Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра электронных вычислительных средств

ПРОЕКТИРОВАНИЕ ЭВС НА ОДНОКРИСТАЛЬНЫХ МИКРОКОНТРОЛЛЕРАХ

МЕТОДИЧЕСКОЕ ПОСОБИЕ

по курсам

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА,
ПРОЕКТИРОВАНИЕ КОМПЬЮТЕРНЫХ СИСТЕМ

для студентов специальности

«Проектирование и технология электронных вычислительных средств»

В 3-х частях

Часть 2

8-РАЗРЯДНЫЕ МИКРОКОНТРОЛЛЕРЫ СЕМЕЙСТВА M68HC11

Минск 2002

ББК 32.97
УДК 681.3
Л 12

Авторы: А.А.Петровский, М.В.Качинский, В.Б.Клюс, А.Б.Давыдов

Проектирование ЭВС на однокристальных микроконтроллерах: Метод. пособие по курсам "Микропроцессорная техника", "Проектирование компьютерных систем" для студентов спец. "Проектирование и технология электронных вычислительных средств". В 3 ч. Ч. 2: 8-разрядные микроконтроллеры семейства М68НС11 / А.А.Петровский, М.В.Качинский, В.Б.Клюс, А.Б.Давыдов. – Мн.: БГУИР, 2002. – 44 с.

ISBN 985-444-154-7.

Данное пособие продолжает серию материалов по микроконтроллерам американской фирмы Motorola – признанного мирового лидера по объему производства и продажи микроконтроллеров.

Во второй части пособия описывается базовая модель семейства МС68НС11А8, которая имеет типовой состав периферийных устройств.

ISBN 985-444-154-7

В В Е Д Е Н И Е

Данное методическое пособие продолжает серию материалов по микроконтроллерам (МК) фирмы Motorola, которая предлагает сегодня самую широкую в мире номенклатуру микроконтроллеров, охватывающую практически все области применения и включающую в себя около 300 моделей, от простейших дешевых до высокопроизводительных 32-разрядных микроконтроллеров с RISC-ядром и мощной периферией.

Семейство 8-разрядных микроконтроллеров M68HC11 является одним из наиболее распространенных в мире семейств микроконтроллеров. Это семейство включает несколько десятков моделей (табл. 1), которые имеют одинаковое процессорное ядро, но отличаются объемом и типом используемой памяти, набором периферийных устройств и рядом других характеристик (тактовая частота, температурный диапазон, тип корпуса) [1, 2].

Таблица 1

Основные характеристики МК семейства M68HC11

Модель	РПЗУ УФ	ПЗУ	РПЗУ	ОЗУ	Вх/вых	АЦП	ШИМ
MC68HC11A8	–	8К	512	256	38	8 каналов, 8 разрядов	–
MC68HC11A1	–	–	512	256	22	8 каналов, 8 разрядов	–
MC68HC11A0	–	–	–	256	22	8 каналов, 8 разрядов	–
MC68HC11E9	–	12К	512	512	38	8 каналов, 8 разрядов	–
MC68HC11E1	–	–	512	512	22	8 каналов, 8 разрядов	–
MC68HC811E2	–	–	2К	256	38	8 каналов, 8 разрядов	–
PC68HC711E9	12К	–	512	512	38	8 каналов, 8 разрядов	–
MC68HC11F1	–	–	512	1К	54	8 каналов, 8 разрядов	–
MC68HC11K4	–	24К	640	768	62	8 каналов, 8 разрядов	4 канала, 8 разрядов
MC68HC711K4	24К	–	640	768	62	8 каналов, 8 разрядов	4 канала, 8 разрядов
MC68HC11G5	–	16К	–	512	66	8 каналов, 10 разрядов	4 канала, 8 разрядов

В данном пособии описывается базовая модель семейства MC68HC11A8, которая имеет типовой состав периферийных устройств [3].

1. СОСТАВ И РЕЖИМЫ РАБОТЫ МИКРОКОНТРОЛЛЕРА MC68HC11A8

Микроконтроллер MC68HC11A8 содержит:

- 8-разрядный процессор;
- внутреннюю память общим объемом 9152 байт, которая включает ПЗУ емкостью 8 Кбайт, электрически репрограммируемое ПЗУ (РПЗУ) емкостью 512 байт, служебное ПЗУ емкостью 192 байта и ОЗУ емкостью 256 байт;
- блок программирования (БПР);
- пять параллельных 8-разрядных портов A, B, C, D, E;
- синхронный (СПП) и асинхронный (АПП) последовательные порты;
- 16-разрядный таймер;
- блок счетчика импульсов (БСИ);
- блок контроля функционирования (БКФ);
- генератор тактовых импульсов (ГТИ);
- 8-разрядный аналого-цифровой преобразователь (АЦП).

Процессор содержит два 8-разрядных аккумулятора A и B, которые при выполнении ряда команд используются как 16-разрядный регистр D, два 16-разрядных индексных регистра X и Y, регистр флагов CCR, 16-разрядные указатель стека SP и программный счетчик PC (рис. 1). В состав процессора входят также служебные регистры CONFIG, OPTION, HPRIO, INIT, TEST1, определяющие конфигурацию и режим работы микроконтроллера.

Аккумуляторы (A, B и D). Аккумуляторы A и B представляют собой 8-разрядные аккумуляторы общего назначения, используемые для хранения

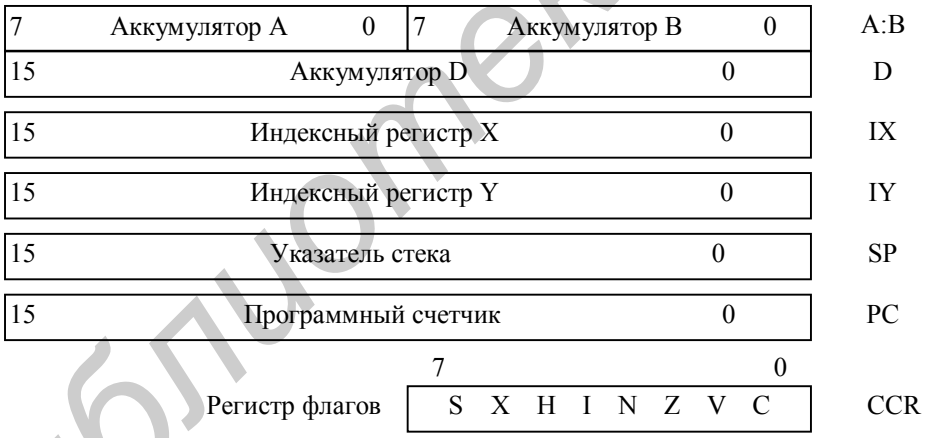


Рис. 1. Программистская модель МК MC68HC11A8

операндов и результатов операций. Некоторые команды используют эти два 8-разрядных аккумулятора как 16-разрядный аккумулятор двойной длины (аккумулятор D). Большинство операций могут использовать как аккумулятор A, так и аккумулятор B. Однако существует несколько исключений. Команды ABX и ABY добавляют содержимое 8-разрядного аккумулятора B к содержимому 16-разрядного индексного регистра X или Y, они не являются эквивалентными командами, использующими регистры A или B. Команды TAP и TPA используются для пересылки данных из аккумулятора A в регистр флагов и наоборот. Команда десятичной коррекции аккумулятора A (DAA)

используется после арифметических операций в двоично-десятичном коде (BCD). Однако эти команды неприменимы к аккумулятору B. Наконец, команды сложения, вычитания и сравнения, использующие оба аккумулятора A и B (ABA, SBA и CBA), работают только в одном направлении. Поэтому важно заранее предусмотреть правильное расположение данных в соответствующем аккумуляторе.

Индексные регистры (X и Y). 16-разрядные индексные регистры X и Y используются в индексном способе адресации. В индексном способе адресации содержимое 16-разрядного индексного регистра добавляется к 8-разрядному смещению, заданному в команде, для формирования эффективного адреса операнда, используемого в команде. В большинстве случаев команды, использующие индексный регистр Y, имеют один дополнительный байт в машинном коде и требуют один дополнительный цикл при выполнении по сравнению с соответствующими командами, использующими индексный регистр X. Второй индексный регистр особенно полезен при пересылках и в случае, когда используемые при вычислении операнды находятся в двух разных таблицах.

Указатель стека (SP). Стек может быть размещен в любом месте адресного пространства и может иметь любой размер вплоть до всего объема, доступного в системе. Обычно указатель стека инициализируется перед первой командой прикладной программы. Каждый раз, когда байт записывается в стек, указатель стека автоматически декрементируется, а при извлечении байта из стека – автоматически инкрементируется. При этом указатель стека всегда содержит адрес первой свободной ячейки в стеке. Стек используется для вызова подпрограмм, обработки прерываний, а также временного хранения данных.

Программный счетчик (PC). Программный счетчик представляет собой 16-разрядный регистр, в котором содержится адрес следующей подлежащей выполнению команды.

Регистр флагов (CCR). Этот регистр содержит пять признаков (код состояния), два бита маскирования прерываний и бит запрещения останова.

Пять флагов отражают результат арифметической или другой операции выполненной команды:

- дополнительный перенос (H);
- отрицательный результат (N);
- нулевой результат (Z);
- переполнение (V);
- перенос/заем (C).

Бит запрещения останова (S) используется для того, чтобы разрешить или запретить действие команды STOP. Некоторые пользователи считают команду STOP опасной, т. к. она блокирует системный генератор. Поэтому пользователь может установить бит S в единицу, чтобы запретить команду STOP. В этом

случае команда STOP интерпретируется процессором как пустая операция (NOP), и процессор переходит к следующей команде.

Маска (бит I) запроса прерывания (IRQ) представляет собой глобальную маску, которая запрещает все маскируемые прерывания. При инициализации микроконтроллера бит I устанавливается по умолчанию в единицу и может быть сброшен в ноль только по команде программы. После поступления запроса на прерывание бит I автоматически устанавливается в единицу после сохранения регистров в стеке, но до начала выполнения подпрограммы обработки прерывания, что приводит к запрещению прерываний. В конце обработки прерывания по команде RTI восстанавливается содержимое регистров, которое было в них до возникновения прерывания. При этом бит I сбрасывается в ноль (как это было до обработки прерывания), и прерывания опять разрешаются. Таким образом обслуживание новых запросов на прерывание запрещается на время обработки текущего запроса.

Маска XIRQ (бит X) используется для запрещения прерываний, поступающих на вход XIRQ#. При инициализации микроконтроллера бит X устанавливается по умолчанию в единицу и может быть сброшен в ноль только по команде программы. После поступления запроса на прерывание бит X (и бит I) автоматически устанавливается в единицу после сохранения регистров в стеке, но до начала выполнения подпрограммы обработки прерывания, что приводит к запрещению прерываний. В остальном бит X аналогичен биту I.

Микроконтроллер MC68HC11A8 может использоваться в одном из четырех режимов: однокристальном, расширенном, загрузки или тестирования.

Режимы задаются значениями внешних сигналов, поступающих на входы MODB, MODA при начальной установке (процедура RESET) (табл. 2). Рабочие режимы обеспечивают функционирование микроконтроллера с использованием только внутренней памяти (однокристальный режим) или с подключением внешней памяти к портам B, C (расширенный режим). Специальные режимы реализуют загрузку внутреннего ОЗУ из внешнего источника через асинхронный последовательный порт (режим загрузки) под управлением специальной программы, выбираемой из служебного ПЗУ при обращении к адресам $BF40-$BFFF, или тестирование микроконтроллера, которое выполняется заводом-изготовителем (режим тестирования).

Таблица 2

Режимы использования микроконтроллера MC68HC11A8

Режимы использования	Внешние выводы		Содержимое регистра HPRIO				
	MODB	MODA	RBOOT	SMOD	MDA	IRV	PSEL 3-0
Рабочие							
Однокристальный	1	0	0	0	0	0	0110
Расширенный	1	1	0	0	1	0	0110
Специальные							
Загрузка	0	0	1	1	0	1	0110
Тестирование	0	1	0	1	1	1	0110

Значения сигналов на входах MODA, MODB воспринимаются микроконтроллером только в процессе начальной установки. При последующей работе микроконтроллера на вывод MODA выдается сигнал *LIR#* = 0 при выборке очередной команды программы, что позволяет осуществлять внешний контроль за работой микроконтроллера в процессе отладки системы.

В процессе выполнения процедуры RESET устанавливается соответствующее значение в служебном регистре HPRIO (см. табл. 2). Разряды регистра HPRIO имеют следующее назначение (рис. 2):

- RBOOT – чтения служебного ПЗУ:
  1 = разрешается использование служебного ПЗУ при обращении к адресам $BF40-$BFFF для выборки специальной программы, выполняющей загрузку ОЗУ из внешнего источника через асинхронный последовательный порт;
  0 = выборка из служебного ПЗУ запрещена, и его адреса могут использоваться для обращения к внешней памяти;
- SMOD – специальные режимы:
  1 = специальные режимы;
  0 = рабочие режимы;
- MDA – выбор режима:
  1 = рабочий расширенный или специальный тестирования;
  0 = рабочий однокристальный или специальный загрузки;
- IRV – разрешение выдачи данных на внешние выводы порта С при обращении к внутренней памяти:
  1 = выдача данных разрешена;
  0 = выдача данных запрещена.
  Используется в режиме тестирования или отладки для контроля внутреннего обмена, при переходе в рабочий режим устанавливается IRV = 0;
- PSEL3-PSEL0 – приоритет обслуживания запросов прерывания.

7	6	5	4	3	2	1	0
RBOOT	**SMOD**	**MDA**	**IRV**	**PSEL3**	**PSEL2**	**PSEL1**	**PSEL0**

Рис. 2. Служебный регистр HPRIO (адрес $103C)

Значения разрядов RBOOT, SMOD, MDA, IRV могут быть записаны в регистр HPRIO только в специальном режиме (при SMOD = 1), в рабочем режиме они могут быть только считаны.

Микроконтроллер адресует до 64 Кбайт памяти (адреса $0000-$FFFF), которые делятся на 16 страниц по 4 Кбайта. Полный объем памяти доступен в расширенном режиме, когда к выводам портов В, С подключается внешняя память. В однокристальном режиме используется только внутренняя память микроконтроллера: ПЗУ, РПЗУ, ОЗУ, а порты В, С служат для обмена данными с внешними устройствами. Две последние страницы адресного пространства

(адреса $E000-$FFFF) занимает внутреннее масочное ПЗУ, содержимое которого программируется в процессе изготовления микроконтроллера по заказу пользователя. РПЗУ размещается по адресам $B600-$B7FF.

ОЗУ при начальной установке микроконтроллера (процедура RESET) занимает адресное пространство $0000-$00FF. ОЗУ можно перемещать в начало любой страницы адресного пространства путем загрузки содержимого битов RAM3-RAM0 в регистр INIT (рис. 4). При обращении к этим адресам производится выборка ОЗУ, а не ячеек ПЗУ или внешней памяти, расположенных на данной странице.

Распределение адресного пространства памяти микроконтроллера показано на рис. 3.

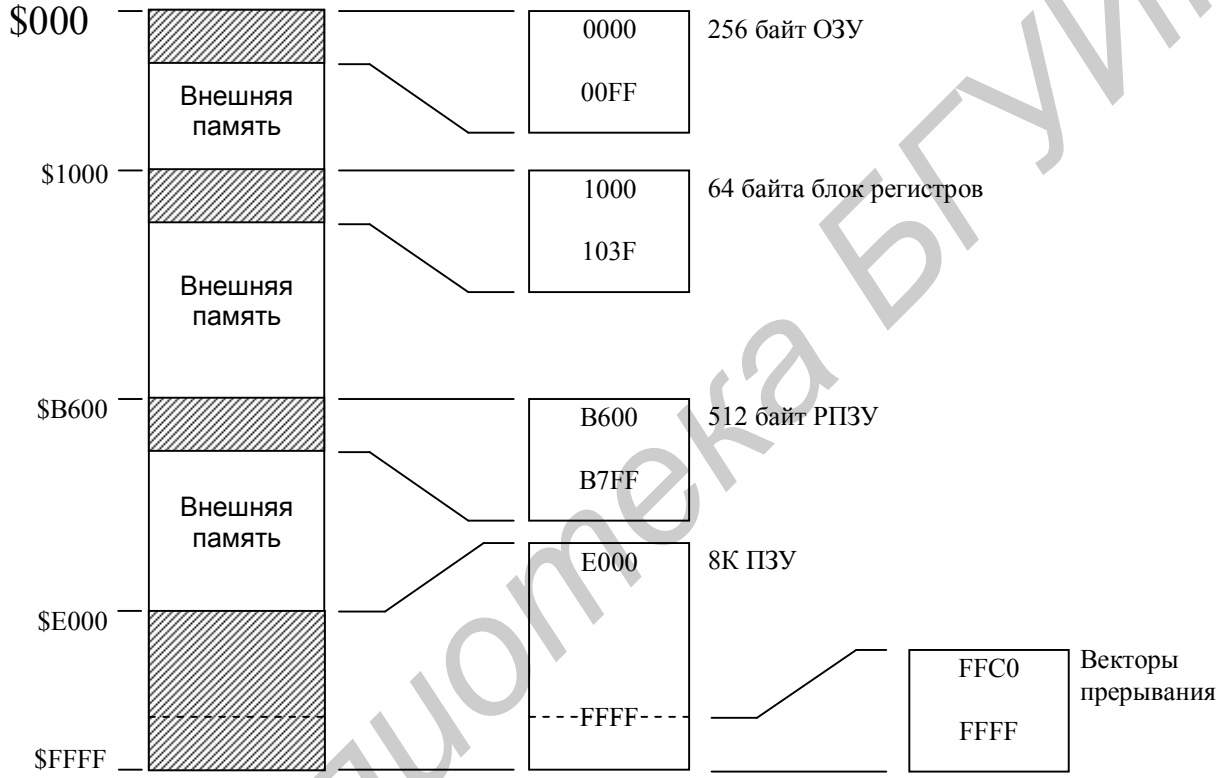


Рис. 3. Карта распределения адресного пространства

Внутренние регистры в микроконтроллере отображаются на память. Для обращения к ним выделено 96 адресов, которые при начальной установке (процедура RESET) располагаются в области $1000-$105F (начало страницы 1) адресного пространства (рис. 3). В число этих регистров, кроме служебных регистров процессора, входят следующие регистры: параллельных портов PORTA, PORTB, PORTC, PORTCL, DDRC, PORTD, DDRD, PORTE; последовательных портов SPDR, SPCR, SPSR, SCDR, SCCR1, SCCR2, SCSR, BRR; таймера TCNT, TIC1-TIC3, TOC1-TOC5, TMSK1, TMSK2, TFLG1, TFLG2, TCTL1, TCTL2, OCIM, OCID, CFORC (16-разрядные регистры TCNT, TIC, TOC занимают по два байта в адресном пространстве); счетчика импульсов PACTL, PACNT; блока контроля функционирования COPRST;

аналого-цифрового преобразователя ADR1-4, ADCTL. Регистровый блок можно перемещать в начало любой страницы адресного пространства путем загрузки разрядов REG3-REG0 в регистре INIT (рис. 4). При обращении к данным адресам производится выборка соответствующего регистра, а не размещенных на этой странице ячеек ОЗУ, ПЗУ или внешней памяти. Часть из 96 адресов в этой модели микроконтроллера не используется, они зарезервированы для использования различными периферийными блоками в других моделях данного семейства.

Служебные регистры определяют конфигурацию и режимы работы микроконтроллера. Назначение и формат регистра HPRIO были рассмотрены выше.

Содержимое регистра INIT (рис. 4) определяет старшие четыре разряда адреса (номер страницы) размещения ОЗУ (биты RAM3-RAM0) и блока внутренних регистров (биты REG3-REG0). При начальной установке микроконтроллера биты данного регистра принимают значения RAM3-RAM0 = 0000 (обращение к странице 0), REG3-REG0 = 0001 (обращение к странице 1).

7	6	5	4	3	2	1	0
RAM3	**RAM2**	**RAM1**	**RAM0**	**REG3**	**REG2**	**REG1**	**REG0**

Рис. 4. Служебный регистр INIT (адрес $103D)

Регистр CONFIG (рис. 5) определяет конфигурацию микроконтроллера, разрешая или запрещая использование внутренних ПЗУ, РПЗУ и блока контроля функционирования. Разряды этого регистра имеют следующее назначение:

- NOSEC – выключение защиты РПЗУ:
  1 = защита РПЗУ выключена;
  0 = защита внутренней памяти от внешнего считывания включена. Включение защиты разрешает работу микроконтроллера только в однокристальном режиме или режиме загрузки, которые не обеспечивают выдачу содержимого внутренней памяти на внешние выводы. При этом в регистре HPRIO устанавливается значение бита MDA = 0 независимо от сигнала на входе MODA в процессе начальной установки;
- NOCOP – выключение схемы контроля выполнения программы, входящей в состав блока контроля функционирования (БКФ) микроконтроллера:
  1 = схема контроля выполнения программы БКФ выключена и не формирует сигнала системного сброса;
  0 = схема контроля выполнения программы БКФ включена;
- ROMON – включение внутреннего ПЗУ:
  1 = внутреннее ПЗУ включено;

0 = внутреннее ПЗУ выключено и не занимает место в адресном пространстве МК (можно использовать адреса $E000-$FFFF для обращения к внешней памяти);

- EEON – включение внутреннего РПЗУ:
  1 = внутреннее РПЗУ включено;
  0 = внутреннее РПЗУ выключено и не занимает место в адресном пространстве МК (можно использовать адреса $B600-$B7FF для обращения к внешней памяти).

7	6	5	4	3	2	1	0
0	0	0	0	**NOSEC**	**NOCOP**	**ROMO N**	**EEON**

Рис. 5. Служебный регистр CONFIG (адрес $103F)

Регистр OPTION (рис. 6) определяет функционирование отдельных блоков микроконтроллера. Разряды этого регистра имеют следующее назначение:

- ADPU – включение питание аналого-цифрового преобразователя:
  1 = питание АЦП включено;
  0 = питание АЦП выключено;
- CSEL – разрешение периодического заряда емкостей в АЦП и РПЗУ:
  1 = используется внутренний RC-генератор;
  0 = заряд емкостей производится тактовыми импульсами МК;
- IRQE – конфигурирование входа запроса прерывания IRQ#:
  1 = запросом служит поступление отрицательного перепада потенциала на входе IRQ#;
  0 = запросом служит подача низкого уровня на вход IRQ#;
- DLY – включение задержки начала функционирования микроконтроллера после выхода из режима останова:
  1 = задержка начала функционирования микроконтроллера после выхода из режима останова составляет около 4000 тактов ГТИ. Эта задержка необходима, чтобы обеспечить установку заданной рабочей частоты ГТИ;
  0 = задержка функционирования составляет четыре такта;
- CME – включение схемы контроля тактовой частоты, входящей в состав БКФ:
  1 = схема контроля выполнения программы БКФ включена;
  0 = схема контроля тактовой частоты БКФ выключена и не формирует сигнала системного сброса;
- CR1, CR0 – определяют коэффициент деления частоты при работе схемы контроля выполнения программы, входящей в состав БКФ, в соответствии с табл. 3. Этот коэффициент определяет длительность интервалов времени T_m при контроле выполнения программы.

Таблица 3

Коэффициент деления частоты и длительность интервалов времени T_m схемы контроля выполнения программы

CR1	CR2	Коэффициент деления частоты $F_t / 2^{15}$	T_m		
			Выражение для расчета	При $F_g = 2^{23}$ Гц F_t = 2,1 МГц	При F_g = 8 МГц F_t = 2 МГц
0	0	1	$32768T_C$	15,625 мс	16,384 мс
0	1	4	$(32768T_C) \times 4$	62,5 мс	65,536 мс
1	0	16	$(32768T_C) \times 16$	250 мс	262,14 мс
1	1	64	$(32768T_C) \times 64$	1 с	1,049 мс

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

Рис. 6. Служебный регистр OPTION (адрес $1039)

Содержимое регистров HPRIO, INIT, OPTION записывается непосредственно после начальной установки микроконтроллера и затем может быть только считано. Исключение составляют только разряды PSEL3-PSEL0 в регистре HPRIO и разряды ADPU, CSEL в регистре OPTION, которые могут изменяться в процессе работы микроконтроллера путем записи в эти регистры. Содержимое регистра CONFIG программируется таким же образом, как и содержимое внутреннего РПЗУ, и может изменяться только в процессе репрограммирования. Этим обеспечивается его сохранение при отключении питания.

Длительность такта микроконтроллера $T_C = 1/F_t$ определяется генератором тактовых импульсов (ГТИ). Частота следования тактовых импульсов F_t задается кварцевым резонатором, подключаемым к выводам EXTAL и XTAL, или внешним генератором, подключаемым к выводу XTAL. При этом частота F_t в 4 раза меньше частоты внешнего резонатора или генератора $F_t = F_g/4$. Импульсы с частотой F_t поступают на выход E микроконтроллера и используются для синхронизации работы других устройств системы. Максимальное значение F_t для данной модели составляет 4 МГц при номинальном напряжении питания $V_П$ = 5 В. Ток, потребляемый от источника питания в рабочем режиме при $V_П$ = 5 В, F_t = 4 МГц, не превышает 50 мА.

Микроконтроллер имеет два режима функционирования с пониженным энергопотреблением. В режиме ожидания, который реализуется после поступления команды WAI, прекращает работу процессор, но продолжают функционировать последовательные порты и таймер. При этом максимальный потребляемый ток снижается до 25 мА при $V_П$ = 5 В, F_t = 4 МГц. Выход из

режима ожидания происходит при поступлении запроса прерывания от таймера, СПП или АПП. В режиме останова, который реализуется после поступления команды STOP, прекращается работа процессора, таймера, СПП и АПП. При этом максимальный потребляемый ток уменьшается до 50 мкА. Выход из этого режима происходит при поступлении внешнего запроса прерывания на вход IRQ# или внешнего сигнала сброса *RESET#*.

Установка начального состояния микроконтроллера (процедура сброса RESET) производится в следующих случаях:

1) включение напряжения питания $V_П$;
2) поступление внешнего сигнала сброса *RESET#* = 0;
3) поступление сигнала сброса от блока контроля функционирования.

Если микроконтроллер работает в рабочем режиме (однокристальном или расширенном), то в процессе начальной установки при включении питания или поступлении сигнала *RESET#* = 0 в программный счетчик PC загружаются два байта: старший байт PC_H из ячейки памяти с адресом $FFFE, младший байт PC_L – $FFFF. Эти байты являются адресом первой команды, выполняемой микроконтроллером после начальной установки. В однокристальном режиме данные байты выбирается из внутреннего ПЗУ, при расширенном режиме – из внутреннего ПЗУ, если в регистре CONFIG значение бита ROMON = 1, или из внешней памяти, если ROMON = 0. Если микроконтроллер работает в специальном режиме (загрузка или тестирование), то содержимое PC загружается из ячеек внешней памяти с адресами $BFFE, $BFFF. При включении питания требуется время $4064T_C$ для запуска ГТИ и установки начального состояния регистров, после чего начинается нормальная работа микроконтроллера. Для начальной установки по сигналу *RESET#* его длительность должна быть не меньше $4T_C$.

2. БЛОК КОНТРОЛЯ ФУНКЦИОНИРОВАНИЯ МК MC68HC11A8

Блок контроля функционирования (БКФ) осуществляет контроль выполнения программы и частоты тактовых импульсов. Если БКФ обнаруживает нарушение функционирования, то производится сброс микроконтроллера в начальное состояние. При этом на выводе RESET# устанавливается низкий уровень, сохраняющийся в течение 16 тактов.

Контроль выполнения программы в БКФ осуществляется при установке в регистре CONFIG значения бита NOCOP = 0 путем периодической записи в регистр COPRST (адрес $103A) чисел $55, $AA с промежутком времени не более T_m. Величина T_m определяется значением бит CR1-CR0 в регистре OPTION (рис. 6) в соответствии с табл. 3. Если в течение времени T_m не произведено указанное изменение содержимого регистра COPRST, то осуществляется сброс микроконтроллера в начальное состояние (процедура RESET). При этом в программный счетчик PC загружается содержимое из ячеек памяти с адресами $FFFA, $FFFB в рабочих режимах или из ячеек с адресами $BFFA, $BFFB в специальных режимах функционирования. Таким

образом, в выполняемой программе должны присутствовать команды, осуществляющие периодическую запись чисел $55, $AA в регистр COPRST. Прекращение или сбой выполнения программы, нарушающие периодичность записи этих чисел, указывают на неправильную работу МК и вызывают его сброс.

Схема контроля частоты тактовых импульсов в БКФ включается при установке в регистре OPTION значения бита CME = 1. Она производит сброс микроконтроллера в начальное состояние (процедура RESET), если частота тактовых импульсов F_t < 200 кГц. При этом в программный счетчик PC загружается содержимое из ячеек памяти с адресами $FFFC, $FFFB в рабочих режимах или из ячеек с адресами $BFFA, $BFFB в специальных режимах. Поэтому данную схему следует отключать установкой значения бита CME = 0, если предполагается работа микроконтроллера с тактовой частотой ниже 200 кГц.

3. СПОСОБЫ АДРЕСАЦИИ И СИСТЕМА КОМАНД МК MC68HC11A8

Микроконтроллер выполняет операции над 8- или 16-разрядными операндами, размещенными в регистрах процессора и памяти, и реализует набор из 108 команд. Команды имеют длину от одного до четырех байт: первый байт содержит код операции, перед ним может идти префиксный байт, указывающий на обращение к одному из индексных регистров X или Y (имеет значения $18, $1A или $CD), второй и третий байты обеспечивают адресацию операнда. Два варианта команд условных ветвлений по значению бита BRCLR, BRSET с индексной адресацией, использующие регистр Y, содержат пять байт.

В микроконтроллере M68HC11A8 для обращения к памяти используется шесть способов адресации:

1) непосредственная;
2) прямая;
3) расширенная (длинная прямая адресация);
4) индексная;
5) неявная (регистровая);
6) относительная.

Непосредственная адресация (IMM). При данном способе адресации операнд содержится в байте (байтах), непосредственно следующем за командой. Число байтов определяется длиной используемого регистра. Команды, использующие непосредственную адресацию, состоят из двух, трех или четырех (если требуется префиксный байт) байт. Непосредственная адресация обозначается в команде с помощью символа #.

Примеры использования непосредственной адресации приведены ниже. В примерах используются следующие префиксы, определяющие тип константы:

Префикс	Определение
Нет	Десятичный
$	Шестнадцатеричный
@	Восьмеричный
%	Двоичный
'	ASCII символ

Машинный код			Метка	Операция	Операнд	Комментарии
			CAT	EQU	7	CAT равно 7
86	16			LDAA	#22	Десятичное 22 ⇒ ACCA
C8	34			EORB	#$34	XOR ($34,ACCB) ⇒ ACCB
81	24			CMPA	#%100100	ACCA - %100100
86	07			LDAA	#CAT	7 ⇒ ACCA
CC	12	34		LDD	#$1234	$1234 ⇒ ACCA:ACCB
CC	00	07		LDD	#7	7 ⇒ ACCA:ACCB
86	12			LDAA	#@22	Восьмеричное 22 ⇒ ACCA
86	41			LDAA	#'A	ASCII

Расширенная (длинная прямая адресация, EXT). При данном способе адресации эффективный адрес явно указывается в двух байтах, следующих за кодом операции в команде. Поэтому длина команд, использующих расширенную адресацию, составляет три или четыре байта: один или два для кода операции и два для эффективного адреса. Последние два байта команды содержат абсолютный адрес операнда.

Прямая адресация (DIR). При прямом способе адресации в байте команды, следующем за кодом операции, содержится младший байт эффективного адреса. Старший байт эффективного адреса принимается равным $00 и не включается в команду (для экономии пространства памяти программ и времени выполнения). Этот факт ограничивает использование прямой адресации для обращения к области памяти $0000-$00FF. Длина команд, использующих прямую адресацию, составляет два байта: один для кода операции и один для эффективного адреса. В большинстве применений область памяти $0000-$00FF соответствует ОЗУ, что удобно для размещения в этой области часто используемых переменных (данных).

В микроконтроллерах семейства M68HC11 часть команд допускает использование как прямой, так и расширенной адресации. Некоторые команды допускают использование только расширенной адресации. Следующие примеры иллюстрируют использование прямой и расширенной адресации.

Машинный код			Метка	Операция	Операнд	Комментарии
B3	00	12		SUBD	CAT	Расширенная адресация
			CAT	EQU	$12	CAT определяется равным $12
93	12			SUBD	CAT	Прямая адресация
7F	00	12		CLR	CAT	Только расширенная адресация

В первой команде SUBD ссылка на метку CAT является ссылкой вперед, поэтому ассемблер выбирает расширенную адресацию. Во второй команде SUBD ссылка на метку CAT осуществляется назад, что позволяет ассемблеру определить значение CAT на этапе обработки команды и выбирать для данной команды прямой способ адресации. В последней команде ассемблер использует расширенную адресацию, так как команда CLR допускает только этот способ адресации.

Индексная адресация (INDX, INDY). При индексном способе адресации для вычисления эффективного адреса используется один из индексных регистров X или Y: текущее содержимое индексного регистра суммируется с фиксированным 8-разрядным беззнаковым смещением, заданным в команде. Этот способ адресации может использоваться для обращения к любой ячейке в 64-Кбайтном адресном пространстве. Команды, использующие индексную адресацию, состоят из двух или трех (если требуется префиксный байт) байт: код операции и 8-разрядное смещение. Смещение представляет собой беззнаковое однобайтное значение (0-255), которое добавляется к текущему значению индексного регистра для получения эффективного адреса, оставляя содержимое индексного регистра без изменения.

Примеры использования непосредственной адресации приведены ниже.

Машинный код		Метка	Операция	Операнд	Комментарии
E3	00		ADDD	X	EA = (X)
E3	00		ADDD	,X	EA = (X)
E3	00		ADDD	0,X	EA = (X)
E3	04		ADDD	4,X	EA = (X) + 4
		CAT	EQU	7	CAT определяется равным 7
E3	07		ADDD	CAT,X	EA = (X) + 7
E3	22		ADDD	$22,X	EA = (X) + $22
E3	22		ADDD	CAT*8/2+6,X	EA = (X) + 34

Неявная адресация (INH). При неявной адресации все необходимое для выполнения команды заранее известно процессору. В этом случае операнды или операнд представляют собой регистры процессора и их не надо выбирать из памяти. Такие команды содержат один или два байта.

Многие команды микроконтроллера M68HC11A8 используют один или больше регистров в качестве операндов. Например, команда ABA заставляет процессор сложить содержимое аккумуляторов A и B и поместить результат в аккумулятор A. Команда INCB увеличивает содержимое аккумулятора B на единицу. Аналогично по команде INX на единицу увеличивается содержимое индексного регистра X. Все эти три команды являются примерами использования неявной адресации:

Машинный код	Метка	Операция	Операнд	Комментарии
1B		ABA		$A + B \Rightarrow A$
5C		INCB		$B + 1 \Rightarrow B$
08		INX		$X + 1 \Rightarrow X$

Относительная адресация (REL). Относительная адресация используется только в командах ветвления. Для команд ветвления формируется два байта машинного кода: один для кода операции и второй для относительного смещения. Поскольку желательно осуществлять переходы в обоих направлениях, смещение представляет собой байт со знаком, представленный в дополнительном коде (от -128 до +127). Если условие перехода выполняется, то байт, следующий за кодом операции (смещение), прибавляется к содержимому программного счетчика, в результате чего формируется эффективный адрес перехода. В противном случае выполняется команда, непосредственно следующая за командой ветвления.

Байт смещения является всегда последним байтом в команде ветвления. Если байт смещения равен нулю, то выполняется команда, непосредственно следующая за командой ветвления, независимо от значения проверяемого условия. По команде безусловного ветвления (BRA) со смещением $FE организуется бесконечный пустой цикл (переход на себя).

Примеры использования относительной адресации приведены ниже.

Машинный код			Метка	Операция	Операнд	Комментарии
20	00		THERE	BRA	WHERE	Безусловное ветвление вперед
22	FC		WHERE	BHI	THERE	Ветвление назад
24	04			BCC	LBCC	
27	FE		HANG	BEQ	HANG	Ветвление на себя
27	FE			BEQ	*	Эквивалентно предыдущей команде
7E	10	00	LBCC	JMP	$1000	

Полный список команд микроконтроллера M68HC11A8 приведен в приложении. Рассмотрим особенности выполнения некоторых команд.

Команды пересылки осуществляют загрузку операндов из памяти в регистры, запись содержимого регистров в память или передачу между регистрами. При загрузке (команды LD, LDA) используются все способы адресации, кроме относительного, при записи в память (команды ST, STA) не используется также непосредственная адресация. При загрузке и записи в память содержимого регистра D (аккумулятор двойной разрядности), указателя стека SP или индексных регистров X, Y адресуемая ячейка памяти содержит старший байт (D_H, SP_H, X_H или Y_H), следующая за ней ячейка – младший байт (D_L, SP_L, X_L или Y_L) используемого регистра. Команды PSH выполняют запись содержимого регистров в стек, команды PUL — загрузку регистров из стека. Команды очистки CLR производят запись 0 в регистры A, B или ячейку памяти. Команды XGDX, XGDY выполняют обмен содержимым регистра D с индексным регистром X или Y.

Команды арифметических операций в большинстве случаев выполняют действия над операндами, один из которых располагается в памяти (M), а второй – в аккумуляторе A, B или D, куда помещается затем результат. При операциях сложения и вычитания второй операнд (M) адресуется любым способом, кроме относительного. Команды ADDD, SUBD осуществляют сложение и вычитание 16-разрядных операндов. Один из них располагается в

аккумуляторе D, а второй (M) выбирается из двух рядом расположенных ячеек памяти (в команде задается адрес старшего байта). Команда десятичной коррекции результата DAA выполняется после команд сложения ABA, ADDA, ADDB, ADCA, ADCB, если операндами служили упакованные двоично-десятичные числа.

Операция умножения MUL выполняется над 8-разрядными операндами без знака, расположенными в регистрах A и B. 16-разрядное произведение размещается в аккумуляторе D. Операция деления выполняется над двумя 16-разрядными операндами без знака, размещенными в регистрах D и X. Частное от деления располагается в регистре X, остаток r в D. В микроконтроллере имеется два типа команды деления. Команда целочисленного деления IDIV используется, когда делимое больше делителя. Если же делимое меньше делителя, то используется команда дробного деления FDIV, которая сдвигает делимое на 16 разрядов влево, а затем делит его на делитель. При этом в регистре X получается двоичное число, представляющее дробную часть результата. Выполняя последовательное деление остатка с помощью команды FDIV, можно получать дробную часть результата с требуемой точностью.

Команды битовых операций BCLR, BSET устанавливают значение 0 или 1 для одного или нескольких бит в операнде, адрес которого содержится во втором байте команды (прямая адресация) или формируется с помощью индексной адресации. В последнем байте команды задается значение маски устанавливаемых бит *mm*. Если *i*-й бит маски $m_i = 1$, то соответствующий бит операнда принимает значение 0 или 1 в соответствии с выполняемой командой; если $m_i = 0$, то соответствующий бит сохраняет свое значение. Таким образом, данные команды позволяют одновременно устанавливать заданные значения нескольких бит операнда. Команды CLC, CLI, CLV и SEC, SEI, SEV устанавливают значение 0 или 1 признаков C, I, V в регистре флагов CCR. При этом команды CLI и SEI разрешают и запрещают обслуживание маскируемых запросов прерывания.

Для управления выполнением программы используются команды безусловного перехода JMP, условных и безусловных ветвлений B*cc*, BRCLR, BRSET и BRA, BRN, перехода и ветвления к подпрограмме JSR, BSR, возврата из подпрограммы RTS, программного прерывания SWI, возврата из прерывания RTI. Команды условного ветвления Bcc производят загрузку в PC нового адреса, если выполняются соответствующие условия (см. приложение). Мнемокод условия *cc* приписывается к букве B, образуя мнемокод соответствующей команды условного ветвления, например BMI – мнемокод команды ветвления при отрицательном результате предыдущей операции (признак N = 1).

Команда программного прерывания SWI сохраняет в стеке содержимое программного счетчика PC и регистров процессора Y, X, A, B, CCR. При выполнении этой команды заполняется девять ячеек стека. Причем в стек загружается сначала младший, а затем старший байт соответствующего регистра. Команда возврата из прерывания RTI производит восстановление

прежнего содержимого регистров CCR, A, B, X, Y, PC из стека. Команда NOP не выполняет никаких операций, она используется для задания временной задержки длительностью два такта. Для управления процессором служат команды WAI, STOP, переводящие микроконтроллер в энергосберегающие режимы ожидания и останова. При этом команда WAI осуществляет запись в стек содержимого всех регистров процессора аналогично команде SWI.

Большинство команд микроконтроллера для своего выполнения требуют от 2 до 8 тактов. Исключение составляют команды: MUL – 10 тактов; FDIV, IDIV – 41; SWI – 14; RTI, WAI – 12.

4. ПАРАЛЛЕЛЬНЫЕ ПОРТЫ ВВОДА/ВЫВОДА МК MC68HC11A8

Параллельный обмен данными между микроконтроллером и внешними устройствами производится с помощью пяти портов A, B, C, D, E.

Каждый из параллельных портов A, B, C, D, E имеет регистр данных PORTx, где x = A, B, C, D, E, в который поступают принимаемые или выводимые данные. Если порт используется для вывода, то данные, записываемые в его регистр PORTx, поступают на выводы порта. При этом они сохраняются в данном регистре и могут быть затем считаны. Если порт используется для приема, то данные, поступившие на выводы порта, заносятся в этот регистр и могут быть считаны при обращении к нему. Обмен данными осуществляется между аккумуляторами A, B и регистрами данных портов с помощью команд LDAA, LDAB и STAA, STAB, адресующими регистр соответствующего порта.

Порт A (регистр PORTA, адрес $1000) имеет восемь внешних выводов, из которых три служат входами PA2-PA0, пять – выходами PA6-PA3, один является двунаправленным выводом PA7. Назначение данного вывода определяется разрядом DDRA7 в регистре управления PACTL (рис. 7): при DDRA7 = 0 вывод PA7 используется как вход, при DDRA7 = 1 – как выход.

7	6	5	4	3	2	1	0
DDRA7	**PAEN**	**PAMOD**	**PEDGE**	0	0	**RTR1**	**RTR0**

Рис. 7. Регистр управления PACTL (адрес $1026)

Кроме использования для ввода/вывода данных выводы порта A могут использоваться таймером или блоком счета импульсов (БСИ), которые описываются ниже.

Функционирование **портов B и C** зависит от режима использования микроконтроллера. При работе в однокристальном режиме и режиме загрузки эти порты служат для ввода/вывода данных, при работе в расширенном режиме и режиме тестирования они служат для обращения к внешней памяти. При работе этих портов используется блок квитированного обмена (БКО), который обеспечивает прием и выдачу сигналов квитирования *STRA*, *STRB*. Функционирование портов B и C определяется регистром управления PIOC (рис. 8), входящим в состав БКО. При значении разряда HNDS = 0 в этом

регистре порт B работает в режиме стробированного вывода, порт C – в режиме стробированного ввода. При значении HNDS = 1 порт B работает в режиме нестробированного вывода, а порт C используется для двунаправленного обмена с квитированием.

7	6	5	4	3	2	1	0
STAF	**STAI**	**CWOM**	**HNDS**	**OIN**	**PLS**	**EGA**	**INVB**

Рис. 8. Регистр управления PIOC (адрес $1002)

Порт B (регистр PORTB, адрес $1004) используется как 8-разрядный порт вывода данных. Если в регистре управления PIOC установлено значение разряда HNDS = 0, то порт B работает в режиме стробированного вывода. В этом режиме запись данных в регистр PORTB сопровождается формированием импульса на выходе STRB длительностью $2T_C$, который служит для внешних устройств запросом на прием данных с выводов порта PB7-PB0. Активный уровень этого импульса определяется значением разряда INVB в регистре PIOC: при INVB = 0 формируется импульс низкого уровня, при INVB = 1 – высокого уровня. Если значение разряда HNDS = 1, то порт B работает в режиме нестробированного вывода, при котором не происходит формирование стробирующего сигнала на выходе STRB. При работе в расширенном режиме или режиме тестирования на выводы этого порта при обращении к памяти поступает старший байт адреса A15-A8.

Порт C используется как 8-разрядный двунаправленный порт. Направление передачи данных определяется для каждого вывода порта путем установки значения соответствующего разряда в регистре направления DDRC (адрес $1007). При установке в этом регистре значения *i*-го разряда в 0 *i*-й вывод порта используется как вход, при установке в 1 – как выход. При этом в качестве выходного каскада порта используются схемы с "открытым стоком", если в регистре PIOC установлено значение разряда CWOM = 1.

Порт C имеет два регистра данных: PORTC (адрес $1003) и PORTCL (адрес $1005). Запись данных в регистр PORTCL с выводов порта PC7-PC0 стробируется сигналом на входе STRA, запись или чтение регистра PORTC не сопровождается сигналами квитирования. При чтении регистра PORTC в микроконтроллер вводятся данные, соответствующие текущему состоянию выводов PC7-PC0, при чтении регистра PORTCL вводятся данные, записанные в этот регистр при подаче стробирующего сигнала на вход STRA. При записи в регистры PORTC и PORTCL данные поступают также на выводы порта C, используемые в качестве выходов. При этом запись в PORTCL сопровождается формированием соответствующих сигналов квитирования на выводах STRA и STRB, как это описано ниже.

Таким образом, при обращении к регистру PORTC порт C служит для нестробированного двунаправленного обмена данными. При обращении к регистру PORTCL работа порта зависит от значения разряда HNDS в регистре PIOC.

Если значение HNDS = 0, то выполняется стробированный ввод данных в регистр PORTCL при поступлении сигнала на вход STRA. При этом если в регистре PIOC разряд EGA = 0, то ввод производится по отрицательному сигналу на этом входе, если EGA = 1 – по положительному. Если значение HNDS = 1, то порт C при обращении к регистру PORTCL осуществляет двунаправленный обмен с использованием выводов STRA, STRB для сигналов квитирования. В этом случае разряды OIN, PLS в регистре PIOC определяют формирование сигналов квитирования. При OIN = 0 осуществляется квитированный ввод, когда по стробирующему сигналу, поступающему на вход STRA, данные с выводов PC7-PC0 записываются в регистр PORTCL. После считывания этого регистра на выходе STRB устанавливается сигнал подтверждения приема, разрешающий внешнему устройству выдачу новых данных. При OIN = 1 выполняется квитированный вывод. В этом случае после записи в регистр PORTCL данные поступают на выводы PC7-PC0, и на выходе STRB формируется сигнал готовности, являющийся запросом на их прием внешним устройством. Выполнив прием, внешнее устройство подает на вход STRA сигнал подтверждения. Активный уровень сигнала, поступающего на вход STRA, определяется значением разряда EGA: положительный перепад при EGA = 1, отрицательный перепад при EGA = 0. Активный уровень сигнала, формируемого на выходе STRB, определяется значением разряда INV, как описано выше. При значении бита PLS = 1 активный сигнал на выходе STRB поддерживается в течение времени $2T_C$, при PLS = 0 активный сигнал сохраняется до поступления ответного сигнала от внешнего устройства на вход STRA.

При поступлении активного сигнала на вход STRA в регистре PIOC устанавливается признак STAF = 1, который вызывает формирование запроса прерывания, если разряд разрешения STAI = 1. По этому запросу должна быть выполнена запись или считывание регистра PORTCL. При этом происходит сброс признака STAF в 0. Такая последовательность действий необходима для того, чтобы избежать потери информации при квитированном обмене данными с внешним устройством.

При использовании микроконтроллера в расширенном режиме или режиме тестирования выводы PC7-PC0 порта C служат в качестве линий мультиплексированной передачи адреса/данных AD7-AD0.

Двунаправленный **порт D** (регистр PORTD, адрес $1008) имеет шесть выводов PD5-PD0, которые служат входами или выходами данных в зависимости от установки значения битов в регистре направления DDRD (адрес $1009). При этом в качестве выходных каскадов порта используются схемы с "открытым стоком", если в регистре SPCR (адрес $1028) установлено значение пятого разряда DWOM = 1.

Выводы порта D совмещены со входами и выходами последовательных портов АПП, СПП.

Порт E (регистр PORTE, адрес $100A) является 8-разрядным портом ввода данных. Входы порта E служат для приема аналоговых сигналов при работе АЦП.

5. ТАЙМЕР МК MC68HC11A8

Таймер реализован на базе 16-разрядного счетчика TCNT (адрес старшего байта $100E, адрес младшего байта $100F). Этот счетчик запускается при начальной установке микроконтроллера, и после запуска его состояние может быть только считано, например командами LDD, LDX, LDY. Частота счета зависит от значения разрядов PR1-PR0 в регистре TMSK2 (рис. 9, г) и составляет $F_{CNT} = F_t/K_d$, где значение K_d определяется по табл. 4. Таким образом, при тактовой частоте $F_t = 2$ МГц временное разрешение таймера равно $T_{CNT} = 500$ нс, а максимальное время счета при $K_d = 16$ составляет 524 мс. При переключении счетчика из состояния $FFFF в состояние $0000 устанавливается признак переполнения TOF = 1 в регистре TFLG2 (рис. 9, в). При этом формируется запрос прерывания, если в регистре TMSK2 (см. рис. 9, г) установлен разряд разрешения данного запроса TOI = 1.

Таблица 4

Коэффициенты K_d и K_t таймера

PR1	PR0	Коэффициент деления частоты таймера K_d	RTR1	RTR0	Коэффициент K_t
0	0	1	0	0	1
0	1	4	0	1	2
1	0	8	1	0	4
1	1	16	1	1	8

Функционирование таймера определяется 8-разрядными регистрами управления, которые показаны на рис. 9.

Таймер может использоваться для **периодической генерации запросов прерывания**. Для этого используется специальный флаг RTIF в регистре TFLG2. Этот флаг принимает значение RTIF = 1 через заданное время $T_t = 8192T_CK_t$, где величина коэффициента K_t определяется разрядами RTR1-RTR0 (см. табл. 4) в регистре PACTL (см. рис. 9). При этом формируется запрос прерывания, если разряд разрешения данного запроса RTII в регистре TMSK2 имеет значение RTII = 1. Сброс признака RTIF в состояние 0 производится путем записи соответствующего значения в регистр TFLG2.

Таймер может работать в режиме захвата и режиме сравнения.

В **режиме захвата** поступление сигнала на вход IC1, IC2 или IC3 таймера вызывает запись текущего содержимого счетчика в соответствующий 16-разрядный регистр TIC1, TIC2 или TIC3. Данные регистры имеют адреса: TIC1 — $1010, 1011; TIC2 — $1012, 1013; TIC3 — $1014, 1015 (четные адреса соответствуют старшим байтам регистров). При этом в регистре TFLG1 устанавливается значение признака захвата IC1F, IC2F или IC3F, равное

единице. Установка признака захвата вызывает формирование запроса прерывания таймера, если в регистре TMSK1 установлено в единицу значение соответствующего разряда разрешения прерывания IC1I, IC2I или IC3I. Последующее считывание содержимого регистра TIC1, TIC2 или TIC3 позволяет определить время поступления соответствующего сигнала захвата. В качестве входов для приема сигналов захвата IC1, IC2 или IC3 служат выводы PA2-PA0 порта A. Вид сигнала, вызывающего захват, определяется значением разрядов EDGxB, EDGxA в регистре управления TCTL2 в соответствии с табл. 5.

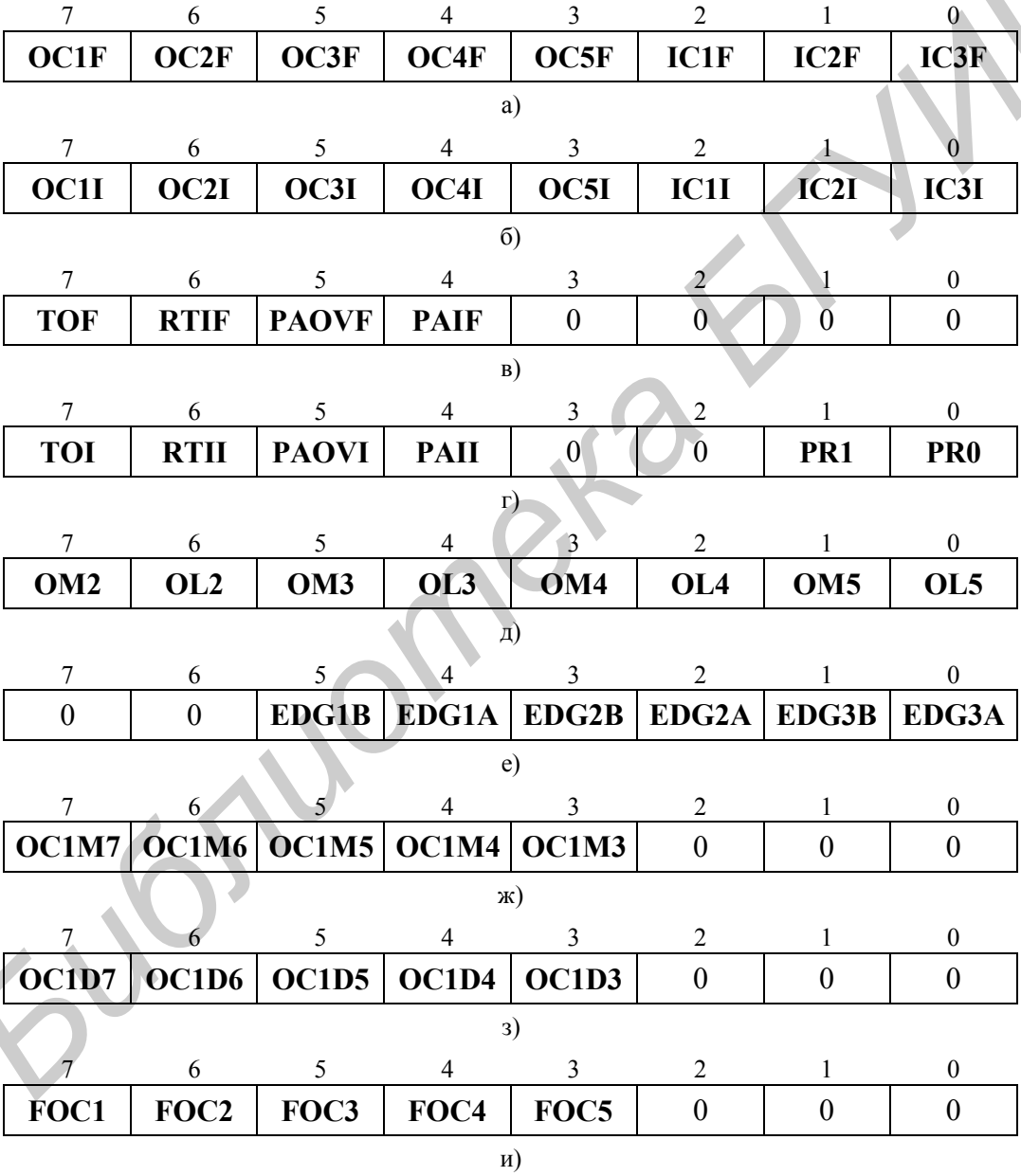


Рис. 9. Регистры управления таймера

а – TFLG1 (адрес $1023); б – TMSK1 (адрес $1022); в – TFLG2 (адрес $1025); г – TMSK2 (адрес $1024); д – TCTL1 (адрес $1020); е – TCTL2 (адрес $1021); ж – OC1M (адрес $100C); з – OC1D (адрес $100D); и – CFORC (адрес $100B).

Таблица 5

Виды сигналов захвата ICx и совпадения OCx

EDGxB	EDGxA	Сигнал захвата ICx	OMx	OLx	Состояние выхода OCx
0	0	Запрещение захвата	0	0	Не изменяется
0	1	Положительный перепад	0	1	Переключение
1	0	Отрицательный перепад	1	0	Установка в 0
1	1	Любой перепад	1	1	Установка в 1

В **режиме сравнения** происходит сравнение содержимого счетчика таймера с предварительно записанными в 16-разрядные регистры TOCx, где x = 1, 2, 3, 4 или 5 числами. Эти регистры имеют адреса: TOC1 – $1016, 1017; TOC2 – $1018, 1017; TOCÇ – $101A, 101B; TOC4 – $101C, 101D; TOC5 – $101E, 101F. Когда содержимое счетчика становится равным числу, записанному в каком-либо регистре TOCx, формируется сигнал на соответствующем выходе OCx и устанавливается значение признака совпадения OCxF = 1 в регистре TFLG1. Этот признак вызывает формирование запроса прерывания таймера, если в регистре TMSK1 установлено значение соответствующего разряда разрешения прерывания OCxI = 1. В качестве выходов сигналов совпадения OCx используются выводы PA7-PA3 порта A. Изменение состояния этих сигналов определяется значениями битов OMx, OLx в регистре TCTL1 (см. табл. 5). Если эти выводы служат для выдачи выходных сигналов таймера, то при записи в регистр PORTA данные не поступают на соответствующие выводы PA7-PA3. Только после установки значений OMx = OLx = 0 данные из регистра PORTA будут поданы на эти выводы. Так как направление передачи данных через вывод PA7 определяется значением бита DDRA7 в регистре PACTL (см. рис. 7), то при использовании этого вывода для формирования сигнала совпадения OC1 необходимо установить DDRA7 = 1.

В режиме сравнения таймер может быть запрограммирован на формирование в определенные моменты времени заданных выходных кодов. Для этого используются регистр сравнения TOC1 и дополнительные регистры OC1M, OC1D. В регистр TOC1 записывается число, определяющее момент выдачи заданного кода. В регистре OC1M устанавливаются значения разрядов OCIMx = 1 для тех выводов PA7-PA3, на которые должен поступать заданный код. В регистр OC1D записывается код, поступающий на эти выводы в момент совпадения содержимого TOC1 и TCNT. Таким образом, может выдаваться код, содержащий до пяти разрядов. Если при этом часть выводов PA7-PA3 не используется, то они могут программироваться для выдачи отдельных сигналов совпадения или служить для вывода данных из регистра PORTA.

С помощью регистра CFORC можно программно форсировать выдачу сигналов совпадения на выводы PA7-PA0 до наступления фактического совпадения содержимого TOCx и TCNT. Для этого необходимо в регистре CFORC установить значение соответствующего разряда FOCx = 1. Выводы, для

которых в регистре CFORC значения FOCx = 0, не изменяют своего значения при таком форсировании.

6. БЛОК СЧЕТЧИКА ИМПУЛЬСОВ МК MC68HC11A8

Блок счетчика импульсов (БСИ) содержит 8-разрядный счетчик PCNT (адрес $1027). Управление БСИ осуществляется с помощью регистра PACTL (см. рис. 9). Счетчик функционирует при установке в этом регистре значения разряда PAEN = 1. При этом вывод PA7 порта A становится входом PAI счетчика. БСИ может работать в двух режимах: счета событий или стробируемого таймера. Режим работы задается значением разряда PAMOD: счет событий при PAMOD = 0, стробируемый таймер при PAMOD = 1. Значение разряда PEDGE определяет вид входного сигнала PAI, переключающего счетчик или запускающего таймер. В режиме счета событий переключение PCNT происходит при поступлении соответствующего фронта сигнала на вход PAI: отрицательного при значении бита PEDGE = 0, положительного при PEDGE = 1. В режиме стробируемого таймера PCNT переключается внутренними импульсами таймера, частота следования которых равна $F_{CNT}/64$, где F_{CNT} – частота переключения основного таймера. Сигнал PAI разрешает или запрещает переключение таймера: если значение бита PEDGE = 0, счет разрешается при PAI = 1 и запрещается при PAI = 0, если значение PEDGE = 1, то наоборот.

При работе БСИ в регистре TFLG2 (см. рис. 9, в) устанавливаются два признака:

1) признак переполнения счетчика PAOVF = 1, когда содержимое PCNT изменяется с $FF на $00;

2) признак поступления входного сигнала PAIF = 1, когда на вход PAI подается сигнал, заданный значением бита PEDGE.

Каждый из этих сигналов вызывает формирование запроса прерывания, если соответствующий разряд разрешения PAOVI или PAII в регистре TMSK2 (см. рис. 9, г) равен 1.

Благодаря наличию таймера и блока счетчика импульсов микроконтроллеры семейства M68HC11 обладают широкими возможностями для выполнения таких процедур, как измерение временных интервалов, формирование управляющих кодов в заданные моменты времени, выдача нескольких сигналов с различными задержками, периодическое выполнение необходимых подпрограмм, генерация импульсов, многоканальная широтно-импульсная модуляция и др.

7. ПРЕРЫВАНИЯ В МК MC68HC11A8

При выполнении процедур начальной установки и обслуживания прерываний микроконтроллер обращается к таблице прерываний, которая содержит 16-разрядные адреса входа в соответствующие подпрограммы

обслуживания. Эта таблица занимает последние 64 адреса в адресном пространстве МК $FFC0-$FFFF (см. рис. 3). Для обращения к подпрограмме обслуживания микроконтроллер в цикле подтверждения прерывания формирует вектор прерывания, который представляет собой адрес старшего байта адреса входа в подпрограмму. В следующих циклах МК считывает из таблицы адрес входа в подпрограмму обслуживания, который заносится в программный счетчик. Таким образом, обеспечивается обращение к 32 различным подпрограммам обслуживания. Микроконтроллер MC68HC11A8 использует только 21 точку входа из 32: три для подпрограмм начальной установки, три для подпрограмм обслуживания немаскируемых прерываний и пятнадцать для подпрограмм обслуживания маскируемых запросов прерывания Соответствующие векторы прерывания приведены в табл. 6, 7. Остальные точки входа (вектора $FFD4-$FFC0) зарезервированы для использования в других моделях.

Таблица 6

Векторы прерывания при установке начального состояния и немаскируемых запросах прерывания

Сигналы установки начального состояния, немаскируемые запросы прерывания	Вектор
Внешний сигнал RESET# или включение питания	$FFFE
Сигнал схемы контроля частоты тактовых импульсов	$FFFC
Сигнал схемы контроля выполнения программы	$FFFA
Неправильный код операции	$FFF8
Программное прерывание SWI	$FFF6
Немаскируемый внешний запрос *XIRQ#*	$FFF4

Три первых вектора $FFFE, $FFFC, $FFFA в табл. 6 используются для установки начального состояния в рабочих режимах использования микроконтроллера. Три следующих вектора $FFF8, $FFF6, $FFF4 служат для обращения к подпрограммам обслуживания немаскируемых запросов прерывания, которые формируются при поступлении неправильного кода операции, команды программного прерывания SWI или внешнего сигнала на вход XIRQ#. Обслуживание запроса XIRQ# запрещается, если в регистре условий CCR (см. рис. 1) установлено значение бита X = 1. Последовательность обслуживания данных запросов определяется их приоритетом, который соответствует порядку размещения векторов в табл. 6: максимальный приоритет имеют сигнал *RESET#* и включение питания, минимальный приоритет – немаскируемый внешний запрос XIRQ#.

Таблица 7

Векторы и приоритеты обслуживания маскируемых прерываний

PSEL3-PSEL0	Запрос прерывания	Вектор
0110	Внешний сигнал *IRQ#*	$FFF2
0111	Периодическое прерывание таймера	$FFF0
1000	Сигнал захвата таймера TIC1	$FFFE
1001	Сигнал захвата таймера TIC2	$FFEC
1010	Сигнал захвата таймера TIC3	$FFEA
1011	Сигнал совпадения таймера TOC1	$FFE8
1100	Сигнал совпадения таймера TOC2	$FFE6
1101	Сигнал совпадения таймера TOC3	$FFE4
1110	Сигнал совпадения таймера TOC4	$FFE2
1111	Сигнал совпадения таймера TOC5	$FFE0
0000	Переполнение таймера	$FFDE
0001	Переполнение счетчиков импульсов	$FFDC
0010	Входной сигнал PAI счетчика	$FFDA
0011	Запрос СПП	$FFD8
0100	Запрос АПП	$FFD6

К числу маскируемых запросов прерывания относятся внешний сигнал *IRQ#* и внутренние сигналы, формируемые периферийными блоками микроконтроллера: таймером, счетчиком импульсов и портами АПП, СПП. Запрещение (маскирование) обслуживания этих запросов производится установкой значения признака I = 1 в регистре условий CCR. Приоритет их обслуживания определяется содержимым поля PSEL3-PSEL0 в регистре HPRIO (см. рис. 2). В процессе начальной установки в этом регистре устанавливается значение PSEL3-PSEL0 = 0110, при котором максимальный приоритет из маскируемых запросов получает внешний сигнал *IRQ#*. Приоритеты остальных запросов соответствуют порядку размещения их векторов в табл. 7, при этом минимальный приоритет имеет запрос от СПП. Путем изменения содержимого поля PSEL3-PSEL0 можно изменять приоритеты запросов. При этом запрос, код которого устанавливается в поле PSEL3-PSEL0, получает максимальный приоритет, а приоритеты остальных запросов циклически изменяются с сохранением их относительных позиций в соответствии с табл. 7 (минимальный приоритет получает запрос, предшествующий запросу с максимальным приоритетом).

При реализации прерывания в стек загружается содержимое основных регистров процессора в такой последовательности: CCR, A, B, X_H, X_L, Y_H, Y_L, PC_H, PC_L, где X_H, Y_H, PC_H и X_L , Y_L, PC_L – соответственно старший и младший

байты регистров X, Y, PC. После этого в программный счетчик PC загружается адрес первой команды подпрограммы обслуживания из таблицы прерываний в соответствии с векторами прерываний (см. табл. 6, 7). Выполнение подпрограммы обслуживания должно заканчиваться командой возврата из прерывания RTI, которая выбирает из стека и восстанавливает содержимое регистров PC, Y, X, B, A, CCR, обеспечивая таким образом продолжение прерванной программы.

ЗАКЛЮЧЕНИЕ

Помимо перечисленных периферийных модулей некоторые модели микроконтроллеров семейства MC68HC11 содержат синтезаторы частот, контроллеры прямого доступа к памяти, сопроцессоры для ускоренного выполнения операций умножения и деления и некоторые другие устройства.

Основными областями применения микроконтроллеров данного семейства являются средства беспроводной связи, телефония, локальные системы сбора информации и управления, автомобильная электроника, сложная бытовая техника.

Л И Т Е Р А Т У Р А

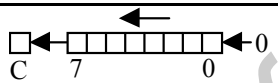
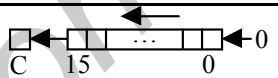
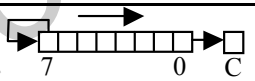
1. Motorola Master Selection Guide. SG73/D. Rev 17. © Motorola, Inc. 1998.
2. Motorola MC68HC11 Reference Manual. Rev 3. © Motorola, Inc. 1991.
3. Шагурин И.И. Микропроцессоры и микроконтроллеры фирмы Motorola: Справ. пособие. – М.: Радио и связь, 1998.

П Р И Л О Ж Е Н И Е

Система команд микроконтроллера MC68HC11A8

Мнемоника	Операция	Выполняемое действие	Способ адресации операнда	Машинный код (hex)		Байты	Цикл	Признаки							
				Код операции	Операнд(ы)			S	X	H	I	N	Z	V	C
1	2	3	4	5	6	7	8	9							
ABA	Добавить B к A	A + B ⇒ A	INH	1B		1	2			□		□	□	□	□
ABX	Добавить B к X	IX + 00:B ⇒ IX	INH	3A		1	3								
ABY	Добавить B к Y	IY + 00:B ⇒ IY	INH	18 3A		2	4								
ADCA (opr)	Добавить с переносом к аккумулятору A	A + M + C ⇒ A	A IMM	89	ii	2	2			□		□	□	□	□
			A DIR	99	dd	2	3								
			A EXT	B9	hh ll	3	4								
			A IND, X	A9	ff	2	4								
			A IND, Y	18 A9	ff	3	5								
ADCB (opr)	Добавить с переносом к аккумулятору B	B + M + C ⇒ B	B IMM	C9	ii	2	2			□		□	□	□	□
			B DIR	D9	dd	2	3								
			B EXT	F9	hh ll	3	4								
			B IND,X	E9	ff	2	4								
			B IND,Y	18 E9	ff	3	5								
ADDA (opr)	Добавить память к аккумулятору A	A + M ⇒ A	A IMM	8B	ii	2	2			□		□	□	□	□
			A DIR	9B	dd	2	3								
			A EXT	BB	hh ll	3	4								
			A IND, X	AB	ff	2	4								
			A IND, Y	18 AB	ff	3	5								
ADDB (opr)	Добавить память к аккумулятору B	B + M ⇒ B	B IMM	CB	ii	2	2			□		□	□	□	□
			B DIR	DB	dd	2	3								
			B EXT	FB	hh ll	3	4								
			B IND, X	EB	ff	2	4								
			B IND, Y	18 EB	ff	3	5								
ADDD (opr)	Добавить 16-разрядный операнд к аккумулятору D	D + M:M + 1 ⇒ D	IMM	C3	jj kk	3	4					□	□	□	□
			DIR	D3	dd	2	5								
			EXT	F3	hh ll	3	6								
			IND, X	E3	ff	2	6								
			IND, Y	18 EB	ff	3	7								

Продолжение приложения

1	2	3	4	5	6	7	8	9
ANDA (opr)	Побитовое И аккумулятора А и памяти	A & M □A	A IMM A DIR A EXT A IND, X A IND, Y	84 94 B4 A4 18 A4	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □
ANDB (opr)	Побитовое И аккумулятора В и памяти	B & M □B	B IMM B DIR B EXT B IND, X B IND, Y	C4 D4 F4 E4 18 E4	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □
ASL (opr)	Арифметический сдвиг влево		EXT IND,X IND,Y	78 68 18 68	hh ll ff ff	3 2 3	6 6 7	□ □ □ □
ASLA			A INH	48		1	2	
ASLB			B INH	58		1	2	
ASLD	Двойной арифметический сдвиг влево		INH	05		1	3	□ □ □ □
ASR (opr)	Арифметический сдвиг вправо		EXT IND,X IND,Y	77 67 18 67	hh ll ff ff	3 2 3	6 6 7	□ □ □ □
ASRA			A INH	47		1	2	
ASRB			B INH	57		1	2	
BCC (rel)	Переход, если нет переноса	? C = 0	REL	24	rr	2	3	
BCLR (opr) (msk)	Очистить бит(ы)	M & ($\overline{mm}$ □M	DIR IND,X IND,Y	15 1D 18 1D	dd mm ff mm ff mm	3 3 4	6 7 8	□ □

BCS (rel)	Переход, если есть перенос	? C = 1	REL	25	rr	2	3	
BEQ (rel)	Переход, если равно нулю	? Z = 1	REL	27	rr	2	3	
BGE (rel)	Переход, если больше или равно нулю	? N ⊕ V = 0	REL	2C	rr	2	3	
BGT (rel)	Переход, если больше нуля	? Z + (N ⊕ V) = 0	REL	2E	rr	2	3	
BHI (rel)	Переход, если больше	? C + Z = 0	REL	22	rr	2	3	
BHS (rel)	Переход, если больше или равно	? C = 0	REL	24	rr	2	3	

Продолжение приложения

1	2	3	4	5	6	7	8	9
BITA (opr)	Побитовое сравнение аккумулятора A и байта памяти	A & M	A IMM A DIR A EXT A IND, X A IND, Y	85 95 B5 A5 18 A5	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □
BITB (opr)	Побитовое сравнение аккумулятора B и байта памяти	B & M	B IMM B DIR B EXT B IND, X B IND, Y	C5 D5 F5 E5 18 E5	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □
BLE (rel)	Переход, если меньше или равно нулю	? Z + (N ⊕ V) = 1	REL	2F	rr	2	3	
BLO (rel)	Переход, если меньше	? C = 1	REL	25	rr	2	3	
BLS (rel)	Переход, если меньше или равно	? C + Z = 1	REL	23	rr	2	3	
BLT (rel)	Переход, если меньше нуля	? N ⊕ V = 1	REL	2D	rr	2	3	
BMI (rel)	Переход, если минус	? N = 1	REL	2B	rr	2	3	
BNE (rel)	Переход, если не равно нулю	? Z = 0	REL	26	rr	2	3	
BPL (rel)	Переход, если плюс	? N = 0	REL	2A	rr	2	3	
BRA (rel)	Переход, всегда	? 1 = 1	REL	20	rr	2	3	
BRCLR (opr) (msk) (rel)	Переход, если бит(ы) сброшен(ы)	? M & mm = 0	DIR IND,X IND,Y	13 1F 18 1F	dd mm rr ff mm rr ff mm rr	4 4 5	6 7 8	
BRN (rel)	Переход никогда	? 1 = 0	REL	21	rr	2	3	
BRSET (opr) (msk) (rel)	Переход, если бит(ы) установлен(ы)	? $\overline{M}$ & mm = 0	DIR IND,X IND,Y	12 1E 18 1E	dd mm rr ff mm rr ff mm rr	4 4 5	6 7 8	

BSET (opr) (msk)	Установить бит(ы)	M + mm □M	DIR IND,X IND,Y	14 1C 18 1C	dd mm rr ff mm rr ff mm rr	3 3 4	6 7 8		□ □
BSR (rel)	Переход к подпрограмме		REL	8D	rr	2	6		
BVC (rel)	Переход, если нет переполнения	? V = 0	REL	28	rr	2	3		
BVS (rel)	Переход, если переполнение	? V = 1	REL	29	rr	2	3		
CBA	Сравнить A с B	A – B	INH	11		1	2		□ □ □ □
CLC	Очистить флаг переноса C	0 □C	INH	0C		1	2		

Продолжение приложения

1	2	3	4	5	6	7	8	9
CLI	Сбросить маску прерывания I	0 □I	INH	0E		1	2	
CLR (opr)	Очистить байт памяти	0 □M	EXT IND,X IND,Y	7F 6F 18 6F	hh ll ff ff	3 2 3	6 6 7	1
CLRA	Очистить аккумулятор A	0 □A	A INH	4F		1	2	1
CLRB	Очистить аккумулятор B	0 □B	B INH	5F		1	2	1
CLV	Очистить флаг переполнения	0 □V	INH	0A		1	2	
CMPA (opr)	Сравнить A с байтом памяти	A –M	A IMM A DIR A EXT A IND,X A IND,Y	81 91 B1 A1 18 A1	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □ □ □
CMPB (opr)	Сравнить B с байтом памяти	B – M	B IMM B DIR B EXT B IND,X B IND,Y	C1 D1 F1 E1 18 E1	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □ □ □
COM (opr)	Инвертирование (дополнение до 1) байта памяти	$FF – M □M	EXT IND,X IND,Y	73 63 18 63	hh ll ff ff	3 2 3	6 6 7	□ □ 1
COMA	Инвертирование (дополнение до 1) A	$FF – A □A	A INH	43		1	2	□ □ 1
COMB	Инвертирование (дополнение до 1) B	$FF – M □B	B INH	53		1	2	□ □ 1

CPD (opr)	Сравнить D с 16-разрядной памятью	D – M:M + 1	IMM DIR EXT IND,X IND,Y	1A 83 1A 93 1A B3 1A A3 CD A3	jj kk dd hh ll ff ff	4 3 4 3 3	5 6 7 7 7	□ □ □ □
CPX (opr)	Сравнить X с 16-разрядной памятью	IX -- M:M + 1	IMM DIR EXT IND,X IND,Y	8C 9C BC AC CD AC	jj kk dd hh ll ff ff	3 2 3 2 3	4 5 6 6 7	□ □ □ □

Продолжение приложения

1	2	3	4	5	6	7	8	9
CPY (opr)	Сравнить Y с 16-разрядной памятью	IY -- M:M + 1	IMM DIR EXT IND,X IND,Y	18 8C 18 9C 18 BC 1A AC 18 AC	jj kk dd hh ll ff ff	4 3 4 3 3	5 6 7 7 7	□ □ □ □
DAA	Десятичная коррекция A	Adjust Sum to BCD	INH	19		1	2	□ □ □ □
DEC (opr)	Декремент байта памяти	M – 1 □M	EXT IND,X IND,Y	7A 6A 18 6A	hh ll ff ff	3 2 3	6 6 7	□ □ □
DECA	Декремент аккумулятора A	A – 1 □A	A INH	4A		1	2	□ □ □
DECB	Декремент аккумулятора B	B – 1 □B	B INH	5A		1	2	□ □ □
DES	Декремент указателя стека	SP – 1 □SP	INH	34		1	3	
DEX	Декремент индексного регистра X	IX – 1 □IX	INH	09		1	3	□
DEY	Декремент индексного регистра Y	IY – 1 □IY	INH	18 09		2	4	□
EORA (opr)	Исключающее ИЛИ аккумулятора A с байтом памяти	A □M □A	A IMM A DIR A EXT A IND,X A IND,Y	88 98 88 A8 18 A8	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □

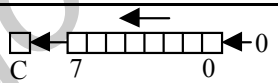
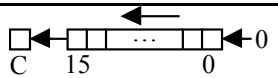
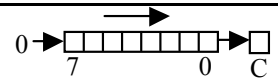
EORB (opr)	Исключающее ИЛИ аккумулятора B с байтом памяти	B ⊕ M ⇒ B	B IMM B DIR B EXT B IND,X B IND,Y	C8 D8 F8 E8 18 E8	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □
FDIV	Дробное 16-разрядное деление	D / IX ⇒ IX; r ⇒ D	INH	03		1	41	□ □ □
IDIV	Целое 16-разрядное деление	D / IX ⇒ IX; r ⇒ D	INH	02		1	41	□ □
INC (opr)	Инкремент байта памяти	M + 1 ⇒ M	EXT IND,X IND,Y	7C 6C 18 6C	hh ll ff ff	3 2 3	6 6 7	□ □ □
INCA	Инкремент аккумулятора A	A + 1 ⇒ A	A INH	4C		1	2	□ □ □
INCB	Инкремент аккумулятора B	B + 1 ⇒ B	B INH	5C		1	2	□ □ □
INS	Инкремент указателя стека	SP + 1 ⇒ SP	INH	31		1	3	
INX	Инкремент индексного регистра X	IX + 1 ⇒ IX	INH	08		1	3	□
INY	Инкремент индексного регистра Y	IY + 1 ⇒ IY	INH	18 08		2	4	□

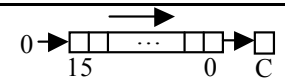
Продолжение приложения

1	2	3	4	5	6	7	8	9
JMP (opr)	Безусловный переход		EXT IND,X IND,Y	7E 6E 18 6E	hh ll ff ff	3 2 3	3 3 4	
JSR (opr)	Переход к подпрограмме		DIR EXT IND,X IND,Y	9D BD AD 18 AD	dd hh ll ff ff	2 3 2 3	5 6 6 7	
LDAA (opr)	Загрузить аккумулятор A	M ⇒ A	A IMM A DIR A EXT A IND,X A IND,Y	86 96 B6 A6 18 A6	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □
LDAB (opr)	Загрузить аккумулятор B	M ⇒ B	B IMM B DIR B EXT B IND,X B IND,Y	C6 D6 F6 E6 18 E6	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □

LDD (opr)	Загрузить 16-разрядный аккумулятор D	M □A, M + 1 □B	IMM	CC	jj kk	3	3		□ □
			DIR	DC	dd	2	4		
			EXT	FC	hh ll	3	5		
			IND,X	EC	ff	2	5		
			IND,Y	18 EC	ff	3	6		
LDS (opr)	Загрузить указатель стека	M:M+1 □SP	IMM	8E	jj kk	3	3		□ □
			DIR	9E	dd	2	4		
			EXT	BE	hh ll	3	5		
			IND,X	AE	ff	2	5		
			IND,Y	18 AE	ff	3	6		
LDX (opr)	Загрузить индексный регистр X	M:M+1 □IX	IMM	CE	jj kk	3	3		□ □
			DIR	DE	dd	2	4		
			EXT	FE	hh ll	3	5		
			IND,X	EE	ff	2	5		
			IND,Y	CD EE	ff	3	6		

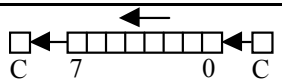
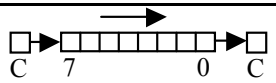
Продолжение приложения

1	2	3	4	5	6	7	8	9
LDY (opr)	Загрузить индексный регистр Y	M:M+1 □IY	IMM	18 CE	jj kk	4	4	□ □
			DIR	18 DE	dd	3	5	
			EXT	18 FE	hh ll	4	6	
			IND,X	1A EE	ff	3	6	
			IND,Y	18 EE	ff	3	6	
LSL (opr)	Логический сдвиг влево		EXT	78	hh ll	3	6	□ □ □ □
			IND,X	68	ff	2	6	
			IND,Y	18 68	ff	3	7	
LSLA			A INH	48		1	2	
LSLB			B INH	58		1	2	
LSDL	Двойной логический сдвиг влево		INH	05		1	3	□ □ □ □
LSR (opr)	Логический сдвиг вправо		EXT	74	hh ll	3	6	□ □ □
			IND,X	64	ff	2	6	
			IND,Y	18 64	ff	3	7	
LSRA			A INH	44		1	2	
LSRB			B INH	54		1	2	

LSRD	Двойной логический сдвиг вправо		INH	04		1	3	□ □ □
MUL	8-разрядное умножение	A × B □D	INH	3D		1	10	□
NEG (opr)	Изменить знак байта памяти (дополнение до 2)	0 – M □M	EXT IND,X IND,Y	70 60 18 60	hh ll ff ff	3 2 3	6 6 7	□ □ □ □
NEGA	Изменить знак аккумулятора A (дополнение до 2)	0 – A □A	A INH	40		1	2	□ □ □ □
NEGB	Изменить знак аккумулятора B (дополнение до 2)	0 – B □B	B INH	50		1	2	□ □ □ □
NOP	Пустая операция		INH	01		1	2	
ORAA (opr)	Побитовое ИЛИ аккумулятора A и байта памяти	A + M □A	A IMM A DIR A EXT A IND,X A IND,Y	8A 9A BA AA 18 AA	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □

Продолжение приложения

1	2	3	4	5	6	7	8	9
ORAB (opr)	Побитовое ИЛИ аккумулятора B и байта памяти	B + M □B	B IMM B DIR B EXT B IND,X B IND,Y	CA DA FA EA 18 EA	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	□ □
PSHA	Занести A в стек	A □Stk, SP = SP – 1	A INH	36		1	3	
PSHB	Занести B в стек	B □Stk, SP= SP – 1	B INH	37		1	3	
PSHX	Занести X в стек (младший байт первым)	IX □Stk, SP= SP – 2	INH	3C		1	4	
PSHY	Занести Y в стек (младший байт первым)	IX □Stk, SP= SP – 2	INH	18 3C		2	5	
PULA	Извлечь A из стека	SP = SP + 1, A □Stk	A INH	32		1	4	
PULB	Извлечь B из стека	SP = SP + 1, B □Stk	B INH	33		1	4	
PULX	Извлечь X из стека (старший байт первым)	SP = SP + 2, IX □Stk	INH	38		1	5	
PULY	Извлечь Y из стека (старший байт первым)	SP = SP + 2, IY □Stk	INH	18 38		2	6	

ROL (opr)	Циклический сдвиг влево		EXT IND,X IND,Y	79 69 18 69	hh ll ff ff	3 2 3	6 6 7	˜ ˜ ˜ ˜ □ □ □ □
ROLA			A INH	49		1	2	
ROLB			B INH	59		1	2	
ROR (opr)	Циклический сдвиг вправо		EXT IND,X IND,Y	76 66 18 66	hh ll ff ff	3 2 3	6 6 7	˜ ˜ ˜ ˜ □ □ □ □
RORA			A INH	46		1	2	
RORB			B INH	56		1	2	
RTI	Возврат из прерывания		INH	3B		1	12	□ □ □ □ □ □ □ □
RTS	Возврат из подпрограммы		INH	39		1	5	˜ ˜ ˜ ˜ ˜ ˜ ˜ ˜
SBA	Вычесть B из A	A – B □A	INH	10		1	2	˜ ˜ ˜ ˜ □ □ □ □

Продолжение приложения

1	2	3	4	5	6	7	8	9
SBCA (opr)	Вычесть с переносом из A	A – M – C □A	A IMM A DIR A EXT A IND,X A IND,Y	82 92 B2 A2 18 A2	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	˜ ˜ ˜ ˜ □ □ □ □
SBCB (opr)	Вычесть с переносом из B	A – M – C □B	B IMM B DIR B EXT B IND,X B IND,Y	C2 D2 F2 E2 18 E2	ii dd hh ll ff ff	2 2 3 2 3	2 3 4 4 5	˜ ˜ ˜ ˜ □ □ □ □
SEC	Установить флаг переноса	1 □C	INH	0D		1	2	˜ ˜ ˜ ˜ ˜ ˜ ˜ 1
SEI	Установить маску прерывания I	1 □I	INH	0F		1	2	˜ ˜ ˜ ˜ 1 ˜ ˜ ˜
SEV	Установить флаг переполнения	1 □V	INH	0B		1	2	˜ ˜ ˜ ˜ ˜ ˜ 1 ˜
STAA (opr)	Сохранить аккумулятор A	A □M	A DIR A EXT A IND,X A IND,Y	97 B7 A7 18 A7	dd hh ll ff ff	2 3 2 3	3 4 4 5	˜ ˜ ˜ ˜ □ □ ˜ ˜

STAB (opr)	Сохранить аккумулятор B	B ⍗M	B DIR	D7	dd	2	3	~ ~ ~ ~ □ □ ~ ~
			B EXT	F7	hh ll	3	4	
			B IND,X	E7	ff	2	4	
			B IND,Y	18 E7	ff	3	5	
STD (opr)	Сохранить аккумулятор D	A ⍗M, B ⍗M + 1	DIR	DD	dd	2	4	~ ~ ~ ~ □ □ ~ ~
			EXT	FD	hh ll	3	5	
			IND,X	ED	ff	2	5	
			IND,Y	18 ED	ff	3	6	
STOP	Остановить внутренний системный генератор		INH	CF		1	2	~ ~ ~ ~ ~ ~ ~ ~
STS (opr)	Сохранить указатель стека	SP ⍗M:M + 1	DIR	9F	dd	2	4	~ ~ ~ ~ □ □ ~ ~
			EXT	BF	hh ll	3	5	
			IND,X	AF	ff	2	5	
			IND,Y	18 AF	ff	3	6	
STX (opr)	Сохранить индексный регистр X	IX ⍗M:M + 1	DIR	DF	dd	2	4	~ ~ ~ ~ □ □ ~ ~
			EXT	FF	hh ll	3	5	
			IND,X	EF	ff	2	5	
			IND,Y	CD EF	ff	3	6	

Продолжение приложения

1	2	3	4	5	6	7	8	9
STY (opr)	Сохранить индексный регистр Y	IY ⍗M:M + 1	DIR	18 DF	dd	3	5	~ ~ ~ ~ □ □ ~ ~
			EXT	18 FF	hh ll	4	6	
			IND,X	1A EF	ff	3	6	
			IND,Y	18 EF	ff	3	6	
SUBA (opr)	Вычесть байт памяти из A	A – M ⍗A	A IMM	80	ii	2	2	~ ~ ~ ~ □ □ □ □
			A DIR	90	dd	2	3	
			A EXT	B0	hh ll	3	4	
			A IND,X	A0	ff	2	4	
			A IND,Y	18 A0	ff	3	5	
SUBB (opr)	Вычесть байт памяти из B	B – M ⍗B	B IMM	C0	ii	2	2	~ ~ ~ ~ □ □ □ □
			B DIR	D0	dd	2	3	
			B EXT	F0	hh ll	3	4	
			B IND,X	E0	ff	2	4	
			B IND,Y	18 E0	ff	3	5	

SUBD (opr)	Вычесть слово памяти из D	D – M:M + 1 □D	IMM	83	jj kk	3	4	□ □ □ □
			DIR	93	dd	2	5	
			EXT	B3	hh ll	3	6	
			IND,X	A3	ff	2	6	
			IND,Y	18 A3	ff	3	7	
SWI	Программное прерывание		INH	3F		1	14	1
TAB	Скопировать A в B	A □B	INH	16		1	2	□ □
TAP	Скопировать A в регистр флагов CCR	A □CCR	INH	06		1	2	□ □ □ □ □ □ □ □
TBA	Скопировать B в A	B □A	INH	17		1	2	□ □
TEST	Тестирование (только в режиме тестирования)		INH	00		1		
TPA	Скопировать регистр флагов CCR в A	CCR □A	INH	07		1	2	
TST (opr)	Проверка на нуль или минус	M – 0	EXT	7D	hh ll	3	6	□ □
			IND,X	6D	ff	2	6	
			IND,Y	18 6D	ff	3	7	
TSTA		A – 0	A INH	4D		1	2	□ □
TSTB		B – 0	B INH	5D		1	2	□ □
TSX	Скопировать указатель стека в X	SP + 1 □IX	INH	30		1	3	
TSY	Скопировать указатель стека в Y	SP + 1 □IY	INH	18 30		2	4	
TXS	Скопировать X в указатель стека	IX – 1 □SP	INH	35		1	3	

Окончание приложения

1	2	3	4	5	6	7	8	9
TYS	Скопировать Y в указатель стека	IY – 1 □SP	INH	18 35		2	4	
WAI	Ожидание прерывания		INH	3E		1		
XGDX	Поменять D и X	IX □D, D □IX	INH	8F		1	3	
XGDY	Поменять D и Y	IY □D, D □IY	INH	18 8F		2	4	

Операнды:

dd – 8-разрядный прямой адрес $0000-$00FF. (Старший байт принимается равным $00).
ff – 8-разрядное положительное смещение от $00 (0) до $FF (255). (Используется при индексной адресации).
hh – старший байт 16-разрядного расширенного адреса.
ii – один байт непосредственных данных.
jj – старший байт непосредственных 16-разрядных данных.
kk – младший байт непосредственных 16-разрядных данных.

ll – младший байт 16-разрядного расширенного адреса.
mm – 8-разрядная маска (set Bits to be Affected).
rr – знаковое относительное смещение от $80 (-128) до $7F (+127). Смещение относительно адреса первого байта следующей команды.

Признаки:

– – не изменяется.
0 – сбрасывается в нуль.
1 – устанавливается в единицу.
 – сбрасывается в нуль или устанавливается в единицу в зависимости от результата операции.
 – может быть сброшен в нуль, но не установлен в единицу.