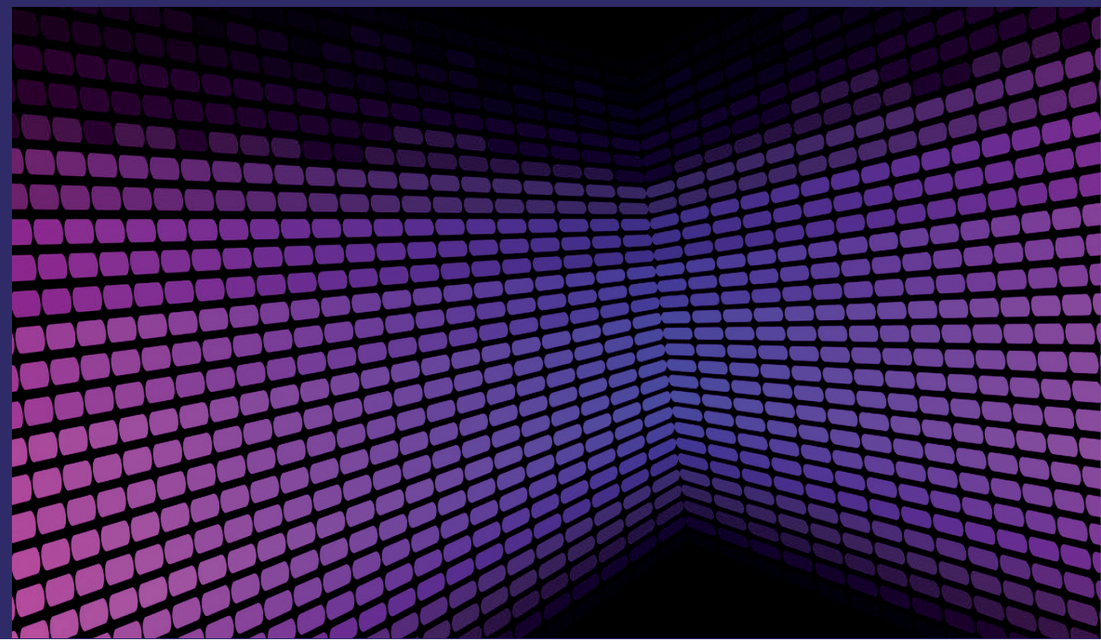
Рассмотрены микроконтроллеры фирмы INTEL 8051 (31), 80C51 (31), 8x51 FA, FB, FC, и микроконтроллеры фирмы Analog Devices ADuC812 для сбора, обработки информации и управления различными объектами с ядром 8051. Приведены струк-туры микроконтроллеров, базовые системы команд и примеры программирования. Встраиваемый однокристальный микроконтроллер (embedded microcontroller) представляет собой изготовленную на одном кристалле микропроцессорную систему, ориентированную на реализацию алгоритмов цифрового управления различными объектами и процессами. Микроконтроллер содержит центральный процессор, внутреннюю постоянную и оперативную память, параллельные и последовательные порты ввода/вывода информации, периферийные устройства: таймеры, аналого-цифровые и цифро-аналоговые преобразователи, широтно-импульсные модуляторы, контроллеры прерываний и др. Поэтому на базе такого функционально законченного изделия с включением минимального количества дополнительных элементов можно создавать достаточно сложные цифровые системы.

Богаевский Александр Борисович, 1955 г.р. Доктор технических наук, профессор кафедры автомобильной электроники Харьковского национального автомобильно-дорожного университета. Научная и преподавательская деятельность связана с бортовыми энергосберегающими микроконтроллерными системами управления транспортных средств.

978-3-659-59380-2

Восьмиразрядные микроконтроллеры

Богаевский, Осичев



Александр Богаевский
Александр Осичев

Восьмиразрядные микроконтроллеры. Архитектура и программирование

LAP LAMBERT Academic Publishing

Александр Богаевский
Александр Осичев

Восьмиразрядные микроконтроллеры. Архитектура и программирование

Александр Богаевский
Александр Осичев

Восьмиразрядные микроконтроллеры. Архитектура и программирование

LAP LAMBERT Academic Publishing

Impressum / Выходные данные
Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Библиографическая информация, изданная Немецкой Национальной Библиотекой. Немецкая Национальная Библиотека включает данную публикацию в Немецкий Книжный Каталог; с подробными библиографическими данными можно ознакомиться в Интернете по адресу http://dnb.d-nb.de.

Coverbild / Изображение на обложке предоставлено: www.ingimage.com

Verlag / Издатель:
LAP LAMBERT Academic Publishing
ist ein Imprint der / является торговой маркой
OmniScriptum GmbH & Co. KG
Heinrich-Böcking-Str. 6-8, 66121 Saarbrücken, Deutschland / Германия
Email / электронная почта: info@lap-publishing.com

Herstellung: siehe letzte Seite /
Напечатано: см. последнюю страницу
ISBN: 978-3-659-59380-2

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

В настоящее время созданию новых и совершенствованию уже известных встраиваемых микроконтроллеров уделяют внимание все большее количество зарубежных фирм - производителей электронных компонентов. Это связано расширением сфер применения этих изделий.

Встраиваемый однокристальный микроконтроллер (embedded microcontroller) представляет собой изготовленную на одном кристалле микропроцессорную систему, ориентированную на реализацию алгоритмов циф-рового управления различными объектами и процессами. Микроконтроллер содержит центральный процессор, внутреннюю постоянную и оперативную память, параллельные и последовательные порты ввода/вывода информации, периферийные устройства: таймеры, аналого-цифровые и цифро-аналоговые преобразователи, широтно-импульсные модуляторы, контроллеры преры-ваний и др. Поэтому на базе такого функционально законченного изделия с включением минимального количества дополнительных элементов можно создавать достаточно сложные цифровые системы.

По совокупности функциональных возможностей и технических характеристик микроконтроллеры ориентированы на реализацию управления раз-личными приборами и устройствами. Они используются как компоненты в системах управления технологическими процессами, на транспорте, в информационно−измерительных (мониторинговых) и диагностических системах.

Номенклатура микроконтроллеров, их значительные функциональные возможности, высокие технические характеристики, относительно низкая стоимость позволяют настоящим и потенциальным пользователям решать самые различные задачи.

Как показывает опыт, применение в СНГ микроконтроллеров

различного функционального назначения любой фирмы-изготовителя зависит напрямую от доступности и стоимости средств программирования и отладки и от наличия соответствующей подробной технической литературы на родном языке, понятной начинающему пользователю и несущей полезную информацию для подготовленного специалиста.

Высокая популярность микроконтроллеров, к сожалению, не привела по ряду общеизвестных причин к появлению на рынке учебных изданий как по техническим средствам, так и по программному обеспечению. Литература выходит очень малыми тиражами, не доступна студентам и преподавателям. В основном это переводы технических материалов фирм-изготовителей, полученные из интернетовских страниц.

Естественно, что в этой литературе отсутствует систематизированный материал учебного характера, ее терминология рассчитана на опытного специалиста.

Изучение микроконтроллеров студентами электромеханических специальностей сопряжено со следующими трудностями (общими практически для всех родственных специальностей вузов):

- недостаточная материальная база;

- ограниченное число часов аудиторных занятий;

- студенты до этого цикла лекций в подавляющем большинстве не знакомы с микроконтроллерами.

В этих условиях студентам необходимо предлагать информацию о “клас-сических”, концептуальных микроконтроллерах, изучив которые, они само-стоятельно смогут освоить другие типы микроконтроллеров.

Такими “классическими”, по мнению авторов пособия, являются структуры 8-разрядных универсальных микроконтроллеров фирмы Intel семейства MCS51 и его различных интерпретаций других фирм-изготовителей: Atmal, Philips, Dallas Semiconductor, Analog Devices и т.д. Это семейство имеет все функциональные узлы, присущие современным

микроконтроллерам, в том числе и с большей разрядностью. С использованием микроконтроллеров этого семейства уже более 10 лет на харьковских предприятиях выпускаются серийные изделия общепромышленного назначения, а также специализиро-ванные системы для транспортных средств.

2. НЕКОТОРЫЕ ОСНОВНЫЕ СВЕДЕНИЯ

2.1. Терминологический толковый словарь

Словарь терминов в микропроцессорной тематике является мощной поддержкой студентам на начальном этапе освоения материала. И это естествен-но, поскольку, в отличие от многих других сфер техники, микропроцессорные средства проходят этап бурного развития, во время которого появляются новые решения и понятия, они быстро входят в практику и в своей массе являются англоязычными. Кроме того, в силу этого отсутствует хорошо продуманная методология изложения материала курса и, зачастую, лектору приходится в первичном представлении материала использовать понятия, с которыми студентам предстоит познакомиться лишь через несколько занятий. Поэтому предлагаемый толковый словарь является тем минимумом знаний, без кото-рых невозможно качественно прочитать (воспринять) ни одну лекцию. Кроме того, он комментированно иллюстрирует объем изучаемой проблематики и является инструментом быстрого старта в изучении микропроцессора (МП).

Микропроцессор — программно управляемое устройство (микросхема) для обработки информации (сложение, вычитание, пересылка, ...) и управления внутрисхемной реализацией этого процесса. МП, как правило, не имеет развитых средств связи с внешними объектами.

Микроконтроллер — небольшая ЭВМ, выполняющая функции управления каким-либо объектом. Кроме того, в последнее время, понятие «микро-контроллер» стали использовать для обозначения однокристальных микро-ЭВМ.

Однокристальная микроЭВМ — микросхема, на кристалле которой реализован микропроцессор и блоки, необходимые для связи с объектом управления.

Контроллер — микросхема, выполняющая ту или иную функцию в ходе

рабочего процесса ЭВМ. Часть контроллеров не доступны для программиста и обслуживают саму ЭВМ, другая часть может использоваться программистом для того, чтобы задать какой-либо необходимый режим их работы, как правило, для связи с внешним оборудованием.

Память — функциональный блок ЭВМ для хранения, записи и считывания информации, представленной в основном в форме электрических напряжений, соответствующих записи чисел в двоичной системе счисления.

ПЗУ — микросхема постоянного запоминающего устройства, микросхемная реализация памяти. Программируется одноразово на заводе−производителе.

ППЗУ — однократно программируемое пользователем ПЗУ (микросхема).

СППЗУ — микросхема перепрограммируемого пользователем ПЗУ с ультрафиолетовым стиранием хранимой программы через специальное окно в корпусе микросхемы. Для записи и стирания используется специальное

устройство — программатор.

ЭСППЗУ — ПЗУ с электрически стираемой и записываемой программой. Запись и стирание производятся без использования дополнительных внешних устройств непосредственно на плате, где реализована микроЭВМ. Для хранения информации не требуется напряжение питания.

ОЗУ — микросхема (микросхемы), реализующие оперативное запоминающее устройство. Предназначено в основном для записи, считывания и временного хранения программ и данных пользователя. При исчезновении напряжения питания информация теряется. Однако время доступа к ОЗУ при записи значительно меньше, чем к ЭСППЗУ и поэтому микросхемы ОЗУ в ближайшей перспективе не будут заменены схемами ЭСППЗУ.

Флэш-память — внешняя для микропроцессора память, реализованная

микросхемно и выполняющая функции жесткого диска. От последнего отличается высокой устойчивостью к ударам и вибрации.

Кэш-память — вспомогательная оперативная память, не доступная для программиста. Размещается функционально между микропроцессором и ОЗУ. Служит для повышения быстродействия. В ней хранится и обновляется сдублированное из ОЗУ содержимое ячеек с наиболее часто употребляемыми в программе командами.

Таймер — микросхема для подсчета числа импульсов напряжения, реализации задержки импульсов на заданное время, выработки серии импульсов заданной частоты, определения длительности одиночного импульса и других сходных применений.

Интерфейс — совокупность устройств и правил, по которым реализуется обмен информацией. В более узком смысле интерфейс — это микросхема, выполняющая указанные функции.

Параллельный интерфейс — микросхема, реализующая пересылку информации в параллельном коде (электрическое представление двоичнокодированных чисел) по шине из 8, 12, 16 параллельных электрических проводников. Параллельный интерфейс обеспечивает ввод-вывод информации, например вывод из компьютера на принтер.

Последовательный интерфейс — микросхема, реализующая пересылку информации в последовательном коде по двухпроводной линии. Импульсы напряжения двоично-кодированных чисел передаются последовательно один за другим. Последовательный интерфейс служит для пересылки информации между удаленными компьютерами (по телефонным линиям) или между управляющим компьютером и удаленным объектом управления (более чем на 5 - 10 м). Как правило, последовательный интерфейс используется в паре с МОДЕМОМ — модулятором-демодулятором.

Обмен — обмен информацией, процедура пересылки информации в форме параллельного или последовательного кода.

Прерывание — временное прекращение исполнения текущей программы и переход к исполнению другой программы (как правило, подпрограммы) с последующим возвратом назад. Прерывание инициируется «неожиданно» для микроЭВМ по сигналу какого-либо внешнего устройства, например датчика аварийной ситуации, либо по сигналам внутреннего таймера для выполнения одной и той же подпрограммы через строго фиксированные промежутки времени.

Обмен по прерыванию — пересылка информации, которая производится в результате вхождения микроЭВМ в режим обработки поступившего сигнала (запроса) на прерывание.

Разряд — позиция для записи цифр числа в какой-либо системе счисления. Используются двоичная, восьмеричная, десятичная и шестнадцатеричная системы счисления. Команды, исполняемые ЭВМ, хранятся в памяти, пересылаются и исполняются только в двоичном представлении. Каждому разряду двоичного числа соответствует проводник (линия) шины, по которой передается это число в параллельном коде.

Бит — минимальная единица измерения объема передаваемой информации (не смысловое наполнение!). В записи двоичного числа каждый разряд соответствует одному биту и может быть заполнен либо единицей, либо нулем. Младший разряд числа называется младшим битом, старший разряд — старшим битом.

Байт — единица измерения объема информации. Двоичное число, записанное с помощью восьми разрядов (восьми битов), образует один байт информации и занимает один байт в памяти ЭВМ. С помощью восьми двоичных разрядов можно обозначить, закодировать (зашифровать) 2^8=256 различных комбинаций любых элементов, в том числе 256 целых десятичных чисел (от 0 до 255 включительно).

Кбайт — 2^{10}=1024 байт. Утвердился разговорный вариант этого термина: «килобайт».

Мбайт, Гбайт — мегабайт=2^{20} байт ≈1000 Кбайт; гигабайт=2^{30} байт≈1000000 Кбайт.

Данные — информация (числа), предназначенная для обработки в ЭВМ: процессов сложения, вычитания, пересылки и т.д.

Команда — запись в двоичной, восьмеричной, шестнадцатеричной системах счисления или в сокращенном буквенном виде, указывающая, как именно нужно обработать данные и где эти данные хранятся. Команда непосредственно исполняется в ЭВМ. Из команд составляют программу.

Адрес — порядковый номер ячейки памяти, в которой хранится команда, данные (или часть длинной записи команды, данных). Адрес может быть физическим — при сквозной нумерации всех ячеек от нуля и до последней, либо логическим — при более сложной организации памяти.

Операнд — часть команды, которая указывает адрес (или наименование регистра микропроцессора), где хранятся данные, предназначенные к обработке с помощью этой команды. Операндом могут быть также данные (число), непосредственно включенные программистом в запись команды для их обработки. Код операции, содержащийся в записи команды, указывает, «как надо обработать», а операнды указывают, «что надо обработать».

Регистры общего назначения (РОН) — элементы памяти в составе самого микропроцессора (но не ячейки памяти), используемые программистом для кратковременного хранения данных. Программа в них не хранится. Кроме термина РОН для их обозначения используются еще два: «сверхоперативное ОЗУ» (не корректный) и «рабочие регистры».

АЛУ — арифметико−логическое устройство, основной узел микропроцессора, который непосредственно реализует процедуры сложения, вы-читания и другие операции.

Машинное слово — один или несколько байт информации, которые обрабатываются микропроцессором за один раз, а не по частям. Размер (разрядность) машинного слова обычно равен разрядности АЛУ, разрядности

РОН или разрядности шины, по которой пересылаются команды и данные.

Языки программирования нижнего уровня — машинный код, машинный язык, ассемблер.

Машинный код — запись программы в виде команд, представленных в двоичном коде. Каждой записи команды в машинном коде взаимно однозначно соответствует команда в машинном языке или в ассемблере.

Машинный язык — запись программы с помощью восьмеричного или шестнадцатеричного представления команд. Основное достоинство по сравнению с машинным кодом — компактность записи и лучшая читаемость.

Ассемблер — язык программирования, в котором для записи команд используются мнемоники: сокращенные буквенные обозначения названий исполняемых операций. Кроме этого, ассемблером часто называют программу, которая автоматически переводит ассемблерный текст программы пользователя в машинные коды. Более полное название такой программы - ассемблирующая программа или программа-транслятор. Ассемблер — язык, жестко связанный с типом микропроцессора: у каждого микропроцессора свой набор машинных кодов команд (за редкими исключениями) и свой язык ассемблера. Вместе с тем, ряд мнемоник для разных микропроцессоров имеет одинаковую или сходную запись, что облегчает освоение новых микропроцессоров.

Эмулятор (симулятор) — программная модель микропроцессора, используемая для его изучения и отладки программ на персональном компьютере.

Порт — с точки зрения программиста, порт — это адрес внешнего устройства (параллельного интерфейса, таймера и др.), по которому можно произвести запись информации (например, для программирования микросхемы на определенный режим работы или для вывода информации из компьютера) или считывание. С аппаратурной точки зрения порт — это элемент памяти, временно хранящий пересылаемую информацию. С точки зрения пользователя

понятие порта часто отождествляется с каким-либо внешним устройством (ЦАП, АЦП, датчик, ...), подключенным к этому порту.

Раздельная адресация памяти и внешних устройств — такой способ организации памяти, когда ее ячейки пронумерованы от нуля до максимальной, при этом адреса внешних устройств тоже пронумерованы от нуля до макси-мального, часть численных значений этих адресов совпадает и в ассемблер вводятся специальные команды обращения к внешним устройствам, чтобы исключить двусмысленность такой адресации. Основная же масса команд ассемблера не может работать с внешними устройствами и предназначена для обслуживания ячеек памяти.

Совместная адресация ячеек памяти и внешних устройств — такой способ организации памяти, при котором ячейки памяти и внешние устройства включаются в единую сквозную нумерацию в пределах адресного пространства. Специальных команд ввода-вывода нет, но все команды ассемблера позволяют одинаково успешно обрабатывать содержимое как ячеек памяти, так и портов. Другое название такого способа — построение с отображением адресов внешних устройств на основную память.

Адресное пространство — максимальное количество ячеек памяти (в Кбайтах, Мбайтах, ...), которое можно пронумеровать (проадресовать) с помощью шины адреса, реализованной в микропроцессоре. Если шина адреса содержит 16 проводов (16 разрядов), то адресное пространство равно 2^{16} байт, то есть 64 Кбайта.

Разрядность микропроцессора — определяется в общем случае разрядностью двоичного числа, обрабатываемого на основных операциях в АЛУ за один раз. Как правило, разрядность микропроцессора соответствует разрядности регистров общего назначения и шины данных. В настоящее время используемые микропроцессоры имеют разрядности, равные 8, 16, 32. Разрядность шины адреса может быть иной, она не определяет величину разрядности микропроцессора, хотя обычно большей разрядности шины

данных соответствует большее адресное пространство.

Архитектура микроЭВМ — пожалуй, наиболее емкое (а значит, наименее конкретное) понятие микропроцессорной техники. Его полное определение, как правило, требует комментариев, поясняющих смысл. В тематическом словаре издательства Oxford Union Press указано, что «архитектура — это описание вычислительной системы на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд, средств пользовательского интерфейса, организация памяти и системы адресации, операций ввода−вывода и управления и т.д. В контексте разработки вычислительной системы и проектирования ее аппаратных средств термин “архитектура” используется для описания принципа действия, конфигурации взаимного соединения основных логических узлов ЭВМ. Обычно частью, а зачастую и основой такого описания служит подробная структурная или принципиальная схема конкретной машины».

Неймановская архитектура, Гарвардская архитектура — два типа архитектуры микропроцессорных средств, применяемых в настоящее время. Существует несколько отличительных признаков этих архитектур. Исторически первой была Неймановская архитектура, при которой программа пользователя

обрабатываемые данные хранились в одном ОЗУ. На этом принципе построен, например, классический микропроцессор серии I8080 (К580). Гарвардская архитектура появилась позднее в связи с попытками повысить быстродействие ЭВМ за счет изменения ее структурной схемы, а не только за счёт повышения тактовой частоты. Эта архитектура подразумевает наличие двух ОЗУ (и двух шин данных) - отдельно для исполняемой программы и отдельно для обрабатываемых данных с возможностью одновременного к ним обращения. Гарвардская архитектура стала основой построения ряда микроконтроллеров (например, I8051 или К1816ВЕ51). С ее применением

сегодня строятся наиболее производительные цифровые процессоры обработки сигналов.

ASCII-код — стандартизированная система обозначений различных символов (букв и цифр клавиатуры компьютера) для пересылки информации: American Standart Code for Information Interchange.

ISO7 bit, ISO8 bit — семи битный и восьми битный код, европейский стандарт кодирования символов для пересылки информации: International Organization for Standardization. В основном совпадает с ASCII кодом.

IEEE754 — стандарт на запись чисел с плавающей точкой (запятой). В рамках IEEE754 предложен ряд форматов: одинарной точности (диапазон представимых чисел $-10^{\pm 38}$ с 6 - 7 десятичными знаками), двойной точности (диапазон чисел $-10^{\pm 308}$ с 16 - 17 достоверными десятичными знаками). Стандарт IEEE754 реализован, например, в микропроцессоре I8087 (К1810ВМ87).

2.2. Области применения микропроцессорных средств и решаемые задачи

Приведем классификацию решаемых задач в пределах специальностей электромеханики, выделив лишь основные направления и их отличительные особенности.

2.2.1. Управление технологическим процессом без введения ЭВМ в контур регулирования

Управляющая программа хранится в ЭВМ и обеспечивает заданную последовательность выполнения технологических операций, заданную длительность во времени выполнения каждой из них, темп ускорения и замедления, обработку сигналов от технологических датчиков, а также датчиков аварийных ситуаций с соответствующим отключением. Исполнительным (управляемым) органом является электродвигатель с его

собственной системой управления (более или менее простой). ЭВМ не включается в контуры регулирования электропривода и в контур управления медленным технологическим процессом. К решению подобных задач хорошо адаптированы различные программируемые контроллеры, не обладающие высоким вычислительным быстродействием. Схема включения такой управляющей ЭВМ (рис.2.1) представлена на примере системы подчиненного регулирования электропривода постоянного тока.

На этой схеме РП, РС, РТ — регуляторы положения, скорости, тока; П — преобразователь тиристорный, транзисторный; ЭЧД, МЧД — электрическая и механическая части двигателя; М — механизм; ОСТ, ОСС, ОСП — обратные связи по току, скорости, положению (включая датчики), $ЭВМ_1$, $ЭВМ_2$, $ЭВМ_3$, $ЭВМ_4$ — возможные варианты применения ЭВМ, различающиеся по глубине внедрения в аналоговую часть системы управления.

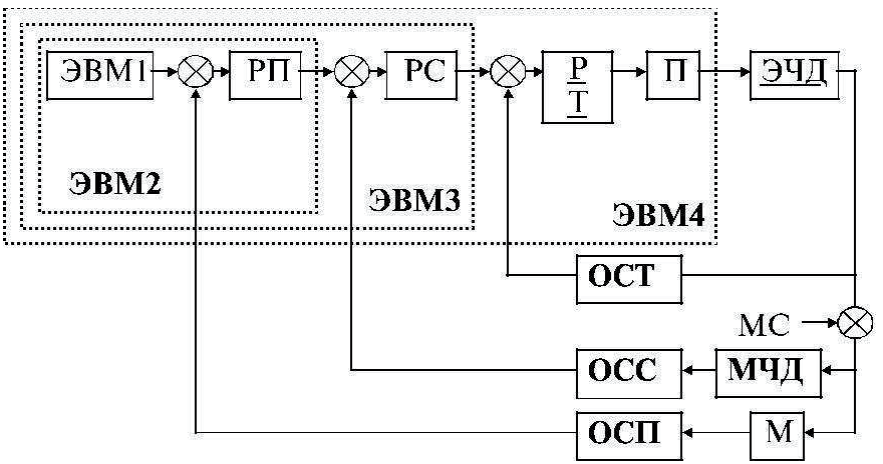


Рисунок 2.1 — Схема включения управляющей ЭВМ

$ЭВМ_1$ соответствует решению рассматриваемой задачи — управление технологическим процессом без введения микроконтроллера в контур регулирования.

Отметим, что в общем случае по такой же схеме включаются различные устройства автоматики, реализованные на микропроцессорной элементной базе и предназначенные для решения огромного количества различных задач автоматизации электромеханических объектов, не содержащих электропривода.

2.2.2. Управление технологическим процессом с включением ЭВМ в контур регулирования технологического параметра

Поставленной задаче соответствует применение $ЭВМ_2$. В данном случае реализуется цифровое управление положением механизма. Это характерно, например, для станков с числовым программным управлением. Такие ЭВМ требуют повышенного быстродействия вычислителя по сравнению с обычными программируемыми контроллерами (ПК). Однако, если попытаться указать конкретную группу вычислительных систем, подходящих для функционирования в качестве $ЭВМ_2$, то нужно иметь в виду следующее. Во-первых, возможности программируемых контроллеров быстро возрастают, во-вторых, появился класс промышленных компьютеров, обладающих очень высоким вычислительным быстродействием и, в-третьих, целый ряд технологических процессов, гораздо более медленных, чем обработка в станке, не требуют особого быстродействия от вычислительных средств. Например, регулирование давления воды с помощью асинхронных электроприводов или процессы в химической промышленности. Поэтому выбор типа управляющей $ЭВМ_2$ остается открытым и решается разработчиком альтернативно.

В качестве примера границ быстродействия регуляторов «медленного» технологического процесса можно привести технологический регулятор микропроцессорного асинхронного электропривода Simoreg фирмы SIEMENS. Этот регулятор обеспечивает программное задание параметров цифрового

регулятора технологического процесса с периодом повторения вычислений от $30*10^{-3}$ с до 6 с.

2.2.3. Управление технологическим процессом с частичным включением ЭВМ в контуры регулирования электропривода

Поставленной задаче соответствует $ЭВМ_3$ (рис.2.1). Такое решение встречается относительно редко. Так, в силу недостаточно высокого быстродействия вычислительных средств низкой стоимости в начале 90-х годов фирма SIEMENS выпускала электроприводы с аналоговым контуром тока и цифровым контуром скорости. В качестве вычислительных средств в таких системах используются электронные платы оригинальной разработки, а не типовые программируемые контроллеры. Их быстродействие должно быть достаточно велико при хорошей точности вычислений. Поэтому восьмиразрядные однокристальные микроЭВМ, сходные с МП I8051, используются в подобных системах, как правило, для решения частных задач при мультипроцессорной организации вычислительной платы.

2.2.4. Системы с прямым цифровым управлением двигателем постоянного тока

Рассматриваемые системы реализуются по схеме $ЭВМ_4$, представленной на рис. 2.1. Микропроцессорный вычислитель заменяет все аналоговые регуляторы и систему импульсно-фазового управления. Следует отметить, что термин «прямое цифровое управление» (ПЦУ) сложился естественным образом, но отличается в рамках таких специальностей, как автоматизация технологических процессов, электропривод и силовая электроника. Первые две специальности под введением ПЦУ подразумевают функциональную замену всех аналоговых регуляторов на цифровые. В силовой электронике под введением ПЦУ подразумевается исключение буферных усилительных

устройств, размещенных между выходом управляемой ЭВМ и входами преобразователя за счет применения соответствующих силовых вентилей независимо от этих различий, требования к быстродействию вычислителя предъявляются очень высокие. Он строится на базе 16-разрядных микропроцессорных средств в ходе разработки оригинальных плат фирмами−производителями электротехнической продукции. Программируемые контроллеры и промышленные компьютеры — не применяются (первые — медленные, а вторые — чересчур дороги и обладают избыточными ресурсами в силу универсальности). Кроме реализации регуляторов, на такие УВМ возлагаются самые разные задачи технологического характера, рассматриваемые в п. 2.2.1 — 2.2.3, и другие, например, запоминание и хранение всей информации от датчиков в предаварийной ситуации, автоматическая настройка регуляторов в результате изменения параметров двигателя (сопротивления, момента инерции...) перед его пуском.

2.2.5. Реализация систем управления с математически сложными законами управления

К этой группе можно отнести асинхронные и синхронные электроприводы с векторным управлением. Они широко выпускаются многими фирмами, такими как SIEMENS, ABB, Mitsubishi, Lenze, Shneider Electric, Baldor и другими. Их специализированные вычислители строятся на основе цифровых процессоров обработки сигналов, чаще всего фирмы Texas Instrument либо Analog Devices. Эти специализированные микросистемы позволяют реализовать сложные вычислительные алгоритмы с нужным быстродействием. Например, в электроприводе ACS6000 (мощность в диапазоне нескольких мегаватт) фирмы ABB интервал повторения основных вычислений составляет 25 микросекунд.

Следует заметить, что комплексные микропроцессорные электроприводы, в которых были бы реализованы другие законы регулирования, массово не производятся. Поэтому при необходимости повышения качества динамики механических устройств со сложными кинематическими схемами. Пока что следует разрабатывать управляющие программы для самостоятельно разрабатываемых управляющих плат либо использовать более дорогие, но высокопроизводительные промышленные компьютеры. К числу таких задач относится построение цифровых систем с наблюдателями состояния, динамической идентификацией параметров, адаптивное управление, системы с эталонной моделью прогнозным управлением и другие.

2.2.6. Управление экспериментом в науке и промышленных испытаниях

Рассматриваемая задача весьма сходна с управлением технологическим процессом. Вместе с тем, кроме собственно проведения эксперимента по его программе. Как правило, на ЭЦМ возлагаются наукоемкие задачи текущей обработки результатов эксперимента и его документирования: получение в конечном счете графиков и таблиц, характеризующих качества исследуемого оборудования. В итоге это приводит к разработке специализированных вычислительных управляющих комплексов с мощным математизированным программным обеспечением.

2.3. Классификация микропроцессорных средств по их типам (элементная база)

Микропроцессорные средства выпускаются давно, большим количеством фирм, с разными характеристиками, для разных применений, и поэтому необходимо провести их классификацию.

Среди множества традиционных фирм производителей выделим три:

Intel, Motorola и DEC. Микропроцессоры, совместимые с разработками Intel, выпускались в Советском Союзе и в СНГ. Поэтому наши инженеры хорошо знакомы с проблемами разработки аппаратных и программных средств на базе Intel-микропроцессоров. Схемы фирмы DEC также имели отечественные аналоги в сериях К1801, К1809, применявшихся в ЭВМ станочных электроприводов (*«Электроника-60»). Фирма Motorola — ведущая в США по производству сверхбольших интегральных схем (СБИС), однако для славянской территории ее продукция не имела аналогов и практически не была представлена в технической литературе до 1991 года.

Кроме этих фирм, на рынок соответствующей продукции вышли сравнительно недавно с новым видом СБИС такие производители цифровых процессоров обработки сигналов, как Texas Instrument и Analog Devices. Эти фирмы выделены как опорные. На примере их продукции удобно провести классификацию.

2.3.1. Однокристальные микропроцессоры (ОМП)

Определение ОМП соответствует представленному в п. 2.1. К ОМП относятся 8-, 16-, 32-разрядные МП с 16-, 20-, 32-разрядной шиной адреса и тактовыми частотами, лежащими в пределах от 2 МГц до тысяч мегагерц.

Примерами восьмиразрядных ОМП являются

- I8080 — восьмиразрядная шина данных, 16 разрядная шина адреса, 2,5 МГц тактовая частота, 243 команды ассемблера, каждая из которых выполняется за время от 4 до 18 тактов;
- MC6800 — продукт фирмы Motorola, появившийся несколько позднее I8080, который по характеристикам и вычислительным возможностям близок к I8080.

Они построены по неймановской архитектуре. В начале 80-х годов прошлого века на базе процессора К580 в Харькове (НИИ ХЭМЗ) был

спроектирован и выпущен микропроцессорный электропривод по системе подчиненного регулирования.

К 16-разрядным ОМП относятся:

- I8086 (К1810ВМ86) с 20-разрядной шиной адреса, тактовой частотой 5 МГц, временем выполнения одной команды сложения-вычитания 0,4 — 0,6 мкс, временем умножения-деления 30 — 40 мкс;
- MC68010 — эквивалентная по уровню схема фирмы Motorola.

ОМП I8086 имели развитие в виде I80186, I80286 (последние знакомы пользователю персональных компьютеров начала 90-х). На базе двух процессоров I80186 в 1985 году фирмой SIEMENS был разработан асинхронный электропривод с векторным управлением “TRAVSVECTOR”.

Особым видом ОМП является арифметический (математический) сопроцессор. Это ОМП для обработки чисел, представленных с плавающей запятой. Сопроцессором является I8087 (КР1810ВМ87). Время умножения-деления чисел по стандарту IEEE754 находится в пределах 30 — 40 мкс при тактовой частоте 5 МГц.

К 32-разрядным ОМП относятся:

- I8086, I80486 и более поздние ОМП фирмы Intel (кроме I80386, со встроенным сопроцессором);
- MC68020, MC68030, MC68040, MC68060;
- серия эквивалентных разработок фирмы Motorola.

ОМП изменялись в основном для производства персональных компьютеров, частично использовались для выпуска одноплатных управляющих микроконтроллеров. В последнее время они стали основой выпуска промышленных компьютеров, устойчивых к действиям агрессивной внешней среды (ударам, вибрации, влажности, загрязнению и т.д).

2.3.2. Однокристальные микроЭВМ (микроконтроллеры)

Определение ОМЭВМ дано в п. 2.1. В них значительно чаще, чем в ранних сериях однокристальных микропроцессоров, используется гарвардская архитектура. Предназначены ОМЭВМ для решения относительно несложных задач автоматизации различного рода технологических процессов, как правило, связанных с необходимостью построения компактных управляющих локальных систем с малым энергопотреблением.

В микросхему ЭВМ ранних моделей в качестве дополнения к микропроцессору входили таймер, параллельный и последовательный интерфейсы, система обработки прерываний, ОЗУ и разные виды постоянных запоминающих устройств. Несколько позже появились встроенные АЦП, блоки ШИМ−модуляции, индикаторы трехфазного * напряжения и некоторые другие блоки.

К 8-разрядным ОМЭВМ относятся:

- I8031, I8039, I8048, I8051, построенные с использованием гарвардской архитектуры. Их тактовая частота — до 12 МГц, время исполнения одной команды - 1 — 2 мкс. Модель I8051 имеет команду умножения и деления однобайтного положительного числа на однотипное (время исполнения 4 мкс). Емкость ОЗУ мала — 256 Байт, встроенного ПЗУ — от 1 до 7 КБайт, но его адресное пространство 64 КБайт и можно наращивать внешние микросхемы памяти.
- MC68HC05, MC68HC11 — ОМЭВМ фирмы Motorola. Они имеют более развитые возможности по сравнению с I8051. В составе MC68HC11 восьмиразрядный восьмиканальный АЦП, более мощный по функциональным возможностям таймер, команды умножения и деления знаковых и беззнаковых чисел с двухбайтным представлением произведения (либо делителя), система прерываний в реальном времени и др.

К 16-разрядным ОМЭВМ относятся:

- I8096, I80196, I80296 (последняя модель). Их тактовая частота 20 — 30 МГц и они имеют полный набор вышеуказанных функциональных блоков. Однако ЦАП отсутствует и вместо него в микросхеме применен блок ШИМ с установкой внешнего фильтра. На базе схемы I8096 английская фирма SSD выпустила в 1987 году комплектный электропривод.
- MC68HC16 — модель, в которую кроме всех вышеуказанных блоков введены 10-разрядный АЧП, специальные команды для реализации цифровых фильтров (МАС — умножение с накоплением). В ассемблер добавлены некоторые команды языков высокого уровня (циклов). Вместе с тем, в этих ОМЭВМ обработка чисел с плавающей запятой не предусмотрена.

К 32-разрядным ОМЭВМ можно отнести модели MC68330, MC68331, MC68332, которые содержат в качестве вычислительного ядра микропроцессор MC68020. Это весьма привлекательное решение для специалистов, имеющих программные наработки в области персональных компьютеров. В ассемблер введена команда LookupTable — задание просматривает таблицы с линейной интерполяцией между ее точками, блок чередования трехфазного опорного напряжения, но удалены некоторые данные для разработки блоки, традиционные для ОМЭВМ.

Фирма Intel не выпустила 32-разрядный ОМЭВМ и прекратила разработки в области этих СБИС.

2.3.3. Процессоры цифровой обработки сигналов

Процессоры цифровой обработки сигналов (ПЦОС) появились достаточно давно как отдельная ветвь микропроцессорных средств. В Советском Союзе были выпущены две серии КР1813ВЕ1 (8-разрядный Intel−совместимый) и КР1827ВЕ2 (16-разрядный). Современные ПЦОС

предназначены для решения достаточно узкого класса технических задач и поэтому имеют экстремально высокие характеристики в своей области применения. Они выпускаются рядом фирм. В основном известны Texas Instrument (адаптированы непосредственно к решению задач фильтрации сигналов и управления электроприводами), Analog Devices (обработка звука и изображений) и Motorola. Высокие характеристики достигаются за счет удачного сочетания решений на разных этапах создания схем ЦПОС. Они используют гарвардскую архитектуру, имеют все блоки связи с внешними объектами, характерные для последних моделей ОМЭВМ, плюс цифро−аналоговый преобразователь, встроенные умножители и сопроцессор для обработки чисел с плавающей запятой. В их ассемблеры введены команды языков высокого уровня, команды цифровой фильтрации и обработки матричных данных. Кроме того, ряд команд позволяет производить одновременно обработку аналоговой информации и обработку цифровой информации. Так, ЦПОС типа DSP56000, DSP96000 фирмы Motorola выполняет умножение матриц размером 3×3 за 3 мкс, расчет цифрового фильтра восьмого порядка — за 3 мкс, быстрое преобразование Фурье на 1024 * - за 3 мс. Близкие характеристики имеет ЦПОС TMS320 фирмы Texas Instrument. В настоящее время он широко применяется для построения асинхронных электроприводов с векторным управлением.

2.3.4. Секционируемые микропроцессоры (СП)

СП появились, когда было необходимо реализовать вычислитель большой разрядности, а существующие технологии не позволяли изготовить его в виде одной микросхемы. Поэтому были выпущены 2-разрядные секции, из которых разработчик ЭЦМ мог набрать компьютер с нужной длиной разрядной сетки.

В области задач управления Intel-совместимая известна серия К589 —

двухразрядная секция с тактовой частотой 10 МГц. На ее базе в середине 80-х запорожскими специалистами был разработан электропривод постоянного тока по системе подчиненного регулирования. Однако дальнейшего развития этот вид микропроцессоров не получил.

2.3.5. PIC-контроллеры

Периферийные интерфейсные контроллеры предназначены для построения массово применяемых несложных систем автоматики. Они появились недавно и являются по существу вариантом модели ОМЭВМ, упрощенной настолько, насколько это возможно для снижения их цены до 2 — 3 у.е.

2.3.6. Специализированные процессоры

Имеются реализации процессоров, направленные на решения других конкретных задач управления. К их числу относятся фаззи-контроллеры, нейро-контроллеры, контроллеры БПФ и другие.

Для высокопроизводительных ЭВМ, допускающих распараллеливание вычислительной задачи, разработаны транспьютеры.

Следует заметить, что повышение требований к возможностям микропроцессоров сопровождалось разработками со все более сложными схемотехническими решениями и наращиванием количества команд ассемблеров до нескольких тысяч. Такие процессоры относились к группе CISC с полным набором команд (Complex Instruction Set Command).

На определенном этапе разработок наращивание системы команд оказалось непродуктивным. После статистического анализа «популярности» применения команд в программах их число было сокращено и соответствующие упрощения в схеме управления были выполнены. Так появились RISC−процессоры с сокращенным набором команд (Reduced

Instruction Set Command), более эффективно работающие. Однако такое распределение на CISC−RISC не настолько четкое, чтобы его можно было привести к пунктам 2.3.1 — 2.3.6 классификации по назначению и типам. И вообще следует отметить, что удачно найденные решения в ходе разработки одного типа процессоров быстро распространяются на разработки других типов. Так, новые решения при создании ЦПОС были использованы в разработках МП для персональных компьютеров уровня Pentium.

3. ОБЩИЕ СВЕДЕНИЯ ОБ ОДНОКРИСТАЛЬНЫХ МИКРО ЭВМ СЕМЕЙСТВА МК51

Восьмиразрядные высокопроизводительные однокристальные микро-ЭВМ (ОМЭВМ) семейства МК51 выполнены по высококачественной n-МОП — технологии (серия 1816) и КМОП — технологии (серия 1830).

Семейство МК51 включает большое количество модификаций ОМЭВМ. Они имеют идентичные основные характеристики, а отличаются между собой реализацией памяти программ и мощностью потребления.

Рассмотрим особенности ОМЭВМ по их типам:

- ОМЭВМ 8051 и 80С51 (аналоги К1816 ВЕ 51 и К1830 ВЕ 51) содержат масочно— программируемое (в процессе изготовления кристалла) ПЗУ памяти программ емкостью 4 Кбайт и рассчитаны на применение в массовой продукции. За счет использования дополнительных (внешних) микросхем памяти общий объем памяти программ может быть расширен до 64 Кбайт.
- ОМЭВМ 8751 (аналог 1816 ВЕ751) — модель, содержащая ППЗУ программ емкостью 4 Кбайт со стиранием ультрафиолетовым излучением и удобная на этапе разработки системы при отладке программ, а также при производстве небольших партий устройств или при создании систем, требующих в процессе эксплуатации периодической подстройки. За счет использования внешних ИС памяти общий объем памяти программ может быть расширен до 64 Кбайт.
- ОМЭВМ 8031 и 80С31 (аналоги 1816 ВЕ 31 и 1830 ВЕ 31) не содержат встроенной памяти программ, однако могут использовать до 64 Кбайт внешней постоянной или перепрограммируемой памяти программ и эффективно использоваться в системах, требующих существенно большего по объему (чем 4 К на кристалле) ПЗУ программной памяти.

Вышеприведенные ОМЭВМ содержат все узлы, необходимые для автономной работы:

- центральный 8-— разрядный процессор;
- память программ 4 Кбайт;
- память данных 128 байт;
- 4 восьмиразрядных программируемых канала ввода/вывода;
- два 16— разрядных многорежимных таймера/счетчика;
- систему прерываний с пятью векторами и двумя уровнями;
- последовательный интерфейс;
- тактовый генератор.
- 32 регистра общего назначения (РОН);
- 128 определяемых пользователем программно - управляемых флагов;
- набор регистров специальных функций (SFRam).

Система команд ОМЭВМ содержит 111 базовых команд с форматом 1, 2 или 3 байта.

ОМЭВМ имеет:

РОН и определяемые пользователем программно — управляемые флаги расположены в адресном пространстве внутреннего ОЗУ данных. Регистры специальных функций (SFRam) занимают объем 128 байт. Структурная схема базового микроконтроллера 8051 представлена на рис. 3.1. Пространство памяти микроконтроллера 8051 (80C51) представлено на рис. 3.2.

Пространство внутренней памяти данных базового микроконтроллера 8051 представлено на рис. 3.3.

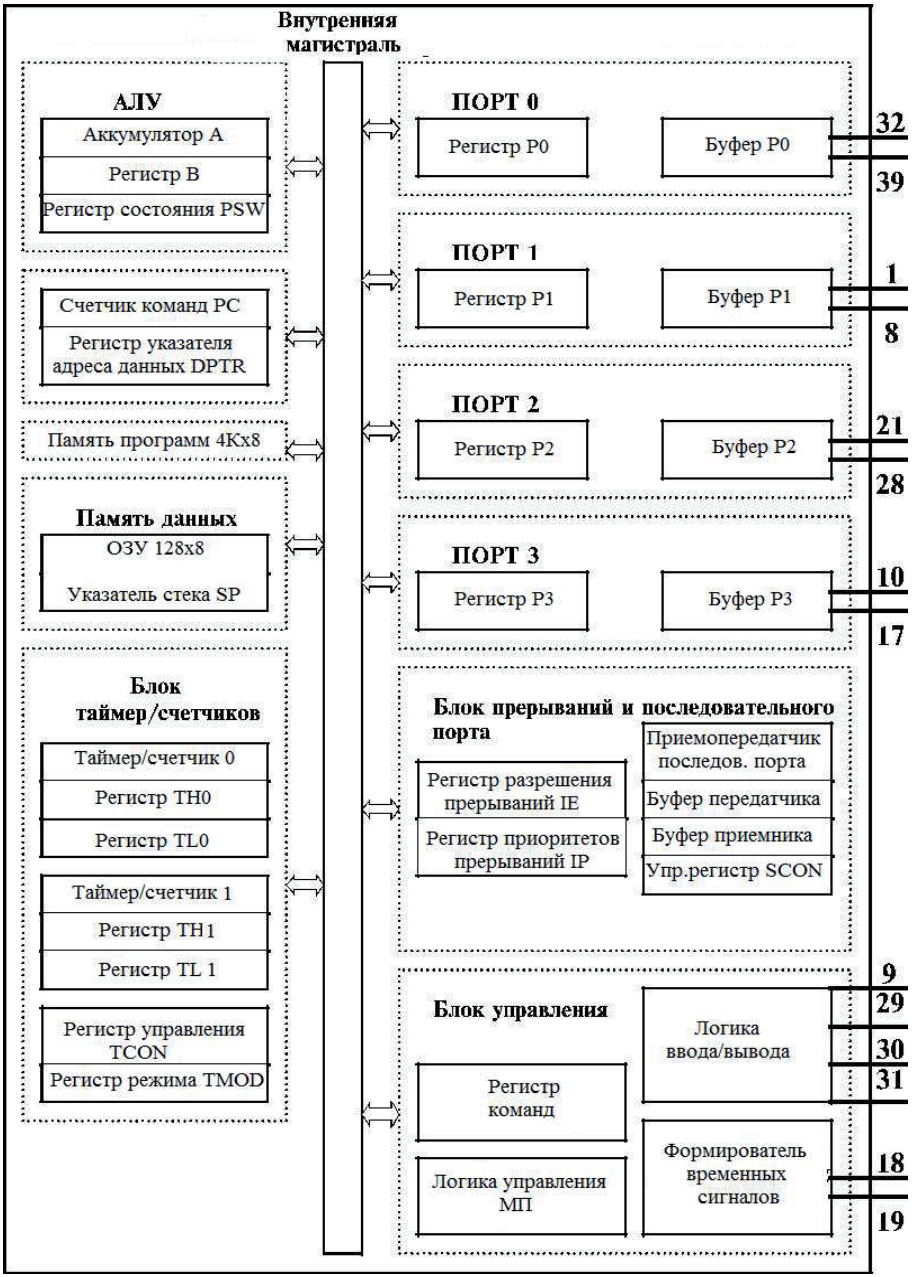


Рисунок 3.1 – Структурная схема микроконтроллера 8051

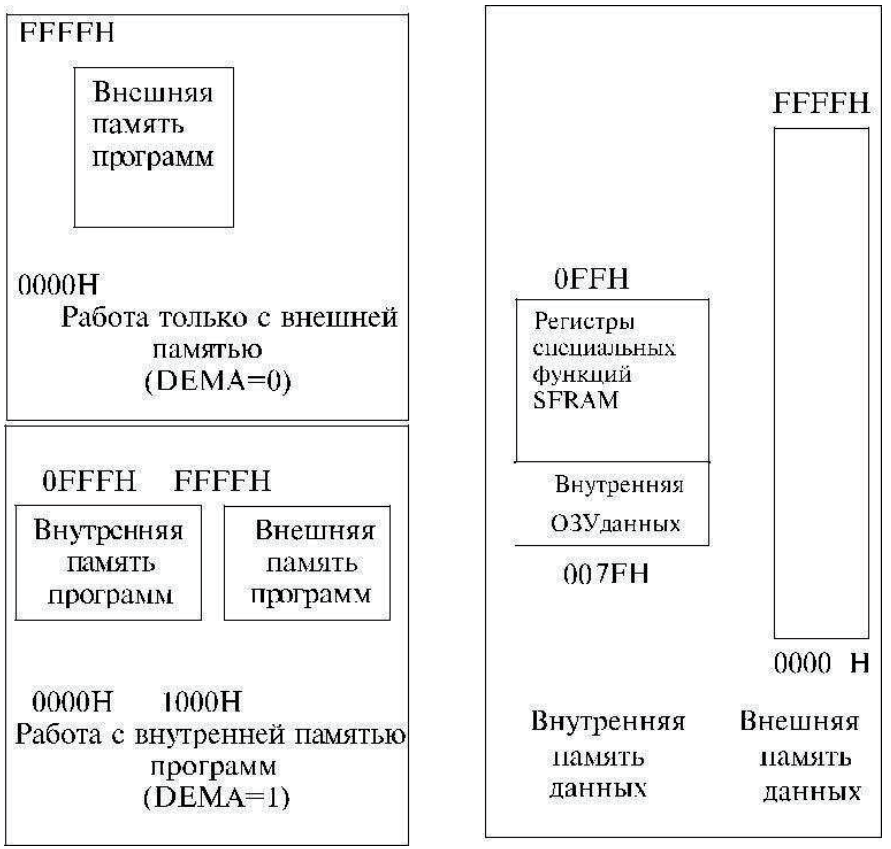


Рисунок 3.2 — Пространство памяти базового микроконтроллера 8051

В число регистров специальных функций семейства 8051 входят следующие регистры:

- аккумулятор ACC — регистр аккумулятора;
- регистр B — используется во время операций деления и умножения;
- регистр состояния программы PSW — содержит информацию о состоянии программы. Является эквивалентом регистра признаков (флагов) широко

известного микропроцессора 8080 (аналог К580);

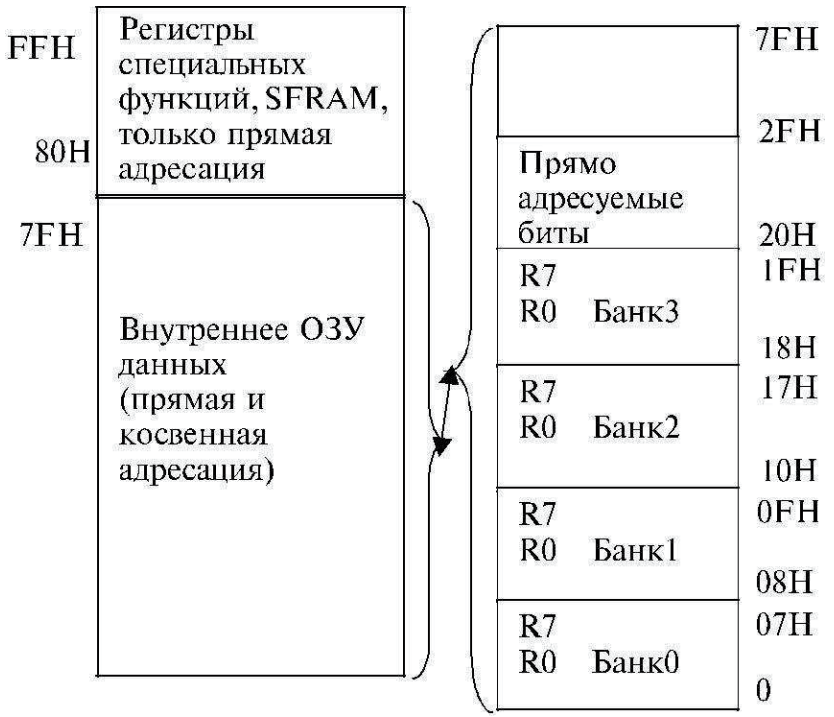


Рисунок 3.3 — Пространство внутренней памяти данных

- указатель стека SP — 8-битовый регистр, обеспечивающий функционирование стековой памяти. Его содержимое изменяется при выполнении команд работы со стековой памятью или команд вызова подпрограмм. При начальном сбросе устанавливается в 07H, а область стека начинается с адреса 08H;
- указатель данных DPTR — состоит из двух 8-разрядных регистров: регистра старшего байта DPH и регистра младшего байта DPL. DPTR содержит 16−разрядный адрес при обращении к внешней памяти. Он может использоваться как 16−разрядный регистр или как два независимых

8-разрядных регистра;

- порт 0 — порт 3. Регистрами специальных функций P0, P1, P2 и P3 являются регистры-«защелки» соответственно портов P0, P1, P2 и P3;
- буфер последовательного порта SBUF — представляет собой два отдельных регистра: буфер передатчика и буфер приемника. Когда информация в ходе выполнения соответствующей команды записывается в SBUF, то она поступает в буфер передатчика, причем запись байта информации в SBUF автоматически инициирует его передачу через последовательный порт. Когда информация читается из SBUF соответствующей командой, то она выбирается из буфера приемника;
- регистры таймера. Регистровые пары TH0,TL0 и TH1,TL1 образуют 16-разрядные счетные регистры соответственно таймера/счетчика 0 и таймера/счетчика 1. Счетчики работают на сложение, в отличие от счетчиков извест-ного таймера К580 ВИ53, работающих на вычитание;
- регистры управления. Регистры IP,IE,TMOD,TCON,SCON и PCON содержат биты управления и биты состояния системы прерываний, таймеров/счетчиков и последовательного порта.

Архитектура семейства МК51 имеет ряд замечательных свойств, которые делают контроллеры этого семейства, а также контроллеры, являющиеся дальнейшим развитием этого семейства, особенно удобными для практических применений. Это прежде всего достаточно полная система команд с широким выбором удобных режимов адресации, которая обеспечивает побайтовую и побитную адресацию, работу в двоичной и двоично-десятичной арифметике. Арифметико-логическое устройство (АЛУ) процессора может наряду с выполнением операций над 8-разрядными типами данных манипулировать с одноразрядными данными. В силу этого свойства микроконтроллеры семей-ства МК51 иногда называют битовыми процессорами.

4. СТРУКТУРНАЯ ОРГАНИЗАЦИЯ КОНТРОЛЛЕРА 8051

Структурно контроллер состоит из следующих основных функциональных узлов, соединенных между собой по внутренним магистралям адреса, данных и управления:

1. блок управления и синхронизации;
2. арифметико-логическое устройство (АЛУ);
3. блок таймеров/счетчиков;
4. блок последовательного интерфейса и прерываний;
5. программный счетчик;
6. память данных и память программ.

Двухсторонний обмен информацией между функциональными блоками осуществляется с помощью внутренней 8-разрядной магистрали данных.

4.1. Блок управления. Синхронизация микроконтроллера. Регистр PCON. Режимы уменьшенного энергопотребления

Блок управления предназначен для выработки синхронизирующих и управляющих сигналов, обеспечивающих координацию совместной работы блоков микро ЭВМ во всех допустимых режимах ее работы.

В состав блока входят:

- устройство выработки временных интервалов;
- логика ввода/вывода;
- регистр команд;
- регистр управления энергопотреблением PCON$
- дешифратор команд;
- ПЛМ (программируемая логическая матрица) и логика управления ЭВМ.

Рассмотрим эти узлы.

Устройство выработки временных интервалов предназначено для формирования и выдачи внутренних синхросигналов фаз, тактов и циклов.

Количество машинных циклов определяет продолжительность выполнения команд. Практически все команды ЭВМ выполняются за 1 или 2 машинных цикла, кроме команд умножения MUL A,B и деления DIV A,B, продолжи-тельность выполнения которых составляет 4 машинных цикла.

Машинный цикл имеет фиксированную длительность, которая равна 12 периодам тактовой частоты генератора. Тактовый сигнал вырабатывается в микроЭВМ тактовым генератором при подключении к выводам 18 и 19 микросхемы кварцевого резонатора OSC по предлагаемой на рис. 4.1 схеме, либо внешним синхросигналом.

Машинный цикл, как показано на рис. 4.2, имеет шесть состояний, обозначенных S1 . . . S6, каждое из которых соответствует такту и в свою очередь состоит из двух временных интервалов: P1 и P2.

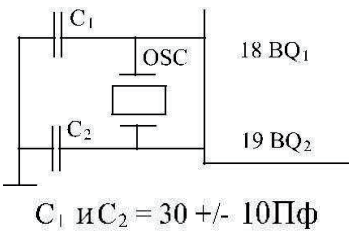


Рисунок 4.1 — Подключение внешних элементов синхронизации

Формирование машинных циклов приведено на рис. 4.2.

Дважды за один цикл формируется сигнал ALE — сигнал разрешения фиксации адреса.

Логика ввода/вывода предназначена для приема и выдачи сигналов, обеспечивающих обмен информацией ОЭВМ с внешними устройствами через порты ввода/вывода P0 . . . P3.

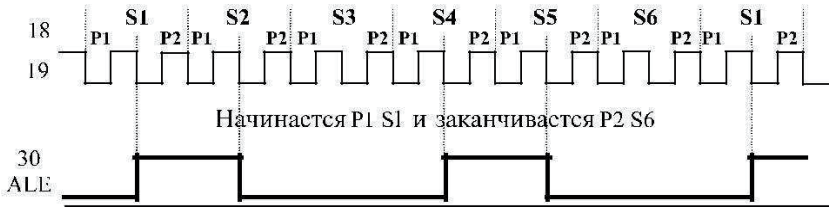


Рисунок 4.2 — Формирование машинных циклов

Регистр команд предназначен для записи и хранения 8 разрядного кода операции выполняемой команды, который далее с помощью дешифратора команд преобразуется в 24 — разрядный код для ПЛМ. С помощью ПЛМ вырабатывается набор микроопераций в соответствии с микропрограммой выполнения команды. Регистр команд программно не доступен.

Регистр управления энергопотреблением PCON — 8-разрядный регистр, управляет режимами энергопотребления. Назначение разрядов (битов) регистра PCON, их нумерация от старшего к младшему и их общепринятое обозначение представлены в табл. 4.1.

Таблица 4.1 — Назначение битов регистра PCON

№ бита	Наименование	Назначение	Примечание
7	SMOD	Бит удвоения скорости передачи: если установить в 1, то скорость передачи удваивается	При работе последовательного порта
6	---	Резервный	
5	---	Резервный	
4	---	Резервный	
3	GF1	Флаг общего назначения	
2	GF0	Флаг общего назначения	
1	PD	Бит включения режима микропотребления: если в 1, то режим микропотребления	Если в PD и IDL — 1, то приоритет PD
0	IDL	Бит режима х.х.	

Режим холостого хода. В этом режиме блокируются функциональные

узлы центрального процессора (CPU), что уменьшает энергопотребление. Сохраняются состояния указателя стека, программного счетчика, регистра PSW, аккумулятора и других регистров, а также внутреннего ОЗУ данных. Из режима х.х. можно выйти двумя способами:

- посредством активизации любого разрешенного прерывания;
- выполнив аппаратный сброс по входу RST.

Режим микропотребления. В этом режиме задающий генератор выключается, тем самым прекращается работа всех узлов микроконтроллера и сохраняется только содержимое ОЗУ. Единственным способом выхода из этого режима является аппаратный сброс по входу RST.

Логика управления ОЭВМ в зависимости от режима работы ОЭВМ вырабатывает необходимый набор управляющих сигналов.

4.2.Арифметико-логическое устройство (АЛУ). Регистр PSW

АЛУ представляет собой параллельное 8-разрядное устройство, обеспечивающее выполнение арифметических и логических операций, а также операций сдвига, обнуления, установки и т.п.

АЛУ состоит из буферного регистра аккумулятора и регистра временного хранения, ПЗУ констант, сумматора, дополнительного регистра (регистр В), аккумулятора и регистра состояния программы PSW.

Регистр аккумулятора и регистр временного хранения — 8-разрядные регистры, предназначенные для приема и хранения операндов на время выполнения операций над ними. Программно не доступны.

ПЗУ констант обеспечивает выработку корректирующего кода при двоично-десятичном представлении данных, кода маски при битовых операциях и кода констант.

Сумматор — схема комбинационного типа, предназначенная для выполнения арифметических операций сложения, вычитания, логических

операций, умножения, неравнозначности и тождественности.

Регистр B — 8-разрядный регистр, участвующий в операциях деления и умножения. Программно доступен.

Аккумулятор — 8-разрядный регистр, предназначенный для приема и хранения результата, полученного при выполнении арифметических операций, операций пересылки и других. Программно доступен.

Регистр состояния программы (PSW) — предназначен для хранения информации о состоянии АЛУ при выполнении программы.

Назначение разрядов PSW (начиная со старшего разряда).

7. CY(C) — флаг переноса. Может устанавливаться программным и аппаратным образом. Этот флаг может быть программно прочитан.

Аппаратными средствами CY устанавливается автоматически в ходе исполнения команды в случае, если в старшем разряде результата возникает заем или перенос. Этот перенос или заем фиксируется фактически в 9-м разряде аккумулятора CY, когда результат операции превышает 255_{10}. При выполнении операций умножения и деления флаг CY сбрасывается.

4	указатели	0	0	1	1
3	банка РОН	0	1	0	1
		Банк 0	Банк 1	Банк 2	Банк 3
	адреса банка	0-7h	8h-0fh	10-17h	18-1fh

6. AC — флаг дополнительного переноса — программно доступен для записи и чтения (0 или 1). Аппаратно устанавливается/сбрасывается во время выполнения инструкций сложения или вычитания для указания переноса или заёма в бите 3 при образовании младшего полубайта результата.

5. Флаг F0 — флаг состояния, определяемый пользователем. Доступен программно по чтению и записи.

4,3. RS1, RS0 — указатели банков рабочих регистров. Программно доступны по записи и чтению во внутреннем ОЗУ

2. Флаг переполнения OV (overflow). Программно доступен по чтению и записи.

Бит OV устанавливается аппаратно, если результат арифметической операции не укладывается в семи разрядах и восьмой разряд уже нельзя интерпретировать как знаковый. При выполнении операции деления флаг OV аппаратно сбрасывается, а при делении на 0 этот флаг устанавливается автоматически. При умножении OV устанавливается, если результат больше 255.

1. Резервный разряд.

0. P — бит четности. Программно доступен только для чтения. Аппаратно сбрасывается/устанавливается в каждом цикле инструкций для указания четного/нечетного количества разрядов аккумулятора, находящихся в состоянии 1. В 9-разрядном слове , состоящем из 8 разрядов аккумулятора и бита P, всегда содержится четное число единичных битов. Если все биты аккумулятора содержат 0, то флаг P примет нулевое состояние.

4.3. Блок таймеров/счетчиков

Таймеры/счетчики (T/C) предназначены для подсчета внешних событий, для получения программно-управляемых временных задержек и выполнения времязадающих функций микроконтроллера.

В состав блока таймеров/счетчиков входят:

0. два 16-разрядных регистра T/C 0 и T/C 1;
1. восьмиразрядный регистр режимов T/C (TMOD);
3. восьмиразрядный регистр управления (TCON);
4. схема инкремента;
5. схема фиксации состояний входов INT0, INT1, T0,T1;
6. схема управления флагами;
7. логика управления T/C.

Рассмотрим их назначение.

16-разрядные регистры T/C

16-разрядные регистры T/C выполняют функцию хранения содержимого счета. Каждый из них состоит из пары восьмиразрядных регистров TH0 и TL0, TH1 и TL1 соответственно, причем TH1 и TH0 — это старшие байты (timer high), TL1 и TL0 — младшие байты (timer low). Каждый из этих четырех регистров имеет свой адрес и может быть использован как РОН, если T/C не используется по своему назначению. Код начального счета (числа, от которого начинается счет) заносится в регистры таймер/счетчиков программно. В процессе счета содержимое регистров T/C инкрементируется (т.е. увеличивается на 1). Признаком окончания счета, как правило, является переход содержимого регистра T/C из состояния «все 1» в состояние «все 0», т.е. переполнение. Все регистры TH0, TH1, TL0, TL1 программно доступны по чтению, и, при необходимости, контроль достижения требуемой величины счета может вы-полняться программно.

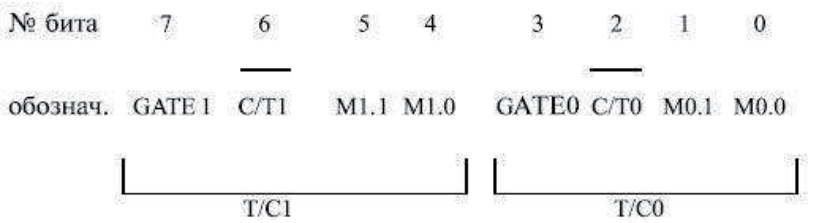
Регистр режимов T/C (TMOD)

Регистр режимов T/C (TMOD) предназначен для хранения и приема кода, определяющего:

1. один из четырех возможных режимов работы каждого T/C;
2. работу в качестве таймеров или счетчиков;
3. управление T/C от внешнего вывода.

Назначение отдельных битов регистра TMOD следующее:

Биты с 0 по 3 определяют режим работы T/C0, биты с 4 по 7 определяют режим работы T/C1.



Биты 0 – 1 и 4 – 5 определяют один из 4-х режимов работы, отдельно для T/C1 и T/C0:

M1	M0	Режим
0	0	0
0	1	1
1	0	2
1	1	3

Подробное описание режимов работы приведено ниже.

Биты 2 и 6 определяют работу каждого из T/C в качестве:

таймера C/T 0, C/T1= 0;

счетчика C/T0, C/T1 = 1.

Биты 3 и 7 разрешают управлять таймером/счетчиком от внешнего вывода (INT0 для T/C0 и INT1 для T/C1):

GATE 0, GATE 1= 0 — управление запрещено;

GATE 0, GATE 1= 1 — управление разрешено.

При работе в качестве таймера содержимое регистра T/C инкрементируется в каждом машинном цикле, т.е. T/C является счетчиком машинных циклов микроконтроллера.

Так как машинный цикл состоит из 12 периодов частоты синхронизации, то частота счета в данном случае равна $f_c/12$, т.е. 1 Мгц при t_c =12 Мгц.

При работе в качестве счетчика внешних событий содержимое регистра T/C инкрементируется в ответ на переход из 1 в 0 сигнала на счетном входе микроконтроллера (вывод T0 для T/C 0 и вывод T1 для T/C 1).

Чтобы уровень сигнала на счетном входе был гарантированно зафиксирован, он должен оставаться неизменным, как минимум, один машинный цикл.

Регистр управления TCON

Регистр управления TCON предназначен для приема и хранения кода управляющего слова, которое регламентирует как управление таймерами/ счетчиками, так и работу системы прерываний.

№ бита	7	6	5	4	3	2	1	0
обознач.	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Биты с 4 по 7 предназначены для управления таймерами/счетиками, Биты с 0 по 3 предназначены для управления системой прерываний. Биты 5 и 7 (TF0 и TF1) — флаги переполнения, которые устанавливаются аппаратно при переполнении таймер/счетчиков (переход T/C из состояния «все 1» в состояние «все 0»). Если при этом разрешено прерывание от соответствующих T/C, то происходит одновременно с установкой флага TF и пре рывание. Флаги TF и TF1 сбрасываются аппаратно при передаче управления программе обработки соответствующего прерывания.

Флаги TF0 и TF1 программно доступны и могут быть установлены/сброшены программой.

Биты 4 и 6 (TR0 и TR1) предназначены для включения таймеров/счетчиков, отдельно для T/C 0 и T/C 1:

TR0,TR1 = 0, T/C 0 и T/C 1 — выключены;

TR0,TR1 = 1, T/C 0 и T/C 1 — включены.

Биты 1 и 3 флаги IE 1 и IE 0 устанавливаются аппаратно от внешних прерываний (соответственно входы INT0, INT1) или программно и

инициируют вызов программы обработки соответствующих прерываний.

Сброс этих флагов выполняется аппаратно при обслуживании прерывания в том случае, если прерывание было вызвано фронтом сигнала. Если же прерывание вызвано уровнем сигнала на вход (INT0, INT1), то сброс флага IE должна выполнять программа обслуживания соответствующего прерывания, воздействуя на источник прерывания для снятия им запроса.

Биты 0 и 2 (IT 0 и IT 1) − биты определяющие вид прерывания по входам INT0, INT1:

IT 0, IT 1= 0 — прерывание по низкому уровню (по логическому 0);

IT 0, IT 1= 1 — прерывание по фронту (переход из 1 в 0).

Биты, доступны по чтению, они устанавливаются и сбрасываются программно. Регистр TCON доступен программно для чтения и записи как побайтно, так и побитно.

Схема инкремента.

Схема инкремента предназначена:

- для увеличения на 1 в каждом машинном цикле содержимого регистров T/C0 и T/C1, для которых установлен режим таймера и счет разрешен;
- для увеличения на 1 соответствующего T/C0 и T/C1, для которых установлен в TMOD режим счетчика, разрешен счет, и на соответствующем входе микро-контроллера (T0 для T/C 0 и T1 для T/C 1) зафиксирован счетный импульс.

Схема фиксации INT0, INT1, T0, T1 представляет собой 4 триггера. В каждом машинном цикле в них запоминается информация с выводов INT0, INT1, T0, T1 микроконтроллера.

Схема управления флагами вырабатывает и снимает флаги переполнения T/C и флаги запросов внешних прерываний.

Логика управления T/C синхронизирует работу регистров T/C0 и T/C1 в соответствии с запрограммированными режимами работы и синхронизирует

работу блока Т/С с работой микроконтроллера.

Режимы работы Т/С

Режим работы 0

Т/С в режиме 0 представляет собой устройство на основе 13-разрядного регистра. 13-разрядный регистр состоит для Т/С 0 из 8 разрядов регистра TH0 и 5 младших разрядов регистра TL0, а для Т/С 1 — из 8 разрядов TH1 и 5 младших разрядов регистра TL 1.

В данном случае идет речь о 8-разрядном Т/С с предделителем на 32 на регистре TL. Регистры TL1 и TL0 являются программно доступными, однако следует помнить, что значащими в режиме 0 являются лишь 5 младших битов регистров TL1 и TL0.

При переполнении Т/С (переход из состояния «все 1» в состояние “все 0”) устанавливается флаг TF1 для Т/С 1 или TF0 для Т/С 0 в регистре TCON.

Биты C/T0 и C/T1 регистра TMOD определяют работу таймеров/счетчиков или в качестве таймера (C/T=0), или в качестве счетчика внешних событий (C/T=1). Счет начинается при установке бита TR регистра TCON в состояние логической 1. При необходимости управления счетом извне бит GATE регистра TMOD устанавливается в состояние логической 1. Тогда при TR = 1 счет будет разрешен, если на входе INT0 (для Т/С0) или на входе INT1 (для Т/С1) будет установлено состояние логическая 1, и будет запрещен, если на этих входах будет установлено состояние логический 0.

Установка бита TR0 для Т/С0 и бита TR1 для Т/С1 в состояние логический 0 выключает таймеры/счетчики независимо от состояния других битов.

В режиме 0 Т/С0 и Т/С1 функционируют независимо друг от друга.

Режим работы 1

Режим 1 аналогичен режиму 0, однако установка режима 1 превращает T/C в устройство на основе 16-разрядного регистра. Для T/C 0 регистр состоит из программно доступной пары TL0 и TH0, а для T/C1 — из программно доступной пары TL1 и TH1.

Режим работы 2

В этом режиме T/Cx представляет собой устройство на основе 8 - разрядного регистра TLx. При каждом переполнении TLx кроме установки соответствующего флага TFx в регистре TCON происходит автоматическая перезагрузка из THx в TLx. Регистры THx (т.е. TH0 или TH1) загружаются программно. Их содержимое остается неизменным, т.к. перезагрузка на него не влияет. Назначение битов управления TRx, GATEx, C/Tx такое же, как в режиме 0.

Режим работы 3

В режиме 3 T/C 1 заблокирован и просто сохраняет своё содержание (значения кода в регистре T/C). Эффект такой же, как при установке TR1 = 0.

T/C 0 в режиме 3 представляет собой два независимых устройства на основе 8-разрядных регистров TH0 и TL0. Устройство на основе регистра TL0 может работать в режиме счетчика и в режиме таймера. За ним сохраняются все биты управления T/C0 (т.е. GATE0, TR0, CT/0), оно реагирует на воздействия по входам T0 и INT0/. При переполнении TL0 аппаратно устанавливается флаг TF0 = 1.

Устройство на основе TH0 может работать только в режиме таймера. Оно использует бит включения TR1, при переполнении TH0 устанавливается бит TF1. Других битов управления таймер на основе TH0 в режиме 3 не имеет.

Установка Т/С 0 в режим 3 лишает Т/С1 бита включения TR1. Поэтому Т/С 1 в режимах 0,1, 2 при GATE1 = 0 всегда включен и при переполнении в режимах 0 и 1 обнуляется, а в режиме 2 перезагружается, не устанавливая флаг TF1, если Т/С0 находится в режиме 3. Управление от входов INT1/, T1, биты C/T1, GATE1 для Т/С 1 не зависит от режима Т/С 0. Т/С1 аппаратно связан в блоком синхронизации последовательного интерфейса (ПИ). При работе в режимах 0, 1, 2 при переполнении Т/С1 всегда вырабатывается импульс тактирования ПИ.

Выключение Т/С выполняется двумя способами:

- с помощью битов TR0, TR1 (установка этих битов в логический 0),
- с помощью входов микроконтроллера INT0/ и INT1/ (установка на этих входах логического 0 при GATE =1) и не изменяет состояние регистра таймера/счетчика. Таймер/счетчик можно выключить, через произвольное время вновь включить и счет продолжится с той величины, которая была в регистре Т/С на момент выключения.

Новая загрузка Т/С сразу же означает новую величину счета, а предыдущая теряется. Если загрузка произведена при включенном таймере/счетчике , то счет продолжится с новой величины. Очередность загрузки регистров TL0,TH0, TL1 и TH1 — произвольная.

Во всех режимах, кроме режима 2, после переполнения Т/С счет продолжается с величины 00h, если Т/С не выключить с помощью битов TR0, TR1.

Работа таймеров/счетчиков во всех режимах иллюстрируется схемой, приведённой на рис. 4.3.

4.4. Блок последовательного интерфейса и прерываний. Регистры SCON, IE, IP

Рассматриваемый блок предназначен для организации ввода-вывода последовательных потоков информации и организации системы прерывания

программ.

Состав блока последовательного интерфейса и прерываний:

- буфер блока;
- логика управления блока;
- регистр управления SCON;
- буфер приемника;
- буфер передатчика;
- приемник/передатчик последовательного порта;
- регистр приоритетов прерываний IP;
- регистр разрешения прерываний IE;
- логика обработки флагов прерываний;
- схема выработки вектора прерывания.

Буфер блока последовательного интерфейса и прерываний (ПИП) обеспечивает побайтовый обмен информацией между внутренней магистралью данных и шиной блока.

Логика управления блока предназначена для выработки сигналов управления, обеспечивающих 4 режима работы последовательного интерфейса, и организации прерывания программ.

Регистр управления ПИ SCON предназначен для приема и хранения кода 8-битного слова, управляющего последовательным интерфейсом.

Все разряды SCON программно доступны по чтению и записи (логические 1 или 0).

Назначение битов регистра SCON

Биты 7 и 6 (SM0 и SM1 соответственно) — биты определяющие один из 4-х режимов работы (0-й, 1-й, 2-й или 3-й), которые отличаются форматом передаваемой/принимаемой информации, частотой синхронизации приема/передачи и источником синхронизации.

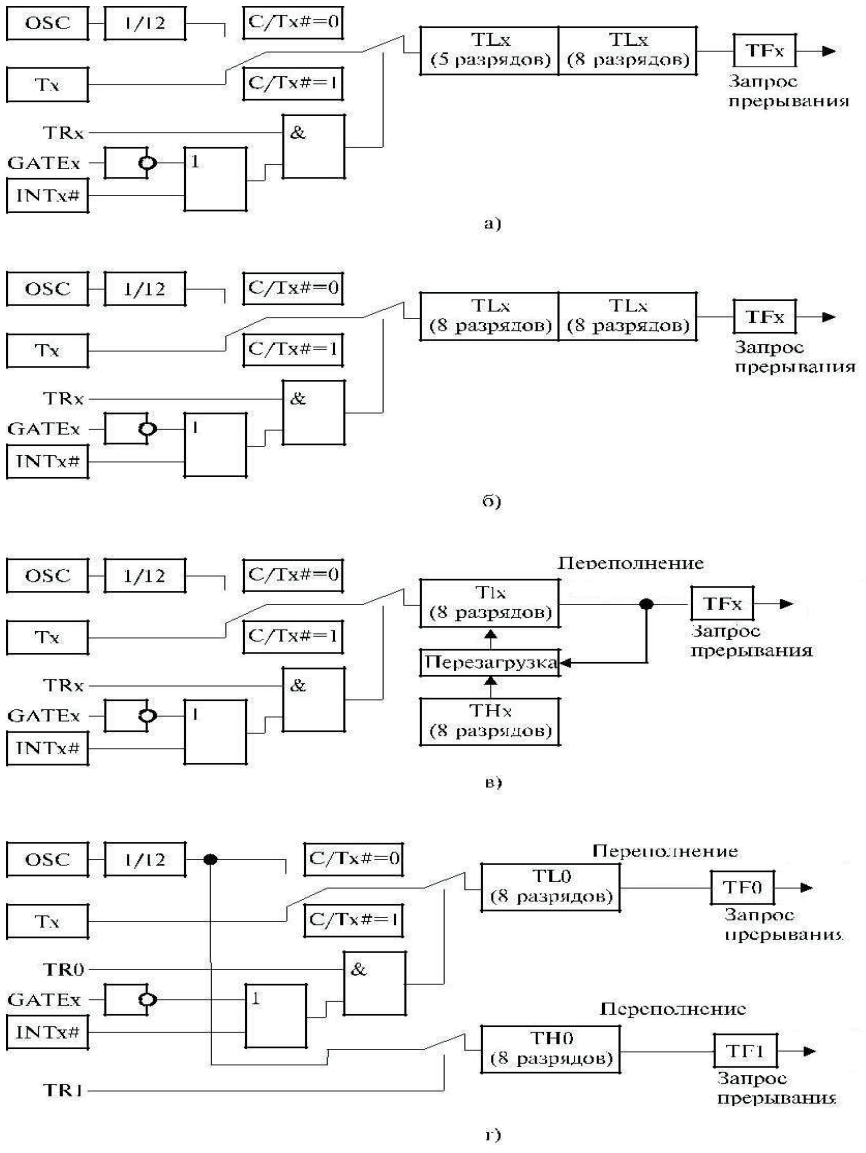


Рисунок 4.3 — Работа счётчиков/таймеров в режиме 0 (а), режиме 1 (б), режиме 2 (в) и режиме 3 (г)

Бит 5 (SM2) — бит разрешения многопроцессорной работы.

В режиме 0 бит SM2 должен быть установлен в лог. 0.

В режиме 1 при SM2, установленном в состояние «лог.1», флаг прерывания приемника RI (бит 0 регистра SCON) не устанавливается (не активизируется), если не принят стоп-бит, равный логической 1. В режимах 2 и 3 при SM2, установленном в состояние «лог.1», флаг прерывания приемника RI (бит 0 регистра SCON) не устанавливается, если девятый принятый бит данных равен логическому 0.

Бит 4 (REN) — бит разрешения приема последовательных данных. Устанавливается и сбрасывается программным образом соответственно для раз-решения или запрещения приема.

Бит 3 (TB8) — девятый бит передаваемых данных в режимах 2 и 3. Устанавливается и сбрасывается программно.

Бит 2 (RB8) — девятый бит принятых данных в режимах 2 и 3. В режиме 1, если бит SM2= лог.0, RB8 является принятым стоп - битом. В режиме 0 бит RB8 не используется.

Бит 1 (T1) — флаг прерывания передатчика. Устанавливается аппаратно в конце выдачи 8-го бита данных в режиме 0 или в начале стоп−бита в других режимах. Сбрасывается программно.

Бит 0 (R1) — флаг прерывания приемника. Устанавливается аппаратно в конце времени приема 8-го бита данных в режиме 0 или через половину интервала стоп-бита в других режимах при состоянии бита SM2= лог. 0. При состоянии бита SM2 = лог.1 см. описание бита SM2.

Последовательный интерфейс семейства МК51 может работать в следующих 4-х режимах.

SM0	SM1	Режим	Наименование режима передачи	Скорость
0	0	0	Сдвиговый регистр	$f_{зг}$=12
0	1	1	8-битовыйт универсальный - асинхронный приемник/передатчик (УАПП)	Переменная, задается Т/С1
1	0	2	9-битовый УАПП	$f_{зг}$=36 или $f_{зг}$=64
1	1	3	9-битовый УАПП	Переменная, задается Т/С1

Режим 0

Информация передается и принимается через вход приемника R_XD (вывод Р3.0 микроконтроллера). Через выход передатчика T_XD (вывод Р3.1 микроконтроллера) выдаются импульсы синхронизации, стробирующие каждый передаваемый или принимаемый бит информации. Формат посылки 8 бит. Частота приема и передачи $f_{зг}$/12, где $f_{зг}$ — частота кварцевого резонатора (см. разд. 4.1).

Режим 1

Информация передается через выход передатчика T_XD, а принимается через вход приемника R_XD. Формат посылки — 10 бит: старт — бит — лог.0; 8 бит информации (данных) и стоп - бит — лог. 1. Частота приема и передачи определяется Т/С1.

Режим 2

Информация передается через выход передатчика T_XD, а принимается через вход приемника R_XD. Формат посылки — 11 бит: старт — бит — лог. 0… 8 бит информации (данных), программируемый бит 9 (разряд TB8 регистра SCON) и стоп — бит — лог.1.

Формат приема — 11бит. При приеме бит 9 принятых данных

записывается в бит RB8 регистра SCON. Частота приема и передачи в режиме 2 уста-навливается программно и может составлять fзг/32 или fзг/64 (устанавливается в регистре PCON битом SMOD).

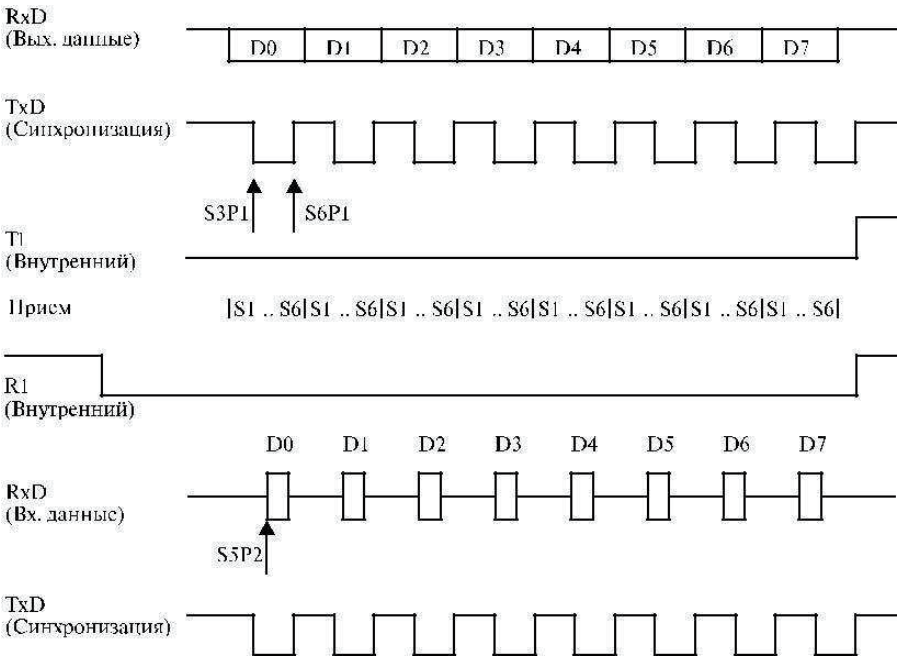


Рисунок 4.4 - Работа последовательного порта в режиме 0

Режим 3

Полностью идентичен режиму 2 за исключением частоты приема и передачи, которая в режиме 3 определяется T/C1.

Буфер передатчика предназначен для приема с шины параллельной информации и выдачи ее в виде последовательного кода на передатчик последовательного порта.

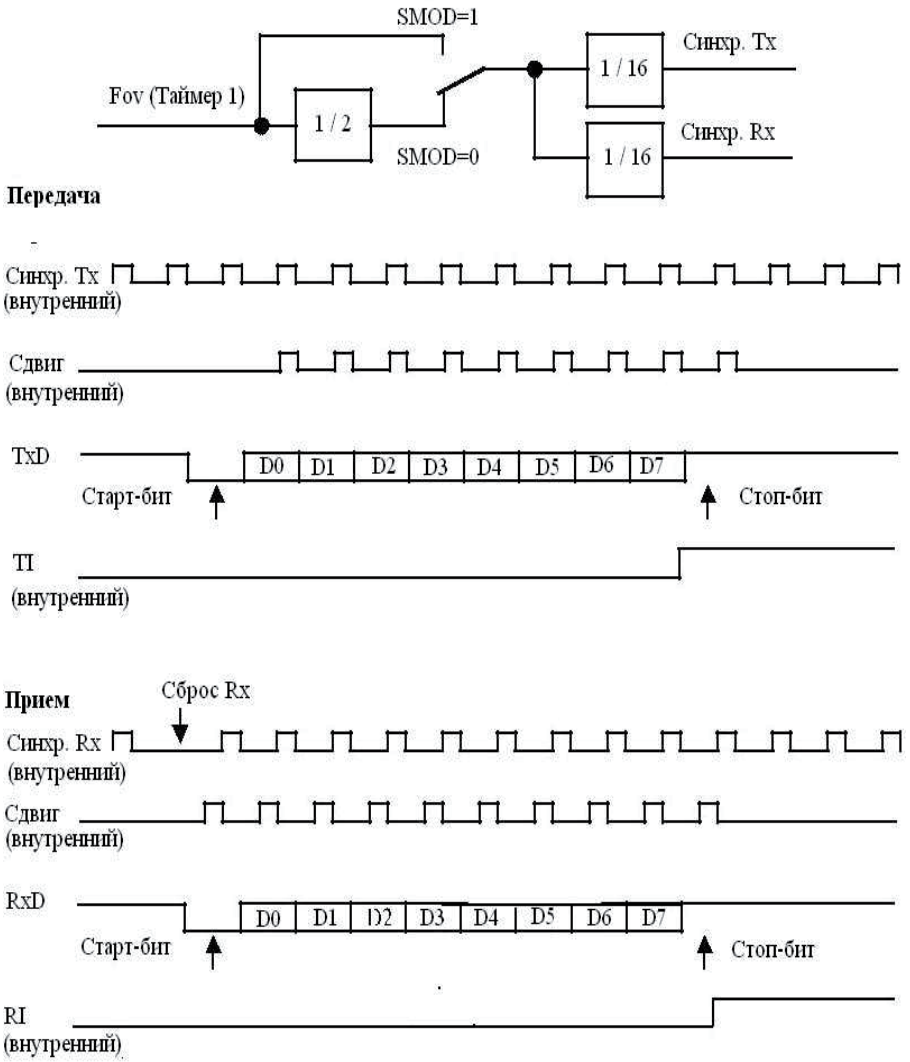


Рисунок 4.5 — Работа последовательного порта в режиме 1

Буфер приемника предназначен для приема данных в последовательном коде, преобразования их в параллельный вид и передачи их в параллельном виде на внутреннюю шину блока последовательного порта.

Буфер приемника и буфер передатчика при программном доступе имеют одинаковые названия (SBUF) и адрес (99h).

Примеры команд обращения к последовательному порту:

MOV A, SBUF — загрузка аккумулятора принятыми данными из SBUF, приемник информации;

MOV SBUF, A — пересылка содержимого аккумулятора в SBUF для передачи, передатчик информации.

Во всех четырех режимах работы последовательного порта передача инициируется любой командой, которая использует SBUF как регистр назначения. Прием в режиме 0 инициируется условием RI=лог. 0 и REN=лог.1. В остальных режимах прием инициируется приходом старт-бита, если REN=лог.1.

Приемник - передатчик последовательного интерфейса (порта) предназначен для перераспределения передаваемых/принимаемых данных между буферами приемника и передатчика.

Регистр приоритетов прерываний IP предназначен для установки уровня приоритета прерывания для каждого из пяти источников прерываний. Назначение отдельных битов регистра IP следующее:

бит 0 (PX0) — установка приоритета прерывания от входа INT0;

бит 1 (PT0) — установка приоритета прерывания от T/C 0;

бит 2 (PX1) — установка приоритета прерывания от входа INT1;

бит 3 (PT1) — установка приоритета прерывания от T/C 1;

бит 4 (PS) — установка приоритета прерывания от последователь ного порта;

биты 5,6 и 7 — резервные.

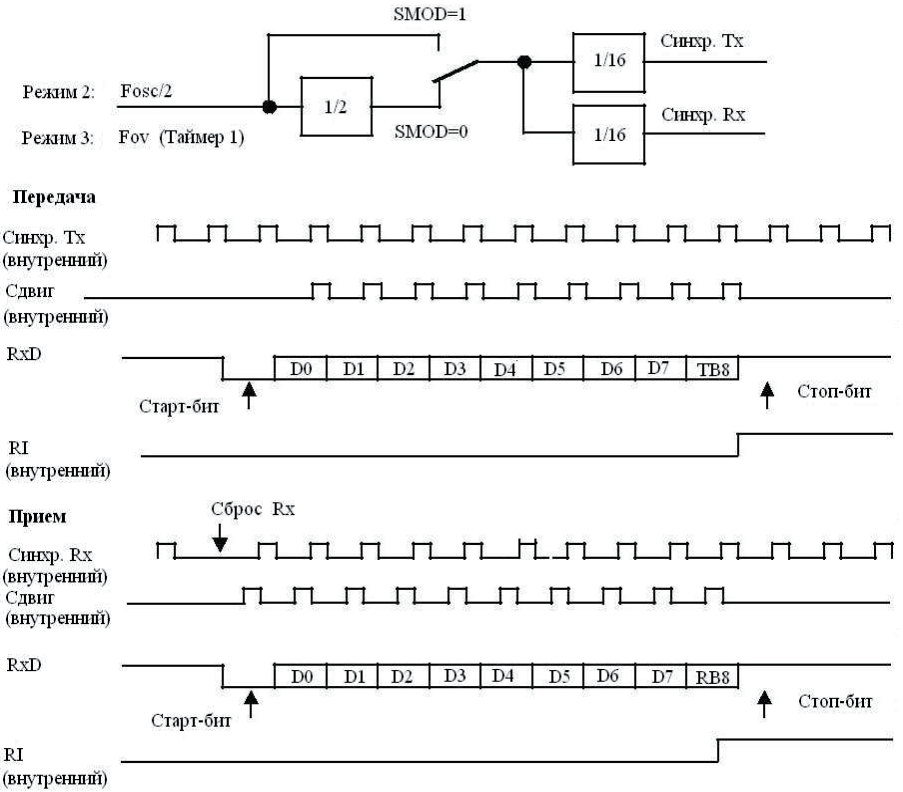


Рисунок 4.6 — Работа последовательного порта в режимах 2 и 3

Наличие в разряде регистра IP «1» устанавливает для соответствующего источника высокий уровень приоритета, а наличие в разряде IP “0” — низкий уровень приоритета. В резервные разряды регистра IP необходимо всегда записывать “0”.

Регистр разрешения прерываний IE

Предназначен для разрешения или запрещения прерываний от соответствующих источников. Обозначение и назначение отдельных битов регистра IE следующее:

бит 0 (EX0) — разрешение прерывания от входа INT0, если EX0=1;

бит 1 (ET0) — разрешение прерывания от T/C0, если ET0=1;

бит 2 (EX1) — разрешение прерывания от входа INT1, если EX1=1;

бит 3 (ET1) — разрешение прерывания от T/C1, если ET1=1;

бит 4 (ES) — разрешение прерывания от последовательного порта, если ES=1;

биты 5 и 6 — резервные;

бит 7 (EA) — разрешение всех пяти прерываний, если EA=1.

В резервные разряды регистра IE необходимо записывать “0”.

Логика обработки флагов прерывания осуществляет приоритетный выбор запроса прерывания, сброс его флага и инициирует выработку аппаратно реализованной команды перехода на подпрограмму обслуживания прерывания.

Схема выработки вектора прерывания вырабатывает двухбайтовые адреса подпрограмм обслуживания прерывания в зависимости от источника прерываний:

	адрес
Внешнее прерывание INT0	0003h
Внутреннее прерывание T/C0	000Bh
Внешнее прерывание INT1	0013h
Внутреннее прерывание T/C1	001Bh
Внешнее/внутреннее прерывание от последовательного порта	0023h

4.5. Память данных (ОЗУ) и программ (ПЗУ)

4.5.1. Память данных

Предназначена для приема, хранения и выдачи информации, используемой в процессе выполнения программы.

Память данных, расположенная на кристалле процессора, состоит из регистра адреса ОЗУ, дешифратора, ОЗУ и указателя стека SP.

Регистр адреса ОЗУ предназначен для записи и хранения адреса выбираемой с помощью дешифратора ячейки памяти, которая может содержать как бит, так и байт информации.

Указатель стека SP представляет собой 8 — разрядный регистр, предназначенный для приема и хранения адреса ячейки стека, к которой было последнее обращение (прединкрементная схема). При выполнении команд LCALL, ACALL содержимое указателя стека увеличивается на 2, при выполнении команд возврата RET, RETi содержимое указателя стека уменьшается на 2. При выполнении команды PUSH direct содержимое SP увеличивается на 1, при выполнении команды POP direct содержимое указателя уменьшается на 1. После сброса содержимое указателя стека составляет 07h, что соответствует адресу начала стека 08h (первой свободной ячейки, в которую производится запись).

В контроллерах семейства МК 51 предусмотрена возможность расширения памяти данных путем подключения внешних устройств ОЗУ емкостью до 64 Кбайт. При этом обращение к внешней памяти данных осуществляется только при помощи команд MOV X.

Команды MOVX A, @ R_i и MOVX @ R_i , A формируют 8 — разрядный адрес, который выдается через порт P0, команды MOVX A,@DPTR и MOVX,@DPTR, A формируют 16 — разрядный адрес, младший байт выдается через порт P0, а старший — через порт P2.

Байт адреса, выдаваемый через P0, должен быть зафиксирован во внешнем регистре по спаду сигнала ALE, т. к. в дальнейшем эти же линии будут задействованы как шина данных, через которую байт данных

принимается из внешней памяти при чтении, или байт данных выдается во внешнюю память при записи. При этом чтение стробируется сигналом контроллера RD, а запись — сигналом контроллера WR. При работе с внутренней памятью данных эти сигналы не формируются.

Страничная организация внешней памяти данных.

При обеспечении страничной организации памяти исходят из типа микросхемы ОЗУ, т. е. из ее емкости и команд обращения.

Пример 1: внешняя память данных состоит из одной микросхемы 2Кх8 бит. Обращение осуществляется командами типа MOVX A, @ R_i и MOVX @ R_i , A, т. е. 8 — разрядными посылками. Схема соединений показана на рис. 4.7.

Представленная схема реализует деление 2Кбайт внешнего ОЗУ на 8 страниц по 256 байт каждая 5 линий порта P2 можно использовать как линии ввода/вывода. Страницы могут программироваться при помощи команд SETB bit.

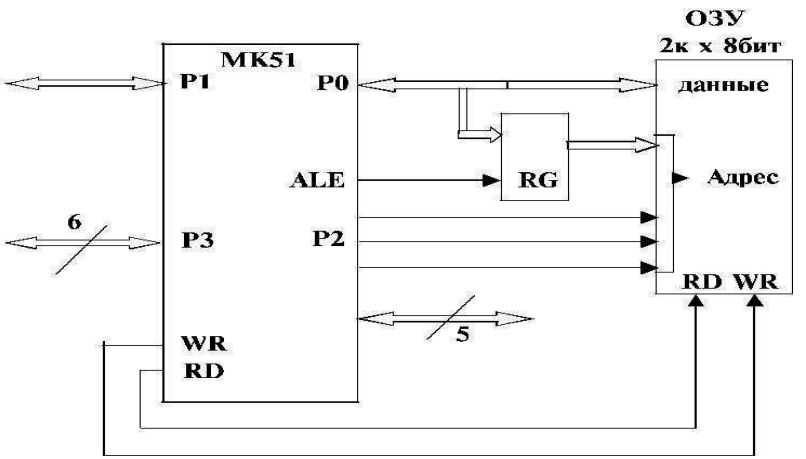


Рисунок 4.7

Пример 2: внешняя память данных состоит из ОЗУ размером 8 К х8 бит. Обращение осуществляется при помощи команды MOVX A, @DPTR.

Схема соединений показана на рис. 4.8.

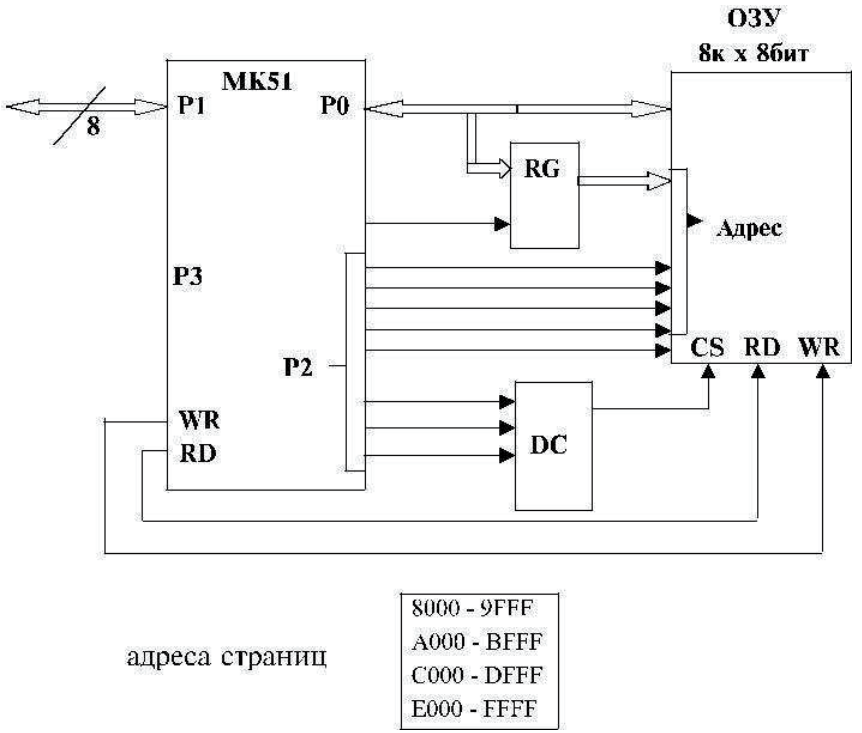


Рисунок 4.8

Пространства внутренней и внешней памяти данных не пересекается, т. к. при обращении к ним используется различные команды.

4.5.2. Память программ

Память программ предназначена для хранения программ и имеет отдельное от памяти данных адресное пространство объемом до 64 Кбайт.

Для управления обращением к памяти программ на корпусе микросхемы имеется вывод DEMA. Если на этом выводе +5V, то адреса с 0000 до 0FFFh используются как внутренняя память программ, а адреса с 1000h до 0FFFFh процессор автоматически рассматривает как внешнюю. Если на DEMA — 0V, то микроконтроллер работает только с внешней памятью программ.

Память программ при работе процессора только считывается (т. е. представляет собой ПЗУ), чтение из внешней памяти программ стробируется (тактируется) сигналом PME (сигнал чтения из внешней памяти программ). Сигнал PME формируется дважды в течение машинного цикла. При работе с внутренней памятью программ сигнал PME не формируется. Микроконтроллер не имеет команд и аппаратных средств для программной записи в память программ.

Пример соединения микроконтроллера с ПЗУ программ представлен на рис. 4.9.

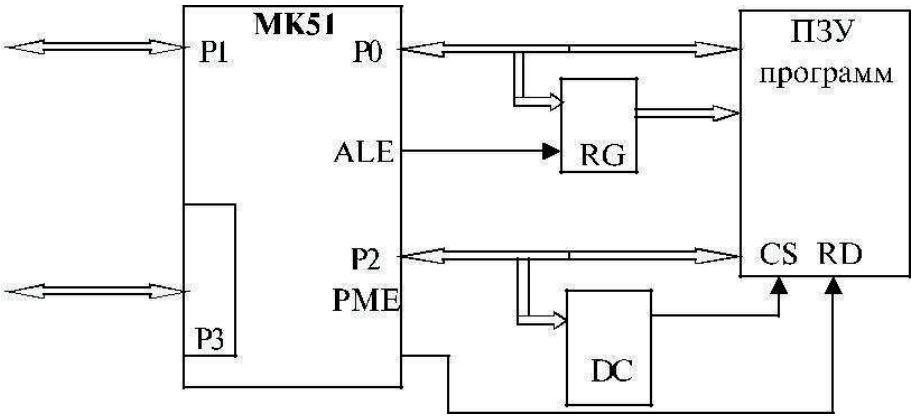


Рисунок 4.9

Пример цоколевки ИС ОЗУ данных показан на рис 4.10.

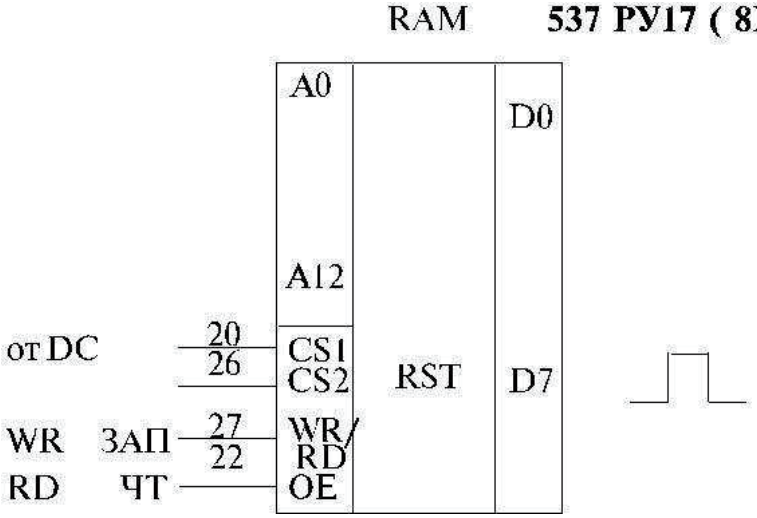


Рисунок 4.10

Пример цоколевки ИС ПЗУ программ приведен на рис. 4.11.

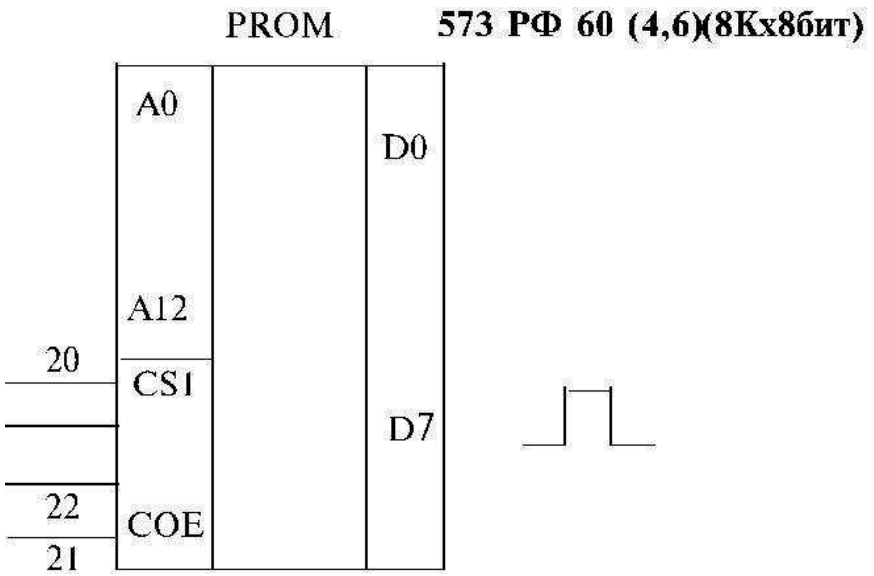


Рисунок 4.11

4.6. Счетчик команд. Регистр DPTR

Счетчик команд(PC) предназначен для формирования текущего 16-разрядного адреса программной памяти и 8/16-разрядного адреса внешней памяти данных.

Состав счетчика команд:

- 16-разрядный буфер PC;
- регистр указателя данных DPTR;
- регистр программного счетчика;
- схема инкремента;
- регистр адреса памяти.

Буфер PC осуществляет связь между 16-разрядной шиной PC и 8-рязрядной магистралью данных, обеспечивая запись, хранение и коммутацию информации.

Регистр DPTR предназначен для хранения 16-разрядного адреса внешней памяти данных. Он состоит из двух 8-разрядных регистров DPH и DPL, относящихся к регистрам специальных функций. Они программно доступны и могут использоваться в качестве двух независимых РОН, если нет необходимости в хранении 16-разрядного адреса внешней памяти данных.

Регистр PC предназначен для хранения текущего 16-разрядного адреса памяти программ.

Схема инкремента увеличивает содержимое регистра PC на 1.

Регистр адреса памяти предназначен для записи и хранения исполнительного 16-разрядного адреса памяти программ или 8/16-разрядного адреса внешней памяти данных, а также для передачи данных в порт P0 при выполнении команд MOVX @Ri, A и MOVX @DPTR, A, обеспечивающих запись данных через порт P0 во внешние устройства.

4.7. Порты

Порты P0,P1,P2,P3 являются двунаправленными портами ввода/вывода и предназначены для обеспечения обмена информацией микроконтроллера с внешними устройствами. Они образуют 32 линии ввода/вывода. Каждый из портов содержит фиксатор-защелку, который представляет собой 8-разрядный регистр, имеющий байтовую и битовую адресацию, для сброса/установки программными средствами.

Физические адреса фиксаторов:

P0 — 80h, при битовой адресации 80 . . . 87 h
P1 — 90h, при битовой адресации 90 . . . 97 h
P2 — A0h, при битовой адресации A0 . . A7 h
P3 — B0h, при битовой адресации B0 . . . B7 h.

Помимо работы в качестве обычных линий ввода/вывода, все порты могут выполнять ряд дополнительных функций.

Порт P0 :

- служит для выдачи младшего байта адреса A0 . . . A7 при работе с внешней памятью данных и внешней памятью программ;
- служит для приема / передачи данных при работе с внешней памятью. При этом младший байт адреса и байт данных разнесены во времени и привязаны к системному сигналу ALE;
- участвует в процессе программирования внутреннего ППЗУ программ как шина данных (чтение / запись).

Порт P1:

- используется как адресный (младший байт адреса) при программировании внутреннего ППЗУ и при чтении внутренней памяти программ.

Порт P2:

- используется как адресный (старший байт адреса) при работе с внешней памятью программ и данных (для внешней памяти данных только при использовании команд MOVX A, @ DPTR и MOVX @ DPTR, A, которые вырабатывают 16 — разрядный адрес);
- как адресный (старший байт адреса A8 . . . A14) при программировании внутреннего ППЗУ и при чтении внутренней памяти программ.

Порт P3:

Каждая линия порта P3 имеет альтернативную функцию:

P3.0 — R_X вход приемника последовательного интерфейса (ПИ);

P3.1 — T_X выход передатчика ПИ;

P3.2 — INT 0 — вход 0 внешнего запроса прерывания;

P3.3 — INT 1 — вход 1 внешнего запроса прерывания;

P3.4 — T0 — вход счетчика внешних событий Т/С 0;

P3.5 — T1 — вход счетчика внешних событий Т/С 1;

P3.6 — WR — строб записи во внешнюю память данных, выходной сигнал, сопровождающий вывод данных через P0 при выполнении команд MOV X @ R_i , A, MOV X @ DPTR, A.

P3.7 — RD — строб чтения из внешней памяти данных, выходной сигнал, сопровождающий ввод данных через порт P0 при выполнении команд MOV X A, @ R_i , MOV X A, @ DPTR.

Альтернативная функция любой из линий P3 реализуется лишь при условии, что в соответствующий разряд фиксатора/защелки программно записана «1».

Режим “чтение — модификация — запись”

Команды чтения портов микроконтроллеров МК51 делятся на две категории: команды чтения информации с выходов защелок и команды чтения информации с физических выводов портов.

Команды, в которых порт служит операндом — источником, считывают информацию непосредственно с выводов порта, например ADD A,P1: содержимое аккумулятора складывается с информацией на выводах порта P1 и результат заносится в аккумулятор.

Команды, считывающие информацию с выходов защелок, реализуют так называемый режим “чтение — модификация — запись”. Режим состоит в том, что команда считывает информацию с защелки, модифицирует её при необходимости и обратно записывает в ту же защелку.

Во всех случаях, когда операндом — регистром назначения является порт или бит порта, то команды считывают информацию с защелок портов, а не с внешних контактов выводов портов.

Примеры таких команд:

ANL P1,A

XRL P2,A

ORL P0,A

CPL P3,0

INC P2

DEC P2

DJNZ P3,метка

4.8. Начальная установка микроконтроллера 8051

Инициализация (сброс) микросхем осуществляется сигналом RST (активный высокий уровень напряжения) при условии подачи на микросхему внешнего сигнала синхронизации или при подключенном кварцевом резонаторе. Для того чтобы сброс гарантированно произошел, длительность сигнала высокого уровня на входе RST должна быть не менее двух машинных циклов.

Сигналы ALE и PME устанавливаются в “1” и находятся в этом

состоянии пока на входе RST присутствует логическая 1. После подачи на вход RST низкого логического уровня начинают формироваться сигналы ALE и PME.

При подаче сигнала RST внутренний алгоритм сброса микроконтроллера выполняет следующие действия:

- устанавливает в ноль счетчик команд PC, регистры специальных функций (кроме защелок портов, указателя стека SP и регистра SBUF);
- указатель стека SP устанавливается в 07h;
- запрещает работу всех прерываний, таймеров, счетчиков и последовательного порта;
- устанавливает фиксаторы — защелки портов P0 ... P3 в «1»;
- в регистрах SBUF устанавливаются случайные значения.

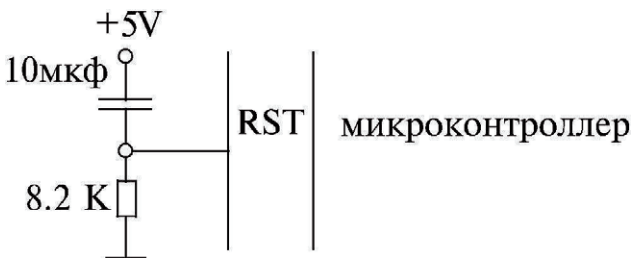


Рисунок 4.12 — Пример реализации сброса по включению питания

Сигнал сброса RST не влияет на внутреннее ОЗУ данных. После включения питания содержимое ячеек внутреннего ОЗУ данных принимает случайные значения.

5. СИСТЕМА КОМАНД. СПОСОБЫ АДРЕСАЦИИ ОПЕРАНДОВ

Способы адресации операндов Существуют следующие способы адресации операндов-источников:

- регистровая адресация;
- прямая адресация;
- косвенная регистровая (включая стековую адресацию);
- непосредственная адресация;
- косвенная адресация по сумме базового и индексного регистров.

Первые три способа используются также для адресации операнда назначения. Указанные 5 способов адресации в различных сочетаниях обеспечивают 21 режим адресации.

Регистровая адресация используется для обращения к 8 РОН (рабочим регистрам) выбранного банка регистров. Регистровая адресация используется также для обращения к регистрам A, B, AB (сдвоенный регистр), DPTR и к флагу переноса C.

Прямая адресация. Прямая байтовая адресация используется для обращения к ячейкам внутреннего ОЗУ данных и к регистрам специального назначения.

Прямая битовая адресация используется для обращения к отдельно адресуемым битам (128 бит) в ячейках с адресами 20H – 2FH и к битам регистров специального назначения.

Косвенно-регистровая адресация используется для обращения к внутренним ячейкам ОЗУ данных. В качестве регистров−указателей используются регистры R0, R1 выбранного банка регистров.

В командах обращения к стеку POP и PUSH используется содержимое указателя стека (SP); 16-разрядный указатель данных (DPTR) может быть использован для обращения к любой ячейке адресного пространства внешней

памяти данных до 64 Кбайт.

Непосредственная адресация позволяет выбрать из адресного пространства памяти программ константы, явно заданные в команде.

Косвенно-регистровая адресация по сумме базового и индексного регистров. В качестве базовых могут быть регистры PC и DPTR, в качестве индексного — содержимое аккумулятора A.

5.1 Общие сведения о базовых командах микроконтроллера 8051

Система команд предоставляет большие возможности обработки данных, обеспечивает реализацию логических, арифметических операций, а также управление в реальном времени. Реализована побитовая, потетрадная, побайтная и 16-разрядная обработка данных.

БИС семейства MK51- 8 разрядная ОЭВМ: ПЗУ, ОЗУ, регистры специального назначения, АЛУ; и внешние шины имеют байтовую организацию. Двухбайтовые данные используются только регистром-указателем (DPTR) и счетчиком команд (PC).

Набор команд ОЭВМ содержит 42 мнемонических обозначений (аббревиатур) команд для конкретизации 33 функций этой системы.

Синтаксис большинства команд

В команде вначале записывается мнемоническое обозначение функции, а затем идут операнды, указывающие метод адресации и типы данных.

В машинном коде команда занимает 1, 2 или 3 байта. Команды выполняются за 1, 2 или 4 машинных цикла (4 цикла — только команды умножения и деления).

Систему команд ОЭВМ условно можно разбить на пять групп:

- арифметические;
- логические команды с байтовыми переменными;
- команды передачи данных;

- команды работы с битами;
- команды ветвления программ и передачи управления ОМЭВМ.

Арифметические команды

В базовом наборе команд микроконтроллера 8051 имеются следующие арифметические операции: сложение, сложение с учетом флага переноса, вычитание с учетом заема, инкрементирование, декрементирование, сравнение, десятичная коррекция, умножение и деление. В АЛУ производятся действия над целыми числами без знака. Операции сложения, сложения с учетом флага переноса, вычитания с учетом заема влияют на состояние флагов в регистре PSW: переполнения OV, переноса C, дополнительного переноса AC и четности P. Выполнение операций со знаком осуществляется программным способом. Операции инкрементирования и декрементирования на флаги не влияют. Только на аккумуляторе выполняются следующие команды: проверка содержимого аккумулятора JZ и JNZ, а также десятичная коррекция DAA. Команды умножения и деления помимо аккумулятора используют дополнительный регистр B.

Логические команды с байтовыми переменными

Система команд позволяет реализовать следующие логические операции:

- двухоперандные — «И», «ИЛИ», «Исключающее ИЛИ»;
- одноопернадные — сброс (очистка), инверсия байта, циклический сдвиг влево и вправо с учетом флага переноса и без его учета, обмен местами старшей и младшей тетрад (ниблов) внутри аккумулятора.

Команды передачи данных

Это самая многочисленная группа команд, благодаря которой в

микроконтроллере имеется возможность осуществить передачу информации между любыми ячейками внутреннего и внешнего ОЗУ данных.

Команды, оперирующие с битами, обеспечивают прямую адресацию 128 битов в шестнадцати ячейках внутреннего ОЗУ данных (ячейки с адресами 20h — 2Fh) и прямую побитовую адресацию регистров специального назначения, если таковая предусмотрена. Каждый из отдельно адресуемых битов может быть установлен в «1» или «0», инвертирован и проверен. По состоянию бита могут быть организованы переходы в программе. Между битом и флагом переноса могут быть произведены логические операции «И», «ИЛИ», при этом результат заносится в разряд флага переноса С.

Команды ветвления программ и передачи управления

В системе команд имеются команды 16 — разрядных, 11 — разрядных и 8 — разрядных переходов. Команды 16 — разрядных переходов и вызовов подпрограмм позволяют осуществить переход в любую точку адресного пространства памяти программ объемом 64 Кбайт. Команды 11 — разрядных переходов и вызовов подпрограмм позволяют осуществить переход внутри программного модуля емкостью 2 Кбайт. Команды 8 — разрядных пере-ходов — это команды условных и безусловных переходов относительно на-чального адреса следующей команды в пределах от (PC) — 128 до (PC) + 127, которые обеспечивают переход внутри программного модуля емкостью 256 байт. Условием перехода являются содержимое бита, аккумулятора или регистра.

Рассмотрим подробно описание всех команд базовой системы.

5.2 Описание команд микроконтроллера 8051

Команда ACALL <addr 11>

Команда “абсолютный вызов подпрограммы” вызывает безусловно подпрограмму, размещенную по указанному адресу. При этом счетчик команд

увеличивается на 2 для получения адреса следующей команды, после чего полученное 16-битовое значение PC помещается в стек (сначала следует младший байт), и содержимое указателя стека также увеличивается на 2. Адрес перехода получается с помощью конкатенации старших бит увеличенного содержимого счетчика команд, битов старшего байта команды и младшего байта команды.

Ассемблер: ACALL <метка>

Код : 2 байта <**A10 A9 A8 10001**> <**A7 A6 A5 A4 A3 A2 A1 A0**>

Время: 2 цикла

Алгоритм:(PC):=(PC)+2

(SP):=(SP)+1

((SP)):=(PC[7-0])

(SP):=(SP)+1

((SP)):=(PC[l5-8l)

(PC[10-0]):=A10A9A8 II A7A6A5A4A3A2A1A0,

где II — знак конкатенации (сцепление)

Пример:; ДО ВЫПОЛНЕНИЯ КОМАНДЫ ACALL

;(SP)=07H

;метка MT1 соответствует адресу: 0345H,

;т.е. (PC)=0345H

ACALL MT1; расположена по адресу 028DH, т.е.;

(PC)=028DH

;ПОСЛЕ ВЫПОЛНЕНИЯ КОМАНДЫ

;(SP)=09H, (PC)=0345H,

;ОЗУ (08)=8FH, ОЗУ (09)=02H.

Команда ADD A, < байт-источник>

Эта команда (“сложение”) складывает содержимое аккумулятора A с содержимым байта-источника, оставляя результат в аккумуляторе. При

появлении переносов из разрядов 7 и 3 устанавливаются флаги переноса (C) и дополнительного переноса (AC) соответственно, в противном случае эти флаги сбрасываются. При сложении целых чисел без знака флаг переноса “C” указывает на появление переполнения. Флаг переполнения (OV) устанавливается, если есть перенос из бита 6 и нет переноса из бита 7, или есть перенос из бита 7 и нет — из бита 6**,** в противном случае флаг OV сбрасывается. При сложении целых чисел со знаком флаг OV указывает на отрицательную величину, полученную при суммировании двух положительных операндов или на положительную сумму для двух отрицательных операндов. Для команды сложения разрешены следующие режимы адресации байта-источника:

1) регистровый;
2) косвенно-регистровый;
3) прямой;
4) непосредотвенный.

1) Ассемблер: ADD A,Rn ; где n=0-7

Код:1 байт <**00101 rrr**> ,где rrr=000-111

Время:1 цикл

Алгоритм: (A):=(A)+(Rn), где n=0-7

C:=X, OV:=X, AC:=X, где X=(0 или 1)

Пример:

;(A)=C3H, (R6)=AAH

ADD A,R6 ;(A)=6DH, (R6)=AAH

;(AC)=0, (C)=1, (OV)=1

2) Ассемблер: ADD A,@Ri; где i=0,1

Код: 1 байт <**0010011i**>, где i=0,1

Время:1 цикл

Алгоритм:(A):=(A)+((Ri)), где 1=0,1

C:=X, OV:=X, AC:=X, где X=(0 или 1)

Пример:

;(A)=95H, (R1)=31H,(ОЗУ[31])=4CH

ADD A,@R1 ;(A)=E1H. (ОЗУ [31])=4CH,

; (C)=0, (AC)=1, (OV)=0

3). Ассемблер: ADD A,<direct>

Код: 2 байта <**00100101**> <**direct addres**>,

Время:1 цикл

Алгоритм:(A):=(A)+(direct)

C:=X,OV:=X,AC:=X,где X=(0 или 1)

Пример:

;(A)=77H, (ОЗУ [90])=FFH

ADD A.90H ;(A)=76H, (ОЗУ[90])=FFH,

;(C)=1, (OV)=0, (AC)=1

4). Ассемблер: ADD A,<#data>

Код: 2 байта <**00100100**> <**#data**>

Время:1 цикл

Алгоритм:(A):=(A)+#data

C:=X, OV:=X, AC:=X, где X=(0 или 1)

Пример: ;(A)=09H

ADD A,#0D3H ;(A)=DCH,

;(C)=0, (OV)=0, AC)=0

Команда ADDC A, < байт-источник>

Эта команда (“сложение с переносом”) одновременно складывает содержимое байта-источника, флаг переноса и содержимое аккумулятора A, оставляя результат в аккумуляторе. При этом флаги переноса и дополнительного переноса устанавливаются, если есть перенос из бита 7 или бита 3, и сбрасываются в противном случае. При сложении целых чисел без знака флаг переноса указывает на переполнение. Флаг переполнения (OV)

устанавливается, если имеется перенос бита 6 и нет переноса из бита 7 или есть перенос из бита 7 и нет — из бита 6, в противном случае OV сбрасывается. При сложении целых чисел со знаком OV указывает на отрицательную величину, полученную при суммировании двух положительных операндов, или на положительную сумму от двух отрицательных операндов.

Для этой команды разрешены следующие режимы адресации байта-источника:

1) регистровый;
2) косвенно-регистровый;
3) прямой;
4) непосредственный.

1) Ассемблер: ADDC A,Rn ; где n=0-7

Код: 1 байт <**00111 rrr**>, где rrr=000-lll

Время:1 цикл

Алгоритм:(A):=(A)+(C)+(Rn)

(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример: ;(A)=B2H, (R3)=99, ;(C)=1

ADDC A.R3;(A)=4CH, (R3)=99.

;(C)=1, (AC)=0, (OV)=1

2). Ассемблер: ADDCA,@Ri ; где i=0,l

Код: 1 байт <**0011011 i**>, где i=0,l

Время: 1 цикл

Алгоритм: (A):=(A)+(C)+((Ri))

(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример:;(A)=D5H, (R0)=3AH,

;(ОЗУ[3A])=1AH, (C)=1

ADDC A,@R0;(A)=F0H,(ОЗУ[3A]=1AH),

;(C)=0, (AC)=1, (OV)=0

3). Ассемблер: ADDC A,<direct>

Код: 2байта < **00110101**> <**direct addres**>

Время: 1 цикл

Алгоритм: (A):=(A)+(C)+(direct)

(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример: ;(A)=11H, (03У[80])=DFH, (C)=1

ADDC A,80H ;(A)=F1H,(C)=0,(AC)=1,(OV)=0

4). Ассемблер: ADDC A,#data

Код: 2 байта <**0 0 1 1 0 1 0 0**> <**#data**>

Время:1 цикл

Алгоритм: (A):=(A)+(C)+(direct)

(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример: ;(A)=55H, (C)=0

ADDC A,#55H ;(A)=AAH, (0=0, (AC)=0, (OV)=1

Команда AJMP <addrll>

Команда “абсолютный переход” передает управление по указанному адресу, который получается при конкатенации пяти старших бит счетчика команд PC (после увеличения его на два), 7—5 битов кода операции и второго байта команды. Адрес перехода должен находиться внутри одной страницы объемом 2Кбайт памяти программы, определяемой пятью старшими битами счетчика команд.

Ассемблер: AJMP <метка>

Код: 2 байта <**A10 A9 A8 00001**> <**A7 A6 A5 A4 A3 A2 A1 A0**>

Время: 2 цикла

Алгоритм: (PC[15-0]):=(PC[15-01)+2,

(PC[10-0]):=<addrll>

Пример: ;(PC)=028FH

;Метке MT2 соответствует адрес 034AH

AJMP MT2;(PC)=034AH

Команда ANL <байт-назначения>,<байт-источник>

Команда “логическое “И” для переменных-байтов” выполняет операцию логического “И” над битами указанных переменных и помещает результат в байт-назначения. Эта операция не влияет на состояние флагов.

Два операнда обеспечивают следующие комбинации шести режимов адресации:

- байтом назначения является аккумулятор (A):
  1) регистровый;
  2) прямой;
  3) косвенно-регистровый;
  4) непосредственный;
- байтом назначения является прямой адрес (direct):
  5) прямой аккумуляторный;
  6) непосредственный (байт-источник равен константе).

1). Ассемблер: ANL A,Rn ; где n=0-7

Код: 1 байт <**01011 rrr**>, где rrr=000-lll

Время:1 цикл

Алгоритм:(A):=(A) AND (Rn)

Пример:;(A)=FEH, (R2)=C5H

ANL A,R2;(A)=C4H, ,(R2)=C5H

2) Ассемблер: ANL A,<direct>

Код: 2 байта **<01010101> <direct address>**

Время:1 цикл

Алгоритм: (A):=(A) AND (direct)

Пример:;(A)=A3H, (PSW)=86H

ANL A,PSW;(A)=82H, (PSW)=86H

3). Ассемблер: ANL A,@Ri ; где i=0,l

Код:1 байт **<0101011 i >**, где i=0,1

Время:1 цикл

Алгоритм:(A):=(A) AND ((Ri))

Пример: ;(A)=0BCH, (ОЗУ[35])=47H, (R0)=35H

ANL A,@R0 ;(A)=04H, (ОЗУ[35])=47H

4). Ассемблер: ANL A.#data

Код: 2 байта **<01010100 > <#data>**

Время:1 цикл

Алгоритм:(A):=(A) AND #data

Пример: ;(A)=36H

ANL A,#0DDH ;(A)=14H

5). Ассемблер: ANL <direct>,A

Код: 2 байта **<0 1 0 1 0 0 1 0> <direct address>**

Время:1 цикл

Алгоритм:(direct):=(direct) AND (A)

Пример: ;(A)=55H, (P2)=AAH

ANL P2,A ;(P2)=00H, (A)=55H

6). Ассемблер: ANL <direct>,#data

Код: 3 байта **<0 1 0 1 0 0 1 1> <direct address> <#data8>**

Время:2 цикла Алгоритм:(direct):=(direct) AND #data

Пример: ;(P1)=FFH

ANL P1,#73H ;(P1)=73H

Примечание. Если команда “ANL” применяется для изменения содержимого порта, то значение, используемое в качестве данных порта, будет считываться из “защелки” порта, а не с выводов БИС.

Команда ANL C,<бит источника>

Команда “логическое “И” для переменных-битов” выполняет операцию

логического “И” над указанными битами. Если бит-источник равен “0”, то происходит сброс флага переноса, в противном случае *флаг* переноса не изменяет текущего значения; “/” перед операндом в языке ассемблера указывает на то, что в качестве значения используется логическое отрицание адресуемого бита, однако сам бит источника при этом не изменяется. На другие флаги эта команда не влияет.

Для операнда-источника разрешена только прямая адресация к битам.

1). Ассемблер: ANL C,<bit>

Код:2 байта **<1 0 0 0 0 0 1 0> <bit address>**

Время: 2 цикла

Алгоритм:(C):=(C) AND (bit)

Пример: ;(C)=1, Pl[0]=0

ANL C,P1.0 ;(C)=0, Pl[0] =0

2). Ассемблер: ANL C,</bit>

Код: 2 байта **<1 0 1 1 0 0 0 0> < bit address>**

Время: 2 цикла

Алгоритм: (C):=(C) AND (/bit)

Пример: ;(C)=1, (AC)=0

ANL C,/AC ;(C)=1, (AC)=0

Команда CJNE <байт назначения,<байт источник>,<смешение>

Команда “сравнение и переход, если не равно” сравнивает значения первых двух операндов и выполняет ветвление, если операнды не равны. Адрес перехода (ветвления) вычисляется при помощи сложения значения (со знаком), указанного в последнем байте команды, с содержимым счетчика команд после увеличения его на три.

Флаг переноса “C” устанавливается в “1”, если значение целого без знака <байта назначения > меньше, чем значение целого без знака < байта источника >, в противном случае перенос сбрасывается (если значения

операндов равны, флаг переноса сбрасывается). Эта команда не оказывает влияния на операнды. Операнды, стоящие в команде, обеспечивают комбинации четырех режимов адресации:

- если байтом-назначения является аккумулятор:
  1) прямой;
  2) непосредственный;
- если байтом-назначения является любая ячейка ОЗУ с косвенно-регистровой или регистровой адресацией:
  3) непосредственный к регистровому;
  4) непосредственный к косвенно-регистровому;

1) Ассемблер: CJNE A,<direct>,<метка>

Код: 3 байта **<10110101> <direct address> <re18>**

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,

если (direct)<(A) то (PC):=(PC)+<re18>, C:=0

если (direct)>(A), то (PC):=(PC)+<re18>, C:=1

Пример: ;(A)=97H, (P2)=0F0H, (C)=0

CJNE A,P2,MT3

MT3: CLR A ;(A)=97H, (P2)=0F0H, (C)=1

;Адрес, соответствующий метке ;MT3 вычисляется , как

;(PC):=(PC)+3+(re18)

2). Ассемблер: CJNE A,#data,<метка>

Код: 3 байта **<10110100> <#data > <re18>**

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,

если #data<(A), то (PC)+<re18>,C:=0

если #data8>(A), то (PC):=(PC)+<re18>, C:=1

Пример: ;(A)=FCH, (C)=1

CJNE A,#0BFH,MT4

MT4: INCA ;(A)=FDH, C=0,

;(PC):=(PC)+3+(re18)

3). Ассемблер: CJNE Rn,#data,<метка> ; где n=0-7

Код: 3 байта **<10111rrr> <#data > <re18>**

Время: 2 цикла

Алгоритм**:** (PC):=(PC)+3,

если #data<(Rn), то (PC):=(PC)+<re18>, C:=0

если #data8>(Rn), то (PC)+<re18>, C:=1

Пример: ;(R7)=80H, (C)=0

CJNE R7,#81,MT5

MT5: NOP ;(R7)=80H, (C)=1

;(PC):=(PC)+3+(rel8)

4). Ассемблер: CJNE @Ri,#data, <метка>; где i=0,1

Код: 3 байта **<1011011i> <#data > <re18>**

Время:2 цикла

Алгоритм:(PC):=(PC)+3,

если #data<((Ri)), то (PC)+<rel8>,C:=0

если #data8>((Ri)), то (PC)+<rel8>,C:=1

Пример: ;(R0)=41H, (C)=1, (ОЗУ[41])=57H

CJNE @R0,#29H,MT6

MT6: DEC R0 ;(ОЗУ[41])=57H, (C)=0,

;(PC):=(PC)+3+(rel8)

Команда CLR A

Команда “сброс аккумулятора” сбрасывает (обнуляет) содержимое аккумулятора А. На флаги команда не влияет.

Ассемблер: CLR A

Код: 1 байт **<11100100>**

Время:1 цикл

Алгоритм:(A):=0

Пример: ; (A)=6DH, (C)=0, (AC)=1

CLR A ; (A)=00H, (C)=0, (AC)=1

Команда CLR <bit>

Команда “сброс бита” сбрасывает указанный бит в нуль. Эта команда работает с флагом переноса “C” или любым битом с прямой адресацией.

1). Ассемблер: CLR C

Код: 1 байт **<11000011>**

Время:1 цикл

Алгоритм:(C):=0

Пример: ;(C)=1

CLR C ;(C)=0

2). Ассемблер: CLR <bit>

Код: 2 байта **<11000010> <bit address>**

Время:1 цикл

Алгоритм:(bit):=0

Пример: ; (P1)=5EH (01011110B)

CLR P1.3 ; (P1)=56H (01010110B)

Команда CPL A

Команда “инверсия аккумулятора” каждый бит аккумулятора инвертирует (изменяет на противоположный). Биты, содержащие “единицы”, после этой команды будут содержать “нули”, и наоборот. На флаги эта операция не влияет.

Ассемблер: CPL A

Код: 1 байт **<11110100>**

Время: 1 цикл Алгоритм: (A):=/(A) Пример: ;(A)=65H (01100101B)

CPL A ;(A)=9AH (10011010B)

Команда CPL <bit>

Команда “инверсия бита” инвертирует (изменяет на противоположное значение) указанный бит. Бит, который был “единицей”, изменяется в “нуль”, и наоборот. Команда CPL может работать с флагом переноса или с любым прямоадресуемым битом. На другие флаги команда не влияет.

1). Ассемблер: CPL <bit>

Код: 2 байта **<10110010> <bit address>**

Время: 1 цикл

Алгоритм:(bit):=/(bit)

Пример: ;(P1)=39H (00111001B)

CPL P1. 1

CPL P1.3 ;(P1)=33H (00110011B)

2). Ассемблер: CPL C

Код: 1 байт **<10110011>**

Время:1 цикл

Алгоритм:(C):=/(C)

Пример: ;(C)=0, (AC)=1, (OV)=0

CPL C ;(C)=1, (AC)=1, (OV)=0

Примечание. Если эта команда используется для изменения информации на выходе порта, значение, используемое как исходные данные, считывается из “защелки” порта, а не с выводов БИС.

Команда DA A

Команда “десятичная коррекция аккумулятора для сложения” упорядочивает 8-битовую величину в аккумуляторе после выполненной ранее команды сложения двух переменных (каждая в упакованном двоично-

десятичном фор-мате). Для выполнения сложения может использоваться любая из типов ко-манд ADD или ADDC. Если значение битов 3—0 аккумулятора (A) превышает 9 (XXXX 1010—XXXX 1111) или, если флаг AC равен “I”, то к содержимому (A) прибавляется 06 и получается соответствующая двоично-десятичная цифра в младшем полубайте. Это внутреннее побитовое сложение устанавливает флаг переноса, если перенос из поля младших четырех бит распространяется через все старшие биты, а в противном случае — не изменяет флаг переноса. Если после этого флаг переноса равен “I”, или если значение четырех старших бит (7—4) Превышает 9 (1010 XXXX — 1111 XXXX), значения этих старших бит увеличивается на 6 и создаётся соответствующая двоично-десятичная цифра в старшем полубайте. И снова при этом флаг переноса устанавливается, если перенос получается из старших битов, но не изменяется в противном случае. Таким образом, флаг переноса указывает на то, что сумма двух исходных двоично-десятичных переменных больше, чем 100. Эта команда выполняет десятичное преобразование с помощью сложения 06, 60, 66 с содержимым аккумулятора в зависимости от начального состояния аккумулятора и слова состояния программы (PSW).

Ассемблер: DA A

Код: 1 байт **<11010100>**

Время: 1 цикл

Алгоритм: если((A[3-0])>9 или (AC)=1),то A[3-0]:=A(3-0]+6

если((A[7-4])>9 или (0=1),то A[7-4]:=A[7-4]+6

Примеры:

```
а)              ;(A)=56H, (R3)=67H, (C)=1
ADDC   A,R3
DA   A                ;(A)=24H, (R3)=67H, (C)=1
б)              ;(A)=30H, (C)=0
ADD A, #99H
```

DA A;(A)=29, (C)=1

Примечание. Команда DA A не может просто преобразовать шестнадцатиричное значение в аккумуляторе в двоично-десятичное представление, и она не применяется, например, для десятичного вычитания.

Команда DEC <байт>

Команда “декремент” производит вычитание “1” из указанного операнда. Начальное значение 00H перейдет в 0FFH. Команда DEC не влияет на флаги. Этой командой допускается четыре режима адресации операнда:

1) к аккумулятору;
2) регистровый;
3) прямой;
4) косвенно-регистровый.

1). Ассемблер: DEC A

Код: 1 байт **<00010100>**

Время:1 цикл

Алгоритм:(A):=(A)-1

Пример: ;(A)=11H, (0=1, (AC)=1

DEC A ;(A)=10H, (0=1, (AC)=1

2). Ассемблер: DEC Rn ; где n=0-7

Код: 1 байт **<00011 rrr >**, где rrr=000-111

Время:1 цикл

Алгоритм:(Rn):=(Rn)-l

Пример: ;(R1)=7FH,

;(ОЗУ[7F])=40H, (ОЗУ[7 F])=00H

DEC @R1

DEC Rl

DEC @R1 ;(R1)=7EH, ;(03У[7F])=3FH. (03У(7F])=FFH

3). Ассемблер: DEC <direct>

Код: 2 байта **<00010101> <direct address>**

Время:1 цикл

Алгоритм:(direct):=(direct)-l

Пример: ;(SCON)=A0H, (0=1, (AC)=1

DEC SCON ;(SCON)=9FH, (0=1. (AC)=1

4). Ассемблер: DEC @Ri ; где i=0,1

Код: 1 байт **<000101 1 i >**

Время: 1 цикл

Алгоритм: ((Ri)):=((Ri)-l)

Пример: ;(R1)=7FH,

;(ОЗУ(7Р])=40Н, (ОЗУ[7Р1)=00Н DEC @R1

DEC Rl

DEC @R1 ;(R1)=7EH, ;(03У[7F})=3FH, (03У[7F])=FFH

Примечание. Если эта команда используется для изменения информации на выходе порта, значение, используемое как исходные данные, считывается из “защелки”’ порта, а не с выводов БИС.

Команда DIV AB

Команда “деление” делит 8-битовое целое без знака из аккумулятора А на 8-битовое целое без знака в регистре В. Аккумулятору присваивается целая часть частного (старшие разряды), а регистру В — остаток. Флаги переноса (С) и переполнения (OV) сбрасываются. Если (А) < (В), то флаг дополнительного переноса (АС) не сбрасывается. Флаг переноса сбрасывается в любом случае.

Ассемблер: DIV AB

Код: 1 байт **<100001 00 >**

Время:4 цикла

Алгоритм: (А):=((А)/(В))Ц5-8],

(В):=((А)/(В))[7-0]

Пример: Пусть аккумулятор содержит число 251_{10} (0FBH или 11111011B), а регистр B — число 18 (12H или 00010010B). После выполнения команды DIV AB в аккумуляторе будет число 13 (0DH или 00001101B), а в регистре B— число 17 (11H или 00010001B), т.к. 251=(13*18)+17. Флаги C и OV будут сброшены.

Примечание. Если B содержит 00, то после команды DIV содержимое аккумулятора A и регистра B будут не определены. Флаг переноса сбрасывается, а флаг переполнения устанавливается в “1”.

Команда DJNZ < байт >, < смешение >

Команда “декремент и переход, если не равно нулю” выполняет вычитание “1” из указанной ячейки и осуществляет ветвление по вычисляемому адресу, если результат не равен нулю. Начальное значение 00H перейдет в 0FFH. Адрес перехода (ветвления) вычисляется сложением значения смещения (со знаком), указанного в последнем байте команды, с содержимым счетчика команд, увеличенным на длину команды DJNZ. На флаги эта команда не влияет и допускает следующие режимы адресации:

1) регистровый;
2) прямой.

1) Ассемблер: DJNZ Rn,<метка> ; где n=0-7

Код: 2 байта **<11011 rrr > <rel8 >**, где rrr=000-111

Время: 2 цикла

Алгоритм: (PC):=(PC)+2, (Rn):=(Rn)-l,

если ((Rn)>0 или (Rn)<0), то (PC):=(PC)+<rel8>

Пример: ;(R2)=08H, (P1)=0FFH (11111111B)

LAB4: CPL PI.7

DJNZ R2.LAB4 ;(R2)={07-00}

;Эта последовательность команд переключает PI.7

;восемь раз и приводит к появлению четырех импульсов

;на выводе БИС, соответствующем биту P1.7. 2).

2) Ассемблер: DJNZ <direct>, <метка>

Код: 3 байта **<11010101> <direct address> <rel8 >**

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,

(direct):=(direct)-l,

если ((direct)>0 или (direct)<0), то (PC):=(PC)+<rel8>

Пример: ;(ОЗУ[40])=01H, (ОЗУ[50])=80H,

;(ОЗУ[60])=25H

DJNZ 40H, LAB1 ;(ОЗУ[40]) =00H

DJNZ 50H, LAB2 ;(ОЗУ[50]) =7FH

DJNZ 60H, LAB3 ;(ОЗУ[601) =25H

LAB1: CLR A

LAB2: DEC R1 ;осуществился переход на

;метку LAB2

Примечание. Если команда DJNZ используется для изменения выхода порта, значение используемое как операнд, считывается из “защелки” порта, а не с выводов БИС.

Команда INC <байт>

Команда “инкремент” выполняет прибавление “1” к указанной переменной и не влияет на флаги. Начальное значение 0FFH перейдет в 00H. Эта команда допускает четыре режима адресации:

1) к аккумулятору;
2) регистровый;
3) прямой;
4) косвенно-регистровый.

1). Ассемблер: INC A

Код: 1 байт **<000001 00 >**

Время:1 цикл

Алгоритм:(A):=(A)+1

Пример: ;(A)=1FH, (AC)=0

INC A ;(A)=20H, (AC)=0

2). Ассемблер: INC Rn ; где n=0-7

Код: 1 байт **<11011 rrr >** , где rrr=000-lll

Время:1 цикл

Алгоритм:(Rn):=(Rn)+l

Пример: ;(R4)=FFH, (C)=0, (AC)=0

INC R4 ;(R4)=00H. (C)=0, (AC)=0

3). Ассемблер: INC <direct>

Код: 2 байта **<00000101> <direct address>**

Время:1 цикл

Алгоритм:(direct):=(direct)+l

Пример: ;(ОЗУ[43H])=22H

INC 43H ;(ОЗУ[43H])=23H

4). Ассемблер: INC @Ri ; где i=0,l

Код: 1 байт **<0000011 i >**, где i=0,l

Время:1 цикл

Алгоритм:((Ri)):=((Ri))+l

Пример: ;(R1)=41H, (ОЗУ[41H])=4FH, (AC)=0

INC @R1 ;(R1)=41H, (ОЗУ[41H])=50H, (AC)=0

Примечание. При использовании команды INC для изменения содержимого порта, величина, используемая как операнд, считывается из “защелки” порта, а не с выводов БИС.

Команда INC DPTR

Команда “инкремент указателя данных” выполняет инкремент (прибавление “I”) содержимого 16-битового указателя данных (DPTR).

Прибавление “1” осуществляется к 16 битам, причем переполнение младшего байта указателя данных (DPL) из FFH в 00H приводит к инкременту старшего байта указателя данных (DPH). На флаги эта команда не влияет.

Ассемблер: INC DPTR

Код: 1 байт **<10100011 >**

Время:2 цикла

Алгоритм:(DPTR):=(DPTR)+1

Пример: ;(DPH)=12H, (DPL)=0FEH

INC DPTR

INC DPTR

INC DPTR ;(DPH)=13H, (DPL)=01H

Команда JB <bit>,<rel8>

Команда “переход, если бит установлен” выполняет переход по адресу ветвления, если указанный бит равен”1”, в противном случае выполняется следующая команда. Адрес ветвления вычисляется с помощью прибавления относительного смещения со знаком в третьем байте команды (rel8) к содержимому счетчика команд после прибавления к нему 3. Проверяемый бит не изменяется. Эта команда на флаги не влияет.

Ассемблер: JB (bit),<метка>

Код: 3 байта **<00100000> <bit address> <rel8 >**

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,

если (bit)=l, то (PC):=(PC)+<re18>

Пример: ;(A)=96H (10010110B)

JB ACC.2,LAB5 ;эта команда обеспечивает переход

;на метку LAB5

LAB5: INC A

Команда JBC <bit>, <re18>

Команда “переход, если бит установлен и сброс этого бита”, выполняет ветвление по вычисляемому адресу, если бит равен “I”. В противном случае выполняется следующая за JBC команда. В любом случае указанный бит сбрасывается. Адрес перехода вычисляется сложением относительного смещения со знаком в третьем байте команды (rel8) и содержимого счетчика команд по- сле прибавления к нему 3. Эта команда не влияет на флаги.

Ассемблер: JBC (bit),<метка>

Код: 3 байта **<00010000> <bit address> <rel8 >**

Время: 2 цикла

Алгоритм: (PC):=(PC)+3,

если (bit)=l, то (bit):=0, (PC):=(PC)+<rel8>

Пример: (A)=76H (0111 0110B)

```
JBC   ACC.3,LAB6        ; Перехода на LAB6 нет, т.к.
; (A[3]) = 0
JBC   ACC.2,LAB7        ; (A)=72H (0111 0010B) и переход
; на адрес, соответствующий  метке LAB7.
```

Примечание. Если эта команда используется для проверки бит порта, то значение, используемое как операнд, считывается из “защелки” порта, а не с вывода БИС.

Команда JC <re18>

Команда “переход, если перенос установлен” выполняет ветвление по адресу, если флаг переноса равен “I”, в противном случае выполняется следующая команда. Адрес ветвления вычисляется с помощью сложения относительного смещения со знаком во втором байте команды (re18) и содержимого счетчика команд после прибавления к нему 2. Эта команда на флаги не влияет.

Ассемблер: JC <метка>

Код: 2 байта **<01000000> <rel8 >**

Время:2 цикла

Алгоритм:(PC):=(PC)+2

если (C)=1, то (PC):=(PC)+<re18>

Пример: ;(C)=0

JC LAB8 ;нет перехода на метку LAB8

CPL C ;(C):=1

LAB8: JC LAB9 ;переход на метку LAB9, т.к. (C)=1

LAB9: NOP

Команда JMP @A+DPTR

Команда “косвенный переход” складывает 8-битовое содержимое аккумулятора без знака с 16-битовым указателем данных (DPTR) и загружает полученный результат в счетчик команд, содержимое которого является адресом для выборки следующей команды; 16-битовое сложение выполняется по модулю 2, перенос из младших восьми бит распространяется на старшие биты программного счетчика. Содержимое аккумулятора и указателя данных не изменяется. Эта команда на флаги не влияет.

Ассемблер: JMP @A+DPTR

Код: 1 байт **<01110011>**

Время:2 цикла

Алгоритм:(PC):=(A)l7-0]+(DPTR)[15-0]

Пример: ;(PC)=034EH, (A)=86H, (DPTR)=0329H

JMP @A+DPTR ;(PC)=03AFH, (A)=86H, (DPTR)=0329H

Команда JNB (bit),<метка>

Команда “переход, если бит не установлен” выполняет ветвление по

адресу, если указанный бит равен “нулю”, в противном случае выполняется следующая команда. Адрес ветвления вычисляется с помощью сложения относительного смещения со знаком в третьем байте команды (re18) и содержимого счетчика команд после прибавления к нему 3. Проверяемый бит не изменяется. Эта команда на флаги не влияет.

Ассемблер: JNB (bit),<метка>

Код: 3 байта **<00110000> <bit address> <re18>**

Время:2 цикла

Алгоритм:(PC):=(PC)+3,

если (bit)=0, то (PC):=(PC)+<re18>

Пример: ;(P1)=0CAH (11001010B),

;(A)=56H (0101 0110B)

JNB P1.3,LAB10 ;нет перехода на LAB10

JNB ACC.3,LAB11 ;переход на метку LAB11

LAB11: INC A

Команда JNC <re18>

Команда “переход, если перенос не установлен” выполняет ветвление по адресу, если флаг переноса равен нулю, в противном случае выполняется следующая команда. Адрес ветвления вычисляется с помощью сложения относительного смещения со знаком во втором байте команды (re18) и содержимого счетчика команд после прибавления к нему 2. Флаг переноса не изменяется. Эта команда на другие флаги не влияет.

Ассемблер: JNC <метка>

Код: 2 байта **<01010000> <rel8 >**

Время:2 цикла

Алгоритм:(PC):=(PC)+2,

если (C=0, то (PC):=(PC)+<re18>

Пример: ;(C)=1

JNC LAB12 ;нет перехода на LAB12 CPL C

LAB12: JNC LAB13 ;переход на метку LAB13

Команда JNZ <метка>

Команда “переход, если содержимое аккумулятора не равно нулю” выполняет ветвление по адресу, если хотя бы один бит аккумулятора равен “I”, в противном случае выполняется следующая команда. Адрес ветвления вычисляется сложением относительного смещения со знаком во втором байте команды (re18) и содержимого счетчика команд (PC) после прибавления к нему 2. Содержимое аккумулятора не изменяется. Эта команда на флаги не влияет.

Ассемблер: JNZ <метка>

Код: 2 байта **<01110000> <rel8 >**

Время:2 цикла

Алгоритм:(PC):=(PC)+2, если (A) ≠ 0 то (PC):=(PC)+<re18>

Пример: ;(A)=00H

JNC LAB14 ;нет перехода на LAB14

INC A

LAB14: JNZ LAB15 ;переход на метку LAB15

LAB15: NOP

Команда JZ <метка>

Команда “переход, если содержимое аккумулятора равно “0” выполняет ветвление по адресу, если все биты аккумулятора равны “0”, в противном случае выполняется следующая команда. Адрес ветвления вычисляется сложением относительного смещения со знаком во втором байте команды (re18) и содержимым счетчика команд после прибавления к нему 2. Содержимое аккумулятора не изменяется. Эта команда на флаги не влияет.

Ассемблер: JZ <метка>

Код: 2 байта **<01100000> <rel8 >**

Время: 2 цикла

Алгоритм:(PC):=(PC)+2, если (A)= 0 то (PC):=(PC)+<re18>

Пример: ;(A)=01H

JZ LAB16 ;нет перехода на LAB14

DEC A

JZ LAB17 ;переход на метку LAB17

LAB17: CLR A

Команда LCALL <метка>

Команда “длинный вызов” вызывает подпрограмму, находящуюся по указанному адресу. По команде LCALL к счетчику команд (PC) прибавляется 3 для получения адреса следующей команды и после этого полученный 16-битовый результат помещается в СТЕК (сначала следует младший байт, за ним — старший), а содержимое указателя СТЕКа (SP) увеличивается на 2. Затем старший и младший байты счетчика команд загружаются соответственно вторым и третьим байтами команды LCALL. Подпрограмма, следовательно, может начинаться в любом месте адресного пространства памяти программ объемом до 64 Кбайт. Эта команда на флаги не влияет.

Ассемблер: LCALL <метка>

Код: 3 байта **<00010010> <A15….A8> <A7….A0>**

Время: 2 цикла

Алгоритм: (PC):=(PC)+3

(SP):=(SP)+1 ((SP)):=(PC[7-0l) (SP):=(SP)+1 ((SP)):=(PC[15-8l) (PC):=<adddr[15-0]>

Пример: (SP)=07H,

; метке PRN соответствует адрес 1234H,

; по адресу 0126H находится команда LCALL

LCALL PRN ; (SP)=09H, (PC)=1234H,

; (ОЗУ[08])=26H, (ОЗУ[09])=01H

Команда LJMP <addrl6>

Команда “длинный переход” выполняет безусловный переход по указанному адресу, загружая старший и младший байты счетчика команд (PC) соответственно вторым и третьим байтами, находящимися в коде команды. Адрес перехода, таким образом, может находиться по любому адресу пространства памяти программ в 64 Кбайт. Эта команда на флаги не влияет.

Ассемблер: LJMP <метка>

Код: 3 байта **<00010010> <A15….A8> <A7….A0>**

Время: 2 цикла

Алгоритм: (PC):=<addr(15-0]>

Команда MOV <байт-назначения>,<байт-источника>

Команда “переслать переменную-байт” пересылает байт — переменную, указанную во втором операнде, в ячейку, указанную в первом операнде. Содержимое байта источника не изменяется. Эта команда на флаги и другие регистры не влияет. Команда “MOV” допускает 15 комбинаций адресации байта-источника и байта-назначения.

1). Ассемблер: MOV A,Rn ; где n=0-7

Код : 1 байт **<11101 rrr>,** где rrr=000-lll

Время :1 цикл

Алгоритм :(A):=(Rn)

Пример: ;(A)=0FAH, (R4)=93H

MOV A,R4 ;(A)=93H, (R4)=93H

2). Ассемблер : MOV A,<direct>

Код: 2 байта **<11100101> <direct address>**

Время :1 цикл

Алгоритм : (A):=(direct)

Пример: ;(A)=93H, (ОЗУ[40])=10H, (R0)=40H

MOV A,40H ;(A)=10H, (ОЗУ[40])=10H, (R0)=40H

3). Ассемблер : MOV A,@Ri ; где i=0,l

Код: 1 байт **<1110011 i >**

Время :1 цикл

Алгоритм : (A):=((Ri))

Пример: ;(A)=10H, (R0)=41H, (ОЗУ[41])=0CAH

MOV A,@R0 ;(A)=0CAH, (R0)=41H, (ОЗУ[41])=0CAH

4). Ассемблер : MOV A,#data

Код: 2 байта **<01110100> <#data >**

Время :1 цикл

Алгоритм : (A):=<#data8>

Пример: ;(A)=0C9H (11001001B)

MOV A,#37H ;(A)=37H (00110111B)

5). Ассемблер : MOV Rn,A ; где n=0-7

Код: 1 байт **<11111rrr>,** где rrr=000-lll

Время :1 цикл

Алгоритм :(Rn):=(A)

Пример: ;(A)=38H,(R0)=42H

MOV R0,A ;(A)=38H,(R0)=38H

6). Ассемблер: MOV Rn,<direct> ;где n=0-7

Код: 2 байта **<1 0 1 0 1 rrr> <direct address>,** где rrr=000-lll

Время:2 цикла

Алгоритм:(Rn):=(direct)

Пример: ;(R0)=39H, (P2)=0F2H

MOV R0,P2 ;(R0)=0F2H 7)

7). Ассемблер: MOV Rn,#data ; где n=0-7

Код: 2 байта **<1 1 1 1 1 rrr > <#data8>,** где rrr=000-lll

Время:1 цикл

Алгоритм:(Rn):=<#data8>

Пример: ;(R0)=0F5H

MOV R0,#49H ;(R0)=49H

8). Ассемблер: MOV <direct>,A

Код: 2 байта **<1 1 1 1 0 1 0 1> <direct address>**

Время:1 цикл

Алгоритм:(direct):=(A)

Пример: ;(P0)=0FFH,(A)=4BH

MOV P0,A ;(P0)=4BH,(A)=4BH

9). Ассемблер: MOV <direct>,Rn ; где n=0-7

Код: 2 байта **<1 0 0 0 1 rrr> <direct address>** , где rrr=000-lll

Время:2 цикла

Алгоритм: (direct):=(Rn)

Пример: ;(PSW)=0C2H, (R7)=57H

MOV PSW,R7 ;(PSW)=57H, (R7)=57H

10). Ассемблер: MOV <direct>,<direct>

Код: 3 байта **<1 0 0 0 0 1 0 1> < direct address> < direct address>**

Время:2 цикла

Алгоритм:(direct):=(direct)

Пример: ;(ОЗУ[45])=33H, (ОЗУ 148])=0DEH

MOV 48H,45H ;(ОЗУ[45])=33H, (ОЗУ[48])=33H

11). Ассемблер: MOV <direct>,@Ri ; где i=0,l

Код: 2 байта **<1 0 0 0 0 1 1 i > <direct address>**

Время:2 цикла

Алгоритм: (direct):=((Ri))

Пример: ;(R1)=49H, (ОЗУ[49])=0E3H

MOV 51H,@R1 ;(ОЗУ[51])=0E3H, (ОЗУ[49])=0E3H

12). Ассемблер: MOV <direct>,#data

Код: 3 байта **<0 1 1 1 0 1 0 1> < direct address> <#data8 >**

Время:2 цикла

Алгоритм:(direct):=<#data8>

Пример: ;(ОЗУ[5F])=9BH

MOV 5FH,#07H ;(ОЗУ[5F])=07H

13). Ассемблер: MOV @Ri,A, где i=0,1

Код: 1 байт **<1 1 1 1 0 1 1 1 >**, где i =0,1

Время:1 цикл

Алгоритм:((Ri)):=(A)

Пример: ;(R1)=48H, (ОЗУ[48])=75H, (A)=0BDH

MOV @R1,A ; ОЗУ[48])=0BOH

14). Ассемблер: MOV @Ri,<direct> ; где i=0,l

Код: 2 байта **<1 0 1 0 0 1 1 1> <direct address >**

Время:2 цикла

Алгоритм:((Ri)):=(direct)

Пример: ;(R0)=51H,(ОЗУ[51H])=0E3H,

(P0)=0ACH MOV @R0,P0 ;(ОЗУ[51H])=0ACH

15). Ассемблер: MOV @Ri,#data ; где i=0,1

Код: 2 байт **<0 1 1 1 0 1 1 i > <#data8>**

Время:1 цикл

Алгоритм: ((Ri)):=<#data8>

Пример: ;(ОЗУ[7EH])=67H,(R1)=7EH

MOV @R1, #0A9H ;(ОЗУ[7E])=0A9H, (R 1)=7EH

Команда MOV <бит назначения>, <бит источника>

Команда “переслать бит данных” битовую переменную, указанную во втором байте, копирует в разряд, который указан в первом операнде. Одним из операндов должен быть флаг переноса C, а другим может быть любой бит, к которому возможна прямая адресация.

1). Ассемблер: MOV C,<bit>

Код: 2 байта **<1 0 1 0 0 0 1 0> <bit address>**

Время:1 цикл

Алгоритм: (C):=(bit)

Пример: ;(C)=0, (P3)=0D5H (11010101B)

MOV C,P3.0 ;C:=1

MOV C,P3.3 ;C:=0

MOV C,P3.7 ;C:=1

2). Ассемблер: MOV <bit>,C

Код: 2 байта <1 0 0 1 0 0 1 0> <bit address>

Время:2 цикла

Алгоритм:(bit):=(C)

Пример: ;(C)=1,(P0)=20H (00100000B)

MOV P0.1,C

MOV P0.2,C

MOV P0.3,C ;(C)=1,(P0)=2EH (00101110B)

Команда MOV DPTR,#datal6

Команда “загрузить указатель данных 16-битовой константой” загружает указатель данных DPTR 16-битовой константой, указанной во втором и третьем байтах команды. Второй байт команды загружается в старший байт указателя данных (DPH), а третий байт — в младший байт указателя данных (DPL). Эта команда на флаги не влияет и является единственной командой, которая одновременно загружает 16 бит данных.

Ассемблер: MOV DPTR,#<datal6>

Код: 3 байта **<1 0 0 1 0 0 0 0 > <#data[l5-8]> <#data[7-0]>**

Время:2 цикла

Алгоритм: (DPTR):=#data[15-0],

причем DPH:=#data115-8], DPL:=#data[7-0]

Пример: ;(DPTR)=01FDH

MOV DPTR,#1234H ;(DPTR)=1234H, ;(DPH)=12H, (DPL)=34H

Команда MOVC A,@A+(<R16>)

<R16> — 16-разрядный регистр.

Команда “переслать байт из памяти программ” загружает аккумулятор байтом кода или константой из памяти программы. Адрес считываемого байта вычисляется как сумма 8-битового исходного содержимого аккумулятора без знака и содержимого 16-битового регистра. В качестве 16-битового регистра может быть:

1) указатель данных DPTR;
2) счетчик команд PC.

В случае, когда используется PC, он увеличивается до адреса следующей команды перед тем как его содержимое складывается с содержимым аккумулятора; 16-битовое сложение выполняется так, что перенос из младших восьми бит может распространяться через старшие биты. Эта команда на флаги не влияет.

1). Ассемблер: MOVC A,@A+DPTR

Код: 1 байт **<1 0 0 1 0 0 1 1>**

Время:2 цикла

Алгоритм: (A):=((A)+(DPTR))

Пример: ;(A)=1BH, (DPTR)=1020H,

;(ПЗУ[103BH])=48H,

MOVC A,@A+DPTR ;(A)=48H, (DPTR)=1020H

2). Ассемблер: MOVC A,@A+PC

Код: 1 байт **<1 0 0 0 0 0 1 1>**

Время:2 цикла

Алгоритм:(A):=((A)+(PC))

Пример: ;(A)=0FAH, (PC)=0289H, (ПЗУ[0384H])=9BH

MOVC A,@A+PC ;(A)=9BH, (PC)=028AH

Команда MOVX <байт приемника >,< байт источника>

Команда “переслать во внешнюю память (из внешней памяти) данных” пересылает данные между аккумулятором и байтом внешней памяти данных. Имеется два типа команд, которые отличаются тем, что обеспечивают 8-битовый или 16-битовый косвенный адрес к внешнему ОЗУ данных.

В первом случае содержимое R0 или R1 в текущем банке регистров обеспечивает 8-битовый адрес, который мультиплексируется с данными порта P0. Для расширения дешифрации ввода-вывода или адресации небольшого массива ОЗУ достаточно восьми бит адресации. Если применяются ОЗУ, немного больше, чем 256 байт, то для фиксации старших битов адреса можно использовать любые другие выходы портов, которые переключаются командой, стоящей перед командой MOVX.

Во втором случае, при выполнении команды MOVX указатель данных DPTR генерирует 16-битовый адрес. Порт P2 выводит старшие восемь бит адреса (DPH), а порт P0 мультиплексирует младшие 8 бит адреса (DPL) с данными. Эта форма является эффективной при доступе к большим массивам данных (до 64К байт), так как для установки портов вывода не требуется дополнительных команд.

1). Ассемблер: MOVX A,@Ri ; где i=0,1

Код: 1 байт **< 1 1 1 1 0 0 0 1 i>**

Время:2 цикла

Алгоритм: (A):=((Ri))

Пример: ;(A)=32H, (R0)=83H, ячейка

; внешнего ОЗУ по адресу 83H

; содержит 0B6H

MOVX A,@R0 ; (A)=0B6H, (R0)=83H

2). Ассемблер: MOVX A, @DPTR

Код: 1 байт **<1 1 1 0 0 0 0 0>**

Время :2 цикла

Алгоритм: (A):=((DPTR))

Пример: ;(A)=5CH,(DPTR)=1ABEH,

;ячейка внешнего ОЗУ по адресу

;1ABEH содержит 72H

MOVX A,@DPTR ;(A)=72H, (DPTR)=1ABEH

3). Ассемблер: MOVX @Ri,A ; где i=0,1

Код: 1 байт **<1 1 1 1 0 0 1 i>**

Время: 2 цикла

Алгоритм : ((Ri)):=(A)

Пример: ;(A)=95H, (R1)=0FDH,

; ячейка внешнего ОЗУ с адресом

; 0FDH содержит 00H

MOVX @Ri,A ;(A)=95H , (R1)=0FDH,

;ячейка внешнего ОЗУ с адресом

; 0FDH содержит 95H

4). Ассемблер: MOVX @DPTR,A

Код: 1 байт **<1 1 1 1 0 0 0 0>**

Время:2 цикла

Алгоритм: ((DPTR)) := (A)

Пример: ;(A)=97H, (DPTR)=1FFFH,

;ячейка внешнего ОЗУ с адресом

;1FFFH содержит 00H

MOVX @DPTR,A ;(A)=97H ,

;ячейка внешнего ОЗУ с адресом

;1FFFH содержит 97H

Команда MUL AB

Команда “умножение” умножает 8-битовые целые числа без знака из аккумулятора и регистра B. Старший байт 16-битового произведения

помещается в регистр B, а младший — в аккумулятор A. Если результат произведения больше, чем 0FFH(255_{10}), то устанавливается флаг переполнения (OV), в противном случае он сбрасывается. Флаг переноса всегда сбрасывается.

Ассемблер: MUL AB

Код: 1 байт **<1 0 1 0 0 1 0 0>**

Время: 4 цикла

Алгоритм: (A)[7-0]=(A)*(B),

(B)[15-8]=(A)*(B)

Пример: a) ;(A)=50H (50H=80_{10}), (C)=1,

;(B)=0A0H (0A0H=160_{10}), (OV)=0

MUL AB ;(A)=00H, (B)=32H, (C)=0, (OV)=1

b) ;(A)=24H, (OV)=1, (B)=06H, (C)=1

MUL AB ;(A)=0D8H, (B)=00H, (OV)=0, (C)=0

Команда NOP

Команда “нет операции” выполняет холостой ход и не влияет на регистры и флаги, за исключением счетчика команд (PC).

Ассемблер: NOP

Код: 1 байт **<0 0 0 0 0 0 0 0>**

Время: 1 цикл

Алгоритм: (PC):=(PC)+1

Пример: Пусть требуется создать отрицательный выходной импульс на линии 6 порта P1 длительностью 3 цикла. Это выполнит следующая последовательность команд:

```
CLR  P1.6      ;P1[6]:=0
NOP
NOP
NOP
```

SETB P1.6 ;P1(6]:=1

Команда ORL <байт назначения>,<байт источника>

Команда “логическое “ИЛИ” для переменных-байтов” выполняет операцию логического “ИЛИ” над битами указанных переменных и записывает результат в байт назначения. Эта команда на флаги не влияет. Допускается шесть комбинаций режимов адресации:

- если байтом назначения является аккумулятор:
  1) регистровый;
  2) прямой;
  3) косвенно-регистровый;
  4) непосредственный;
- если байтом назначения является прямой адрес:
  5) к аккумулятору;
  6) к константе;

1). Ассемблер: ORL A,Rn ; где n =0-7

Код: 1 байт **<0 1 0 0 0 1 rrr>** , где rrr = 000-l11

Время:1 цикл

Алгоритм: (A) : = (A) OR (Rn), где OR — операция логического “ИЛИ”

Пример: ; (A)=15H, (R5)=6CH

ORL A,R5 ;(A)=7DH, (R5)=6CH

2). Ассемблер: ORL A,<direct>

Код: 2 байта **< 0 1 0 0 0 1 0 1 >** **<direct address>**

Время:1 цикл

Алгоритм: (A):=(A) OR (direct)

Пример: ;(A)=84H, (PSW)=0C2H

ORL A,PSW ;(A)=0C6H, (PSW)=0C2H

3). Ассемблер: ORL A,@Ri ; где i=0,l

Код: 1 байт **<0 1 0 0 0 1 1 i>**

Время: 1 цикл

Алгоритм: (A):=(A) OR ((RD)

Пример: ;(A)=52H, (R0)=6DH, (03У[6DH])=49H

ORL A,@R0 ;(A)=5BH, (ОЗУ[6DH])=49H

4). Ассемблер: ORL A,#<data>

Код: 2байта **<0 1 0 0 0 1 0 0 > <#data8>**

Время:1 цикл

Алгоритм: (A):=(A) OR #<data>

Пример: ;(A)=0F0H

ORL A,#0AH ;(A)=0FAH

5). Ассемблер: ORL (direct),A

Код: 2 байта **<0 1 0 0 0 0 1 0 > <direct address>**

Время:1 цикл

Алгоритм: (direct):=(direct) OR (A)

Пример: ;(A)=34H, (IP)=23H

ORL IP,A ;(IP)=37H, (A)=34H

6). Ассемблер: ORL (direct),#<data>

Код: 3 байта **<0 1 0 0 0 0 1 1 > <direct address> <#data8>**

Время:2 цикла

Алгоритм: (direct):=(direct) OR #<data>

Пример: ;(P1)=00H

ORL P1,#0C4H ;(P1)=11000100B (0C4H)

Примечание. Если команда используется для работы с портом, величина, используемая в качестве исходных данных порта, считывается из “защелки” порта, а не с выводов БИС.

Команда ORL C,<бит источника>

Команда “логическое “ИЛИ” для переменных-битов” устанавливает флаг переноса С, если булева величина равна логической “I”, в противном случае

она устанавливает флаг С в “0”. Косая дробь (“/”) перед операндом на языке Ассемблера указывает на то, что в качестве операнда используется логическое отрицание значения адресуемого бита, но сам бит источника не изменяется. Эта команда на другие флаги не влияет.

1). Ассемблер: ORL C,<bit>

Код: 2 байта **<0 1 1 1 0 0 1 0 > <bit address>**

Время: 2 цикла

Алгоритм:(C):=(C) OR (bit)

Пример: ;(C)=0, (P1)=53H (01010011B)

ORL C,P1.4 ;(C)=1, (P1)=53H (01010011B)

2). Ассемблер: ORL C,/<bit>

Код: 2 байта **<1 0 1 0 0 0 0 0 > <bit address >**

Время:2 цикла

Алгоритм: (C):=(C) OR /(bit)

Пример: ;(C)=0, (ОЗУ[25H])=39H (00111001B)

ORL C,/2A ;(C)=1, (ОЗУ[25H])=39H (00111001B)

Команда POP <direct>

Команда “чтение из стека” считывает содержимое ячейки, которая адресуется с помощью указателя стека, в прямо адресуемую ячейку ОЗУ, при этом указатель стека уменьшается на единицу. Эта команда не воздействует на фла- ги и часто используется для чтения из стека промежуточных данных.

Ассемблер: POP <direct>

Код: 2 байта **<1 1 0 1 0 0 0 0 > <direct address>**

Время:2 цикла

Алгоритм:(direct):=((SP)), (SP):=(SP) — 1

Пример: ;(SP)=32H, (DPH)=01H, (DPL)=0ABH,

;(ОЗУ[32H])=12H, (ОЗУ[31H])=56H,

;(ОЗУ[30H])=20H

```
POP   DPH
POP   DPL          ;(SP)=30H, (DPH)=12H, (DPL)=56H,
;(ОЗУ[32H])=12H, (ОЗУ[31H])=56H
;(SP)=20H, (ОЗУ[30H])=20H
```

Команда PUSH <direct>

Команда “запись в стек” увеличивает указатель стека на единицу и после этого содержимое указанной прямо адресуемой переменной копируется в ячейку внутреннего ОЗУ, адресуемого с помощью указателя стека. На флаги эта команда не влияет и используется для записи промежуточных данных в стек.

Ассемблер: PUSH <direct>

Код: 2 байта **<1 1 0 0 0 0 0 0 > <direct address>**

Время:2цикла

Алгоритм: (SP):=(SP)+1, ((SP)):=(direct)

Пример:

```
;(SP)=09H, (DPTR)=1279H
PUSH   DPL
PUSH   DPH                    ; (SP)=0BH, (DPTR)=1279H
; (ОЗУ[0AH])=79H•, (ОЗУ[0BH])=12H,
```

Команда RET

Команда “возврат из подпрограммы” последовательно выгружает старший и младший байты счетчика команд из стека, уменьшая указатель стека на 2. Выполнение основной программы обычно продолжается по адресу команды, следующей за ACALL или LCALL. На флаги эта команда не влияет.

Ассемблер: RET

Код: 1 байт **<0 0 1 0 0 0 1 0>**

Время:2 цикла

Алгоритм: (PC)[15-8]:= ((SP)), (SP):=(SP)-1, (PC)[7-0]:= ((SP)), (SP):=(SP)-1

Пример: ; (SP)=0DH, (ОЗУ[0CH])=93H, (0ЗУ[0DH])=02H

RET ; (SP)=0BH, (PC)=0293H

Команда RETI

Команда “возврат из прерывания” выгружает старший и младший байты счетчика команд из стека и устанавливает “логику прерываний”, разрешая прием других прерываний с уровнем приоритета, равным уровню приоритета только что обработанного прерывания. Указатель стека уменьшается на 2. Слово состояния программы (PSW) не восстанавливается автоматически. Выполнение основной программы продолжается с команды, следующей за командой, на которой произошел переход к обнаружению запроса на прерывание. Если при выполнении команды RETI обнаружено прерывание с таким же или меньшим уровнем приоритета, то одна команда основной программы успевает выполниться до обработки такого прерывания.

Ассемблер: RETI

Код: 1 байт **<0 0 1 1 0 0 1 0>**

Время:2 цикла

Алгоритм: (PC)[15-8]:=((SP)), (SP):=(SP)-1,

(PC)(7-0]:=((SP)), (SP):=(SP)-1

Пример: ; (SP)=0BH, (ОЗУ[0AH])=2AH, (ОЗУ[0BH])=03H,

; (PC)=УУУУН, где У=0-FH

RETI ; (SP)=09H, (PC)=032AH

Команда RL A

Команда “сдвиг содержимого аккумулятора влево”, сдвигает восемь бит аккумулятора на один бит влево, бит 7 засылается на место бита 0. На флаги эта команда не влияет.

Ассемблер: RL A

Код: 1 байт **<0 0 1 0 0 0 1 1>**

Время:1 цикл

Алгоритм: (A[N+1]):=(A[N]), где N=0-6

(A[0]):=(A[7])

Пример: ; (A)=0D5H (11010101B), (C)=0

RL A ; (A)=0ABH (10101011B), (C)=0

Команда RLC A

Команда “сдвиг содержимого аккумулятора влево через флаг переноса” сдвигает восемь бит аккумулятора и флаг переноса влево на один бит. Содержимое флага переноса помещается на место бита 0 аккумулятора, а содержимое бита 7 аккумулятора переписывается в флаг переноса. На другие флаги эта команда не влияет.

Ассемблер: RLC A

Код: 1 байт **<0 0 1 1 0 0 1 1>**

Время:1 цикл

Алгоритм: (A[N+1]):=(A[N]), где N=0-6 (A[0]):=(C)

(C:=(A(7])

Пример: ; (A)=56H (01010110B), (C)=1

RLC A ; (A)=0ADH (10101101B), (C)=0

Команда RR A

Команда “сдвиг содержимого аккумулятора вправо” сдвигает вправо на один бит все восемь бит аккумулятора. Содержимое бита 0 помещается на место бита 7. На флаги эта команда не влияет.

Ассемблер: RR A

Код: 1 байт **< 0 0 0 0 0 0 1 1>**

Время: 1 цикл

Алгоритм: (A[N]) =(A[N+1]), где N=0-6

(A[7]):=(A[0])

Пример: (A)=0D6H (11010110B), (C)=l

RR A ; (A)=6BH (01101011B), (C)=1

Команда RRC A

Команда “сдвиг содержимого аккумулятора вправо через флаг переноса” сдвигает восемь бит аккумулятора и флаг переноса на один бит вправо. Бит 0 перемещается в флаг переноса, а исходное содержимое флага переноса помещается в бит 7. На другие флаги эта команда не влияет.

Ассемблер: RRC A

Код: 1 байт **<0 0 0 1 0 0 1 1>**

Время:1 цикл

Алгоритм: (A[N]):=(A[N+lj), где N=0-6

(A[7]):=(C), (C:=(A[0])

Пример: ; (A)=95H (10010101B), (C)=0

RRC A ; (A)=4AH (01001010B), (C)=1

Команда SETB <бит>

Команда “установить бит” устанавливает указанный бит в “1”. Адресуется:

1) к флагу переноса (C);
2) к биту с прямой адресацией.

1) Ассемблер: SETB C

Код: 1 байт **<1 1 0 1 0 0 1 1>**

Время:1 цикл

Алгоритм: (C):=1

Пример: ; (C)=0

SETB C ; (C)=1

2) Ассемблер: SETB (bit)

Код: 2 байта **< 1 1 .0 1 0 0 1 0>** **< bit address>**

Время:1 цикл

Алгоритм: (bit):=l

Пример: ; (P2)=38H (00111000B)

SETB P2.0

SETB P2.7 ; (P2)=0B9H (10111001B)

Команда SJMP <метка>

Команда “короткий переход” выполняет безусловное ветвление в программе по указанному адресу. Адрес ветвления вычисляется сложением смещения со знаком во втором байте команды с содержимым счетчика команд после прибавления к нему 2.

Таким образом, адрес перехода должен находиться в диапазоне от 128 байт, предшествующих команде, до 127 байт, следующих за ней.

Ассемблер: SJMP <метка>

Код: 2 байта **<1 0 0 0 0 0 0 0 > <re18>**

Время:2 цикла

Алгоритм: (PC):=(PC)+2,

(PC):=(PC)+(re18)

Пример: ; (PC)=0418H,

; метка MET1 соответствует адресу 039AH

SJMP MET1 ; (PC)=039AH, где (rel8)=80H=-128 $_{10}$

SJMP MET2 ; (PC)=041AH, где метка MET2 соответствует

; адресу 041AH,

; (rel8)=7DH=+125 $_{10}$

Команда SUBB A, <байт источника>

Команда “вычитание с заемом” вычитает указанную переменную вместе с флагом переноса из содержимого аккумулятора, засылая результат в

аккумулятор. Эта команда устанавливает флаг переноса (заема), если при вычитании для бита 7 необходим заем, в противном случае флаг переноса сбрасывается. Если флаг переноса установлен перед выполнением этой команды, то это указывает на то, что заем необходим при вычитании с увеличенной точностью на предыдущем шаге, поэтому флаг переноса вычитается из содержимого аккумулятора вместе с операндом источника. (AC) устанавливается, если заем необходим для бита 3 и сбрасывается в противном случае. Флаг переполнения (OV) устанавливается, если заем необходим для бита 6, но его нет для бита 7, или есть для бита 7, но нет для бита 6.

При вычитании целых чисел со знаком (OV) указывает на отрицательное число, которое получается при вычитании отрицательной величины из положительной, или положительное число, которое получается при вычитании положительного числа из отрицательного.

Операнд источника допускает четыре режима адресации:

1) регистровый;
2) прямой;
3) косвенно-регистровый;
4) непосредственный (к константе).

1) Ассемблер: SUBB A,Rn; где n=0-7

Код: 1 байт **<1 0 0 1 1 rrr>**, где r=000-111 Время: 1 цикл

Алгоритм: (A):=(A)-(C)-(Rn);

(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример: ; (A)=C9H, (R2)=54H, (C)=1

SUBB A,R2 ; (A)=74H, (R2)=54H, (0=0,

; (AC)=0, (OV)=1

2) Ассемблер: SUBB A,<direct>

Код: 2 байта <1 0 0 1 0 1 0 1> < direct address>

Время: 1 цикл

Алгоритм: (A):=(A)-(0-(direct);

(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример: ; (A)=97H, (B)=25H, (C) =0

SUBB A,B ; (A)=72H, (B)=25H, (C)=0,

; (AC)=0, (OV)=1

3) Ассемблер: SUBB A,@Ri; где i=0,l

Код: 1 байт **< 1 0 0 1 0 1 1 1>**

Время: 1 цикл

Алгоритм: (A):=(A)-(C)-((Ri));

(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример: ; (A)=49H, (C)=1, (R0)=33H,

; (ОЗУ[33H])=68H

SUBB A,@R0 ; (A)=E0H, (C)=1, (AC)=0, (OV)=0

4) Ассемблер SUBB A,#data

Код: 2 байта < 1 0 0 1 0 1 0 0> < #data8>

Время: 1 цикл

Алгоритм: (A):=(A)-(C)-(#data8);

(C):=X, (AC):=X, (OV):=X, где X=(0 или 1)

Пример: ; (A)=0BEH, (C)=0

SUBB A,#3FH ; (A)=7FH, (C)=0,(AC)=1, (OV)=1

Команда SWAP A

Команда “обмен тетрадами внутри аккумулятора” осуществляет обмен между младшими четырьмя и старшими четырьмя битами аккумулятора (между старшей и младшей тетрадами). Эта команда может рассматриваться так же, как команда четырех битового циклического сдвига. На флаги эта команда не влияет.

Ассемблер: SWAP A

Код: 1 1 0 0 0 1 0 0

Время: 1 цикл

Алгоритм: (A[3-0]):=(A[7-4]), (A[7-4П:=(A[3-0])

Пример: ; (A)=0D7H (il010111B)

SWAP A ; (A)=7DH (01111101B)

Команда XCH A,<байт>

Команда “обмен содержимого аккумулятора с переменной-байтом” осуществляет обмен содержимого аккумулятора с содержимым источника, указанным в команде. Операнд источника может использовать следующие режимы адресации:

1) регистровый;
2) прямой;
3) косвенно-регистровый.

1) Ассемблер: XCH A,Rn; где n=0-7

Код: 1 байт **<1 1 0 0 1 rrr>** , где rrr=000-lll Время:1 цикл

Алгоритм: (A):=(Rn), (Rn):=(A)

Пример: ; (A)=3CH , (R4)=15H

XCH A,R4 ; (A)=15H, (R4)=3CH

2) Ассемблер: XCH A,<direct>

Код: 2 байта **<1 1 0 0 0 1 0 1> < direct address>**

Время: 1 цикл

Алгоритм: (A):=(direct), (direct):=(A)

Пример: ; (A)=0FEH, (P3)=0DAH

XCH A,P3 ; (A)=0DAH, (P3)=0FEH

3) Ассемблер XCH A,@Ri , где i=0,l

Код: 1 байт <1 1 0 0 0 1 1 i>

Время: 1 цикл

Алгоритм: (A):=((Ri)), ((Ri)):=(A)

Пример: ; (R1)=39H, (ОЗУ[39])=44H, (A)=0BCH

XCH A,@R1 ; (ОЗУ[39H])=0BCH, (A)=44H

Команда XCHD A,@Ri

Команда “обмен тетрадой” выполняет обмен младшей тетрады (биты 3—0) аккумулятора с содержимым младшей тетрады (биты 3—0) ячейки внутреннего ОЗУ, косвенная адресация к которой производится с помощью указанного регистра. На старшие биты (биты 7—4) эта команда не влияет (так же, как и на флаги).

Ассемблер: XCHD A,@Ri; где i=0,1

Код: 1 байт **<1 1 0 1 0 1 1 i>**

Время:1 цикл

Алгоритм: (A(3-0]):=((Ri[3-0])),

((Ri[3-0])):=(A[3-0])

Пример: ; (R0)=55H, (A)=89H, (ОЗУ[55H])=0A2H

XCHD A,@R0 ; (A)=82H, (ОЗУ[55H])=0A9H

Команда XRL <байт назначениям>,< байт источника>

Команда “логическое “ИСКЛЮЧАЮЩЕЕ ИЛИ” для переменных-байтов” выполняет операцию “ИСКЛЮЧАЮЩЕЕ ИЛИ” над битами указанных переменных, записывая результат в байт назначения. На флаги эта команда не влияет.

Допускается шесть режимов адресации:

- байтом назначения является аккумулятор:
  1) регистровый;
  2) прямой;
  3) косвенно-регистровый;
  4) непосредственный
- байтом назначения является прямой адрес:
  5) к аккумулятору;

6) к константе.

1) Ассемблер: XRL A,Rn; где n=0-7

Код: 1байт **< 0 1 1 0 1 rrr >**, где rrr=000-111 Время: 1 цикл

Алгоритм: (A):=(A) XOR (Rn)

Пример: ; (A)=C3H, (R6)=0AAH

XRL A,R6 ; (A)=69H, (R6)=0AAH

Ассемблер: XRL A,<direct>

Код: 2 байта **< 0 1 1 0 0 1 0 1> < direct address>**

Время: 1 цикл

Алгоритм: (A):=(A) XOR (direct)

Пример: ; (A)=0FH, (P1)=0A6H

XRL A,PI ; (A)=A9H, (P1)=0A6H

3) Ассемблер: XRL A,@Ri; где i=0,l

Код: 1 байт **<0 1 1 0 0 1 1 1>**

Время: 1 цикл

Алгоритм: (A):=(A) XOR ((Ri))

Пример: ; (A)=55H,R1=77H, (ОЗУ[77H])=5AH XRL A,@R1

; (A)=0FH, (ОЗУ[77H])=5AH

4) Ассемблер: XRL A,#data

Код: 2 байта <0 1 1 0 0 1 0 0> <#data8>

Время: 1 цикл

Алгоритм: (A):=(A) XOR <data>

Пример: ; (A)=0C3H

XRL A,#0F5H ; (A)=36H

5) Ассемблер: XRL <direct>,A

Код: 2 байта **< 0 1 1 0 0 0 1 0 > <direct address>**

Время : 1 цикл

Алгоритм: (direct):=(direct) XOR (A)

Пример: ; (A)=31H, (P1)=82H

XRL PI,A ; (A)=31H, (P1)=0B3H

6) Ассемблер: XRL <direct>,#data

7) Код:3 байта **< 0 1 1 0 0 0 1 1> <direct address> <#data8>**

Время: 2 цикла

Алгоритм: (direct):=(direct) XOR #data

Пример: ; (IP)=65H

XRL IP,#65H ; (IP)=00H

Примечание. Если эта команда используется для работы с портами, то значение, используемое в качестве операнда, считывается из “защелки” порта, а не с выводов БИС.

6. МИКРОКОНТРОЛЛЕРЫ 8XC51FA, FB, FC

Основным отличием микроконтроллеров группы 8XC51FX от рассмотренного ранее в разделах 3 — 5 базового микроконтроллера семейства MCS-51 является наличие блока PCA (programmable counter array) — матрицы (массива) программируемых счетчиков. Это дополнительный блок ввода-вывода, предназначенный для выполнения различных операций счета и определения временных интервалов, в том числе при широтно-импульсной модуляции. Также в эти микроконтроллеры добавлен третий таймер T2. Микроконтроллеры 8XC51FX имеют расширенный набор регистров специальных функций, приведенный на рис. 6.1 (регистры, вновь введенные в сравнении с базовым микроконтроллером 8051для работы с матрицей PCA, отмечены серым фоном).

F8		CH 00000000	CCAP0H xxxxxxxx	CCAP1H xxxxxxxx	CCAP2H xxxxxxxx	CCAP3H xxxxxxxx	CCAP4H xxxxxxxx		FF
F0	B 00000000								F7
E8		CL 00000000	CCAP0L xxxxxxxx	CCAP1L xxxxxxxx	CCAP2L xxxxxxxx	CCAP3L xxxxxxxx	CCAP4L xxxxxxxx		EF
E0	ACC 00000000								E7
D8	CCON 00x00000	CMOD 00xxx000	CCAPM0 x0000000	CCAPM1 x0000000	CCAPM2 x0000000	CCAPM3 x0000000	CCAPM4 x0000000		DF
D0	PSW 00000000								D7
C8	T2CON 00000000	T2MOD 00000000	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			CF
C0									C7
B8	IP x0000000	SADEN 00000000							BF
B0	P3 11111111							IPH x0000000	B7
A8	IE 00000000	SADDR 00000000							AF
A0	P2 11111111								A7
98	SCON 00000000	SBUF xxxxxxxx							9F
90	P1 11111111								97
88	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8F
80	P0 11111111	SP 00000111	DPL 00000000	DPH 00000000				PCON 00x0000	87

Рисунок 6.1 — Регистры специального назначения микроконтроллеров 8XC5-

1FA,FB,FC

Назначение других регистров будет рассмотрено в соответствующих разделах.

Рассмотрим работу блока PCA подробнее.

6.1. Блок PCA

Блок PCA состоит из 16-разрядного таймера-счетчика и пяти 16-разрядных модулей фиксации-сравнения. Таймер-счетчик является источником временной базы и счетчиком событий, значения его текущего отсчета передаются в модули фиксации-сравнения. Счетный регистр таймера-счетчика состоит из пары регистров спецфункций CH-CL, а значения, зафиксированные модулями, хранятся в пяти парах регистров CCAPxH/CCAPxL. Управление работой и режимами таймера-счетчика осуществляется при помощи регистров спецфункций CMOD (регистр режима) и CCON (регистр управления). Режимы работы модулей сравнения-захвата определяются пятью регистрами CCAPMx. Рассмотрим назначение отдельных битов этих регистров.

Регистр CMOD

CIDL	WDTE	---	---	---	CPS1	CPS0	ECF

Регистр CCON

CF	CR	---	CCF4	CCF3	CCF2	CCF1	CCF0

Имя бита	Номер бита	Функция
Регистр CMOD		
CIDL	CMOD.7	Бит разрешения функционирования блока PCA в режиме Idle: CIDL=1 функционирование запрещено, CIDL=0 функционирование разрешено (Idle — x.x.)
WDTE	CMOD.6	Бит разрешения функции сторожевого таймера

		модуля 4: WDTE=1 функция разрешена, WDTE=0 функция запрещена
---	CMOD.5	Зарезервирован
---	CMOD.4	Зарезервирован
---	CMOD.3	Зарезервирован
SPS1	CMOD.2	Бит выбора источника синхросигнала
SPS0	CMOD.1	Бит выбора источника синхросигнала
ECF	CMOD.0	Бит разрешения прерывания блока PCA: ECF=1. Разрешает прерывание по флагу CF ECF=0, прерывание запрещено

Регистр CCON

CF	CCON.7	Флаг переполнения таймера-счетчика PCA. Вызывает прерывание, если установлен флаг ECF регистра CMOD
CR	CCON.6	Бит включения таймера-счетчика PCA: CR=1 счетчик включен, CR=0 счетчик выключен
---	CCON.5	Зарезервирован
CCF4	CCON.4	Флаг прерывания модуля 4
CCF3	CCON.3	Флаг прерывания модуля 3
CCF2	CCON.2	Флаг прерывания модуля 2
CCF1	CCON.1	Флаг прерывания модуля 1
CCF0	CCON.0	Флаг прерывания модуля 0

Более подробно о прерываниях смотри ниже в разделе 6.6.

Регистр CCAPMx

---	ECOMx	CAPPx	CAPNx	MATx	TOGx	PWMx	ECCFx

Имя бита	Номер бита	Функция
---	CCAMPx.7	Зарезервирован
EOMx	CCAMPx.6	ECOMx=1 разрешает выполнение функции сравнения

CAPPx	CCAMPx.5	CAPPx=1 разрешает сравнение по положительному фронту
CAPNx	CCAMPx.4	CAPNx=1 разрешает сравнение по отрицательному фронту
MATx	CCAMPx.3	При MATx=1 равенство значений таймера-счетчика и регистра модуля устанавливает соответствующий флаг CCFx
TOGx	CCAMPx.2	При TOGx=1 равенство значений таймера-счетчика и регистра модуля переключает уровень сигнала на соответствующем выходе CEXx
PWMx	CCAMPx.1	Бит включения режима широтно-импульсной модуляции. При PWMx=1 модуль работает в режиме ШИМ, модулированный сигнал выдается на выход CEXx
ECCFx	CCAMPx.0	Бит разрешения прерываний по флагу CCFx регистра CCON. При ECCFx=1 прерывания разрешены.

Линии Порта 1 (P1) обеспечивают ввод-вывод для блока PCA в качестве альтернативных функций:

Имя бита	Номер бита	Функция
CEX4	P1.7	Модуль 4. Вход при фиксации, выход при сравнении и ШИМ
CEX3	P1.6	Модуль 3. Вход при фиксации, выход при сравнении и ШИМ
CEX2	P1.5	Модуль 2. Вход при фиксации, выход при сравнении и ШИМ
CEX1	P1.4	Модуль 1. Вход при фиксации, выход при сравнении и ШИМ
CEX0	P1.3	Модуль 0. Вход при фиксации, выход при срав-

нении и ШИМ

ECI	P1.2	Вход таймера-счетчика

Таймер-счетчик PCA и пять модулей сравнения имеют единственный вектор прерывания. Если бит ECF разрешения прерывания от PCA установлен, то установка флага CF приводит к генерации запроса прерывания от блока PCA.

6.2. Таймер-счетчик

Пара регистров CH-CL работает как 16-разрядный таймер-счетчик. Выбранный вход наращивает регистр младшего байта CL. Через два периода синхросигнала после переполнения CL наращивается регистр CH старшего байта. При переполнении CH устанавливается флаг CF регистра CCON и, если установлен бит ECF регистра CMOD, формируется запрос прерывания

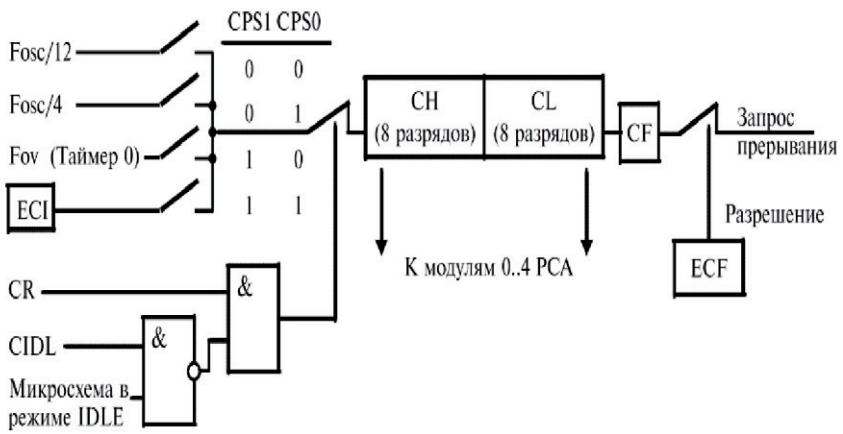


Рисунок 6.2 — Таймер-счетчик блока PCA

Биты CPS1 и CPSO регистра CMOD выбирают один из следующих четырех сигналов в качестве входного для таймера-счетчика:

- Fosc/12. Сигнал активен в такте S5P2 каждого машинного цикла. При Fosc=16 МГц таймер-счетчик наращивается (увеличивает содержимое) каждые 750 нс;
- Fosc/4. Сигнал активен в тактах S1P2, S3P2 и S5P2 каждого машинного цикла;
- переполнение Таймера 0. Регистр CL наращивается в такте S5P2 каждого машинного цикла, если Таймер 0 переполнен. Это режим работы PCA с программируемой частотой на входе;
- внешний сигнал на линии P1.2/ECI. Центральный процессор проверяет вход ECI в тактах S1P2, S3P2 и S5P2 каждого цикла обмена. Если на линии ECI имел место переход «1» — «0», содержимое регистра CL наращивается. Максимальная частота переключений на входе ECI может быть Fosc/8.

Установка бита управления работой (CR в регистре CCON) включает таймер-счетчик PCA. Таймер-счетчик PCA продолжает работать в пассивном режиме, если не установлен бит CIDL регистра CMOD. Центральный процессор всегда может прочесть содержимое регистров CH и CL. Запись в эти регистры во время счета (когда установлен бит CR) запрещена.

6.3.Модули фиксации-сравнения

Каждый модуль включает пару регистров CCAPxH/CCAPxL, 16-разрядный компаратор, логические вентили и селекторы сигналов. В регистрах запоминается время или значение счетчика, при котором внешнее событие фиксации (capture) произошло или должно произойти действие сравнения (comparison). В режиме ШИМ регистр младшего байта управляет шириной выходного сигнала.

Конфигурация каждого модуля зависит от выбранного режима его работы. Каждый модуль может быть независимо запрограммирован на работу в одном из следующих режимов:

- фиксация 16-разрядного значения по положительному фронту, отрицательному фронту или по обоим фронтам сигнала на входе CEXx;
- режимы сравнения: 16-разрядный программируемый таймер, 16-разрядный скоростной вывод, сторожевой таймер (только модуль 4), 8-разрядный ШИМ модулятор;
- нет операции.

Режим работы каждого модуля определяется комбинацией битов в регистре режима CCAPMx. Возможные комбинации битов приведены в таблице 6.1. Другие комбинации запрещены. Регистр режима CCAPMx допускает только байтовое обращение.

Для работы модулей фиксации-сравнения необходима работа таймера-счетчика PCA. Он включается-выключается в соответствии со значением бита CR регистра CCON. Для запрещения работы любого модуля его следует перевести в режим «нет операции». При наличии события (фиксация, срабатывание таймера, скоростной вывод) устанавливается флаг CCFx регистра CCON и формируется запрос прерывания от PCA (если в регистре CCAPMx установлен бит разрешения). Центральный процессор всегда может прочитать или записать информацию в регистры CCAPxH и CCAPxL.

Таблица 6.1 — Возможные комбинации битов в регистре CCAPMx

Функция модуля	---	ECOMx	CAPPx	CAPNx	MATx	TOGx	PWMx	ECCFx
Фиксация значения по положительному фронту на входе CEXx	x	x	1	0	0	0	0	X

Фиксация значения по отрицательному фронту на входе CEXx	x	x	0	1	0	0	0	X
Фиксация значения по изменению значения на входе CEXx	x	x	1	1	0	0	0	X
16-разрядный таймер	x	1	0	0	1	0	0	X
Скоростной вывод	x	1	0	0	1	1	0	X
Широтно-импульсный модулятор	x	1	0	0	0	0	1	0
Сторожевой таймер	x	1	0	0	1	X	0	X
Нет операции	x	0	0	0	0	0	0	0

6.4. Режим фиксации

Этот режим дает возможность измерять длительность импульсов, циклов, разность фаз по каждому из пяти физических входов. Входы CEX0-CEX4 проверяются на положительный или отрицательный перепад сигнала. Когда модуль фиксирует его наличие, он запоминает значение таймера-счетчика на этот момент в регистрах CCAPxH/CCAPxL и устанавливает флаг CCFx регистра CCON. Время на обработку этого события составляет один период синхронизации таймера-счетчика.

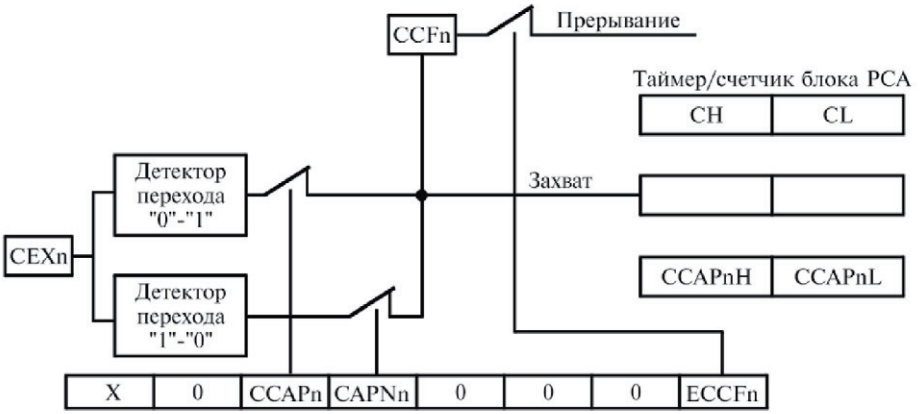


Рисунок 6.3 — Блок PCA в режиме фиксации

Если установлен соответствующий бит разрешения ECCFx регистра CCAPMx, блок PCA посылает соответствующий запрос прерывания.

Поскольку аппаратно при прерывании флаг события не очищается, пользователь должен это сделать программно. При следующем событии на этом же модуле значение таймера-счетчика в регистрах перезаписывается. Для сохранения зафиксированного значения его следует сохранить в ОЗУ в процессе обработки прерывания, пока не произошло следующее событие.

6.5. Режимы сравнения

Функция сравнения обеспечивает четыре режима: режим 16-разрядного таймера, режим ускоренного вывода, режим сторожевого таймера, режим ШИМ. В первых трех режимах модуль постоянно сравнивает содержимое таймера-счетчика со значением, загруженным предварительно в пару его регистров CCAPxH/CCAPxL. В режиме ШИМ модуль постоянно сравнивает содержимое младшего регистра таймера-счетчика CL со значением в его регистре CCAPxL. Сравнение производится три раза за цикл обмена, т.е. с

наибольшей возможной частотой (Fosc/4).

Функция сравнения для конкретного модуля выбирается установкой бита ECOMx в регистре CCAPMx. Для использования модулей в режимах сравнения следует выполнить следующие шаги:

- выбрать режим работы модуля;
- выбрать входной сигнал для таймера-счетчика;
- загрузить значение эталона в пару регистров модуля;
- установить бит управления запуском таймера-счетчика;
- после прерывания очистить флаг события.

Режим программируемого 16-разрядного таймера.

В этом режиме сравнивается текущее значение таймера-счетчика и предварительно загруженная в пару регистров CCAPxH/CCAPxL величина. При совпадении устанавливается флаг события CCFx регистра CCON. Пользователь должен программно сбросить этот флаг при обработке прерывания. При обслуживании прерывания можно загрузить в регистры CCAPxH/CCAPxL новое значение. В процессе загрузки регистров рекомендуется вначале записывать данные в CCAPxL, а затем в CCAPxH. При записи в регистр младшего байта очищается бит ECOMx, что запрещает выполнять сравнение. При записи в регистр старшего байта этот бит устанавливается, вновь разрешая сравнения. Эта последовательность защищает от ложных срабатываний.

Режим ускоренного вывода (HSO) сигнала CEXx.

В этом режиме совпадение значений таймера-счетчика и величины, загруженной в регистры CCAPxH/CCAPxL вызывает аппаратную смену сигнала на выходе CEXx и установку флага CCFx. Это обеспечивает более быстрое изменение сигнала на выходе CEXx и более высокую точность

фиксации момента совпадения, чем при программном переключении сигнала, т.к. переключение осуществляется до обслуживания запроса прерывания. Таким образом, интер- вал времени, связанный с обслуживанием прерывания, не вклинивается в диаграмму формирования выходного сигнала. Программно задавая уровень сигнала на выходе CEXx, пользователь задает тип перепада: «0»-«1», или «1»-«0». Флаг CCFx при обработке прерывания должен сбрасываться программно.

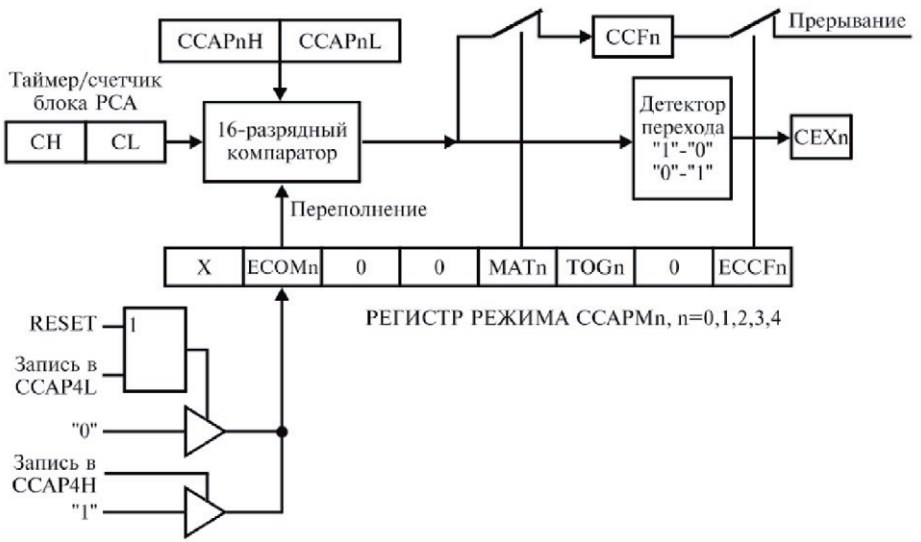


Рисунок 6.4 — Блок PCA в режиме 16-разрядного таймера и ускоренного вывода

Если в процедуре прерывания в регистры CCAPxH/CCAPxL новое значение не заносилось, то следующее совпадение произойдет через полный цикл таймера-счетчика. В процессе загрузки следует придерживаться методики, изложенной при описании предыдущего режима.

Режим сторожевого таймера.

Модуль 4 блока PCA может быть запрограммирован на выполнение

функции 16-разрядного сторожевого таймера (watchdog timer — WDT). В этом режиме при совпадении числа в таймер-счетчике с величиной, занесенной предварительно в регистры данных модуля, осуществляется сброс и инициализация микроконтроллера. Сброс микроконтроллера по истечении установленного времени является стандартным приемом выхода из зависаний программы управления.

Чтобы перевести модуль 4 в режим сторожевого таймера, нужно установить биты ECOM4 и MAT4 регистра CCAPM4, а также бит WDTE регистра CMOD в «1».

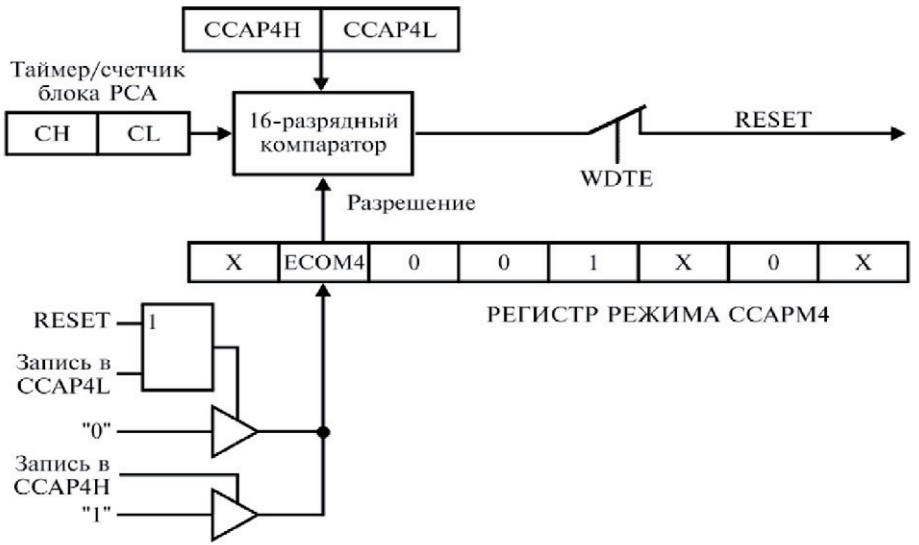


Рисунок 6.5 — Блок PCA в режиме сторожевого таймера

Необходимо также выбрать нужный вход для таймер-счетчика с помощью битов CPS0,CPS1 регистра CMOD. Далее следует загрузить 16-разрядное число для сравнения в регистры CCAP4H/CCAP4L и 16-разрядное начальное значение в таймер-счетчик CH/CL (можно использовать значение 0000H по сбросу). Разность между этими числами, умноженная на частоту входных импульсов PCA, определяет интервал времени, на который взведен

сторожевой таймер.

Чтобы предотвратить сброс от сторожевого таймера PCA, имеется три возможности:

- периодически менять сравниваемое число в CCAP4H/CCAP4L так, чтобы совпадения не произошло;
- периодически менять значения в таймер-счетчике так, чтобы совпадения не произошло;
- очищать бит WDTE до совпадения, а затем вновь устанавливать его.

Второй вариант не рекомендуется применять, когда работают другие модули, поскольку все они пользуются временным отсчетом.

Режим широтно-импульсной модуляции.

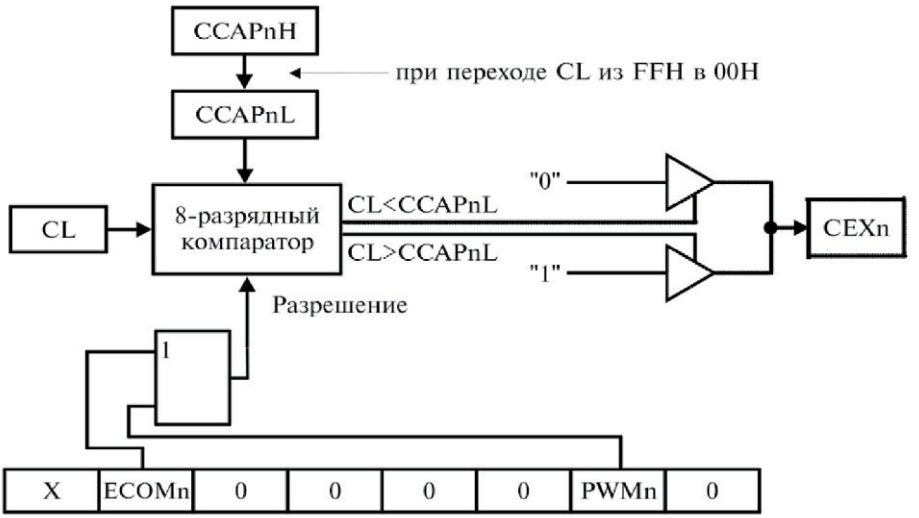
Все пять модулей блока PCA могут быть запрограммированы на режим ШИМ (рис. 6.6). При этом на выходах CEXx выдаются модулированные сигналы, ширина импульсов которых определяется 8-разрядным разрешением. Это позволяет преобразовать цифровой код в аналоговый сигнал при помощи простой внешней схемы (например, интегрирующей цепочки).

В этом режиме младший байт таймера-счетчика (CL) постоянно сравнивается с содержимым регистра CCAPxL. Если CL<CCAPxL, то на выходе CEXx уровень сигнала низкий. При совпадении сигнал на выходе приобретает значе- ние «1>> и остается таким, пока счетчик не достигает окончания счета (00H). После этого выходной сигнал возвращается на низкий уровень, в регистр

CCAPxL загружается значение из регистра CCAPxH и начинается новый цикл счета.

Число в CCAPxL определяет ширину импульса в текущем цикле, а число в CCAPxH определяет ширину импульса в следующем цикле. При CCAPxL=0 ширина импульса составляет 100%, а при CCAPxL=225 она равна 0,4%.

Частота сигнала на выходе ШИМ равна частоте сигнала на входе таймера-счетчика, деленной на 256. Самая высокая частота имеет место при входе Fosc/4. Если Fosc=16 МГц, то частота сигнала на выходе равна 15,6 кГц.



Регистр режима CCAPMn, где n=0…4

Рисунок 6.6 — Блок PCA в режиме широтно-импульсного модулятора

Для перевода модуля в режим ШИМ нужно установить биты ECOMx и PWMx регистра CCAPMx. Далее следует определить вход посредством комбинации битов CPS0, CPS1 регистра CMOD. Затем необходимо занести 8-разрядные числа в регистры CCAPxL и CCAPxH. Наконец, следует установить бит CR управления запуском таймера-счетчика в регистре CCON.

6.6.Таймер 2

Таймер 2 представляет собой 16-разрядный счетчик, который может работать в следующих трех режимах: режим фиксации текущего счетного значения, режим счета в прямом и обратном направлении с автоматической

перезагрузкой исходного значения, режим задающего генератора для последовательного порта.

Шестнадцатиразрядный регистр данных Таймера 2 состоит из регистров TH2 (старший байт) и TL2 (младший байт). Данные для автоперезагрузки хранятся в регистрах RCAP2H (старший байт) и RCAP2L (младший байт), а регистрами управления являются регистры T2CON и T2MOD.

6.7.Последовательный порт

Последовательный порт имеет дополнительные функции в сравнении с базовым микроконтроллером 8051. Это функция автоматического распознавания адреса и функция определения ошибки адреса.

Определение ошибки адреса

Эта функция реализуется через проверку уровня сигнала во время интервала времени, когда на линии должен находится стоп — бит. Проверка выполняется в режимах 1, 2 и 3. Если стоп — бит в определенное время не обнаружен, то устанавливается флаг ошибки кадра FE. Этот флаг делит с битом управления SM0 один и тот же разряд SCON.7. Для идентификации бита в разряде SCON.7 используется шестой разряд регистра PCON — SMOD0 (в базовом микроконтроллере этот бит резервный). Если бит SMOD0=0, то разряд SCON.7 идентифицируется как бит управления SM0; если бит SMOD0=1, то разряд SCON.7 идентифицируется как флаг ошибки кадра FE. Флаг ошибки кадра FE может проверяться после каждой операции приема. Сбрасываться этот флаг должен программно, прием правильного кадра не сбрасывает ранее установленного флага FE.

Автоматическое распознавание адреса

При работе нескольких микроконтроллеров по одним и тем же линиям последовательного канала ввода — вывода Ведущий может обратиться к Ведомому по индивидуальному адресу, либо к группе Ведомых по широковещательному адресу. Для хранения индивидуального адреса предназначен специальный регистр SADDR, а для хранения кода маски предназначен специальный регистр SADEN. Маска дает возможность адресовать в каждый момент времени одно или несколько устройств. Рассмотрим это на примере.

Ведомый 1	Ведомый 2
SADDR 11110001 SADEN 11111010 Адрес — отзыва11110X0X	SADDR 11110011 SADEN 11111001 Адрес отзыва — 11110XX1

В приведенном примере в адресе Ведомого 1 замаскированы биты 0 и 2, а в адресе Ведомого 2 замаскированы биты 1 и 2. Для одновременного обращения к обоим ведомым контроллерам широковещательными адресами являются следующие: 11110001 и 11110101.

6.8. Система прерываний

У микроконтроллеров 8XC51FX имеется семь источников прерываний (добавлено прерывание от блока PCA) и четырехуровневая система приоритетов.

Формат регистров IE, IP, IPH и назначение отдельных битов следующее:

Регистр IE

EA	EC	ET2	ES	ET1	EX1	ET0	EX0

Имя бита	Номер бита	Функция
EA	IE.7	Запрещение запросов от всех источников, имеет место при EA=0
EC	IE.6	Запрещение запроса от блока PCA, имеет место при EC=0

ET2	IE.5	Запрещение запроса от Таймера 2, имеет место при ET2=0
ES	IE.4	Запрещение запроса от последовательного порта, имеет место при ES=0
ET1	IE.3	Запрещение запроса от Таймера 1, имеет место при ET1=0
EX1	IE.2	Запрещение запроса по входу , имеет место при EX0=0
ET0	IE.1	Запрещение запроса от Таймера 0, имеет место при ET0=0
EX0	IE.0	Запрещение запроса по входу , имеет место при EX0=0

Регистр приоритетов прерываний **IP младший**

---	PPC	PT2	PS	PT1	PX1	PT0	PX0

Имя бита	Номер бита	Функция
---	IP.7	Зарезервирован
PC	IP.6	Определяет приоритет Таймера 2, младший бит
PT2	IP.5	Определяет приоретит Таймера 2, младший бит
PS	IP.4	Определяет приоретит последовательного порта, младший бит
PT1	IP.3	Определяет приоритет Таймера 1, младший бит
PX1	IP.2	Определяет приоритет входа INT1#, младший бит
PT0	IP.1	Определяет приоритет Таймера 0, младший бит
PX0	IP.0	Определяет приоритет входа INT0#, младший бит

Регистр приоритетов прерываний **IPH старший**:

---	PPCH	PT2H	PSH	PT1H	PX1H	PT0H	PX0H

Имя бита	Номер бита	Функция
---	IPH.7	Зарезервирован
PCH	IPH.6	Определяет приоритет Таймера 2, старший бит
PT2H	IPH.5	Определяет приоретит Таймера 2, старший бит
PSH	IPH.4	Определяет приоретит последовательного порта, старший бит
PT1H	IPH.3	Определяет приоритет Таймера 1, старший бит
PX1H	IPH.2	Определяет приоритет входа INT0#, старший бит
PT0H	IPH.1	Определяет приоритет Таймера 0, старший бит
PX0H	IPH.0	Определяет приоритет входа INT1#, старший бит

Вектор прерывания от блока PCA равен 00ЗЗН. Уровень приоритета каждого источника (вектора) прерываний устанавливается путем записи комбинации «0» и «1» в соответствующие разряды регистров IP, IPH. Так, комбинация 00 соответствует 0-му (низшему) уровню приоритета, а комбинация 11 соответствует 4-му (высшему) уровню приоритета.

При поллинге (последовательном опросе) источники прерываний опрашиваются в следующем порядке.

Приоритеты прерываний 8XC51FX при поллинге

Источник	Приоритет внутри уровня
Вход INT0#	Высший
Таймер 0	
Вход INT1#	
Таймер 1	
Блок PCA	
Последовательный порт	
Таймер 2, флаг TF2	
Таймер 2, флаг T2EX	Низший

7.ВЫСОКОИНТЕГРИРОВАННЫЙ МИКРОКОНТРОЛЛЕР ДЛЯ СБОРА ИНФОРМАЦИИ И УПРАВЛЕНИЯ ADUC812

7.1. Общие сведения о микроконтроллере ADuC 812

Аналоговый ввод-вывод

- 8-канальный прецизионный 12-бит АЦП.
- Встроенный термостабильный источник опорного напряжения (ИОН).
- Высокая скорость выборок 200К/сек.
- Два 12-битных ЦАПа (преобразователь код — напряжение). Внутренний температурный сенсор.

Память

- 8Кбайт FLASH памяти программ.
- 640 байт FLASH памяти данных.
- Внутренний источник программирования типа «зарядовый насос» (внешний источник не требуется).
- 256 байт внутренней памяти данных
- 16Мбайт адресного пространства внешней памяти данных

8051 — совместимое ядро

- 12МГц номинальная частота (16МГц — максимальная)
- Три 16-битных счетчика-таймера
- 32 программируемых линии ввода-вывода
- Порт с повышенной нагрузочной способностью — Порт 3
- 9 источников прерываний, 2 уровня приоритета

Питание

- Допускает напряжение питание 3В или 5В.
- Режимы: нормальный, холостой и дежурный.

Встроенная периферия

- Последовательный UART
- 2-проводной (I2C) и SPI
- Сторожевой таймер (WDT)
- Монитор источника питания

Основные области применения

- Интеллектуальные сенсоры (в соответствии с IEEE 1451.2)
- Батарейные системы (портативные PC, инструмент, мониторы)
- Системы слежения
- Системы сбора информации, коммуникационные системы

ADuC812 — интегральная 12-битная система сбора информации, включающая в себя прецизионный многоканальный АЦП с самокалибровкой, два 12-битных ЦАПа и программируемое 8-битное микропроцессорное ядро (совместимое с микроконтроллером 8051) (MCU). MCU поддерживается внутренними 8Кбайт FLASH ЭРПЗУ программ, 640 байт ЭРПЗУ памяти данных и 256 байт статической памяти данных с произвольной выборкой (RAM).

MCU поддерживает следующие дополнительные функции: Сторожевой Таймер, Монитор Питания и канал прямого доступа для АЦП. Для мультипроцессорного обмена и расширения ввода-вывода (в-в) имеются 32 программируемых линии в-в и последовательные интерфейсы I2C, SPI и стандартный UART.

Для гибкого управления в применениях с низким потреблением в MCU и аналоговой части предусмотрены 3 режима работы: нормальный, холостой и дежурный. Система ADuC812 допускает работу с напряжением питания 3В и 5В в индустриальном диапазоне температур (- 45 С … + 85 С) и конструктивно выполнена в 52-выводном пластмассовом корпусе (тип PQF).

Функциональная блок-схема микроконтроллера представлена на рис.7.1. Технические характеристики ADuC812 приведены в таблице 7.1.

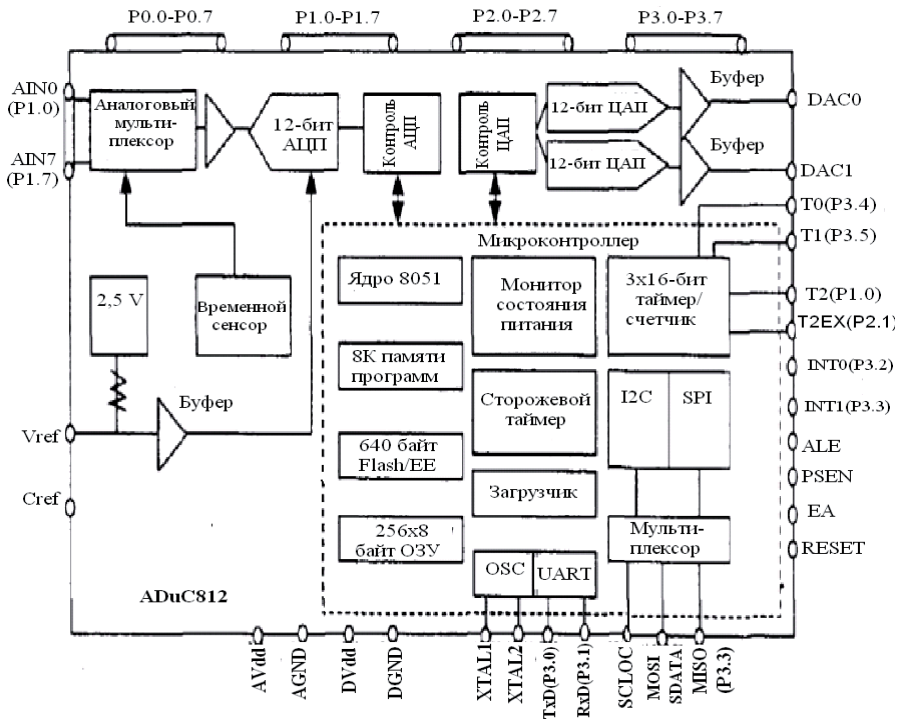


Рисунок 7.1 — Функциональная блок-схема микроконтроллера ADuC812

Все характеристики в таблице 7.1 приводятся для температурного диапазона от Тмин до Тмакс, если не указано особо. Частота кварцевого резонатора (MCLCIN) составляет 16 МГц. Выходное напряжение \/out ЦАП (DAC)

относительно нулевого потенциала (AGND) питания аналоговой части снимается при нагрузке R1=10 КОм и C1=100 пФ. Питание аналоговой части микроконтроллера AVdd составляет +3В или +5В +/-10%, величина опорного встроенного напряжения ИОН Vref=2,5 В. Сокращение LSB (Least Significant Bit) означает младший бит.

Таблица 7.1 — Технические характеристики ADuC812

Параметр	ADuC812 V_{dd} =		Единицы	Условия/ примечания
	5В	3В		
1	2	3	4	5
АЦП−спецификация каналов				
Точность по постоянному току[3,4]				
Разрешение	12	12	Биты	Fsampl=100КГц
Интегральная нелинейность	±1/2	±1/2	LSB средняя	Fsampl=100КГц
	±1/5		LSB максим.	Fsampl=200КГц
	±1/5	±1/5	LSB миним.	Fsampl=100КГц
Дифференциальная нелинейность	±1	±1	LSB средняя	Отсутствие пропуска кодов при 5В гарантируется
КАЛИБРОВОЧНЫЕ ОШИБКИ КОНЕЧНЫХ ТОЧЕК ШКАЛЫ[5,6]				
Ошибка смещения	±5		LSB максим.	
	±1	±2	LSB средняя	
Согласованность ошибки смещения (по каналам)	1	1	LSB средняя	
Ошибка усиления	±6	±2	LSB максим.	
	±1		LSB средняя	
Согласованность ошибки усиления	1,5	1,5	LSB средняя	

Продолжение таблицы 7.1

Параметр	ADuC812 V_{dd} =		Единицы	Условия/ примечания
	5В	3В		
1	2	3	4	5
ПОЛЬЗОВАТЕЛЬСКАЯ СИСТЕМНАЯ КАЛИБРОВКА-				
Диапазон калибровки сдвига	±5	±5	% от V_{ref} средн.	
Диапазон калибровки	±2,5	±2,5	% от V_{ref} средн.	
ДИНАМИЧЕСКОЕ ПРЕОБРАЗОВАНИЕ				F_{in}=10 КГц , синус. сигнал Fsampl=100КГц
Отношение сигнал/шум (SNR)[6]	70	70	дБ, среднее	
Полный коэффициент гармоник (TH)	-78	-78	дБ, средний	
Пиковая гармоника или шумовая помеха	-78	-78	дБ, средняя	
АНАЛОГОВЫЙ ВХОД				
Диапазон входных напряжений	0-Vref	0-Vref	Вольты	
Входной ток	± 10		мкА, макс.	
	± 1	± 1	мкА, средн.	
Входная емкость	20	20	пФ, макс.	
ТЕМПЕРАТУРНЫЙ СЕНСОР[9]				Измеряется встроенным АЦП с точностью ± 0,513 В
Выходное напряжение (25 ^{0}C)	600	600	мВ, среднее	
Температурный коэффициент	- 3,0	- 3,0	мВ/ ^{0}C	

Продолжение таблицы 7.1

Параметр	ADuC812 Vdd=		Единицы	Условия/ примечания
	5 В	3 В		
ЦАП – СПЕЦИФИКАЦИЯ КАНАЛОВ				
Точность по постоянному току[10]				
Разрешение	12	12	Биты	Гарантируется
Относительная точность	± 3	± 3	LSB, средн.	12 – битная
Дифференциальная нелинейность	± 0,5	± 1	LSB, средн.	монотонность
Ошибка смещения	± 50		мВ, макс.	
	± 25	± 25	мВ, средн.	
Ошибка шкалы	± 25		мВ, макс.	
	± 10	± 10	мВ, средн.	% полной шкалы
Согласование шкал	± 0,5	± 0,5	%, средн.	по ЦАП
АНАЛОГОВЫЕ ВЫХОДЫ				
Диапазон напряжений 0	0-Vref	0-Vref	В, средний	
Диапазон напряжений 1	0-Vdd	0-Vdd	В, средний	
Величина резистивной нагрузки	10	10	кОм, средн.	
Величина емкостной нагрузки	100	100	пФ, средн.	
Выходной импеданс	0,5	0,5	Ом, средн.	
Isink	50	50	мкА, средн.	
ВХОДЫ/ВЫХОДЫ				
Диапазон входных напряжений	2.3/		Вольты	
	Vdd		мин/макс	
Входной импеданс	150		КОм средний	
Величина выходного напряжения	2.4/		Вольты	
	2.55	2.5	мин/макс	
	2.5	40	Вольты средн.	
Температурный коэффициент выходного напряжения	40		ppm/’C	

Продолжение таблицы 7.1

Параметр	ADuC812 V_{dd} =		Единицы	Условия/ примечания
	5В	3В		
1	2	3	4	5
РАБОЧИЕ ХАРАКТЕРИСТИКИ ЭРПЗУ (FLASH)[11,12]				
Допустимое число циклов программирования	10000		Циклов минимум	
	50000	50000	Циклов среднее	
Сохранность данных	10		Лет, минимум	
СТОРОЖЕВОЙ ТАЙМЕР (WDT)				
Частота генерации	64	64	КГц средняя	
ХАРАКТЕРИСТИКИ МОНИТОРА ПИТАНИЯ (PSM)				
Точность установки порога срабатывания	±2.5		% от номинал. значения выбранного порога	
	±1.0	±1.0	максим. % от номинального значения выбранного порога в среднем	
ЦИФРОВЫЕ ВХОДЫ				
Вх. напряжение высокого уровня Вх.	2.4		Вольты мин.	
напряжение низкого уровня	0.В		Вольты макс.	
Входной ток утечки (Порт 0, EA)	±10		мкА макс.	Vin=0В или Vdd
	±1	±1	мкА средний	Vin =0В или Vdd
Входной ток Лог.1 (все цифровые	±10		мкА макс.	Vin =Vdd
входы)	±1	±1	мкА средний	Vin =Vdd
Входной ток Лог.О (Порт 1,2, 3)	-80		мкА макс.	
	-40	-40	мкА средний	Vin =450мВ
Ток при переходе Лог. 1-0	-700		мкА макс.	Vin =2В
(Порт 1,2,3)	-400	-400	мкА средний	Vin =2В
Входная емкость	10	10	пф средняя	

Продолжение таблицы 7.1

Параметр	ADuC812 Vdd=		Единицы	Условия/ примечания
	5В	3В		
ЦИФРОВЫЕ ВЫХОДЫ				
Выходное напряжение высокого уровня Voh)	2.4	2.6	Вольт мин.	Vdd=4.5В —5.5В, Isrc=80мкА
	4.0		Вольт среднее	Vdd=2.7В —3.3В, Isrc=20мкА
Выходное напряжение низкого уровня (Vol) ALE, PSEN, Порт 0, 2,	0.4	0.2	Вольты макс.	Isink =1.6мА
Порт 3	0.2	0.2	Вольты средн.	Isink =1.6мА
	0.4		Вольты макс.	
	0.2		Вольты средн.	Isink =8мА
Ток утечки в «плавающем состоянии»	±10	±5	мкАмакс.	Isink =8мА
Выходная емкость в «плавающем состоянии»	10	10	пф средняя	
ИСТОЧНИК ПИТАНИЯ[13,14,15]				
Нормальный режим [16]	42	16	мА макс	MCLKIN=16МГц
	32		мА средний	MCLKIN=16МГц
	26	12	мА средний	MCLKIN=12МГц
	8	3	мА средний	MCLKIN=1МГц
Холостой режим	25		мА макс.	MCLKIN=16МГц
	18	17	мА средний	MCLKIN=16МГц
	15	6	мА средний	MCLKIN=12МГц
Дежурный режим[17]	7	2	мА средний	MCLKIN=1МГц
	50	50	мА макс.	
	5	5	мА средний	

Примечания:

1. Характеристики используются после проведения калибровки.
2. Температурный диапазон от -40 до +85’С.
3. Линейность гарантирована при нормальной работе MCU ядра.
4. Линейность может ухудшаться при программировании или стирании 640Б ЭРПЗУ во время выполнения А-Ц преобразования из-за работы схемы зарядного насоса.
5. Измерено при производстве при Vdd=5В после выполнения процедуры калибровки и только при +25’С. ‘
6. Пользователю возможно потребуется выполнить процедуру калибровки

для получения этих характеристик, которые зависят от конфигурации.

7. Диапазон коррекции при калибровке смещения и усиления определяется как диапазон напряжений, который ADuC812 может скомпенсировать при выполнении системной калибровки.
8. Вычисление коэффициента шума (SNR) учитывает шумовую компоненту и искажения.
9. Температурный сенсор измеряет непосредственно температуру кристалла, из этих результатов можно вычислить температуру окружающей среды.
10. Линейность ЦАП вычисляется с учетом: сокращенного диапазона кодов от 48 до 4095, для диапазона от 0 до Vref; сокращенного диапазона кодов от 48 до 3995, для диапазона от 0 до Vdd; нагрузка ЦАПа составляет 10КОм и 50пф.
11. Рабочие спецификации FLASH ЭРПЗУ такие же, как и в JEDEC спецификации A103 (Сохранность данных) и в JEDEC предварительной спецификации A117 (Допустимое число циклов программирования).
12. Допустимое число циклов программирования оценивается в следующих условиях:

Режим	Байтовое программирование
Циклическое стирание страницы	
Циклические данные	00(H) до FF(H)
Время стирания	20мс
Время программирования	100мкс

13. Токопотребление (Idd) при других значениях тактовой частоты MCLKIN определяется выражениями:

Нормальный режим (Vdd=5В)	Idd=(1.6*МСЦ<1М)+6
Нормальный режим (Vdd=3В)	Idd=(0.8*МСЦ<1М)+3
Режим х.х.(Vdd=5В)	Idd=(0.75*МС1-К1М)+6
Режим х.х.(Vdd=3В)	Idd=(0.25*МС1-К1М)+3

Здесь MCLKIN выражается в МГц, а результат Idd — в мА.

14. Idd ток выражается суммой аналогового и цифрового питания при работе микроконтроллера AduC812 в нормальном режиме.
15. Idd не измеряется в циклах стирания или программирования ЭРПЗУ; для этих циклов Idd обычно увеличивается на 10 мА.
16. Аналоговая часть Idd=2мА (в среднем) при нормальной работе (внутренний ИОН, АЦП и ЦАП включены).
17. Потенциалы выводов EA и Порт0 равны DVdd, вход XTAL1 во время этих измерений также имеет потенциал DVdd.

Средние (Typical) спецификации не проверяются, но подтверждаются данными при выпуске изделий. Дополнительную информацию можно получить в Справочнике Пользователя, Кратком Справочнике, Справочнике по Применению и Листе Ошибок по http://www.analog.com

Предельно допустимые параметры*

*(Ta = +25 °C, если не оговаривается особо)

AVDD к DVdd	± 0.3 В
AGND к DGND	±0.3 В
DVdd к DGND, Avdd к AGND	-0.3 В ..+7 В
Цифровой вход к DGND	-0.3В, DVdd+0.3 В
Цифровой выход к DGND	-0.3 В, DVdd+ 0.3 В
Vref к AGND	-0.3 В, Avdd + 0.3 В
Аналоговые входы к AGND	-0.3 В, Avdd + 0.3 В
Индустриальный диапазон рабочих температур	-40˚C… +85˚C
Температура хранения	-65˚C.. +150˚C
Температура перехода	+150˚C
Температурное сопротивление	+90˚C/Вт
Температура при пайке: в паровой фазе (60 сек)	+215˚C
инфракрасная (15 сек)	+220˚C

Превышение указанных выше предельных параметров может вызвать повреждение микроконтроллера. Эксплуатация устройства при предельных значениях параметров может повлиять на его надежность.

Микроконтроллер чувствителен к электростатическим разрядам (ESD). Несмотря на то, что устройство имеет цепи защиты, для сохранения его работоспособности следует предпринять соответствующие меры.

Расположение контактов ADuC812 приведено в табл. 7.2.

В табл. 7.3 приведено функциональное назначение контактов микроконтроллера.

Таблица 7.2. - Расположение контактов ADuC812

№ конт.	Наименование контакта	№ конт	Наименование контакта
1	P1.0/ADC0/T2	27	SDATA/MOSI
2	P1.1/ADC1/T2EX	28	P2.0/A8/A16
3	P1.2/ADC2	29	P2.1/A9/A17
4	P1.3/ADC3	30	P2.2/A10/A18
5	Avdd	31	P2.3/A11/A19
6	AGND	32	XTAL1 (in)
7	Cref	33	XTAL2 (out)
8	Vref	34	DVdd
9	DAC0	35	DGND
10	DAC1	36	P2.4/A12/A20
11	P1.4/ADC4	37	P2.5/A13/A21
12	P1.5/ADC5/SS/	38	P2.6/A14/A22
13	P1.6/ADC6	39	P2.7/A15/A23
14	P1.7/ADC7	40	EA//Vpp
15	RESET	41	PSEN/
16	P3.0/RxD	42	ALE
17	P3.1/TxD	43	P0.0/AD0
18	P3.2/INT0/	44	P0.1/AD1
19	P3.3/INT1//MIS0	45	P0.2/AD2
20	DVdd	46	P0.3/AD3
21	DGND	47	DGND
22	P3.4/T0	48	DVdd
23	P3.5/T1/CONVST/	49	P0.4/AD4
24	P3.6/WR/	50	P0.5/AD5
25	P3.7/RD/	51	P0.6/AD6
26	SCLOCK	52	P0.7/AD7

Таблица 7.3 - Функциональное назначение контактов микроконтроллера

Мнемоника	Тип	Функция
DVdd	P	Положительное номинальное цифровое питание +3В или +5В
Avdd	P	Положительное номинальное аналоговое питание +3 или +5В
Cref	I	Блокирующий конденсатор для внутреннего ИОН 0,1 мкФ на
Vref	I/O	AGND ИОН вход/выход. Этот контакт внутри соединён через последовательный резистор с ИОН для АЦП и ЦАП. Номинальное напряжение ИОН 2.5В и появляется на контакте (как только АЦП и ЦАП разрешены). Внутренний ион подавляется подключением к этому контакту внешнего источника
AGND	G	Аналоговая земля. Общая точка аналоговых цепей
P1.0-P1.7	I	Порт 1 только на ввод. Порт 1 по умолчанию настраивается на ввод аналоговых сигналов, для конфигурирования контактов на цифровой ввод следует записать 0 соответствующий бит порта. Порт 1 – многофункционален и перечисленные функции выполняет
ADC0-ADC7	I	Аналоговые входы. Восемь однофазных входов. Выбор канала осуществляется через регистр специального назначения (SFR)ADCCON2
T2	I	Цифровой вход Таймер/счётчика 2, при работе Счётчик 2 нкрементируется по перепаду 1-0 на входе T2
T2EX	I	Цифровой вход. для триггера захвата/перезагрузки Счётчика2,также работает как вход управления направлением счёта Счётчика2
SS/	I	Выбор ведомого (Slave Select). Для синхронного интерфейса (SPI)
SDATA	I/O I	Выбираемый пользователем ввод/вывод для I2C и SPI
SCLOCK	I/O	Синхронизация для I2C и SPI
MOSI	I/O	Для I2C Ведущий Выход/Ведомый Вход
MISO	I/O	Для SPI Ведущий вход /Ведомый Выход
DAC0	O	Выходное напряжение с ЦАП0
DAC1	O	Выходное напряжение с ЦАП1
RESET	I	Цифровой вход. Высокий уровень сигнала на этом контакте в течении 24 периодов тактовой частоты при работающем генераторе вызывает выполнение устройством сброса

Продолжение таблицы 7.3

Мнемоника	Тип	Функция
P3.0-P3.7	I/0	Двунаправленный Порт3 с внутренними, подтягивающими к питанию резисторами. Контакты Порта3 с записанными в них 1 подтянуты вверх и могут использоваться также как входы. При использовании контактов в качестве входов, следует иметь в виду, что они дают ток во внешнюю цепь. Контакты Порта3 — мультиплексны
RxD	I/0	Вход приёмника асинхронного последовательного интерфейса (UART) или Ввод/Вывод данных для синхронного обмена
TxD	0	Выход передатчика асинхронного последовательного интерфейса (UART) или выход синхронизации для синхронного обмена
INT0/	I	Вход внешнего прерывания 0, программируется по перепаду/ уровню; устанавливается один из 2-х уровней приоритета. Контакт может использоваться как строб управления для Таймера0.
INT1/	I	Вход внешнего прерывания 1, программируется по перепаду/ уровню; устанавливается один из 2-х уровней приоритета. Контакт может использоваться как строб управления для Таймера1
T0	I	Вход Таймер/Счётчика0
T1	I	Вход таймер/Счётчика1.
CONVST/	I	Вход запуска преобразования АЦП (активный низкий уровень) при разрешённом внешнем запуске. Переход 0-1 переводит схему в режим хранения и запускает цикл преобразования.
WR/	0	Выход сигнала управления Записью. Защёлкивает байт данных из Порта0 во внешнюю память
	0	данных
RD/		Выход сигнала управления Чтением. Разрешает
	0	ввод данных из внешней памяти в Порт0
XTAL2	I	Инвертирующий выход генератора усилителя.
XTAL1		Вход усилителя и вход доступа к внутренним цепям генератора
DGND	G	Цифровая земля. Общая точка цифровых цепей.

Продолжение таблицы 7.3

Мнемоника	Тип	Функция
P2.0-2.7 (A8-A15) (A16-A23)	I/0	Двунаправленный Порт2 с внутренними, подтягивающими к питанию резисторами. Контакты Порта2, с записанными в них 1 подтянуты к DVdd и могут использоваться также как входы. При использовании контактов в качестве входов следует иметь ввиду, что они дают ток во внешнюю цепь. При выборке памяти программ Порт2 содержит старший байт адреса, при обращении к памяти данных порт выдаёт средний и старший байты 24-х разрядного адресного пространства.
PSEN/	0	Выход строба разрешения внешней памяти программ. Являет- ся сигналом управления внешней памяти программ. Активен в течение 6 периодов тактового генератора, исключая время до- ступа к внешней памяти данных. Контакт находится в состоянии Лог.1 при работе с внутренней памятью программ. Контакт можно использовать для разрешения режима последовательной загрузки в ЭРПЗУ, для этого контакт подключается через последовательный резистор к земле на время включения питания или генерации сигнала RESET/.
ALE	0	Выход строба записи адреса. Используется для защелкивания младшего байта адреса (при 24-битном пространстве - средне- го байта адреса) при обращении к внешней памяти. Активен дважды в одном машинном цикле, исключая обращение к внутренней памяти данных
EA/	I	Вход разрешения доступа к внешней памяти программ. Если =1, выборка производится из внутренней памяти OOOOOH ... 1FFFH, если=0, то все инструкции выбираются из внешней памяти.
P0.0-P0.7 (A0-A7)	I/0	Двунаправленный Порт0 с открытым истоком. Контакты порта с записанными в них 1 являются плавающими и могут быть высокоимпедансными входами. При обращении к внешней памяти программ или данных Порт0 мультиплексирован магистралями младшего байта адреса и данных. При такой операции порт подтянут внутренним образом при наличии в нем 1.

Параметры АЦП

Интегральная нелинейность

Параметр представляет собой максимальное отклонение любого кода от прямой линии, проведенной через крайние точки передаточной функции АЦП (статической характеристики входа-выхода). Крайними точками являются: нулевая — на 0.5LSB ниже точки появления первого кода и последняя — на 0.5LSB выше граничного кода шкалы.

Дифференциальная нелинейность

Параметр представляет собой разницу между измеренной и идеальной шириной 1 кванта (1 LSB) АЦП.

Ошибка смещения

Параметр представляет собой отклонение первичной смены кода с (000H) до (001H) от идеального значения т.е. +0.5LSB.

Ошибка полной шкалы

Параметр представляет собой отклонение момента последней смены кода от идеального входного напряжения, соответствующего разности (полная шкала — 1.5LSB), после компенсации ошибки смещения.

Отношение сигнал/шум (шум квантования)

Параметр представляет собой измеренное отношение сигнала к шуму на выходе АЦП. Сигнал — среднеквадратичный выходной сигнал с АЦП. Шум — среднеквадратичная сумма составляющих в полосе до (Fs/2 —

половина частоты выборки), исключая постоянную составляющую. Отношение зависит от величины квантования в процессе преобразования сигнала. Чем больше число квантов, тем меньше шум квантования. Для идеального АЦП с синусоидальным сигналом на входе: SNR=(6.02*N+1/76) (дБ), N — число разрядов. Таким образом, для 12 разрядного АЦП SNR=74 дБ.

Коэффициент гармоник

Параметр представляет собой отношение суммы среднеквадратичных сигналов гармоник к основной гармонике.

Параметры ЦАП

Относительная точность

Относительная точность (или нелинейность в конечной точке шкалы) Параметр есть величина максимального отклонения функции передачи ЦАП от идеальной прямой, проведенной через крайние точки. Она измеряется после компенсации ошибок сдвига нуля и полной шкалы.

Время установления выходной величины

Параметр представляет собой интервал времени, в течение которого выходное напряжение достигает заданного уровня при изменении входного кода до значения полной шкалы.

Величина импульсной помехи на аналоговом выходе со стороны цифрового входа

Параметр представляет собой некоторую величину заряда, инжектированного на аналоговый выход при изменении входного кода. Помеха опреде-

ляется площадью импульса в (нВ* сек).

7.2. Архитектура и основные характеристики AduC812.

ADuC812 представляет собой 12-битную систему сбора информации высокой степени интеграции. Ядро системы представлено высокопроизводительным 8-битным микроконтроллером совместимым с 8051 MCU со встроенным не разрушаемым FLASH ЭРПЗУ и 12-битным АЦП. Для поддержки ядра системы сбора чип содержит в себе необходимые вторичные элементы. Они включают в себя пользовательское ЭРПЗУ данных, Сторожевой таймер (WDT), Монитор питания (PSM), различные параллельные и последователь- ные интерфейсы промышленного стандарта.

Организация памяти

Подобно 8051 ADuC812 имеют раздельное пространство памяти программ и данных, как показано на рис 7.2а и 7.2б. Для пользователя доступны 640 байт Пользовательского ЭРПЗУ в области данных. Пользовательское ЭРПЗУ доступно косвенно через группу регистров управления в области Регистров Специального Назначения (Special Function Registers — SFR).

Пространство памяти программ (только чтение)

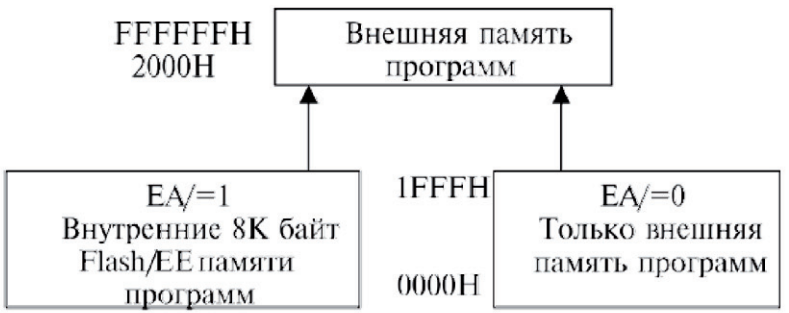


Рис. 7.2а. — Распределение памяти программ

Пространство памяти данных (чтение/запись)

Рис. 7.2б. — Распределение памяти данных.

Нижние 128 байт внутренней памяти данных распределяются так, как показано на рис 7.3

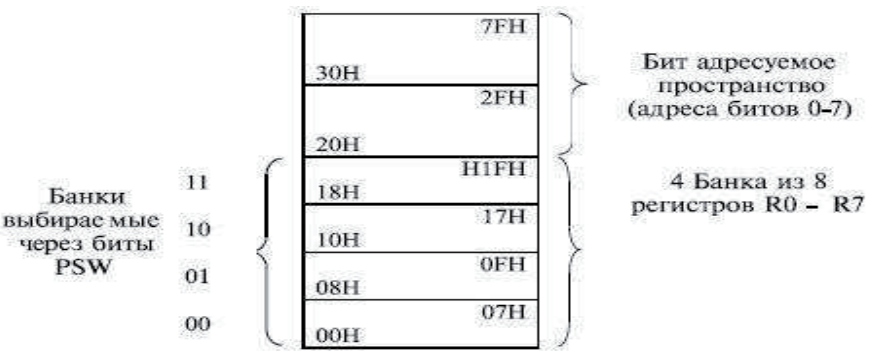


Рисунок 7.3. — Распределение 128 нижних байтов внутренней RAM

Пространство от 0 до 31 байта разделено на 4 банка по 8 регистров с R0 по R7. Следующие 128 бит (16 байт), над банками, формируют блок бит-адресуемой памяти с адресами 00H до 7FH.

Пространство Регистров Специального Назначения (SFR) расположено в верхних 128 байтах внутренней памяти. SFR адресуются только непосредствен-но, и они служат интерфейсом между MCU и всей периферией. На рис 7.4. приведена программная модель микроконтроллера.

Рис 7.4. Программная модель ADuC812

7.3. Блок АЦП

Общие сведения

Блок АЦП включает в себя 12-битный 8-ми канальный А-Ц преобразователь с однополярным питанием и временем преобразования 5 мкс на канал. Пользователю дается многоканальный мультиплексор, устройство выборки-хранения, встроенный источник опорного напряжения (ИОН), система калибровок и собственно АЦП. Все компоненты блока легко управляется через 3 интерфейсных SFRa.

А-Ц преобразователь состоит из стандартного преобразователя (конвертера) последовательного приближения и емкостного ЦАПа. Конвертер получает аналоговые входные сигналы в диапазоне 0. Vref. На кристалле расположен ИОН: прецизионный блок с низким дрейфом, откалиброванный изготовителем до 2.5В. На контакт Vref можно подавать напряжение от внешнего ИОН. В этом случае внутренний ИОН будет подавлен внешним. Внешний ИОН может быть в диапазоне напряжений от 2.3 В до Vref .

Однократный или повторяющийся режимы преобразования могут выполняться программно или подачей внешнего сигнала Запуска Преобразования на контакт 25 (CONVST/). Так же для инициирования повторяющегося процесса преобразования можно использовать сигналы Таймера 2. АЦП можно установить в режим передачи данных по каналу прямого доступа КПДП (DMA), когда блок повторяет циклы преобразования и посылает выборки во внешнюю память данных (RAM), минуя процессор. Этот процесс может охватывать весь объем внешней памяти 16Мбайт.

ADuC812 поставляется с заводскими калибровочными коэффициентами, которые загружаются автоматически по включению питания, обеспечивая тем самым оптимальную работу устройства. Ядро АЦП содержит внутренние регистры калибровок Смещения и Усиления, причем, предусмотрено, чтобы программная процедура калибровки пользователя подавляла заводские установки, давая тем самым минимум ошибок в конечной системе. Если необходимо, через АЦП можно так же преобразовать сигнал внутреннего температурного сенсора (канал-9).

Функция передачи АЦП

Диапазон входных напряжений АЦП 0..Vref. Для этого диапазона напряжений смена соответствующих кодов происходит посередине между последовательными квантами (т.е. 1/LSB, 3/2 LSBs, 5/2 LSBs, .., FS-3/2 LSBs). Выходной код — прямая в двоичном коде с 1LSB=FS/4096 или 2.5В/4096=0.61мВ при Vref=2.5В. Идеализированная функция передачи от 0 до Vref показана на рис. 7.5.

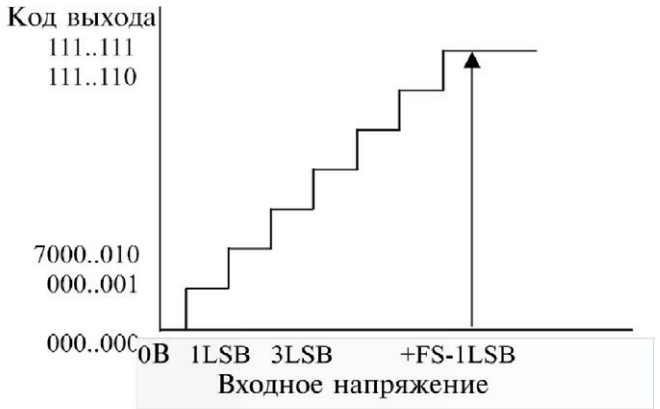


Рисунок 7.5. — Функция передачи АЦП ADuC812

SFR – интерфейс управления работой АЦП

Работа АЦП полностью контролируется тремя регистрами из SFR:

ADCCON1 — (SFR #1 управления АЦП)

MD1	DM0	CK1	CKO	AQ1	AQ2	T2C	EXC

Регистр ADCCON1 управляет преобразованием, временем переключения, режимами преобразования и токопотреблением устройства.

Адрес SFR 0EFH

Значение SFR по включению питания 20H

Наличие битовой адресации нет

В таблице 7.4 приведено распределение битов регистра ADCCON1

Таблица 7.4. Распределение битов SFR-регистра ADCCON1

Расположение бит	Мнемоника	Описание
ADCCON1.7 ADCCON1.6	MD1 MD0	(MDO M01) биты режима выбирают режимы работы АЦП следующим образом: MD1 MDO Режим АЦП 0 0 Дежурный 0 1 Нормальный 1 0 Дежурный, если не выполняется цикл преобразования 1 1 Холостой, если не выполняется цикл преобразования
ADCCON1.5 ADCCON1.4	CK1 CK0	Биты деления тактовой частоты, выбирают коэффициент деления основной частоты микропроцессора для получения тактовой частоты АЦП. Цикл преобразования АЦП занимает 16 тактов, в дополнение к числу тактов переключения (см. ниже об AQ0-AQ1). Коэффициент выбирается из: CK1 CKO Делитель для MCLK 0 0 1 0 1 2 1 0 4 1 1 8
ADCCON1.3 ADCCON1.2	AQ1 AQ0	Биты задержки переключения, выбирают время, необхо- димое для перезарядки УВХ при переключении мульти- плексора: AQ1 AQ0 Число тактов задержки запуска АЦП 0 0 1 0 1 2 1 0 3 1 1 4 Примечание: При импедансе входного источника сигналов менее 8 КОм выбор (AQ1-AQ0=00 т.е. 1 такт задержки). В противном случае задержку увеличивают до 2,3 или 4 тактов.
ADCCON1.1	T2C	Бит запуска преобразования от Таймера2. Если бит установлен, то сигнал переполнения Таймера 2 используется для запуска АЦП.
ADCCON1.0	EXC	Бит разрешения внешнего запуска. Если установлен, то контакт 23 (CONVST/) будет использоваться как сигнал запуска (активный низкий должен быть не менее 100нс)

Замечание: если АЦП находится в Холостом Режиме, Vref удерживается включенным, в то время как в Дежурном Режиме с целью минимизации потребления вся периферия АЦП

выключена. Среднее потребление тока блоком АЦП составляет 1.6мА при Vdd=5В.

ADCCON2 — (SFR #2 управления АЦП)

ADCI	DMA	CCONV	SCONV	CS3	CS2	CS1	CS0

Регистр ADCCON2 управляет выбором номера канала и режимами преобразования.

Адрес SFR	0D8H
Значение SFR по включению питания	00H
Наличие битовой адресации	есть

В таблице 7.5 приведено распределение битов регистра ADCCON2

ADCCON3 — (SFR #3 управления АЦП)

BUSY	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD	RSVD

Регистр ADCCON3 дает индикацию занятости АЦП для прикладных программ.

Адрес SFR	0F5H
Значение SFR по включению питания	00H
Наличие битовой адресации	нет

В таблице 7.6 приведено распределение битов регистра ADCCON3

Таблица 7.6. Распределение битов SFR-регистра ADCCON3

Расположение бит	Мнемоника	Описание
ADCCON3.7	BUSY	Бит занятости АЦП только для чтения. Устанавливается на время преобразования или калибровки АЦП. Автоматически снимается по завершению циклов преобразования или калибровки.
ADCCON3.6… ADCCON3.0	RSVD	Биты ADCCON3.6…ADCCON3.0 зарезервированы, их при программировании следует записывать только нулями.

Таблица 7.5. Распределение битов SFR-регистра ADCCON2

Расположение бит	Мнемоника	Описание
ADCCON2.7	ADCI	Бит прерывания АЦП устанавливается аппаратно по концу однократного цикла преобразования АЦП или по концу передачи блока в режиме КПДП. ADCI очищается аппаратно при переходе по вектору на Процедуру Обслуживания Прерывания.
ADCCON2.6	DMA	Бит разрешения режима КПДП. Устанавливается пользователем для начала операции КПДП со стороны АЦП.
ADCCON2.5	CCONV	Бит циклического преобразования. Устанавливается пользователем для установки АЦП в режим непрерывного циклического преобразования. В этом режиме АЦП выполняет преобразование в соответствие с типом синхронизации и конфигурацией каналов, выбранными в других SFR.
ADCCON2.4	SCONV	Бит запуска однократного преобразования. Устанавливается пользователем для однократного запуска АЦП. Бит сбрасывается автоматически по завершению преобразования.
ADCCON2.3 ADCCON2.2 ADCCON2.1 ADCCON2.0	CS3,CS2 CS1,CS0	Биты выбора входных каналов (CS3..CS0). Позволяют пользователю осуществлять выбор номера канала АЦП под управлением программы. Преобразование будет выполняться для канала, номер которого указан данными битами. В режиме КПД выбор номера канала осуществляется из 10 канала, записанного во внешней памяти. CS3 CS2 CS1 CSO CH 0 n2 n1 n0 Номер входного канала (n2, n1, n0) 1 0 0 0 Температурный сенсор (внутренний) 1 X X X Другие комбинации 1 1 1 1 Останов КПДП

Встроенный источник опорного напряжения (ИОН) АЦП

Если используется внутренний ИОН оба контакта Vref и Cref должны быть блокированы конденсаторами 100нФ на аналоговую землю AGND. Емкости следует располагать к контактам так близко, как только возможно. Для правильной работы устройства при использовании внешнего ИОН его величина должна быть в пределах от 2.3 В до аналогового питания AVdd.

Если требуется, чтобы внутренний ИОН использовался вне устройства, его необходимо буферизовать от контакта Vref, конденсатор 100нФ на AGND так же следует использовать. Внутренний ИОН калибруется на заводе с точностью 2.5 В +/-50мВ. Следует отметить, что внутренний ИОН будет выключен до тех пор пока либо ЦАП либо АЦП не будут включены соответствующими битами разрешения.

Калибровка

Блок АЦП имеет четыре SFR, ответственные за проведение калибровки. Эти регистры управляют логикой калибровки, всегда гарантируя оптимальную работу 12-битного АЦП. Будучи частью логики инициализации по включению питания, эти регистры автоматически и прозрачно для пользователя загружаются константами, запрограммированными при производстве устройства. Во многих приложениях использование заводских констант является достаточным, однако иногда для компенсации ошибок коэффициента усиления и смещения нуля всей системы в целом заводские константы могут быть подавлены пользовательскими, загружаемыми в SFR.

Обзор калибровки

Блок АЦП включает в себя аппаратуру, которая всегда гарантирует оптимальную работу АЦП. Режимы калибровки выполняются как часть заводских процедур конечного тестирования. Результаты заводской калибровки записываются в ЭРПЗУ и автоматически перегружаются в регистры калибровки при инициализации АЦП по включению питания. Во многих приложениях эта функция автокалибровки является достаточной. В противном случае, для компенсации значительных изменений эксплуатационных условий (например, тактовой частоты, диапазона входных сигналов, напряжения питания или ИОН), калибровку можно

выполнить с помощью пользовательских программ.

Это свойство встроенной программной калибровки позволяет пользова- телю ликвидировать системные ошибки (какой бы характер они не носили: внутренний или внешний) и использовать весь динамический диапазон АЦП путем подстройки диапазона входных сигналов для каждой конкретной системы. Для получения дополнительной информации по применению процедур калибровки в ваших конкретных приложениях необходимо связаться с фирмой Analog Devices.

Режимы работы АЦП

Типовая работа

Как только АЦП сконфигурирован с помощью ADCCON 1-3, он начнет преобразовывать аналоговые входные сигналы и давать 12-битные выходные коды в SFR: ADCDATAH(L). В четырех верхних битах ADCDATAH будет записан код выбора канала результата. Формат 12-разрядного слова результата показан на рис. 7.6.

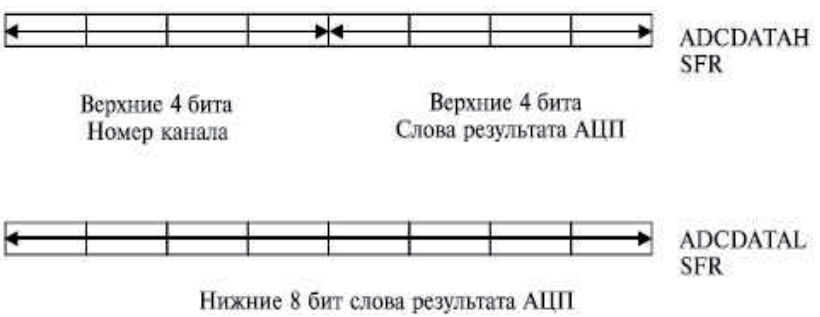


Рис. 7.6.− Формат слова результата АЦП

Режим канала прямого доступа (КПДП) к памяти.

Внутренний АЦП сконструирован таким образом, что может осуществлять выборки каждые 5мксек (частота выборок 200КГц). Таким образом, от пользовательских программ требуется обслужить прерывание, прочитать с АЦП результат и записать его для дальнейшей обработки, все следует выполнить в течение 5мксек, иначе результат следующей выборки можно потерять. Для приложений, где устройство не может поддерживать высокую скорость обработки прерываний существует режим КПДП(DMA) АЦП к внешней памяти.

Режим КПДП разрешается битом разрешения КПДП (ADCCON2.6), позволяющим АЦП выполнять циклические выборки, как при конфигурировании через ADCCON SFR. Результат каждой выборки записывается во внешнюю статическую память (SRAM), минуя микропроцессорное ядро. Этот режим работы гарантирует, что устройство может выполнять цикли- ческие выборки с максимальной скоростью. До разрешения режима КПДП пользователь сначала должен разметить внешнюю память, в которую будут записываться выборки. Разметка состоит в записи идентификаторов номеров каналов ID (четыре старших бита) во внешней памяти. На рис. 7.7. показана типовая разметка внешней памяти.

Рисунок 7.7. — Типовая разметка внешней памяти для режима КПДП

После разметки заносится значение указателя памяти КПДП (DMAP, DMAH и DMAL) SFRs. В этих SFRs следует указывать стартовый адрес КПДП во внешней памяти. Например, 000000H, как на рис. 7.7. 3-х байтовый стартовый адрес следует записывать в следующем порядке: DMAL, DMAH и DMAP. Конец таблицы КПД обозначается записью «1 1 1 1» в поле выбора канала. Теперь, для запуска КПДП и передачи результатов в последовательные ячейки внешней памяти можно установить бит разрешения (ADCCON2.6,). Следует помнить, что режим КПДП включится только тогда, когда пользователь предварительно установит время преобразования и режим запуска через SFR ADCCON1 и 2. Конец КПДП- преобразования устанавливается битом прерывания АЦП в ADCCON2.7. По окончанию КПДП внешняя память данных окажется загруженной новыми результатами работы АЦП, как показано на рис. 7.8. Следует отметить, что результаты разметки сохраняются.

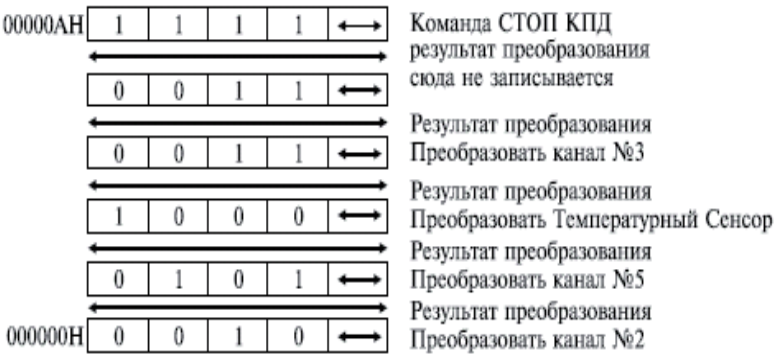


Рисунок 7.8. — Типовое содержание внешней памяти после окончания режима КПДП

Микрооперации во время выполнения режима КПДП

Во время выполнения КПДП ядро ОМЭВМ свободно для выполнения кода программы, включая внутреннее обслуживание и связь. Однако, следует

особо отметить, что доступ MCU к Портам2 и 3 (которые безусловно используются контроллером КПДП) во время выполнения КПДП блоком АЦП запрещен. Это означает, что если даже при выполнении программы встретится обращение к Портам2 или 3, данных на внешних контактах этих портов не будет. Как только требуемый блок данных по КПДП будет набран и записан во внешнюю память, микроконтроллер выполняет прерывание, что позволяет выполнять последующую обработку данных без потери времени

7.4. Блок ЦАП. SFR-интерфейс к блокуЦАП.

ADuC812 на кристалле содержит два 12-битных ЦАПа с временем установления 15 мкс каждый. Один SFR управления и четыре SFR данных осуществляют управление работой ЦАП:

DAC0L/DAC1L — содержат младших 8 бит байта ЦАП

DAC0H/DAC1H — содержат старших 4 бита байта ЦАП

DACCON — содержат биты управления общего назначения для контроля ЦАП.

При нормальной работе каждый ЦАП модифицируется только тогда, когда записывается младший ниббл (тетрада) SFR (DACxL). Можно модифицировать оба ЦАПа одновременно путем использования бита SYNC в DACCON SFR.

При 8-ми битной работе байт, записанный в регистры DACxL, автоматически направляется в верхнюю часть 12-битного регистра ЦАП. Распределение бит DACCON SFR показано в таблице 7.7.

DACCON (SFR- управление ЦАП)

MODE	RNG1	RNG0	CLR1	CLR0	SYNC	PD1	PD0

Адрес SFR 0FDH

Значение SFR по включению питания 04H

Наличие битовой адресации нет

Таблица 7.7. Распределение битов SFR DACCON

Расположение бит	Мнемоника	Описание
DACCON. 7	MODE	Бит устанавливает режим работы обоих ЦАП. Если = 1, то 8-ми битный (запись 8-ми битов в DACxL SFR). Если = 0, то 12-битный.
DACCON.6	RNG1	Бит выбора диапазона ЦАП1. Если = 1, то диапазон ЦАП1 0 .. Vdd. Если = 0, то диапазон ЦАП1 0 .. Vref.
DACCON.5	RNGO	Бит выбора диапазона ЦАПО. Если = 1, то диапазон ЦАПО 0 .. Vdd. Если = 0, то диапазон ЦАПО 0 .. Vref.
DACCON.4	CLR1	Бит очистки ЦАП1. Если = 1, то выход ЦАП1 соответствует коду. Если = 0, то выход ЦАП1 = 0В.
DACCON.3	CLRO	Бит очистки ЦАПО. Если = 1, то выход ЦАПО соответствует коду. Если = 0, то выход ЦАПО = 0В.
DACCON.2	SYNC	Бит синхронизации ЦАПО/1. Если = 1, то выходы ЦАПов изменяются сразу, как только данные попадают в регистры DACxL SFRs. Пользователь может одновременно обновить выходы обоих ЦАПов путем предварительной записи данных в DACxL/H при SYNC = 0. Выходы обоих ЦАПов одновременно обновятся теперь при установке SYNC= 1.
DACCON.1	PD1	Бит выключения ЦАП1. Если = 1, то ЦАП1 включен. Если = 0, то ЦАП1 выключен.
DACCON.O	PDO	Бит выключения ЦАПО. Если = 1, то ЦАПО включен. Если = 0, то ЦАПО выключен.

7.5. Неразрушаемая Flash-память

Обзор Flash-памяти.

ADuC812 включает в себя внутреннее ЭРПЗУ, выполненное по FLASH технологии для предоставлению пользователю не разрушаемой, программируемой в системе памяти программ (кода) и данных. FLASH ЭРПЗУ — новейший тип в технологии памяти и основывается на архитектуре однотранзисторной ячейки. Эта технология вышла из известной технологии создания ЭПЗУ и была разработана в конце 1980-х годов. FLASH память обладает гибкостью программирования в системе (изделии), присущей электрически стираемой программируемой памяти (ЭСПЗУ) и минимальным объемом, присущем электрически программируемой памяти (ЭПЗУ) (см. рис. 7.9.).

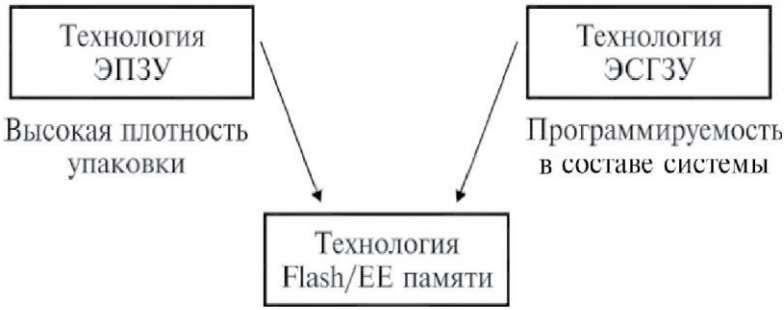


Рисунок 7.9. – Разработка FLASH памяти

Так как FLASH-технология базируется на архитектуре однотранзисторной ячейки, то FLASH память, подобно ЭПЗУ, можно применять в изделиях, где требуется очень высокая плотность размещения памяти.

Подобно ЭСПЗУ FLASH память можно программировать в составе системы на уровне байтов, хотя прежде она должна быть стерта; причем,

стирание выполняется блоками. Таким образом, FLASH память часто и, более правильно, называют FLASH/EE память (с электрическим стиранием).

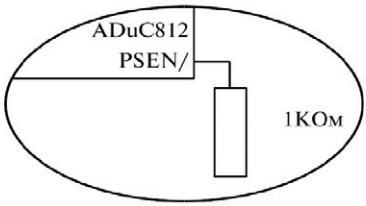


Рисунок 7.10. — Программирование FLASH/EE памяти в режиме последовательной загрузки

В итоге, FLASH/EE память представляет следующий шаг в направлении идеального устройства памяти, обладающего свойством не разрушаемости, программируемостью в составе системы, высокой плотностью упаковки и низкой стоимостью. FLASH/EE память в составе ADuC812 позволяет модифицировать программный код дистанционно в узлах системы без необходимости их смены в случае однократно программируемых устройств (OTP).

FLASH/EE память в ADuC812.

Для приложений пользователя ADuC812 предоставляет два массива FLASH/EE памяти:

- 8К байт внутренней FLASH/EE памяти программ для выполняемого кода без необходимости установки внешней дискретной памяти ПЗУ. Эту память можно программировать стандартными программаторами от различных производителей. Кроме того, данную память можно программировать в составе системы, используя имеющийся режим последовательной загрузки.
- 640 байт внутренней FLASH/EE памяти данных. Она может использоваться как не разрушаемая блокнотная память данных общего применения. Пользователь получает доступ к данной памяти через группу из шести SFR

регистров. Память можно программировать на байтовом уровне, хотя, сначала, ее следует стереть 4-х байтовыми секторами.

Использование FLASH/EE памяти программ.

Это 8Кбайт FLASH/EE памяти программ в нижней части 64Кбайт полной памяти программ, адресуемой устройством и они используются для пользовательского кода его приложений. Память программ может быть запрограммирована одним из 2-х способов:

1. Последовательная загрузка (программирование в составе системы)

ADuC812 обладает программой загрузки кода через стандартный асинхронный последовательный порт (UART), являющейся частью заводского загрузчика. Режим последовательной загрузки включается автоматически при подаче питания, если контакт PSEN/ подключен через внешний резистор на землю, как показано на рис. 7.10. Находясь в этом режиме, пользователь может загружать код в память программ в то время, как его устройство находится в составе аппаратуры конечной системы. Программа загрузки с ПЭВМ так же существует как часть системы разработки QuickStart для ADuC812. Протокол последовательной загрузки детализирован в заметках по применению ADuC812 и может быть получен на сайте фирмы-производителя микроконтроллера.

2. Параллельное программирование

Режим параллельного программирования полностью совместим с работой стандартных программаторов FLASH/EE памяти, поставляемых различными поставщиками. На рис. 7.11 приводится блок-схема и конфигурация внешних контактов, требуемых для поддержки параллельного программирования. В этом режиме Порты P0, P1 и P2 работают как интерфейсные магистрали

внешних данных и адреса, сигнал ALE служит стробом разрешения записи, а Порт P3 используется в качестве порта общей конфигурации, задающего при параллельном программировании режимы программирования и стирания. Источник высокого напряжения (12В), необходимый для программирования FLASH/EE памяти выполнен на кристалле в виде повышающего преобразователя — «зарядного насоса».

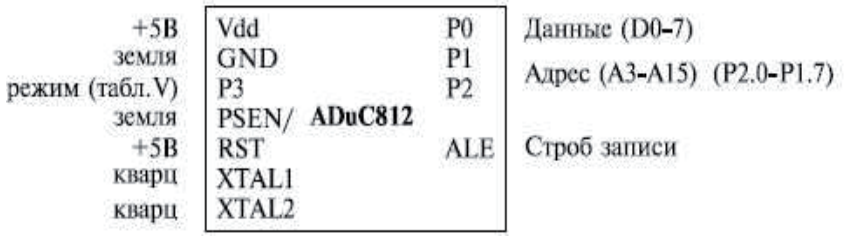


Рисунок 7.11. — Параллельное программирование FLASH/EE памяти

В таблице 7.8 показаны режимы программирования, которые могут быть реализованы с помощью Порта 3.

Таблица 7.8. Режимы программирования FLASH/EE памяти

Контакты Порта (P3.O 2-P3.7 17) 7 6 5 4 3 2 1 0	Режим программирования
1 X X X 0 0 0 1	Стирание FLASH программ. Стирание FLASH пользователя
1 X X X 0 0 1 1	Чтение идентификаторов производителя и кристалла
1 X X X 0 1 0 1	Программирование байта
1 X X X 0 1 1 1	Чтение байта
1 X X X 1 0 0 1	Зарезервирована
1 X X X 1 0 1 1	Зарезервирована
Остальные коды	Резервные

Использование памяти данных

Память данных пользователя состоит из 640 байт, которые составляют 160 (от 00H до 9FH) 4-байтовых страниц, как показано на рис. 7.13. Как и для прочей периферии доступ к этой памяти производится через SFR регистры. Группа из 4-х регистров (EDATA1-4) используется для хранения данных 4-х байт страницы из последнего обращения. EADRL используется для хранения адреса страницы, куда будет осуществляться доступ. И, наконец, ECON — 8-битный регистр управления, в который записывается одна из пяти команд управления доступом к памяти, допускающих различные операции чтения, записи, стирания и верификации. Блок схема регистрового интерфейса к памяти пользователя показана на рис. 7.12.

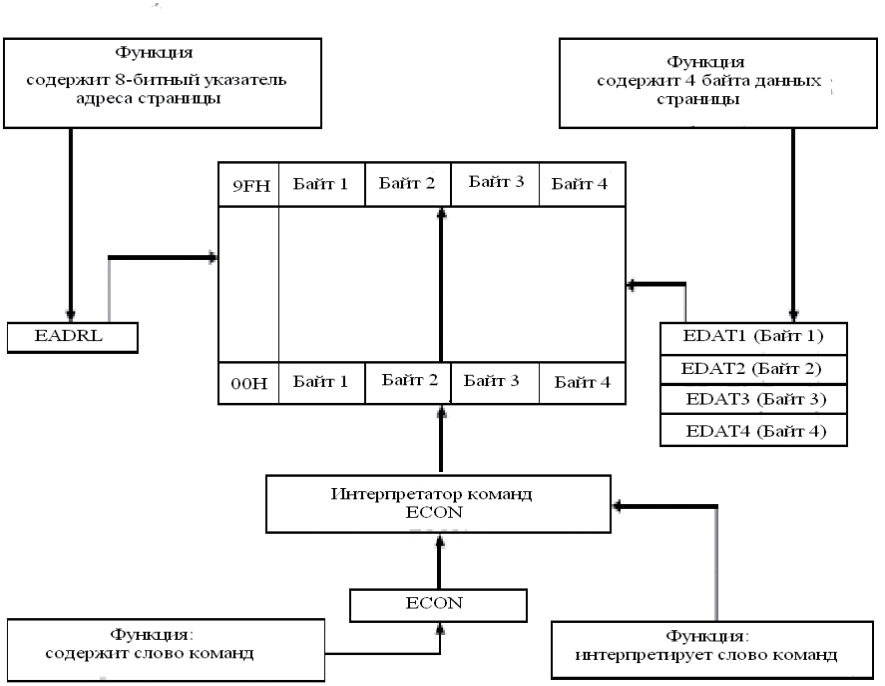


Рисунок 7.12. — Управление и конфигурация FLASH/EE памяти пользователя

Рисунок 7.13. — Конфигурация FLASH/EE памяти пользователя

ECON — регистр управления памятью.

Регистр является интерпретатором команд и в него можно записать одну из пяти команд различных циклов чтения, программирования и стирания, как указано в таблице 7.9

Таблица 7.9. Регистр управления памятью ECON

Байт управления	Команда
01H	Команда Чтения. Результаты заносятся в регистры EDATA 1-4 со страницы, адрес которой содержится в EADRL.
02H	Команда Записи. Данные, содержащиеся в 4-х байтах (EDATA 1-4) записываются в память по адресу, указанному в EADRL. Предполагается, обозначенная для записи страница предварительно стерта.
O3H	Резервная команда. Не использовать.
04H	Команда Верификации. Позволяет пользователю проверифицировать данные/ которые содержатся в EDATA 1-4 с уже записанными по адресу указателя EADRL. Следующее чтение ECON SFR даст ноль, если верификация правильна и не ноль, в противном случае.
05H	Команда Стирания. Приводит к стиранию 4-байтовой страницы, адрес которой указан в EADRL.
6H	Команда Стирать Все. Приводит к стиранию всей памяти пользователя 160-станиц (640 байт).
07H .. FFH	Резервные команды. Зарезервированы для дальнейшего применения.

Временные соотношения при записи и стирании FLASH/EE памяти

Средние временные соотношения для FLASH/EE памяти составляют:

Стирание всего массива (640 байт)	20мсек
Стирание одной страницы (4 байта)	20мсек
Программирование страницы (4 байта)	250мксек
Чтение страницы (4 байта)	1 командный цикл.

Использование интерфейса к FLASH/EE памяти

Как в случае памяти программ, данная память может быть запрограммирована в составе системы по байтно, при этом, конечно, она предварительно должна быть стерта страничными блоками.

Типовой цикл доступа к FLASH/EE памяти включает в себя установку адреса страницы доступа EADRL SFR, запись данных для программирования в EDATA 1-4 (в случае чтения — не записываются) и, наконец, запись команды в ECON, инициирующей действие в соответствие с таблицей 7.9.

Следует отметить, что заданный режим работы инициируется по записи слова команды в ECON SFR. При этом, микропроцессорное ядро переходит в холостой режим и находится там до тех пор, пока выполнение команды не завершится.

На практике это означает, что даже если режим работы с FLASH/EE памятью инициируется 2-мя машинными циклами (инструкция MOV для записи в ECON SFR), следующая инструкция будет выполнена только после окончания цикла обслуживания FLASH/EE памяти (т.е. спустя 250мксек или 20мсек). Это означает, что ядро не будет обслуживать запросы на прерывание до тех пор, пока операция с FLASH/EE памятью не завершится, хотя функции управления ядра периферией будут выполняться, как, например, продолжение счета времени/событий Счетчиками/Таймерами на протяжении всего псевдохолостого режима.

Стирание всей памяти

Хотя 640-байтовая FLASH/EE память пользователя с завода отгружается стертой т.е. в ячейки записан код FFH, является хорошей практикой при программировании включать цикл Стереть Всю Память при выполнении процедур ее реконфигурирования. Команда Стереть Все состоит в записи в регистр ECON SFR кода 06H, при этом инициируется стирание всех 640 байт памяти. На ассемблере 8051 это выглядит следующим образом:

```
MOV ECON, #06H
;Команда Стереть Все
;Длительность 20мсек
```

Программирование байта

Вообще говоря, в FLASH/EE памяти запрограммировать байт можно только тогда, когда этот байт был предварительно стерт (в ячейке предварительно записано FFH). Вследствие особенности архитектуры FLASH/EE памяти, стирание можно производить только для 1 страницы (минимум 4-байта) при инициировании Команды Стирания,

Пример процесса Байт Программирования показан на рис. 7.14. графически. В этом примере пользователь записывает код 0F3H во 2-ой байт на странице 03H пользовательской FLASH/EE памяти.

Однако, страница 03H уже содержит данные в четырех байтах, а пользователю требуется изменить только содержимое одного байта; всю страницу следует сначала прочитать с тем, чтобы можно было стереть содержимое этой страницы без потери данных Затем новый байт записывается в EDATA SFR вслед за циклом стирания.

Если попытаться начать цикл Программирования (ECON=02H), не выполняя цикла Стирания (ECON=05H), то в этом случае будут модифицированы только те биты, которые содержат единицы. Т.о. для

правильной записи массива необходимо выполнить предварительное стирание его. Так же следует отметить, что циклы стирания страницы и всей памяти имеют одинаковую длительность — 20мсек. Ассемблерный код 8051 приведенного примера выглядит следующим образом:

```
MOV EADRL,   #03H      ;Установка указателя страницы
MOV ECON,    #01H      ;Команда Чтения Страницы
MOV EDATA2,  #OF3H     ;Запись нового байта
MOV ECON,    #02H      ; Команда стирания страницы
MOV ECON,    #05H      ;Команда Программирования Страницы
```

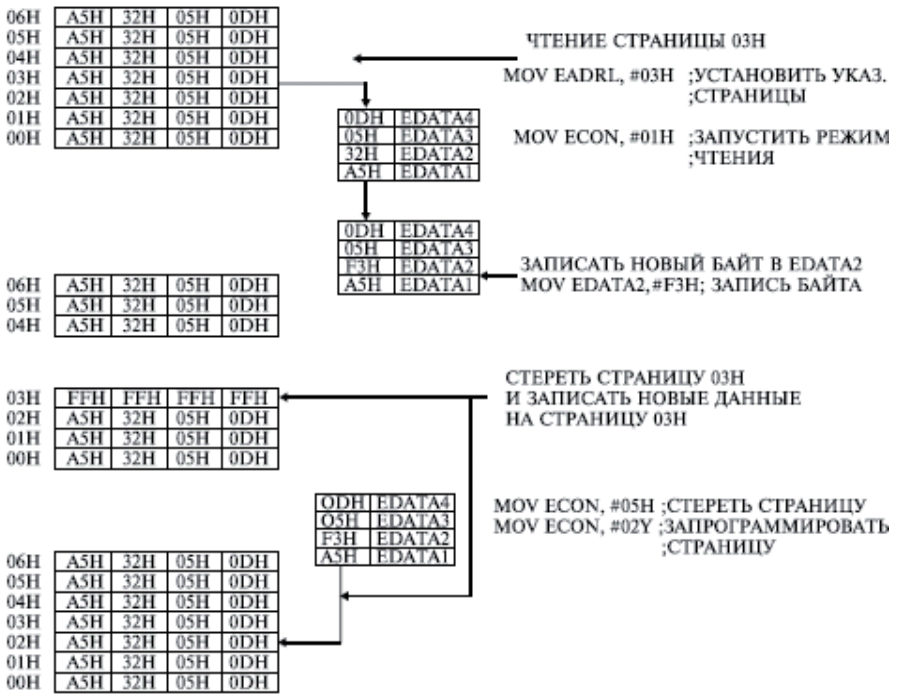


Рисунок 7.14. — Пример программирования байта памяти пользователя

7.6. Система прерываний.

ADuC812 обеспечивает обработку девяти источников и два уровня приоритета прерываний. На рис. 7.15. приводятся прерывания данного уровня в порядке убывания приоритета, здесь же дается общий обзор источников прерываний их флагов запросов и управления. Адреса векторов прерываний приводятся в таблице 7.10.

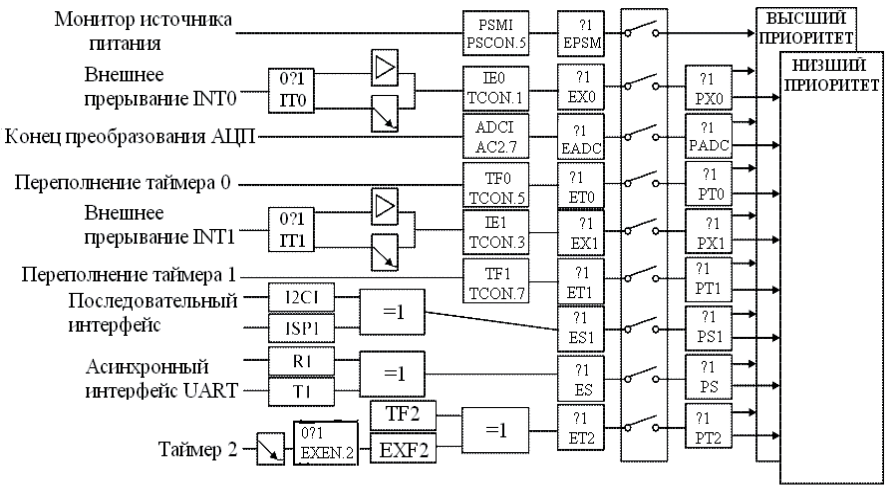


Рисунок 7.15. — Источники запроса прерываний

Таблица 7.10. Адреса векторов прерываний

Прерывание	Наименование источника прерывания	Адрес вектора	Номер прерывания
PSMI	Монитор Источника Питания	43H	1
IEO	Внешнее прерывание INTO/	O3H	2
ADCI	Конец преобразования АЦП	33H	3
TFO	Переполнение Таймера 0	OBH	4
IE1	Внешнее прерывание INT1/	13H	5
TF1	Переполнение Таймера 1	1BH	6
I2CI/ISPI	Прерывание последовательного интерфейса	3BH	7
RI/TI	Прерывание асинхронного интерфейса UART	23H	8
TF2/EXF2	Прерывание от Таймера 2	2BH	9

Использование прерываний

Для обработки любого из прерываний следует предпринять следующие три шага:

1. Расположить начало процедуры обслуживания прерывания по адресу соответствующего прерывания (см. таблицу 7.10).
2. Установить бит разрешения всех прерываний (EA) «1» в регистре IE SFR.
3. Установить бит разрешения индивидуального прерывания в «1» в IE или IE2 SFR.

Для разрешения и установки приоритета различных прерываний используются три регистра SFR. Распределение битов этих SFR приводится в таблицах 7.11, 7.12 и 7.13. Следует отметить, что в то время как регистры IE и IP SFR бит адресуемые, регистр IE2 — адресуется только байтом.

IE — SFR регистр разрешения прерывания

EA	EADC	ET2	ES	ET1	EX1	ET0	EX0

Регистр IE разрешает прерывание системе и семи источникам прерываний

Адрес SFR IE	A8H
Значение SFR по включению питания	00H
Наличие битовой адресации	есть

IE2 — (SFR 2 разрешения прерывания)

NU	NU	NU	NU	NU	NU	EPSM	ESI

IE2 регистр разрешает прерывание двум дополнительным источникам прерываний.

Адрес SFR IE2	0A9H
Значение SFR по включению питания	00H
Наличие битовой адресации	нет

Таблица 7.11. Распределение битов в регистре разрешения прерывания (IE)

Расположение битов	Мнемоника	Описание
IE.7	EA	Бит Разрешения Глобального Прерывания (EA); должен быть установлен «1» для опознания любого источника прерывания ядром. Если EA=0, все прерывания запрещены.
IE.6	EADC	Бит Разрешения Прерывания АЦП (EADC) устанавливается «1» для разрешения прерывания от АЦП.
IE.5	ET2	Бит Разрешения Прерывания по Переполнению Таймера 2 (ET2) устанавливается «1» для разрешения прерывания от Таймера 2.
IE.4	ES	Бит Разрешения Прерывания от Последовательного Порта UART (ES) устанавливается «1» для разрешения прерывания от последовательного порта.
IE.3	ET1	Бит Разрешения Прерывания по Переполнению Таймера 1 (ET1) устанавливается «1»для разрешения прерывания от Таймера 1.
IE.2	EX1	Бит Разрешения Внешнего Прерывания INT1 (EX1) устанавливается «1»для разрешения внешнего прерывания.
IE.1	ET0	Бит Разрешения Прерывания по Переполнению Таймера 0 (ET0) устанавливается «1» для разрешения прерывания от Таймера 0.
IE.0	EX0	Бит Разрешения Внешнего Прерывания INT0 (EX0) устанавливается «1» для разрешения внешнего прерывания.

Таблица 7.12. Распределение битов в регистре разрешения прерывания 2 (IE2)

Расположение бит	Мнемоника	Описание
IE2.7 IE2.6	NU NU	Не используется Не используется
IE2.5	NU	Не используется
IE2.4	NU	Не используется
IE2.3	NU	Не используется
IE2.2	NU	Не используется
IE2.1	EPSM	Бит Разрешения Прерывания по Монитору Питания устанавливается «I» для разрешения прерывания от PSM.
IE2.0	ESI	Бит Разрешения Прерывания от Интерфейсов SPI/I2C (ESI) устанавливается «I» для разрешения прерывания от данных интерфейсов.

IP — SFR регистр приоритета прерывания

PS1	PADC	PT2	PS	PT1	PX1	PT0	PX0

Регистр IP устанавливает один из двух возможных уровней прерывания для различных источников прерываний. Установите соответствующий бит в 1 для присвоения высокого уровня данному прерыванию и 0 — низкого.

Адрес SFR IP 0B8H

Значение SFR по включению питания 00H

Наличие битовой адресации есть

Таблица 7.13. Распределение битов в регистре приоритета прерываний (IP)

Расположение бит	Мнемоника	Описание
IP.7	PSI	Устанавливает приоритет прерыванию от SPI/I2C
IP.6	PADC	Устанавливает приоритет прерыванию от АЦП
IP.5	PT2	Устанавливает приоритет прерыванию от Таймера 2
IP.4	PS	Устанавливает приоритет прерыванию от последовательного порта UART
IP.3	PT1	Устанавливает приоритет прерыванию от Таймера 1
IP.2	PX1	Устанавливает приоритет прерыванию от Внешнего источника INT1
IP.1	PT0	Устанавливает приоритет прерыванию от Таймера 0
IP.0	PX0	Устанавливает приоритет прерыванию от Внешнего источника INT 0

7.7. Внутренние периферийные устройства

Следующие далее разделы представляют собой краткий обзор различных вторичных устройств периферии, имеющихся в составе кристалла. Ниже приводятся краткие данные для набора регистров SFR, используемых для управления этой периферией.

7.7.1. Параллельные порты ввода – вывода

Для обмена с внешними устройствами в составе ADuC812 имеется четыре порта общего назначения. В дополнение к функции общего ввода вывода (В-В), некоторые порты могут управлять операциями с внешней памятью, в то время как другие мультиплексируются альтернативными функциями для периферии. В общем случае, когда периферийная функция для контакта порта разрешена, тогда данный контакт не может употребляться в качестве бита порта В-В общего назначения.

Порты 0, 2 и 3 — двунаправленные, тогда как Порт 1 служит только для ввода. Все порты содержат выходную защелку и входной буфер, порты В-В содержат так же выходной буфер (драйвер). Доступ к контактам Портов 0 — 3 по Чтению и Записи выполняется через соответствующие регистры специального назначения. Контакты Портов 0, 2 и 3 можно конфигурировать независимо как для цифрового ввода так и для вывода через соответствующие биты SFR порта. В то время как контакты Порта 1 можно конфигурировать только либо на цифровой ввод/ либо на ввод аналоговый; возможность цифрового вывода по Порту 1 не поддерживается.

7.7.2. Порты последовательного ввода-вывода.

Асинхронный интерфейс (UART)

Последовательный порт полнодуплексный, что означает возможность одновременной передачи и приема. Имеется буфер приема, что обеспечивает наличие возможности приема второго байта до считывания из регистра приемника предыдущего байта. Однако, если предыдущий байт не будет считан из регистра к моменту окончания приема второго байта, то один из байтов будет утерян.

Физический интерфейс к сети последовательных данных осуществляется

через контакты интерфейс к сети последовательных данных осуществляется через контакты RxD(P3.0) и TxD(P3.1), а сам порт можно конфигурировать на четыре режима работы.

Последовательный периферийный интерфейс (SPI).

SPI является промышленным стандартным интерфейсом синхронного последовательного обмена, который допускает одновременно передавать и принимать синхронно восемь бит данных. Систему можно конфигурировать как Ведущую (Master) и как Ведомую (Slave).

I2C — совместимый последовательный интерфейс.

ADuC812 поддерживает 2-проводный I2S-совместимый последова-тельный интерфейс. Этот интерфейс можно сконфигурировать как Программно Ведущий (Software Master) или как Аппаратно Ведомый (Hardware Slave) и он мультиплексируется с Портом SPI.

7.7.3. Таймеры — счетчики.

ADuC812 содержит три 16-битных счетчика — таймера: Таймер 0, Таймер 1 и Таймер 2. Аппаратура таймеров — счетчиков включена в состав чипа для того, чтобы высвободить микропроцессорное ядро от излишних затрат ресурса, свойственных программной эмуляции процесса счета. Каждый счетчик — таймер состоит из двух 8-битных регистров THx и TLx (x=0, 1 и 2). Все три можно сконфигурировать как таймеры, либо как счетчики событий.

В режиме «Таймер» регистр TLx инкрементируется в каждом машинном цикле. Т.о. в этом режиме работу можно рассматривать как счет машинных циклов. Так как машинный цикл состоит из 12 периодов генератора, то максимальная скорость счета составляет 1/12 от частоты генератора.

В режиме «Счетчик» регистр TLx инкрементируется по перепаду 1 — 0 на соответствующем контакте микросхемы T0, T1 или T2.

7.7.4. Внутренние мониторы

Для минимизации порчи кода или данных вследствие возникновения катастрофических программных или внешних сбоев система ADuC812 включает в себя две мониторных функции. Обе мониторные функции конфигурируются через соответствующие регистры SFR.

Сторожевой таймер (WDT)

Назначение WDT — сгенерировать сигнал Сброса устройства, если ADuC812 выполняет ошибочные действия, вероятно, по причине сбоя программы, из-за электрических или электромагнитных помех. Действие WDT можно запретить очисткой бита Разрешения WDE в регистре Управления Сторожевым Таймером (WDCON) SFR. При разрешенном WDT таймер будет генерировать системный сброс если программа пользователя не обновляет его содержимое в интервале предустановленного времени. Интервал можно менять с помощью бит предустановки в диапазоне от 16мсек до 2048мсек через регистр SFR.

Монитор источника питания (PSM).

PSM генерирует прерывание, когда значение аналогового или цифрового напряжения питания падает ниже одной из пяти, устанавливаемой пользователем, пороговой величины (от 2.6В до 4.6В). Бит прерывания не будет очищаться в течение не менее 256мс и до тех пор, пока напряжение источника не станет выше порогового значения.

Эта функция гарантирует, что пользователь успеет спасти рабочие регистры во избежание возможной порчи данных из-за низкого питания, и,

что продолжение выполнения программного кода не начнется до тех пор, пока не установится «безопасный» уровень питания. Монитор питания так же защи- щен от импульсных помех в цепи прерывания.

7.8. Система разработки QuickStart.

На рис. 7.16 приведена система “Быстрого старта”

Рисунок 7.16. — Типичная конфигурация системы Быстрого старта.

Система представляет собой функционально законченный не дорогой инструмент разработки, поддерживающий устройство ADuC812. Система включает в себя следующие (основанные на ПЭВМ и Win-95 технологии) аппаратные и программные инструменты.

Разработка программного кода:

Ассемблер и С — компилятор (ограничен до 2К кода). Проверка работы: ADSIM812, Windows Симулятор. Загрузчик кода: Последовательный Загрузчик с асинхронного порта FLASH/EE памяти. Отладчик кода: Отладчик с Последовательного Порта.

Прочее: Документация на CD-ROM, источник питания, кабель последовательного порта.

7.9. Регистры Специального Назначения (SFR)

Все регистры, исключая счетчик команд и четыре банка регистров общего назначения, располагаются в области регистров специального назначения (SFR). Эти регистры включают в себя регистры управления,

конфигурирования и регистры данных, которые все обеспечивают интерфейс между MCU и внутренней периферией.

Примечания:

SFRs регистры, адреса которых оканчиваются на 0H или 8H являются бит адресуемыми.

Основной функцией Порта 1 является ввод аналоговых сигналов, по этой причине для разрешения цифрового ввода по его контактам необходимо записать «0» в соответствующие SFR биты Порта 1.

Калибровочные коэффициенты загружаются в соответствующие регистры по включению питания как величины, записанные на предприятии — изготовителе.

7.9.1. Регистры управления и конфигурации АЦП и ЦАП.

Регистр управления АЦП #1 ADCCON1.

Адрес регистра EFh, состояние после сброса системы 20h

Назначение битов регистра:

ADCCON1.7	Биты управления питанием АЦП
ADCCON1.6	(выключено, норма/авто выключено, автохолостое)
ADCCON1.5	Время преобразования =
ADCCON1.4	16/ADCCLK;ADCCLK=MCLK/ (1,2,4,8)
ADCCON1.3	Биты выбора задержки переключения
ADCCON1.2	AQT=(1,2,3,4) / ADCCLK
ADCCON1.1	Разрешение запуска от Таймера 2
ADCCON1.0	Разрешение внешнего запуска

Регистр управления АЦП #2 ADCCON2.

Адрес регистра E0h, состояние после сброса системы 00h

Назначение битов регистра:

ADCI.7	Флаг прерывания АЦП

DMA.6 Разрешение режима КПД

CCONV.5 Бит разрешения циклического преобразования

SCONV.4 Бит однократного запуска АЦПCS3.3

Биты выбора входного канала:

CS2.2 0000-0111 =ADCO-ADC7

CS1.1 1XXX=Температурный сенсор

CS0.0 1111=Команда остановки КПД(Только для режима КПД)

Регистр управления ЦАПом DACCON

Адрес регистра FDh, состояние после сброса системы 00h

Назначение битов регистра:

DACCON.7 Режим ЦАП (0=12бит, 1=8бит)

DACCON.6 Диапазон ЦАП1 (0=Vref, l=Vdd)

DACCON.5 Диапазон ЦАПО (0=Vref, l=Vdd)

DACCON.4 Очистка ЦАП1 (0=ОВ, 1=Нормальная работа)

DACCON.3 Очистка ЦАП0 (0=ОВ, 1=Нормальная работа)

DACCON.2 Синхронизация ЦАПов (1=Синхронизация)

DACCON.1 Питание ЦАП1 (0=выключено, 1=включено)

DACCON.O Питание ЦАП0 (0=выключено, 1=включено)

Регистр управления АЦП #3 ADCCON3

Адрес регистра F5h, состояние после сброса системы 00h

Назначение битов регистра:

ADCCON3.7 Флаг занятости (0=АЦП свободен)

ADCCON3.6 Бит должен содержать ноль

ADCCON3.5 Бит должен содержать ноль

ADCCON3.4 Бит должен содержать ноль

ADCCON3.3 Бит должен содержать ноль

ADCCON3.2 Бит должен содержать ноль

ADCCON3.1 Бит должен содержать ноль

ADCCON3.0 Бит должен содержать ноль

Регистры данных АЦП:

ADCDATAH Адрес регистра DAh, состояние после сброса системы 00h

ADCDATAL Адрес регистра D9h, состояние после сброса системы 00h

Указатель адреса КПД:

DMAP Адрес регистра D4h, состояние после сброса системы 00h

DMAL Адрес регистра D3h, состояние после сброса системы 00h

DMAH Адрес регистра D2h, состояние после сброса системы 00h

Регистры калибровочных коэффициентов АЦП.

ADCGAINH ADCGAINL Калибровочный коэффициент по усилению.

Адрес регистра F2h, состояние после сброса системы 00h.

ADCOFSH ADCOFSL Калибровочный коэффициент по смещению.

Адрес регистра F1h, состояние после сброса системы 00h.

Регистры данных ЦАП1

DAC1H Адрес регистра FCh, состояние после сброса системы 00h

DAC1L Адрес регистра FBh, состояние после сброса системы 00h

Регистры данных ЦАП0

DAC0H Адрес регистра FAh, состояние после сброса системы 00h

DAC0L Адрес регистра F9h, состояние после сброса системы 00h

7.9.2. Регистры SFR ядра 8051, встроенных мониторов, Flash памяти данных.

Регистр Порта 0.

P0 (A0-A7, D0-D7), адрес регистра 80h, состояние после сброса системы

FFh.

Регистр Порта1.

P1 (ввод), адрес регистра 90h, состояние после сброса системы FFh.

Назначение битов:

T2EX.1	Таймер/Счетчик 2 Триггер Захвата/Перезагрузки
T2.0	Внешний вход Таймера/Счетчика 2

Регистр Порта2.

P2 (A8-A15, A16-A23), адрес регистра A0h, состояние после сброса системы FFh.

Регистр Порта3.

P3, адрес регистра B0h, состояние после сброса системы FFh.

Назначение битов:

RD.7	Строб чтения внеш. памяти данных
WR.6	Строб записи во внеш. память данных
T1.5	Внешний вход Таймера/Счетчика 1
T0.4	Внешний вход Таймера/Счетчика 0
INT1.3	Внешнее прерывание 1
INT0.2	Внешнее прерывание 0
TxD.1	Выход передатчика последовательного порта
RxD.0	Вход приемника последовательного порта

Регистр управления последовательной связью SCON.

Адрес регистра 98h, состояние после сброса системы 00h.

Назначение битов регистра:

SM0.7	Биты управления скоростью
SM1.6	00-8 бит регистр сдвига, Fзг/12
	01-8 бит UART, частота переполнений Таймера2/32*2
	10-9 бит UART, Fзг/64*2

	11-9бит UART ,частота переполнений Таймера2/32*2
SM2.5	В режимах 2и3 разрешает многопроцессорную связь
REN.4	Бит разрешения приема
TB8.3	В режимах 2и3 9-й переданный бит
RB8.2	В режимах 2и3 9-й принятый бит
TI.1	Флаг прерывания передатчика
RI.0	Флаг прерывания приемника

WDCON Регистр управления WDT.

Адрес регистра C0h, состояние после сброса системы 00h.

Назначение битов регистра:

PRE2.7	Биты выбора тайм-аута WDT
PRE1.6	Тайм-аут(16, 32, 64, 128,256,512,
PRE0.5	1024, 2048) мсек
WDR1.3	Биты обновления WDT
WDR2.2	Устанавливать последовательно
WDS.1	Флаг состояния WDT
WDE.O	Разрешение WDT

Регистр управления монитором питания PSMCON.

Адрес регистра DFh, состояние после сброса системы DCh.

Назначение битов регистра:

PSMCON.7	Не используется
PSMCON.6	Бит состояния PSM (1=норма, 0=сбой)
PSMCON.5	Бит прерывания от PSM
PSMCON.4, PSMCON.3,	Биты установки порога срабатывания
PSMCON.2	[4.63B, 4.37B, 3.08B, 2.93B, 2.63B]
PSMCON. 1	Слежение (1=аналог., 0=цифровое)
PSMCON.0	Управление питанием PSM (1=вкл)

Буферный регистр последовательного порта SBUF.

Адрес регистра 99h, состояние после сброса системы 00h.

Регистр управления питанием PCON.

Адрес регистра 87h, состояние после сброса системы 00h.

Назначение битов регистра:

PCON.7	Удвоение скорости передачи
PCON.4	Запрет строба «ALE» (0=норма, 1=ALE-высокий уровень)
PCON.3	Флаг общего назначения
PCON.2	Флаг общего назначения
PCON.1	Бит выключения питания (восстанавливается при аппаратном сбросе)
PCON.O	Управление Холостым Режимом (восстанавливается при разрешенном прерывании)

Слово состояния программы PSW.

Адрес регистра D0h, состояние после сброса системы 00h.

Назначение отдельных битов:

CY.7	Флаг переноса
AC.б	Флаг вспомогательного переноса
F0.5	Флаг общего назначения 0
RS1.4	Биты выбора банка регистров
RS0.3	Активный банк = [0, 1, 2, 3]
OV.2	Флаг переполнения
F1.1	Флаг общего назначения 1
P.0	Четность аккумулятора

Указатель страницы данных DPP.

Адрес 84h, состояние после сброса системы 00h.

Указатель данных DPTR.

DPH адрес 83h, состояние после сброса системы 00h. DPL адрес 82h, состояние после сброса системы 00h.

Аккумулятор ACC.

Адрес E0h, состояние после сброса системы 00h.

Вспомогательный регистр B.

Адрес регистра F0h, состояние после сброса системы 00h.

Указатель Стека SP.

Адрес регистра 81h, состояние после сброса системы 07h.

Регистр команд управления FLASH памятью данных — ECON.

Адрес регистра B9h, состояние после сброса системы 00h.

Регистр адреса Flash памяти данных — EADRL.

Адрес регистра C6h, состояние после сброса системы 00h.

Регистры памяти Flash памяти данных.

EDATA1 Адрес регистра BCh, состояние после сброса системы 00h.

EDATA2 Адрес регистра BDh, состояние после сброса системы 00h.

EDATA3 Адрес регистра BEh, состояние после сброса системы 00h.

EDATA4 Адрес регистра BFh, состояние после сброса системы 00h.

Регистры синхронизации Flash памяти данных

ETIM1 Адрес регистра BAh, состояние после сброса системы 52h.

ETIM2 Адрес регистра BBh, состояние после сброса системы 04h.

ETIM3 Адрес регистра C4h, состояние после сброса системы C9h.

7.9.3. SFR регистры управления Прерыванием, Таймером и Интерфейсами SPI и I2C.

Регистр разрешения прерывания #1IE

Адрес регистра A8h, состояние после сброса системы 00h.

Назначение битов регистра:

EA.7	Разрешение прерываний (0=всепрерывания запрещены)
EADC.6	Разрешение прерываний от АЦП
ET2.5	Разрешение TF2/EXF2 прерываний по переполнению Таймера2
ES.4	Разрешение прерываний от последовательного порта UART
ET1.3	Разрешение TF1 прерываний по переполнению Таймера 1
EX1.2	Разрешение внешних прерываний 1(INT 1)
ET0.1	Разрешение TFO прерываний по переполнению Таймера 0
EX0.0	Разрешение внешних прерываний 0 (INT 0)

Регистр разрешения прерывания #2 IE2.

Адрес регистра A9h, состояние после сброса системы 00h.

Назначение битов регистра:

IE2.1	Разрешение прерываний от монитора питания PSMI
IE2.0	Разрешение прерываний от интерфейса ISPI / I2CI

Регистр приоритета прерывания IP.

Адрес регистра B8h, состояние после сброса системы 00h.

Назначение битов регистра:

PSI.7	Приоритет интерфейса I2CI/ISPI
PADC.6	Приоритет АЦП
PT2.5	Приоритет переполнения Таймера2
PS.4	Приоритет послед, порта RI/TI

PT1.3 Приоритет переполнения Таймера 1

PX1.2 Приоритет внешнего прерывания 1 (INT 1)

PT0.1 Приоритет переполнения Таймера 0

PX0.0 Приоритет внешнего прерывания 0 (INT 0)

Регистр режима таймера TMOD.

Адрес регистра 89h, состояние после сброса системы 00h.

Назначение битов регистра:

TMOD.3/.7 Бит разрешения управления от входов INTx (0-игнорируется управление от входов INTx)

TMOD.2/.6 Бит выбора режима Счетчик/Таймер (0=Таймер)

TMOD.1/5. Биты выбора режима таймера

TMOD.0/.4 Биты (О-3): Таймер 0, (4-7):Таймер 1

Регистр управления SPI — SPICON.

Адрес регистра F8h, состояние после сброса системы 00h.

Назначение битов регистра:

ISPI.7 Прерывание от SPI (устанавливается в конце передачи)

WCOL.6 Флаг ошибки по Столкновению при записи

SPE.5 Разрешение SPI (0=запрет)

SPIM.4 Выбор режима Ведущий (0= Ведомый)

CPOL.3 Выбор полярности синхронизации

SPHA.2 Выбор фазы синхронизации (защелкивание по переднему фроту)

SPR1.1 Биты выбора скорости обмена по SPI.

SPRO.O Скорость =Fзг/[4, 8, 32, 64]

Адресный регистр интерфейса I2C — I2CADD.

Адрес регистра 9Bh, состояние после сброса системы 00h.

Регистр данных интерфейса I2C — I2CDAT.

Адрес регистра 9Ah, состояние после сброса системы 00h.

Регистр управления таймером TCON

Адрес регистра A8h, состояние после сброса системы 00h

Назначение битов регистра:

TF1.7 Флаг переполнения Таймера 1 (автоматически очищается по ISR)

TR1.6 Бит управления работой Таймера 1 (0=выключен,1=включен)

TF0.5 Флаг переполнения Таймера 0 (автоматически очищается по ISR)

TR0.4 Бит управления работой Таймера 0 (0=выключен,1=включен)

IE1.3 Флаг внешнего прерывания 1 (INT 1) (автоматически очищается по ISR)

IT1.2 Тип IE1 (0=по уровню/ 1=по фронту)

IE0.1 Флаг внешнего прерывания 0 (INT 0) (автоматически очищается по ISR)

IT0.0 Тип IE0 (0=по уровню, 1=по фронту)

Регистры Таймера 0

TH0. Адрес регистра 8Ch, состояние после сброса системы 00h

TL0. Адрес регистра 8Ah, состояние после сброса системы 00h

Регистры Таймера1

TH1. Адрес регистра 8Dh, состояние после сброса системы 00h

TL1. Адрес регистра 8Bh, состояние после сброса системы 00h

Регистр управления Таймером 2 T2CON

Адрес регистра C8h, состояние после сброса системы 00h

Назначение битов регистра:

TF2.7 Флаг переполнения Таймера 2

EXF2.6 Внешний флаг

RCLK.5 Разрешение синхронизации приемника (0= используется Таймер1)

TCLK.4 Разрешение синхронизации передатчика (1= используется Таймер 1)

EXEN2.3 Разрешение внешнего (0=игнорировать T2EX, 1=CAP/RL)

TR2.2 Бит управления работой Таймера 2 (0=выключен, 1= включен)

CNT2.1 Бит выбора режима Счетчик/Таймер (0=Таймер, 1=Счетчик)

CAP2.0 Выбор режима Захват/Перезагрузка (0=Перезагрузка, 1= Захват)

Регистры Таймера2

TH2 Адрес регистра CDh, состояние после сброса системы 00h

TL2 Адрес регистра CCh, состояние после сброса системы 00h

Регистры таймера2 Захват/Перезагрузка

RCAP2H Адрес регистра CBh, состояние после сброса системы 00h

RCAP2L Адрес регистра CAh, состояние после сброса системы 00h

Регистр данных SPI — SPIDAT.

Адрес регистра F7h, состояние после сброса системы 00h

Регистр управления I2C — I2CCON

Адрес регистра E8h, состояние после сброса системы 00h

Назначение битов регистра:

MD0.7 Выход последовательных данных в режиме Ведущий

MDE.6 Бит разрешения Выхода последовательных данных в режиме Ведущий

MC0.5 Бит синхронизации в режиме Ведущий

MDI.4 Вход последовательных данных в режиме Ведущий

I2CM.3 Выбор режима Ведущий

I2CRS.2 Сброс последовательного порта

I2CTX.1 Состояние направления передачи

7.10. Диаграммы временных соотношений

Временная диаграмма синхронизации микроконтроллера

Временная диаграмма синхронизирующего сигнала на входе XTAL 1 приведена на рисунке 7.17.

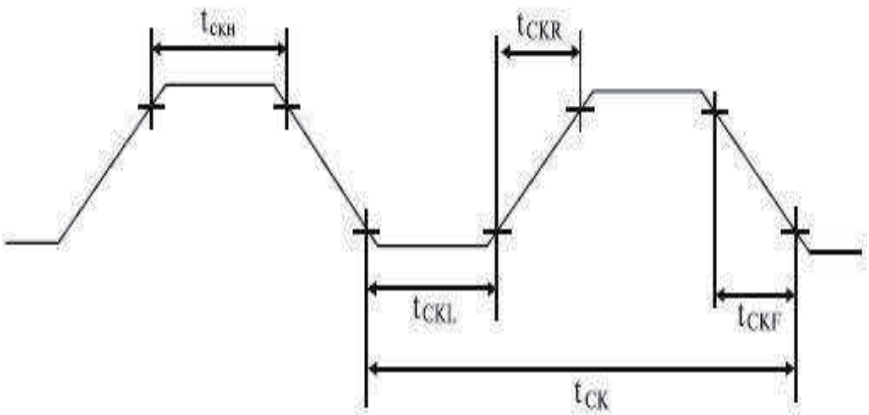


Рисунок 7.17. — Временная диаграмма синхронизации на входе XTAL1

Параметры временных интервалов цикла синхронизации на рис. 7.17 следующие:

1. Период синхронизации t_{CK}, нс

   мин. 62,5

   тип. 83,33

   макс. 1000,0

2. Длительность низкого уровня на входе XTAL1,
   t_{CKL}, нс, не менее 20,0
3. Длительность высокого уровня на входе XTAL1,
   t_{CKH}, нс, не менее 20,0

4. Длительность времени нарастания синхросигнала на входе XTAL1, t_{CKR}, нс, не более 20,0
5. Длительность времени убывания синхросигнала на входе XTAL1, t_{CKF}, нс, не более 20,0
6. Время машинного цикла ADuC812, T_{CK}, мкс 12x t

Примечание.

1. AC вводы в течении испытания управляются DV_{DD} - 0,5 В для логической «1» и 0,45 В для логического «0». Синхронизация измерений сделана в V_{IH} min для логической «1» и V_{IL} max для логического “0”.
2. Для синхронизации, порт ввода больше не работает вхолостую, когда происходит изменение напряжения на нагрузке на 100 мВ. Порт ввода начинает работать вхолостую, когда происходит изменение на 100 мВ от загруженных $V_{OH/OL}$ норм.
3. C_{LOAD} для порта 0, ALE, PSEN выводы = 100 нФ; C_{LOAD} для всех остальных выводов = 80 нФ , если иначе не отмечено.

Временные характеристики сигналов

Изменение уровней сигналов синхронизации и управления во времени показано на рисунке 7.18.

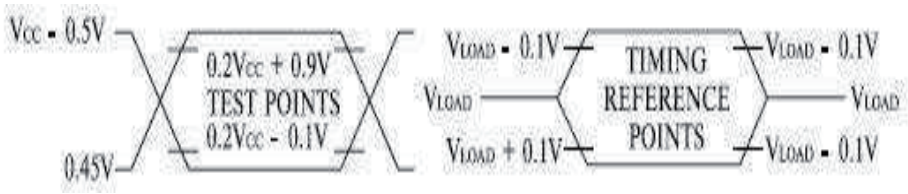


Рисунок 7.18. — Изменение сигналов во времени

Временные соотношения при выполнении цикла чтения из внешней памяти программ.

Временная диаграмма цикла чтения информации из внешней памяти программ приведена на рисунке 7.19.

Параметры временных интервалов на диаграмме рисунка 7.19 следующие:

1. Длительность высокого уровня импульс сигнала ALE, t_{LHLL}, нс, не менее

тип. 127

при переменной синхронизации $2T_{CU}$-40

2. Время совпадения высокого уровня импульса сигнала ALE и адреса на шине, t_{AVLL}, нс, не менее

тип. 43

3. Время сохранения адреса на шине после появления низкого уровня сигнала ALE, t_{LLAX}, нс, не менее

тип. 53

при переменной синхронизации TCU-30

4. Время от момента появления низкого уровня ALE до появления инструкции чтения, t_{LLIV}, нс, не более

тип. 234

при переменной синхронизации $4T_{CU}$-100

5. Время от момента появления низкого уровня ALE до появления низкого сигнала PSEN, t_{LLPL}, нс, не менее

тип. 53

при переменной синхронизации T_{CU}-30

6. Длительность низкого (активного) уровня импульса на выводе PSEN, t_{PLPH}, с, не менее

тип. 205

при переменной синхронизации $3T_{CU}$-45

7. Время от момента появления низкого уровня сигнала PSEN до появления инструкции чтения, t_{PLIV}, нс, не более

тип. 145

при переменной синхронизации $3T_{CU}$-105

8. Время сохранения инструкции ввода после момента окончания низкого уровня на выводе PSEN,t_{PXIX}, нс, не менее 0.
9. Время от момента окончания низкого уровня на выводе PSEN до момента окончания инструкции ввода, t_{PXIZ}, нс, не более

тип. 59

при переменной синхронизации T_{CU}-25

10. Время от начала появления адреса на шине до момента появления инструкции ввода, t_{AVIV}, нс, не более

тип. 312

при переменной синхронизации $5T_{CU}$-105

11. Время от начала низкого уровня на выводе PSEN до исчезновения адреса на шине, t_{PLAZ}, нс, не более 25
12. Время хранения старшего байта адреса на шине порта P2 после появления высокого уровня сигнала на выводе PSEN, t_{PHAZ}, нс, не менее 0

Примечание.

1. Типовые значения и здесь и далее приведены для частоты кварцевого резонатора 12 МГц.
2. T_{CU} - период при переменной синхронизации, 62,5 — 1000 нс.

Временные соотношения при выполнении цикла чтения из внешней памяти данных

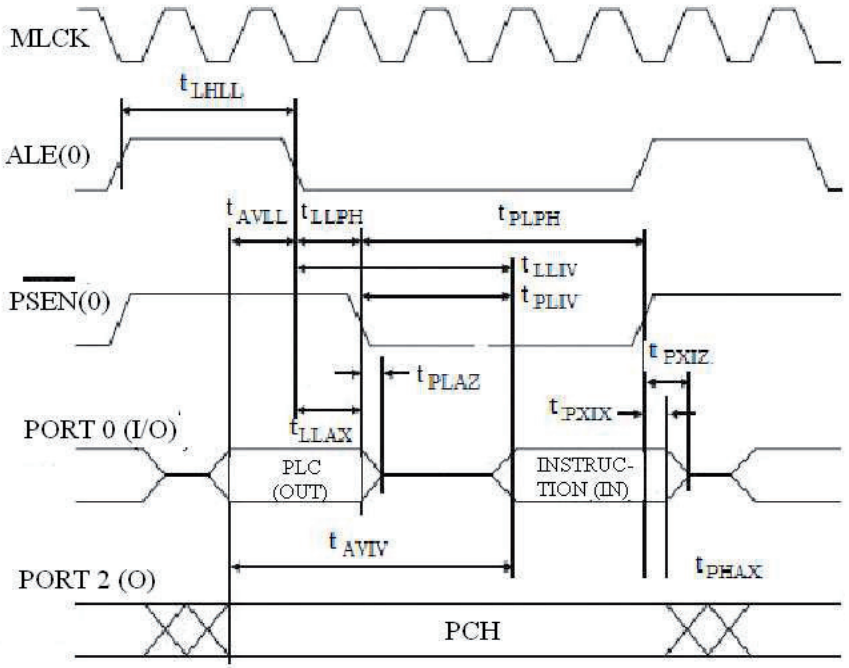


Рисунок 7.19. — Цикл чтения из внешней памяти программ

Временная диаграмма цикла чтения информации из внешней памяти данных приведена на рисунке 7.20. Параметры временных интервалов на диаграмме рисунка 7.20 следующие:

1. Длительность активной фазы импульса чтения RD, t_{PLPH}, нс, не менее

тип. 400

при переменной синхронизации $6T_{CU}$-100

2. Время совпадения высокого уровня сигнала ALE и адреса на шине, t_{AVLL}, нс, не менее

тип. 43

при переменной синхронизации T_{CU}-40

3. Время сохранения адреса после появления низкого уровня сигнала ALE, t_{LLAX}, нс, не менее

тип. 48

при переменной синхронизации T_{CU}-35

4. Время от момента появления низкого уровня RD до появления данных на , шине t_{PLIV}, нс, не более

тип. 252

при переменной синхронизации $5T_{CU}$-165

5. Время сохранения старшей части адреса и данных на шинах после окончания активной фазы сигнала RD, t_{PHAX}, нс, не менее 0.

6. Время от момента появления низкого уровня сигнала ALE до появления данных на шине, t_{LLIV}, нс, не более

тип. 517

при переменной синхронизации $8T_{CU}$-150

7. Время присутствия данных на шине после окончания активной фазы сигнала RD, t_{PX1Z}, нс, не более

тип. 97

при переменной синхронизации $2T_{CU}$-70

8. Время от момента появления адреса на шине до момента появления данных, t_{AVIV}, нс, не более

тип. 585

при переменной синхронизации $9T_{CU}$-165

9. Время от момента появления низкого уровня сигнала ALE до начала активной фазы, сигнала RD, t_{LL1V}, нс

тип. от 200 до 300

при переменной синхронизации от $3T_{CU}$-50 до $3T_{CU}$+50

10. Время от момента появления адреса на шине до начала активной фазы сигнала RD, t_{AV1V}, нс, не менее

тип. 203

при переменной синхронизации $4T_{CU}$-130

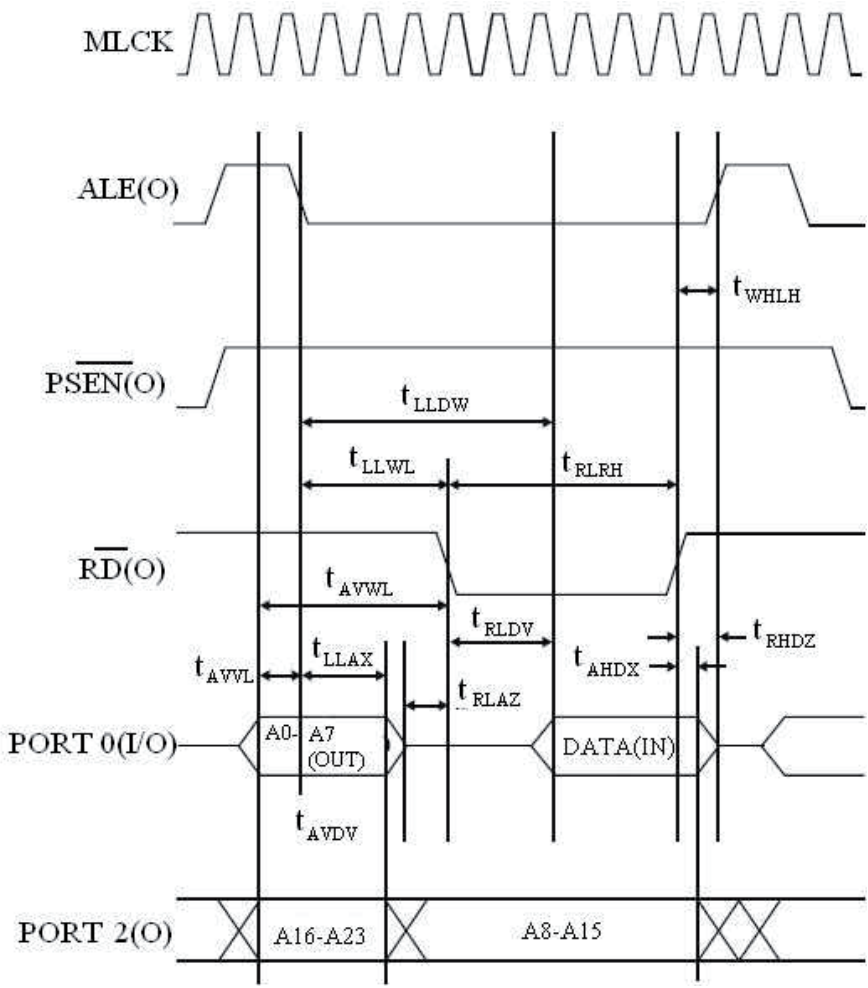


Рисунок 7.20. — Цикл чтения из внешней памяти данных

11. Время от момента исчезновения младшего байта адреса на шине до начала активной фазы сигнала RD, t_{RLAZ}, нс, не более 25.
12. Время от момента окончания активной фазы сигнала RD до момента появления высокого уровня ALE, t_{WHLH}, нс

тип. от 43 до 123

при переменной синхронизации от T_{CU}-40 до $6T_{CU}$+100

Временные соотношения при выполнении цикла записи во внешнюю память данных.

Временная диаграмма цикла записи информации во внешнюю память данных приведена на рисунке 7.21. Параметры временных интервалов на диаграмме рисунка 7.21 следующие:

1. Длительность активной фазы импульса записи WR, t_{WLWH}, нс, не менее

тип. 400

при переменной синхронизации $6T_{CU}$-100

2. Время совпадения высокого уровня сигнала ALE и адреса на шине, t_{AVLL}, нс, не менее

тип. 43

при переменной синхронизации T_{CU}-40

3. Время сохранения адреса после появления низкого уровня сигнала ALE, t_{LLAX}, нс, не менее

тип. 48

при переменной синхронизации T_{CU}-35

4. Время от момента появления низкого уровня сигнала ALE до начала активной фазы сигнала WR, t_{LLWL} нс

тип. от 200 до 300

при переменной синхронизации от $3T_{CU}$-50 до $3T_{CU}$+50

5. Время от момента появления адреса на шине до начала активной фазы сигнала WR, t_{AVWL}, нс

тип. 203

при переменной синхронизации $4T_{CU}$-130

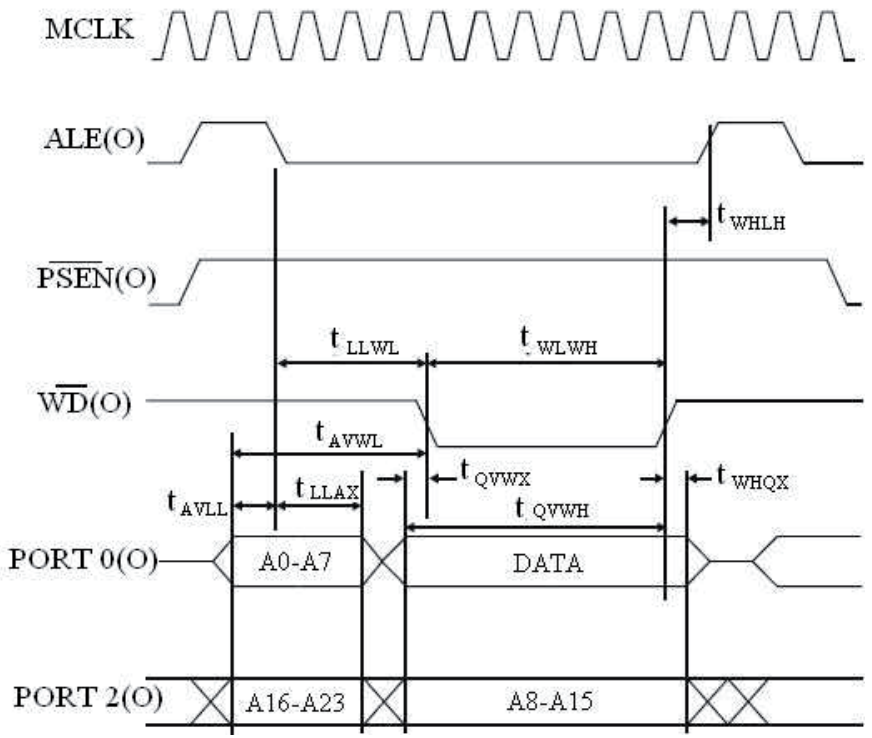


Рисунок 7.21. — Цикл записи во внешнюю память данных

6. Время опережения установки данных на шине перед началом активной фазы сигнала WR, t_{QVWX}, нс, не менее

тип. 33

при переменной синхронизации T_{CU}-50

7. Время присутствия данных на шине во время активной фазы сигнала WR, t_{QVWH}, нс, не менее

тип. 433

при переменной синхронизации $7T_{CU}$-150

8. Время сохранения данных и адреса на шинах после окончания активной фазы сигнала WR, t_{WHQX}, нс, не менее

тип. 33

при переменной синхронизации T_{CU}-50

9. Время от момента окончания активной фазы сигнала WR до момента появления высокого уровня ALE, t_{WHLH}, нс

тип. от 43 до 123

при переменной синхронизации от T_{CU}-40 до $6T_{CU}$+100

Временные соотношения интерфейса UART в режиме сдвигового регистра.

Временная диаграмма работы интерфейса UART в режиме сдвигового регистра приведена на рисунке 7.22.

Параметры временных интервалов на диаграмме рисунка 7.22 следующие:

1. Период синхронизации на передачу/прием бита, t_{XLXL},мкс

тип. 1,0

при переменной синхронизации $12T_{CU}$

2. Время совпадения бита передаваемой информации и активной фазы синхронизации, t_{QVXH}, нс, не менее

тип. 700

при переменной синхронизации $10T_{CU}$-133

3. Время совпадения бита принимаемой информации и активной фазы синхронизации, t_{DVXH}, нс, не менее

тип. 300

при переменной синхронизации $2T_{CU}$+133

4. Время сохранения бита принимаемой информации после окончания активной фазы синхронизации, t_{XHDX},нс, не менее 0.

5. Время сохранения бита передаваемой информации после окончания активной фазы синхронизации, нс, не менее

тип. 50

при переменной синхронизации $2T_{CU}$-117

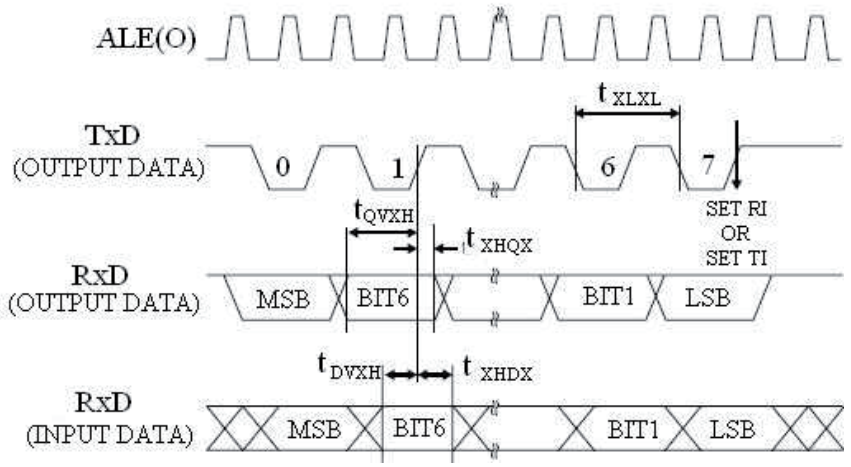


Рисунок 7.22. — Синхронизация UART в режиме сдвигового регистра

Временные соотношения I2C - совместимого интерфейса

Временная диаграмма работы последовательного интерфейса I2C приведена на рисунке 7.23.

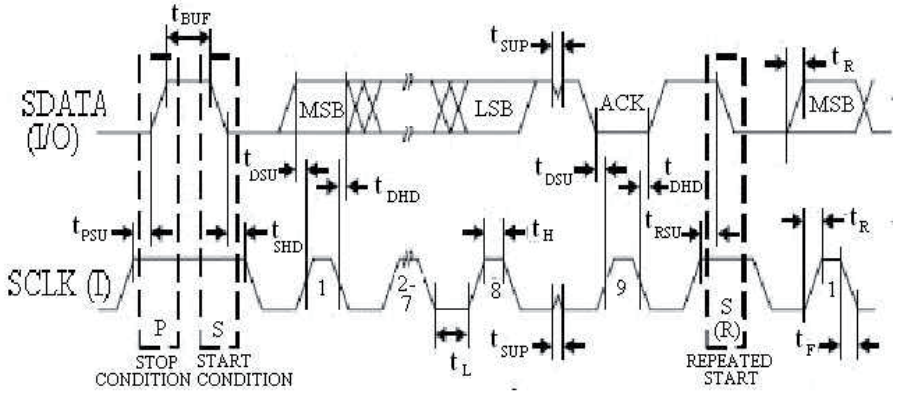


Рисунок 7.23. — Временная диаграмма I2C — совместимого интерфейса

Параметры временных интервалов на диаграмме рисунка 7.23 следующие:

1. Длительность низкого уровня синхронизирующего импульса SCLOCK, t_L, мкс, не менее 4,7.
2. Длительность высокого уровня синхронизирующего импульса SCLOCK t_H, мкс, не менее 4,0.
3. Задержка времени после наступления условия старта, t_{SHD}, мкс, не менее 0,6.
4. Время опережения появления в линии бита информации относительно активной фазы синхронизирующего импульса, t_{DSU}, нс, не менее 100
5. Время стхранения бита информации на линии после окнчания активной фазы синхронизирующего импульса, t_{DHD}, мкс, от 0 до 0,9
6. Задержка времени после наступления условия повторного старта, t_{RSU}, мкс, не менее 0,6
7. Задержка времени после наступления условия остановки, t_{PSU}, мкс, не менее 0,6
8. «Свободное время» между условием остановки и условием старта, t_{BUF}, мкс, не менее 1,3
9. Время нарастания в линии сигналов синхронизации SCLOCK и данных SDATA, t_R, нс, не более 300
10. Время убывания в линии сигналов синхронизации SCLOCK и данных SDATA, t_F, нс, не более 300
11. Длительность импульса подавленного шумового выброса, t_{SUP}, нс, более 50

Примечание:

1. Фильтрация входа на SCLOCK и входа SDATA подавляют шумовые выбросы, которые составляют менее 50 нс.

Временные соотношения интерфейса SPI в режиме Ведущий (CPHA=1)

Временная диаграмма работы последовательного интерфейса SPI в режиме Ведущий (CPHA=1) приведена на рисунке 7.24.

Параметры временных интервалов на диаграмме рисунка 7.24

следующие:

1. Длительность низкого уровня импульса SCLOCK, t_{SL}, нс – 330.
2. Длительность высокого уровня импульса SCLOCK, t_{SH},нс – 330.
3. Время допуска на появление выходных данных в линии после фронта импульса SCLOCK, t_{DAV},нс, не более 50.
4. Время опережения появления вводимых данных до начала фронта импульса SCLOCK, t_{DSU}, нс, не менее 100.
5. Время хранения вводимых данных на линии после фронта импульса SCLOCK, t_{DHD}, нс, не менее 100.
6. Время убывания выводимых данных на линии, t_{DF}, нс – 10-25.
7. Время нарастания выводимых данных на линии, t_{DR}, нс – 10-25.
8. Время нарастания импульса SCLOCK, t_{SR}, нс – 10-25.
9. Время убывания импульса SCLOCK, t_{SF}, нс – 10-25.

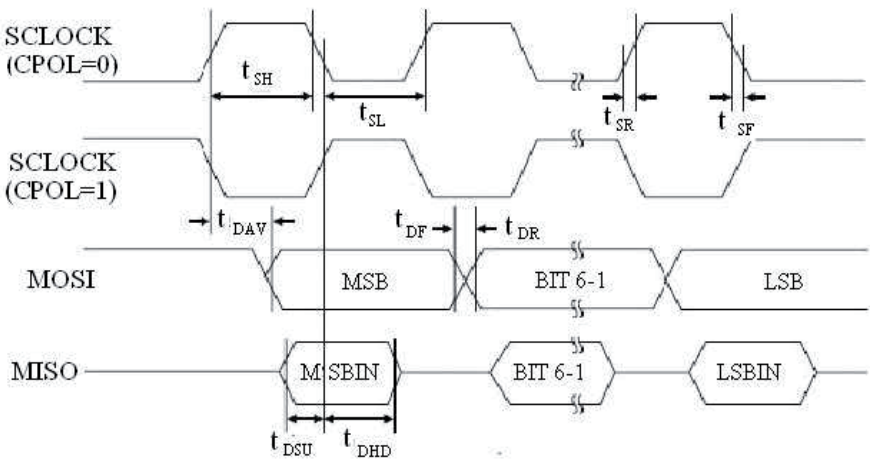


Рисунок 7.24. — Синхронизация интерфейса SPI в режиме Ведущий (CPHA=1)

Временные соотношения интерфейса SPI в режиме Ведущий (CPHA=0)

Временная диаграмма работы последовательного интерфейса SPI в режи-

ме Ведущий (CPHA=0) приведена на рисунке 7.25.

Параметры временных интервалов на диаграмме рисунка 7.25 следующие:

1. Длительность низкого уровня импульса SCLOCK, t_{SL}, нс – 330.
2. Длительность высокого уровня импульса SCLOCK, t_{SH},нс – 330.
3. Время допуска на появление выходных данных в линии после фронта импульса SCLOCK, t_{DAV},нс, не более 50.
4. Время опережения появления выводимых данных в линии до начала фронта импульса SCLOCK, t_{DOSU}, нс, не менее 150.
5. Время опережения появления вводимых данных до начала фронта импульса SCLOCK, t_{DSU}, нс, не менее 100.
6. Время хранения вводимых данных на линии после фронта импульса SCLOCK, t_{DHD}, нс, не менее 100.
7. Время убывания выводимых данных на линии, t_{DF}, нс – 10-25.
8. Время нарастания выводимых данных на линии, t_{DR}, нс – 10-25.
9. Время нарастания импульса SCLOCK, t_{SR}, нс – 10-25.
10. Время убывания импульса SCLOCK, t_{SF}, нс – 10-25.

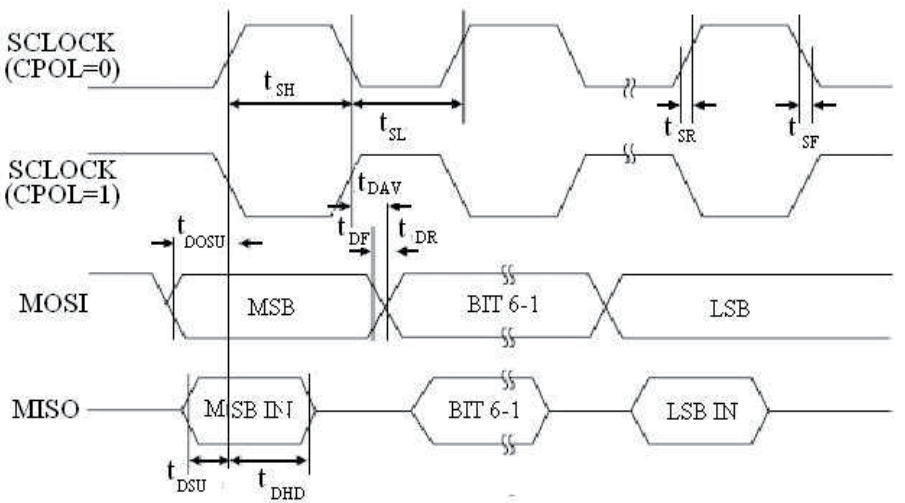


Рисунок 7.25. — Синхронизация интерфейса SPI в режиме Ведущий

(CPHA=0)

Временные соотношения интерфейса SPI в режиме Ведомый (CPHA=1)

Временная диаграмма работы последовательного интерфейса SPI в режиме Ведомый (CPHA=1) приведена на рисунке 7.26.

Параметры временных интервалов на диаграмме рисунка 7.26 следующие:

1. Время опережения низкого уровня сигнала SS до начала фронта импульса SCLOCK, t_{SS}, нс, не менее 0.
2. Длительность низкого уровня импульса SCLOCK, t_{SL},нс – 330.
3. Длительность высокого уровня импульса SCLOCK, t_{SH},нс – 330.
4. Время допуска на появление выходных данных в линии после фронта импульса SCLOCK, t_{DAV},нс, не более 50.
5. Время опережения появления вводимых данных до начала фронта импульса SCLOCK, t_{DSU}, нс, не менее 100.
6. Время хранения вводимых данных на линии после фронта импульса SCLOCK, t_{DHD}, нс, не менее 100.
7. Время убывания выводимых данных на линии, t_{DF}, нс – 10-25.
8. Время нарастания выводимых данных на линии, t_{DR}, нс – 10-25.
9. Время нарастания импульса SCLOCK, t_{SR}, нс – 10-25.
10. Время убывания импульса SCLOCK, t_{SF}, нс – 10-25.
11. Время установления выходного сигнала SS после окончания фронта импульса SCLOCK, t_{SFS}, нс, не менее 0.

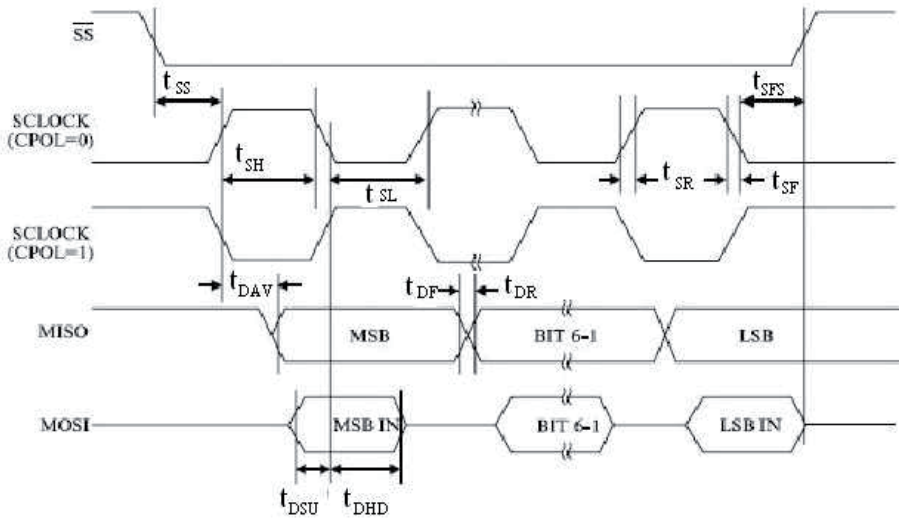


Рисунок 7.26. — Синхронизация интерфейса SPI в режиме Ведомый (CPHA=1)

Временные соотношения интерфейса SPI в режиме Ведомый (CPHA=0)

Временная диаграмма работы последовательного интерфейса SPI в режиме Ведомый (CPHA=0) приведена на рисунке 7.27.

Параметры временных интервалов на диаграмме рисунка 7.27 следующие:

1. Время опережения низкого уровня сигнала SS до начала фронта импульса SCLOCK, t_{SS}, нс, не менее 0.
2. Длительность низкого уровня импульса SCLOCK, t_{SL},нс – 330.
3. Длительность высокого уровня импульса SCLOCK, t_{SH},нс – 330.
4. Время допуска на появление выходных данных в линии после фронта импульса SCLOCK, t_{DAV},нс, не более 50.
5. Время опережения появления вводимых данных до начала фронта импульса SCLOCK, t_{DSU}, нс, не менее 100.
6. Время хранения вводимых данных на линии после фронта импульса

SCLOCK, t_{DHD}, нс, не менее 100.

7. Время убывания выводимых данных на линии, t_{DF}, нс – 10-25.
8. Время нарастания выводимых данных на линии, t_{DR}, нс – 10-25.
9. Время нарастания импульса SCLOCK, t_{SR}, нс – 10-25.
10. Время убывания импульса SCLOCK, t_{SF}, нс – 10-25.
11. Время допуска на появление віходніх данніх в линии после появления низкого уровня сигнала SS , t_{DOSS}, нс – 10-25.
12. Время установления выходного сигнала SS после окончания фронта импульса SCLOCK, t_{SFS}, нс, не более 25.

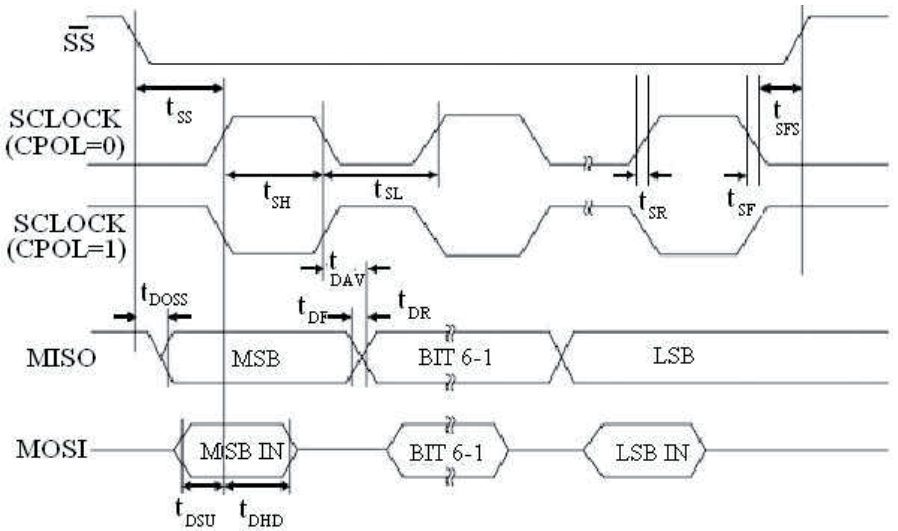


Рисунок 7.27. — Синхронизация интерфейса SPI в режиме Ведомый (CPHA=0)

ПРИЛОЖЕНИЯ

1. Системы счисления

В дискретной технике используются следующие системы счисления:

- двоичная (обозначается индексом b);
- восьмеричная (обозначается индексом q);
- шестнадцатеричная (обозначается индексом h).

8-ричная и 16-ричная системы обычно используются для сжатой (экономной) записи информации вместо двоичной системы счисления.

Соотношения между десятичной, двоичной, 8-ричной и 16-ричной системами указаны в таблице.

10	2	8	16
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Алгоритм прямого преобразования целого числа из 10-й системы счисления в двоичную (8-ричную, 16-ричную) систему счисления.

Преобразование достигается путем многократного повторного деления

исходного 10-го числа и целых частных от деления на величину системы основания. Остатки от деления образуют разряды преобразованного числа. Последнее целое частное от деления, меньшее величины основания, без изменений сносится в итоговый результат в качестве старшего разряда результата

Пример.

Преобразовать в 2-ю системы счисления 10–е число 76.

76 :2 остаток 0 мл. разряд результата

38 : 2 остаток 0

19 : 2 остаток 1

9 : 2 остаток 1

4 : 2 остаток 0

2 : 2 остаток 0

1 : 2 остаток 1 ст. разряд результата

Получен результат преобразования в виде 100 1100 b или в 8-разрядной форме — 0100 1100 b. Из 2-го представления с помощью таблицы можно получить представление в 16-ричной форме числа 76 : 4C h.

Алгоритм обратного преобразования информации из 2-ой (16-ричной) системы счисления в 10-ю.

Обратное преобразование представляет собой сумму произведений соответствующих разрядов исходного 2-го (16-ричного) представления числа на величину основания в степени номера разряда.

Пример.

Преобразовать 16-ричное число 7A h в 10-ю форму.

$7x16^1 + Ax16^0 = 7x16 + 10x1 = 122$

2. Дополнительный код

Алгоритм получения дополнительного кода:

1. Исходное двоичное число инвертируется , т.е. нули меняются на 1, а единицы меняются на нули.
2. К проинвертированному числу арифметически добавляется 1.

Пример.

Получить дополнительный код от 16-ричного числа 0AC h.

1. Преобразуем заданное число в 2-ю форму (по таблице приложения 1) — 1010 1100 b
2. Получаем инвертированное значение — 0101 0011 b
3. Арифметическое сложение с 1 0101 0011 b

   + 1

   0101 0100 b — результат,

   или в 16-ричной форме 54 h, или 84 в 10-й системе.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Вуд А. Микропроцессоры в вопросах и ответах. — М.; Энергоатомиздат, 1985. — 184с.
2. Богумирский Б.С. Руководство пользователя ПЭВМ: ч.1. — Санкт-Петербург: Печатный двор, 1984.
3. Справочник по микропроцессорным устройствам /А.А.Молчанов и др. — К.; Техніка, 1987 — 288 с.
4. Погорелый С.Д., Слободянюк Т.Ф. Програмное обеспечение микропроцессорных систем: Справочник. — К.: Техніка, 1985. — 240 с.
5. Микропроцессоры: В 3-х кн., под ред. Преснухина Л.Н. — М.: Высш. шк., 1986.
6. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифрових устройств на однокристальных микроконтроллерах. — М.: Энергоатомиздат, 1990. — 201 с.
7. Проектирование микропроцессорной электронно-вычислительной аппаратуры: (справочник). Артюхов В.Г. и др. — К.: Техніка, 1987. — 256 с.
8. Однокристальные микро ЭВМ. М.: МИКАП, 1994г — 400с.
9. Бродин В.Б., Шагунов И.И. Микроконтроллеры . М.; Эконом,- 1999г.-400с
10. www.analog.com
11. Электронные компоненты и системы, № за 2000-2002 г.