключен с помощью квитирующей кнопки на DI6, или он будет автоматически выключен через 25 секунд.

Если шланг не удалось своевременно освободить, то по средствам кнопки DI5 можно выполнить аварийное заполнение в течение 30 секунд.

Контроль избыточного давления в бункере также автоматически отключает процесс заполнения. Это отображается с помощью сигнальной лампы на DO4.

Используемые компоненты

- DI1 Деблокирующий выключатель (замыкающий контакт)
- DI2 Язычковый контакт загрузочного штуцера (замыкающий контакт)
- DI3 Предельный выключатель уровня заполнения (замыкающий контакт)
- DI4 Выключатель избыточного давления (размыкающий контакт)
- DI5 Кнопка аварийного заполнения (замыкающий контакт)
- DI6 Кнопка квитирования звукового сигнала (замыкающий контакт)
- DO1 Фильтр
- DO2 Запорный клапан
- DO3 Звуковой сигнал
- DO4 Световой сигнал избыточного давления

Контрольные вопросы к защите

1. При помощи какого программного продукта и каким образом возможно с конфигурировать контроллера?
2. Какая технология позволяет подключать контроллер к SCADA-системе?
3. Возможно ли подключение других контроллеров (LOGO!, VIPA) к SCADA-системе и возможно ли аналогично ПЛК ADAM прошить ядро системы в ПЛК?
4. Какой язык программирования стандарта МЭК 6 используется для создания встраиваемого ядра MasterSCADA?
5. Возможно ли создание визуализации при помощи подключения разных ПЛК одновременно?
6. Какой интерфейс связи необходим для прошивки ПЛК?

Способ оценки результатов

По лабораторной работе предусмотрена защита. Лабораторная работа будет зачтена, если преподавателю будут продемонстрирована программа, отрабатывающая полученное задание, а также, если студент ответит на заданные преподавателем вопросы. Папка с созданным проектом передается преподавателю.

Лабораторная работа №5. Управление микроклиматом помещения АТС на базе ПЛК серии ADAM-5510

Требования к содержанию, оформлению и порядку выполнения

Перед выполнением лабораторной работы студент должен изучить теоретическую часть курса. Студент, выполняющий лабораторную работу в составе с реальным лабораторным стендом, обязан получить к ней допуск путем собеседования с преподавателем, который заключается в подтверждении готовности студента к ее выполнению, а именно, знание основ теории по данной теме, порядка выполнения лабораторной работы, особенностей и правил работы с лабораторным стендом. Преподаватель определяет вариант задания для выполнения лабораторной работы конкретным студентом.

Студент при работе с персональным компьютером, на котором установлено прикладное и коммуникационное программное обеспечения для работы с данным лабораторным стендом, должен создать новую папку под именем «Ваша фамилия, Лабораторная работа №_» (например: «Симонов А.М. Лабораторная работа №5»).

Выполнение лабораторной работы производится в 2 этапа:

4. Разработка программы и отладка ее на программном эмуляторе того продукта, который предназначен для данного стенда.
5. Демонстрация преподавателю работы отлаженной программы, которая должна функционировать в составе с реальным лабораторным стендом в строгом соответствии с полученным заданием (только для студентов очной формы обучения).
6. Сохранение созданного проекта, содержащего разработанную и отлаженную программу, в вышеуказанную папку.

После успешного выполнения лабораторной работы студент должен оформить отчет в соответствии с предъявляемыми требованиями и осуществить ее защиту, проводимую в форме собеседования с преподавателем. Цель защиты – подтверждение факта самостоятельного выполнения лабораторной работы и определение степени усвоения студентом теоретического материала по данной теме.

Описание лабораторного стенда

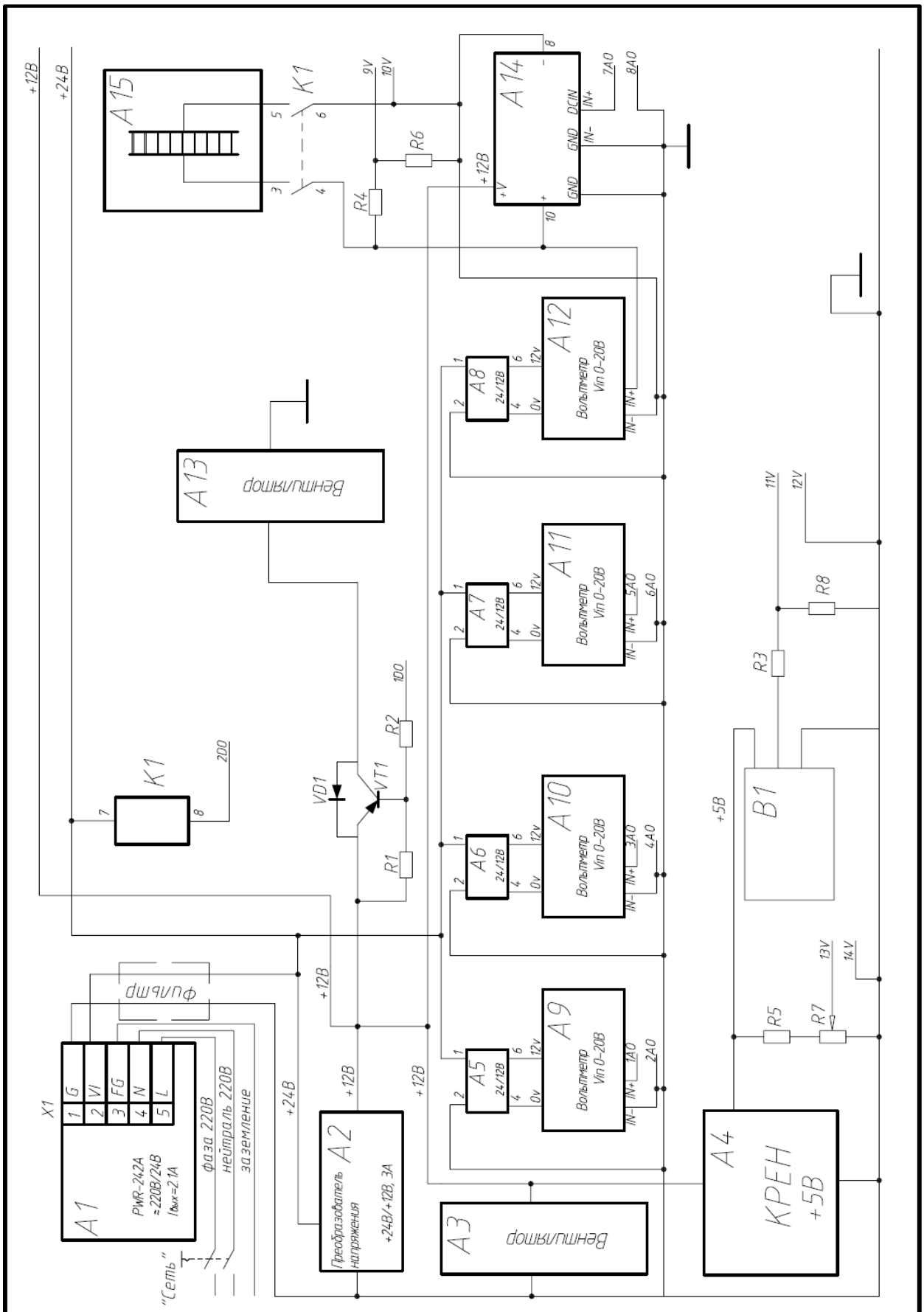
Лабораторный стенд выполнен в переносном корпусе чемоданного типа, он состоит из верхней откидной панели, и нижнего основания. В нижней части смонтированы: блок питания, промышленный контроллер, модули ввода-вывода, устройства коммутации, органы управления и индикации. В верхней части стенда смонтированы органы индикации и цифровые измерительные модули аналоговых сигналов. Ниже приведены электрические принципиальные схемы и схемы монтажные исполнения лабораторного стенда. На схемах введены обозначения:

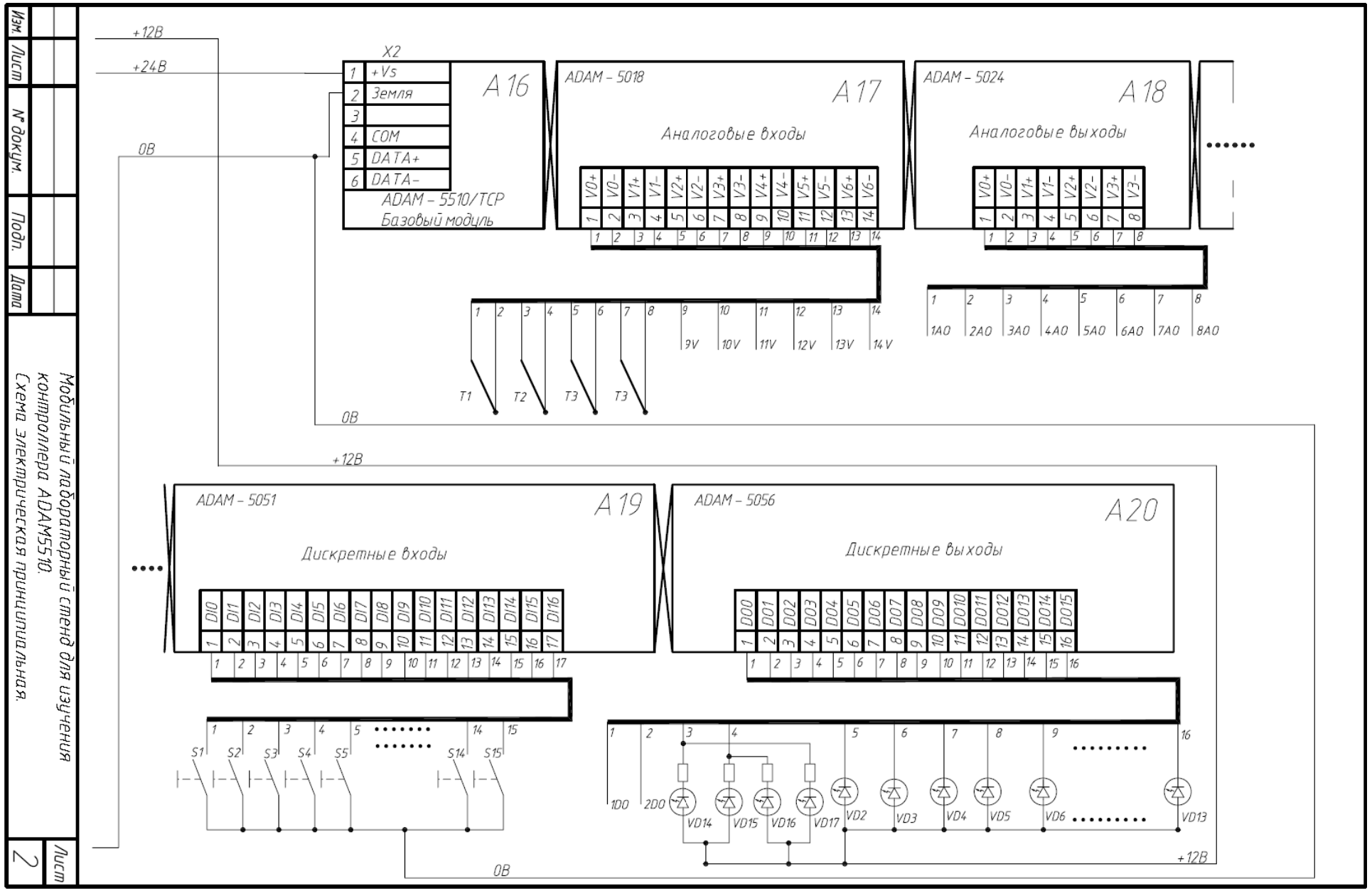
- A1 – блок питания 24VDC;
- A2, A5-A8 – преобразователь напряжения 24/12 VDC;
- A3 – вентилятор;
- A4 – стабилизатор напряжения 5 VDC;
- A9-A12 – вольтметр цифровой;
- A14 – силовой ключ;
- A15 – элемент Пельтье;
- A16 – ПЛК ADAM-5510, базовый модуль;
- A17 – модуль аналогового ввода ADAM-5018;
- A18 – модуль аналогового вывода ADAM-5024;
- A19 – модуль дискретного ввода ADAM-5051;
- A18 – модуль дискретного вывода ADAM-5056;
- K1 – электромагнитное реле;
- T1-T4 – термоэлектрический преобразователь;
- VD2-VD17 – светодиод (средства индикации);
- S1-S15 – переключатели (органы управления);
- X1-X2 – клеммные соединители и разъемы – средства коммутации.

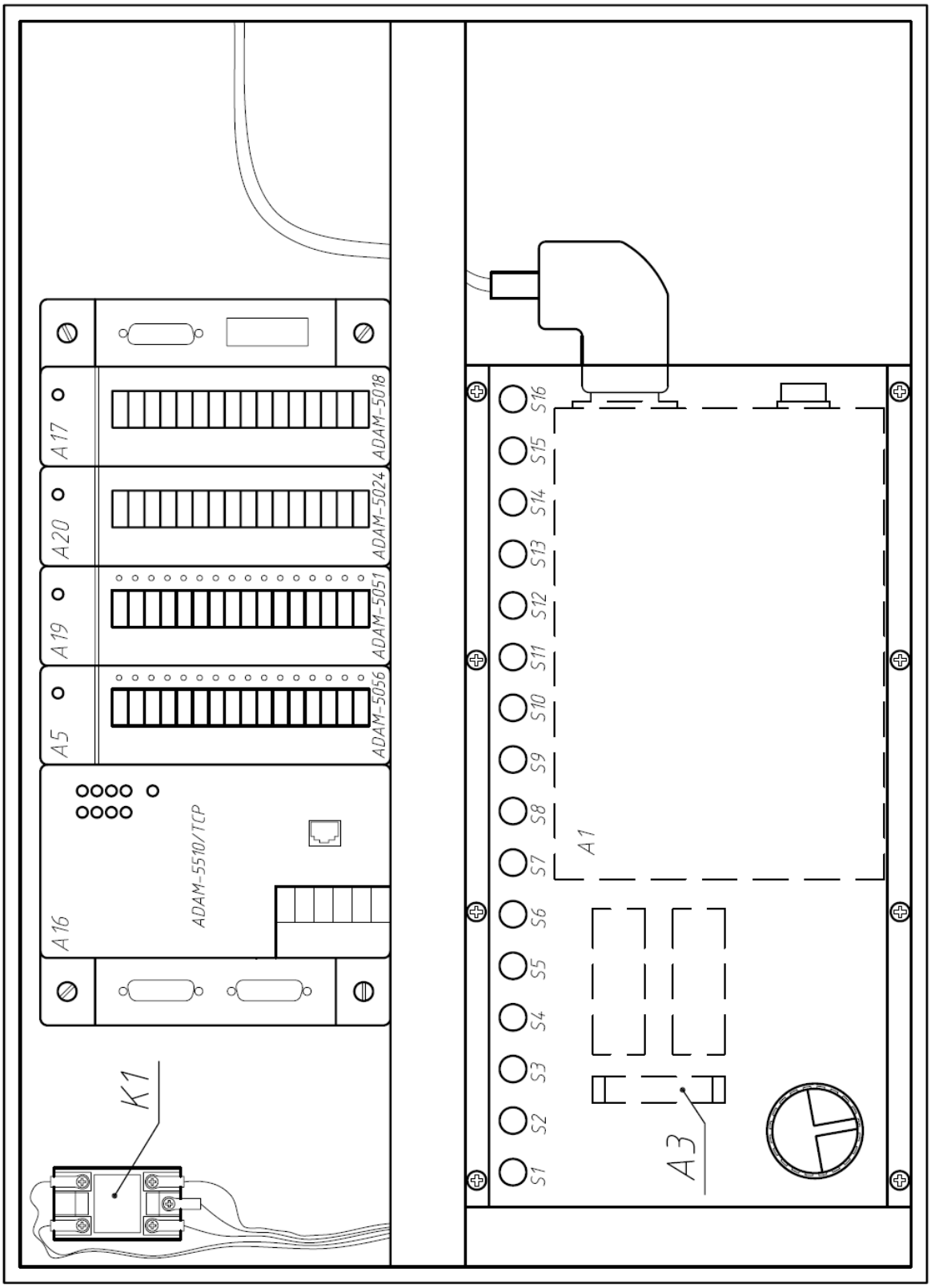
Модули ввода-вывода имеют следующие обозначения входов и выходов:

- дискретный ввод – DI;
- дискретный вывод – DO;
- аналоговый ввод/вывод – V;

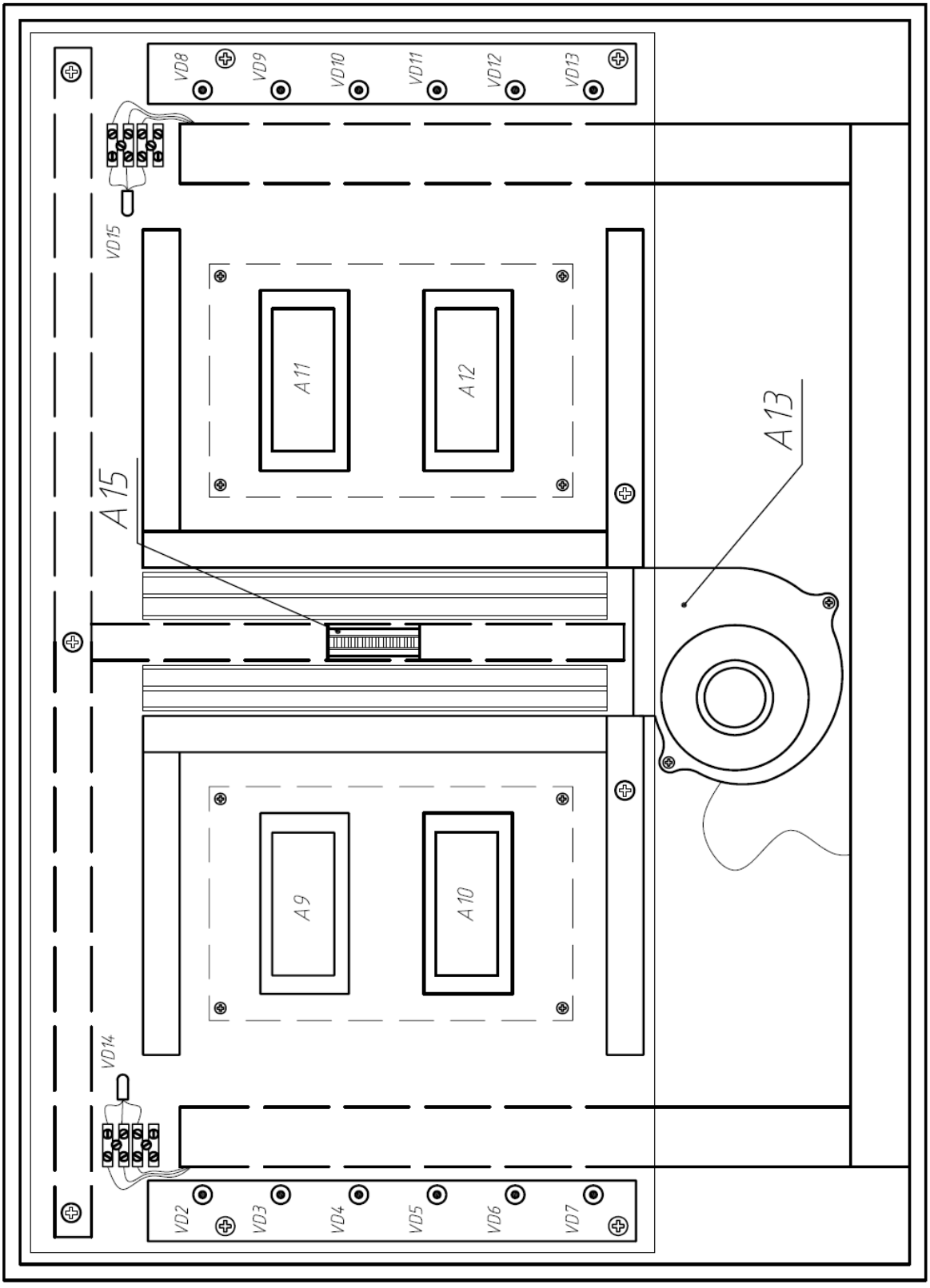
Подключение к сети питания 220В, 50 Гц и к персональному компьютеру осуществляется специальными кабелями под контролем преподавателя.


+12В
+24В
A15
K1
9V
10V
R6
R4
A14
DCIN
GND
IN-
IN+
7AO
8AO
A13
Вентилятор
A12
A8
24/12В
Вольтметр
Vin 0-20В
A11
A7
5AO
6AO
R8
R3
11V
12V
R2
100
200
VD1
VT1
R1
A10
A6
3AO
4AO
+5В
B1
Фильтр
X1
A1
PWR-242A
~220В/24В
Iвых=2.1А
Фаза 220В
нейтраль 220В
заземление
"Сеть"
A2
Преобразователь напряжения
+24В/+12В, 3А
A9
A5
1AO
2AO
R5
R7
13V
14V
A3
Вентилятор
A4
КРЕН
+5В
Мобильный лабораторный стенд для изучения контроллера ADAM5510.
Схема электрическая принципиальная.
Изм.
Лист
№ докум.
Подп.
Дата
Лист
1


Изм
Лист
N° докум.
Подп.
Дата
Мобильный лабораторный стенд для изучения контроллера ADAM5510.
Схема электрическая принципиальная.
Лист
2
+12В
+24В
0В
X2
1 +Vs
2 Земля
3
4 COM
5 DATA+
6 DATA-
A16
ADAM – 5510/TCP
Базовый модуль
ADAM – 5018
A17
Аналоговые входы
V0+ V0- V1+ V1- V2+ V2- V3+ V3- V4+ V4- V5+ V5- V6+ V6-
ADAM – 5024
A18
Аналоговые выходы
V0+ V0- V1+ V1- V2+ V2- V3+ V3-
1A0 2A0 3A0 4A0 5A0 6A0 7A0 8A0
9V 10V 11V 12V 13V 14V
T1 T2 T3 T3
ADAM – 5051
A19
Дискретные входы
DI0 DI1 DI2 DI3 DI4 DI5 DI6 DI7 DI8 DI9 DI10 DI11 DI12 DI13 DI14 DI15 DI16
S1 S2 S3 S4 S5 S14 S15
ADAM – 5056
A20
Дискретные выходы
DO0 DO1 DO2 DO3 DO4 DO5 DO6 DO7 DO8 DO9 DO10 DO11 DO12 DO13 DO14 DO15
1DO 2DO
VD14 VD15 VD16 VD17 VD2 VD3 VD4 VD5 VD6 VD13



Изм.	Лист	№ докум.	Подп.	Дата	Мобильный лабораторный стенд для изучения контроллера ADAM5510. Нижняя панель. Схема монтажная.	Лист



Изм.	Лист	№ докум.	Подп.	Дата	Мобильный лабораторный стенд для изучения контроллера ADAM5510. Верхняя панель. Схема монтажная.	Лист

Общая постановка задачи

В ходе выполнения лабораторной работы студент должен получить навыки работы с контроллерами фирмы Advantech модификации ADAM-5510 и программного обеспечения SCADA-системы MasterSCADA v.2.5, С-ядро которой прошито в ПЛК. В процессе выполнения лабораторной работы студент должен правильно подобрать необходимую для решения поставленной задачи конфигурацию промышленного контроллера с установленным наборами модулей, произвести прошивку ПЛК на основе SCADA-системы, определить величины входных сигналов и их типы, создать исполнительную программу на базе стандартных функциональных блоков, входящих в программное обеспечение MasterSCADA. При работе с программным продуктом MasterSCADA необходимо также получить навыки создания визуальных объектов и мнемосхем, в том числе интерактивных и анимированных.

Выполнение работы

Задание№1 «Управление микроклиматом на базе ПЛК ADAM»

На базе контроллера ADAM-5510 необходимо разработать систему управления воротами и освещением гаража.

Описание алгоритма работы:

Нажатие на кнопку DI3 включает освещение в гараже (DO11) и включает двигатель, открывающий ворота гаража (DO9). Двигатель выключается по срабатыванию концевого выключателя (DI5). Кнопка DI4 включает двигатель закрытия ворот (DO10). Отключение двигателя закрытия ворот выполняет концевой выключатель DI6. Свет должен гореть, пока работает датчик движения. Имитация работы датчика выполнена на DI7 (есть движение) и DI8 (нет движения). Индикация движения в гараже выполнена на DO12.

Свет в гараже гаснет через 5 секунд после закрытия ворот гаража, если нет движения. На экране компьютера должно отражаться включение-выключение двигателей управления воротами, включение-выключение освещения гаража.

Также можно включать/отключать кондиционер (DI1/DI15) и управлять мощностью работы кондиционера AI6. Мощность кондиционера выдается на индикатор №3. Нагреватель включается входом DI2 и выключается DI14. При этом меняется температура в каналах отопления и охлаждения, что отображается на индикаторах 1,2. Напряжение на элементе Пельтье отображается на индикаторе №4.

Используемые компоненты:

- ADAM-5510;
- DI1 кнопка включения вентилятора;
- DI2 кнопка включения нагревателя;
- DI3 кнопка «открыть ворота гаража» (замыкающий контакт);
- DI4 кнопка «закрыть ворота гаража» (замыкающий контакт);

- DI5 концевой выключатель двигателя, открывающего ворота;
- DI6 концевой выключатель двигателя, закрывающего ворота;
- DI7 датчик присутствия (движения);
- DI14 кнопка выключения нагревателя;
- DI15 кнопка выключения вентилятора;
- AI1,2 термопары каналов воздуховода;
- AI6 аналоговый задатчик;
- DO9 двигатель, открывающий ворота;
- DO10 двигатель, закрывающий ворота;
- DO11 освещение гаража;
- DO12 индикатор перемещения;
- AO1 индикация температуры;
- AO2 индикация температуры;
- AO3 индикация процентов мощности кондиционера.

Задание№2 «Управление безопасностью и аварийной вентиляцией помещения на базе ПЛК ADAM»

На базе контроллера ADAM-5510 необходимо реализовать управление охранной сигнализации помещения и аварийной вентиляцией.

Описание алгоритма работы:

Помещение является охраняемым объектом с сигнализацией и возможностью управления вентиляцией. Входы DI1, DI2, DI3, AI6 соответственно обозначают работу датчика разбития стекла, датчика движения, датчика шума, датчик давления на пол (аналоговый) и работают в ночное время (эмуляция временными импульсами). При появлении хотя бы одного компонента начинает работать сигнализация (DO3), при этом происходит блокировка всех возможных замков (DO4). Также есть пожарная сигнализация (DO7), которая срабатывает при превышении максимальной заданной температуры AI1, либо при срабатывании датчика задымления (DI4). При пожарной сигнализации срабатывает аварийная вентиляция (DO1) с повышенной вытяжной мощностью для удаления дыма. Все элементы а также исполнительные устройства, включая звуки сигнализации должны быть представлены на мнемосхеме в виде визуально понятных элементов и существующих звуков.

Используемые компоненты

- DI1 датчика разбития стекла
- DI2 датчика движения
- DI3 датчика шума
- DI4 датчика задымления
- AI1 температура помещения
- AI6 давления на пол помещения
- DO1 аварийная вентиляция
- DO3 охранная сигнализация
- DO4 блокировка замков

- DO7 пожарная сигнализация

Контрольные вопросы к защите

1. Что такое SCADA-система?
2. Различаются ли интерфейсы прошивки и работы контроллера?
3. Можно ли производить загрузку разработанной программы из MasterSCADA?
4. Можно ли прошить в ПЛК отдельную программу помимо уже созданной MasterSCADA, не удаляя ее, если да то каким образом?
5. Что необходимо сделать, чтобы загрузить программу или прошивку в ПЛК, что при этом происходит?
6. Может ли часть программы выполняться на контроллере, а часть на компьютере, если да, то как организовать данный обмен?

Способ оценки результатов

По лабораторной работе предусмотрена защита. Лабораторная работа будет зачтена, если преподавателю будут продемонстрирована программа, отрабатывающая полученное задание, а также, если студент ответит на заданные преподавателем вопросы. Папка с созданным проектом передается преподавателю.

Список сокращений

АЛУ – арифметико-логическое устройство;
АЦП – аналого-цифровой преобразователь;
БИС – большая интегральная схема;
БПВ – быстрое преобразование Фурье;
ВМК – внутренняя контроллерная магистраль;
ВУ – внешнее устройство;
ЗУ – запоминающее устройство;
ЗЯ – запоминающая ячейка памяти;
ИМС – интегральная микросхема;
КОП – код операции;
МК – микроконтроллер;
МОП – металл-окисел-полупроводник;
МП – микропроцессор;
МС – микропроцессорная система;
НК – массив ячеек памяти (накопитель);
ОЗУ – оперативное запоминающее устройство;
ОК – открытый коллектор;
ОС – обратная связь;
ПД – память данных;
ПДП – прямой доступ к памяти;
ПЗУ – постоянное запоминающее устройство;

ПК – память команд;
ПЛИС – программируемая логическая интегральная схема;
ПЛК – программируемый логический контроллер;
ПЦОС – процессор цифровой обработки сигналов;
РОН – регистр общего назначения;
СБИС – сверхбольшая интегральная схема;
СОЗУ – сверхоперативное запоминающее устройство;
ССП – слово состояния процессора;
ТТЛ – транзисторно-транзисторная логика;
УВВ – устройство ввода/вывода;
УУ – устройство управления;
ЦАП – цифро-аналоговый преобразователь;
ЦП – центральный процессор;
ША – шина адреса;
ШД – шина данных;
ШУ – шина управления;
ЭВМ – электронная вычислительная машина.

Список источников*

1. Новиков, Ю.В. Основы микропроцессорной техники / Ю.В. Новиков, П.К. Скоробогатов. – М.: Интернет-Университет Информационных Технологий, 2006. – 440 с.
2. Кустарев, В.П. Специализированные процессоры. Процессоры для встраиваемых применений: Конспект лекций. Разделы 1-2/ В.П. Кустарев. – СПб.: СПбГИТМО (ТУ), 2002. – 30 с.
3. Круг, П.Г. Процессоры цифровой обработки сигналов: Учебное пособие/ П.Г. Круг. – М.: Издательство МЭИ. 2001 – 128 с.
4. Денисов, К.М. Микропроцессорная техника: Электронный конспект лекций / К.М. Денисов – СПб: Санкт-Петербургский государственный университет информационных технологий, механики и оптики.
5. Рафикузаман, М. Микропроцессоры и машинное проектирование микропроцессорных систем: В 2-х кн. Кн. 1. Пер. с англ. / М. Рафикузаман – М.: Мир, 1988. – 312 с.
6. Прищепа, С.Л. Цифровые и микропроцессорные устройства систем телекоммуникаций: Конспект лекций по курсу: Ч. I. Цифровые устройства систем телекоммуникаций / С.Л. Прищепа.
7. Гребешков, А.Ю. Микропроцессорные системы и программное обеспечение в средствах связи: Учебное пособие. Гриф УМО/ А.Ю. Гребешков. – Самара: ПГАТИ, 2007. – 223 с.

(*) Кроме указанных источников при составлении настоящего курса лекций использовалась открытая информация из сети Интернет.

Учебно-методическое пособие

МИШУНИН ВАДИМ ВАСИЛЬЕВИЧ
ЛИХОЛОБ ПЕТР ГЕОРГИЕВИЧ

МИКРОПРОЦЕССОРЫ И ЦИФРОВАЯ ОБРАБОТКА СИГНАЛОВ

Изд. лиц.

Подписано в печать . Формат 60x84/16. Уч.-изд.л. . Усл.печ.л. .

Тираж экз. Заказ Цена договорная

Отпечатано в Белгородском государственном университете

308015, г. Белгород, ул. Победы, 85