Министерство образования Российской Федерации
Государственное образовательное учреждение высшего профессионального образования
СЕВЕРО-ЗАПАДНЫЙ ГОСУДАРСТВЕННЫЙ ЗАОЧНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

О.А. ГОТШАЛЬК

ПРОМЫШЛЕННЫЕ КОНТРОЛЛЕРЫ

МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ ЭНЕРГЕТИЧЕСКИХ ОБЪЕКТОВ

Утверждено редакционно-издательским советом университета в качестве письменных лекций

Санкт–Петербург
2003

УДК 621. 375(03)
ББК 32.97

Готшальк О. А. Промышленные контроллеры. Микропроцессорные системы энергетических объектов: Письменные лекции.–СПб.: СЗТУ, 2003-64с.

Письменные лекции соответствуют требованиям государственных образовательных стандартов высшего профессионального образования подготовки дипломированных специалистов по направлениям 657900–"Автоматизированные технологии и производства" (специальность 210200–"Автоматизация технологических процессов и производств в машиностроении", специализация 210217–"Компьютерные системы управления в производстве и бизнесе"), 650900–"Электроэнергетика" (специальность 100400–"Электроснабжение"), а также подготовки бакалавров 552900–"Технология, оборудование и автоматизация" и 551700–"Электроэнергетика".

Для студентов специальности 210200 (и специализации 210217) дисциплина называется "Промышленные контроллеры" и читается на 4-м курсе, а для студентов специальности 100400–"Микропроцессорные системы энергетических объектов" и читается на 5-м курсе.

В письменных лекциях изложены основы архитектуры, программного и аппаратного обеспечения, даны технические характеристики современных промышленных контроллеров. Отдельный раздел посвящен описанию средств разработки микропроцессорных систем. Рассмотрены примеры программирования для автоматизации конкретных технологических объектов.

Данные письменные лекции не являются исчерпывающими. Они призваны дать лишь основы знаний по данной дисциплине. Более подробную информацию можно получить в справочной системе help и приведенной специальной литературе.

Рецензенты: кафедра автоматизации производственных процессов СЗТУ (заведующий кафедрой А.А. Сарвин, д-р техн. наук, проф.); Е.Ю. Белкова, инж. 1-й категории ЗАО "СПб-Гипрошахт".

ВВЕДЕНИЕ

Повышение надежности, быстродействия и функциональной насыщенности, а также уменьшение габаритов современной вычислительной техники и потребляемой ею электроэнергии привело к широкому ее использованию для управления сложными технологическими объектами с большим количеством датчиков физических величин и управляемых агрегатов [1].

Но системы управления, в случае построения их на базе персональных компьютеров, как наиболее универсальных средств вычислительной техники, имеют ряд существенных недостатков.

Во-первых, возникает необходимость оснащения компьютеров специальными дополнительными устройствами, позволяющими подключать к ним технологическое оборудование и соответствующим образом преобразовывать сигналы, как поступающие на компьютер, так и вырабатываемые компьютером для управления технологическими объектами.

Во-вторых, в большинстве случаев однажды настроенный компьютер на управление конкретным технологическим объектом длительное время не требует вмешательства оператора, а поэтому нет необходимости иметь такие компоненты, как дисплей и клавиатуру, приводы дисков и сами диски.

В-третьих, принцип управления из единого центра (от одного компьютера) подразумевает наличие общего алгоритма управления, что отрицательно может сказаться в процессе модернизации отдельных составляющих технологического оборудования при непрерывном технологическом процессе.

Для устранения вышеуказанных недостатков применения компьютеров в системах управления технологическим оборудованием было разработано специальное устройство–микроконтроллер. Первый микроконтроллер был разработан в 1976 г. [2] и представлял собой большую интегральную схему (БИС), выполненную на одном кристалле. Микроконтроллер включал как основные элементы системной платы компьютера, так и устройств ее сопряжения с технологическим оборудованием: микропроцессор, генератор тактовых импульсов, постоянное и оперативное запоминающие устройства, порты ввода/вывода информации, таймеры, аналого-цифровые и цифро-аналоговые преобразователи, каналы широтно-импульсной модуляции сигналов и т. д.

Микроконтроллеры нашли широкое применение для интеллектуального управления различными объектами на транспорте, в машиностроении, энергетике и других отраслях промышленности.

Микроконтроллер можно рассматривать как миникомпьютер, оснащенный периферийными устройствами, позволяющими сочленять его с технологическим оборудованием, но лишенный дисплея и клавиатуры. Микроконтроллер может производить сбор информации о состоянии технологического оборудования, обработку собранной информации по

алгоритму любой сложности и выработку команд управления. При этом ввиду своей дешевизны, малых габаритов и высокой надежности для управления одним технологическим объектом могут использоваться сразу несколько микроконтроллеров, каждый из которых предназначен для управления отдельным агрегатом данного технологического объекта. При использовании нескольких микроконтроллеров они объединяются в единую локальную вычислительную сеть (ЛВС) с центральным компьютером (сервером) во главе. В этом случае роль центрального компьютера сводится к контролю и координации работы микроконтроллеров, включенных в локальную вычислительную сеть. Кроме того, посредством центрального компьютера производится программирование микроконтроллеров в период наладочных работ или модернизации технологического оборудования.

Для управления технологическим оборудованием микроконтроллеры устанавливаются на платы с разъемами для подключения технологических объектов и локальной вычислительной сети. В некоторых случаях для увеличения функциональных возможностей микроконтроллеру придаются дополнительные интегральные схемы электронной памяти, которые также помещаются на этой же плате. В этом случае такую плату называют промышленным контроллером (ПК).

Современные промышленные контроллеры при малых габаритах плат (порядка 60х90 мм) имеют от 10 до 24 каналов аналого-цифровых преобразователей, 7-10 таймеров, от 1 до 32 устройств формирования сигналов с широтно-импульсной модуляцией, сетевой CAN-контроллер для подключения промышленного контроллера к локальной вычислительной сети, до двух последовательных каналов передачи информации, от 23 до 110 линий ввода/вывода цифровой информации. Время выполнения одной команды в микроконтроллере колеблется от 80 до 500 нс при тактовой частоте 20 МГц [1].

Стоимость одного промышленного контроллера в настоящее время колеблется в пределах от 2500 до 17500 рублей (без НДС).

Выпуском микроконтроллеров и промышленных контроллеров занимаются такие известные фирмы, как MOTOROLA, MICROCHIP, MITSUBISHI, HITACHI, SIEMENS, PHILIPS и др.

В данных письменных лекциях студенту предлагается познакомиться с общими принципами построения и типовыми техническими и программными решениями промышленных контроллеров модели М-167-1 фирмы SIEMENS на базе микроконтроллера 80С167.

1.СТРУКТУРНАЯ СХЕМА МИКРОКОНТРОЛЛЕРА

Микроконтроллер (МК) представляет собой большую интегральную схему (БИС), собранную на одном полупроводниковом кристалле и объединяющую основные функциональные блоки, позволяющие производить сбор и обработку информации о текущем состоянии технологического оборудования с последующей выработкой управляющих сигналов.

Структурная схема МК как составная часть ПК, представлена на рис. 1.1. В нее входят следующие устройства:

генератор тактовых импульсов (ГТИ), предназначенный для формирования многофазной последовательности импульсов, обеспечивающих тактирование работы МК и внешних магистралей;

центральный процессор (ЦП), обеспечивающий выполнение основных логических и арифметических операций над информацией, характеризующей текущее состояние технологического оборудования, с последующей выработкой управляющих сигналов на базе заранее разработанного алгоритма управления;

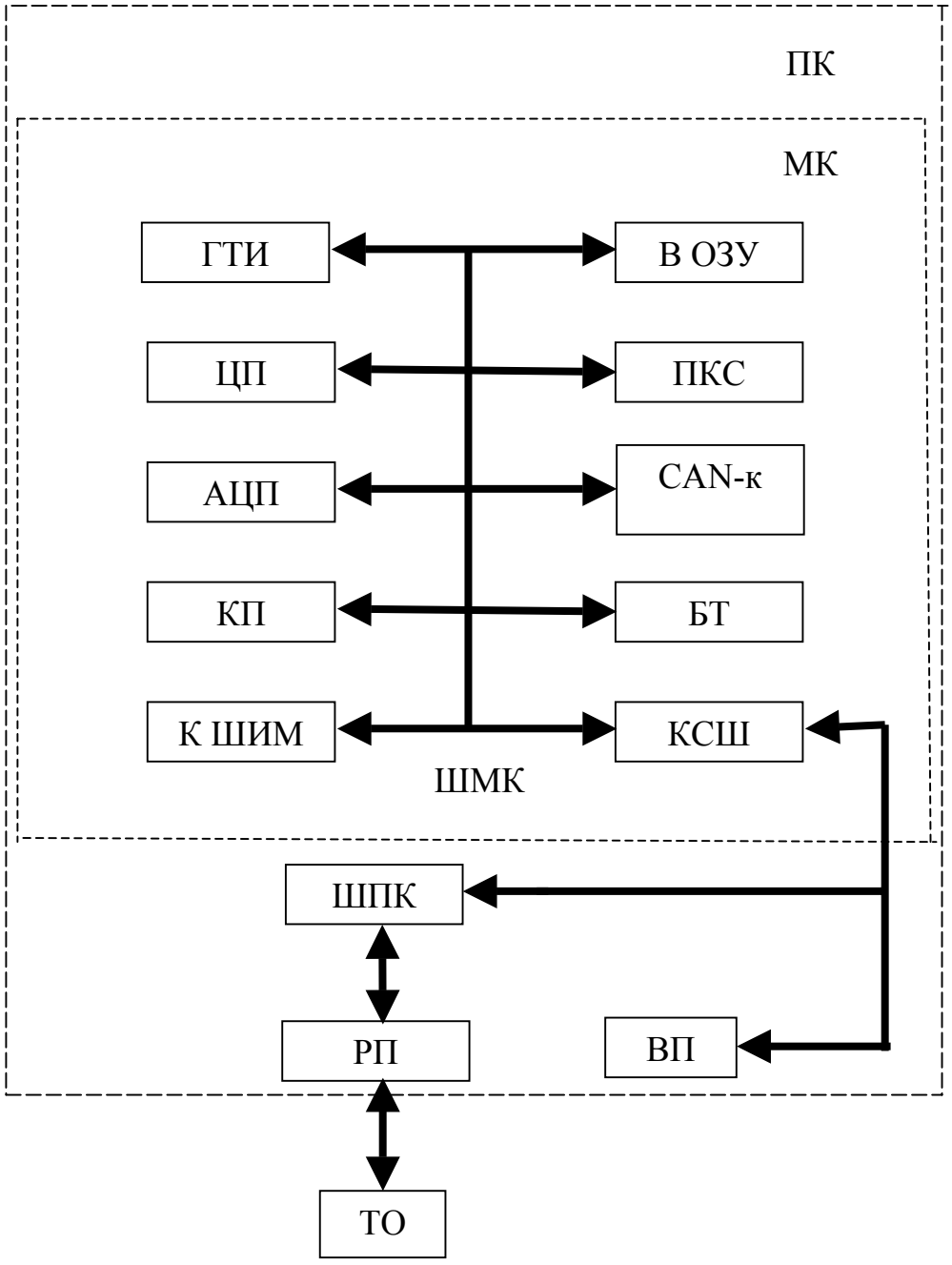


Рис. 1.1

аналого-цифровой преобразователь (АЦП), преобразующий аналоговые электрические сигналы, поступающие с соответствующих датчиков физических величин технологического оборудования, в цифровую форму;

контроллер прерываний (КП), служащий для реализации различных дисциплин обслуживания источников информации;

каналы широтно-импульсной модуляции (К ШИМ), позволяющие осуществлять плавное изменение напряжения или тока на нагрузке, за счет подачи на нее последовательности импульсов с различной частотой и длительностью. Выбирая частоту следования импульсов и их длительность в соответствии с параметрами нагрузки, возможно получать на нагрузке непрерывные значения напряжения или тока заданной величины или с заданным законом изменения их во времени;

внутреннее оперативное запоминающее устройство (В ОЗУ), обеспечивающее временное хранение промежуточных результатов обработки информации и отлаживаемых управляющих программ. В нем также располагаются некоторые устройства центрального процессора;

последовательный канал связи (ПКС), соединяющий МК с персональным компьютером в режиме ввода и отладки управляющей программы;

CAN-контроллер (CAN-к), обеспечивающий подключение ПК к локальной вычислительной сети (ЛВС);

блок таймеров (БТ), обеспечивающий работу МК в реальном масштабе времени;

контроллер системной шины (КСШ), обеспечивающий связь МК с внешними шинами и внешней памятью;

внутренняя шина МК (ШМК), включающая совмещенную системную шину адреса, данных и управления, обеспечивает информационную связь между всеми устройствами МК;

шина ПК (ШПК), дающая возможность обеспечить связь МК с технологическим оборудованием и внешней памятью;

разъемы пользователя (РП), представляющие собой устройства для подсоединения электрических линий связи ПК с технологическим оборудованием;

внешняя память (ВП), предназначенная для расширения функциональных возможностей МК;

технологическое оборудование (ТО).

Вопросы для самоконтроля

1. В чем состоит отличие микроконтроллера от промышленного контроллера?

2. Для каких целей применяется в ПК внешняя память?

2. ЦЕНТРАЛЬНЫЙ ПРОЦЕССОР

Центральный процессор (ЦП) обеспечивает считывание команд управляющей программы из памяти, их дешифрирование, пересылку информации между устройствами МК, выполнение логических и арифметических операций, сохранение промежуточных результатов в памяти или регистрах, выработку команд управления и пересылку их на соответствующие объекты технологического оборудования [2]. В рассматриваемом ПК типа М167-1 используется ЦП С167 фирмы SIEMENS. В приложении 1 приведены технические характеристики некоторых микроконтроллеров фирмы SIEMENS.

ЦП через 24-разрядную (24 бит) внутреннюю шину связан с внутренним ОЗУ.

Обращение ЦП к внешней памяти осуществляется посредством контроллера системной шины.

Обращение периферийных устройств к ЦП происходит путем посылки запросов на прерывание работы ЦП. Контроллер прерываний определяет приоритетный уровень одного из нескольких одновременно поступивших запросов и посылает его на ЦП.

ЦП содержит арифметико-логическое устройство (АЛУ) для реализации логических и арифметических операций.

Для обслуживания ЦП и АЛУ в ЦП имеется два вида регистров.

Регистры специального назначения (РСН)

1. Регистр системной конфигурации (SYSCON), определяющий состав ЦП и контроллера системной шины.

2. Регистры-указатели команд (IP), сегмента кода (CSP) и страниц данных (DPPx).

3. Регистры-указатели вершины (SP), переполнения (STKOM) и дна стека (STKUN).

4. Регистры выполнения операций умножения и деления (MDH, MDL).

Регистр MDH представляет собой старшие 16 бит 32-битового регистра, предназначенного для выполнения операций умножения и деления. После выполнения операции умножения в регистре MDH хранятся 16 старших разрядов произведения. Перед операцией деления 32-битового числа (делимого) на 16-битовое число (делитель) в регистр MDH помещается старшая 16-битовая часть делимого. После выполнения операции деления в этом регистре хранится 16-битовое значение остатка от операции деления.

Регистр MDL представляет собой младшую часть 32-битового регистра, предназначенного для выполнения операций умножения и деления. После выполнения операции умножения в регистре MDL хранятся 16 младших разрядов результата операции умножения. Перед операцией деления 32-битового числа на 16-битовое в регистр MDL помещается младшая 16-битовая часть делимого. После выполнения операции деления в этом регистре хранится 16-битовое значение частного.

5. Регистр отображения текущего состояния (PSW) ЦП и АЛУ. Каждый бит этого регистра может принимать значения 0 или 1 в зависимости от результата выполнения штатных операций в ЦП или АЛУ. В табл. 2.1 указаны некоторые значения битов этого регистра при выполнении штатных операций.

Т а б л и ц а 2.1

Бит	Наименование	Функция
0	N	Результат отрицательный
1	C	Результат с переносом бита
2	V	Переполнение
3	Z	Результат равен нулю
4	E	Конец таблицы
5	IEN	Прерывание операций деления и умножения

Регистры общего назначения (РОН)

Эти регистры (GPR–General Purposes Register) предназначены для временного хранения промежуточных результатов.

Регистр – это электронное устройство для временного хранения операндов (команд, чисел) разрядностью 8 или 16 бит (1 или 2 байта). Отличие регистра от ячейки памяти состоит в том, что регистры имеют специализацию (хранение определенного вида информации), в регистрах имеется возможность изменять информацию в пределах одного бита и некоторые разряды регистра (биты) могут быть электрически соединены с другими схемами ЦП для постоянной пересылки информации.

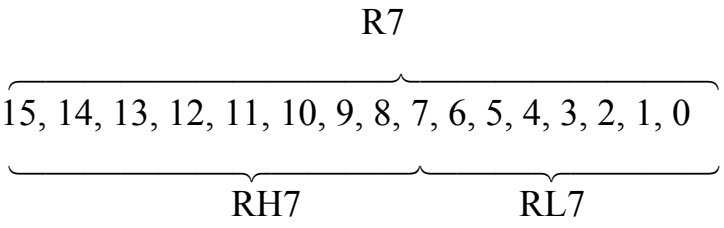
В первых ЦП для хранения промежуточных результатов использовались именно регистры. В дальнейшем регистры заменили ячейками памяти ОЗУ, а те ячейки памяти, которые стали исполнять роль регистров, по традиции продолжают называть регистрами, тем более что эти ячейки памяти трансформированы и имеют все признаки регистров.

Таким образом, часть ячеек памяти ОЗУ МК отводится под регистры, и чтобы обратиться к ним, необходимо указать адрес той или иной ячейки памяти. В программе чаще указывается не адрес ячейки памяти, исполняющей роль регистра, а установленное название регистра, в то время как транслятор программы название переводит в адрес.

В МК М167-1 регистры общего назначения размещены в нулевом сегменте внутреннего ОЗУ. Там размещены 16 регистров общего назначения объемом 2 байта каждый: R0...R15.

Каждый из этих регистров можно использовать и как двухбайтовый, и как два регистра объемом один байт каждый. Каждый однобайтовый регистр имеет свой символ. В этом случае к символам двухбайтовых регистров добавляются латинские буквы H (high) – старший или L (low) – младший: RL0, RH0, RL1, RH1 и т. д.

Например, двухбайтовый регистр общего назначения R7 можно представить как два однобайтовых регистра RH7 и RL7:



Вопросы для самоконтроля

1. В чем отличие регистров специального назначения от регистров общего назначения ?

2. Какую функцию выполняет регистр состояния при реализации управляющей программы?

3. ОРГАНИЗАЦИЯ ПАМЯТИ МИКРОКОНТРОЛЛЕРА

Максимальный объем адресуемой памяти МК при разрядности внутренней шины в 24 бит составляет 16 Мбайт.

Вся память МК разделена на 256 сегментов по 64 Кбайт в каждом сегменте. Каждый сегмент содержит 4 страницы объемом 16 Кбайт каждая.

Вся память МК делится на внутреннюю и внешнюю.

Внутренняя память размещена в нулевом сегменте. Она содержит:

1) внутреннее оперативное запоминающее устройство (В ОЗУ);

2) регистры специального назначения (РСН);

3) регистры общего назначения (РОН).

Так как объем любой ячейки памяти составляет 1 байт (8 бит), а разрядность данных и команд может составлять 2 байта, то в памяти МК данные и команды занимают по две ячейки памяти. Младшие байты данных или команд размещаются по четным адресам ячеек памяти, а старшие байты данных или команд размещаются по нечетным адресам ячеек памяти.

Область регистров общего назначения и регистров специального назначения, а также часть области внутреннего оперативного запоминающего устройства являются бит-адресуемыми.

Обращение к той или иной ячейке памяти может производиться двумя способами:

1) путем указания номера сегмента и адреса ячейки памяти внутри указанного сегмента (смещение внутри сегмента);

2) путем указания номера страницы в сегменте или во всем адресном пространстве памяти и адреса ячейки памяти внутри указанной страницы (смещение внутри страницы).

В первом случае в регистр-указатель команды (IP) заносится адрес ячейки памяти внутри выбранного сегмента, а номер сегмента помещается в регистр-указатель сегмента кода (CSP). Во время выполнения управляющей программы

содержимое регистра IP изменяется автоматически после очередной выборки команды из ячейки памяти.

Если в формировании адреса участвует регистр-указатель сегмента кода (в регистре конфигурации SYSCON разрешена сегментация памяти), то полный адрес ячейки памяти, к которой обращается ЦП, складывается из содержимого регистров-указателей команды (IP) и сегмента кода (CSP). Если же в регистре конфигурации занесено запрещение сегментации, то адрес ячейки памяти определяется только содержимым регистра-указателя команды (IP) в нулевом сегменте.

Вопросы для самоконтроля

1. Информацию какой разрядности возможно записать в одну ячейку памяти?

2. Какая разрядность одной команды управляющей программы и как она заносится в память МК?

4. ЯЗЫК АССЕМБЛЕР

Внутри МК информация циркулирует и обрабатывается в двоичной системе счисления. Составить программу управления в символах двоичной системы счисления (в машинных кодах) чрезвычайно сложно. Составление же программы управления на языке высокого уровня приводит к резкому увеличению объема памяти и к значительному понижению быстродействия МК. Это связано с необходимостью применения сложного транслятора и с избыточностью транслятора и языка высокого уровня.

Для облегчения работы программиста, уменьшения объема транслятора и повышения быстродействия МК применяют упрощенный язык программирования Ассемблер.

Ассемблер – это упрощенный язык программирования, в котором группам команд с одним целевым назначением присваивается единый символ-мнемокод, а числа задаются в шестнадцатеричной системе счисления (ШСС).

Чтобы отличать числа, выраженные в различных системах счисления, после числа ставятся условные символы (в скобках или без них): ДСС–d; ДвСС–e; ШСС–h.

ШСС имеет 16 символов. Числа от 0 до 9, как и в десятичной системе счисления (ДСС), обозначаются теми же символами, а числа от 10 до 15 обозначаются символами латинского алфавита: A, B, C, D, E, F.

ДСС: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15.

ШСС: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Для перевода числа из ДСС в ШСС могут быть использованы два метода.

При использовании первого метода десятичное число сначала переводится из ДСС в двоичную систему счисления (ДвСС), а затем из ДвСС в ШСС. Для перевода числа из ДвСС в ШСС двоичное число разбивается на

группы по 4 бита в каждой группе и каждая группа выражается символом ШСС как самостоятельное двоичное число.

Пример 4.1

Перевести число 143d в ШСС.

Перевод числа из ДСС в ДвСС.

128	64	32	16		8	4	2	1
1	0	0	0		1	1	1	1

Двоичное число разбивается на группы по 4 бита, и каждая группа выражается символом ШСС. При этом каждая группа рассматривается как отдельное двоичное число.

8421	8421	
1000	1111	$\Rightarrow$8Fh.

При применении второго метода используются весовые коэффициенты ШСС. Весовыми коэффициентами (K) в ШСС являются числа

…4096; 256; 16; 1.

Для того чтобы перевести число из ДСС в ШСС, необходимо определить, какое количество и каких весовых коэффициентов ШСС находится в данном двоичном числе (A). Для этого данное число делят на максимальный весовой коэффициент, исходя из условия $K \le A$. Целая часть частного, переведенная в символ ШСС, будет характеризовать старшую цифру шестнадцатеричного числа. После этого целую часть частного умножают на данный коэффициент, а полученное произведение (B) вычитают из заданного числа (A). С полученной разностью вновь производят все вышеуказанные операции, но уже с более низкими весовыми коэффициентами.

Пример 4.2

Перевести число 58900(d) в ШСС.

58900:4096=14,3798.
14·4096=57344.
58900-57344=1556.

1556:256=6,0781.
6·256=1536.
1556-1536=20.

20:16=1,25.
1·16=16.
20-16=4.

4:1=4.

Таким образом, в заданном числе 58900(d) содержится шестнадцатеричных разрядов с весовыми коэффициентами:

4096 – 14 (в ШСС E);
256 – 6 (в ШСС 6);
16 – 1 (в ШСС 1);
1 – 4 (в ШСС 4).

Окончательно заданное число в ШСС можно записать в виде E614(h).

Перевод чисел из ШСС в ДСС производится в обратном порядке. Каждая цифра числа, выраженного в ШСС, умножается на свой весовой коэффициент, и полученные произведения складываются.

Пример 4.3

Перевести число 1A4F(h) в ДСС. Для наглядности представим заданное число в виде

4096	256	16	1
1	A	4	F

Тогда

4096·1=4096.
256·10=2560.
16·4= 64.
1·15= 15.

6735(d).

Команды в Ассемблере состоят из следующих частей:

<метка>: <мнемокод><операнды>; <комментарий>

Метка обеспечивает ссылку на команду из других мест программы при выполнении операций условных и безусловных переходов. Метка помещается перед командой и отделяется от нее двоеточием и пробелом.

Мнемокод – это указание в символической форме, какую операцию должен выполнить ЦП. Для восприятия команды транслятор переводит символ мнемокода в команду, выраженную в ДвСС.

Мнемокод без дополнительного символа **B** оперирует информацией объемом 2 байта (условное обозначение двухбайтовой информации w-word).

Если после мнемокода стоит символ **B**, то мнемокод оперирует информацией объемом 1 байт (условное обозначение однобайтовой информации b-byte).

Если символ **B** стоит перед мнемокодом, то мнемокод оперирует информацией объемом 1 бит (условное обозначение однобитовой информации bit).

Операнды – это числа или (и) символы переменных, которыми оперирует ЦП. В команде могут быть указаны один или два операнда. В

последнем случае они разделяются запятой. Между операндами пробелы не допускаются. В общем случае в команде условно можно указать операнды как

<op1,op2>.

Чтобы произвести обработку информации, операнды заранее должны быть размещены в одном из следующих устройств МК: в регистрах общего назначения объемом 2 байта (R_w-word) или объемом 1 байт (R_b-byte); в регистрах данных порта (PX), подаваться в ЦП непосредственно из программы. В этом случае условное обозначение этой информации имеет вид #data x, где x характеризует разрядность вводимой информации (количество бит).

При выполнении различных операций над операндами (логических или арифметических) результаты операций всегда помещаются в первом (левом) операнде команды, стирая находящуюся там до операции информацию.

Комментарии – это пояснение команды в любой форме (в виде символов или текста). Комментарий помещается после команды и отделяется от нее точкой с запятой. Комментарии транслятором не воспринимаются и в формировании команд не участвуют.

Каждая строка управляющей программы (команда) заканчивается точкой с запятой.

Пример 4. 4

Переслать содержимое РОН RL1 в РОН RH3.

Программа

MOVB RH3, RL1; RH3←RL1

В приведенной команде используются следующие символы:

MOV – мнемокод пересылки информации;

B – символ, указывающий на разрядность пересылаемой информации в 1байт;

RH3 – старший байт РОН R3;

RL1 – младший байт РОН R1;

RH3←RL1 – комментарий к команде (пересылка содержимого РОН RL1 в РОН RH3); этот же комментарий можно записать и как RH3:=RL1, т. е. присвоить РОН RH3 значение РОН RL1.

Вопросы для самоконтроля

1. В чем отличие языков высокого уровня от Ассемблера?

2. Для каких целей служит мнемокод?

3. С какой целью каждый регистр общего назначения возможно разделить на два регистра?

5. СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРА

Система команд МК 80С167 включает следующие номинации [1].

5.1. Пересылка информации

5.1.1. Пересылка информации объемом 2 байта (word)

1. Между различными РОН

MOV R_w, R_w;

2. Из программы в РОН

MOV R_w, #date 16;

3. Между РОН и портами

MOV R_w, PX;

MOV PX, R_w.;

При этой операции порты должны иметь разрядность 2 байта.

Пример 5.1.1

Переслать содержимое РОН R2 в РОН R0.

Программа

MOV R0, R2; R0←R2

Записать в РОН R0 число 015Fh.

Программа

MOV R0, #015Fh; R0←015Fh

Переслать содержимое порта P0 в РОН R5.

Программа

MOV R5, P0; R5←P0

5.1.2. Пересылка информации объемом 1 байт (byte)

1. Между различными РОН

MOVB R_b, R_b.;

2. Из программы в РОН

MOVB R_b, #date 8;

3. Между РОН и портами

MOVB R_b, PX;

MOVB PX, R_b.;

При этой операции объем порта должен быть 1 байт.

Пример 5.1.2

Переслать содержимое РОН RL2 в РОН RH0.

Программа

MOVB RH0, RL2; RH0←RL2

Записать в РОН RL0 число 5Fh.

Программа

MOVB RL0, #5Fh; RL0←5Fh

Переслать содержимое порта P8 в РОН RH5.

Программа

MOVB RH5, P8; RH5←P8

5.1.3. Пересылка информации объемом 1 бит (bit)

При выполнении этой операции пересылается значение одного двоичного разряда (bit) между устройствами любой разрядности с указанием устройств, откуда и куда пересылается информация, и номера пересылаемого бита

BMOV bitaddr Z.z, bitaddr Q.q;

где bitaddr Z.z, и bitaddr Q.q − условное обозначение устройств, куда и откуда пересылается информация; Z и Q − наименование устройств, а z и q − номера пересылаемых битов.

Если после мнемокода команды стоит символ N, то пересылаемая информация (бит) подвергается инверсии

BMOVN bitaddr Z.z, bitaddr Q.q;

5.1.4. Изменение информации объемом 1 бит

1. Установка нуля в кодовой комбинации

BCLR bitaddr Q.q;

2. Установка единицы в кодовой комбинации

BSET bitaddr Q.q;

В приведенных командах bitaddr Z.z и bitaddr Q.q − условные обозначения устройств, в которых производится изменение информации; Z и Q −наименование устройств, а z и q − номера изменяемых битов.

5.1.5. Пересылка информации через стек

1. Запись информации на стек

PUSH RX;

2.Считывание информации со стека

POP reg;

где reg − в общем случае специальный регистр или регистр общего назначения.

Пример 5.1.3

1. Переслать 7-й бит РОН R1 в 5-й бит РОН R0.

Программа

BMOV R0.5, R1.7;

Если, предположим, до пересылки в РОН R0 была информация

R0–0000 0000 00**0**0 0101,

а в РОН R1

R1–0000 0000 **1**001 1111,

то после операции пересылки значение информации в РОН R0 изменится и примет вид

R0–0000 0000 00**1**0 0101.

Переслать 7-й бит РОН R1 в 5-й бит РОН R0 с инверсией.

Программа

BMOVN R0.5, R1.7;

Если, предположим, до пересылки в РОН R0 была информация

R0–0000 0000 00**0**0 0101,

а в РОН R1

R1–0000 0000 **0**001 1111,

то после операции пересылки значение информации в РОН R0 изменится и примет вид

R0–0000 0000 00**1**0 0101.

Установить в 4-м бите РОН RH1 логическую единицу.

Программа

BSET RH1.4;

Если, предположим, до команды установки в РОН RH1 была информация

RH1–000**0** 0101,

то после команды установки значение информации в РОН RH1 изменится и примет вид

R0–000**1** 0101.

Установить в 5-м бите порта P0 логический ноль.

Программа

BCLR P0.5;

Если, предположим, до команды установки в P0 была информация

P0–0000 0000 00**1**0 0101,

то после команды установки значение информации в P0 изменится и примет вид

R0–0000 0000 00**0**0 0101.

5.1.6. Пересылка информации между РОН и ОЗУ

1. Пересылка информации из ОЗУ в РОН

MOV R_w,[R_w];

где [R_w] − условное обозначение ячейки памяти ОЗУ, адрес которой заранее занесен в РОН объемом два байта.

2. Пересылка информации из РОН в ОЗУ

MOV [R_w],R_w;

3. Пересылка информации из одной ячейки ОЗУ в другую ячейку ОЗУ

MOV [R_w],[R_w];

4. Пересылка информации из одной ячейки ОЗУ в другую ячейку ОЗУ с увеличением или уменьшением адреса на 2, участвующих в пересылке ячеек памяти:

MOV [R_w],[R_w+];
MOV [R_w+],[R_w];
MOV [−R_w],[R_w];

5.2. Арифметические сложение и вычитание

5.2.1. Арифметическое сложение

1. Сложение информации двухбайтовых регистров

ADD R_w, R_w;

2. Сложение информации двухбайтового регистра и числа из программы

ADD R_w, #data 16;

3.Сложение информации однобайтовых регистров

ADDB R_b, R_b;

4. Сложение информации однобайтового регистра и числа из программы

ADDB R_b, #data 8;

Пример 5.2.1

Сложить содержимое РОН R0 и РОН R3.

Программа

ADD R0, R3; R0: =R0+R3

Сложить содержимое R0 с числом 015Fh.

Программа

ADD R0, #015Fh; R0:=R0+015Fh

Сложить содержимое РОН RH0 и РОН RL3.

Программа.

ADDB RH0, RL3; RH0:=RH0+RL3

Сложить содержимое РОН RH0 с числом 5Ch.

Программа

ADDB RH0, #5Ch; RH0:=RH0+5Ch

5.2.2. Арифметическое вычитание

1. Операция вычитания в двухбайтовых регистрах

SUB R_w, R_w;

2. Операция вычитания заданного числа из двухбайтового регистра

SUB R_w, #data 16;

3. Операция вычитания в однобайтовых регистрах

SUBB R_b, R_b;

4. Операция вычитания заданного числа из однобайтового регистра

SUBB R_b, #data 8;

Пример 5.2.2

Вычесть содержимое РОН R1 из содержимого РОН R0.

Программа

SUB R0, R1; R0:=R0−R1;

Вычесть из содержимого R0 число 015Fh.

Программа

SUB R0, #015Fh; R0:=R0−015Fh

Вычесть из содержимого РОН RH0 содержимое РОН RL3.

Программа

SUBB RH0, RL3; RH0:=RH0−RL3

Вычесть из содержимого РОН RH0 число 5Ch

Программа

SUBB RH0, #5Ch; RH0:=RH0−5Ch

5.3. Логические сложение и умножение

5.3.1. Логическое сложение

1. Логическое сложение двухбайтовой информации

OR R_w, R_w;

2. Логическое сложение однобайтовой информации

ORB R_b, R_b;

3. Логическое сложение однобитовой информации

BOR bitaddr Z.z, bitaddr Q.q;

Пример 15.3.1

Произвести логическое сложение содержимого РОН R0 и РОН R2.

Программа

OR R0, R2;

Если, предположим, до сложения в РОН R0 была информация

R0–0000 0000 0000 0101,

а в РОН R2

R2–0000 0000 0001 0110,

то после операции сложения значение информации в РОН R0 изменится и примет вид

R0–0000 0000 0000 0101
R2–0000 0000 0001 0110

R0–0000 0000 0001 0111.

Произвести логическое сложение содержимого РОН RH0 и РОН RL2.

Программа

ORB RH0, RL2;

Если, предположим, до сложения в РОН RH0 была информация

RH0–0000 0101,

а в РОН RL2

RL2–0001 0110,

то после операции сложения значение информации в РОН RH0 изменится и примет вид

RH0–0000 0101
RL2–0001 0110

RH0–0001 0111.

Произвести логическое сложение 2-го бита РОН RH0 и 0-го бита РОН RL3.

Программа

BOR RH0.2, RL3.0;

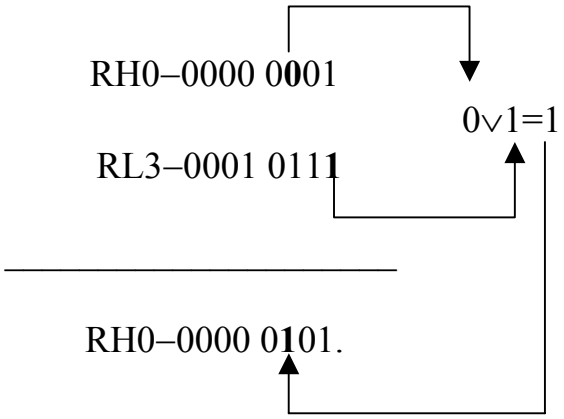
Если, предположим, до сложения в РОН RH0 была информация

RH0–0000 0**0**01,

а в РОН RL3

RL3–0001 011**1**,

то после операции сложения значение информации в РОН RH0 изменится и примет вид



5.3.2. Логическое умножение

1. Логическое умножение двухбайтовой информации
   AND R_w, R_w;
2. Логическое умножение однобайтовой информации
   ANDB R_b, R_b;
3. Логическое умножение однобитовой информации
   BAND bitaddr Z.z, bitaddr Q.q;

Пример 5.3.2

Произвести операцию логического умножения чисел, находящихся в РОН R0 и РОН R2.

Программа

AND R0, R2;

Если, предположим, до сложения в РОН R0 была информация

R0–0000 0000 0000 0101,

а в РОН R2

R2–0000 0000 0001 0110,

то после операции умножения значение информации в РОН R0 изменится и примет вид

R0–0000 0000 0000 0101
R2–0000 0000 0001 0110

R0–0000 0000 0000 0100.

Произвести операцию логического умножения чисел, находящихся в РОН RH0 и РОН RL2.

Программа

ANDB RH0, RL2;

Если, предположим, до умножения в РОН RH0 была информация

RH0–0001 0101,

а в РОН RL2

RL2–0001 0110,

то после операции умножения значение информации в РОН RH0 изменится и примет вид

RH0–0001 0101
RL2–0001 0110

RH0–0001 0100.

Произвести операцию логического умножения 2-го бита РОН RH0 и 0-го бита РОН RL3.

Программа

```
BAND    RH0.2, RL3.0;
```

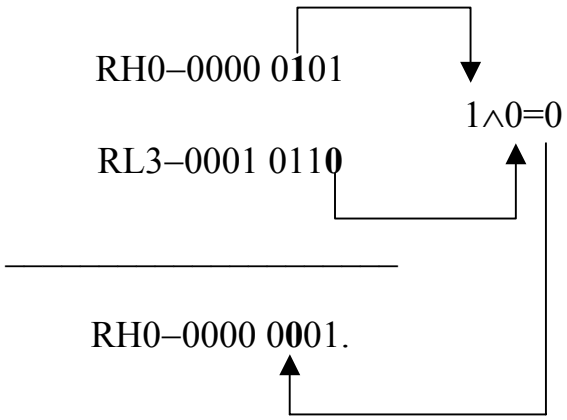
Если, предположим, до умножения в РОН RH0 была информация

RH0–0000 0**1**01,

а в РОН RL3

RL3–0001 011**0**,

то после операции умножения значение информации в РОН RH0 изменится и примет вид



5.4. Арифметические умножение и деление

5.4.1. Арифметическое умножение

```
MUL     Rw, Rw;
```

Арифметическое умножение производится над числами, заранее занесенными обязательно в двухбайтовые РОН. После выполнения операции умножения произведение будет находиться в регистрах MDH и MDL.

Пример 5.4.1

Перемножить числа, находящиеся в РОН R0 и РОН R1.

Программа

```
MUL     R0, R1; MDL:=R0×R1
```

5.4.2. Арифметическое деление

```
DIV     Rw;
```

При выполнении операции деления делитель помещается в любой РОН, а делимое заранее заносится в регистры MDL и MDH (при размерности делимого в 32 бит) или в MDL, а регистр MDH обнуляется (при размерности делимого в

16 бит). Регистры MDL и MDH в команде не указываются. Результат операции деления помещается в MDL, а остаток в MDH.

Пример 5.4.2

Разделить число FF8Dh на число 005Ch.

Программа

```
MOV     MDL, #FF8Dh; MDL←FF8Dh
MOV     R1, #005Ch; R1←005Ch
DIV     R1; MDL:=R0:R1
```

5.5. Сдвиг информации

5.5.1. Логический сдвиг информации влево

SHL R_w, #data x;

где символы #data x характеризуют количество бит, на которое сдвигается информация.

Пример 5.5.1

Сдвинуть число, находящееся в R0, на 2 бита влево.

Программа

```
SHL     R0,#2; сдвиг влево на 2 бита
```

Если, предположим, до выполнения операции сдвига в РОН R0 информация имела вид

R0–0000 0000 0110 0000,

то после выполнения операции сдвига влево содержимое РОН R0 примет вид

R0–0000 0001 1000 0000.

5.5.2. Логический сдвиг информации вправо

SHR R_w,#data x;

Пример 5.5.2

Сдвинуть число, находящееся в R0, на 2 бита вправо.

Программа

```
SHR     R0,#2; сдвиг вправо на 2 бита
```

Если, предположим, до выполнения операции сдвига в РОН R0 информация имела вид

R0–0000 0000 0110 0000,

то после выполнения операции сдвига вправо содержимое РОН R0 примет вид

R0–0000 0000 0001 1000.

5.6. Безусловные и условные переходы

5.6.1. Безусловный переход

Безусловный переход обеспечивает переход программы с данного адреса на указанный в программе без каких-либо условий. Безусловный переход

применяется в тех случаях, когда необходимо изменить последовательность адресов ячеек памяти программы, из которых считывается обрабатываемая информация.

Безусловный переход задается командами

```
JMPR    CC_UC,<метка>;
JMPI    CC_UC,[Rb];
```

где JMPR – мнемокод перехода, со смещением; JMPI – мнемокод перехода по указанному адресу; CC_UC – условие (безусловный переход); <метка> – символ, установленный перед командой программы, к которой необходимо перейти.

Пример 5.6.1

Дана программа

```
    MOV     R0,#FF8Dh; R0←FF8Dh
    MOV     R1,#005Ch; R1←005Ch
    ADD     R0,R1;
    SHR     R0,#2h; сдвиг вправо на 2 бита
    MUL     R0,R1; R0:=R0×R1
    SUB     R0,R1; R0:=R0–R1
```

Необходимо после выполнения команды арифметического сложения (ADD) без каких-либо условий произвести операцию вычитания (SUB), игнорируя команды сдвига (SHR) и умножения (MUL).

В этом случае программа примет вид

```
    MOV     R0,#FF8Dh; R0←FF8Dh
    MOV     R1,#005Ch; R1←005Ch
    ADD     R0,R1;
    JMPR    CC_UC, WQ;
    SHR     R0,#2h; сдвиг вправо на 2 бита
    MUL     R0,R1; R0:=R0×R1
WQ: SUB     R0,R1; R0:=R0–R1
```

5.6.2. Условные переходы

Команды условных переходов делятся на двухбайтовые и битовые.

Двухбайтовые команды условных переходов реализуются после выполнения тех операций в АЛУ, которые изменяют значения сигнала в регистре состояний.

Команды условных переходов имеют вид

<мнемокод> <условие>,<метка>

Мнемокод двухбайтовых команд условных переходов

JMPR– переход по смещению.

Условия в двухбайтовых командах условных переходов:

CC_Z – результат операции равен нулю;

CC_NZ – результат операции не равен нулю;

CC_ULT – левый операнд меньше правого (без знака);
CC_UGT – левый операнд больше правого (без знака);
CC_SLT – меньше (со знаком);
CC_SGT – больше (со знаком);
CC_SLE – меньше или равно (со знаком);
CC_SGE – больше или равно (со знаком).

Мнемокоды битовых команд условных переходов:
JB – переход по метке, если в устройстве, указанном в условии, установлен бит (единица);
JNB – переход по метке, если в устройстве, указанном в условии, не установлен бит (единица).

Если объявленное в команде условие в ходе реализации программы выполняется, то программа переходит по метке; если условие не выполняется, то команда условного перехода игнорируется.

Пример 5.6.2

Если в результате операции вычитания разность станет равной нулю, то, игнорируя команду пересылки, произвести операцию сдвига информации.
Программа

```
        SUB     R0,R3; R0:=R0–R3
        JMPR    CC_Z,F; если разность равна нулю, то перейти по метке F
        MOV     R5,R6; R5←R6

F:      SHL     R0,#2; сдвиг влево на 2 бита
```

5.6.3. Условный переход со сравнением операндов

При этом условном переходе задаются две команды: команда сравнения операндов (мнемокод CMP) и команда собственно условного перехода. При этом результат сравнения операндов никуда не заносится, содержимое регистров не меняется, но в результате выполнения операции сравнения устанавливаются новые значения сигналов в регистре состояний.

Команды сравнения операндов имеют вид

```
CMP     Rw,Rw;.
CMP     Rw,#data 16;
CMPB    Rb,Rb;
CMPB    Rb,#data 8;
```

Пример 5.6.3

В программе

```
MOV     R0,R5; R0←R5
MOV     R3,R6; R3←R6
SUB     R0,R3; R0:=R0–R3
JMPR    cc_Z,F; если разность равна нулю, то перейти по метке F
```

```
        MOV     R5,R6; R5←R6

F:      SHL   R0, #2; сдвиг влево на 2 бита
```

произвести сравнение содержимого регистров R0 и R3, не изменяя их значения. Программа

```
        MOV     R0,R5; R0←R5
        MOV     R3,R6; R3←R6
        CMP     R0,R3; R0−R3
        JMPR    CC_Z,F; если разность равна нулю, то перейти по метке F
        MOV     R5,R6; R5←R6

F:      SHL     R0,#2; сдвиг влево на 2 бита
```

В данном пункте письменных лекций указаны только те команды, которые могут быть использованы студентами при выполнении контрльных работ. Полный список команд дается в [1, 2].

Вопросы для самоконтроля

1. Перечислите все методы пересылки информации в МК.

2. В чем отличие логических от арифметических операций?

3. На базе каких сигналов ЦП реализуются условные переходы?

6. ПАРАЛЛЕЛЬНЫЕ ПОРТЫ ВВОДА/ВЫВОДА ИНФОРМАЦИИ

Для ввода информации о состоянии технологического оборудования или вывода команд управления технологическим оборудованием промышленный контроллер имеет определенное количество электрических линий связи, которые объединяются в группы и образуют порты ввода/вывода информации [2]. Объем одного порта может составлять 4, 8 или 16 бит (линий). Каждая линия связи соединяет ПК с одним из объектов технологического оборудования и обеспечивает пересылку информации в режиме либо контроля, либо управления. В режиме контроля по линии связи в ПК могут поступать сигналы в виде логической единицы (+5 вольт), либо в виде логического нуля (нулевое значение напряжения). Это дает возможность судить о состоянии технологического объекта: включено/выключено, больше/меньше и т. д. В режиме управления по линии связи из ПК подаются на технологический объект сигналы в виде логической единицы или логического нуля, что должно соответствовать командам типа “включить”, “выключить” и т. д. Такое управление называется бинарным.

В адресном пространстве памяти каждый порт представлен регистрами данных и направления.

Регистр данных порта обозначается как PX или PX.x, где P – символ порта; X – номер порта; x – номер бита порта при битовых операциях. Регистр

данных служит для промежуточного запоминания вводимой или выводимой информации. В режиме ввода информация с линий связи поступает в виде логических единиц или логических нулей в регистры данных. В дальнейшем эта информация может быть считана в один из регистров общего назначения. В режиме вывода информация в виде логических единиц или нулей, заранее занесенная программным методом в регистр данных, выводится на линии связи, а следовательно, на технологические объекты.

Регистр направления порта обозначается как DPX или DPX.x, где D – символ регистра направления. Перед выводом или вводом информации в регистр направления программным методом заносится сигнал, задающий режим работы порта или отдельного бита порта. При занесении в регистр направления всего порта или отдельного бита порта единицы порт или отдельный бит порта работают на вывод информации из МК. При занесении в регистр направления всего порта или отдельного бита порта нуля порт или отдельный бит порта работают на ввод информации в МК.

Порты, в зависимости от реализуемых ими функций, могут быть трех типов.

1. Однонаправленные порты, предназначенные только для ввода или только для вывода информации.

2. Двунаправленные порты, через которые информация может как выводиться из МК, так и вводиться в МК.

3. Многофункциональные порты, связанные с функциональными блоками аналого-цифровых преобразователей, таймерами, каналами широтно-импульсной модуляции и т. д. Если соответствующий функциональный блок МК в данный момент не используется, то линии порта, связанные с этим блоком, могут быть использованы как обычные порты для бинарного управления.

Каждый порт может работать в одном из трех режимов.

1. Режим программного ввода/вывода информации, когда моменты ввода или вывода информации определяются алгоритмом выполняемой программы и не зависят от времени поступления в порт пересылаемой информации.

2. Режим ввода/вывода информации со стробированием, при котором каждое изменение информации на линиях порта сопровождается стробом подтверждения.

3. Режим ввода/вывода информации с набором сигналов квитирования. В данном режиме пересылаемая через порт информация сопровождается как со стороны пересылки информации, так и со стороны приема информации сигналами о начале передачи и окончании приема.

МК типа 80C167 имеет 111 электрических линий связи, которые организованы в девять параллельных портов. Нумерация и обозначения линий связи портов МК типа 80C167 даны в приложении 2.

Двунаправленный порт P0 разрядностью 16 бит обслуживает внешнюю шину данных.

Однонаправленный порт P1 разрядностью 16 бит обслуживает внешнюю шину адреса.

Каждый из этих портов может быть использован в режиме двух 8-битовых портов (PH0, PL0 и PH1, PL1).

Двунаправленный порт P2 разрядностью 16 бит либо обслуживает каналы с широтно-импульсной модуляцией, либо служит для связи МК с технологическими объектами в обычном режиме бинарного управления.

Двунаправленный порт P3 разрядностью 15 бит либо обслуживает таймеры и последовательные порты, либо служит для связи МК с технологическими объектами в режиме бинарного управления.

Смешанный порт P4 (биты 0, 1, 2, 3, 4 и 7 являются однонаправленные, а биты 5 и 6 – двунаправленные) разрядностью 8 бит обслуживает внешнюю шину адреса и CAN-контроллер.

Однонаправленный порт P5 разрядностью 16 бит служит для ввода аналоговой информации с технологического оборудования для преобразования ее в цифровую форму.

Двунаправленный порт P6 разрядностью 8 бит служит для вывода информации выбора внешних устройств или связи МК с технологическими объектами в режиме бинарного управления.

Двунаправленные порты P7 и P8 разрядностью 8 бит каждый либо обслуживают каналы с широтно-импульсной модуляцией, либо обеспечивают связь МК с технологическими объектами в режиме бинарного управления.

Кроме порта P5, все порты битадресуемые. Двунаправленные порты в режиме ввода информации включаются в высокоимпедансное состояние. Большинство портов могут выполнять как минимум две функции. Поэтому все биты каждого порта имеют, как правило, два наименования. Одно наименование относится к работе порта в режиме бинарного управления и состоит из номера порта и номера бита. Второе наименование связано с работой того или иного функционального блока МК.

Правила программирования работы портов в режиме бинарного управления.

1. Режим ввода информации в объеме порта.

Для задания этого режима необходимо:

записать нули в регистр направления порта передачи информации

```
CLR     DPX;
```

где X – номер порта;

считать информацию из регистра данных порта в любой РОН

```
MOV     RZ,PX;
```

2. Режим вывода информации в объеме порта.

Для задания этого режима необходимо:

переслать выводимую информацию из РОН (куда она была заранее помещена или находилась там после очередной операции в ЦП) в регистр данных порта

```
MOV     PX,RZ;
```

записать единицы в регистр направления передачи информации

```
SET     DPX;
```

3. Режим ввода информации с определенной линии порта (битовый режим).

Для задания этого режима необходимо:

записать ноль в бит регистра направления порта передачи информации, номер которого характеризует номер выбранной линии порта,

BCLR DPX.x;

где X характеризует номер порта, а x характеризует номер бита порта;

считать содержимое указанного бита порта из регистра данных порта в определенный бит выбранного РОН

BMOV RZ.z, PX.x;

4. Режим вывода информации из определенного бита выбранного РОН через определенный бит порта (битовый режим).

Для задания этого режима необходимо:

поместить в выбранный бит регистра данных порта выводимую информацию

BMOV PX.x,RZ.z; BSET PX.x; BCLR PX.x;

записать единицу в бит регистра направления порта передачи информации с номером используемой линии

BSET DPX.x;

Вопросы для самоконтроля

1. Можно ли через один и тот же порт выводить информацию как в параллельном коде, так и в объеме одного бита?

2. Какую информацию выведет ПК через порт, если в программе сначала задать направление вывода информации, а затем переслать информацию в регистр данных порта?

7. ТАЙМЕРЫ

Таймеры предназначены для генерации временных интервалов, обеспечивающих работу промышленного контроллера в реальном масштабе времени [2].

Микроконтроллер 80C167 имеет два блока таймеров GPT1 и GPT2 (GPT– General Purpose Timer). Блок GPT1 содержит 3 таймера (T2, T3 и T4). Блок GPT2 содержит 2 таймера (T5 и T6).

Все таймеры могут работать в одном из двух режимов.

В первом режиме таймер вырабатывает временной интервал, на величину которого происходит задержка выполнения основной управляющей программы. Этот режим называется режимом выходного сравнения.

Во втором режиме таймер производит измерение временных интервалов между двумя импульсами, поступившими на его вход. Этот режим называется режимом входного захвата.

Программно-логическая модель одного таймера представлена на рис.7.1.

В режиме выходного сравнения на вход таймера поступают тактовые импульсы с внутреннего генератора тактовых импульсов МК (f_{CPU}). При

питании таймера тактовыми импульсами от внутреннего генератора возможно программным методом производить изменение частоты следования тактовых импульсов, меняя коэффициент деления в делителе частоты ДЧ командой TXI. С делителя частоты тактовые импульсы поступают на счетчик Сч, в котором происходит суммирование (вычитание) тактовых импульсов.

Счетчик имеет разрядность 16 бит, и максимальное число импульсов, которое он может запомнить, равно 2^{16}=65535. При поступлении на счетчик 65535+1 импульсов он переходит в нулевое состояние.

Включение таймера происходит за счет замыкания электрической цепи между делителем частоты и счетчиком. Для этого необходимо подать команду TXR. В момент переполнения счетчика (на счетчик поступило 65536 импульсов) в триггер переполнения TXOTL записывается единица, характеризующая переполнение счетчика. При наличии разрешения TXDE на выходе TXOUT формируется сигнал запроса на прерывание.

В регистр данных TX программным методом заносится код выдержки времени (*A*), который характеризует количество импульсов, на которое необходимо предварительно уменьшить объем счетчика для достижения заданной выдержки времени. В регистре данных происходит автоматическое вычитание кода выдержки времени из общей суммы счетчика (65535), а полученная разность вновь заносится в регистр данных. Следовательно, код выдержки времени при известном периоде следования импульсов обеспечит заданную величину выдержки времени.

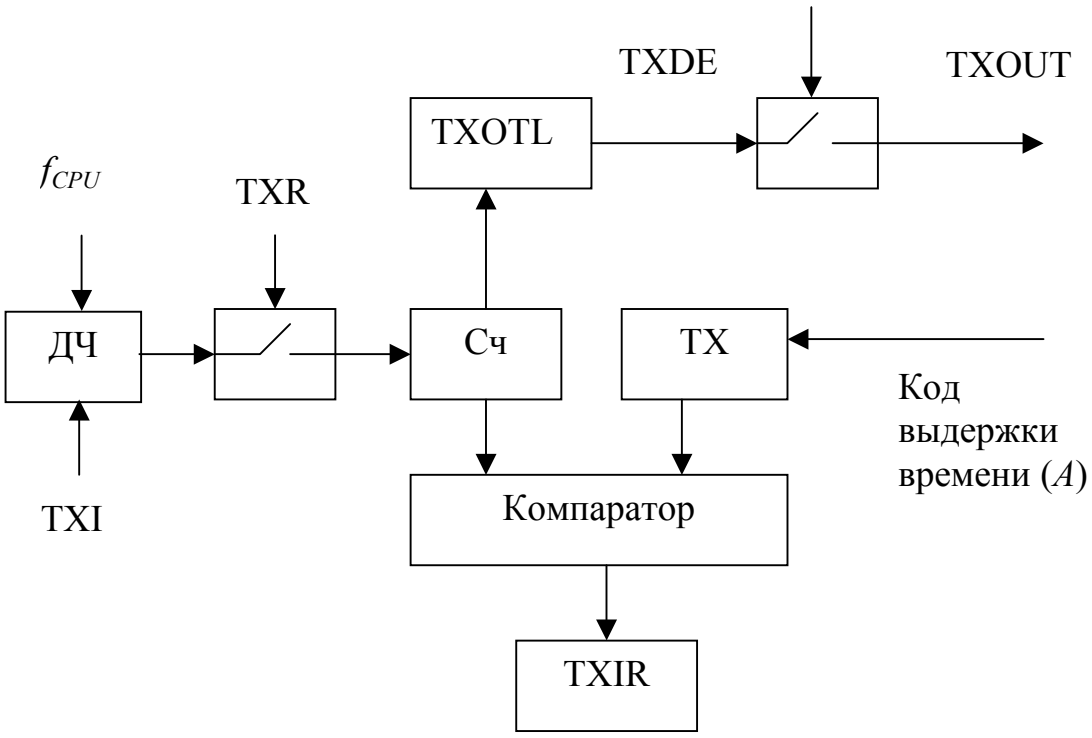


Рис. 7.1

По мере заполнения счетчика его текущая сумма постоянно сравнивается в компараторе с кодом, находящимся в регистре данных (рис. 7.2). В момент равенства текущей суммы счетчика и кода в регистре данных в триггер окончания выдержки времени TXIR заносится единица. Она и характеризует окончание работы счетчика, а, следовательно, и всего таймера.

Величина выдержки времени (T) определяется по формуле

$$T = \frac{2^{16} - A}{f_{CPU}} \cdot k_d ,$$

где 2^{16} – полный объем счетчика; A – код выдержки времени; f_{CPU}- частота следования тактовых импульсов с внутреннего генератора тактовых импульсов (20 МГц); k_d– коэффициент деления делителя частоты.

При подготовке управляющей программы чаще необходимо определять не величину выдержки времени (T), которая, как правило, задана, а код выдержки времени (A) и коэффициент деления (k_d) по заданной величине выдержки времени.

Код выдержки времени находится по формуле

$$A = 2^{16} - \frac{T \cdot f_{CPU}}{k_d} .$$

При выборе коэффициента деления следует руководствоваться следующим ограничением:

$$2^{16} \geq \frac{T \cdot f_{CPU}}{k_d} ,$$

откуда при f_{CPU}=20 10^6Гц

$$k_d \geq 308{,}18 \cdot T .$$

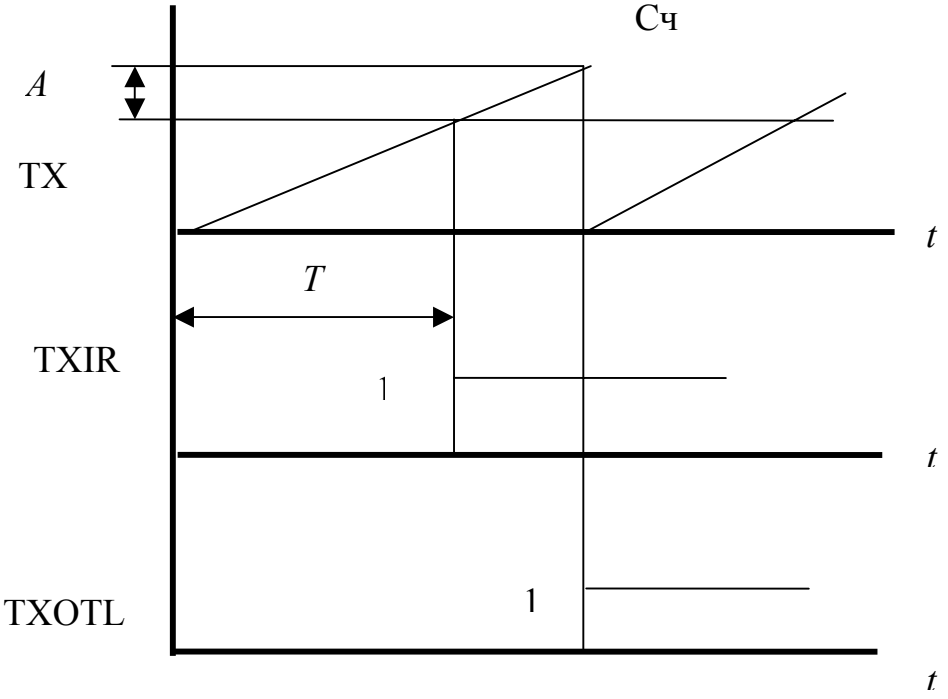


Рис. 7.2

Величина коэффициента деления лежит в диапазоне $8 \le k_d \le 1024$.

Все таймеры содержат следующие устройства управления.

1. Регистр управления – TXCON (Control Registers), формат (количество двоичных разрядов) и содержание (наименование полей) которого указаны в табл. 7.1.

Т а б л и ц а 7.1

15 14 13 12 11	10	9	8	7	6	5 4 3	2 1 0
	TXOTL	TXOE	TXUDE	TXUD	TXR	TXM	TXI

Поля регистра управления имеют следующие значения.

TXI. Три бита (0, 1 и 2) регистра управления предназначены для установки коэффициента деления в делителе частоты (ДЧ) в случае питания таймера от внутреннего генератора тактовых импульсов (f_{CPU}) частотой 20 МГц (при TXM=000). В табл. 7.2 указаны коды, которые могут устанавливаться в поле TXI, значения коэффициентов деления, частоты на входе счетчика после делителя частоты, величины периодов входной частоты и периодов заполнения счетчика импульсами поделенной частоты.

TXM. Три бита (3, 4 и 5) поля TXM регистра управления предназначены для выбора генератора тактовых импульсов:

000 – таймер питается тактовыми импульсами от внутреннего генератора МК;

001 – на вход таймера поступают импульсы с внешних источников.

TXR. Один бит (6) поля TXR регистра управления предназначен для включения таймера в работу:

0 – таймер выключен;

1 – таймер включен.

TXUD, TXUDE. Два бита (7 и 8) двух полей TXUD и TXUDE регистра управления предназначены для установки направления счета в счетчике.

TXUD=0 и TXUDE=0 – суммирование тактовых импульсов в счетчике;

TXUD=0 и TXUDE=1 – вычитание тактовых импульсов из суммы, находящейся в счетчике в данный момент.

TXOE. Один бит (9) поля TXOE регистра управления предназначен для разрешения выхода на TXOUT команды переполнения счетчика:

0 – выход отключен;

1 – выход включен.

TXOTL. Один бит (10) поля TXOTL регистра управления предназначен для формирования команды, сигнализирующей о переполнении счетчика:

0 – счетчик не переполнен;

1 – счетчик переполнен.

2. Регистр данных TX (Data Registers) предназначен для преобразования и хранения занесенного в него программным способом кода выдержки времени (A).

Т а б л и ц а 7.2

Код	k_d	Входная частота	Период входной частоты	Период заполнения счетчика
000	8	2,5 мГц	400 нс	26 мс
001	16	1,25 мГц	800 нс	52,5 мс
010	32	625 кГц	1,6 мкс	105 мс
011	64	312,5 кГц	3,2 мкс	210 мс
100	128	156,25 кГц	6,4 мкс	420 мс
101	256	78,125 кГц	12,8 мкс	840 мс
110	512	39,06 кГц	25,6 мкс	1,68 с
111	1024	19,53 кГц	51,2 мкс	3,36 с

3. Триггер окончания временной задержки TXIR предназначен для формирования команды, сигнализирующей об окончании выдержки времени:

TXIR=0 – выдержка времени не закончилась;

TXIR=1 – выдержка времени закончилась.

Таймеры Т2, Т3 и Т4 содержат двухбайтовые счетчики, что дает возможность при коэффициенте деления, равном 1024, обеспечить выдержку времени в 3,36 с. Таймеры Т5 и Т6 обладают однобайтовыми счетчиками. Кроме того, таймер Т6 может работать в режиме генератора импульсов заданной частоты.

Для задания выдержки времени необходимо:

1) рассчитать код выдержки времени (*A*) и результат записать в любой РОН;

2) переслать через стек значение кода выдержки времени из выбранного РОН в регистр данных TX;

2) заполнить битовые поля регистра управления TXCON и полученный код записать в регистр управления;

3) установить программу на ожидание окончания выдержки времени (ожидание появления единицы в триггере окончания временной задержки TXIR);

4) по окончании выдержки времени выключить таймер (TXR=0) и записать ноль в триггер окончания выдержки времени TXIR.

Пример 7.1

Запрограммировать выдержку времени длительностью в 1с, использовав таймер 2.

Расчет кода выдержки времени. Предварительно определяется коэффициент деления из условия

$$k_d \geq 308{,}18 \cdot T .$$

При T=1с условие примет вид k_d≥308,18. Ближайшее большее значение коэффициента деления равно k_d=512. Тогда код выдержки времени будет равен

$$A = 2^{16} - \frac{T \cdot f_{CPU}}{k_d} = 65535 - \frac{1 \cdot 20 \cdot 10^6}{512} = 65535 - 39063 = 26472.$$

В двоичной системе счисления код выдержки времени примет вид

37768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
0	1	1	0	0	1	1	1	0	1	1	0	1	0	0	0

В шестнадцатеричной системе счисления код выдержки времени примет вид

A=6768h.

Величина выдержки времени заносится в один из регистров общего назначения. Регистр общего назначения выбирается произвольно.

```
MOV     R0, #6768h;
```

Производится пересылка содержимого R0 в регистр данных через стек

```
PUSH    R0;
POP     T2;
```

Заполняются битовые поля регистра управления T2CON в соответствии с выбранным режимом работы таймера и формируется код, который должен быть занесен в регистр управления.

T2I=110– коэффициент деления делителя частоты 512;

T2M=000 – таймер работает от внутреннего генератора тактовых импульсов;

T2R=1 – таймер включен;

T2UD=0, T2UDE=0 – суммирование в счетчике тактовых импульсов;

T2OE=0 – выход отключен;

T2OTL=0 – счетчик не переполнен.

В соответствии с данным режимом работы таймера код в двоичной системе счисления примет вид

0000 0000 0100 0110

или в шестнадцатеричной системе счисления

0046h.

Код режима работы таймера заносится в регистр управления.

```
MOV     T2CON,#0046h;
```

Установка программы на ожидание окончания выдержки времени (появление единицы в T2IR).

```
W:  NOP; пустой шаг – некоторая задержка времени в выполнении программы
    JNB  T2IR,W; условный переход: если вT2IR нет единицы, то перейти по метке W
```

Выключение таймера и установка триггера T2IR в ноль.

```
      MOV     T2CON,#0000h;
      BCLR    T2IR;
```

Компактно программа будет иметь вид

```
      MOV     R0,#6768h;
      PUSH    R0;
      POP     T2;
      MOV     T2CON,#0046h;
W:    NOP;
      JNB     T2IR,W;
      MOV     T2CON,#0000h;
      BCLR    T2IR;
```

Вопросы для самоконтроля

1. Максимальное время выдержки одного таймера 3,36 с. Возможно ли программным методом увеличить время выдержки и каким образом?

2. Что характеризует собой код выдержки времени? Может ли он количественно быть равным или превышать объем счетчика?

8. КАНАЛЫ ШИРОТНО-ИМПУЛЬСНОЙ МОДУЛЯЦИИ

Каналы широтно-импульсной модуляции (К ШИМ) предназначены для выработки на выходах МК электрических сигналов в виде последовательности импульсов, частота следования и длительность которых задаются программным методом [2]. Такая последовательность импульсов с переменной длительностью позволяет плавно менять напряжение на управляемом технологическом объекте. Изменение длительности импульсов на 1 % вызывает изменение напряжения на объекте на 0,05 В.

МК содержит 2 блока каналов ШИМ. Каждый блок состоит из 16 каналов ШИМ, двух специальных таймеров (T0 и T1 для первого блока – T01CON и T7 и T8 для второго блока – T78CON). Это позволяет формировать до 32 независимых каналов ШИМ.

Программно-логическая модель одного канала ШИМ приведена на рис. 8.1.

Специальные таймеры (TXZCON), предназначенные для работы с каналами ШИМ, тактируются импульсами, поступающими с внутреннего генератора тактовых импульсов f_{CPU}.

Выбирая коэффициент деления делителя частоты данного таймера и занося в регистр переполнения TXREL код, характеризующий периодическое переполнение счетчика, возможно программным методом менять период заполнения счетчика, а следовательно, и частоту следования импульсов на выходе канала ШИМ.

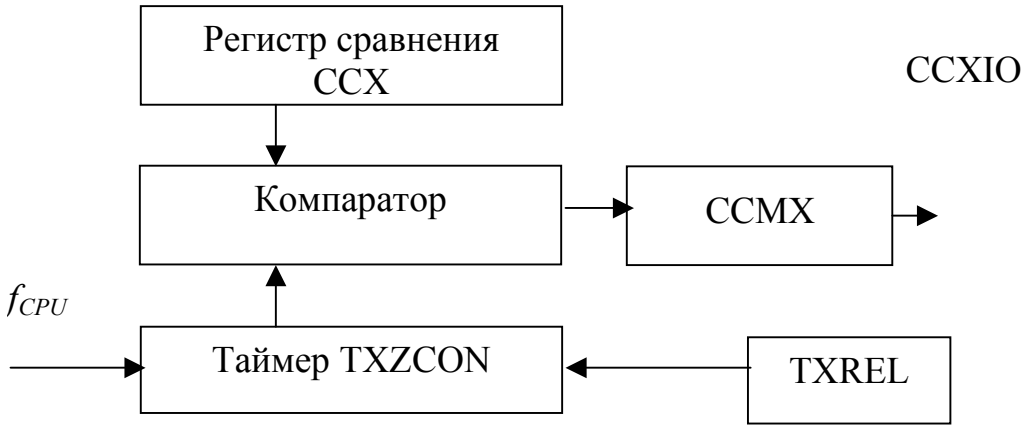


Рис. 8.1

В регистр сравнения CCX заносится код индекса модуляции, характеризующий длительность импульсов на выходе канала ШИМ. В компараторе происходит постоянно сравнение текущих кодов счетчика специального таймера и регистра сравнения.

В моменты равенства этих кодов на выходе канала ШИМ логический уровень сигнала меняется на противоположный. Так, если в момент переполнения счетчика таймера на выходе канала ШИМ устанавливается логический ноль, то в момент равенства кодов на выходе канала ШИМ устанавливается логическая единица (рис. 8.2).

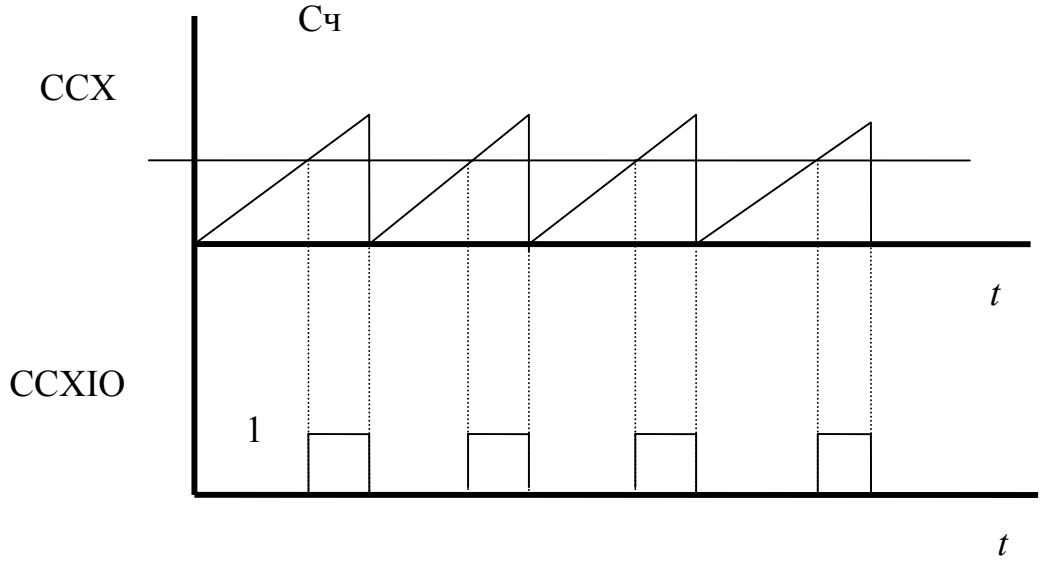


Рис.8.2

Каждый канал ШИМ может работать в одном из трех режимов. Вид используемого режима определяется кодом, который заносится в регистр режима CCMX.

Один регистр режима CCMX содержит коды режимов одновременно для четырех каналов ШИМ. Всего регистров режимов в МК 8 штук. Например, в

регистре режима CCM0 могут быть записаны коды режимов для каналов ШИМ с номерами 0, 1, 2 и 3. Каждый канал ШИМ в регистре режима располагает 4 битами. В первых трех битах помещается информация о режиме работы канала ШИМ. При помещении в этих трех битах трех единиц канал ШИМ работает без фазового сдвига импульсов. Четвертый бит характеризует выбор специального таймера для работы с данным каналом ШИМ. При помещении в этот бит нуля выбирается нулевой таймер T0 (T01CON).

Значение кода переполнения (B) определяется из выражения

$$B = 2^{16} - \frac{f_{CPU}}{k_d \cdot f_{ВЫХ}},$$

где 2^{16} – объем счетчика специального таймера; f_{CPU} – частота внутреннего тактового генератора; k_d – коэффициент деления делителя частоты; $f_{ВЫХ}$ – выходная частота импульсов с широтно-импульсной модуляцией.

Выбор коэффициента деления определяется неравенством вида

$$k_d \geq \frac{308{,}18}{f_{ВЫХ}}.$$

Величина коэффициента деления лежит в диапазоне $8 \leq k_d \leq 1024$.

Расчет кода индекса модуляции производится по формуле

$$M = \left(2^{16} - B\right) \cdot \left(1 - I_m\right) = \left(2^{16} - 2^{16} + \frac{f_{CPU}}{k_d \cdot f_{ВЫХ}}\right) \cdot \left(1 - I_m\right) = \frac{f_{CPU}}{k_d \cdot f_{ВЫХ}} \left(1 - I_m\right),$$

где I_m – величина индекса модуляции, выраженная в долях от заданного периода следования импульсов на выходе канала с широтно-импульсной модуляцией.

Каждый канал с ШИМ может работать в одном из трех режимов.

1. В режиме генерации импульсов с изменяющейся длительностью и с фиксированными частотами (табл. 8.1), которые определяются выбранными коэффициентами деления. В этом режиме регистры переполнения не участвуют.

Т а б л и ц а 8.1

k_d	8	16	32	64	128	256	512	1024
$f_{ВЫХ}$, Гц	38,46	19,04	9,52	4,76	2,38	1,19	0,59	0,29

2. В режиме генерации импульсов с изменяющейся длительностью и произвольно выбранной частотой, определяемой коэффициентом деления и кодом переполнения. В этом режиме частоты (с минимальной дискретностью, которая определяется единицей младшего разряда кода переполнения) будут находиться в следующих диапазонах (табл. 8.2).

Т а б л и ц а 8.2

k_d	8	16	32	64	128	256	512	1024
При TXRTL=0, Гц	38,46	19,04	9,52	4,76	2,38	1,19	0,59	0,29
При TXREL=65534	25 кГц	12,5 кГц	6,25 кГц	3,125 кГц	1,562 кГц	781 Гц	390 Гц	195 Гц

3. В режиме генерации импульсов с изменяющейся длительностью, фазой и произвольно выбранной частотой.

Изменение фазы достигается за счет использования одновременно двух регистров сравнения в одном канале ШИМ, что позволяет сдвигать импульсы во всем пространстве периода выходной частоты (рис. 8.3).

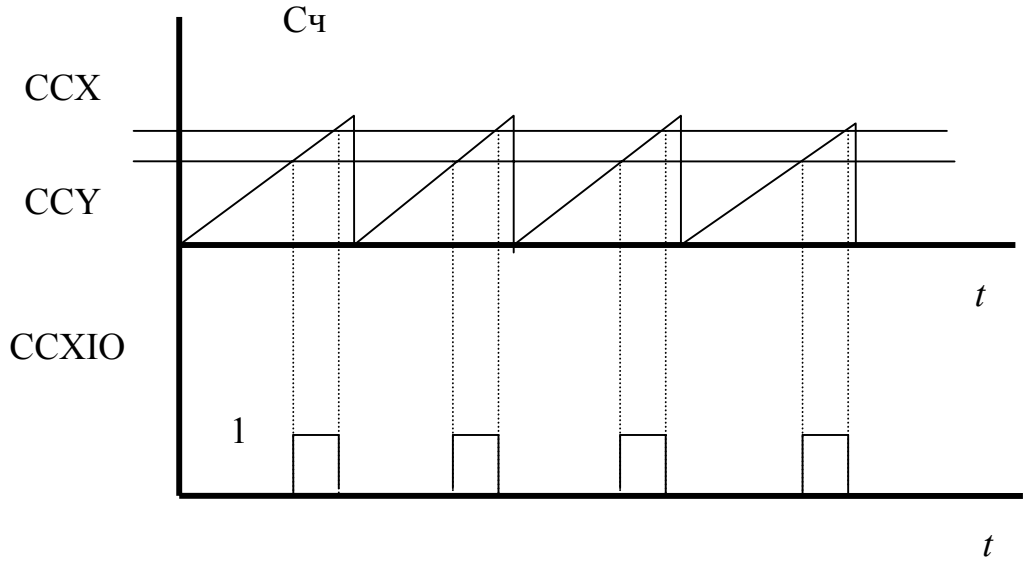


Рис.8.3

Каналы ШИМ содержат следующие органы управления.

1. Регистры управления специальных таймеров – T01CON или **T78CON** (Control Registers). Формат и содержание регистра управления T01CON указаны в табл. 8.3.

Т а б л и ц а 8.3

15	14	13 12	11	10 9 8	7	6	5 4	3	2 1 0
	T1R		T1M	T1I		T0R		T0M	T0I

T0I. Три бита поля T0I регистра управления T01CON предназначены для установки коэффициента деления в делителе частоты (ДЧ) в случае питания таймера от внутреннего генератора (f_{CPU}) тактовой частоты. В табл. 8.4 указаны коды, которые могут устанавливаться в поле T0I, значения коэффициентов деления частоты, частоты на входе таймера после делителя частоты, величины

периодов входной частоты и периодов заполнения счетчика импульсами поделенной частоты в случае объема счетчика 65535 импульсов.

T0M. Один бит поля T0M регистра управления T01CON предназначен для выбора режима работы таймера:

T0M=0 – режим генерации импульсов с подачей тактовых импульсов с внутреннего генератора МК;

T0M=1 – режим генерации импульсов с подачей тактовых импульсов с внешнего генератора;

Т а б л и ц а 8.4

Код	k_d.	Входная частота	Период входной частоты	Период заполнения счетчика
000	8	2,5 мГц	400 нс	26 мс
001	16	1,25 мГц	800 нс	52,5 мс
010	32	625 кГц	1,6 мкс	105 мс
011	64	312,5 кГц	3,2 мкс	210 мс
100	128	156,25 кГц	6,4 мкс	420 мс
101	256	78,125 кГц	12,8 мкс	840 мс
110	512	39,06 кГц	25,6 мкс	1,68 с
111	1024	19,53 кГц	51,2 мкс	3,36 с

T0R. Один бит поля T0R регистра управления предназначен для включения или выключения канала ШИМ:

T0R=0 – канал выключен;

T0R=1 – канал включен.

В этом же регистре управления указаны аналогичные поля и для таймера T1.

2. Регистр сравнения CCX предназначен для запоминания кода индекса модуляции.

3. Регистр переполнения TXREL предназначен для запоминания кода переполнения.

4. Регистр режима работы CCMX канала ШИМ (один регистр CCMX предназначен для одновременного управления четырьмя каналами ШИМ; например, регистр CCM0 управляет работой следующих каналов ШИМ: 0, 1, 2 и 3). Формат и содержание регистра режима CCM0 указаны в табл. 8.5.

Таблица 8.5

15	14 13 12	11	10 9 8	7	6 5 4	3	2 1 0
ACC3	CCMOD3	ACC2	CCMOD2	ACC1	CCMOD1	ACC0	CCMOD0

CCMODX. Три бита поля регистра режима работы CCMX предназначены для выбора режима работы канала ШИМ:

111 – канал работает в режиме генерации ШИМ без фазового сдвига.

ACCX. Один бит поля регистра управления CCMX предназначен для выбора специальных таймеров, которые должны работать с данным каналом ШИМ:

0 – таймеры T0, T1;

1 – таймеры T7, T8.

Для задания режима работы канала с широтно-импульсной модуляцией с фиксированными частотами следования импульсов в программе необходимо:

1) рассчитать код индекса модуляции (*M*) и занести его в регистр сравнения CCX через стек, использовав любой РОН;

2) заполнить битовые поля регистра режима и полученный код режима занести в регистр режима CCMX;

3) заполнить битовые поля регистра управления специальным таймером и полученный код управления занести в регистр управления TXZCON;

4) задать направление работы порта на вывод информации.

Пример 8.1

Образовать на выходе канала ШИМ с номером 0 сигнал с фиксированной частотой следования импульсов 0,29 Гц и индексом модуляции 0,5.

Расчет кода индекса модуляции. Так как частота следования импульсов (0,29Гц) является фиксированной, то *B*=0 и код индекса модуляции в десятичной системе счисления будет равен

$$M = 2^{16} \cdot (1 - I_m) = 65535 \cdot 0{,}5 = 32767 .$$

В шестнадцатеричной системе счисления код индекса модуляции примет вид 8000h.

Запись кода индекса модуляции в регистр сравнения. Эта операция производится через стек. В качестве регистра общего назначения может быть выбран любой регистр; в данном примере выбирается R1. Тогда фрагмент программы примет вид

```
MOV     R1,#8000h;
PUSH    R1;
POP     CC0;
```

Заполняются битовые поля регистра режима.

CCM0=111,

ACC0=0.

В этом примере все битовые поля остальных каналов ШИМ заполняются нулями. Тогда код режима работы будет иметь вид 0007h. Полученный код заносится в регистр режима

MOV CCM0, #0007h;

Заполняются битовые поля регистра управления специальным таймером (см. пример 7.1). Разница состоит в том, что в примере 7.1 коэффициент деления был 512, а в данном примере этот коэффициент равен 1024. Поэтому код примет значение 0047h. Полученный код управления записывается в регистр управления таймером

MOV T01CON,#0047h;

Задается направление работы порта

BSET DP2.0;

Компактно программа будет иметь вид

```
MOV     R1,#8000h;
PUSH    R1;
POP     CC0;
MOV     CCM0,#0007h;
MOV     T01CON,#0047h;
BSET    DP2.0;
```

Для задания режима работы канала ШИМ с произвольно выбранной частотой следования импульсов в программе необходимо:

1) рассчитать код переполнения и занести его в регистр переполнения (TXREL) через стек, использовав любой РОН;

2) рассчитать код индекса модуляции (*M*) и занести его в регистр сравнения CCX через стек, использовав любой РОН;

3) заполнить битовые поля регистра режима и полученный код режима занести в регистр режима CCMX;

4) заполнить битовые поля регистра управления специальным таймером и полученный код управления занести в регистр управления TXZCON;

5) задать направление работы порта на вывод информации.

Вопросы для самоконтроля

1. Каким образом в МК возможно реализовать цифро-аналоговый преобразователь?

2. Укажите минимальную и максимальную частоты, которые можно получить на выходе канала ШИМ?

9. АНАЛОГО-ЦИФРОВЫЕ ПРЕОБРАЗОВАТЕЛИ

Аналого-цифровые преобразователи (АЦП) предназначены для ввода в микроконтроллер аналоговых сигналов с преобразованием их в двоичную систему счисления [2]. Коэффициент АЦ-преобразования равен $K_{АЦП}$=205 (на 1 вольт входного сигнала).

Программно-логическая модель аналого-цифрового преобразователя представлена на рис. 9.1. В соответствии с информацией, занесенной в регистр конфигурации ADCON, коммутатор К подключает к аналого-цифровому преобразователю АЦП выбранную линию порта P5.x. После преобразования аналогового сигнала в двоичную систему счисления результат преобразования помещается в регистр данных ADDAT, из которого его возможно считать в один из регистров общего назначения. Время окончания преобразования характеризуется появлением нуля в триггере окончания процесса преобразования ADBSY. После считывания результата преобразования из регистра данных в триггере окончания процесса преобразования автоматически устанавливается единица.

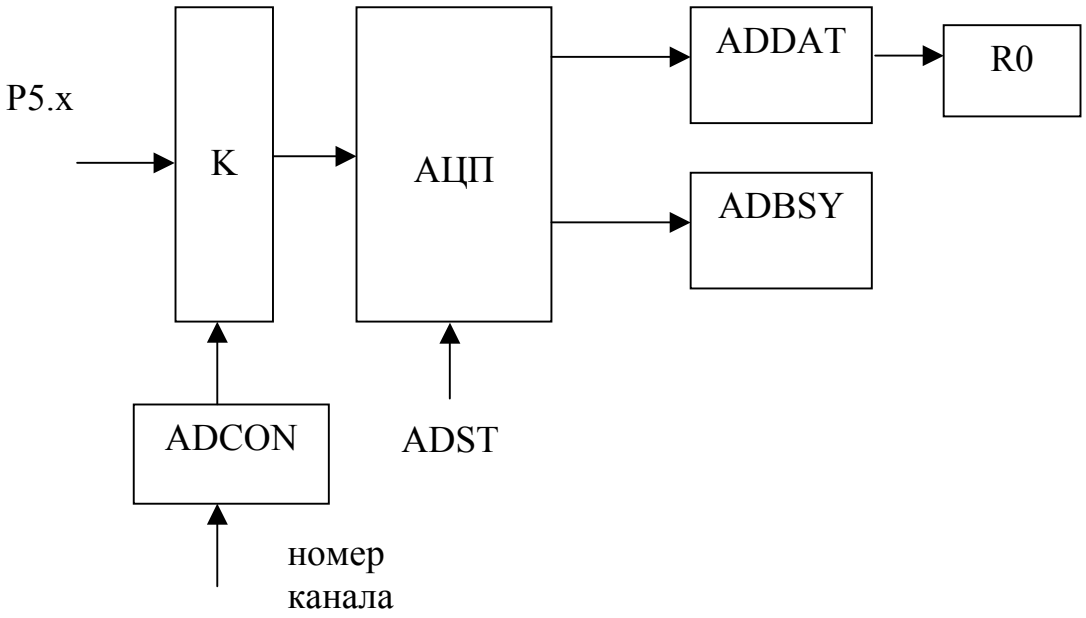


Рис. 9.1

Аналого-цифровой преобразователь имеет следующие органы управления.

1. Регистр конфигурации ADCON для запоминания номера линии порта, с которой в данный момент должна считываться информация в аналоговой форме.

2. Регистр данных ADDAT для хранения результата преобразования.

3. Триггер окончания преобразования ADBSY, в котором вырабатывается ноль по окончании преобразования.

4. Триггер ADST включения АЦП.

Для реализации режима аналого-цифрового преобразования необходимо:

1) записать в регистр конфигурации ADCON через стек номер линии порта, с которой должен поступать аналоговый сигнал, использовав любой РОН;

2) подать команду на включение аналого-цифрового преобразователя (ADST=1);

3) установить программу на ожидание окончания преобразования;

4) считать из регистра данных ADDAT в РОН через стек результат преобразования.

При использовании результата преобразования необходимо учитывать, что на вход АЦП возможно подавать аналоговый сигнал, амплитуда которого может изменяться в пределах от нуля до +5 вольт. При коэффициенте преобразования $К_{АЦП=}205$ результат работы АЦП помещается в 12 младших битах числа регистра данных. Остальные старшие 4 бита этого числа характеризуют номер линии связи, с которой производилось преобразование. Поэтому, например, при преобразовании сигнала в 2 вольта с линии связи номер 5 результат будет иметь следующее значение – 519Ah. Чтобы получить истинное значение преобразования, необходимо произвести операцию маскирования результата преобразования кодом 0FFFh

```
AND     R3,#0FFFh;
```

Пример 9.1

Преобразовать аналоговую информацию, поступающую по линии 15 порта P5 (P5.15).

Программа

```
    MOV     R10,#000Fh; запись в R10 номера линии 15 (000Fh) порта P5
    PUSH    R10;
    POP     ADCON; запись в регистр конфигурации содержимого R10
    BSET    ADST; включение АЦП
Q:  NOP;
    JB      ADBSY,Q; ожидание окончания преобразования; если преобразование не закончено, перейти по метке Q
    PUSH    ADDAT; считать из регистра данных информацию в РОН R3
    POP     R3;
    AND     R3,#0FFFh; маскирование результата преобразования
```

Компактно программа будет иметь вид.

```
MOV     R10,#000Fh;
PUSH    R10;
POP     ADCON;
BSET    ADST;
```

```
Q:    NOP;
      JB      ADBSY,Q;
      PUSH    ADDAT;
      POP     R3;
      AND     R3,#0FFFh;
```

Вопросы для самоконтроля

1. Возможно ли одновременно произвести ввод в МК нескольких аналоговых сигналов?

2. С какой целью введен фрагмент программы ожидания преобразования?

10. КОМПЛЕКТНОСТЬ ПРОМЫШЛЕННОГО КОНТРОЛЛЕРА

Комплект, обеспечивающий нормальную работу ПК, включает следующие аппаратные и программные средства [1].

К аппаратным средствам относятся:

1) покупная плата ПК с установленными на ней МК типа 80C167, интегральными схемами дополнительных ОЗУ и ПЗУ, программатором для записи кодов в ПЗУ, супервизором питания и приемными частями разъемов для подключения к ПК технологических объектов (приложение 3);

2) персональный компьютер под управлением операционной системы MS DOS 3.00 и выше;

3) кабель RS232 для подключения ПК к персональному компьютеру и ответные части разъемов для соединения ПК с технологическими объектами;

4) блок питания для ПК.

К программным средствам относятся полноэкранный отладчик, обеспечивающий режим интерактивного доступа ко всем ресурсам ПК и удобный интерфейс для отладки управляющих программ.

Полноэкранный отладчик представляет собой пакет следующих программ:

1) модуль отладчика – SFD7.EXE;

2) ядро отладчика – DBG167.PGM;

3) файл конфигурации отладчика – SFD7.CFG;

4) текст подсказки отладчика – SFD7.HLP.

Кроме того, пакет включает подкаталог PROC с модулями расширенных функций отладчика.

Для работы на ПК его через последовательный порт подключают к персональному компьютеру, в который заносятся программы полноэкранного отладчика. Через разъемы портов ввода/вывода информации к ПК подключаются датчики и исполнительные устройства технологического оборудования.

Разработка и отладка управляющих программ производится под управлением полноэкранного отладчика на персональном компьютере. В процессе отладки персональный компьютер передает на ПК команды и массивы данных, а с ПК получает информацию, характеризующую процесс реализации команд по заданному управляющей программой алгоритму. В процессе отладки управляющих программ на мониторе персонального компьютера отображается содержимое регистров ПК, ячеек внутренней памяти и несколько строк отлаживаемой программы. Отладка программы может происходить при реальном ее выполнении с индикацией поступающих с технологического оборудования и посылаемых на него сигналов.

После окончания отладки программа может быть переписана из ОЗУ в ПЗУ с помощью специальных функций отладчика и программатора, что дает возможность в дальнейшем многократного ее использования.

При наличии на производстве локальной вычислительной сети несколько ПК с отлаженными программами могут быть включены в эту сеть и управлять технологическим оборудованием самостоятельно, но под общим контролем сервера этой сети.

Вопросы для самоконтроля

1. Поясните назначение полноэкранного отладчика.

2. Для каких целей используется внешнее ПЗУ?

11. ПОДКЛЮЧЕНИЕ И ЗАПУСК ПРОМЫШЛЕННОГО КОНТРОЛЛЕРА

Для осуществления отладочного режима ПК его необходимо подключить к персональному компьютеру, в память которого уже введены программы полноэкранного отладчика.

Процесс подключения и запуска ПК состоит из следующих этапов.

1. Подключение ПК к персональному компьютеру с операционной системой MS DOS 3.00 и выше, либо совместимой с ней. Для этого необходимо соединить кабелем последовательного порта RS разъем J1 ПК (приложение 3) с разъемом порта COM2 персонального компьютера (к COM1 подключена мышь). Через этот последовательный порт в ПК из персонального компьютера пересылаются команды и массивы данных, а из ПК в персональный компьютер пересылаются результаты выполненных операций.

2. Подключить кнопку сброса ПК к разъему J3 ПК (сброс контроллера). Установить перемычку J2, а переключатели J8A и J8B – в положения 1–2.

3. Включить персональный компьютер.

4. Подать питание на ПК от стабилизированного источника питания на контакты разъема J9. Номера контактов:

1 или 49 – VCC, напряжение +5 вольт;

2 или 50 – GND, общий провод (корпус).

5. При работе персонального компьютера в операционной оболочке Microsoft Windows перейти в операционную систему MS DOS. Для этого необходимо:

1) на рабочем столе персонального компьютера щелкнуть левой клавишей мыши по кнопке “Пуск”;

2) установить курсор на ячейке “Завершение работы” и щелкнуть левой клавишей мыши;

3) установить курсор на ячейке “Перезагрузить компьютер в режиме MS DOS” и нажать кнопку “ОК”;

4) в появившемся окне в командную строку вписать символы NC (Norton Commander) и нажать клавишу Enter.

6. Скопировать программы полноэкранного отладчика с дискеты на один из логических дисков винчестера (например, на диск С).

7. Установить курсор на каталоге SFD7, который характеризует полноэкранный отладчик, и нажать клавишу Enter.

8. Установить курсор в каталоге SFD7 на файле sfd7.exe.

9. Нажать кнопку на ПК “Сброс контроллера”.

10. Нажать клавишу Enter. На экране монитора появится основной интерфейс полноэкранного отладчика с информационной строкой

80C167 Debugger.

Основной интерфейс отладчика состоит из 5 окон: окна управляющей программы (окно дисассемблера), окна памяти, окна регистров общего назначения и окна регистров специального назначения.

В окне управляющей программы отображается несколько строк дисассемблированного кода. В этом окне возможно не только контролировать последовательность выполнения команд программы, но и производить коррекцию команд с помощью встроенного ассемблера. При установке конфигурации основного интерфейса (при нажатии клавиши F10) возможно изменять количество отображаемых строк программы. Перемещение курсора в окне производится с помощью клавиш Up, Down, PgUp и PgDn. Вызов ассемблера для коррекции команд программы осуществляется нажатием клавиш Alt+A, а заканчивается вводом команды Enter и Tab. Отказ от изменения производится с помощью клавиши Esc.

В окне регистров общего назначения (GPR) отображается содержимое этих регистров (R0…R15) в шестнадцатеричной системе счисления после исполнения последней операции или трассировки программы. Перемещение курсоров производится клавишами Up, Down, Home, End, Ctrl+Home, Ctrl+End.

В окне регистров специального назначения (CPU) отображаются состояния этих регистров:

системной конфигурации (SYSCON);

указателей команд (IP, CSP, DPP0…DPP3);

указателей вершины (SP), переполнения (STKOV) и дна (STKUN) стека;

выполнения операций умножения и деления (MDH, MDL, MDC);

состояния АЛУ (PSW).

В окне регистров специального назначения ввод коррекции и перемещение курсора производятся, как и в окне регистров общего назначения.

Окно памяти позволяет просматривать и изменять адресное пространство МК.

Все переходы курсора между окнами осуществляются клавишами Tab и Shift+Tab.

После входа в основной интерфейс полноэкранного отладчика необходимо произвести конфигурацию системы, нажав клавишу F10 и откорректировав все позиции появившегося окна.

Выход из полноэкранного отладчика производится нажатием клавиш Esc и Enter.

Выход из программы-оболочки Norton Commander (NC) и переход в Windows осуществляются нажатием клавиш F10, Enter и набором в командной строке Exit с последующим нажатием клавиши Enter.

Вопросы для самоконтроля

1. В какой системе счисления отображается информация в регистрах полноэкранного отладчика?

2. Что характеризует собой информация в регистре состояния (Flags)?

12. СОСТАВЛЕНИЕ И ВВОД УПРАВЛЯЮЩИХ ПРОГРАММ В ПРОМЫШЛЕННЫЙ КОНТРОЛЛЕР

В пакет программ полноэкранного отладчика входит подкаталог EXAMPLES с примерами управляющих программ для ПК М167-1. Составление новых управляющих программ удобно производить в указанном подкаталоге. Для этого подкаталог EXAMPLES необходимо скопировать из каталога SFD7 в соседнюю панель этого же диска.

Для создания нового файла с управляющей программой в подкаталоге EXAMPLES необходимо:

1) нажать клавиши Shift+F4;

2) в появившемся окне ввести имя вновь создаваемого файла с расширением asm и нажать клавишу Enter (например, EX5.ASM);

3) в появившемся окне набрать управляющую программу на языке Ассемблер с учетом всех выше рассмотренных правил;

4) после окончания создания управляющей программы произвести запись файла в подкаталог на диск нажатием клавиши F2 и выход в NC нажатием клавиши Esc.

Для удаления файла в NC (а это влечет и удаление файла в отладчике) необходимо:

1) установить курсор на удаляемом файле в подкаталоге;

2) нажать клавишу F8;

3) в появившемся окне выделить ячейку Delete;

4) нажать клавишу Enter.

Прежде чем вводить вновь созданный файл в ПК, его необходимо перевести из asm в hex-формат. Для этого необходимо:

1) установить курсор на файле asm167.exe в каталоге SFD7 и нажать клавиши Ctrl+Enter; в командной строке появится информация

C:\Windows.000>asm167.exe

2) установить курсор на вновь созданном файле (в формате asm) и нажать клавиши Ctrl+Enter; в командной строке добавится название нового файла;

3) нажать клавишу Enter; в подкаталоге EXAMPLES помимо вновь созданного файла с расширением asm появится новый файл того же названия, но с расширением hex.

Если после операции перевода созданного файла из asm в hex-формат в подкаталоге EXAMPLES не появился файл в hex-формате, то необходимо вызвать программу определения ошибок. Для этого надо нажать клавиши Ctrl+O (буква O). В открывшемся окне указываются допущенные ошибки в вновь созданной программе. Выход из программы определения ошибок производится повторным нажатием клавиш Ctrl+O.

Для ввода вновь созданной управляющей программы в hex-формате в ПК необходимо:

1) включить полноэкранный отладчик;

2) нажать клавишу F2; на экране появится окно сохранения файла на диске;

3) в появившемся окне установить курсор на EXAMPLES и нажать клавишу Enter;

4) в ячейке Name набрать название вводимого файла;

5) в ячейке File format с помощью кнопки Space выбрать формат файла (hex-формат) и нажать клавишу Enter.

Все остальные данные открывшегося окна устанавливаются системой самостоятельно по умолчанию.

13. ЗАПУСК УПРАВЛЯЮЩИХ ПРОГРАММ

Для запуска управляющих программ в ПК необходимо произвести чтение управляющей программы с диска. Для этого необходимо:

1) в полноэкранном отладчике нажать клавишу F3;

2) в появившемся окне установить курсор на выбранную программу (файл);

3) нажать клавишу Enter; в окне дисассемблера появится выбранная управляющая программа.

Запуск управляющей программы осуществляется в одном из трех режимах:

1) пошаговое исполнение при помощи нажатия клавиши Enter; после каждого нажатия исполняется одна команда программы;

2) исполнение управляющей программы в объеме процедуры при нажатии клавиш Ctrl+Enter; шаг по процедуре отличается от шага по программе исполнением инструкций подпрограмм;

3) автоматическое исполнение управляющей программы при нажатии клавиши F7; в появившемся окне перевести курсор на Goto при помощи клавиши Tab (или ↓) и нажать клавишу Enter. В окне отладчика появится надпись:

Execute program…
Program is running. Press ESC to break.

Исправление или изменение команд в окне дисассемблера производится в следующей последовательности:

1) обозначить строку дисассемблера при помощи клавиш ↑ или ↓;

2) нажать клавиши Alt+A;

3) исправить или изменить данные в строке при помощи клавиш →, ← и Backspace;

4) заканчивается исправление нажатием клавиши Enter;

5) возвращение управляющей программы в исходное положение производится посредством установки в регистр команд (IP) начального адреса программы.

Исполнение управляющей программы может производиться с любой команды программы. Для этого в регистр команд (IP) заносится адрес (из окна дисассемблера) той команды, с которой необходимо запустить программу.

Прерывание исполнения управляющих программ производится нажатием кнопки “Сброс контроллера”.

Остановка выполнения программы производится нажатием клавиш Esc и Enter.

Важной особенностью работы ПК является задание точек останова выполнения программ или количества их повторения. Если точка останова указана в конце программы и задано количество повторений программы, то программа будет повторяться столько раз, сколько было указано. Если точка останова указана внутри программы, то будет повторяться только часть программы, заключенная между началом программы и точкой останова. Если в одной программе указано несколько точек останова, то при запуске программы она останавливается на ближайшей точке останова и для запуска следующего фрагмента программы между текущей точкой останова и следующей точкой необходимо сдвинуть программу на один шаг и вновь запустить программу в автоматическом режиме. Для задания точек останова необходимо:

1) войти в отладчик, нажать F3, активизировать нужную программу, нажать Enter;

2) нажать F9, в появившемся окне выбрать номер (строку) точки останова, указать адрес команды (в ШСС) из окна дисассемблера, на которой должна остановиться программа, и в счетчик (Count) занести цифру, характеризующую количество повторений программы;

3) нажать клавишу Enter;

4) запустить программу в автоматическом режиме (F7, ↓, Enter).

Если при выполнении программы не заданы точки останова (в конце программы), то возможны произвольные многократные повторения программы.

14. УПРАЖНЕНИЯ

Для получения некоторых навыков составления управляющих программ студенту предлагаются уже решенные задачи.

В качестве объекта автоматизации рассматривается сушильная камера, в которой необходимо поддерживать заданную температуру (рис.14.1).

На рис.14.1 сделаны следующие обозначения:

СК – сушильная камера;

НЭ – нагревательный элемент;

РН – регулятор напряжения на нагревательном элементе;

ДТ – датчик температуры;

В1 – вентилятор;

P2.10 – десятый бит порта P2, с которого снимается показание о включенном (P2.10=1) или выключенном (P2.10=0) нагревательном элементе;

P2.8 – восьмой бит порта P2, с которого снимается показание о включенном (P2.8=1) или выключенном (P2.8=0) вентиляторе;

P2.2 и P2.4 – второй и четвертый биты порта P2, с которых снимаются показания предельных значений температуры в сушильной камере; при P2.2=1 температура достигла верхнего предельного значения (ВП); при P2.4=1 температура достигла нижнего предельного значения (НП); при P2.2=0 и P2.4=0 температура находится в заданных допустимых пределах;

P5.6 – четвертый бит порта P5, с которого снимается и подается на аналого-цифровой преобразователь (АЦП) непрерывный сигнал, характеризующий величину температуры в сушильной камере;

P2.1 – первый бит порта P2, на который подается сигнал на включение (логическая единица) или на выключение (логический ноль) нагревательного элемента;

P2.5 – пятый бит порта P2, на который подается сигнал на включение (логическая единица) или на выключение (логический ноль) вентилятора;

P2.0 – нулевой бит порта P2, на который подается сигнал с канала широтно-импульсной модуляции (ШИМ) для управления тиристором, который регулирует величину электрического тока, протекающего через нагревательный элемент.

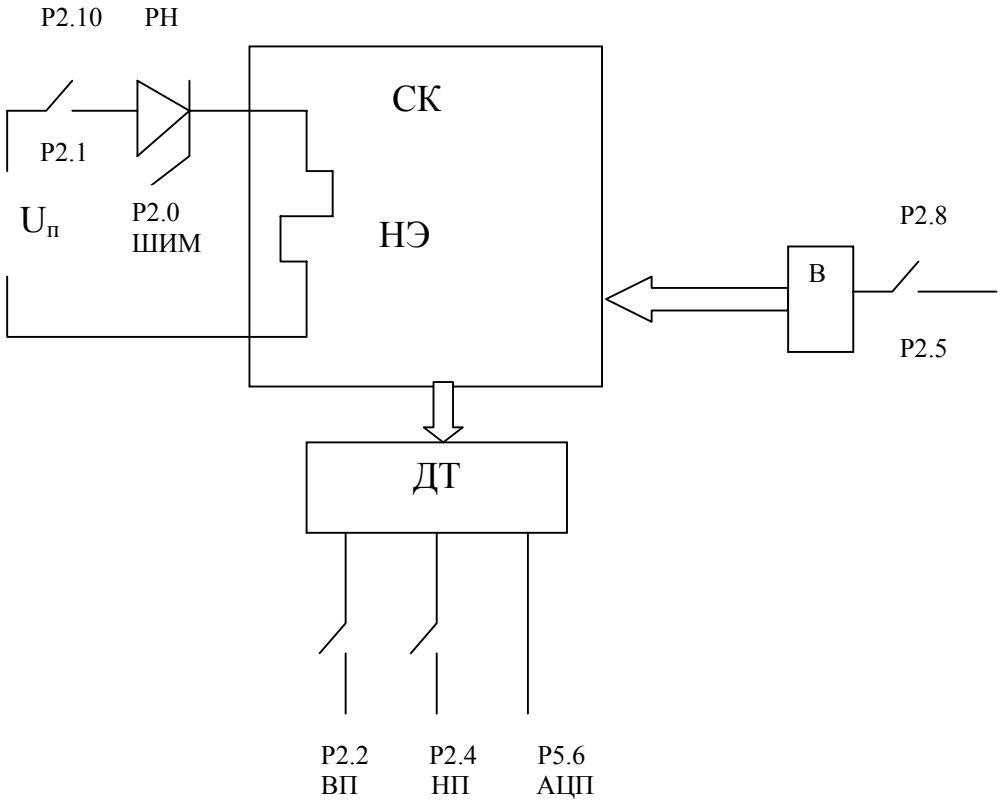


Рис.14.1

Задача № 1

Подключить нагревательный элемент к напряжению питания.

Программа

BSET P2.1; запись в регистр данных порта P2.1 единицы для ее вывода из порта (включить)

BSET DP2.1; запись единицы в регистр направления (вывод)

Задача № 2

Если не включен вентилятор, то отключить нагревательный элемент от напряжения питания.

Граф-схема алгоритма программы представлена на рис.14.2.

По граф-схеме алгоритма составляется программа.

Программа

W: BCLR DP2.8; запись нуля в регистр направления (ввод)

BMOV R1.1,P2.8; пересылка содержимого порта P2.8 в R1.1

JNB R1.1,W; условный переход: если в первом бите порта R1 нет единицы, то перейти по метке

BCLR P2.1; запись в регистр данных порта P2.1 нуля для вывода его из порта (отключение)

BSET DP2.1; запись в регистр направления единицы (вывод)

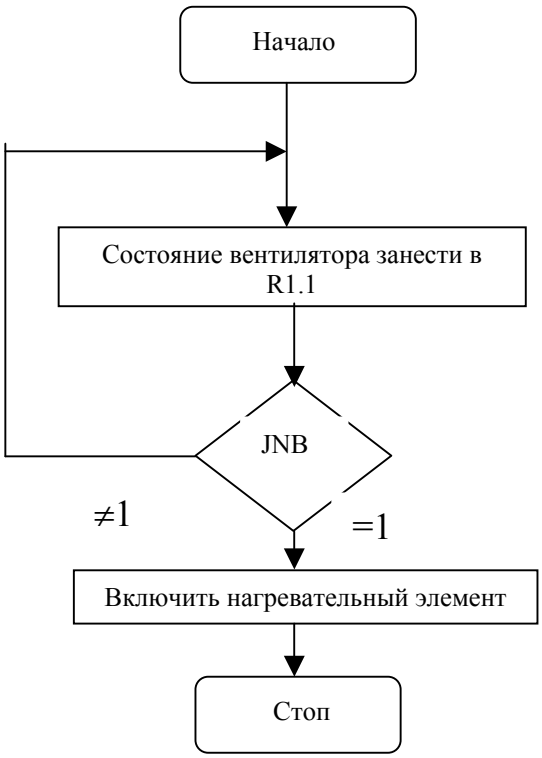


Рис. 14.2

Задача № 3

Если включен вентилятор и подано напряжение питания на нагревательный элемент, то установить напряжение на нагревательном элементе за счет широтно-импульсной модуляции, соответствующее индексу модуляции 0,5 при частоте следования импульсов 100 Гц.

Граф-схема алгоритма программы представлена на рис. 14.3.

По граф-схеме алгоритма составляется программа.

Программа

```
Q:   BCLR    DP2.10; запись нуля в регистр направления (ввод)
     BMOV    R1.1,P2.10; пересылка содержимого порта P2.10 в R1.1
     BCLR    DP2.8; запись нуля в регистр направления (ввод)
     BMOV    R2.1,P2.8; пересылка содержимого порта P2.8 в R2.1
     BAND    R1.1,R2.1; логическое умножение битов регистров R1.1 и
R2.1
     JMPR    CC_Z, Q; условный переход: если результат логического
умножения равен нулю, то перейти по метке (повторить контроль).
```

Расчет кода переполнения.

По формуле

$$k_d \geq \frac{308{,}18}{f_{ВЫХ}}$$

находится коэффициент деления

$$k_d \geq \frac{308\ ,18}{100} \geq 3{,}0818\ .$$

Значение коэффициента деления принимается k_d=8.

Код переполнения в десятичной системе счисления будет равен

$$B = 2^{16} - \frac{f_{CPU}}{k_d \cdot f_{ВЫХ}} = 65535 - \frac{20 \cdot 10^6}{8 \cdot 100} = 40535.$$

В шестнадцатеричной системе счисления код переполнения примет вид

B=9E57h.

Код индекса модуляции определится как

$$M = \frac{f_{CPU}}{k_d \cdot f_{ВЫХ}} \cdot (1 - I_m) = \frac{20 \cdot 10^6}{8 \cdot 100} \cdot 0{,}5 = 12500.$$

В шестнадцатеричной системе счисления код индекса модуляции будет равен M=30D4h.

Рис. 14.3

Занесение кода переполнения и кода индекса модуляции в регистр переполнения и регистр сравнения.

```
MOV     R3,#9E57h;
MOV     R4,#30D4h;
PUSH    R3;
PUSH    R4;
POP     CC0;
```

POP T0REL;

Заполнение битовых полей регистра режима.

CCM0=111,

ACC0=0.

В этом примере все битовые поля остальных каналов ШИМ (1, 2 и 3) заполняются нулями. Тогда код режима работы будет иметь вид 0007h. Полученный код заносится в регистр режима

MOV CCM0,#0007h;

Заполняются битовые поля регистра управления специальным таймером (см. пример 7.1). Разница состоит в том, что в примере 7.1 коэффициент деления был 512, а в данном примере этот коэффициент равен 8. Поэтому код примет значение 0040h. Полученный код управления записывается в регистр управления специальным таймером

MOV T01CON,#0040h;

Задается направление работы порта

BSET DP2.0;

Задача № 4

Если температура в сушильной камере превысила 64^0 С, то включить вентилятор.

Граф-схема алгоритма программы представлена на рис. 14.4.

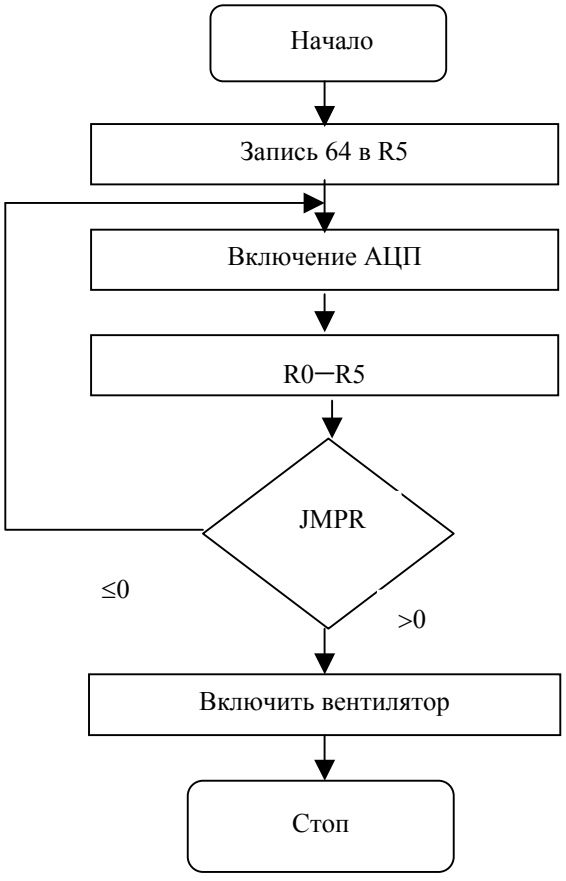


Рис. 14.4

По граф-схеме алгоритма составляется программа.

Число 64 в шестнадцатеричной системе счисления выразится как 0040h.

Программа

```
        MOV     R5,#0040h; запись в R5 числа 64
        MOV     R10,#0005h; занесение номера линии 5 порта P5 в регистр
        конфигурации АЦП
        PUSH    R10;
        POP     ADCON;
Q:      BSET    ADST; включение АЦП
W:      NOP;
        JB      ADBSY,W; ожидание окончания преобразования
        PUSH    ADDAT; помещение результата в R0
        POP     R0;
        AND     R0,#0FFFh;
        CMP     R0,R5; сравнение температур
        JMPR    CC_ULE,Q; если разность R0−R5 меньше или равна нулю, то
        перейти по метке
        BCLR    P2.5; занесение нуля в бит 5 порта P2 (вылючить)
        BSET    DP2.5; запись в регистр направления единицы (вывод)
```

Задача № 5

Если температура в покрасочной камере либо превысила верхний допустимый предел, либо стала меньше нижнего допустимого предела, то с промежутком времени в 10 с включить вентилятор.

Граф-схема алгоритма программы представлена на рис. 14.5.

По граф-схеме алгоритма составляется программа.

Программа

```
ST:     BCLR    DP2.2; запись нуля в регистр направления (ввод)
        BMOV    R3.1,P2.2; пересылка содержимого порта P2.2 в R3.1
        BCLR    DP2.4; запись нуля в регистр направления (ввод)
        BMOV    R2.2,P2.4; пересылка содержимого порта P2.4 в R2.2
        BOR     R3.1,R2.2; логическое сложение битов регистров R3.1 и R2.2
        JMPR    CC_Z,ST; условный переход: если результат логического
        сложения равен нулю, то перейти по метке (повторить контроль)
        MOV     R12,#000Ah;
```

Расчет кода выдержки времени. Предварительно определяется коэффициент деления из условия

$$k_d \geq 308{,}18 \cdot T .$$

При T=1с условие примет вид k_d≥308,18. Ближайшее большее значение коэффициента деления равно k_d=512. Тогда в десятичной системе счисления код выдержки времени будет равен

$$A = 2^{16} - \frac{T \cdot f_{CPU}}{k_d} = 65535 - \frac{1 \cdot 20 \cdot 10^6}{512} = 65535 - 39063 = 26472 .$$

В двоичной системе счисления код выдержки времени примет вид

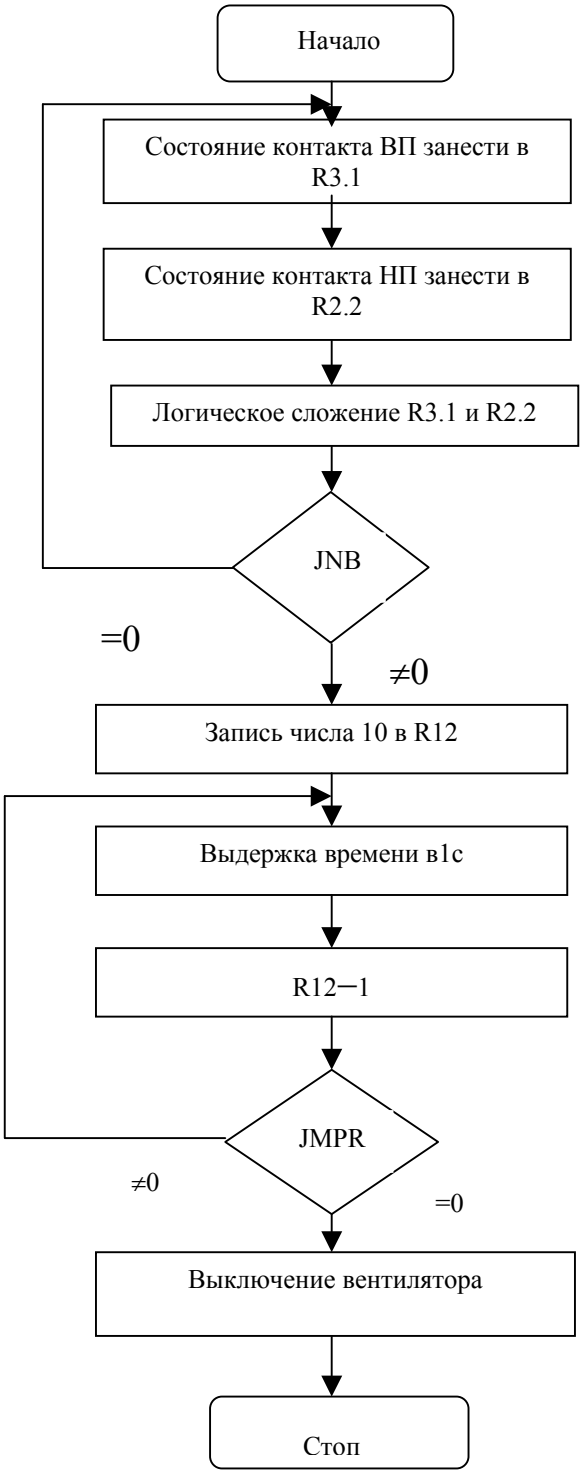


Рис. 14.5

37768 16384 8192 4096 2048 1024 512 256 128 64 32 16 8 4 2 1
0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 0

В шестнадцатеричной системе счисления код выдержки времени примет вид

A=6768h.

Код выдержки времени заносится в один из регистров общего назначения. Регистр общего назначения выбирается произвольно.

```
KL:  MOV     R7,#6768h;
```

Производится пересылка содержимого R7 в регистр данных через стек. Рабочим таймером выбирается T2.

```
PUSH    R7;
POP     T2;
```

Заполняются битовые поля регистра управления T2CON в соответствии с выбранным режимом работы таймера, и формируется код, который должен быть занесен в регистр управления.

T2I=110 – коэффициент деления в делителе частоты 512;

T2M=000 – таймер работает от внутреннего генератора;

T2R=1 – таймер включен;

T2UD=0, T2UDE=0 – суммирование в счетчике тактовых импульсов;

T2OE=0 – выход переполнения отключен;

T2OTL=0 – счетчик не переполнен.

В соответствии с данным режимом работы таймера в регистр управления необходимо записать код в двоичной системе счисления вида

0000 0000 0100 0110

или в шестнадцатеричной системе счисления

0046h.

Код режима работы таймера заносится в регистр управления.

```
MOV     T2CON,#0046h;
```

Установка программы на ожидание окончания выдержки времени (появление единицы в T2IR).

```
W:  NOP; пустой шаг − некоторая задержка времени в выполнении программы
    JNB   T2IR, W; условный переход: если в T2IR нет единицы, то перейти по метке
    MOV     T2CON,#0000h; выключение таймера
    BCLR    T2IR; установка триггера T2IR в ноль
    SUB     R12,#0001h; вычесть из числа 10 единицу
    JMPR    CC_NZ,KL; если не выполнены все 10 циклов, то перейти по метке
    BCLR    P2.5; запись нуля в регистр данных (вылючение)
    BSET    DP2.5; запись единицы в регистр направления (вывод)
```

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Бродин В.Б., Шагурин М.И. Микроконтроллеры, архитектура, программирование, интерфейс: Справочник. М.: Изд-во ЭКОМ, 1999.398 с.
2. Ремизевич Т.В. Микроконтроллеры для встраиваемых приложений: от общих подходов − к семействам HC05 и HC08 фирмы Motorola. М.: Изд-во ДОДЭКА, 2000.272 с.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

ПРИЛОЖЕНИЕ 1
ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ МК ТИПА 80С16х ФИРМЫ SIEMENS

Характеристики МК	Тип МК						
	MO161-x	CS166-104	M164-x	M167-1-CT	M167-1-x	M167-2	167-3U
Тип ЦП	C1610	C166	C164	C167	C167	C167	C167
Время выполнения команды (нс)	250	200	100	100	100	100	100
Объем ОЗУ (Кбайт)	128	128	256	256	256	256	256
Объем ПЗУ (Кбайт)	128	128	256	256	256	256	256
Время выполнения команды умножения (нс) 16х16 с результатом 32 бит	625	625	500	500	500	500	500
Количество уровней прерывания	20	32	32	56	56	56	56
Каналы АЦП: количество время преобразования (мкс) входное напряжение (В)	– – –	10 – +5	8 10 +5	16 10 +5	16 10 +5	16 10 +5	16 10 +5
Общее количество линий портов ввода/вывода	23	46	34	63	63	74	88
Количество универсальных таймеров	5	5	7	9	9	9	9
Последовательный канал RS232	1	2	1	1	1	1	1
CAN-контроллер	–	–	+	–	–	+	+
Супервизор питания	+	+	+	+	+	+	+
Часовой таймер	–	+	–	+	–	+	+
Программатор	+	+	+	+	+	+	+
Интерактивная отладка на плате программ через RS232	+	+	+	+	+	+	+
Напряжение питания (В)	+5	+5	+5	+5	+5	+5	+5
Потребляемая мощность (Вт)	0,1	0,4	1,15	0,4	0,5	1,0	2,0

ПРИЛОЖЕНИЕ 2

ПОРТЫ МК ТИПА 80С167 ФИРМЫ SIEMENS

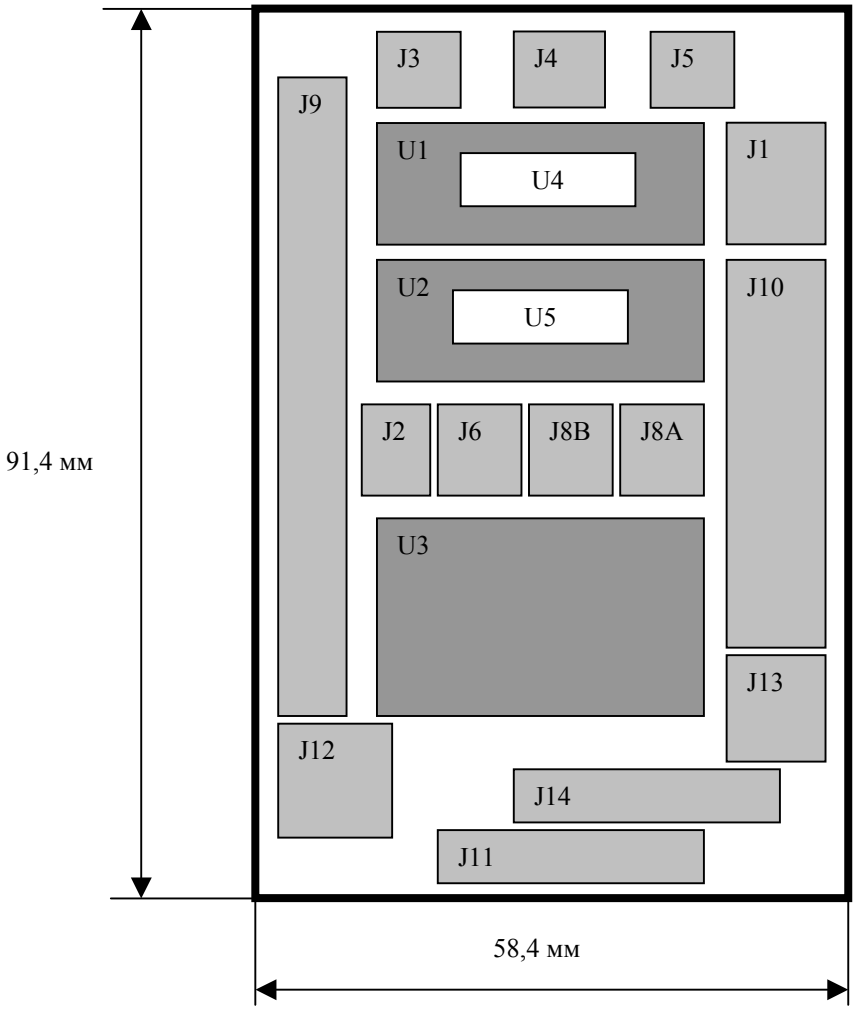
Наименование порта	Назначение битов порта	Обозначение битов порта	Вводимая или выводимая информация
Шина данных P0	Ввод/вывод команд и массивов данных	P0.0	AD0
		P0.1	AD1
		P0.2	AD2
		P0.3	AD3
		P0.4	AD4
		P0.5	AD5
		P0.6	AD6
		P0.7	AD7
		P0.8	AD8
		P0.9	AD9
		P0.10	AD10
		P0.11	AD11
		P0.12	AD12
		P0.13	AD13
		P0.14	AD14
		P0.15	AD15
Шина адреса P1	Вывод адресов ячеек памяти	P1.0	A0
		P1.1	A1
		P1.2	A2
		P1.3	A3
		P1.4	A4
		P1.5	A5
		P1.6	A6
		P1.7	A7
		P1.8	A8
		P1.9	A9
		P1.10	A10
		P1.11	A11
	Вывод адресов ячеек памяти и выход блока 2 захват/сравнение	P1.12	A12/CC24
		P1.13	A13/CC25
		P1.14	A14/CC26
		P1.15	A15/CC27

Наименование порта	Назначение битов порта	Обозначение битов порта	Вводимая или выводимая информация
Порт ввода/вывода цифровой информации P2	или входы/выходы блока 1 захват/сравнение	P2.0	CC0IO
		P2.1	CC1IO
		P2.2	CC2IO
		P2.3	CC3IO
		P2.4	CC4IO
		P2.5	CC5IO
		P2.6	CC6IO
		P2.7	CC7IO
	или входы прерываний или входы/выходы блока 1 захват/сравнение	P2.8	CC8IO/EX0IN
		P2.9	CC9IO/EX1IN
		P2.10	CC10IO/EX2IN
		P2.11	CC11IO/EX3IN
		P2.12	CC12IO/EX4IN
		P2.13	CC13IO/EX5IN
		P2.14	CC14IO/EX6IN
		P2.15	CC15IO/EX7IN
Порт ввода/вывода цифровой информации P3	или входы/выходы блока 1 таймеров	P3.0	T0IN
		P3.1	T6OUT
		P3.2	CAPIN
		P3.3	T3OUT
		P3.4	T3EUD
		P3.5	T4IN
		P3.6	T3IN
		P3.7	T2IN
	или синхронный последовательный порт	P3.8	MRST
		P3.9	MRST
	или асинхронный последовательный порт	P3.10	не используется
		P3.11	не используется
		P3.12	не используется
		P3.13	
		P3.14	не используется
		P3.15	

Наименование порта	Назначение битов порта	Обозначение битов порта	Вводимая или выводимая информация
Порт ввода информации P5	или входы АЦП	P5.0	AN0
		P5.1	AN1
		P5.2	AN2
		P5.3	AN3
		P5.4	AN4
		P5.5	AN5
		P5.6	AN6
		P5.7	AN7
		P5.8	AN8
		P5.9	AN9
	или входы блока 1 таймеров	P5.10	AN10/T6EU
		P5.11	AN11/T5EU
		P5.12	AN12/T6IN
		P5.13	AN13/T5IN
		P5.14	AN14/T4EU
		P5.15	AN15/T2EU
Порт ввода/вывода информации P7	или выходы каналов ШИМ	P7.0	POUT0
		P7.1	POUT1
		P7.2	POUT2
		P7.3	POUT3
	или входы/выходы блока 2 захват/сравнение	P7.4	CC28IO
		P7.5	CC29IO
		P7.6	CC30IO
		P7.7	CC31IO
Порт ввода/вывода информации P8	или входы/выходы блока 2 захват/сравнение	P8.0	CC16IO
		P8.1	CC17IO
		P8.2	CC18IO
		P8.3	CC19IO
		P8.4	CC20IO
		P8.5	CC21IO
		P8.6	CC22IO
		P8.7	CC23IO

ПРИЛОЖЕНИЕ 3
ГАБАРИТНЫЕ РАЗМЕРЫ, ПРИЕМНЫЕ ЧАСТИ РАЗЪЕМОВ И ОСНОВНЫЕ ИНТЕГРАЛЬНЫЕ СХЕМЫ ПК М167-1

На приведенном рисунке указана часть интегральных схем, расположенных на плате ПК типа М167-1 и приемные части разъемов и переключателей.



Интегральные схемы:

U1 и U2 – интегральные схемы внешних постоянных запоминающих устройств (FLASH) объемом до 512 Кбайт;

U3 – интегральная схема МК типа 80С167;

U4 и U5 – интегральные схемы внешних оперативных запоминающих устройств объемом до 256 Кбайт.

Приемные части разъемов:

J1 – разъем предназначен для подключения кабеля RS232 к последовательному порту персонального компьютера;

J2 – переключатель предназначен для выбора режима работы ПК:

-перемычка установлена – отладочный режим работы;

-перемычка удалена – режим старта программы из ПЗУ;

J3 – разъем предназначен для подключения кнопки сброса контроллера;

J4 – разъем предназначен для подключения батареи супервизора;

J5 – разъем предназначен для подачи напряжения +12 вольт в режиме программирования микросхемы ПЗУ;

J6 – переключатель предназначен для установки конфигурации интегральных схем памяти с различным числом выводов;

J8A и J8B – переключатели для установки различных типов памяти;

J9 – разъем предназначен для подключения питания и внешних периферийных устройств к ПК по шинам данных и адреса;

J10 – разъем предназначен для подключения внешних периферийных устройств к ПК через порты P2, P3 и P6;

J11 – разъем предназначен для подключения внешних периферийных устройств к ПК через порты P7 и P8;

J12 – разъем предназначен для подключения ПК локальной вычислительной сети;

J13 – переключатель предназначен для выбора опорного напряжения АЦП;

J14 – разъем используется для подключения внешних периферийных устройств к ПК через порт P5.

ОГЛАВЛЕНИЕ

Олег Алексеевич Готшальк

Промышленные контроллеры

Микропроцессорные системы энергетических объектов

Письменные лекции

Редактор А.В. Алехина
Сводный темпплан 2003г.
Лицензия ЛР № 020308 от 14.02.97.

Подписано в печать 09, 2003. Формат 60x84 1/16
Б. кн.-журн. П.л. . Б.л. . РТП РИО СЗТУ
Тираж . Заказ .

Северо-Западный государственный заочный технический университет
РИО СЗТУ, член Издательско-полиграфической ассоциации
вузов Санкт-Петербурга
191186, Санкт-Петербург, ул. Миллионная, 5