БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ РАДИОФИЗИКИ И ЭЛЕКТРОНИКИ
Кафедра кибернетики

МИКРОКОНТРОЛЛЕРЫ В СИСТЕМАХ УПРАВЛЕНИЯ И ИЗМЕРЕНИЯ

Методические указания
к лабораторному практикуму

В двух частях
Часть 1

Для студентов специальностей:
G 31 04 02 «Радиофизика»,
G 31 04 03 «Физическая электроника»,
E 25 01 10 «Коммерческая деятельность на рынке радиоэлектронных средств и информационных услуг»

МИНСК
2002

УДК 536.3(075.83)
ББК 22.34р30-252.43
И85

А в т о р – с о с т а в и т е л ь :
И. А. Шалатонин, ст. преподаватель;

Р е ц е н з е н т
доктор физико-математических наук,
профессор *В. В. Апанасович*

Утверждено Ученым советом
факультета радиофизики и электроники
28 мая 2002 г., протокол № 12

Микроконтроллеры в системах управления и измерения: Метод. указания к лабораторному практикуму / Авт.-сост. И. А. Шалатонин. – Мн.: БГУ, 2002. – 94 с.
И85

Методические указания предназначены для студентов 3-5-го курсов факультета радиофизики и электроники.

УДК 536.3(075.83)
ББК 22.34р30-252.43

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ О МИКРОКОНТРОЛЛЕРАХ

МИКРОКОНТРОЛЛЕР (МК) – это микропроцессорное устройство, специализированное на выполнении определенных функций управления, регулирования, идентификации. От обычного микропроцессора он отличается наличием встречных таймеров, счетчиков, ПЗУ, ОЗУ, схем сравнения, аналого-цифрового преобразования, последовательной связи и т. д. Выпускается множество микроконтроллеров, различающихся разрядностью, системой команд, объемом ПЗУ и ОЗУ, количеством встроенных функций.

1.1. АРХИТЕКТУРА 8-РАЗРЯДНЫХ МИКРОКОНТРОЛЛЕРОВ

1.1.1. Модульный принцип построения

МК представляют собой законченную МП-систему обработки информации, которая реализована в виде одной большой интегральной микросхемы. МК объединяет в пределах одного полупроводникового кристалла основные функциональные блоки МП управляющей системы: центральный процессор(процессорное ядро), постоянное запоминающее устройство (ПЗУ), оперативное запоминающее устройство (ОЗУ), периферийные устройства для ввода и вывода информации.

Широкое разнообразие моделей МК, возможность разработки и производства новых моделей в короткие сроки обеспечивает модульный принцип построения МК, который взят на вооружение всеми ведущими компаниями. При модульном принципе построения все МК одного семейства содержат в себе базовый функциональный блок, который одинаков для всех МК семейства, и изменяемый функциональный блок, который отличает МК разных моделей в пределах одного семейства (рис. 1).

Базовый функциональный блок включает:

- центральный процессор;
- внутренние магистрали адреса, данных и управления;
- схему формирования многофазной импульсной последовательности для тактирования ЦП и межмодульных магистралей;
- устройство управления режимами работы МК.

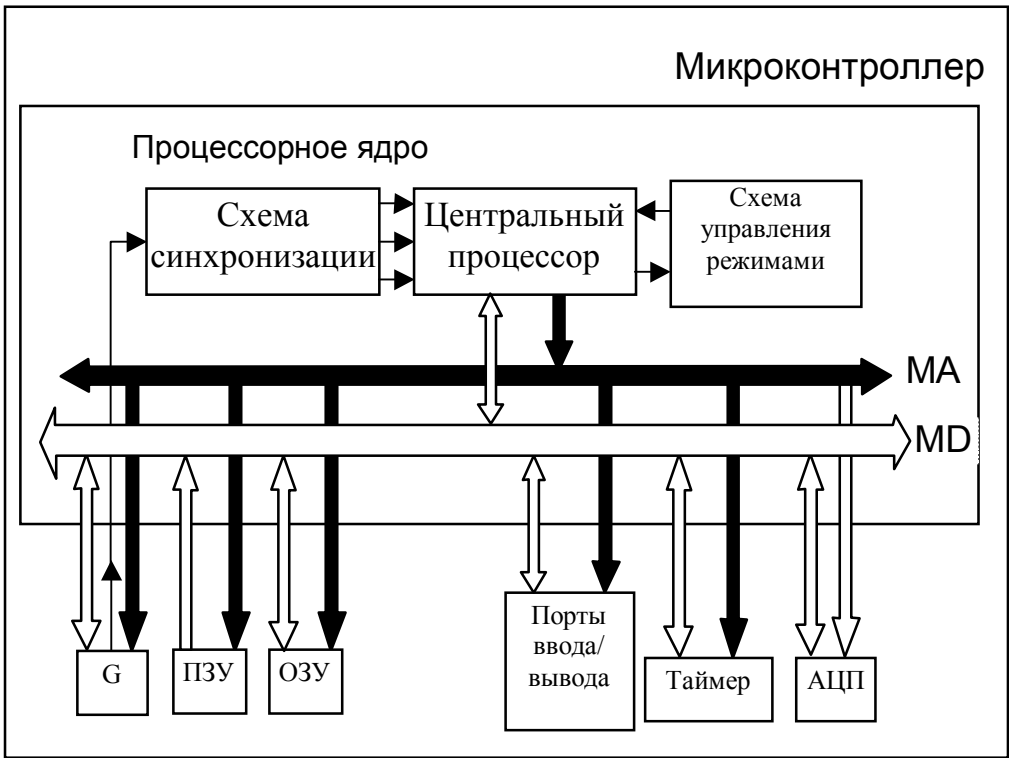

Рис. 1. Модульная организация МК

Базовый функциональный блок принято называть процессорным ядром МК. Процессорное ядро обозначают именем семейства МК, основой которого оно является. Например, ядро HC05 – процессорное ядро семейства Motorola MC68HC05, ядро MCS51 – ядро семейства МК Intel 8xC51, ядро PIC16 – процессорное ядро Microchip PIC16.

Изменяемый функциональный блок включает модули различных типов памяти, модули периферийных устройств, модули генераторов синхронизации и некоторые дополнительные модули специальных режимов работы МК.

1.1.2. Процессорное ядро МК

Ядро современных 8-разрядных МК реализует один из двух принципов построения МП:

- МП с CISC-архитектурой – МП с полной системой команд (Complicated Instruction Set Computer).
- МП с RISC-архитектурой – МП с сокращенной системой команд (Reduced Instruction Set Computer).

Производительность МП, и МК в том числе, принято оценивать числом операций пересылки «регистр-регистр», которые могут быть

выполнены в течение одной секунды. Для МК с RISC-архитектурой время выполнения любой операции составляет $1/f_{BUS}$. Следовательно, их производительность равна f_{BUS} оп/с. Например, производительность PIC16 составляет 5 млн оп/с, Scenix – 25 млн оп/с. В МК с CISC-архитектурой число циклов выполнения операции «регистр-регистр» составляет от 1 до 3, что снижает производительность.

1.1.3. Резидентная память МК

Закрытая архитектура современных 8-разрядных МК стала реализуемой лишь при условии интеграции на кристалл МК модулей памяти двух типов: энергонезависимого запоминающего устройства для хранения кодов прикладных программ (ПЗУ) и оперативного запоминающего устройства для хранения промежуточных результатов вычислений (ОЗУ). С момента появления МК технология энергонезависимых запоминающих устройств претерпела множество изменений, которые позволили не только повысить информационную емкость, быстродействие, надежность хранения информации, но и привели к появлению принципиально новых технологий программирования резидентной памяти МК. С точки зрения пользователей МК следует различать шесть типов энергонезависимой резидентной памяти:

- ПЗУ масочного типа – mask-ROM. Содержимое ячеек ПЗУ этого типа записывается на заводе-изготовителе МК с помощью масок и не может быть заменено или «допрограммировано» в области ранее не использованного сегмента памяти.
- ПЗУ, однократно программируемые пользователем – OTPROM (One-Time Programmable ROM). В незапрограммированном состоянии каждая ячейка памяти модуля однократно программируемого ПЗУ при считывании возвращает код $FF. Программированию подлежат только те разряды, которые после программирования должны содержать 0.
- ПЗУ, программируемые пользователем с ультрафиолетовым стиранием – EPROM (Erasable Programmable ROM). ПЗУ данного типа допускают многократное программирование. Технология программирования близка к технологии однократно программируемых ПЗУ.
- ПЗУ, программируемые пользователем, с электрическим стиранием – EEPROM (Electrically Erasable Programmable ROM). Электрически программируемые и электрически стираемые ПЗУ

совместили в себе три положительных качества рассмотренных выше типов памяти: они программируются пользователем, могут быть многократно подвергнуты операции стирания и дешевле ПЗУ с ультрафиолетовым стиранием.

- ПЗУ с электрическим стиранием типа FLASH – FLASH ROM. ПЗУ типа FLASH сохранили преимущества, присущие EEPROM: возможность многократного стирания и программирования посредством приложения повышенного напряжения. Однако память типа FLASH стирается и программируется страницами или блоками.

Кроме ПЗУ, в состав МК входит также и статическое оперативное запоминающее устройство.

1.1.4. Порты ввода/вывода

Каждый МК имеет некоторое количество линий ввода/вывода, которые объединены в 8-разрядные параллельные порты ввода/вывода PTx («x» – имя порта, используемое в техническом описании). В карте памяти МК каждый порт ввода/вывода представлен регистром данных порта OPTx. В режиме ввода логические уровни сигналов на линиях порта PTx отображаются нулями и единицами в соответствующих разрядах регистра DPTx. В режиме вывода данные, записанные под управлением программы в регистр DPTx, передаются на выводы МК, которые отмечены в качестве линий порта PTx. Обращение к регистру данных DPTx осуществляется теми же командами, что и обращение к ячейкам оперативной памяти. Кроме того, во многих МК отдельные разряды портов могут быть опрошены командами битового процессора.

1.1.5. Таймеры и процессоры событий

Большинство задач управления, которые возлагаются на МП-системы, должны выполняться в реальном времени. Понятие «управление в реальном времени» означает способность МП-системы получить информацию о состоянии управляемого объекта, выполнить необходимые расчеты и сформировать управляющие воздействия в течение интервала времени, по истечении которого эти воздействия вызовут желаемое изменение поведения объекта. Возможность использования того или иного МК для управления конкретным устройством в реальном времени определяется в первую очередь производительностью процессорного ядра, т. к. МК должен успеть за

строго ограниченное время выполнить расчет корректирующего воздействия. Однако только высокой производительности недостаточно. Необходимо организовать прием информации с датчиков и выдачу управляющих сигналов таким образом, чтобы при сохранении требуемой точности на эти операции расходовалось как можно меньше времени. В противном случае не останется времени для выполнения вычислений. Эффективное распределение задач управления между различными модулями МК обеспечивает возможность качественного управления в реальном времени. Многие подсистемы МК используются для решения этих задач, но в первую очередь среди них выделяют подсистему прерываний и модуль таймера. Развитая подсистема прерываний позволяет сократить время реакции МП-системы на изменения состояния объекта. Модули таймеров служат для приема информации от датчиков с времяимпульсными выходами, а также для формирования управляющих воздействий в виде последовательности импульсов с изменяющимися параметрами.

Модуль таймера 8-разрядного МК представляет собой 16-разрядный счетчик со схемой управления. В карте памяти МК счетчик отображается двумя регистрами: TH – старший байт счетчика, TL – младший байт. Регистры доступны для чтения и для записи. Направление счета счетчика – только прямое, т. е. при поступлении тактовых импульсов десятичный эквивалент двоичного кода счетчика изменяется в сторону увеличения.

1.1.6. Аналого-цифровые и цифро-аналоговые преобразователи

Отличительная особенность многих современных 8-разрядных МК – интегрированный на кристалл МК модуль многоканального аналого-цифрового преобразователя (АЦП). Модуль АЦП предназначен для ввода в МК аналоговых сигналов с датчиков физических величин и преобразования этих сигналов в двоичный код с целью последующей программной обработки. Многоканальный аналоговый коммутатор служит для подключения одного из источников аналоговых сигналов (PTx0... PTx7) к входу АЦП. Выбор источника сигнала для измерения осуществляется посредством записи номера канала коммутатора в соответствующие разряды регистра управления АЦП. Заметим, что в модулях АЦП 8-разрядных МК предусмотрена только программная установка номера канала, режим

автоматического последовательного сканирования каналов с записью результата измерения каждого канала в индивидуальную ячейку памяти не реализуется.

Цифро-аналоговые преобразователи в составе МК являются большой редкостью. Модули параллельных ЦАП можно встретить лишь в МК фирм «Mitsubishi» и «Hitachi».

1.1.7. Контроллеры последовательного ввода/вывода

Наличие в составе 8-разрядного МК модуля контроллера последовательного ввода/вывода стало настолько обычным явлением, что лишь самые простые, маловыводные МК в корпусах DIP-16 и DIP-20 не имеют портов последовательного обмена. Задачи, которые решаются средствами модуля контроллера последовательного ввода/вывода, могут быть условно разделены на три группы:

- Связь встраиваемой МП-системы с системой управления верхнего уровня: промышленным компьютером, программируемым контроллером, офисным компьютером. Наиболее часто для этих целей используются интерфейсы RS-232C и RS-485.
- Связь с внешними по отношению к МК периферийными ИС встраиваемой МП-системы, а также с датчиками физических величин с последовательным выходом. Для этих целей используются интерфейсы SPI, I^2C, а также нестандартные протоколы обмена.
- Интерфейс связи с локальной сетью в мультимикропроцессорных системах. В системах с числом МК до пяти обычно используют сети на основе интерфейсов I^2C, RS-232C, RS-485 с собственными сетевыми протоколами верхнего уровня. В белее сложных системах популярен протокол CAN.

Среди не очень большого множества различных типов встроенных контроллеров последовательного обмена, которые входят в состав тех или иных 8-разрядных МК, сложился стандарт «де-факто» – модуль UART (Universal Asynchronous Receiver and Transmitter). В переводе с английского UART – универсальный асинхронный приемопередатчик. Однако большинство модулей UART, кроме асинхронного режима обмена, способны также реализовать режим синхронной передачи данных.

МК фирмы «Motorola» традиционно имеют в своем составе два модуля последовательного обмена: модуль SCI с возможностью

реализации только протоколов асинхронной приемопередачи для интерфейсов RS-232C, RS-422A, RS-485 и модуль контроллера синхронного интерфейса в стандарте SPI.

1.1.8. Аппаратные и программные решения по повышению надежности работы МК

В отличие от персональных компьютеров, которые при стечении определенных обстоятельств могут выполнить нештатные операции и «зависнуть», встраиваемые МП-системы управления лишены этого недостатка. Прикладная программа управления, записанная в память МК, должна обеспечивать формирование адекватных выходных воздействий при любых комбинациях входных сигналов. Однако в результате электромагнитных помех предусмотренный разработчиком ход выполнения прикладной программы может быть нарушен. Именно в этом случае невозможно гарантировать правильную работу МП-системы. Все современные МК предусматривают возможность восстановления правильного хода вычислительного процесса при подобного рода отказах. Для этого используется модуль сторожевого таймера (COP или Watchdog). Основным элементом модуля сторожевого таймера является многоразрядный счетчик. При сбросе МК счетчик обнуляется. После перехода МК в активный режим работы счетчик начинает увеличивать код независимо от выполняемой программы. Если код счетчика достигает максимального указанного в техническом описании МК значения, то генерируется сигнал внутреннего сброса и МК начинает выполнение программы управления сначала. Для исключения события сброса по переполнению сторожевого таймера прикладная программа управления должна периодически сбрасывать счетчик. Операция сброса счетчика сторожевого таймера обычно выполняется посредством записи указанного кода в один из регистров специальных функций. Тогда при нормальном, предусмотренном разработчиком порядке выполнения прикладной программы переполнение счетчика сторожевого таймера не наступает и он не оказывает влияния на работу МП-системы. Однако если ход выполнения прикладной программы был нарушен, то велика вероятность, что счетчик не будет сброшен вовремя. Тогда произойдет сброс по переполнению сторожевого таймера и нормальный ход вычислительного процесса будет восстановлен.

1.2. МИКРОКОНТРОЛЛЕРЫ ФИРМЫ «MOTOROLA»

Фирма «Motorola» выпускает семейство однокристальных контроллеров основанных на базовой конфигурации микропроцессоров MC6800 и MC68000. К этому семейству относятся семейство контроллеров M68HC11, M68300 и другие (см. рис. 2).

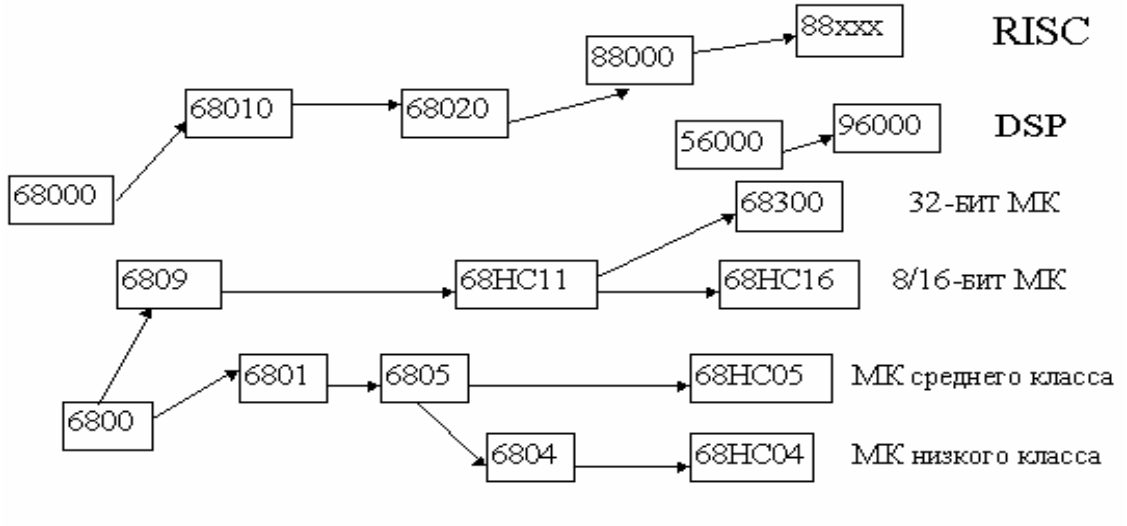

Рис. 2. Эволюция микроконтроллеров фирмы «Motorola»

Место изучаемых микроконтроллеров в структуре микропроцессорной продукции фирмы «Motorola» характеризует рис. 3.

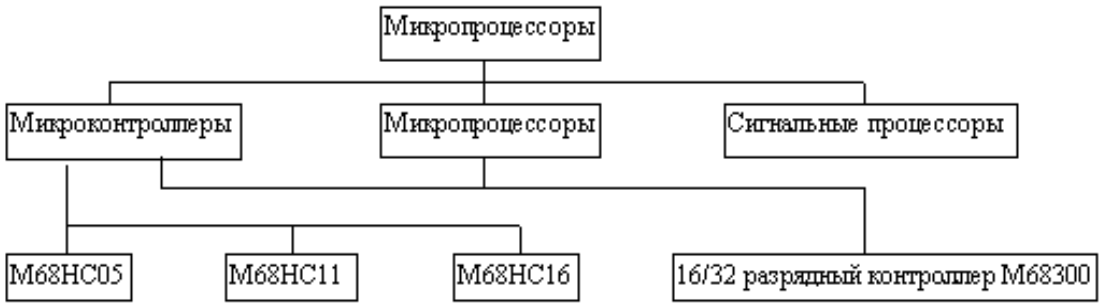

Рис. 3. Структура микропроцессорной продукции фирмы «Motorola»

1.2.1. Обзор 8-разрядных МК фирмы «Motorola»

Данный лабораторный практикум предназначен для изучения схемотехники и получения навыков по отладке программного обеспечения микроконтроллеров MC68HC711E9, относящейся к семейству MC68HC11 фирмы «Motorola».

Фирма «Motorola» является общепризнанным мировым лидером в производстве как электронных устройств вообще (свыше 50000 наименований от транзисторов до RISC-процессоров), так и микроконтроллеров (более 300 моделей, от простых и дешевых 8-разрядных до высокопроизводительных 32-разрядных с поддержкой разнообразных периферийных устройств). Вместе с тем конкурентная борьба фирм-производителей полупроводниковых изделий приводит к необходимости специализации фирм по областям применения, и фирма «Motorola» заняла главенствующее положение на рынке МК для встраиваемых применений

Компания «Motorola» предлагает разработчикам электронной аппаратуры три семейства 8-разрядных микроконтроллеров: HC05, HC11, HC08.

Компанией «Motorola» был разработан свой собственный стандарт 8-разрядных микроконтроллеров, его отличительные особенности:

- CISC архитектура процессорного ядра, которая предлагает развитую систему команд, наличие различных способов адресации.
- Предельно простая программно-логическая модель процессорного ядра. Число регистров центрального процессора HC11 – 7, в HC08, HC05 реализована модель с минимальным набором регистров – 5.
- Единое адресное пространство памяти программ, данных, регистров специальных функций периферийных модулей. Такое решение позволяет выполнять программный код, который размещен не только в ПЗУ программ, но и в ОЗУ данных.
- Размещение в адресном пространстве, не принадлежащем программам пользователя, сервисных программ-мониторов загрузки и мониторов внутрисхемной отладки позволяет упростить процесс отладки.

Семейство MC68HC05 имеет более 200 модификаций. Большинство представителей этого семейства относятся к так называемым заказным устройствам, конфигурация которых определяется требованием конкретной задачи. Микроконтроллеры этого семейства имеют одинаковое 8-разрядное процессорное ядро и отличаются друг от друга набором и количеством периферийных функций (объемом встроенной памяти, наличием АЦП, таймеров).

Семейство MC68HC08 является развитием предыдущего семейства, содержит усовершенствованное центральное ядро и периферию. Кроме того, в состав семейства включены аппаратные средства поддержки нечеткой логики и цифровой обработки сигналов.

Семейство MC68HC11 включает в себя более 40 модификаций, содержит однотипное процессорное ядро на 8 разрядов, имеет более развитую архитектуру и обладает более мощной периферией в сравнении с предыдущей семейством.

Основные характеристики семейств приведены в табл. 1.

Таблица 1

Микроконтроллеры фирмы «Motorola». Основные параметры

Процессорное ядро	HC05	HC08	HC11
Форма данных	8 бит	8 бит	8/16 бит
Частота шины ЦПУ	2 МГц	8 МГц	3 МГц
Число инструкций ЦПУ	65	90	110
Число способов адресации	8	16	7
Разрядность регистров косвенной адресации	8	16	16
Целочисленное умножение	8x8-11 циклов	8x8-5 циклов	8x8-10 циклов
Целочисленное деление	Нет	16:8 7 циклов	16:16 41 цикл
Время доступа к байту при адресации	18 циклов 9 мкс	7 циклов 0.9 мкс	13 циклов 4.3 мкс
Встроенный ШИМ генератор	16 каналов 8 бит	3 канала 12 бит	4 канала 8 бит
Последовательные интерфейсы связи			
SCI – последовательный асинхронный	131 кбит/с	131 кбит/с	131 кбит/с
SPI – последовательный синхронный	1 Мбит/с	4 Мбит/с	1Мбит/с
CAN – сетевой	1 модель	1 Мбит/с	Нет
Аналого-цифровой преобразователь			
Число каналов/разреш. способность	8/8	8/8 или 10	8/8 бит
Время преобразования	32 мкс	17, 8.5 мкс	17 мкс
Модуль сканирования клавиатуры			
Число линий	8	8	Нет

Как видно из таблицы, все серии имеют встроенные SPI, SCI интерфейсы, в 08 серии имеется также встроенный CAN интерфейс. Тактовая частота в 05 серии 2 МГц, 08 серии – 8 МГц, в 11 серии – 3 МГц.

Серии различаются по количеству процессорных инструкций. В HC05 серии их насчитывается 65, в HC08 серии их 90, а в HC11 их 110.

Микроконтроллеры имеют встроенные АЦП, ШИМ генератор. HC05 и HC08 серии имеют модули сканирования клавиатуры.

Микроконтроллеры семейства M68HC05, имеющие наиболее низкую стоимость и менее широкие функциональные возможности по сравнению с другими семействами, ориентированы на использование в относительно несложных устройствах массового применения. Микроконтроллеры семейства M68HC08, выпуск которых начался в 1995 г, программно совместимы с семейством M68HC05, но имеют значительно более высокую производительность, увеличенный объем адресуемой памяти и расширенные функциональные возможности. Данное семейство предназначено для замены микроконтроллеров M68HC05 в современных разработках. Наиболее широкую область применения имеют микроконтроллеры семейства M68HC11, которые сочетают достаточно высокое быстродействие и значительные функциональные возможности с большой номенклатурой реализованных на кристалле периферийных устройств. Таким образом, для каждого семейства существует своя сфера приложения, а в целом их номенклатура обеспечивает потребности многих отраслей промышленности в надежных, высокоэффективных и недорогих устройствах управления и контроля.

Расшифровка обозначения МК семейства HC11 представлена ниже.

MC	**68**	**HC**	**7**	**11XX**	**B**	**C**	**FN**	**3**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

1 – уровень готовности изделия:
- MC – изделие выпускается серийно и прошло все тесты;
- XC – «пилотное» производство;
- PC – выпуск образцов;

2 – цифровое обозначение группы;

3 – технология изготовления и напряжение питания:
- HC – HCMOS, питание 5В;
- L – HCMOS, питание от 3В до 5.5В;

4 – встроенное ПЗУ:

- отсутствие цифры – масочное ПЗУ или отсутствие ПЗУ;
- 7 – ПЗУ с УФ стиранием или однократно программируемое ПЗУ;
- 8 – электрически стираемое ПЗУ (ЭСППЗУ);

5 – обозначение типа МК (11E9, 11A1, 11F1 и т. д.);

6 – наличие монитора-отладчика:

- отсутствие символа – чистая память;
- B – в памяти находится монитор-отладчик BUFFALO;

7 – рабочий температурный диапазон:

- отсутствие цифры – 0...70ºC;
- C – –40...+85ºC;
- V – –40...+105ºC;
- M – –40...+125ºC;

8 – тип корпуса (FN, FX, FU, FB, и т. д.);

9 – максимальная внутренняя тактовая частота:

- 2 – 2.0 МГц;
- 3 – 3.0 МГц;
- 4 – 4.0 МГц.

ЦПУ семейства HC11 оптимизировано по энергопотреблению и быстродействию. Наиболее характерные его черты:

- два 8-битных или один 16-битный аккумулятор;
- два 16-битных индексных регистра;
- два программно управляемых режима пониженного энергопотребления;
- операции умножения 8×8 и деления 16/16;
- внутренняя частота синхронизации повышена до 4 МГц.

ЦПУ некоторых моделей семейства содержит встроенный математический сопроцессор, выполняющий 16-битные операции умножения и деления в 10 раз быстрее, чем процессор. Существуют версии МК с программно управляемым значением тактовой частоты на основе ФАПЧ, что позволяет гибко управлять энергопотреблением в зависимости от сложности вычислительных задач.

МК семейства HC11 имеют в своем составе **все типы внутренней памяти**, упоминавшиеся при рассмотрении семейства MC68HC05:

- размер масочного ПЗУ колеблется от 0 до 32К байт;

- программируемое ПЗУ представлено ПЗУ с УФ стиранием или однократно программируемым ПЗУ с объемом от 0 до 32К байт;
- встроенное ЭСППЗУ имеет в своем составе генератор накачки заряда, что позволяет производить стирание и программирование без использования дополнительного источника питания. Размер ЭСППЗУ варьируется от 0 до 2К;
- объем ОЗУ колеблется от 192 байт до 2К байт. ОЗУ является полностью статичным, и все МК семейства имеют отдельный вывод питания ОЗУ для обеспечения режима микропотребления и сохранения содержимого ОЗУ при напряжении, равном 2В.

Все МК семейства HC11 адресуют внешнюю память, причем есть версии с немультиплексированными магистралями данных и младшей части адреса (HC11F1), а также версии с расширенным до 256К...1М адресным пространством с помощью программируемых выборок внешней памяти.

МК семейства HC11 могут функционировать в одном из трех режимов, которые определяются состоянием специальных входов в момент подачи сигнала начального сброса RESET. **В однокристальном режиме** программа находится во встроенном ППЗУ (или ЭСППЗУ), при этом линии всех портов доступны для ввода/вывода. **В расширенном режиме** возможно подключение внешней памяти программ или данных. **В режиме загрузки** управление после RESET передается программе, записанной в масочном ПЗУ, и производится загрузка кода, например, из персонального компьютера, по последовательному порту в любую область памяти (включая программирование ППЗУ и ЭСППЗУ). Примечательно, что МК семейства HC11 позволяют программно переназначать начало областей ОЗУ, ЭСППЗУ и регистров на границу любой области размером 4К (т. е. 0000, 1000, 2000 и т. д.)

Многофункциональный таймер МК семейства MC68HC11 базируется на 16-битном счетчике, тактируемом через программируемый предделитель на 4, 8 или 16. В отличие от предшествующих моделей МК блок таймера содержит несколько каналов входной фиксации (IC) и несколько каналов выходного сравнения (OC). Функции IC и OC позволяют решать различные задачи управления, критичные ко времени обработки (например, управление трехфазными двигателями переменного тока с обеспечением плавного пуска или управление зажиганием в двигателе автомобиля) с минимальным участием процессора. Блок таймера

содержит также генератор прерываний реального времени с программируемым периодом (от 4.1 до 71.11 мс). Счетчик внешних импульсов, также входящий в подсистему таймера, осуществляет либо подсчет перепадов уровня на входе, либо подсчет импульсов тактовой частоты при наличии активного уровня на входе. К подсистеме таймера относится также COP Watchdog-таймер, формирующий аппаратный сигнал сброса при отсутствии обращения к нему более установленного времени и защищающий тем самым систему от «зависаний».

Аналого-цифровой преобразователь использует метод последовательных приближений и содержит встроенную схему выборки-хранения. Процесс преобразования занимает 32 цикла внутренней тактовой частоты на канал (16 мкс при типичной частоте 2 МГц или 8 мкс при 4 МГц). При тактовой частоте меньшей, чем 750 кГц, используется встроенный RC-генератор частоты для АЦП. Встроенный мультиплексор позволяет проводить преобразование по одному из защищенных от перенапряжения внешних аналоговых входов (до 8), а также измерять Vrl, Vrh и (Vrh+Vrl)/2 для проведения контроля и юстировки. Подсистема АЦП содержит регистр управления (задает режим работы и запуск преобразования), регистр статуса (содержит флаг окончания преобразования) и регистр данных (результат преобразования). АЦП поддерживает также режим последовательного преобразования по всем каналам с записью результатов в индивидуальные регистры данных.

Некоторые представители семейства MC68HC11 имеют до 6 каналов 8-разрядного **широтно-импульсного модулятора**, часто используемого также для формирования программируемого постоянного напряжения. При частоте синхронизации 4 Мгц схема ШИМ позволяет формировать сигналы с частотой от 10 Гц до 40 кГц. В расширенном 16-разрядном режиме возможно формирование сигналов с периодом более минуты.

1.2.2. Отладочные средства для микроконтроллеров фирмы «Motorola»

Для МК фирмы «Motorola» производится множество средств отладки, которые можно разделить на программаторы, отладочные модули (Evaluation Modules), отладочные платы (Evaluation Boards), отладочные системы (Evaluation Systems), компактные отладочные

системы (Compact Development Systems) и модульные отладочные системы (Modular Development System).

Программаторы являются наиболее простыми и дешевыми средствами записи программы (например, отлаженной с помощью программного симулятора) во встроенную память микроконтроллера.

Отладочные модули (EVM) являются универсальными устройствами внутрисхемной эмуляции и поддерживают, как правило, несколько типов МК семейства, относящихся к группам общего назначения (HC05C, HC05B, HC11E, и др.) Они позволяют проводить загрузку объектного кода программ из IBM PC в ОЗУ модуля, эмулирующее встроенное ПЗУ микроконтроллера, выполнять программу пользователя в различных режимах, включая выполнение по шагам команд или с использованием точек останова, просматривать и модифицировать содержимое регистров и памяти, отлаживать разрабатываемую систему в реальном времени (подключив отладочный модуль к разрабатываемой системе вместо МК с помощью специального кабеля), программировать отлаженную программу во встроенную память МК, и т. д. Отладочные модули поставляются в виде печатной платы, кабеля для подключения в гнездо МК отлаживаемой системы и набора программных средств.

Отладочные платы (EVB) выполняют все функции отладочных модулей и отличаются от них тем, что поддержка точек останова в них организована программно, а не аппаратно. Кроме того, EVB не имеет разъема для подключения терминала и, соответственно, встроенного монитора-отладчика для работы с терминалом, и ориентирован на работу только с IBM PC. Отладочная плата комплектуется интегрированной программной оболочкой, включающей в себя ассемблер и отладчик, и имеет значительно меньшую стоимость, чем EVM, и является наиболее эффективным средством отладки по соотношению функциональные возможности / цена.

Таким образом, серия M68HC11EVB выполняет функции, аналогичные EVM, но не содержит программатора.

Серия M68HC11EVBU имеет на плате макетное поле для проводного монтажа дополнительных устройств, необходимых в прототипной системе.

В нашем случае используется платы M68HC11EVB и M68HC11EVBU фирмы «Motorola».

Плата EVB является недорогим решением для задач отладки и эмуляции систем на базе микроконтроллеров семейства MC68HC11. Она может работать в одном из двух режимов: отладки и эмуляции. Первый режим позволяет вам отлаживать код, используя программу-монитор BUFFALO: программа пользователя набирается на компьютере и затем пересылается в ОЗУ платы в S-формате (см. приложение 5). Второй режим позволяет эмулировать пользовательский код, используя память микроконтроллера. Плата EVB эмулирует только однокристальный режим работы, хотя она все время работает в расширенном режиме.

В качестве терминала для работы с программой-монитором BUFFALO используется программа-терминал KERMIT (см. M68HC11EVB User's Manual, M68HC11EVBU User's Manual, там же описана работа с отладчиком BUFFALO)

Примечание. Оценочная плата M68HC11EVB требует питание: +5В, +12В и –12В, которое в нашем случае берется от источников питания для PC либо от универсального источника питания (ТЕС 1300К или блок питания от ПЭВМ). Плата M68HC11EVBU требует для своей работы питание +5В и запитывается от автономного блока питания.

Отладочная система (EVS) состоит из двух печатных плат: универсальной «платформы», не зависящей от типа МК, и сменной платы, индивидуальной для каждого типа МК. Сменные платы существуют практически для любого микроконтроллера семейства HC05. В остальном отладочная система выполняет функции, аналогичные приведенным выше для отладочных модулей и плат.

Компактные отладочные системы (CDS) являются полностью функционально законченными устройствами, включают встроенный источник питания и корпус, и позволяют выполнять некоторые дополнительные функции отладки. Работают совместно с IBM PC.

Модульная отладочная система (Motorola Modular Development System, MMDS) является усовершенствованным вариантом CDS и позволяет вести отладку в полностью интегрированной среде на уровне исходных кодов и с использованием «окна» в памяти, позволяющего изменять ее содержимое непосредственно во время выполнения программы в реальном времени.

1.3. АРХИТЕКТУРА И СТРУКТУРА МИКРОКОНТРОЛЛЕРА MC68HC711E9

1.3.1. Структурная схема однокристального микроконтроллера MC68HC711E9

Структурная схема однокристального микроконтроллера **MC68HC711E9** приведена на рис. 4.

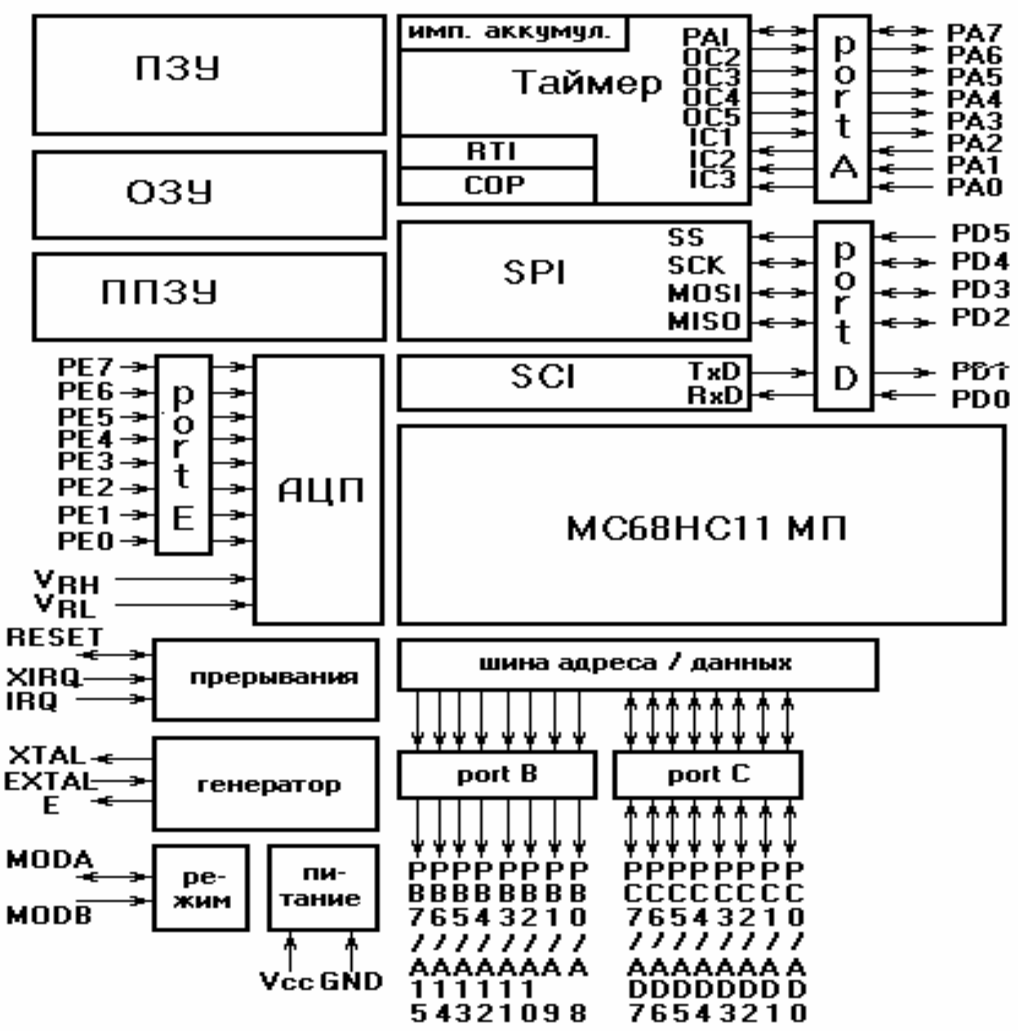

Рис. 4. Блок схема однокристального контроллера MC68HC711E9.

Микроконтроллер MC68HC711E9 включает в себя следующие подсистемы:

- тактовый генератор;
- 8-разрядное микропроцессорное ядро(MC68HC11);
- встроенное 12 Кбайт ПЗУ с ультрафиолетовым стиранием;
- встроенное ППЗУ с электрическим стиранием на 512 байт;

- встроенное ОЗУ на 512 байт;
- 16-разрядный таймер;
- 8-разрядный, 8-канальный АЦП;
- 2 системы последовательной связи;
- систему параллельного ввода/вывода;
- систему обработки прерываний;
- систему выбора режима работы.

Программная модель контроллера MC68HC11 представлена на рис. 5.

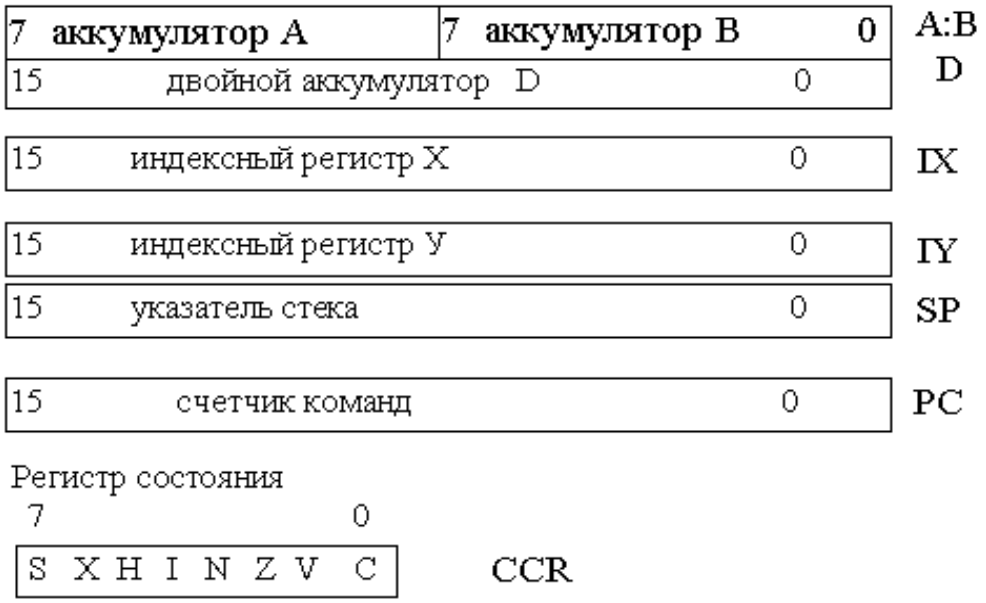

Рис. 5. Программная модель контроллера MC68HC11

1.3.2. Режимы работы контроллера MC68HC11

В контроллере MC68HC11 реализованы следующие режимы работы:

- одиночный режим;
- дополнительный режим;
- специальный режим загрузки;
- специальный тестовый режим.

Режим работы микроконтроллера устанавливается состоянием входов MODA и MODB контроллера во время прохождения сигнала RESET. После прохождения сигнала RESET изменение состояния входов не влияет на режим работы микроконтроллера. Кроме того, в

зависимости от выбранного режима меняются установки управляющих регистров и соответственно алгоритм работы контроллера. Состояния входов MODA и MODB и соответствующие им режимы, а также управляющие биты регистра HPRIO приведены в табл. 2.

Нормальные режимы работы

Выбор одного из нормальных режимов происходит при наличии логической 1 на входе MODB во время сигнала RESET.

Таблица 2

Режимы контроллера MC68HC11

Входы		Описание режима	Управляющие биты в HPRIO			
MODB	MODA		RBOOT	SMOD	MDA	IRV
1	0	Норм. одиночный	0	0	0	0
1	1	Норм. расширенный	0	0	1	0
0	0	Спец. загрузочный	1	1	0	1
0	1	Спец. тест	0	1	1	1

Нормальный одиночный режим работы

Установка нормального одиночного режима происходит установкой входа MODB в логическую 1, а входа MODA в 0. Поскольку в этом режиме все необходимое программное обеспечение находится во внутренней памяти контроллера, то не требуются внешние шины данных и адреса. Таким образом, Port B и Port C может быть использован для организации параллельного ввода/вывода.

Нормальный расширенный режим

Установка нормального одиночного режима происходит установкой входов MODB и MODA в логическую 1.

Этот режим позволяет использовать внешнюю память и периферийные устройства, подключаемые через мультиплексную шину данных/адреса, организованную через Port B и Port C, что не позволяет их использование для параллельного ввода/вывода.

Специальные режимы контроллера

Специальные режимы контроллера устанавливаются заданием логического 0 на входе MODB.

В этих режимах программное обеспечение имеет доступ к специальным тестовым функциям, меняется карта памяти и изменен алгоритм установки обработчиков внешних прерываний.

Для обработки внешних прерываний используется таблица векторов обработчиков, расположенная в ПЗУ по адресу BF40h – BFFFh. Каждый вектор из этой таблицы указывает на 3-байтную область памяти (адреса 00C4h – 00FFh), в которую пользователь помещает JMP на собственный обработчик прерывания. Поскольку таблица векторов прерываний перекрывает адресное пространство встроенного 8К-байтного ПЗУ, последнее отключается.

Специальный режим загрузки

Загрузочный режим назван специальным, т. к. он отличается от нормального однокристального режима. Используется для загрузки необходимой для выполнения программы во внутреннее ОЗУ. Программа-загрузчик содержит 192 байта в ПЗУ по адресу $BF40-$BFFF и начинает работать только после сброса в этом режиме. Загрузочная программа использует для связи асинхронный последовательный интерфейс(SCI) и загружает программы длинной не более 512 байтов во внутреннее ОЗУ по адресу $0000-$01FF. После приема последнего байта управление автоматически передается программе (адрес $0000). После сброса в специальном загрузочном режиме SCI настраивается на частоту работы E/16(7812 бод для E=2МГц). Если установлен бит защиты от несанкционированного доступа, то по SCI передается в EEPROM и RAM. Регистр CONFIG также стирается. Теперь загрузочная программа продолжает работать как в обычном, не защищенном случае. По SCI предается символ BREAK. Пользователь передает байт $FF. Затем пользователь передает 512 байтов программы, которая будет загружена в ОЗУ. Принимаемые байты передаются обратно через передатчик. Когда загрузка завершается, управление передается по адресу $0000.

Если контакт передатчика SCI используется, то требуется внешний подтягивающий резистор, потому что контакты порта D настроены на операцию монтажного ИЛИ.

В специальном загрузочном режиме вектора прерывании переадресуются в ОЗУ, как показано в табл. 3. Это позволяет пользователю обрабатывать прерывания, используя таблицу переходов, расположенную в ОЗУ. Например, инструкция переходов для программного прерывания SWI должна быть расположена в ОЗУ

Таблица 3

Вектора прерывании для загрузочного режима.

Адрес	**Вектор**
00C4	SCI
00C7	SPI
00CA	Импульсный аккумулятор
00CD	Переполнение имп. аккумулятора
00D0	Переполнение таймера
00D3	5-я функция сравнения таймера
00D6	4-я функция сравнения таймера
00D9	3-я функция сравнения таймера
00DC	2-я функция сравнения таймера
00DF	1-я функция сравнения таймера
00E2	3-я линия захвата таймера
00E5	2-я линия захвата таймера
00E8	1-я линия захвата таймера
00EB	Прерывание таймера реального времени
00EE	IRQ
00F1	XIRQ
00F4	SWI
00F7	Неразрешенный код операции
00FA	Прерывание от COP
00FD	Часы
BF40	Сброс

по адресу $00F4, $00F5 и $00F6. Когда возникнет SWI, вектор прерывания (находящийся в области загрузочной программы в ПЗУ) передает управление по адресу $00F4 в ОЗУ, где находится инструкция перехода JUMP к программе обработчику прерывания.

Первым байтом пользователь может передать $55(только при частоте E/16) вместо $FF. Это вызовет программный переход прямо в позицию $0000, минуя загрузку программы.

Если первым байтом передать $00, то управление перейдет к EEPROM($B600).

Заметим, что ни одна из этих опций не позволяет обойти проверку бита защиты.

Специальный тестовый режим

Режим используется для внутреннего тестирования контроллера фирмой «Motorola», тем не менее есть ряд ситуаций, когда пользователь может инициализировать этот режим.

1.3.3. Конфигурирование и управление работой контроллера

Кроме описанных выше входов MODA и MODB, влияние на работу контроллера оказывают следующие специальные управляющие регистры, выполненные на базе ППЗУ:

Регистр **CONFIG** (103Fh).

7	6	5	4	3	2	1	0
0	0	0	0	NOSEC	NOCOP	ROMON	EEON
0	0	0	0	-	-	-	-

NOSEC – включение системы защиты ППЗУ.

Включение системы защиты ППЗУ позволяет защитить данные, хранящиеся во встроенном ППЗУ, от просмотра хакером.

NOCOP = 1 – система COP отключена и не вызывает системных сбросов.

NOCOP = 0 – система COP активна.

ROMON – подключение встроенного 8К-байтного ПЗУ.

ROMON = 1 – встроенное ПЗУ подключено.

ROMON = 0 – встроенное ПЗУ отключено.

EEON – подключение встроенного 512-байтного ППЗУ.

EEON = 1 – встроенное 512-байтное ППЗУ расположено по адресу B600h – B7FFh.

EEON = 0 – встроенное 512-байтное ППЗУ отключено.

Регистр **INIT**(103Dh).

7	6	5	4	3	2	1	0
Ram3	Ram2	Ram 1	Ram0	Reg3	Reg2	Reg1	Reg0
0	0	0	0	0	0	0	0

Ram3-Ram0 – расположение встроенного ОЗУ на карте памяти.

Эти 4 бита определяют старший 16-тиричный байт адреса встроенного ОЗУ. Изменяя значения этих бит, можно располагать встроенное ОЗУ в начале любой 4К-байтной страницы памяти.

Reg3-Reg0 – расположение 64-байтного блока регистров управления.

Эти 4 бита определяют старший 16-тиричный байт адреса встроенного ОЗУ. Изменяя значения этих бит, можно располагать встроенное ОЗУ в начале любой 4К-байтной страницы памяти.

Регистр **TMSK2** (1024h).

7	6	5	4	3	2	**1**	**0**
TOI	RTII	PAOVI	PAII	0	0	**PR1**	**PR0**
0	0	0	0	0	0	**0**	**0**

PR1-PR0 – выбор коэффициента деления таймера.

Эти 2 бита устанавливают коэффициент деления для встроенного 16-разрядного таймера. Частота задающих импульсов для таймера определяется следующим образом: частота системных тактовых импульсов, деленная на коэффициент, задаваемый разрядами **PR1-PR0**. Связь между разрядами **PR1-PR0** приведена в следующей таблице.

PR1	PR0	Коэффициент деления
0	0	1
0	1	4
1	0	8
1	1	16

Регистр **OPTION** (1024h).

7	6	5	4	3	2	1	0
ADPU	CSEL	IRQE	DLY	CME	0	CR1	CR0
0	0	0	1	0	0	0	0

IRQE – настройка входа IRQ на срабатывание по фронту.

IRQE = 1 – запрос прерывания по фронту.

IRQE = 0 – запрос прерывания по уровню.

DLY – задержка при старте.

DLY = 1 – при выходе процессора из состояния STOP вводится задержка примерно на 4000 тактов для стабилизации частоты тактового генератора.

DLY = 0 – задержки нет.

CR1 – CR0 – коэффициент деления для таймера системы **COP**.

Тактовая частота первоначально делится на 2^15, после чего будет подана на таймер системы **COP watchdog**. Дальнейшее деление частоты тактирования таймера определяется битами **CR1-CR0**.

CR1	CR0	Коэффициент деления
0	0	1
0	1	4
1	0	8
1	1	16

Все описанные выше регистры управления можно отнести к регистрам глобального действия, т. к. они влияют на работу всей системы в целом. Кроме этих, каждая система контроллера MC68HC11 имеет собственные регистры управления, обеспечивающие работу только своей системы. Также некоторые разряды регистров общего назначения могут влиять на работу отдельной системы.

1.3.4. Встроенная память

Данный раздел описывает встроенные ПЗУ(ROM), ОЗУ(RAM) и электрически перепрограммируемое ПЗУ(EEPROM).

1.3.4.1. Карта памяти

Карта памяти для каждого режима работы показана на рис. 6, а управляющие регистры показаны в табл. 4.

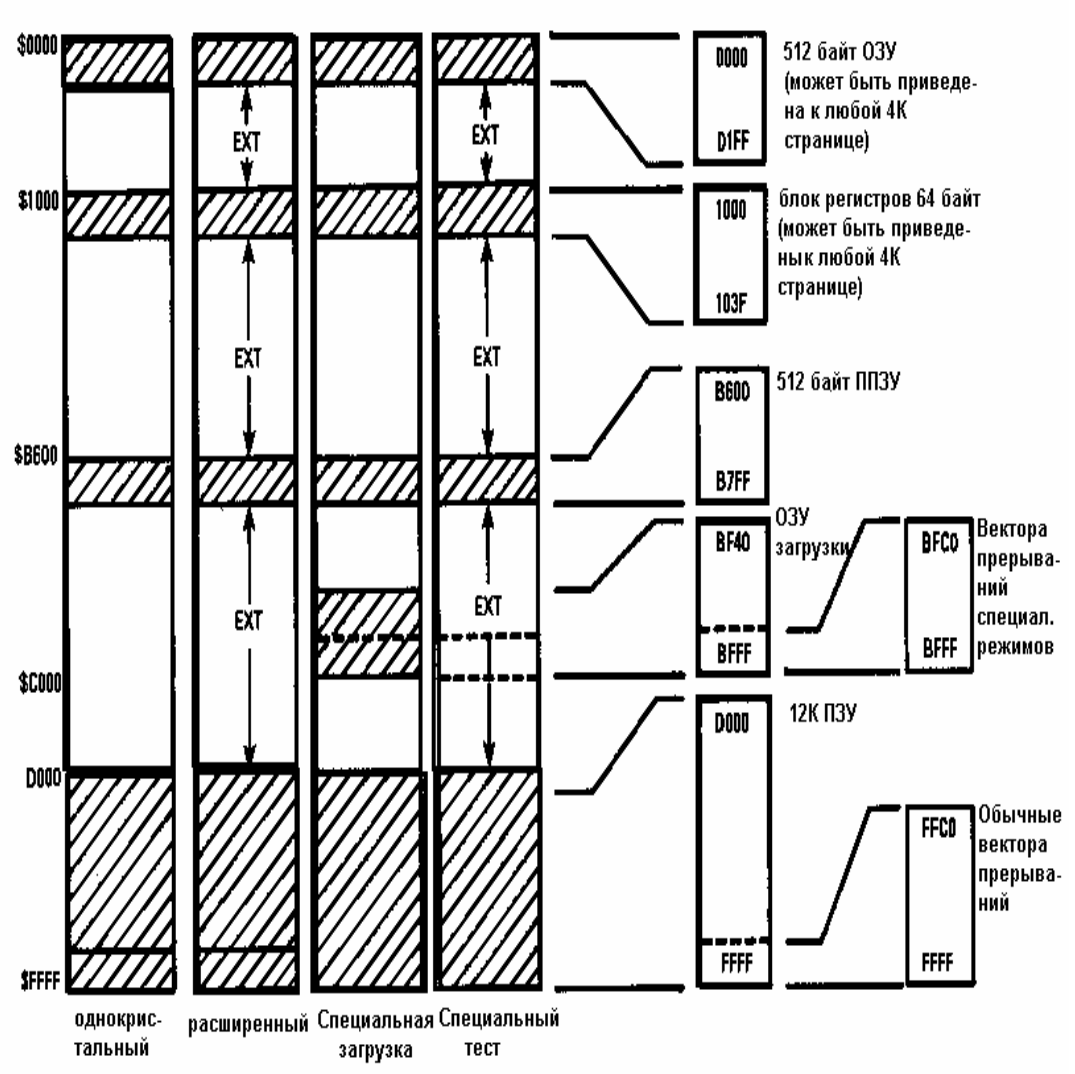

Рис. 6. Карта памяти

Таблица 4.

Назначение битов управляющих регистров

	Бит 7	бит 6	бит 5	бит 4	бит 3	бит 2	Бит 1	бит 0		
$1000	Бит7	-	-	-	-	-	-	Бит0	PORTA	Ввод/ вывод порт А
$1001									Резерв	
$1002	STAF	STAI	CWOM	HNDS	OIN	PLS	EGA	INVB	PIOC	Упр. Регистр ввода/ вывода
$1003	Бит7	-	-	-	-	-	-	бит0	PORTC	Ввод/ вывод порт С
$1004	Бит7	-	-	-	-	-	-	бит0	PORTB	Ввод порта D
$1005	Бит7	-	-	-	-	-	-	бит0	PORTCL	Защелка порта С
$1006									Резерв	
$1007	Бит7	-	-	-	-	-	-	бит0	DDRC	Направление данных для С
$1008			бит5	-	-	-	-	бит0	PORTD	Ввод/ вывод порта D
$1009			бит5	-	-	-	-	бит0	DDRD	Направление данных для D
$100A	Бит7	-	-	-	-	-	-	бит0	PORTE	Ввод порта D
$100B	FOC1	FOC2	FOC3	FOC4	FOC5				CFORC	Регистр сравнений

$100C	OC1M7	OC1M6	OC1M5	OC1M4	OC1M3				OC1M	Регистр маски события
$100D	OC1D7	OC1D6	OC1D5	OC1D4	OC1D3				OC1D	Регистр данных события
$100E	Бит15	-	-	-	-	-	-	бит8	TCNT	Счетчик таймера
$100F	Бит7	-	-	-	-	-	-	бит0		
$1010	Бит15	-	-	-	-	-	-	бит8	TIC1	Регистр 1-й линии захвата таймера
$1011	Бит7	-	-	-	-	-	-	бит0		
$1012	Бит15	-	-	-	-	-	-	бит8	TIC2	Регистр 2-й линии захвата таймера
$1013	Бит7	-	-	-	-	-	-	бит0		
$1014	Бит15	-	-	-	-	-	-	бит8	TIC3	Регистр 3-й линии захвата таймера
$1015	Бит7	-	-	-	-	-	-	бит0		
$1016	Бит15	-	-	-	-	-	-	бит8	TOC1	Регистр 1-й линии сравнения таймера
$1017	Бит7	-	-	-	-	-	-	бит0		
$1018	Бит15	-	-	-	-	-	-	бит8	TOC2	Регистр 2-й линии сравнения таймера
$1019	Бит7	-	-	-	-	-	-	бит0		

Продолжение

\$101A	Бит15	-	-	-	-	-	-	бит8	TOC3	Регистр 3-й линии сравнения таймера
\$101B	Бит7	-	-	-	-	-	-	бит0		
\$101C	Бит15	-	-	-	-	-	-	бит8	TOC4	Регистр 4-й линии сравнения таймера
\$101D	Бит7	-	-	-	-	-	-	бит0		
\$101E	Бит15	-	-	-	-	-	-	бит8	TI4O5	Регистр 5-й линии сравнения и 4й захвата таймера
\$101F	Бит7	-	-	-	-	-	-	бит0		
\$1020	OM2	OL2	OM3	OL3	OM4	OL4	OM5	OL5	TCTL1	Упр. регистр таймера 1
\$1021	EDG4B	EDG4A	EDG1B	EDG1A	EDG2B	EDG2A	EDG3B	EDG3A	TCTL2	Упр. регистр таймера 2
\$1022	OC1I	OC2I	OC3I	OC4I	I4O5I	IC1I	IC2I	IC3I	TMSK1	Маска прерыв. таймера 1
\$1023	OC1F	OC2F	OC3F	OC4F	I4O5F	IC1F	IC2F	IC3F	TFLG1	Рег. флагов прерываний 1
\$1024	TOI	RTII	PAOVI	PAII					TMSK2	Маска прерыв. таймера 2
\$1025	TOF	RTIF	PAOVF	PAIF					TFLG2	Рег. флагов прерываний 2

Продолжение

$1026	DDRA7	PAEN	PAMOD	PEDGE	DDRA3	I4/O5	RTR1	RTR0	PACTL	Упр. рег. Накопителя имп.
$1027	Бит7	-	-	-	-	-	-	Бит0	PACNT	Счетчик накопителя имп.
$1028	SPIE	SPE	DWOM	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR	Упр. рег. SPI
$1029	SPIF	WCOL		MODF					SPSR	Регистр статуса SPI
$102A	Бит7	-	-	-	-	-	-	Бит0	SPDR	Регистр данных SPI
$102B	TCLR		SCP1	SCP0	RCKB	SCR2	SCR1	SCR0	BAUD	Упр. скоростью SCI
$102C	R8	T8		M	WAKE				SCCR1	Упр. рег. 1 SCI
$102D	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SCCR2	Упр. рег. 2 SCI
$102E	TDRE	TC	RDRF	IDLE	OR	NF	FE		SCSR	Регистр статуса SCI
$102F	Бит7	-	-	-	-	-	-	Бит0	SCDR	Регистр данных SCI
$1030	CCF		SCAN	MULT	CD	CC	CB	CA	ADCTL	Упр. рег. АЦП
$1031	Бит7	-	-	-	-	-	-	Бит0	ADR1	Регистр резуль-тата 1

Продолжение

$1032	Бит7	-	-	-	-	-	-	бит0	ADR2	Регистр резуль-тата 2
$1033	Бит7	-	-	-	-	-	-	бит0	ADR3	Регистр резуль-тата 3
$1034	Бит7	-	-	-	-	-	-	бит0	ADR4	Регистр резуль-тата 4
$1035				PTCON	BPRT3	BPRT2	BPRT1	BPRT0	BPROT	Регистр защиты ППЗУ
$1036 До $1038									Резерв	
$1039	ADPU	CSEL	IRQE	DLY	CME		CR1	CR0	OPTION	Конфигур системы
$103A	Бит7	-	-	-	-	-	-	бит0	CORPST	
$103B	ODD	EVEN		BYTE	ROW	ERASE	EELAT	EEPGM	PPROG	Рег. упр. программ ирова нием ППЗУ
$103C	RBOOT	SMOD	MDA	IRV	PSEL3	PSEL2	PSEL1	PSEL0	HPRIO	
$103D	RAM3	RAM2	RAM1	RAM0	REG3	REG2	REG1	REG0	INIT	Размещен ОЗУ и блока регистров
$103E	TILOP		OCCR	CBYP	DISR	FCM	FCOP	TCON	TEST1	Тестовый регистр
$103F					NOSEC	NOCOP	ROMON	EEON	CONFIG	Включени е COP, ROM, EEPROM

В расширенном мультиплексном режиме распределение памяти такое же, как и в однокристальном режиме. Однако адреса, расположенные между заштрихованными областями (обозначены EXT), относятся к внешней памяти и устройством ввода/ вывода. Если адреса внешних устройств перекрывают адреса внутренних подключенных устройств, то внутренние устройства имеют приоритет. При чтении по такому адресу данные, поступающие на входы порта С, игнорируются, и происходит чтение по внутреннему адресу. При записи по такому адресу данные выводятся через порт С, а также записываются по внутреннему адресу. Поэтому ни одно внешнее устройство не должно писать данные в порт С во время записи во внутреннюю ячейку. При правильном использовании сигнала R/W конфликтов не происходит. Сигналы R/W, AS, адрес и записываемые данные действительны при обращении к любому устройству, включая внутреннюю память и регистры.

Распределение памяти в специальном загрузочном режиме такое же, как и в однокристальном, за исключением того, что становится задействована загрузочная программа по адресам $BFC0-$BFFF и векторы прерываний располагаются по адресам $BFC0-$BFFF.

Распределение памяти в специальном тестовом режиме такое же, как в расширенном мультиплексном, за исключением того, что векторы прерываний расположены во внешней памяти по адресам $BFC0-$BFFF.

1.3.4.2. Регистр перемещения RAM и блока регистров (INIT)

Специальный регистр INIT позволяет перемещать RAM и блок управляющих регистров в начало любой 4К-байтной страницы в карте памяти.

Запись в этот регистр возможна только в течении 64 E-циклов после сброса.

	7	6	5	4	3	2	1	0	
$103D	RAM3	RAM2	RAM1	RAM0	REG3	REG2	REG1	REG0	INIT
RESET	0	0	0	0	0	0	0	1	

Старшие четыре бита обозначают старшую шестнадцатиричную цифру адреса RAM в карте памяти. Младшие четыре бита обозначают старшую шестнадцатиричную цифру адрес 64-байтного блока управляющих регистров в карте памяти. После сброса в регистре INIT устанавливается значение $01. Поэтому начальный адрес RAM=$0000.

Начальный адрес блока регистров $1000. Заметим, что в блоке регистров есть неиспользуемые регистры. Чтение по этим адресам возвратит данные с неуправляемой внутренней шины данных, а не из другого устройства, которое может иметь тот же адрес.

1.3.4.3. ROM

Внутренняя ROM (12K байт) занимает последние 12K в карте памяти ($D000 – $FFFF). Эта ROM разрешена для обращения, когда бит ROMON расположен в ячейке EEPROM, поэтому процедура его программирования такая же, как при программировании внутренней EEPROM. Более детальную информацию смотрите в пункте 3.5.3 «Регистр конфигурации системы (CONFIG)».

В однокристальном режиме внутренняя ROM разрешена, несмотря на состояние бита ROMON.

В MC68HC11E9 есть также 192 байта масочной ROM, где записана программа, управляющая специальным загрузочным режимом. Эта программа работает только после сброса в специальном загрузочном режиме. Более подробную информацию смотрите в электронном учебнике «MC68HC11E9-HCMOS Single-Chip Microcontroller» (пункт 2.2.3 «Специальный загрузочный режим»).

1.3.4.4. RAM

Внутренняя RAM объемом 512 байт может быть размещена в карте памяти на границе любой 4K-байтной странице посредством записи нужных битов в регистр INIT. После сброса RAM располагается по адресам $0000-$01FF.

Эта RAM статического типа, и ее содержимое сохраняется в режимах WAIT и STOP.

Содержимое RAM может быть также сохранено после отключения питания контроллера, если подключить маломощный источник питания (батарейку) к выводу MODB/Vstby. Перед отключением и подключением Vdd сигнал сброса должен находится на низком уровне.

1.3.4.5. EEPROM

512 байтов EEPROM расположены по адресам $B600-$B7FF и имеют такое же время чтения, как и ROM. Механизм записи (программирование) в EEPROM управляется регистром PPROG. EEPROM. Запись отключена, когда бит EEON в регистре CONFIG

равен нулю. Бит EEON является электрически перепрограммируемой ячейкой. Очищенным состоянием байта EEPROM является $FF. Операция программирования – это запись нулей вместо единиц там, где это нужно. Если какой-либо бит требуется изменить из нуля в единицу, то сначала нужно в отдельной операции очистить байт, а затем записать нули. Если новый байт данных не имеет единиц в позициях, где уже записаны нули, то его можно записывать без предварительного разрушения байта EEPROM.

Программирование и очищение EEPROM осуществляется с помощью внутреннего высоковольтного преобразователя. Если частота E меньше 2 МГц, эффективность преобразователя уменьшается, и время программирования увеличивается. Рекомендуемое время программирования и стирания – 10 мс при частоте E, равной 2 МГц; должно быть увеличено до 20 мс, когда E между 1 МГц и 2 МГц. Когда частота E меньше 1 МГц, источник синхронизации для преобразователя напряжения следует переключить на внутренний RC-генератор. Это делается установкой бита CSEL в режим OPTIN.

После установки бита CSEL преобразователю требуется 10 мс для стабилизации. Заметим, что бит CSEL также управляет синхронизацией подсистемы аналого-цифрового преобразования.

1.3.4.5.1. Регистр управления программированием EEPROM (PPROG)

Этот регистр используется для управления программированием и стиранием 512-байтной EEPROM. По стробу регистр настраивает EEPROM для чтения.

	7	6	5	4	3	2	1	0	
$103B	ODD	EEVEN	0	BYTE	ROW	ERASE	EELAT	EEPGM	PROG
RESET									

ODD – программировать нечетные ряды (в тестовом режиме);

EEVEN – программировать четные ряды (в тестовом режиме);

ВИТ 5 – не используется; всегда используется как ноль;

BYTE – побайтный режим стирания;

ROW – построчный режим стирания.

Эти два бита определяют тип стирания, как показано в нижеприведенной таблице. Они ни на что не влияют, когда бит ERASE очищен.

BYTE	ROW	ТИП СТИРАНИЯ
0	0	полное (все 512 байт)
0	1	построчное (16 байт)
1	0	Побайтное
1	1	Побайтное

ERASE – выбор режима стирания:

0 – нормальное чтение или программирование;

1 – режим стирания;

EELAT – управление защелкой EEPROM:

0 – шина адреса и данных EEPROM сконфигурирована для чтения;

1 – шина данных и адреса EEPROM сконфигурирована для программирования / стирания;

EEPRGM – включение напряжения программирования EEPROM:

0 – выключено;

1 – включено.

Логический блокировочный механизм предотвращает установку бита EEPGM до того, как будет установлен бит EELAT. Попытка установить эти биты одновременно также ни к чему не приводит. Если установлен бит EEPRGM, то попытки записи по адресу EEPORM игнорируются и текущая операция программирования не нарушается. Эти две меры предосторожности приняты для предотвращения случайного изменения EEPROM в случае программного сбоя.

Пока бит EELAT установлен, EEPROM не может быть прочитана. Это значит, что программа, которая стирает или программирует EEPROM, не может находиться при выполнении в той же самой EEPROM.

1.3.4.5.2. Регистр защиты блока EEPROM (BPROT)

Регистр используется для предотвращения случайной записи в регистр CONFIG и 512 байт EEPROM. Биты в этом регистре могут быть сброшены в ноль только в первые 64 E цикла после сброса в нормальных режимах. Один раз установленные в ноль биты разрешают программировать или стирать EEPROM и/или регистр CONFIG. Биты в регистре BPROT могут быть установлены в единицу (в любом режиме) для защиты EEPROM и регистра CONFIG, но обратно очищены могут быть только в текстовом и загрузочном режимах

	7	6	5	4	3	2	1	0	
$1035				PTCON	BPRT3	BPRT2	BPRT1	BPRT0	BPROT

Биты 7–5 не используются. Эти биты всегда читаются как нуль.

PTCON – защита регистра CONFIG:

1 – программирование / стирание регистра CONFIG запрещено;

0 – программирование / стирание регистра CONFIG разрешено;

BPRT3 – BPRT0 – защита блока.

Эти биты, когда установлены, защищают EEPROM от программирования и стирания, а когда очищены, позволяют программировать или стирать соответствующий блок.

Бит	**Защищаемый блок**	**Размер**
BPRT0	$B600-$B61F	32 байтов
BPRT1	$B620-$B65F	64 байтов
BPRT2	$B660-$B6DF	128 байтов
BPRT3	$B6E0-$B7FF	288 байтов

1.3.4.5.3. Программирование/стирание встроенной EEPROM

Процессами программирования и стирания EEPROM управляет регистр PPROG. Соответствующие биты в регистре BPROT должны быть сброшены перед тем, как изменять EEPROM.

Следующие пункты описывают различные операции с EEPROM и включают программные сегменты, демонстрирующие операции программирования и стирания. Подпрограмма задержки на 10 мс DLY10 не производится.

Нет специальных ограничений на режимы адресации, могут использоваться инструкции манипуляции с битами. Во время программирования и стирания контроллер может выполнять другие действия, за исключением чтения и записи EEPROM.

1.3.4.5.3.1. Чтение

Чтобы прочитать EEPROM, необходимо очистить EELAT в регистре PPROG. В этом случае оставшиеся биты в регистре PPROG и регистр BPROT ни на что не влияют, и EEPROM работает как обычный ROM.

1.3.4.5.3.2. Программирование

Во время программирования EEPROM биты ROW и BYTE не используются. Если частота E равна 1МГц или меньше, должны быть

стерты в отдельной операции стирания. Следующий программный сегмент демонстрирует, как запрограммировать байт EEPROM.

```
*ВХОД: A=байт данных, X=адрес ячейки EEPROM
PROG    LDAB    #$02
        STAB    $103B       установка EELAT=1(EEPGM=0)
        STAA    0,X         запись данных по нужному адресу
        LDAB    #$03
        STAB    $103        установка EEPGM=1(EELAT=1)
        JSR     DLY10       задержка 10мс
        CLR     $103B       выкл. напряжение программирования и
                            установить режим чтения.
```

1.3.4.5.3.3. Полное стирание

Следующий программный сегмент демонстрирует, как одновременно стереть все 512 байтов EEPROM.

```
BULKE   LDAB    #$06
        STAB    $103B       установка режима полного стирания
        STAB    $B600       записать любые данные по любому адресу
        LDAB    #$07
        STAB    $103B       выключить напряжение программирования
        JSR     DLY10       задержка 10мс
        CLR     $103B       выкл. напряжение программирования и
                            установить режим чтения
```

1.1.4.5.3.4. Построчное стирание

Следующий программный сегмент демонстрирует функцию построчного стирания. «Строка» – это 16 байтов ($B600-$B60F, $B610-$B61F, …, $B7F0-$B7FF). Этот тип стирания позволяет сэкономить время, когда нужно стереть большой объем информации в EEPROM.

```
* ВХОД: X=адрес стираемой строки.
ROWE    LDAB    #$0E        установить режим
        STAB    $103B       построчного стирания
        STAB    0,X         записать любые данные по любому адресу
        LTAB    #$0F
        STAB    $103B       включить напряжение программирования
        JSR     DLY10       задержка 10мс
        CLR     $103B       выкл. напряжение программирования и
                            установить режим чтения
```

1.3.4.5.3.5. Стирание байта

Следующий программный сегмент демонстрирует функцию стирания байта.

```
        *ВХОД: X=адрес стираемого байта
BYTEE   LDAB    #$16        установить режим
        STAB    $103B       стирания байта
        STAB    0,X         записать любые данные по стираемому адресу
        LTAB    #$17
        STAB    $103B       включить высокое напряжение
        JSR     DLY10       задержка 10 мс
        CLR     $103B       выключить высокое напряжение и установить
                            режим чтения.
```

1.3.4.5.4. Регистр конфигурации системы(CONFIG)

Регистр конфигурации системы CONFIG выполнен как ячейка EEPROM. Управляющие биты этого регистра позволяют подключать и отключать внутренние EEPROM и ROM, систему слежения за ошибками COP, и также устанавливать защиту данных от несанкционированного доступа.

	7	6	5	4	3	2	1	0	
$103F	0	0	0	0	NOSEC	NOCOP	ROMON	EEON	CONFIG
RESET	см. 3.5.3.2								

Биты 7, 6, 5, 4 не задействованы и всегда читаются как ноль.

NOSEC – установка режима защиты.

Этот бит задействован, если было специальное требование при программировании ROM масочным способом. Если этот бит не задействован, он всегда читается как единица. Когда бит NOSEC задействован и установлен в ноль, он препятствует выбору расширенного мультиплексного режима, а при выборе специального загрузочного режима информация в RAM, EEPROM и CONFIG разрушается прежде, чем начнется процесс загрузки.

0 – включить режим защиты;

1 – выключить режим защиты.

NOCOP – включение системы COP:

0 – система слежения за ошибками подключения;

1 – система слежения за ошибками отключения.

ROMON – подключение встроенной ROM.

Когда бит ROMON очищен, внутренняя ROM отключена. В однокристальном режиме внутренняя ROM подключена независимо от состояния бита ROMON.

EEON – подключение встроенной EEPROM.

0 – отключить встроенную EEPROM;

1 – подключить встроенную EEPROM.

1.3.4.5.4.1. Программирование и стирание режима CONFIG

Регистр CONFIG программируется и стирается так же, как любая ячейка EEPROM, только указывается адрес регистра CONFIG. Следующий программный сегмент демонстрирует, как программируется регистр CONFIG. В этой программе подразумевается, что CONFIG уже был очищен и бит PTCOM в регистре BPROT сброшен.

```
*ВХОД: A=данные, записываемые в CONFIG
PROGC   LDAB    #$02
        STAB    $103B     установить бит EELAN=1(EEPGM=0)
        STAA    $103F     послать данные по адресу регистра CONFIG
        LDAB    #$03
        STAB    $103B     включить напряжение программирования
        JSR     DLY10     задержка 10мс
        CLR     $103B     выключить напряжение программирования
```

Следующий программный сегмент демонстрирует процедуру стирания регистра CONFIG.

```
BYTEC   LDAB    #$16      установить режим
        STAB    $103B     стирания байта
        STAA    $103F     записать любую информацию в CONFIG
        LDAB    #$17      включить напряжения
        STAB    $103B     программирования
        JSR     DLY10     задержка 10мс
        CLR     $103B     выключить напряжение программирования.
```

1.3.4.5.4.2. Действия при изменении конфигурации системы

Регистр CONFIG состоит из байта EEPROM и статической рабочей защелки. Содержимое байта EEPROM передается в рабочую защелку во время любого сброса. Работа контроллера управляется прямо рабочей защелкой. Изменения в байте EEPROM приведут к

результату только после сброса, так как при программировании CONFIG доступен байт EEPROM, а при чтении CONFIG – статическая защелка.

Для изменения конфигурации системы необходимо:

1) стереть регистр CONFIG;

ВНИМАНИЕ!

Не вызывайте сброс в это время.

2) запрограммировать новое значение в регистре CONFIG;

3) вызвать сброс.

1.3.5. Встроенный АЦП

Универсальный контроллер MC68HC11 имеет встроенный 8-разрядный, 8-канальный аналого-цифровой преобразователь, использующий принцип перераспределения зарядов.

АЦП система контроллера MC68HC11 состоит из одиночного АЦП, 16-канального входного мультиплексора (включая 8 каналов, выведенных на входы контроллера) и схемы управления. Также к схеме управления можно добавить 4 регистра-защелки для хранения результата А/Ц преобразования. Схема управления обеспечивает автоматическое преобразование 4-х каналов одновременно или 4-х последовательных преобразования 1-го канала.

Для тактирования АЦП системы могут использоваться или системный тактовый генератор, или встроенный RC-генератор.

Полное время преобразования для одного канала составляет 32 такта.

После окончания А/Ц преобразования схема управления записывает результат в регистры защелки и устанавливает флажок CCF в регистре управления АЦП системой ADCTL.

Инициализация встроенного АЦП

В однокристальном контроллере MC68HC11E9 по сигналу Reset встроенный АЦП переводится в состояние «выключено». Инициализация и настройка АЦП производится с помощью регистра управления Option, расположенного по адресу 1039h.

Регистр Option (1039h).

Биты	7	6	5	4	3	2	1	0
Обознач.	ADPU	CSEL	IRQE	DLY	CME	-	CR1	CR0
Перв. сост.	0	0	0	1	0	0	0	0

Старший бит регистра Option (ADPU) управляет включением встроенного АЦП. Первоначально он сброшен и АЦП выключен. Перед использованием АЦП необходимо установить бит ADPU в 1. Кроме того, после установки бита ADPU необходима задержка для стабилизации схемы сравнения.

Однокристальный контроллер MC68HC11 позволяет уменьшать частоту тактового генератора, что приводит к уменьшению энергопотребления. При частоте тактового генератора меньше 750 кГц использование его для тактирования АЦП становится невозможным из-за утечки заряда конденсаторов АЦП. Для сохранения работоспособности АЦП при тактовой частоте ниже 750 кГц в АЦП микроконтроллера введен RC-генератор.

Бит CSEL регистра Option управляет подключением системного или встроенного тактового генератора АЦП. При установленном бите CSEL используется RC генератор АЦП. При сброшенном бите CSEL используется системный тактовый генератор. По сигналу Reset бит CSEL сбрасывается.

Система управления встроенным АЦП

Управление системой АЦП контроллера MC68HC11 осуществляется управляющим регистром ADCTL и 4 регистрами данных ADR4 - ADR1.

Регистр ADCTL (1030h).

7	6	5	4	3	2	1	0
CCF	-	SCAN	MULT	CD	CC	CB	CA
0	0	U	U	U	U	U	U

CCF – флаг окончания А/Ц преобразования.

Флаг окончания А/Ц преобразования устанавливается после того, как в регистры ADR4 - ADR1 записан результат А/Ц преобразования.

При каждой записи в регистр ADCTL флаг CCF сбрасывается и начинается новое А/Ц преобразование.

Бит 6 не используется и всегда равен 0.

SCAN – управление режимом сканирования.

Если флажок SCAN = 0, то при каждой записи в регистр ADCTL выполняется А/Ц преобразование и запись результата в регистры ADR4 - ADR1.

Если флажок SCAN = 1, то А/Ц преобразование происходит постоянно, а обновление регистров ADR4-ADR1 – только при изменении значения на канале.

MULT – выбор режима 4 / 1 канал.

Если MULT = 0, то выполняется 4 последовательных А/Ц преобразования для канала, выбранного разрядами CD - CA.

Если MULT = 1, то выполняется А/Ц преобразование для группы из 4-х каналов, выбранных разрядами CD - CC.

CD, CC, CB, CA – выбор канала.

D	CC	CB	CA	Сигнал	Результат
0	0	0	0	PE0	ADR1
0	0	0	1	PE1	ADR2
0	0	1	0	PE2	ADR3
0	0	1	1	PE3	ADR4
0	1	0	0	PE4	ADR1
0	1	0	1	PE5	ADR2
0	1	1	0	PE6	ADR3
0	1	1	1	PE7	ADR4
1	0	0	0	Резерв	ADR1
1	0	0	1	Резерв	ADR2
1	0	1	0	Резерв	ADR3
1	0	1	1	Резерв	ADR4
1	1	0	0	Vh	ADR1
1	1	0	1	Vl	ADR2
1	1	1	0	1/2 Vh	ADR3
1	1	1	1	Резерв	ADR4

Разряды управляют выбором каналов, которые будут участвовать в А/Ц преобразовании.

Регистры ADR4 - ADR1 (1034h - 1031h).

Регистры ADR4 - ADR1 используются для хранения результата А/Ц преобразования и допускают только чтение.

2. ЛАБОРАТОРНЫЕ РАБОТЫ

2.1. ИЗУЧЕНИЕ АРХИТЕКТУРЫ, СТРУКТУРЫ МЕТОДОВ ПРОГРАММИРОВАНИЯ МИКРОКОНТРОЛЛЕРА MC68HC711E9 (ЛАБОРАТОРНЫЕ РАБОТЫ 1–5)

Этапы прохождения задач пользователя с выполнением программы микроконтроллером приведены на рис. 7.

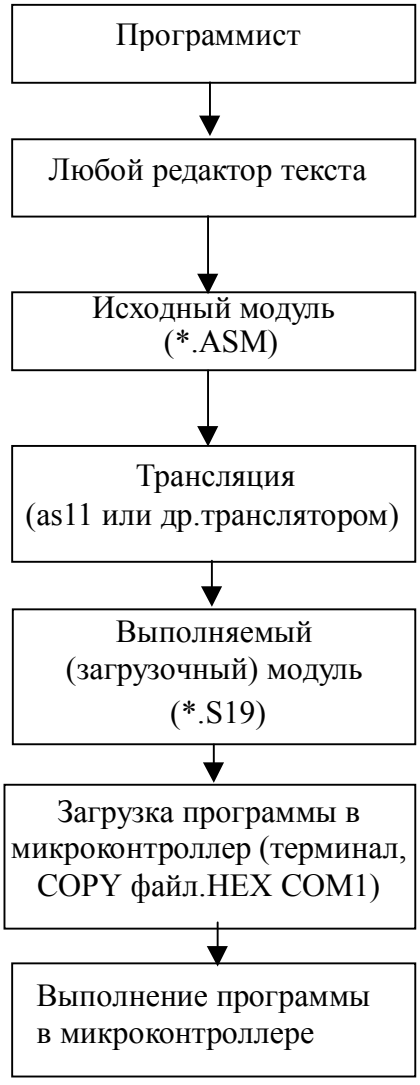

Рис. 7. Этапы прохождения задач пользователя

Получение загрузочного файла для MCU Motorola

Загрузочным модуль для MCU Motorola является файл в формате S19 (см. приложение 5). Для его получения используются различные трансляторы, один из них, AS11.EXE, используется в данных лабораторных работах. Исходный текстовый файл программы создается любым текстовым редактором и имеет расширение .ASM. Для получения файла в формате S19 необходимо запустить AS11 и в качестве параметра указать имя исходного файла.

Пример:

AS11.EXE testcom.asm

Запись программы пользователя в ОЗУ микроконтроллера

Для выполнения программ пользователя и облегчения режима их отладки используются отладчики PCBUG11 (см. приложение 4) и BUFFALO. В качестве терминала для работы с программой-монитором BUFFALO можно использовать программу-терминал KERMIT (см. M68HC11EVB User's Manual, M68HC11EVBU User's Manual, там же описана работа с отладчиком BUFFALO).

Для записи программы пользователя в ОЗУ микроконтроллера, которое начинается с адреса $0000 и заканчивается адресом $01FF (см. карту памяти), можно использовать разработанную в учебном центре программу-загрузчик UPLOAD.EXE, текст которой приведен в приложении 4. В качестве параметра программе UPLOAD передается исполнимый файл в формате S19 и номер COM порта.

Пример:

UPLOAD.EXE testcom.s19 –2,

Где параметр «–2» означает, что программа должна использовать COM2.

Запись программы в EEPROM микроконтроллера

Для записи программы в EEPROM микроконтроллера и ее запуска используется отладчик PCBUG11 (список команд, см. приложение 3). Для его запуска необходимо нажать кнопку RESET на стенде и затем запустить командный файл RAMTALK.BAT.

Командный файл RAMTALK.BAT имеет следующий вид:

PCBUG11.EXE -E MACRO=GL port=2

Для записи необходимо присвоить EEPROM адреса B600-B7FF, разрешить запись в нее и загрузить программу в S-формате,

являющуюся результатом компиляции исходного текста на ассемблере. Все эти действия выполняет макрос PROG, находящийся в библиотеке макросов GL для PCBUG11.

Библиотека макросов GL. MCR

```
DEFM AUTOSTART
BEGIN
  CONTROL BASE HEX
END
DEFM LIST
BEGIN
  db 100 1FF
END
DEFM EDIT
BEGIN
  mm @0
END
DEFM Q
BEGIN
  QUIT Y
END
DEFM PROG
BEGIN
  MS $1035 0
  EEPROM $B600 $B7FF
  LOADS @0
  VERF @0
END
DEFM L
BEGIN
  DB 100 10F
END
DEFM CLEAR
BEGIN
  BF 100 180 01
  CLS
END
DEFM TM
BEGIN
  call 10b
  db 100
END
```

2.1.1. ЛАБОРАТОРНАЯ РАБОТА № 1
СРЕДСТВА РАЗРАБОТКИ И ОТЛАДКИ СИСТЕМ НА БАЗЕ MC68HC711E9

Целью данной работы является изучение:

- комплекса программно-аппаратных средств, предназначенного для отладки программ и схемотехники микроконтроллеров семейства MC68HC11 фирмы «Motorola»;
- структурной схемы комплекса для отладки программ и схемотехнических устройств на базе МК семейства MC68HC11 на базе оценочной платы M68HC11EVBU/EVB (рис 8.).

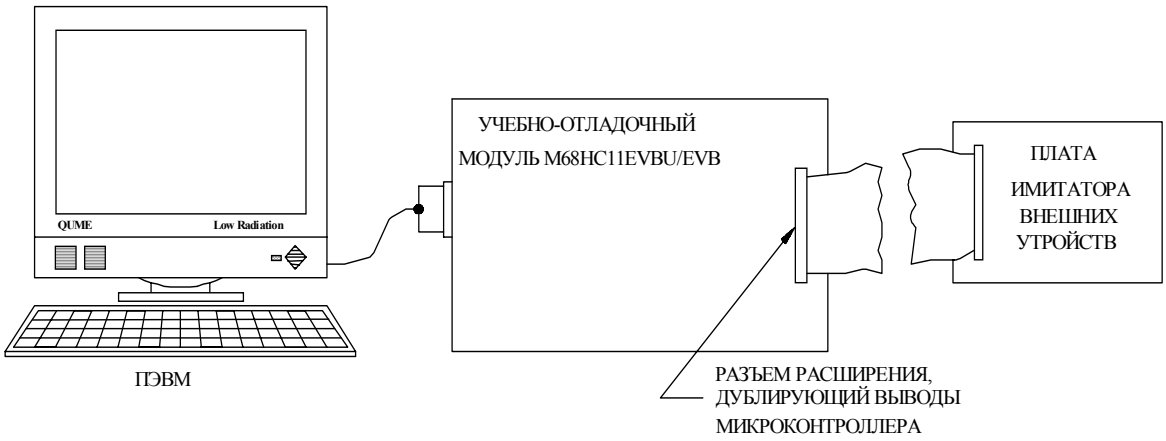

Рис. 8. Структура учебно-отладочного комплекса.

Задание для домашней подготовки

Изучить структуру оценочных плат, команды отладочных мониторов PCBUG11, Buffalo, работу с терминальными программами типа KERMIT (см. описание PCBUG11, M68HC11EVB User's Manual, M68HC11EVBU User's Manual, там же описана работа с отладчиком BUFFALO).

Задания к лабораторной работе

Получить задание у преподавателя и выполнить указанные в нем действия по исследованию элементов структуры микроконтроллера и исследованию команд отладчиков.

Контрольные вопросы

1. Каковы функции директив отладчика PCBUG11
2. Какие символы расширяют возможности команды RM?
3. Каковы функции директивы T?
4. Что представляют из себя данные, выводимые директивой ASM?
5. Какой командой выполняется просмотр содержимого регистров микроконтроллера?
6. Каковы основные режимы выполнения программ?
7. Каковы функции директивы BF?
8. Назовите основные части оценочных плат M68HC11EVBU/EVB.
9. Какие директивы кросс-ассемблера используются для резервирования слова?
10. Каким образом осуществляется загрузка программы в память МК?
11. Какой формат имеют строки программы для кросс-ассемблера?
12. Каковы функции директивы LOAD? Каков ее формат?
13. Какие особенности имеет МК MC68HC711E9?
14. Какая информация может содержаться в поле команды?
15. Как производится временная смена системы счисления в кросс-ассемблере?
16. Каковы функции директивы T?
17. Каким образом в кросс-ассемблере устанавливается адрес начала программы?

2.1.2. ЛАБОРАТОРНАЯ РАБОТА № 2 ВНУТРЕННЯЯ АРХИТЕКТУРА. МЕТОДЫ АДРЕСАЦИИ. КОМАНДЫ ПЕРЕСЫЛКИ ДАННЫХ. КОМАНДЫ ПЕРЕДАЧИ УПРАВЛЕНИЯ

Целью данной работы является изучение:

- внутренней архитектуры (набор регистров, встроенная память);
- методов адресации MC68HC11;
- группы команд пересылки данных;
- группы команд передачи управления.

Задание для домашней подготовки

Пользуясь электронным учебником «MC68HC11E9-HCMOS Single-Chip Microcontroller», изучить:

- внутреннюю архитектуру (набор регистров, встроенная память);
- методы адресации MC68HC11;
- группы команд пересылки данных;
- группы команд передачи управления.

Задания к лабораторной работе

1. Перед началом лабораторных работ сдать коллоквиум по темам заданий для домашней подготовки.
2. Разработайте программу сортировки ячеек памяти $0090-$00BF по возрастанию.
3. Разработайте программу подсчета количества ячеек памяти со значениями, отличными от $38.
4. Напишите программу подсчета количества ячеек памяти, значения которых меньше или равны $A3, причем считается, что числа знаковые.
5. Разработайте программу подсчета количества отрицательных и положительных чисел, при этом считайте, что ноль нейтрален.
6. Составьте программу подсчета статистики чисел по отношению к числу $31, т. е. в одну ячейку памяти записывайте количество чисел равных, в другую меньших, в третью больших данного числа.
7. Пусть в ячейках памяти $80, $81 и $82 находятся соответственно переменные A, B и C. Если они упорядочены по возрастанию, то

заполните ячейки памяти $90-$B0 значением $55, в противном случае заполните указанные ячейки значением $66.

8. Пусть в ячейках памяти $80, $81, $82 и $83 находятся соответственно переменные A, B, C и D. Составить программу, заполняющую ячейку $90 различными значениями в соответствии с таблицей:

Условие			Содержимое ячейки $90
1	2	3	
A≥B	C≥D	B<D	1
		B≥D	2
	C<D	B<C	3
		B≥C	4
A<B	-	-	5

9. Пусть в ячейках памяти $80, $81, $82 и $83 находятся соответственно переменные A, B, C и D. Составить программу, заполняющую ячейку $90 различными значениями в соответствии с таблицей:

Условие			Содержимое ячейки $90
1	2	3	
A>C	B>D	B≤C	0
		B>C	1
	B≤D	A>D	1
		A≤D	2
A≤C	-	-	3

10. Пусть в ячейке $80 находится переменная A. Расписать ее значение побитно в ячейках $90-$97. Команды тестирования битов не использовать.
11. Пусть в памяти находится строка, оканчивающаяся нулем. Разработайте программу подсчета длины строки.
12. Пусть в ячейке $80 находится некоторое беззнаковое число. Проверить, делится ли это число нацело на $03?
13. Пусть в ячейке $80 находится некоторое знаковое число. Проверить, делится ли это число нацело на $0A?
14. Составьте программу сортировки по убыванию.
15. Напишите программу, которая определяет адрес своего первого байта. Считайте, что полезной информации в области стека не накоплено.

Контрольные вопросы

1. В чем заключается отличие команд BRA и JMP?
2. В чем состоит отличие команд BSR и JSR?
3. В чем заключается сходство команд LDA и TXA?
4. В чем состоит отличие команд BRSET и BRCLR?
5. Сколько байт в памяти занимает команда BRN?
6. В чем заключается преимущество команды BRN перед двумя командами NOP?
7. Какие команды перехода влияют на флаги?
8. Какие ошибки допущены при написании фрагмента программы:

```
020E    BRA     $0100
020F    BRCLR   #8, *$22, $0189
```

9. Какая команда проверяет флаг Z на равенство «0»?
10. Какие команды из изученных в данной лабораторной работе взаимодействуют с флагом Z?
11. Какие ошибки допущены при написании фрагмента программы:

```
0100    TAX
0101    LDA     $00, X
0102    RSPA
0103    BRS     $F0, X
```

12. Какие режимы адресации используют команды BRN, LDX, RSP? Каковы пределы адресации для различных режимов адресации?
13. Какие действия выполняет команда RTS?
14. Какие действия выполняет команда BSR?
15. В чем состоит преимущество команды JMP перед командой BRA?
16. Можно ли командой перехода изменить флаг Z?
17. Какие действия выполняет команда TXA?
18. В чем заключается отличие команд BCS и BLO?
19. Существует ли разница между командами BHCC и BCC?
20. Каково значение стека, если Вы находитесь в подпрограмме?
21. Можно ли командами BHI и BHS смоделировать (приблизительно) работу команды BNE? Если нельзя, то почему, а если можно, то как?
22. Какие из приведенных команд взаимодействуют с флагом C: BLO, BRCLR, BLS, BRA, BSR, BHCS, TAX?
23. В чем состоит сходство команд BCS и LDA в формате непосредственной адресации?
24. Какие типы адресации могут использовать команды JMP, BRSET, BNE, LDA, TXA, RSP, LDX?

2.1.3. ЛАБОРАТОРНАЯ РАБОТА № 3 АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ КОМАНДЫ КОМАНДЫ РАБОТЫ С БИТАМИ

Целью данной работы является изучение:

- арифметических команд,
- логических команд,
- битовых операций.

Рассматриваемые здесь команды позволяют выполнять арифметические, логические операции, сдвигать и сравнивать содержимое ячеек памяти.

Задание для домашней подготовки

Пользуясь электронным учебником «MC68HC11E9 – HCMOS Single-Chip Microcontroller», изучить:

- арифметические команды,
- логические команды,
- битовые операции

Задания к лабораторной работе

1. Перед началом лабораторных работ сдать коллоквиум по темам заданий для домашней подготовки.
2. Написать программу сложения или вычитания двух n-байтовых чисел, где число n будет являться параметром.
3. Написать программу, вычисляющую сумму чисел от 1 до n с помощью цикла.
4. Написать программу, вычисляющую первые n чисел Фибоначчи. (Т. е. последовательность, в которой каждый последующий член является суммой двух предыдущих: 0, 1, 1, 2, 3...)
5. Написать программу нахождения наибольшего общего делителя по алгоритму Евклида. (Берется исходная пара чисел, из большего числа вычитается меньшее, потом меньшее число из первой пары и разность первой пары становится следующей парой. Алгоритм повторяется снова до тех пор, пока в паре не окажутся одинаковые числа. Это и будет наибольшим общим делителем.)
6. Написать программы умножения и деления, используя команды сложения и вычитания.
7. Написать программу поиска максимального элемента в массиве.

Реализовать логические операции, таблицы истинности которых приведены ниже:

«стрелка Пирса»

A	B	A↓B
0	0	1
0	1	0
1	0	0
1	1	0

«штрих Шеффера»

A	B	A | B
0	0	1
0	1	1
1	0	1
1	1	0

1. Написать программу, записывающую последовательность бит в аккумуляторе в обратном порядке (00100111→11100100).
2. Написать программу, записывающую в старшую тетраду аккумулятора зеркальное отображение младшей (11110011→ 11000011).
3. Написать программу, осуществляющую преобразование числа в его символьное представление в коде ASCII в восьмиричной системе счисления.
4. Написать программу, осуществляющую преобразование числа в его символьное представление в коде ASCII в десятичной системе счисления. Указание: воспользоваться подпрограммой деления.
5. Написать программу, осуществляющую преобразование числа из его символьного представления в коде ASCII в восьмиричной системе счисления.
6. Написать программу, которая копирует младшие биты байтов $50-$57 в биты байта $58; младший бит байта $58 соответствует биту, взятому из байта $50.
7. Написать программу, инвертирующую бит n в ячейке $dd, где **n** передается в аккумуляторе, а адрес $dd – в регистре X.
8. Написать программу, расширяющую значение байта $50 в 4 последующих байта следующем образом: тетрада каждого из этих четырех байт равна $F, если установлен соответствующий бит в ячейке $50 и $0, если бит сброшен.

Контрольные вопросы

1. Объясните логику работы команд сложения и вычитания с учетом флага C регистра CC при обработке многобайтовых чисел.
2. Опишите работу команд, выполняющих операцию циклического сдвига аккумулятора.
3. Придумайте не менее двух способов инвертирования содержимого аккумулятора.

4. Придумайте по два способа занесения в аккумулятор чисел $00 и $FF, используя только логические команды.
5. Дайте описание и приведите пример использования команды ASR.
6. Каким образом можно обращаться к битам и работать с ними без использования команд работы с битами?
7. Опишите способы адресации команд ADC, ADD, SUB, SBC, AND, ORA, EOR, CMP, CPX.
8. Опишите способы адресации команд LSL, LSR, ASL, ASR, ROL, ROR COM, DEC, INC, NEG, TST, BIT.
9. Опишите способы адресации команд BSET, BCLR.
10. Опишите способы адресации команды MUL.
11. Почему в программе из примера для команд ASL, ASR правильными будут значения в ячейках $52 и $53, а неправильными – в ячейке $51?
12. Как можно организовать циклический сдвиг многобайтового слова?
13. Почему программы для нахождения модуля числа (пример 1 для команд сравнения, пример 2 для команд тестирования) не будут находить модуль числа $80 (-128)?
14. Сколько байт в памяти могут занимать команды ADD, LSL, MUL, BCLR?
15. Чем отличаются команды ASR и LSR?
16. Чем отличаются команды NEG и COM?
17. Чем отличаются команды LSL и ASL?
18. Чем отличаются команды LSL и ROL?
19. Как работают команды CMP, BIT, TST?
20. Для чего чаще всего используются команды INC и DEC?

2.1.4. ЛАБОРАТОРНАЯ РАБОТА №4
СИСТЕМА ПРЕРЫВАНИЙ. СПЕЦИАЛЬНЫЕ РЕЖИМЫ РАБОТЫ

Целью данной работы является изучение 18 типов прерываний, включающих 17 аппаратных прерываний и одно программное прерывание.

Задание для домашней подготовки

Пользуясь электронным учебником «MC68HC11E9 – HCMOS Single-Chip Microcontroller» изучить 18 типов прерываний, включающих 17 аппаратных прерываний и одно программное прерывание.

Задания к лабораторной работе

1. Перед началом лабораторных работ сдать коллоквиум по теме задания для домашней подготовки.
2. Написать программу, которая при первом нажатии на кнопку IRQ увеличивает содержимое порта B на единицу, при втором – на два и т. д.
3. Написать программу, которая по каждому второму нажатию кнопки IRQ сдвигает содержимое порта B на один разряд вправо.
4. Написать программу, которая по нажатию кнопки IRQ циклически сдвигает содержимое порта B на один разряд влево.
5. Написать программу, которая по трехкратному нажатию на кнопку IRQ инвертирует состояние светодиодов.
6. Написать программу, которая подсчитывает число нажатий (от 0 до 65535) на кнопку IRQ в двух ячейках памяти.
7. Переделать программу вычисления факториала так, чтобы результат вычислений был двухбайтным.
8. Пользуясь командой SWI, написать программу, которая записывает в регистры A и X начальный адрес своего местоположения в памяти.
9. Написать программу, которая по первому нажатию на кнопку IRQ считывает первое значение с переключателей, по второму – второе, и затем выводит сумму на светодиоды.
10. Используя механизм подмены векторов прерываний, написать программу, которая при нажатии на кнопку IRQ первый раз сдвигает информацию на светодиодах на один бит влево, второй

раз – вправо, а при нажатии на IRQ в третий раз инкрементирует порт B.

11. Напишите программу, которая мигает левым светодиодом порта B с частотой 1–2 Гц, а по нажатию кнопки IRQ считывает с переключателей адрес и заполняет 10 ячеек ОЗУ, начиная с этого адреса, последовательно числами от 0 до 9. Предусмотрите проверку вводимого адреса на то, что он лежит в пределах $50..$B0, в противном случае запись не производите.
12. Написать программу, которая при включении левого переключателя порта A начинает подсчет количества нажатий на кнопку IRQ, а при выключении – выводит количество нажатий на светодиоды.
13. Написать программу для кодового замка, где шифр (произвольное число от 0 до 255) набирается на переключателях порта A, а для ввода шифра используется кнопка IRQ. Сначала программа должна погасить светодиоды порта B, после чего, в случае правильного набора, должны зажечься все светодиоды, в противном случае светодиоды должны мигать с частотой 1–2 Гц.
14. Написать программу, которая по первому нажатию на кнопку IRQ считывает первое значение с переключателей, по второму – второе, а по третьему нажатию выводит разность на светодиоды.
15. Написать программу, которая в цикле вызывает программное прерывание SWI и инкрементирует счетчик после каждого вызова. Запустив эту программу на 10 сек., оцените время выполнения команд SWI, RTI.

Контрольные вопросы

1. Какие источники прерываний имеются в МК MC68HC711E9?
2. Какие действия выполняет МК при возникновении прерывания?
3. Опишите порядок загрузки регистров в стек при обслуживании прерывания.
4. Каковы условия возникновения программного прерывания?
5. Каковы условия возникновения внешнего аппаратного прерывания?
6. Каковы условия возникновения прерываний от внутренних устройств МК?
7. Чем определяется реакция МК на входной сигнал по линии IRQ?
8. В каких случаях происходит начальная установка микроконтроллера?
9. Кратко опишите процесс начальной установки микроконтроллера.

10. Опишите режим пониженного энергопотребления STOP.
11. Опишите режим пониженного энергопотребления WAIT.
12. Опишите режим сохранения данных.
13. По каким адресам расположена таблица векторов прерываний и каково ее назначение?
14. Перечислите адреса векторов прерываний SWI, IRQ, RESET.
15. Объясните механизм вызова подпрограммы обработки прерывания.
16. Каким образом после возврата из подпрограммы обработки прерывания управление передается на точку, в которой возникло прерывание?
17. Какие подготовительные действия необходимо произвести, чтобы программа, использующая прерывание IRQ, работала правильно?
18. В чем отличие между командами SWI и JSR, и что между ними общего?
19. Для чего предназначен стек, и как он организован?
20. В какой области памяти расположен стек, какова его максимальная длина?
21. Какие различия существуют в реакции ОЭВМ на прерывание RESET и на остальные прерывания?

2.1.5. ЛАБОРАТОРНАЯ РАБОТА № 5
ТАЙМЕР И СВЯЗАННЫЕ С НИМ УСТРОЙСТВА

Целью данной работы является изучение таймерной системы, включающей три регистра захвата и пять регистров сравнения.

Задание для домашней подготовки

Пользуясь электронным учебником «MC68HC11E9-HCMOS Single-Chip Microcontroller», изучить раздел 8 «Программируемый таймер, прерывания реального времени и импульсный аккумулятор».

Задания к лабораторной работе

1. Перед началом лабораторных работ сдать коллоквиум по темам заданий для домашней подготовки.
2. Получить задание у преподавателя и написать программу обработки заданного типа прерываний.
3. Используя функцию OC, организовать прерывания OCI с периодом в 20 мс (10000 тактов таймера) и на их основе написать программу, которая мигает светодиодом с частотой 2 Гц. С помощью секундомера убедитесь в точности отсчета времени в программе.
4. Написать программу, демонстрирующую работу таймера COP миганием светодиода, при срабатывании таймера COP.
5. Напишите программу, которая последовательно считывает значения младшего байта регистра счетчика таймера и заполняет ими 20 ячеек ОЗУ, начиная с $50. Зная время выполнения команд HC05, по содержимому ячеек вычислите частоту тактирования таймера и сравните ее с теоретическим значением.

Контрольные вопросы

1. Для каких целей используется блок таймера в составе микроконтроллера MC68HC705C8?
2. Каковы основные характеристики блока таймера?
3. Каким образом осуществляется доступ к таймеру?
4. Для чего используется счетчик таймера?
5. Каким образом осуществляется тактирование счетчика?
6. В каком случае устанавливается флаг TOF?
7. Для чего необходим альтернативный счетчик таймера?
8. В каком случае младший байт регистра счетчика запоминается в буфере?

9. Опишите назначение и принцип работы блока входной фиксации.
10. Каким образом задается способ обработки сигнала на входе?
11. В каком случае блокируется работа блока входной фиксации?
12. Каким образом можно использовать функцию входной фиксации для определения периода и ширины импульса входного сигнала?
13. Опишите назначение и принцип работы блока выходного сравнения.
14. Каким образом задается логический уровень сигнала, появляющегося на выходе TCMP, при совпадении значения счетчика с содержимым регистра OCR?
15. В каком случае блокируется работа блока выходного сравнения?
16. Каким образом можно использовать функцию выходного сравнения для генерирования импульса определенной длительности?
17. Какие флаги входят в состав управляющего регистра таймера? На что они влияют?
18. Какие флаги входят в состав регистра статуса таймера? Что показывает каждый из них? Каким образом можно изменять их состояние?
19. Каково основное назначение таймера COP?
20. Для чего используется диспетчер синхронизации?
21. Каким образом осуществляется управление таймером COP и диспетчером синхронизации?
22. Опишите последовательность действий для сброса таймера COP?
23. Какие флаги входят в состав управляющего регистра таймера COP? На что они влияют и как их можно изменять?
24. Чем определяется период переполнения таймера COP?

2.2. Доработка платы M68HC11EVBU (лабораторные работы 6,7)

В первом варианте плата была дополнена ЖКИ.

Для отображения информации был выбран жидкокристаллический индикатор WH1602. Структурная схема индикатора и схема его подключения к микроконтроллеру приведена на рис. 9.

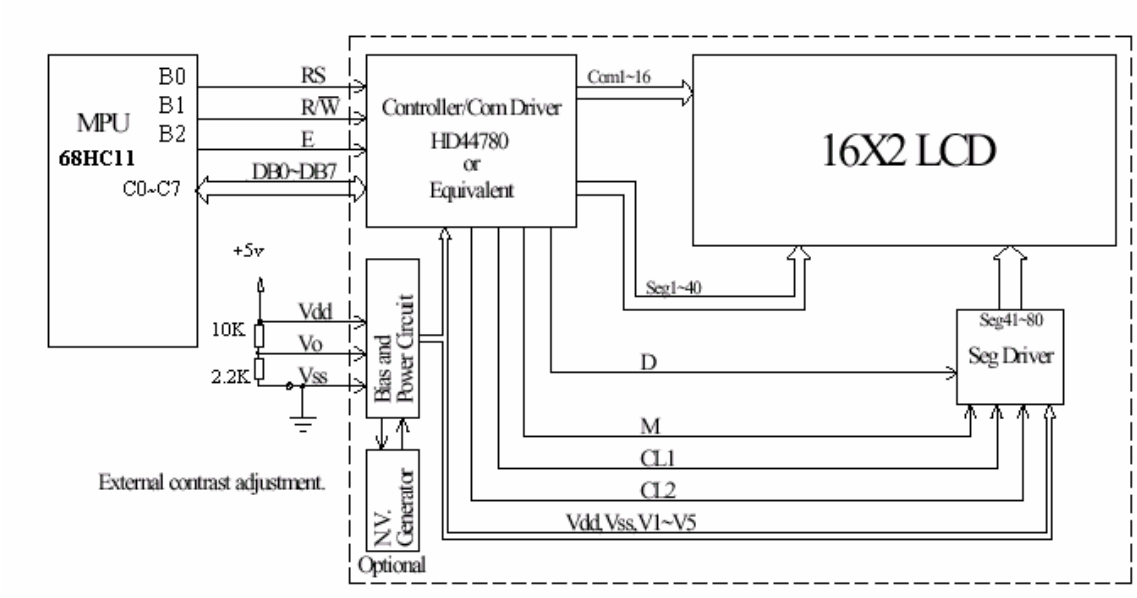

Рис. 9. Первый вариант доработки платы M68HC11EVBU

PIN NO.	SYMBOL
1	Vdd
2	Vss
3	Vo
4	RS
5	R/$\overline{W}$
6	E
7	DB0
8	DB1
9	DB2
10	DB3
11	DB4
12	DB5
13	DB6
14	DB7

Рис. 10. Назначение
выводов индикатора

Назначение выводов индикатора приведено на рис. 10

Микросхема WH1602 – это жидкокристаллический дисплейный модуль, который отображает две строки символов. Каждый символ может использовать 5x10 точек матрицы, из которых 5x7 разрешается выбирать программно.

Для ввода/вывода применяется восьмиразрядная двунаправленная шина данных выводы DB0-DB7. Кроме того, при организации ввода/вывода необходимо управлять следующими сигналами R/W(чтение из микросхемы /запись в микросхему), E – разрешение работы дисплея, RS-сброс.

Дисплей имеет набор инструкций по выполнению различных функций, некоторые из них приведены в табл. 5.

Таблица 5

Набор инструкций жидкокристаллического дисплея

Инструкции	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Очистка дисплея	0	0	0	0	0	0	0	0	0	1
Возврат в начало	0	0	0	0	0	0	0	0	1	0
Установка режима	0	0	0	0	0	0	0	1	1/D	S
Вкл./выкл. дисплея	0	0	0	0	0	0	1	D	C	B
Сдвиг курсора	0	0	0	0	0	1	S/C	R/L	0	0
Запись данных	1	0	Байт данных(ASCII символ)							
Чтение флага ЗАНЯТ	1	1	BF	AC						

Условные обозначения в таблице:
I/D=1: увеличение, I/D=0:уменьшение;
S=1: относительный скроллинг;
S/C=1: скроллинг;
S/C=0: перемещение курсора;
R/L=1: сдвиг вправо;
R/L=0: сдвиг влево.[16]

Во втором варианте плата была дополнена ЖКИ, динамиком и светодиодом (рис. 11)

Третий вариант доработки представляет систему измерения температуры на базе SPI интерфейса. Структурная схема разработанной измерительной системы приведена на рис. 12.

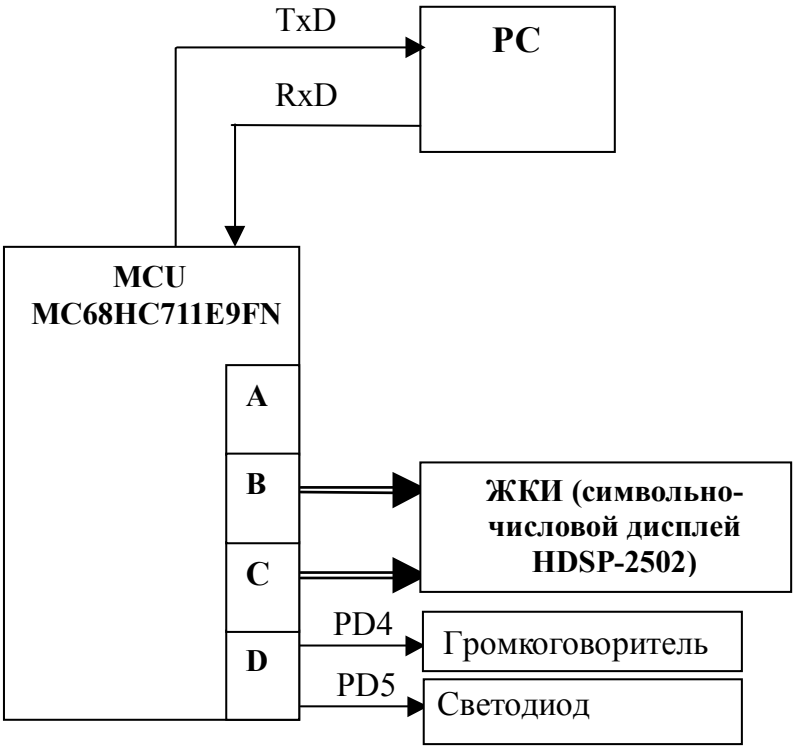

Рис. 11. Структура комплекса на базе платы M68HC11EVB

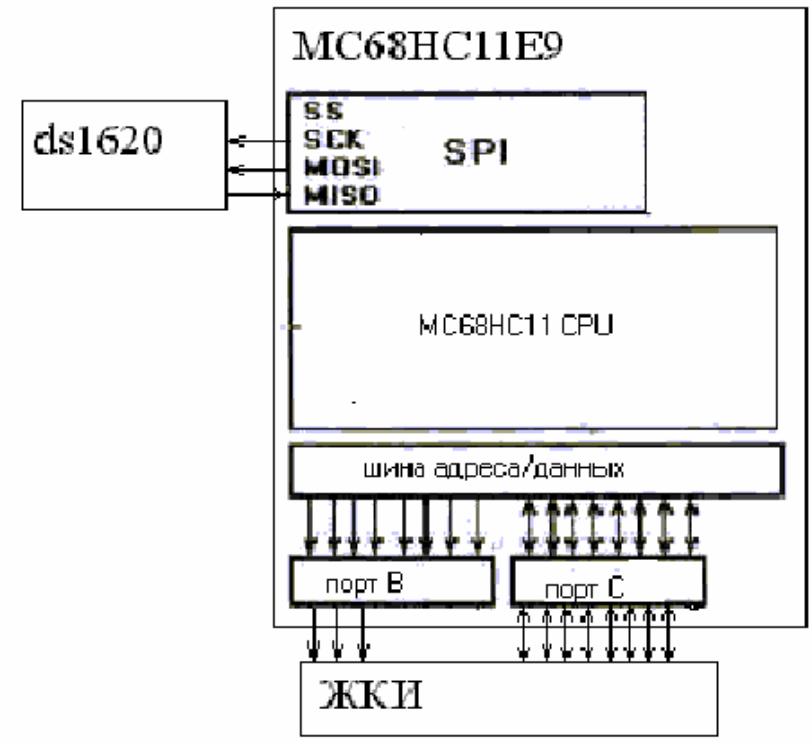

Рис. 12. Второй вариант доработки платы M68HC11EVBU

Основным элементом схемы является микроконтроллер, выполняющий функции обработки поступающей информации и управления для всех остальных узлов системы. В состав устройства также входят:

- датчик температуры DS1620 подключается к выводам SCK, MOSI, MISO, SS микроконтроллера;
- жидкокристаллический индикационный модуль WH1602 для отображения полученных результатов.

Для управления работой SPI порта в составе MC68HC11E9 служат три специальных регистра: SPCR-регистр управления работой SPI порта, адрес во внутреннем адресном пространстве микроконтроллера (1028h). Его формат следующий:

SPIE	SPE	DWON	MSTR	CPOL	CPHA	SPR1	SPR0
7	6	5	4	3	2	1	0

SPIE – разрешение прерываний от SPI;

SPE – разрешение работы SPI;

DWON – включение режима монтажного или на выходах порта;

MSTR – включение режима ведущий:

0 – ведущий (Master);

1 – ведомый (Slave);

CPOL – полярность синхроимпульсов.

Когда бит CPOL очищен и данные не передаются, на выводе SCK устройства MASTER присутствует напряжение низкого уровня. Если этот бит установлен, то в бездействующем состоянии на контакте SCK будет напряжение высокого уровня. Этот бит также используется совместно с битом CPHA для задания требуемых временных диаграмм работы при соединении устройств MASTER и SLAVE.

CPHA – фаза синхроимпульсов. Бит CPHA совместно с битом CPOL управляет сигналом синхронизации.

SPR1 и SPR0 – выбор скорости передачи.

Эти два бита выбирают одну из четырех частот (см. табл. 6), использующуюся как SCK в устройстве MASTER. В режиме SLAVE эти биты не имеют значения.

Таблица 6

SPR1	SPR0	Внутренняя частота деленная
0	0	2
0	1	4
1	0	16
1	1	32

Регистр статуса SPSR(1029h) предназначен для контроля за работой SPI порта, его формат следующий:

SPIF	WCOL	0	MODF	0	0	0	0
7	6	5	4	3	2	1	0

SPIF – флаг окончания передачи между процессором и внешним устройством;

WCOL – флаг столкновения записей;

MODF – ошибка режима.

Регистр данных SPDR(102Ah), непосредственная запись в который инициирует передачу по SPI шине записанных данных [9].

Датчик температуры

В качестве измерителя температуры была выбрана микросхема DS1620 (рис. 13)

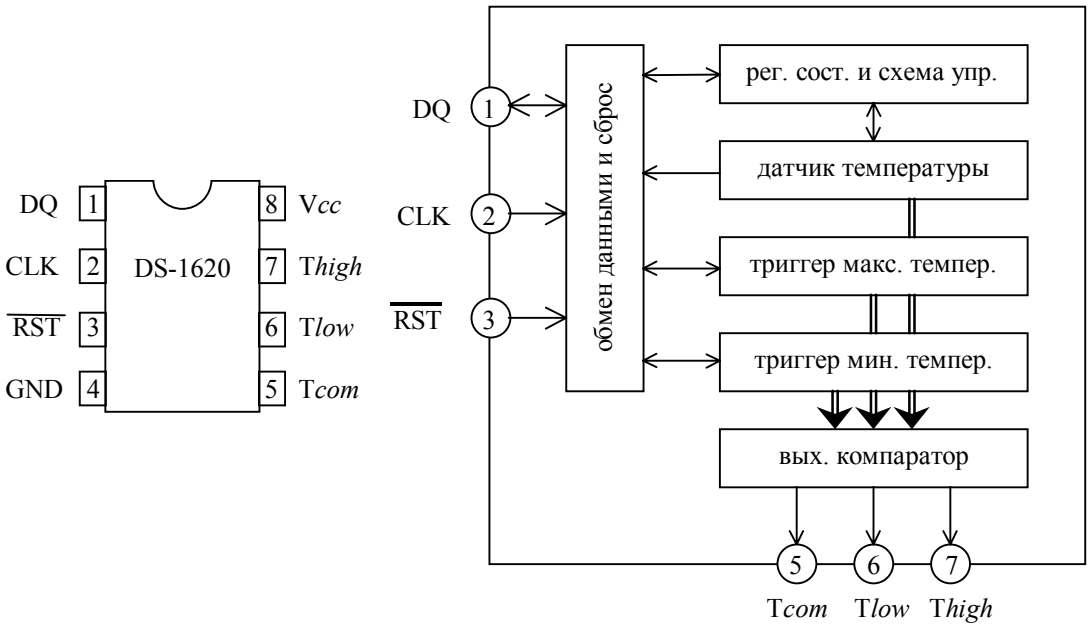

Рис. 13. Назначение выводов и внутренняя блок-схема DS1620

Это девятиразрядный термометр и термостат, служащий для измерения и отображения температуры. Он имеет три выхода, которые используются при работе микросхемы в режиме термостата. Настройки выходного сигнала можно запрограммировать и сохранить во внутренней энергонезависимой памяти. Устройство измеряет температуру от –55 до +125 ºС с шагом 0.5 ºС, преобразование занимает 1с.

Передача данных от микросхемы к внешнему устройству осуществляется по трехпроводной последовательной шине: CLK/CONV (контакт 2), DQ (контакт 1) и RESET (контакт 3). Эти выходы совместимы с уровнями ТТЛ. Thigh (контакт 7) – выход триггера высокой температуры. Если температура превышает установленный верхний порог, то выход Thigh сигнализирует об этом высоким уровнем и остается в током состоянии до тех пор, пока температура не упадет ниже заданного порога. Tlow (контакт 6) – выход триггера низкой температуры. Если температура опускается ниже заданного порога, то на нем появляется сигнал высокого уровня, сохраняющихся в таком состоянии, пока температура не поднимется выше указанного предела. Tcom (контакт 5) – это выход комбинированного триггера высокой и низкой температуры. Tcom =1, когда температура превышает верхний предел, Tcom =0, когда она опускается ниже нижнего предела. Контакты 4 и 8 соединены с отрицательным и положительным проводами источника питания. Потребляемый ток в режиме ожидания равен 1 мкА, в рабочем режиме – 1мА.

Управление устройством осуществляется в два этапа: сначала команды управления последовательно загружаются в микросхему, а затем девятиразрядное число, соответствующее температуре, либо считывается, либо записывается. Микросхема имеет девять команд:

○ Read temp (Aah): чтение значения регистра, содержащего результат последнего измерения – 9 бит данных ;

○ Start conversion T (EEh): запуск процесса измерения температуры. Данные не передаются.

○ Stop conversion T (22h): остановка измерения. Данные не передаются.

○ Write TH (01h): запись верхнего предела в триггер высокой температуры – 9 бит данных

○ Write TL (02h): запись нижнего предела в триггер низкой температуры – 9 бит данных.

○Read TH (A1h): чтение содержимого триггера высокой температуры – 9 бит данных.

○ Read TL (A2h): чтение содержимого триггера низкой температуры – 9 бит данных

○Write configuration (0Ch): запись настроечных данных в регистр настройки – 8 бит данных.

○ Read configuration (Ach): чтение настроечных данных из регистра настройки – 8 бит данных.

Настроечное слово управляет режимами работы микросхемы DS1620. Оно сохраняется в регистре настройки, Функции битов регистров приведены ниже:

DONE	THF	TLF	X	X	X	CPU	1SHOT

(X – любое)

DONE 0=идет преобразование,

1=преобразование завершено;

THF – флаг высокой температуры. Если температура равна или выше верхнего предела, то бит THF=1. Он остается в единичном состоянии до тех пор, пока его не сбросят, записав ноль, или не отключат питание устройства;

TLF – флаг низкой температуры. Если температура равна или ниже нижнего предела, то бит TLF =1. ОН остается в единичном состоянии до тех пор, пока его не сбросят в ноль или не отключат питание;

CPU – если CPU = 0, то вход CLK/CONV управляет началом цикла измерения температуры, в противном случае микросхема работает в режиме обмена информацией с внешним устройством;

1SHOT – если 1SHOT=1, микросхема производит один цикл измерения температуры после поступления команды, в противном случае ИС настроена на непрерывное измерение температуры.

Данные о температуре имеют девятибитовый формат. Дискретность преобразования температуры равна 1/2 ^{0}C. Некоторые соотношения между значениями температуры и выходными данными:

+125 ºC	0	11111010 (0FFh)
+25 ºC	0	00110010 (0032h)
+1.2 ºC	0	00000001 (01h)
0 ºC	0	00000000 (00h)
–1.2 ºC	1	11111111 (01FFh)
–25 ºC	1	11001110 (01CEh)

Передача данных осуществляется при поступлении положительного фронта на вход RST (контакт 3). Если на вход подать 0, то передача прекращается. Процессами чтения и записи управляет тактирующий вход микросхемы. Один тактовый цикл состоит из отрицательного фронта и следующего за ним положительного фронта.

При записи состояние битов должно оставаться неизменным во прохождении положительного фронта.

При считывании данные выводятся из устройства по отрицательному фронту тактовых импульсов. Когда на тактовом входе высокий уровень, выход DQ (контакт 1) имеет высокое сопротивление. При чтении данных младший бит передается первым. Через этот контакт можно как читать, так и передавать данные [13], [16].

Четвертый вариант доработки представляет систему многоканального ввода информации. Структуру системы многоканального ввода информации можно представить следующим образом (см. рис. 14):

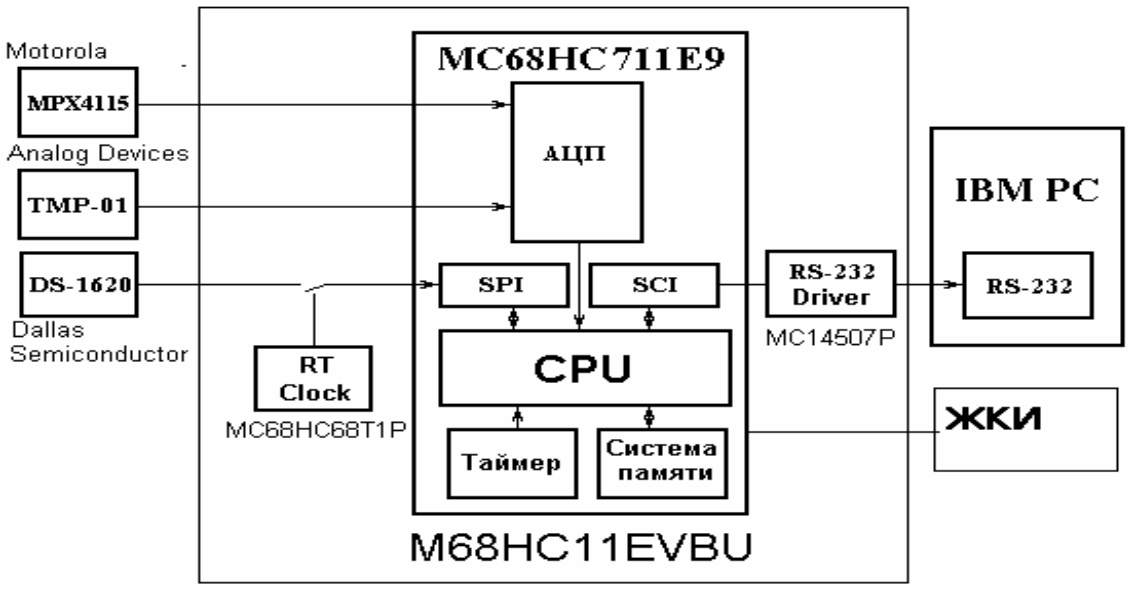

Рис. 14. Функциональная блок-схема устройства многоканального ввода

Устройство работает следующим образом:

- считывает данные с датчиков температуры DS – 1620 и давления MPX4115
- система RTI через заданные промежутки времени генерирует запрос прерывания;
- по запросу прерывания процессор инициирует АЦ преобразование;
- считывает данные из АЦП;
- данные через систему SCI могут быть переданы в компьютер.

2.2.1. ЛАБОРАТОРНАЯ РАБОТА № 6 ПОСТРОЕНИЕ СИСТЕМ ИЗМЕРЕНИЯ ТЕМПЕРАТУРЫ

Цель лабораторной работы: научиться строить системы измерения температуры, изучить архитектуру, структуру и команды программирования индикатора WH1602, принципы его подключения к микроконтроллеру.

Задание для домашней подготовки

Изучить команды и режимы работы датчиков температуры фирмы Dallas и индикатора Winstar WH1602

Задания к лабораторной работе

1. Собрать измерительную систему в соответствии с рис. 12.
2. Написать и отладить программу измерения температуры в лаборатории. Информацию выводить на ЖКИ.
3. Ознакомиться с лабораторным комплексом измерения температуры с использованием других интегральных датчиков фирмы «Dallas» (DS1820 и др.)

Контрольные вопросы

1. Сколько доступных элементов структуры имеет контроллер ЖКИ?
2. Построить программную модель контроллера индикатора.
3. Описать режимы работы индикатора.
4. Какой набор символов используется в данном варианте индикатора?
5. Описать назначение внешних выводов панели индикатора.
6. Описать процедуру инициализации индикатора. В каком режиме он находится по включению питания?

2.2.2. ЛАБОРАТОРНАЯ РАБОТА № 7 ПОДКЛЮЧЕНИЕ ВНЕШНИХ УСТРОЙСТВ К МИКРОКОНТРОЛЛЕРУ

Цель лабораторной работы: изучить АЦП, таймер, архитектуру последовательного периферийного интерфейса микроконтроллера MC68HC711E9, назначение всех регистров, освоить программирование работы последовательного периферийного интерфейса во всех возможных режимах.

Задание для домашней подготовки

Пользуясь электронным учебником «MC68HC11E9-HCMOS Single-Chip Microcontroller», изучить работу АЦП, таймера, последовательный периферийный интерфейс SPI микроконтроллера MC68HC711E9

Задания к лабораторной работе

1. Сдать коллоквиум по теме «АЦП, таймер, последовательный периферийный интерфейс SPI микроконтроллера MC68HC711E9, режимы работы последовательного интерфейса связи (SPI)».
2. Пользуясь общей схемой стенда, с помощью перемычек задать режим работы контроллера с часами реального времени MC68HC68TIP по SPI.
3. На языке ассемблера контроллера написать программу связи микроконтроллера и часов реального времени MC68HC68TIP. Отображения текущего времени производить на экране ПЭВМ или ЖКИ.
4. Ознакомиться с лабораторным комплексом измерения температуры с использованием интегральных датчиков фирмы «Dallas» и «Analog Devices».
5. Ознакомиться с лабораторным комплексом измерения давления на базе интегральных датчиков фирмы «Motorola».
6. Оформить отчет по лабораторной работе, включающий: принципиальную электрическую схему стенда, теоретическую часть, тексты программ на языке ассемблера и Pascal.

Контрольные вопросы

1. Объяснить сопряжение МК с часами реального времени MC68HC68TIP.

2. Рассказать о АЦП микроконтроллера M68HC711E9.
3. Что такое таймерная система МК M68HC711E9?
4. Рассказать о средствах отладки для МК фирмы «Motorola».
5. Объясните назначение функциональных узлов SPI по структурной схеме.
6. Форматы кадров последовательной передачи информации, поддерживаемые SPI.
7. Функциональные характеристики SPI.

ЛИТЕРАТУРА

1. *Бродин В. Б., Шагурин И. И.* Микроконтроллеры. Архитектура, программирование, интерфейс. М.: Изд-во ЭКОМ, 1999;
2. *Шагурин И. И.* Микропроцессоры и микроконтроллеры фирмы «Motorola». Справочное пособие. М.: Радио и Связь, 1998;
3. M68HC11EVB Evaluation Board User's Manual, First Edition – Motorola, Inc., 1989
4. M68HC11EVM Evaluation Module User's Manual, Seventh Edition – Motorola, Inc., 1989
5. RAPID Integrated Development Environment User's Manual – Motorola, Inc., 1993
6. MC68HC711E9 Technical Data – Motorola, Inc., 1991

ПРИЛОЖЕНИЕ 1
ЦЕНТРАЛЬНОЕ ПРОЦЕССОРНОЕ УСТРОЙСТВО (ЦПУ), РЕЖИМЫ АДРЕСАЦИИ И СИСТЕМА КОМАНД

В этом приложении дано описание регистров ЦПУ, режимов адресации и краткое описание системы команд M6811.

Регистры ЦПУ

В дополнение к способности выполнять все инструкции процессоров M6800 и M6801, процессор MC68HC11 имеет 4-страничную карту кодов операций и может выполнять 91 новую инструкцию (см. пребайт). Семь регистров, доступных программисту и показанных на рис. 5, описываются в следующих параграфах.

Аккумуляторы A и B

Аккумулятор A и аккумулятор B являются 8-битными регистрами общего назначения и используются для хранения операторов и результатов арифметических вычислений или для обработки данных. Эти два регистра могут использоваться как один 16-битный регистр, который называется аккумулятор D.

Индексный регистр X (IX)

16-битный регистр IX используется в индексном режиме адресации. Он содержит 16-битный индекс, который складывается с 8-битным смещением, находящемся в теле команды, для получения эффективного адреса. Регистр IX может быть использован как счетчик или для хранения данных.

Индексный регистр Y(IY)

16-битный регистр IY также используется в индексном режиме адресации. Однако, в отличие от регистра IX, все команды, использующие регистр IY, требуют дополнительный байт в машинном коде и дополнительный цикл при выполнении.

Указатель стека (SP)

Указатель стека – это 16-битный регистр, содержащий адрес первой свободной ячейки в стеке. Стек – это последовательность регистров, запись/чтение в которые осуществляется по принципу «первый пришел – последний вышел». В стеке обычно сохраняют важные данные во время вызовов подпрограмм и прерываний. Каждый раз, когда новый байт записывается в стек, SP

декрементируется (команда PUSH). С другой стороны, когда байт извлекается из стека (команда PULL), SP инкременируется.

Счетчик команд (PC)

Счетчик команд – это 16-битный регистр, содержащий адрес команды, которая будет исполняться в следующем цикле.

Регистр состояния (CCR)

Регистр состояния – это 8-битный регистр, в котором каждый бит характеризует результат выполнения предыдущей команды. Эти биты могут быть индивидуально протестированы программой и в результате теста выполнены определенные действия. Каждый бит по отдельности описан ниже.

Перенос/заем (C)

Бит C устанавливается, если в результате выполнения арифметической операции происходит перенос в старший байт или заем из старшего байта. На этот бит также воздействуют операции сдвига и вращения.

Переполнение (V)

Бит V устанавливается, если в результате арифметической операции произошло переполнение. В противном случае бит V очищается.

Ноль (Z)

Бит Z устанавливается, если результатом арифметической или логической операции, а также операции обработки данных был ноль. В противном случае бит Z очищается.

Отрицательный результат (N)

Бит N устанавливается, если результат последней логической или арифметической инструкции, а также инструкции обработки данных был отрицательным. В противном случае бит N очищается. Результат считается отрицательным, если наиболее значащий бит равен единице.

Маска I-прерываний (I)

Бит I – это глобальная маска, которая запрещает обслуживание всех маскируемых источников прерываний. Если бит I установлен, то все маскируемые прерывания запрещены. Иначе – разрешены.

Перенос из младшего полубайта (H)

Бит H устанавливается, если в результате выполнения операций ADD, ABA или ADC в арифметико-логическом устройстве происходит перенос между 3 и 4 битами. В противном случае бит H очищается.

Маска X-прерывания (X)

Бит X используется для запрещения прерываний по входу XIRQ. Этот бит устанавливается только аппаратно (по сбросу или по запросу XIRQ), а очищается только программно (инструкции TAP или RTI).

Запрещение режима STOP (S)

Бит S устанавливается (очищается) программно для запрещения (разрешения) команды STOP. Если бит S установлен, то команда STOP действует как NOP.

Режимы адресации

Для доступа к памяти могут быть использованы шесть режимов адресации: непосредственный, прямой, расширенный, индексный (с использованием одного из 16-битных индексных регистров и 8-битного смещения), внутренний и относительный.

Некоторым командам требуется дополнительный байт перед кодом операции для приспособления к многостраничной карте кодов операций. Этот байт называется пребайтом.

В следующих параграфах описываются каждый режим адресации и пребайт. Под термином «эффективный адрес» понимается адрес в памяти, по которому хранится или загружается аргумент или с которого начинается выполнение программы.

Внутренняя (регистровая) адресация (Inherent Addressing Mode)

При внутренней адресации вся информация содержится в коде операции. Операнды (если они есть) являются регистрами, и обращения к памяти не требуется. Команда занимает один или два байта. Например CLRA, TSX, RTS.

Непосредственная адресация (Immediate Addressing Mode)

При непосредственной адресации аргумент содержится в байте (байтах), непосредственно следующем за командой. Число байтов определяется размером регистра, используемого в данной команде. Инструкции, использующие непосредственную адресацию, состоят из двух, трех или четырех байтов. Например LDAA #$FF, LDX #$1234.

Расширенная адресация (Extended Addressing Mode)

При расширенной адресации второй и третий байты (следующие за кодом операции) содержат абсолютный адрес операнда. Поэтому команда занимает три или четыре (если требуется пребайт) байта. Например LDAA $1010.

Прямая адресация (Direct Addressing Mode)

При прямой адресации используется только младший байт адреса операнды, а старший байт считается равным нулю. Младший байт адреса располагается после кода операции. Прямая адресация позволяет пользователю адресовать ячейки с $0000 по $00FF. При этом команда занимает два байта, а время исполнения команды уменьшается на один цикл. Программа может сконфигурировать память таким образом, чтобы эти ячейки располагались во внутреннем ОЗУ и/или внутренних регистрах, или во внешнем ОЗУ. Например LDX $17.

Индексная адресация (Indexed Addressing Mode)

В индексном режиме адресации один из индексных регистров (X или Y) и 8-битного беззнакового смещения, содержащегося в команде. Команда занимает два или три (если требуется пребайт) байта. Например LDAA OFFSET,X.

Относительная (базовая) адресация (Relative Addressing Mode)

Режим относительной адресации используется командами ветвления. Если условие перехода выполняется, то содержимое байта, следующего за кодом операции, (смещение) прибавляется к счетчику команд для вычисления эффективного адреса перехода. В противном случае управление передается следующей инструкции. Смещение считается знаковой величиной. В этом режиме адресации команда обычно занимает два байта. Например BEQ LOOP.

Пребайт

Для расширения числа команд, используемых MC68HC11, к некоторым командам добавляется пребайт. Эти расширенные команды обычно ассоциируются с индексным регистром Y. Команды, которые не требуют пребайта, могут рассматриваться как первая страница всей карты кодов операций. Остальные команды могут рассматриваться как 2, 3 и 4 страницы карты кодов операций и требуют пребайта: $18 для страницы 2, $1A для страницы 3 и $CD для страницы 4.

ПРИЛОЖЕНИЕ 2
КОМАНДЫ MC68HC11E9

ЦПУ в MC68HC11 в основном является расширением ЦПУ MC6801. В дополнение к способности выполнять все инструкции процессоров M6800 и M6801, ЦПУ MC68HC11 имеет страничную карту кодов операций с 91 новой инструкцией. Основные функциональные дополнения – это второй 16-битный индексный регистр, команды деления 16-разрядных данных двух типов, команды STOP и WAIT команды обработки битов.

Система команд представлена следующими группами:

1. Команды пересылки данных, связанные с аккумуляторами (LDAB (load accumulator. B), LDD, STAB, TAB (transfer A to B), CLRA (Clear A), PSHA (push A to stack), PULA, ...)
2. Команды пересылки для стека и индексных регистров (PSHX, TSX (transfer SP to X), ...)
3. Команды переходов (JMP, JSR (jump to subroutine), RTS (return from subroutine), переходы по условиям и состояниям битов,…)
4. Арифметические команды (ADD, SUB, INC, DEC, MUL, DIV, CMP, ...)
5. Логические команды (AND, OR, EOR, COM, ...)
6. Команды работы с битами (установка, сброс, проверка, сравнение, сдвиги, …)
7. Специальные команды (STOP, WAI (wait for interrupt), SWI (software interrupt), ...)

В таблице приводятся все инструкции MC68HC11 во всех возможных режимах адресации. Для каждой команды приводятся машинный код, возможные операнды, число байтов команды, время исполнения в E-циклах. В конце таблицы находятся примечания, где поясняются условные обозначения. Примеры использования режимов адресации приведены ниже.

Таблица

Мнемоника	Действие	Логическое выражение	Режим адресации операнда	Машинный код	
				Команда	операнд
1	2	3	4	5	6
ABA	Add Accumulators	A+B → A	INH	1B	
ABX	Add B to X	IX+00:B →IX	INH	3A	
ABY	Add B to Y	IY+00:B → IY	INH	18 3A	
ADCA (opr)	Add with Carry to A	A+M+C → A	A IMM	89	ii
			A DIR	99	dd
			A EXT	B9	hh ||
			A IND, X	A9	ff
			A IND, Y	18 A9	ff
ADCB(opr)	Add with Carry to B	B+M+C → B	B IMM	C9	ii
			B DIR	D9	dd
			B EXT	F9	hh ||
			B IND, X	E9	ff
			B IND, Y	18 E9	ff
ADDA(opr)	Add Memory to A	A+M → A	A IMM	8B	ii
			A DIR	9B	dd
			A EXT	BB	hh ||
			A IND, X	AB	ff
			A IND, Y	18AB	ff
ADDB(opr)	Add Memory to B	B+M → B	B IMM	CB	ii
			B DIR	DB	dd
			B EXT	FB	hh ||
			B IND, X	EB	ff
			B IND, Y	18 EB	ff
ADDD(opr)	Add 16-Bit to D	D+M:M+1 → D	IMM	C3	jj kk
			DIR	D3	dd
			EXT	F3	hh ||
			IND, X	E3	ff
			IND, Y	18 E3	ff
ANDA(opr)	AND A with Memory	A*M → A	A IMM	84	ii
			A DIR	94	dd
			A EXT	B4	hh ||
			A IND, X	A4	ff
			A IND, Y	18 A4	ff
ANDB(opr)	AND B with Memory	B*M → B	B IMM	C4	ii
			B DIR	D4	dd
			B EXT	F4	hh ||
			B IND, X	E4	ff
			B IND, Y	18 E4	ff
ASL(opr)	Arithmetic Shift Left	← □←□□□□□□□←0 C b7 b0	EXT	78	hh ||

Продолжение

1	2	3	4	5	6
			IND, X	68	ff
			IND, Y	18 68	ff
ASLA			A INH	48	
ASLB			B INH	58	
ASLD	Arithmetic Shift Left Double	□←□□□□…□□□□ ←0 C b15 b0	INH	05	
ASR(opr)	Arithmetic Shift Right	→ →□□□□□□□□→□ b7 b0 C	EXT	77	hh ll
			IND, X	67	ff
			IND, Y	18 67	ff
ASRA			A INH	47	
ASRB			B INH	57	
BCC(rel)	Branch if Carry Clear	? C=0	REL	24	rr
BCLR(opr) (msk)	Clear Bit(s)	M*(mm) → M	DIR	15	dd mm
			IND, X	1D	ff mm
			IND, Y	18 1D	ff mm
BCS(rel)	Branch if Carry Set	? C=1	REL	25	rr
BEQ(rel)	Branch if = Zero	? Z=1	REL	27	rr
BGE(rel)	Branch if ≥ Zero	? N ⊕ V=0	REL	2C	rr
BGT(rel)	Branch if > Zero	? Z= (N ⊕ V)=0	REL	2E	rr
BHI(rel)	Branch if Higher	? C+Z=0	REL	22	rr
BHS(rel)	Branch if Higher or Same	? C=0	REL	24	rr
BITA(opr)	Bit(s) Test A with Memory	A•M	A IMM	85	ii
			A DIR	95	dd
			A EXT	B5	hh ll
			A IND, X	A5	ff
			A IND, Y	18 A5	ff
BITB(opr)	Bit(s) Test B with Memory	B•M	B IMM	C5	ii
			B DIR	D5	dd
			B EXT	F5	hh ll
			B IND, X	E5	ff
			B IND, Y	18 E5	ff
BLE(rel)	Branch if ≤Zero	? Z+(N ⊕ V)=1	REL	2F	rr
BLO(rel)	Branch if Lower	? C=1	REL	25	rr
BLS(rel)	Branch if Lower of Same	? C+Z=1	REL	23	rr
BLT(rel)	Branch if < Zero	? N ⊕ V=1	REL	2D	rr
BMI(rel)	Branch if Minus	? N=1	REL	2B	rr
BNE(rel)	Branch if Not=Zero	? Z=0	REL	26	rr
BPL(rel)	Branch if Plus	? N=0	REL	2A	rr
BRA(rel)	Branch if Always	? 1=1	REL	20	rr
BRCLR(opr)	Branch if Bit(s) Clear	? M•mm=0	DIR	13	dd mm rr

Продолжение

1	2	3	4	5	6
(msk)			IND, X	1F	ff mm rr
(rel)			IND, Y	18 1F	ff mm rr
BRN(rel)	Branch Never	? 1=0	REL	21	rr
BRSET(opr)	Branch if Bit(s) Set	? (M)•mm=0	DIR	12	dd mm rr
(msk)			IND, X	1E	ff mm rr
(rel)			IND, Y	18 1E	ff mm rr
BSET(opr)	Set Bit(s)	M+mm → M	DIR	14	dd mm
(msk)			IND, X	1C	ff mm
			IND, Y	18 1C	ff mm
BSR(rel)	Branch to Subroutine	See Special Ops	REL	8D	rr
BVC(rel)	Branch if Owerflow Clear	? V=0	REL	28	rr
BVS(rel)	Branch if Owerflow Set	? V=1	REL	29	rr
CBA	Compare A to B	A-B	INH	11	
CLC	Clear Carry Bit	0→C	INH	0C	
CLI	Clear Interrupt Mask	0→1	INH	0E	
CLR(opr)	Clear Memory Byte	0→M	EXT	7F	hh ||
			IND, X	6F	ff
			IND, Y	18 6F	ff
CLRA	Clear Accumulator A	0→A	A INH	4F	
CLRB	Clear Accumulator B	0→B	B INH	5F	
CLV	Clear Overflow Flag	0→V	INH	0A	
CMPA(opr)	Compare A to Memory	A-M	A IMM	81	ii
			A DIR	91	dd
			A EXT	B1	hh ||
			A IND, X	A1	ff
			A IND, Y	18 A1	ff
CMPB(opr)	Compare B to Memory	B-M	B IMM	C1	ii
			B DIR	D1	dd
			B EXT	F1	hh ||
			B IND, X	E1	ff
			B IND, Y	18 E1	ff
COM(opr)	1’s Complement Memory Byte	$FF-M→M	EXT	73	hh ||
			IND, X	63	ff
			IND, Y	18 63	ff
COMA	1’s Complement A	$FF-A→A	A INH	43	
COMB	1’s Complement B	$FF-B→B	B INH	53	
CPD(opr)	Compare D to Memory 16-Bit	D-M:M+1	IMM	1A 83	jj kk
			DIR	1A 93	dd
			EXT	1A B3	hh ||
			IND, X	1A A3	ff
			IND, Y	CD A3	ff
CPX(opr)	Compare X to Memory 16-Bit	IX-M:M+1	IMM	8C	jj kk
			DIR	9C	dd
			EXT	BC	hh ||
			IND, X	AC	ff

Продолжение

1	2	3	4	5	6
			IND, Y	CD AC	ff
CPY(opr)	Compare Y to Memory 16-Bit	IY-M:M+1	IMM	18 8C	jj kk
			DIR	18 9C	dd
			EXT	18 BC	hh ll
			IND, X	1A AC	ff
			IND, Y	18 AC	ff
DAA	Decimal Adjust A	Adjust Sum to BCD	INH	19	
DEC(opr)	Decrement Memory Byte	M-1→M	EXT	7A	hh ll
			IND, X	6A	ff
			IND, Y	18 6A	ff
DECA	Decrement Accumulator A	A-1→A	A INH	4A	
DECB	Decrement Accumulator B	B-1→B	B INH	5A	
DES	Decrement Stack Pointer	SP-1→SP	INH	34	
DEX	Decrement Index Register X	IX-1→IX	INH	09	
DEY	Decrement Index Register Y	IY-1→IY	INH	18 09	
EORA(opr)	Exclusive OR A with Memory	A ⊕ M→A	A IMM	88	ii
			A DIR	98	dd
			A EXT	B8	hh ll
			A IND, X	A8	ff
			A IND, Y	18 A8	ff
EORB(opr)	Exclusive OR B with Memory	B ⊕ M→B	B IMM	C8	ii
			B DIR	D8	dd
			B EXT	F8	hh ll
			B IND, X	E8	ff
			B IND, Y	18 E8	ff
FDIV	Fractional Divide 16 by 16	D/IX→IX; r→D	INH	03	
IDIV	Integer Divide 16 by 16	D/IX→IX; r→D	INH	02	
INC(opr)	Increment Memory Byte	M+1→M	EXT	7C	hh ll
			IND, X	6C	ff
			IND, Y	18 6C	ff
INCA	Increment Accumulator A	A+1→A	A INH	4C	
INCB	Increment Accumulator B	B+1→B	B INH	5C	
INS	Increment Stack Pointer	SP+1→SP	INH	31	
INX	Increment Index Register X	IX+1→IX	INH	08	
INY	Increment Index Register Y	IY+1→IY	INH	18 08	
JMP(opr)	Jump	See Special Ops	EXT	7E	hh ll
			IND, X	6E	ff
			IND, Y	18 6E	ff
JSR(opr)	Jump to Subroutine	See Special Ops	DIR	9D	dd
			EXT	BD	hh ll
			IND, X	AD	ff
			IND, Y	18 AD	ff
LDAA(opr)	Load Accumulator A	M→A	A IMM	86	ii
			A DIR	96	dd
			A EXT	B6	hh ll
			A IND, X	A6	ff

Продолжение

1	2	3	4	5	6
			A IND, Y	18 A6	ff
LDAB(opr)	Load Accumulator B	M→B	B IMM	C6	ii
			B DIR	D6	dd
			B EXT	F6	hh ll
			B IND, X	E6	ff
			B IND, Y	18 E6	ff
LDD(opr)	Load Double Accumulator D	M→A, M+1→B	IMM	CC	jj kk
			DIR	DC	dd
			EXT	FC	hh ll
			IND, X	EC	ff
			IND, Y	18 EC	ff
LDS(opr)	Load Stack Pointer	M:M+1→SP	IMM	8E	jj kk
			DIR	9E	dd
			EXT	BE	hh ll
			IND, X	AE	ff
			IND, Y	18 AE	ff
LDX(opr)	Load Index Register X	M:M+1→IX	IMM	CE	jj kk
			DIR	DE	dd
			EXT	FE	hh ll
			IND, X	EE	ff
			IND, Y	CD EE	ff
LDY(opr)	Load Index Register Y	M:M+1→IY	IMM	18 CE	jj kk
			DIR	18 DE	dd
			EXT	18 FE	hh ll
			IND, X	1A EE	ff
			IND, Y	118 EE	ff
LSL(opr)	Logical Shift Left	←	EXT	78	hh ll
		□←-□□□□□□□□←-0			
		C b7 b0			
			IND, X	68	ff
			IND, Y	18 68	ff
LSLA			A INH	48	
LSLB			B INH	58	
LSLD	Logical Shift Left Double	□←□□□□--□□□□←0	INH	05	
		C b15 b0			
LSR(opr)	Logical Shift Right	→	EXT	74	hh ll
		→□□□□□□□□→□			
		b7 b0 C			
			IND, X	64	ff
			IND, Y	18 64	ff
LSRA			A INH	44	
LSRB			B INH	54	
LSRD	Logical Shift Right Double	□→□□□□….□□□□→0	INH	04	
		C b15 b0			
MUL	Multiply 8 by 8	AxB→D	INH	3D	

Продолжение

1	2	3	4	5	6
NEG(opr)	2's Complement Memory Byte	0-M→M	EXT	70	hh ||
			IND, X	60	ff
			IND, Y	18 60	ff
NEGA	2's Complement A	0-A→A	A INH	40	
NEGB	2's Complement B	0-B→B	B INH	50	
NOP	No Operation	No Operation	INH	01	
ORAA(opr)	OR Accumulator A (Inclusive)	A+M→A	A IMM	8A	ii
			A DIR	9A	dd
			A EXT	BA	hh ||
			A IND, X	AA	ff
			A IND, Y	18 AA	ff
ORAB(opr)	OR Accumulator B (Inclusive)	B+M→B	B IMM	CA	ii
			B DIR	DA	dd
			B EXT	FA	hh ||
			B IND, X	EA	ff
			B IND, Y	18 EA	ff
PSHA	Push A onto Stack	A→Stk, SP=SP-1	A INH	36	
PSHB	Push B onto Stack	B→Stk, SP=SP-1	B INH	37	
PSHX	Push X onto Stack (Lo First)	IX→Stk, SP=SP-2	INH	3C	
PSHY	Push Y onto Stack (Lo First)	IY→Stk, SP=SP-2	INH	18 3C	
PULA	Pull B from Stack	SP=SP+1, A←Stk	A INH	32	
PULB	Pull B from Stack	SP=SP+1, B←Stk	B INH	33	
PULX	Pull X from Stack (Hi First)	SP=SP+2, IX←Stk	INH	38	
PULY	Pull Y from Stack (Hi First)	SP=SP+2, IY←Stk	INH	18 38	
ROL(opr)	Rotate Left	□←-□□□□□□□□←-□ C b7 ← b0 C	EXT	79	hh ||
			IND, X	69	ff
			IND, Y	18 69	ff
ROLA			A INH	49	
ROLB			B INH	59	
ROR(opr)	Rotate Right	□→□□□□□□□□→□ C b7 → b0 C	EXT	76	hh ||
			IND, X	66	ff
			IND, Y	18 66	ff
RORA			A INH	46	
RORB			B INH	56	
RTI	Return from Interrupt	See Special Ops	INH	3B	
RTS	Return from Subroutine	See Special Ops	INH	39	
SBA	Subtract B from A	A-B→A	INH	10	
SBCA(opr)	Subtract with Carry from A	A-M-C→A	A IMM	82	ii
			A DIR	92	dd
			A EXT	B2	hh ||
			A IND, X	A2	ff
			A IND, Y	18 A2	ff

Продолжение

1	2	3	4	5	6
SBCB(opr)	Subtract with Carry from B	B-M-C→B	B IMM	C2	ii
			B DIR	D2	dd
			B EXT	F2	hh ‖
			B IND, X	E2	ff
			B IND, Y	18 E2	ff
SEC	Set Carry	1→C	INH	OD	
SEI	Set Inteerupt Mask	1→1	INH	OF	
SEV	Set Owerflow Flag	1→V	INH	OB	
STAA(opr)	Store Accumulator A	A→M	A DIR	97	dd
			A EXT	B7	hh ‖
			A IND, X	A7	ff
			A IND, Y	18 A7	ff
STAB(opr)	Store Accumulator B	B→M	B DIR	D7	dd
			B EXT	F7	hh ‖
			B IND, X	E7	ff
			B IND, Y	18 E7	ff
STD(opr)	Store Accumulator D	A→M, B→M+1	DIR	DD	dd
			EXT	FD	hh ‖
			IND, X	ED	ff
			IND, Y	18 ED	ff
STOP	Stop Internal Clocks		INH	CF	
STS(opr)	Store Stack Pointer	SP→M:M+1	DIR	9F	dd
			EXT	BF	hh ‖
			IND, X	AF	ff
			IND, Y	18 AF	ff
STX(opr)	Store Index Register X	IX→M:M+1	DIR	DF	dd
			EXT	FF	hh ‖
			IND, X	EF	ff
			IND, Y	CD EF	ff
STY(opr)	Store Index Register Y	IY→M:M+1	DIR	18 DF	dd
			EXT	18 FF	hh ‖
			IND, X	1A EF	ff
			IND, Y	18 EF	ff
SUBA(opr)	Subtract Memory from A	A-M→A	A IMM	80	ii
			A DIR	90	dd
			A EXT	B0	hh ‖
			A IND, X	A0	ff
			A IND, Y	18 A0	ff
SUBB(opr)	Subtract Memory from B	B-M→B	B IMM	C0	ii
			B DIR	D0	dd
			B EXT	F0	hh ‖
			B IND, X	E0	ff
			B IND, Y	18 E0	ff
SUBD(opr)	Subtract Memory from D	D-M:M+1→D	IMM	83	jj kk
			DIR	93	dd
			EXT	B3	hh ‖
			IND, X	A3	ff

Окончание

1	2	3	4	5	6
			IND, Y	18 A3	ff
SWI	Software Interrupt	See Special Ops	INH	3F	
TAB	Transfer A to B	A→B	INH	16	
TAP	Transfer A to CC Register	A→CCR	INH	06	
TBA	Transfer B to A	B→A	INH	17	
TEST	TEST (Only in Test Modes)	Address Bus Counts	INH	00	
TPA	Transfer CC Register to A	CCR→A	INH	07	
TST(opr)	Test for Zero or Minus	M-0	EXT	7D	hh ||
			IND, X	6D	ff
			IND, Y	18 6D	ff
TSTA		A-0	A INH	4D	
TSTB		B-0	B INH	5D	
TSX	Transfer Stack Pointer to X	SP+1→IX	INH	30	
TSY	Transfer Stack Pointer to Y	SP+1→IY	INH	18 30	
TXS	Transfer X to Stack Pointer	IX-1→SP	INH	35	
TYS	Transfer Y to Stack Pointer	IY-1→SP	INH	18 35	
WAI	Wait for Interrupt	Stack Regs & WAIT	INH	3E	
XGDX	Exchange D with X	IX→D, D→IX	INH	8F	
XGDY	Exchange D with Y	IY→D, D→IY	INH	18 8F	

Примечания:

dd – 8-битовый прямой адрес ($0000-$00FF) (старшая половина адреса принимается равной $00);

ff – 8-битовое положительное смещение от $00 до $FF (прибавляется к индексу);

hh – старший байт 16-разрядного адреса;

ii – один байт данных;

jj – старший байт 16-разрядных данных;

kk – младший байт 16-разрядных данных;

ll – младший байт 16-разрядного адреса;

mm – 8-битовая маска;

rr – смещение со знаком (от $80(-128) до $ 7F (+127)).

ПРИЛОЖЕНИЕ 3
КОМАНДЫ ОТЛАДЧИКА PCBUG11

ASM адрес[мнемоника|директива]	Ассемблирование вводимых мнемоник и запись их в память
BAUD [скорость]	Показать или установить скорость обмена информацией по COM-порту
BF адрес1 [адрес2] байт|слово	Заполнение блока памяти заданным байтом или словом
BL	Показать точки останова
BR [адрес[имя макроса]]	Показать или поставить точку останова (с выполнением заданного макроса)
CALL адрес	Выполнение подпрограммы по заданному адресу
CLRM	Очистка всех макросов
CLS	Очистка основного окна
CONTROL [параметр]	Показать или изменить параметры отладчика
DASM адрес1 [адрес2]	Дизассемблирование с адреса 1 (по адрес 2)
DB нач. адрес [кон. адрес]	Показать память микроконтроллера
DEBUG	Зарезервированное слово (нельзя использовать как метку)
DEFINE символ значение|адрес	Присвоить символу значение
DEFM макрос|TRACE|AUTOSTART	Создать командный, трассировочный или запускаемый автоматически макрос
DELM макрос|TRACE|AUTOSTART	Удалить командный, трассировочный или запускаемый автоматически макрос
DIR [маска]	Показать каталог диска
DOS [команда]	Временный выход в DOS или выполнение команды DOS
EDITM имя макроса	Редактирование макроса
EEPROM [нач. адрес [кон. адрес]]	Показать, стереть или установить границы адресов EEPROM
EEPROM DELAY значение	Установить время записи или стирания для EEPROM
EEPROM ERASE [опция] [адрес]	Показать или изменить функцию стирания EEPROM перед записью
EPROM [нач. адрес[кон. адрес]]	Показать, стереть или установить границы адресов EPROM
EPROM DELAY значение	Установить время записи или стирания для EEPROM
FIND байт|слово адрес1 адрес2	Найти все появления байта|слова между адресами 1 и 2

Продолжение

FIND мнемоника адрес1 адрес2	Найти все появления мнемоники между адресами 1 и 2
G[адрес]	Запуск исполнения пользовательской программы
HELP [команда]	Вывести справочную информацию
KLE	Стереть последнее сообщение об ошибке
LOADM[имя файла[имя макроса]]	Загрузить макросы
LOADS имя файла[адрес загрузки]	Загрузить файл S-формата в память микроконтроллера
LS символ	Показать символ
LSTM [макрос|TRACE|AUTOSTART]	Показать имена или содержимое макросов
MD нач. адрес [кон. адрес]	Показать память микроконтроллера
MM адрес	Редактирование памяти микроконтроллера
MOVE адрес1 адрес2 адрес3	Перемещение блока данных в памяти микроконтроллера с адреса 1 по 2 на адрес 3
MS адрес байт|слово [байт|слово]...	Запись слов или байт в память, начиная с заданного адреса
MSG [строка]	Вывести сообщение в главное окно
NOBR[адрес]	Удаление всех или указанных точек останова
PAUSE [мс]	Ожидание нажатия клавиши или времени задержки
PRINT	Показать номер версии отладчика
PROTECT[нач. адрес[кон. адрес]]	Показать, стереть или установить адреса памяти, защищенные от записи
QUIT [Y]	Выход из отладчика(Y – без ожидания подтверждения)
RD [T]	Показать регистры (T – в режиме отладки)
RESET [адрес]	Аппаратный сброс микроконтроллера по существующему или новому вектору сброса
RESTART [опция]	Перезапуск отладчика
RM	Редактирование регистров микроконтроллера
RS регистр значение	Установить новое значение для регистра
S	Остановить выполнение программы
SAVEM [имя файла]	Записать макросы в файл
SHELL [команда]	Временный выход в DOS или выполнение команды DOS
T [адрес]	Трассировка программы
TERM [X1 Y1 X2 Y2]	Эмулятор терминала в заданном окне экрана
TYPE имя файла	Вывести содержимое файла в главном окне
UNDEF символ	Стереть определение символа
VER	Показать номер версии отладчика
VERF имя файла[адрес памяти]	Проверить правильность записи S-файла в память

Окончание

VERF ERASE адрес1 [адрес2]	Проверка памяти на содержание $FF
VERF SET адрес1 адрес2 значение	Проверка памяти на содержание заданного значения
WAIT [мс]	Ожидание в течение заданного времени
^B {Клавиша CTRL+буква}	Послать BREAK по COM-порту
^P	Переключить линию RTS
^R	Попытка заново синхронизироваться с микроконтроллером

ПРИЛОЖЕНИЕ 4
ТЕКСТ ПРОГРАММЫ UPLOAD

```
{$m 2048, 0, 0}
uses crt, dos;

const BR1=7680;{ Скорость обмена информацией }
BR2=9600;{ по последовательному порту }
COM=0; { Номер COM-порта            }

var P : integer; { массив адресов COM-портов}
Ticks : LongInt Absolute $40:$6C; { таймер 1/18.9 сек.}
filename:string;
{ запись байта в COM-порт }
Procedure WriteCom(a:byte);
begin
Repeat until (Port[P+5] and $20=$20) or KeyPressed;
if keypressed then
if ReadKey=#27 then Halt(1);
Port[P]:=a;
end;

{ считывание байта с COM-порта }
Procedure ReadCom(var a:byte);
var ST : longint;
begin
ST:=Ticks;                                    { 5 секунд ожидания }
Repeat Until (Port[P+5] and 1=1)or(Ticks-ST>96) or KeyPressed; { при
отсутствии данных }
if keypressed then
if ReadKey=#27 then Halt(1);
If Port[P+5] and 1=1 then a:=port[P];
end;

{ Инициализация }
procedure Init;
var BT : integer;
bb : byte;
f : file of byte;
begin
assign(f, filename);
reset(f);
BT:=Round(115200.0/BR1);
Port[P + 3] := $83;
Port[P + 0] := Lo(BT);   { Установка делителя частоты }
```

```
Port[P + 1] := Hi(BT);    { для генератора скорости обмена }
Port[P + 3] := $03;       { 8 бит данных, 1 стоповый, без четности}
Port[P + 1] := $00;
WriteLn('Press RESET button on the controller board...');
bb:=1;
While bb<>0 do ReadCom(bb);
While not Eof(f) do
begin
Read(f, bb);
WriteCom(bb);
end;
close(f);
end;

{----------------------Начало основной программы--------------------------}
var key : char;
b1, b2, b3 : byte;
t, u : real;
s:string;
f:file;
begin
Writeln('HC11 program uploader');
if paramcount=0 then
Begin
Writeln('Usage: UPLOAD filename. s19 [-n]');
Writeln(' Where n - COM port number (COM1 is default)');
halt;
end;
if paramcount=2 then s:=paramstr(2) else s:='-1';
if (length(s)=2) and (s[1]='-') then
b1:=byte(s[2])-48 else b1:=1;
p:=memw[$40:2*(b1-1)];
Writeln('Using COM', b1);
filename:=paramstr(1);
exec('upload. scv', filename+' >nul');
delete(filename, length(filename)-2, 3);
filename:=filename+'bin';
Init;
Writeln('Uploading complete.');
assign(f, filename);
erase(f);
end.
```

ПРИЛОЖ5ЕНИЕ 5
ОБЪЕКТНЫЕ ФАЙЛЫ В ФОРМАТЕ MOTOROLA «S19-RECORD»

Компания «Motorola» не выполняет соглашение, соблюдаемое другими производителями микроконтроллеров, по использованию шестнадцатиричного формата файлов «INHX8M» фирмы «Intel». Вместо этого для программирования микроконтроллеров Motorola используются объектные файлы в формате S19-record. Этот формат очень похож на формат «Intel», но имеет некоторые отличия, о которых сказано ниже. Свое название формат S19-record получил потому, что каждая строка объектного файла начинается с символа S# либо S1 для строк данных, либо S9 для обозначения конца файла.

Пример файла с расширением .S19 представлен ниже:

```
S11303005FA680B700A6FEB7041A083ACC030C16A1
S110031008B600A880B70080120A1E008005
S10407F106FD
S10B07F8030F031803000300C2
S9030000FC
```

Вся информация, необходимая для загрузки программы в память микроконтроллера, находится в файле. Следует иметь в виду, что в файле не содержится символьная информация, которая обычно генерируется компилятором или ассемблером специально для некоторых отладчиков и эмуляторов.

Каждую строку можно разбить на несколько полей, которые содержат служебную информацию для программатора или байты данных. Каждое поле начинается с определенного столбца, как показано в следующей таблице:

Столбец	Название	Комментарий
1 – 2	Начало строки	Всегда начинается с «S#», где «#» равно 1 или 9
3 – 4	Число байтов в строке	Определяет число байтов в строке
5 – 8	Начальный адрес данных строки	Определяет начальный адрес для пересылки данных
9 – ...	Байты данных	Каждый байт представлен двумя символами в шестнадцатиричном коде
End-2	Контрольная сумма	Сумма элементов строки

Обычно каждая строка содержит до 16 байтов данных файла. Возьмем первую строку из предыдущего примера и разобьем ее на поля, чтобы показать, что реально в ней находится. Первые два байта (S1) показывают, что в строке присутствуют данные. Цифры в следующих двух позициях определяют количество байтов в строке. В отличие от формата Intel, это число не умножается на два и реально отражает полное число байтов, расположенных в интервале от данной позиции до конца строки, включая байты адреса данных и контрольной суммы. Для первой строки это число 13 в шестнадцатиричном или 19 в десятичном коде. Байты с 5-го по 8-й определяют адрес, начиная с которого будут загружаться данные в память микроконтроллера. В данном случае это число $0300, а это значит, что байты кода будут размещены в памяти, начиная с адреса $0300. За адресом следуют байты команд в шестнадцатиричном коде. Как принято для микропроцессоров Motorola, числа представлены в удобном для чтения виде, где первый байт содержит код команды, а следующие байты содержат параметры инструкции в формате: старший байт – младший байт. Таким образом, можно легко проверить соответствие данных в файле с расширением .S19 листингу ассемблерной программы.

Последние два байта являются контрольной суммой всех байтов, начиная с третьей позиции (то есть числа 1 или 9 не входят в контрольную сумму). Над суммой всех байтов строки производится логическая операция И со значением $0FF, что будет равно $0FF.

Последняя строка в данном примере начинается с «S9», что означает конец.

Учебное издание

Микроконтроллеры в системах управления и измерения

Методические указания
к лабораторному практикуму

Для студентов специальностей
G 31 04 02 «Радиофизика»,
G 31 04 03 «Физическая электроника»,
E 25 01 10 «Коммерческая деятельность на рынке
радиоэлектронных средств и информационных услуг»

А в т о р - с о с т а в и т е л ь
Шалатонин Иван Алексеевич

В авторской редакции
Технический редактор *Г.М. Романчук*
Корректор *С. М. Борисова*

Ответственный за выпуск *И. А. Шалатонин*

Подписано в печать 2002. Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Усл. печ. л. Уч.- изд. л. Тираж экз. Зак.

Белорусский государственный университет.
Лицензия ЛВ №315 от 14.07.98.
220050, Минск, проспект Франциска Скорины, 4.

Отпечатано на копировально-множительной технике
факультета радиофизики и электроники БГУ
220064, Минск, ул. Курчатова, 5.