# Однокристальная микроЭВМ

# МС68НС705С8

## Введение

Однокристальная микроЭВМ (далее ОЭВМ) МС68НС705С8 входит в серию М68НС05 относительно дешевых ОЭВМ, использующих КМОП-технологию, сочетающую малые размеры и высокое быстродействие с низким потреблением энергии и высокой устойчивостью к помехам.

Дополнительным преимуществом КМОП-структуры является то, что компоновка схемы полностью статична, т.е. ОЭВМ может работать при любой частоте синхронизации, меньшей, чем гарантированный максимум. Эта особенность может использоваться в целях экономии энергии, так как ее потребление возрастает с увеличением частоты синхронизации. Кроме того, для снижения потребления энергии используются еще два программно-управляемых режима работы с экономией энергии: Ожидание (WAIT) и Останов (STOP). Благодаря этим режимам ОЭВМ MC68HC705C8 находит особо широкое применение в автомобильной технике, а также в аппаратуре, питающейся от батарей.

Индекс "705С8" в названии говорит о принадлежности данной ОЭВМ к серии С, в которую входят микросхемы общего назначения с наиболее гибкими параметрами. Цифра 8 в названии говорит о том, что это наиболее быстродействующая версия, допускающая частоту внешней синхронизации 8 МГц. Цифры 705 говорят об отличии от базовой модели МС68НС05С8 в конфигурации памяти, состоящее в том, что в базовой модели предусмотрено 8К ПЗУ, а в ее модификации МС68НС705С8 ПЗУ заменено на ППЗУ с ультрафиолетовым стиранием либо однократно программируемым ППЗУ.

Кроме упомянутых ОЭВМ, серия М68НС05 включает в себя ОЭВМ с драйверами жидкокристалических, вакуумных и других дисплеев, ЦАП и АЦП, таймерами, ОЭВМ с различными конфигурациями памяти, быстродействием и другими отличительными особенностями.

## СОДЕРЖАНИЕ

 Страница

Введение 1

СОДЕРЖАНИЕ 2

1. Программная модель ОЭВМ МС68НС705С8. 3

1.1 Внутренняя структура. 3

1.2 Память. 5

1.3 Регистры. 6

1.4. Режимы адресации. 9

1.5. Набор команд. 15

2. Аппаратные средства 28

2.1 Линии (порты) параллельного ввода-вывода. 28

2.2 Система прерываний. 29

2.3 Таймер и связанные с ним устройства. 33

2.4 Последовательный интерфейс 36

3. Специальные режимы работы. 47

3.1 RESET. 47

3.2 Режимы с пониженным энергопотреблением. 49

Приложение 1.

Расположение и назначение выводов МС68НС705С8. 51

Приложение 2.

Типовая схема подключения МС68НС705С8. 54

## 1. Программная модель ОЭВМ МС68НС705С8.

### 1.1 Внутренняя структура.

ОЭВМ имеет ряд отличительных особенностей в аппаратном и программном обеспечении, которые перечислены ниже:

***Особенности аппаратного обеспечения:***

 КМОП-структура с высокой плотностью компоновки;

 8-разрядная архитектура;

 режимы малого потребления энергии (Ожидание, Останов, и режим сохранения данных);

 24 двунаправленные линии ввода-вывода.

 7 однонаправленных линий (только ввод).

 2 линии ввода-вывода таймера.

 тактовая частота при питании 5В - до 2.1 МГц, при питании 3В - до1 Мгц;

 встроенный 16-битовый таймер;

 последовательный интерфейс связи;

 последовательный периферийный интерфейс;

 программируемое ПЗУ с ультрафиолетовым стиранием, либо однократно программируемое ПЗУ;

 возможность конфигурирования памяти;

 система слежения за временем выполнения программы;

 диспетчер тактовой частоты;

 записанная в ПЗУ программа самозагрузки для программирования;

 программируемый режим обнаружения внешних прерываний;

 возможность прерываний от внешнего устройства, таймера и последовательного интерфейса;

 начальная установка по включению и по внешнему сигналу;

 одно питающее напряжение 3-6 В (режим сохранения данных при 2 В);

 встроенный генератор тактовой частоты;

 40-выводной корпус с двумя рядами выводов либо 44-выводной квадратный пластиковый корпус.

 ***Особенности программного обеспечения***:

 совместимость снизу вверх в программном обеспечении с M146805 КМОП серией;

 эффективный набор команд;

 гибкость в управлении с помощью прерываний;

 режимы индексной адресации для организации массивов и таблиц;

 обмен с внешними устройствами, включенными в адресное пространство;

 два режима низкого потребления энергии.

Внутренняя структура ОЭВМ представлена на рисунке 1-1. Блок центрального процессора содержит 8-битовое арифметико-логическое устройство (АЛУ), аккумулятор, индексный регистр, регистр признаков, указатель стека, программный счетчик и управляющую логику. На кристалле также расположены системы памяти: загрузочное ПЗУ, программируемое ПЗУ и ОЗУ.

В состав систем ввода-вывода входят асинхронный последовательный интерфейс связи (SCI), последовательный периферийный интерфейс (SPI) и 16-битовый программируемый таймер. Диспетчер синхронизации и таймер времени выполнения программы следят, соответственно, за наличием сигнала синхронизации на внутренней шине (в противном случае происходит RESET), а также за временем выполнения прикладной программы, которая должна периодически сбрасывать таймер, в противном случае система считает, что произошел сбой программы и формирует RESET. Тактовый генератор может функционировать как с использованием внешнего сигнала синхронизации, так и с помощью кварцевого резонатора. Схемы подключения для обоих случаев приведены в Приложении 2.

Рис. 1-1. Внутренняя структура ОЭВМ.

24 линии ввода-вывода объединены в порты А, В и С, которые могут быть запрограммированы либо на ввод, либо на вывод информации. 7 линий порта D работают только в режиме ввода информации, но могут программно переключаться на работу в режиме обслуживания последовательного интерфейса связи (Serial Communication Interface, далее SCI), либо в режиме последовательного периферийного интерфейса (Serial Periferal Interface, далее SPI).

Выводы TCAP и TCMP позволяют с помощью таймера реализовывать функции фиксации времени внешнего события (например, для измерения длительности внешнего импульса), и выполнения какой-либо операции в определенный момент времени (например, для формирования выходного сигнала в реальном времени).

Вывод Vpp используется для программирования ППЗУ, и при нормальной работе соединяется с выводом питания Vdd. Рассмотрим подробнее каждый из элементов ОЭВМ.

### 1.2 Память.

Память ОЭВМ MC68HC705C8 включает в себя от 176 до 304 байт ОЗУ, 240 байт ПЗУ и от 7600 до 7744 байт программируемого ПЗУ. МС68HC705C8 имеет 4 различных типа конфигурации памяти, как показано на рис. 1-2.

Рис. 1-2 Карта памяти ОЭВМ

Конфигурация задается записью в биты RAM0 и RAM1 регистра конфигурации OPTION, находящегося по адресу 1FDFH. При начальной установке эти биты сбрасываются. Состояние битов RAM0 и RAM1 определяет объем ОЗУ и ПЗУ, который может быть выбран из следующей таблицы:

|  |  |  |  |
| --- | --- | --- | --- |
| RAM0 | RAM1 | ОЗУ байт | ППЗУ байт |
| 0 | 0 | 176 | 7744 |
| 1 | 0 | 208 | 7696 |
| 0 | 1 | 272 | 7648 |
| 1 | 1 | 304 | 7600 |

Таким образом, конфигурация памяти может быть изменена в любой момент в ходе выполнения программы.

### 1.3 Регистры.

Как видно из структурной схемы, ОЭВМ содержит следующие регистры: 5 регистров ЦП, регистр конфигурации, программный регистр (регистр команд программирования). Назначение и структуру этих регистров мы рассмотрим в этом разделе. Кроме того, в состав ОЭВМ входят регистры портов, регистры таймера, последовательных интерфейсов и другие, все эти регистры будут рассмотрены в соответствующих разделах.

***Регистр конфигурации*** ( OPTION ), имеющий адрес 1FDFH во встроенной памяти, используется для задания режима обнаружения внешних прерываний (IRQ), разрешения/запрета доступа к ППЗУ, а также конфигурации памяти.

 Регистр конфигурации

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Назначение | RAM0 | RAM1 | 0 | 0 | SEC | - | IRQ/ | 0 |
| Установка после RESET | 0 | 0 | 0 | 0 | \*  | - | 1 | 0 |

\*бит SEC является ячейкой ППЗУ.

Ниже приводятся функции каждого бита регистра конфигурации.

RAM0-бит конфигурации памяти.

 при RAM0=1 32 байта с адреса 0030H становятся равными 0. Адреса с 0020 по 002F резервируются. Таким образом, удаляется 48 байт ППЗУ, которые были расположены по этим адресам. Этот бит может быть записан и прочитан в любой момент, таким образом, конфигурация памяти может быть изменена во время выполнения программы.

 при RAM0=0 добавляется 48 байт ППЗУ по адресам 0020H-004FH.

RAM1-бит конфигурации памяти.

 при RAM1=1 96 байт, начиная с адреса 0100H, становятся равными FFH. Бит может быть записан и прочитан в любой момент.

 при RAM1=0-добавляется 96 байт ППЗУ c адреса 0100H.

SEC-бит блокировки загрузчика. Он показывает, откуда будет производиться начальная загрузка ОЭВМ: из ППЗУ либо из внешнего источника по последовательному интерфейсу.

 при SEC=1 загрузчик блокирован. ОЭВМ работает в однокристальном режиме.

 при SEC=0 загрузчик разблокирован, управление после RESET передается встроенной программе, осуществляющей начальную загрузку встроенной памяти из внешнего устройства по последовательному интерфейсу. Бит SEC физически расположен в ППЗУ и не изменяется при общем сбросе.

IRQ -режим обнаружения запросов на прерывание от внешних устройств.

 при IRQ=1 вывод IRQ/ микропроцессора реагирует как на низкий уровень сигнала на этом выводе, так и на перепад из высокого уровня в низкий.

 при IRQ=0 ОЭВМ реагирует только на перепад из высокого в низкий уровень на входе IRQ.

***Регистр команд программирования*** (PROG), расположенный по адресу 001СН, используется для установки режима при программировании ППЗУ.

 Регистр команд программирования

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Назначение | 0 | 0 | 0 | 0 | 0 | LAT | 0 | PGM |
| Установка после RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

LAT-бит определяет доступ к триггеру программирования ППЗУ.

 при LAT=1-разрешен доступ к данным ППЗУ и адресной шине для программирования на следующем цикле записи байта.

 при LAT=0-доступ запрещен.

Бит LAT может быть записан и прочитан.

PGM-разрешение/запрет программирования.

 при PGM=1 напряжение программирования подается на ППЗУ.

 при PGM=0 напряжение программирования отключено.

Если бит LAT сброшен, PGM не может быть установлен в 1.

***Центральный процессор*** содержит 5 регистров, показанных на рис. 1-3. Регистры центрального процессора не являются частью карты памяти.

Рис. 3 Регистры ЦП.

***Аккумулятор***(A)-это 8-битный регистр общего назначения, в котором хранятся операнды, результаты арифметических операций, а также данные, с которыми производятся какие-либо операции. Аккумулятор также используется и для логических операций.

***Индексный регистр*** (X)-используется либо при индексном режиме адресации, либо как вспомогательный аккумулятор. Этот регистр может быть загружен как непосредственно, так и из памяти, может быть сохранен в ячейке памяти или сравнен с ее содержимым.

***Программный счетчик*** (PC)-это 13-битовый регистр, содержащий адрес команды, следующей за выполняемой, либо адрес операнда, входящего в код программы. Шина адреса MC68HC705C8 имеет разрядность 13 бит, что позволяет адресовать 8 Кб памяти.

***Указатель стека*** (SP), как и программный счетчик, является 13-битным регистром, содержащим адрес следующей (свободной) ячейки стека. Во время начальной установки, либо при выполнении команды "Сброс указателя стека" (RSP), в указатель записывается адрес 00FFH. Затем это значение уменьшается по мере записи данных в стек. При обращении к памяти 7 старших бит указателя постоянно имеют значение 0000011В. Эти 7 бит добавляются к шести младшим битам для получения адреса от 00FFH до 00C0H. Подпрограммы и прерывания могут, таким образом, использовать до 64 ячеек памяти под стек. Если все ячейки заполнены, то указатель стека автоматически устанавливается в исходную позицию и записанная в стек информация теряется. Вызов подпрограммы использует 2 ячейки стека, прерывание-5 ячеек.

***Регистр признаков*** (СС) содержит 5 флагов, устанавливающихся в зависимости от результата выполнения арифметических и других команд. Этими флагами являются: флаг полупереноса (Н), флаг отрицательного результата (N), флаг нулевого результата (Z), бит маски прерываний (I) и флаг переноса (С).

Флаг полупереноса(Н)-используется для операций с числами в двоично-десятичном коде и изменяется при операциях сложения (ADD) и сложения с флагом переноса (ADC). Бит Н устанавливается в 1 при возникновении переноса из бита 3 в бит 4.

Бит маскирования прерываний (I)-запрещает все маскируемые прерывания при установке его в 1. Бит I автоматически устанавливается при возникновении любого прерывания сразу же после сохранения регистров в стеке, но до передачи управления по вектору прерывания. Если внешнее прерывание происходит в тот момент, когда ,бит I установлен в 1, то запрос на прерывание фиксируется и обрабатывается после сброса бита I. После обслуживания прерывания, команда возврата из прерывания (RTI) восстанавливает содержимое регистров. Таким образом, после обработки прерывания I-бит сбрасывается, а при начальной установке ОЭВМ бит I устанавливается в 1 и может быть сброшен программно.

Флаг отрицательного результата (N) устанавливается в 1, если результат последней операции является отрицательным числом, и сбрасывается в противоположном случае.

Флаг нулевого результата (Z) устанавливается в 1, если результатом последней операции является 0 и сбрасывается в противоположном случае.

Флаг Перенос (заем) (С) индицирует, что произошел перенос в результате операции сложения, либо заем при вычитании. Операции сдвига и циклического сдвига происходят через этот бит, что дает возможность осуществлять операции сдвига с операндами из нескольких слов.

### 1.4. Режимы адресации.

ОЭВМ МС68НС705С8 использует восемь режимов адресации, определяющих способ доступа к данным, требуемым для выполнения команд. Способы адресации, поддерживаемые ОЭВМ МС68НС05С8, перечислены ниже:

 неявная;

 непосредственная;

 прямая;

 расширенная;

 индексная без смещения;

 индексная с 8-разрядным смещением;

 индексная с 16-разрядным смещением;

 относительная.

Команды с ***неявной*** адресацией либо не имеют операндов, либо содержат указание на операнд в мнемонике команды. К таким командам относятся, например, команды возврата из прерывания (RTI), останова (STOP) и т.д. Неявный тип адресации имеют команды работы с данными в регистрах микропроцессора, например, установить флаг переноса (SEC), увеличение числа, хранящегося в аккумуляторе, на единицу (INCA), и другие. Команды с неявной адресацией не требуют обращения к памяти и имеют длину один байт. В Таблице 1-1 приведен список команд, которые используют неявную адресацию.

**Таблица 1-1**. Команды, использующие неявную адресацию.

|  |  |
| --- | --- |
| Команда | Мнемоника |
| Арифметический сдвиг влево | ASLA, ASLX |
| Арифметический сдвиг вправо | ASRA, ASRX |
| Сбросить флаг переноса | CLC |
| Сбросить флаг маскирования прерываний | CLI |
| Обнулить | CLRA, CLRX |
| Дополнение | COMA, COMX |
| Декремент на 1 | DECA, DECX |
| Инкремент на 1 | INCA, INCX |
| Логический сдвиг влево | LSLA, LSLX |
| Логический сдвиг вправо | LSRA, LSRX |
| Умножение | MUL |
| Изменить знак | NEGA, NEGX |
| Нет операции | NOP |
| Циклический сдвиг влево через флаг переноса | ROLA, ROLX |
| Циклический сдвиг вправо через флаг переноса | RORA, RORX |
| Инициализация указателя стека | RSP |
| Возврат из прерывания | RTI |
| Возврат из подпрограммы | RTS |
| Установить флаг переноса | SEC |
| Установить бит маскирования прерываний | SEI |
| Разрешить IRQ\ и останов генератора синхроимпульсов | STOP |
| Программное прерывание | SWI |
| Запись содержимого аккумулятора в индексный регистр | TAX |
| Проверить на отрицательность и равенство нулю | TSTA, TSTX |
| Запись содержимого индексного регистра в аккумулятор | TXA |
| Разрешить прерывание и останов процессора | WAIT |

Пример:

0200 4С INCA Инкремент аккумулятора

Производимые действия:

$0200 $4C [1], [2], [3]

Объяснение:

[1] процессор читает код операции $4C - инкремент аккумулятора;

[2], [3] процессор считывает содержимое аккумулятора, увеличивает на 1, запоминает новую величину в аккумуляторе, в соответствии с результатом устанавливает биты в регистре флагов.

При ***непосредственной*** адресации операнд первый операнд находится в аккумуляторе или индексном регистре, а второй является частью кода команды. Команды с непосредственной адресацией не требуют адреса памяти и имеют длину два байта. Код команды содержится в первом байте, а операнд во втором байте. Таблица 1-2 содержит команды, использующие непосредственную адресацию.

**Таблица 1-2.** Команды, использующие непосредственную адресацию.

|  |  |
| --- | --- |
| Команда | Мнемоника |
| Сложение с флагом переноса | ADC |
| Сложение | ADD |
| Логическое И | AND |
| Битовое сравнение ячейки памяти с аккумулятором | BIT |
| Сравнение аккумулятора и ячейки памяти | CMP |
| Сравнение индексного регистра и ячейки памяти | CPX |
| Исключающее ИЛИ ячейки памяти и аккумулятора | EOR |
| Загрузить аккумулятор содержимым ячейки памяти | LDA |
| Загрузить индексный регистр содержимым ячейки памяти | LDX |
| Операция ИЛИ | ORA |
| Вычитание с флагом переноса | SBC |
| Вычитание | SUB |

Пример:

0200 А6 02 LDA #$02 Загрузить аккумулятор константой $02

Производимые действия:

$0200 $A6 [1]

$0201 $02 [2]

Объяснение:

[1] ЦП читает код операции $A6 - загрузить аккумулятор константой, следующей за кодом операции

[2] ЦП читает константу $02 по адресу $0201 и записывает ее в аккумулятор.

Команды с ***прямой адресацией*** имеют доступ к первым 256 ячейкам памяти и имеют длину два байта. Первым байтом является код команды, второй байт представляет собой младший байт адреса операнда. При использовании прямой адресации ЦП автоматически устанавливает в $00 старший байт адреса операнда. Исключение составляют команды BRSET и BRCLR, занимающие три байта и использующие прямую адресацию для доступа к операнду и относительную адресацию для определения точки перехода. В Таблице 1-3 представлены команды, которые используют режим прямой адресации к памяти.

**Таблица 1-3.** Команды, использующие прямую адресацию к памяти.

|  |  |
| --- | --- |
| Команда | Мнемоника |
| Сложение с флагом переноса | ADC |
| Сложение | ADD |
| Логическое И | AND |
| Арифметический сдвиг влево | ASL |
| Арифметический сдвиг вправо | ASR |
| Сбросить бит в ячейке памяти | BCLR |
| Битовое сравнение ячейки памяти и аккумулятора | BIT |
| Переход, если бит n установлен | BRCLR |
| Переход, если бит n сброшен | BRSET |
| Сбросить бит в памяти | BCLR |
| Установить бит в памяти | BSET |
| Очистить | CLR |
| Сравнение аккумулятора и ячейки памяти | CMP |
| Дополнение | COM |
| Сравнение индексного регистра и ячейки памяти | CPX |
| Декремент на 1 | DEC |
| Исключающее ИЛИ ячейки памяти и аккумулятора | EOR |
| Инкремент на 1 | INC |
| Безусловный переход | JMP |
| Вызов подпрограммы | JSR |
| Загрузить аккумулятор содержимым ячейки памяти | LDA |
| Загрузить индексный регистр содержимым ячейки памяти | LDX |
| Логический сдвиг влево | LSL |
| Логический сдвиг вправо | LSR |
| Изменение знака | NEG |
| ИЛИ | ORA |
| Циклический сдвиг влево через флаг переноса | ROL |
| Циклический сдвиг вправо через флаг переноса | ROR |
| Вычитание с флагом переноса | SBC |
| Записать содержимое аккумулятора в ячейку памяти | STA |
| Записать содержимое индексного регистра в ячейку памяти | STX |
| Вычитание | SUB |
| Проверка знака и равенства нулю | TST |

Пример:

0200 В6 50 LDA $50 Загрузить аккумулятор содержимым ячейки с адресом $50

Производимые действия:

$0200 $B6 [1]

$0201 $50 [2] и [3]

Объяснение:

[1] процессор читает код операции $В6 - загрузить аккумулятор, используя прямую адресацию.

[2] процессор считывает $50 по адресу $0201. $50 интерпретируется как младший байт 16-ти разрядного адреса.

[3] процессор считывает содержимое ячейки $0050 и помещает его в аккумулятор.

Команды, использующие ***расширенную*** адресацию, имеют доступ к любому адресу памяти и представлены в табл. 1-4. Длина таких команд составляет три байта, первый из которых является кодом операции, второй и третий байт - соответственно старшим и младшим байтом адреса операнда.

**Таблица 1-4.** Команды, использующие расширенный способ адресации.

|  |  |
| --- | --- |
| Команда | Мнемоника |
| Сложение с флагом переноса | ADC |
| Сложение | ADD |
| Логическое И | AND |
| Битовое сравнение ячейки памяти и аккумулятора | BIT |
| Сравнение аккумулятора с ячейкой памяти | CMP |
| Сравнение индексного регистра с ячейкой памяти | CPX |
| Исключающее ИЛИ ячейки памяти и аккумулятора | EOR |
| Безусловный переход | JMP |
| Вызов подпрограммы | JSR |
| Загрузка аккумулятора содержимым ячейки памяти | LDA |
| Загрузка индексного регистра содержимым ячейки памяти | LDX |
| ИЛИ | ORA |
| Вычитание с флагом переноса | SBC |
| Записать содержимое аккумулятора в память | STA |
| Записать содержимое индексного регистра в память | STX |
| Вычитание | SUB |

Пример:

0200 С6 06 Е5 LDA $05E5 Загрузить аккумулятор содержимым ячейки с адресом $05Е5

Производимые действия:

$0200 $C6 [1]

$0201 $05 [2]

$0202 $E5 [3]

Объяснение:

[1] процессор читает код операции $C6 - загрузить аккумулятор, используя расширенную адресацию.

[2] процессор считывает $06 по адресу $0201. $06 интерпретируется как старший байт 16-ти разрядного адреса.

[3] процессор считывает $E5 по адресу $0202. $E5 интерпретируется как младший байт 16-ти разрядного адреса

[4] процессор считывает содержимое ячейки $06E5 и помещает его в аккумулятор.

***Индексная адресация без смещения*** часто используется для перемещения по таблице или хранения адреса обращения к RAM или регистру ввода/вывода. В таблице 1-5 приведен список команд, которые используют индексную адресацию без смещения. Команды, использующие ***индексную адресацию без смещения***, имеют длину один байт и имеют доступ к переменным в пределах первых 256 ячеек памяти. Индексный регистр в этом случае содержит младший байт адреса операнда. ЦП автоматически устанавливает в $00 старший байт адреса операнда, поэтому эти команды имеют доступ к адресам от $0000 до $00FF.

**Таблица1-5. Команды, использующие индексную адресацию**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Команда | Мнемоника | Без смещ | 8 бит смещ | 16 бит смещ |
| Сложение с флагом переноса | ADC |  |  |  |
| Сложение | ADD |  |  |  |
| Логическое И | AND |  |  |  |
| Арифметический сдвиг влево | ASL |  |  |  |
| Арифметический сдвиг вправо | ASR |  |  |  |
| Битовое сравнение ячейки памяти и аккумулятора | BIT |  |  |  |
| Очистить | CLR |  |  |  |
| Сравнение аккумулятора с ячейкой памяти | CMP |  |  |  |
| Дополнение | COM |  |  |  |
| Сравнение индексного регистра с ячейкой памяти | CPX |  |  |  |
| Декремент на 1 | DEC |  |  |  |
| Исключающее ИЛИ ячейки памяти и аккумулятора | EOR |  |  |  |
| Инкремент на 1 | INC |  |  |  |
| Безусловный переход | JMP |  |  |  |
| Вызов подпрограммы | JSR |  |  |  |
| Загрузка аккумулятора содержимым ячейки памяти | LDA |  |  |  |
| Загрузка индексного регистра содержимым ячейки памяти | LDX |  |  |  |
| Логический сдвиг влево | LSL |  |  |  |
| Логический сдвиг вправо | LSR |  |  |  |
| Изменение знака | NEG |  |  |  |
| ИЛИ | ORA |  |  |  |
| Циклический сдвиг влево через флаг переноса | ROL |  |  |  |
| Циклический сдвиг вправо через флаг переноса | ROR |  |  |  |
| Вычитание с флагом переноса | SBC |  |  |  |
| Запись содержимого аккумулятора в ячейку памяти | STA |  |  |  |
| Запись содержимого индексного регистра в ячейку памяти | STX |  |  |  |
| Вычитание | SUB |  |  |  |
| Проверка знака и равенства нулю | TST |  |  |  |

Команды с ***индексной адресацией с 8-разрядным смещением*** имеют длину два байта, что позволяет обращаться к операндам в пределах первых 511 байт. Процессор складывает беззнаковый байт в индексном регистре с беззнаковым байтом, следующим за кодом операции. Полученная сумма является адресом операнда. Эта команда позволяет адресовать операнд в пределах от $0000 до $01FE. Индексную адресацию с 8-разрядным смещением удобно применять для выбора k-го элемента из n элементов таблицы. Таблица может начинаться с любого адреса в пределах первых 256 байт и занимать адреса до $01FE. Число k обычно записывается в индексный регистр. Адресом начала таблицы является байт, следующий за кодом операции. В Таблице 1-5 приведен список команд, использующих индексную адресацию с 8-разрядным смещением.

***Индексную адресацию с 16-разрядным смещением*** удобно использовать для выбора k-го элемента в таблице из n элементов в любой точке адресного пространства. Как и в случае прямой и расширенной адресации, ассемблер фирмы Motorola определяет автоматически наиболее короткую форму индексной адресации. В Таблице 1-5 приведен список команд, которые используют индексную адресацию с 16-разрядным смещением.

***Относительная адресация*** используется только командами перехода и битового сравнения. ЦП вычисляет адрес перехода посредством сложения байта, следующего за кодом операции, с содержимым программного счетчика, если выполняется условие перехода. Если условие перехода не выполняется, процессор переходит на выполнение следующей команды. Чтобы определить направление перехода, смещение является знаковой величиной, дополнение байта до двух дает переход в пределах от -127 до 128 байт от адреса команды, следующей за командой перехода. При использовании ассемблера фирмы Motorola программисту не требуется вычислять смещение, потому что ассемблер вычисляет требуемое смещение и проверяет длину перехода на допустимость. В Таблице 1-6 перечислены команды, которые используют относительную адресацию.

**Таблица 1-6.** Команды, использующие относительную адресацию.

|  |  |
| --- | --- |
| Команда | Мнемоника |
| Перейти, если сброшен флаг переноса | BCC |
| Перейти, если флаг переноса установлен | BCS |
| Перейти, если операнды равны | BEQ |
| Перейти, если сброшен перенос из младшей тетрады | BHCC |
| Перейти, если установлен флаг переноса из младшей тетрады | BHCS |
| Перейти, если больше | BHI |
| Перейти, если больше или равно | BHS |
| Перейти, если активизирован вход запроса прерывания | BIH |
| Перейти, если не активизирован вход запроса прерывания | BIL |
| Перейти, если меньше | BLO |
| Перейти, если меньше или равно | BLS |
| Перейти, если бит маскирования прерывания сброшен | BMC |
| Перейти, если результат отрицательный | BMI |
| Перейти, если бит маскирования прерывания установлен | BMS |
| Перейти, если не равно | BNE |
| Перейти, если результат положительный | BPL |
| Безусловный переход | BRA |
| Перейти, если сброшен бит n | BRCLR |
| Перейти, если установлен бит n | BRSET |
| Нет перехода | BRN |
| Вызов подпрограммы | BSR |

### 1.5. Набор команд.

Команды процессора можно разделить на следующие группы:

 команды пересылки данных

 команды передачи управления

 арифметические и логические команды, битовые операции.

Рассмотрим работу команд каждой группы подробнее. При этом будем использовать следующие условные знаки;

 Операнды:

 ()-содержимое регистра или ячейки памяти

 -загружается из

 -извлекается из стека

 -загружается в стек

 \* -логическое умножение (И)

 +-арифметическое сложение (кроме тех случаев, где используется как ИЛИ в логических операциях)

 х-умножение

 \-отрицание

 Регистры:

 AC -аккумулятор

 CCR -регистр признаков

 Х -индексный регистр

 РС -программный счетчик

 РСН -старший байт программного счетчика

 PCL -младший байт программного счетчика

 SP -указатель стека

 Память и адресация:

 М -ячейка памяти либо ее содержимое в зависимости от режима адресации

 Rel -относительное смещение

 Биты регистра признаков:

 H, I, N, C, Z-в соответствии с главой 1.3

 Состояние определенных бит до выполнения операции (n=7, 6, ... , 0):

 An -бит n аккумулятора

 Xn -бит n индексного регистра

 Mn-бит n ячейки памяти

 Состояние определенных бит после выполнения операции:

 Rn-бит n результата

 При описании изменения состояния флагов в регистре признаков после буквенного обозначения флага будет следовать формула, определяющая состояние регистра после данной команды. Подразумевается, что если флаг не упоминается, то его содержимое не изменяется.

**Команды пересылки данных.**

***LDA*** *Загрузить аккумулятор содержимым ячейки памяти***.**

Команда пересылает содержимое ячейки памяти в аккумулятор. В соответствии с записываемыми в аккумулятор данными устанавливаются биты в регистре признаков.

N R7 .

Устанавливается, если установлен старший бит аккумулятора, в противном случае сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

Устанавливается, если в аккумуляторе $00, иначе сбрасывается.

***LDX*** *Загрузить индексный регистр содержимым ячейки памяти***.**

Команда пересылает содержимое ячейки памяти в индексный регистр. В соответствии с записываемыми данными устанавливаются разряды регистра признаков.

N R7

Устанавливается, если установлен старший бит индексного регистра, в противном случае сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

Устанавливается, если в индексный регистр записано число $00, иначе сбрасывается.

***STA*** *Записать содержимое аккумулятора в ячейку памяти.*

Переместить содержимое аккумулятора в ячейку памяти. Содержимое аккумулятора остается неизменным.

N R7

Устанавливается, если установлен старший бит аккумулятора, в противном случае сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

Устанавливается, если в аккумулятор записано число $00, иначе сбрасывается.

***STX*** *Записать содержимое индексного регистра в ячейку памяти.*

Переместить содержимое индексного регистра в ячейку памяти. Содержимое индексного регистра не изменяется.

N R7

Устанавливается, если установлен старший бит в пересылаемом байте, в противном случае сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

Устанавливается, если пересылаемое число $00, иначе сбрасывается.

***TAX*** *Записать содержимое аккумулятора в индексный регистр*

X(AC)

Записать в индексный регистр содержимое аккумулятора. Содержимое аккумулятора не изменяется.

Регистр признаков не изменяется.

***TXA*** *Записать содержимое индексного регистра в аккумулятор*

AC(X)

Записать в аккумулятор содержимое индексного регистра. Содержимое индексного регистра не изменяется.

Регистр признаков не изменяется.

***CLR*** *Обнулить.*

AC $00 или M $00 или X $00

В аккумулятор, индексный регистр или ячейку памяти записывается ноль.

N 0

Z = 1

***RSP*** *Инициализация указателя стека.*

SP$00FF

Установить указатель стека на вершину стека.

Регистр признаков не изменяется.

**Команды передачи управления.**

Команды передачи управления включают в себя команды безусловного перехода, команды вызова подпрограммы и возврата из нее, а также команды условных переходов (ветвления) по различным условиям (состояниям флагов) и по состоянию битов. Команды ветвления вызывают переход по состоянию любого бита из первых 256 ячеек памяти. Эти команды имеют длину три байта и являются комбинацией прямой и относительной адресации. Прямо адресуется бит для тестирования, адрес которого содержится во втором байте команды. Третий байт представляет собой знаковое смещение для перехода по состоянию бита. Процессор вычисляет адрес перехода, складывая содержимое программного счетчика и третьего байта команды, если условие перехода выполняется. Условие тестирования битов содержится в коде операции. Диапазон перехода - от -128 до +127 байт от адреса следующей за командой ячейки памяти. Процессор, кроме того, записывает тестируемый бит в бит переноса регистра условий (СС).

***BRA*** *Безусловный переход.*

Команда осуществляет безусловный переход по адресу, вычисляемому по приведенной ниже формуле, где Rel - относительное смещение, которое содержится в последнем байте кода команды.

PC (PC)+$0002+Rel.

В исходном тексте адрес перехода задается абсолютным значением, символом или выражением, которое может быть вычислено ассемблером (заметим, что правила записи меток или выражений для вычисления адреса перехода могут различаться при использовании разных ассемблеров). Ассемблер вычисляет смещение Rel из абсолютного адреса перехода и текущего значения программного счетчика PC.

Регистр признаков не изменяется.

***BRN*** *Нет перехода.*

Команда используется как двухбайтовый вариант команды NOP (нет операции) при отладке программ, когда нужно отменить действие команды перехода, и выполняется за 3 цикла. Действие этой команды противоположно действию команды BRA.

Регистр признаков не изменяется.

***BRCLR*** *Переход, если бит n ячейки памяти равен 0.*

PC (PC)+$0003+Rel Если бит n ячейки M равен нулю.

Команда проверяет бит n ячейки памяти M на равенство нулю. Переход осуществляется, если проверяемый бит равен нулю. M может быть в адресном пространстве RAM или регистров ввода/вывода в пределах от $0000 до $00FF (таким образом используется прямая адресация). Бит C регистра признаков устанавливается в состояние тестируемого бита.

C Устанавливается, если Mn=1, иначе сбрасывается.

***BRSET*** *Переход, если бит n ячейки памяти равен 1*.

PC (PC)+$0003+Rel Если бит n ячейки M равен единице.

Команда проверяет бит n ячейки памяти M на равенство единице. Переход осуществляется, если проверяемый бит равен единице. M может быть в адресном пространстве RAM или регистров ввода/вывода в пределах от $0000 до $00FF (таким обрезом используется прямая адресация). Бит C регистра признаков устанавливается в состояние тестируемого бита.

C Устанавливается, если Mn=1, иначе сбрасывается.

***BHI*** *Переход, если больше*.

PC(PC)+$0002+Rel Если (C)+(Z)=0

т.е. если (AC)>(M) (беззнаковые двоичные числа)

Возникает переход, если оба бита C и Z регистра признаков сброшены в ноль. Когда команда BHI используется сразу после выполнения команд CMP или SUB, возникает переход, если беззнаковое число, содержащееся в аккумуляторе, больше, чем беззнаковое число в ячейке памяти M.

Регистр признаков не изменяется.

***BLS*** *Переход , если меньше или равно*.

PC(PC)+$0002+Rel Если [(C)+(Z)]=1

т.е. если (AC)(M) (беззнаковые двоичные числа)

Возникает переход, если бит C или Z регистра признаков установлен в единицу. Когда команда BLS используется сразу после выполнения команд CMP или SUB, возникает переход, если беззнаковое число, содержащееся в аккумуляторе, меньше или равно беззнакового числа в ячейке памяти M.

Регистр признаков не изменяется.

***BCC*** Переход, если сброшен флаг переноса.

PC(PC)+$0002+Rel Если (C)=0

Команда BCC является полным аналогом команды BHS. Проверяется состояние бита C регистра признаков. Переход возникает, если бит С сброшен.

Регистр признаков не изменяется.

***BHS*** *Переход, если больше или равно*.

Команда BHS является полным аналогом команды BCC. Мнемонику BCC принято использовать после команд CMP и SUB, так как переход в данном случае возникает, если беззнаковое число в аккумуляторе больше или равно беззнакового числа, содержащегося в ячейке памяти M.

***BCS*** *Переход, если установлен флаг переноса*.

PC(PC)+$0002+Rel Если (C)=-1

Команда BCS является полным аналогом команды BLO. Тестируется состояние бита C регистра признаков. Если бит C установлен в единицу, возникает переход.

Регистр признаков не изменяется.

***BLO*** *Переход, если меньше*.

PC(PC)+@0002+Rel Если (C)=1

т.е. если (ACCX)<(M) (беззнаковые двоичные числа)

Команда BLO является полным аналогом команды BCS. Если команда BLO выполняется сразу после выполнения команд CMP или SUB, переход возникает, если беззнаковое число, содержащееся в аккумуляторе, меньше беззнакового числа, содержащегося в ячейке памяти M.

Регистр условий не изменяется.

***BNE*** *Переход, если не равно*.

PC(PC)+$0002+Rel Если (Z)=0

Тестируется состояние бита Z регистра условий. Возникает переход, если бит Z сброшен. Следуя после сравнения или вычитания, BNE вызовет переход, если аргументы не равны.

Регистр условий не изменяется.

***BEQ*** *Переход, если равно*.

PC(PC)+$0002+Rel Если (Z)=1

Тестируется состояние бита Z регистра условий. Возникает переход, если бит Z установлен. Следуя после сравнения или вычитания, BEQ вызовет переход, если аргументы равны.

Регистр условий не изменяется.

***BHCC*** *Переход, если сброшен флаг переноса из младшей тетрады.*

PC(PC)+$0002+Rel Если (H)=0

Тестируется состояние бита H регистра условий. Возникает переход, если бит H сброшен. Команда используется при работе с числами в BCD формате.

Регистр условий не изменяется.

***BHCS*** *Переход, если установлен флаг переноса из младшей тетрады*.

PC(PC)+$0002+Rel Если (H)=1

Тестируется состояние бита H регистра условий. Возникает переход, если бит H установлен. Команды используется при работе с числами в BCD формате.

Регистр условий не изменяется.

***BPL*** *Переход, если плюс*.

PC(PC)+$0002+Rel Если (N)=0

Тестируется состояние бита N регистра условий. Возникает переход, если бит N сброшен.

Регистр условий не изменяется.

***BMI*** *Переход, если минус*.

PC(PC)+$0002+Rel Если (N)=1

Тестируется состояние бита N регистра условий. Возникает переход, если бит N установлен.

Регистр условий не изменяется.

***BMC*** *Переход, если бит маскирования прерываний сброшен*.

PC(PC)+$0002+Rel Если (I)=0

Тестируется состояние бита I регистра условий. Возникает переход, если бит I сброшен (то есть прерывания разрешения).

Регистр условий не изменяется.

***BMS*** *Переход, если бит маскирования прерываний установлен*.

PC(PC)+$0002+Rel Если (I)=1

Тестируется состояние бита I регистра условий. Возникает переход, если бит I установлен (то есть прерывания запрещены).

Регистр условий не изменяется.

***BIL*** *Переход, если вход запроса прерывания в 0*.

PC(PC)+$0002+Rel Если IRQ\=0

Тестируется состояние выводы микропроцессора запроса внешнего прерывания. Возникает переход, если на входе нулевой логический уровень.

Регистр условий не изменяется.

***BIH*** *Переход, если вход запроса прерывания в 1*.

PC(PC)+$0002+Rel Если IRQ\=1

Тестируется состояние выводы микропроцессора запроса внешнего прерывания. Возникает переход, если на входе логическая единица.

Регистр условий не изменяется.

***BSR*** *Переход на подпрограмму*.

PC(PC)+$0002 В PC адрес возврата

(PCL) Сохраняется младший байт адреса возврата в стек

SP(SP)-$0001

(PCH) Сохраняется старший байт адреса возврата в стек

SP(SP)-$0001

PC(PC)+Rel В PC загружается адрес подпрограммы

Программный счетчик увеличивается на 2 от адреса кода операции BSR, таким образом в PC оказывается адрес следующей за BSR команды, который является адресом возврата из подпрограммы. Младший байт программного счетчика помещается в стек. Указатель стека уменьшается на 1. Затем старший байт программного счетчика помешается в стек, указатель стека уменьшается на 1. Производится переход по адресу, определяемому смещением.

Регистр условий не изменяется.

***JMP*** *Длинный безусловный переход*.

PCЭффективный адрес

Осуществляется переход по эффективному адресу. Эффективный адрес вычисляется в соответствии с правилами для расширенного, прямого и индексного режимов адресации.

Регистр условий не изменяется.

***JSR*** *Длинный вызов подпрограммы*.

PC(PC)+n n=1,2,3 взависимости от режима адресации

(PCL); SP SP-$0001 Младший байт адреса возврата в стек

(PCH); SP SP-$0001 Старший байт адреса возврата в стек

PCЭффективный адрес Загрузить в PC адрес подпрограммы

Программный счетчик увеличивается на n, то есть до адреса следующей за JSR команды. Затем PC сохраняется в стеке. Не используемые биты программного счетчика сохраняются как единицы. Указатель стека указывает на следующую свободную ячейку памяти. Осуществляется переход по эффективному адресу. Эффективный адрес вычисляется в соответствии с правилами для расширенного, прямого и индексного режимов адресации.

Регистр условий не изменяется.

***SWI*** *Программное прерывание*

(PC)(PC)+$0001 В PC адрес возврата

(PCL); SP(SP)-$0001 В стек младший байт адреса возврата

(PCH); SP(SP)-$0001 В стек старший байт адреса возврата

(X); SP(SP)-$0001 В стек индексный регистр

(AC); SP(SP)-$0001 В стек аккумулятор

(CCR); SP(SP)-$0001 В стек регистр признаков

I бит 1

PCH($xFFC) Вектор прерывания (x=1 или 3, в зависимости от версии HC05)

PCL($xFFD)

Содержимое программного счетчика увеличивается на единицу. Программный счетчик, индексный регистр, аккумулятор и регистр признаков помещаются в стек. Содержимое указателя стека уменьшается на единицу каждый раз после того, как в стек помещен байт. Бит маскирования прерываний устанавливается в единицу. В программный счетчик записывается адрес, хранящийся в векторе прерывания SWI (адрес расположен по адресу n-0002 и n-0003, где n-адрес, соответствующий единичному состоянию всех адресных выходов миропроцессора.). Команда SWI не маскируется установкой бита I регистра признаков.

I Устанавливается в 1.

***RTI*** *Возврат из подпрограммы обработки прерывания*

SP(SP)+$0001 ССR CCR из стека

SP(SP)+$0001 AC AC из стека

SP(SP)+$0001 X X из стека

SP(SP)+$0001 PCH PCH из стека

SP(SP)+$0001 PCL PCL из стека

Регистр признаков, аккумулятор, индексный регистр и программный счетчик восстанавливаются из стека. Бит I сбрасывается, если соответствующий бит регистра признаков в стеке был нулевым.

Биты регистра признаков принимают значения в соответствии с байтом, взятым из стека.

***RTS*** Возврат из подпрограммы

SP(SP)+$0001 PCH PCH из стека

SP(SP)+$0001 PCL PCL из стека

Указатель стека увеличивается на единицу. Содержимое ячейки памяти, на которую указывает указатель стека, записывается в старший байт программного счетчика. Указатель стека еще раз увеличивается на единицу. Содержимое ячейки памяти, не которую указывает указатель стека, записывается в младший байт программного счетчика.

Регистр признаков не изменяется.

**Арифметические и логические команды, битовые операции.**

***ADD*** *Сложить содержимое ячейки памяти с аккумулятором*

AC(AC)+(M)

Сложить содержимое M и содержимое AC и разместить в AC.

H A3\*M3+M3\*R3\+R3\\*A3

 Устанавливается, если имеет место перенос из бита 3, иначе сбрасывается.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все битв результата сброшены. Иначе сбрасывается.

C A7\*M7+M7\*R7\+R7\\*A7

 Устанавливается, если имеет место перенос из старшего бита результата, иначе сбрасывается.

***ADC*** *Сложить содержимое ячейки памяти с аккумулятором
 и флагом переноса.*

AC(AC)+(M)+(C)

Сложить содержимое бита C и сумму содержимого M и содержимого AC и разместить в AC.

H A3\*M3+M3\*R3\+R3\\*A3

 Устанавливается, если имеет место перенос из бита 3, иначе сбрасывается.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все битв результата сброшены. Иначе сбрасывается.

C A7\*M7+M7\*R7\+R7\\*A7

 Устанавливается, если имеет место перенос из старшего бита результата, иначе сбрасывается.

***SUB*** *Вычесть*.

AC(AC)-(M)

Вычесть содержимое M из AC и разместить результат в AC.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все битв результата сброшены. Иначе сбрасывается.

C A7\*M7+M7\*R7\+R7\\*A7

 Устанавливается, если абсолютная величина содержимого M больше абсолютной величины аккумулятора, иначе сбрасывается.

***SBC*** *Вычесть содержимое ячейки памяти из аккумуляторе
 с флагом заема*.

AC(AC)-(M)-(C)

Вычесть содержимое M из AC и разместить результат в AC.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все битв результата сброшены. Иначе сбрасывается.

C A7\*M7+M7\*R7\+R7\\*A7

 Устанавливается, если абсолютная величина одержимого M, плюс предыдущее значение C, больше абсолютной величины аккумулятора, иначе сбрасывается.

***AND*** *Операция И содержимого ячейки памяти и аккумулятора*.

AC(AC)\*(M)

Произвести логическое умножение между содержимым аккумулятора и содержимым M и разместить результата в AC. (С каждым битом аккумулятора и соответствующим битом M производится операция логического умножения).

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все битв результата сброшены. Иначе сбрасывается.

***ORA*** *Операция ИЛИ содержимого ячейки памяти и аккумулятора*.

AC(AC)+(M)

Произвести логическое сложение между содержимым аккумулятора и содержимым M и разместить результата в AC. (С каждый бит аккумулятора и соответствующим битом M производится операция логического сложения).

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все битв результата сброшены, иначе сбрасывается.

***CMP*** *Арифметическое сравнение аккумулятора и ячейки памяти*.

(AC)-(M)

Сравнить содержимое M и AC и установить биты в регистре признаков. Содержимое AC и M не изменяется.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все битв результата сброшены. Иначе сбрасывается.

C A7\*M7+M7\*R7\+R7\\*A7

 Устанавливается, если абсолютная величина содержимого M больше абсолютной величины аккумулятора, иначе сбрасывается.

***CPX*** *Арифметическое сравнение индексного регистра и ячейки памяти*.

(X)-(M)

Сравнить содержимое M и X и установить биты в регистре признаков. Содержимое X и M не изменяется.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все битв результата сброшены. Иначе сбрасывается.

C A7\*M7+M7\*R7\+R7\\*A7

 Устанавливается, если абсолютная величина одержимого M больше абсолютной величины регистра X, иначе сбрасывается.

***BIT*** Поразрядное сравнение ячейки памяти с аккумулятором.

(ACCX)\*(M)

Производит сравнение содержимого AC и M посредством операции логическое И и устанавливает соответственно результату биты в регистре признаков. Ни содержимое AC, ни содержимое M не изменяются

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все биты результата сброшены. Иначе сбрасывается.

***MUL*** Умножение.

X : A=X x A

Умножается 8-битное число в индексном регистре X на 8-битное число в аккумуляторе. 16-битный результат помещается в индексный регистр и аккумулятор. В индексном регистре старшие 8 бит результата, в аккумуляторе младшие 8 бит.

H 0 (сбрасывается)

C 0 (сбрасывается)

***INC*** *Инкремент на 1*

AC(AC)+$01 или M(M)+$01 или X(X)+$01

Добавляет единицу к содержимому аккумулятора, индексного регистра или ячейки памяти. Биты N и Z регистра признаков устанавливаются или сбрасываются в соответствии с результатом. Бит C регистра признаков не изменяются, поэтому после команды INC имеют смысл только следующие команды перехода BEQ, BNE, BPL и BMI.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все биты результата сброшены. Иначе сбрасывается.

***DEC*** *Декремент на 1*

AC(AC)-$01 или M(M)-$01 или X(X)-$01

Вычитает единицу из содержимого аккумулятора, индексного регистра или ячейки памяти. Биты N и Z регистра признаков устанавливаются или сбрасываются в соответствии с результатом. Бит C регистра признаков не изменяются, поэтому после команды DEC имеют смысл только следующие команды перехода:

BEQ, BNE, BPL и BMI.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все биты результата сброшены. Иначе сбрасывается.

***COM*** *Дополнение*.

AC(AC\)=$FF-(AC) или M(M\)=$FF-(M) или XX\=$FF-(X)

Содержимое аккумулятора, индексного регистра или ячейки памяти заменяется на его дополнение до 1. Каждый бит заменяется на его дополнение, то есть инвертируется.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все биты результата сброшены. Иначе сбрасывается.

C 1

 Устанавливается

***NEG*** *Дополнение до двух (инверсия знака).*

AC(AC) или M(M) или XX

Содержимое аккумулятора, индексного регистра или ячейки памяти заменяется на его дополнение до 2. Заметим, что величина $80 не изменяется.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все биты результата сброшены. Иначе сбрасывается.

C R7+R6+R5+R3+R2+R1+R0

 Устанавливается во всех случаях за исключением числа $00.

***ROL*** *Циклический сдвиг влево через флаг переноса*

Cb7-...-b0C

Все биты аккумулятора, индексного регистра или ячейки памяти сдвигаются влево. Бит 0 загружается содержимым бита С. Бит C загружается содержимым старшего бита аккумулятора, индексного регистра или ячейки памяти.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все биты результата сброшены. Иначе сбрасывается.

C b7

 Устанавливается, если перед сдвигом был установлен старший бит, иначе сбрасывается.

***ROR*** *Циклический сдвиг вправо через флаг переноса*

Cb7-...-b0C

Все биты аккумулятора, индексного регистра или ячейки памяти сдвигаются вправо. Бит 7 загружается содержимым бита С. Бит C загружается содержимым младшего бита аккумулятора, индексного регистра или ячейки памяти.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все биты результата сброшены. Иначе сбрасывается.

C b0

 Устанавливается, если перед сдвигом был установлен младший бит, иначе сбрасывается.

***LSL*** *Логический сдвиг влево*

Cb7-...-b00

Все биты аккумулятора, индексного регистра или ячейки памяти сдвигаются влево. Бит 0 загружается нулем. Бит C загружается содержимым старшего бита аккумулятора, индексного регистра или ячейки памяти.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все биты результата сброшены. Иначе сбрасывается.

C b7

 Устанавливается, если перед сдвигом был установлен старший бит, иначе сбрасывается.

***LSR*** *Логический сдвиг вправо*.

0b7-...-b0C

Все биты аккумулятора, индексного регистра или ячейки памяти сдвигаются вправо. Бит 7 загружается нулем. Бит C загружается содержимым младшего бита аккумулятора, индексного регистра или ячейки памяти.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все биты результата сброшены. Иначе сбрасывается.

C b0

 Устанавливается, если перед сдвигом был установлен младший бит, иначе сбрасывается.

***ASR*** *Арифметический сдвиг вправо*

b7-...-b0C

Все биты аккумулятора, индексного регистра или ячейки памяти сдвигаются влево. Бит 7 остается неизменным. Бит C загружается содержимым младшего бита аккумулятора, индексного регистра или ячейки памяти.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все биты результата сброшены. Иначе сбрасывается.

C b0

 Устанавливается, если перед сдвигом был установлен младший бит, иначе сбрасывается.

***TST*** *Проверка на отрицательность и равенство нулю*

(AC)-$00 или (X)-$00 или (M)-$00

Установить биты N и Z в регистре признаков в соответствии с содержимым аккумулятора, индексного регистра или ячейки памяти. Содержимое аккумулятора, индексного регистра или ячейки памяти не изменяется.

N R7

 Устанавливается, если старший бит результата установлен, иначе сбрасывается.

Z R7\\*R6\\*R5\\*R4\\*R3\\*R2\\*R1\\*R0\

 Устанавливается, если все биты результата сброшены. Иначе сбрасывается.

***BSET*** n (n=0...7) *Установить бит n*

Mn=1

Устанавливается в 1 бит n (n=7,6,5,4,3,2,1,0) ячейки памяти M, другие биты не изменяются. Ячейкой памяти может быть ОЗУ или I/O регистр от адреса $0000 до $00FF (то есть используется прямая адресация).

Регистр признаков не изменяется.

***BCLR*** n (n=0...7) *Сбросить бит n*

Mn=0

Сбрасывается бит n (n=7,6,5,4,3,2,1,0) ячейки памяти M, другие биты не изменяются. Ячейкой памяти может быть ОЗУ или I/O регистр от адреса $0000 до $00FF (то есть используется прямая адресация).

Регистр признаков не изменяется.

***SEC*** *Установить флаг переноса*

C бит 1

Установить бит C регистра признаков. Команда SEC может применятся для предустановки бита C перед использованием команд сдвига.

Флаг C устанавливается в 1.

***CLC*** *Сбросить флаг переноса*

C бит 0

Сбрасывается бит C регистра признаков. Команда SEC может применятся для предустановки бита C перед использованием команд сдвига.

Флаг C сбрасывается.

***SEI*** *Установить флаг маскирования прерываний*.

I бит1

Устанавливается бит маскирования прерываний в регистре признаков. Когда бит C установлен, прерывания запрещены.

Флаг I устанавливается.

***CLI*** *Сбросить флаг маскирования прерываний*

I бит0

Сбрасывается бит маскирования прерываний в регистре признаков. Когда бит C сброшен, прерывания разрешены. На очистку бита требуется задержка на один цикл, таким образом, если прерывания до выполнения команды CLI были запрещены, то следующая за CLI команда всегда выполняется.

Флаг I сбрасывается.

**Прочие команды.**

***NOP*** *Нет операции*

Эта однобайтная команда вызывает только увеличение программного счетчика на единицу. Другие регистры не изменяются.

***STOP*** *Останов*

Выполнение команды приводит к уменьшению энергопотребления.

1) Сбрасывается предделитель таймера;

2) запрещаются прерывания от таймера;

3) сбрасывается флаг прерывания от таймера;

4) разрешаются внешние прерывания;

5) останавливается тактовый генератор.

Когда поступает внешний сигнал RESET\ или IRQ\, включается тактовый генератор, после задержки на инициализацию, которая длится 1920 циклов процессора, защелкиваются вектор прерывания и вектор начальной установки, и запускается на выполнение процедура обработки RESET или IRQ, в зависимости от поданного сигнала.

Флаг I сбрасывается.

Более подробно о режиме STOP см. раздел "Специальные режимы работы".

***WAIT*** *Ожидание*

Выполнение команды приводит к уменьшению энергопотребления. Таймер, делитель таймера и другая встроенная периферия продолжают работать, так как они являются потенциальными источниками прерываний. WAIT разрешает прерывания сбросом бита I регистра признаков и останавливает цепи синхронизации процессора.

Когда поступает внешний сигнал RESET\, IRQ\ или прерывания от встроенной периферии, включается цепи синхронизации процессора, защелкиваются вектор прерывания и вектор начальной установки, и запускается на выполнение процедура, взвисимости от поданного сигнала.

Более подробно о режиме WAIT см. раздел "Специальные режимы работы".

## 2. Аппаратные средства

### 2.1 Линии (порты) параллельного ввода-вывода.

МС68НС705С8 содержит 31 линию ввода-вывода общего назначения, объединенные в 4 порта. Порты А, В и С представляют собой 8-битные порты общего назначения, в которых направление работы каждой линии задается через программно-доступные регистры, порт D является 7-битным портом, настроенным только на ввод данных, кроме того, линии порта D могут выполнять некоторые специальные функции. Подробнее про порт D смотрите п. 2.4. Каждый 8-битный порт имеет соответствующий 8-битный регистр направления данных (DDR), как показано на рис. 5, а так же регистр данных (DR), в который помещаются данные при передаче, или из которого они читаются при приеме.

Каждый из трех портов - A, B, или C можно представить схематически следующим образом:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Бит 7 | 6 | 5 | 4 | 3 | 2 | 1 | Бит 0 |  |
| DDR7 | DDR6 | DDR5 | DDR4 | DDR3 | DDR2 | DDR1 | DDR0 | DDR |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | После RESET |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | Порт (DR) |

|  |  |
| --- | --- |
| Регистр порта не изменяется после RESET. |  |

 P7 P6 P5 P4 P3 P2 P1 P0 Контакты

Рис. 2-1 Схема порта ввода/вывода.

 Канал настроен на вывод информации, если соответствующий бит в DDR установлен в 1. Если бит сброшен, то канал настроен на ввод (так предусмотрено при начальной установке). То есть разные каналы одного и того же порта могут быть настроены на ввод или вывод информации одновременно независимо друг от друга. На рис. 2-2 представлена структура канала параллельного порта, а в таблице 2-1 приведены различные режимы работы линии ввода-вывода.

*Таблица 2-1. Функции контактов ввода-вывода.*

|  |  |  |
| --- | --- | --- |
| R/W\* | DDR | Функция |
| 0 | 0 | Режим ввода. Данные записываются в выходную защелку |
| 0 | 1 | Данные записываются в выходную защелку и передаются на контакт. |
| 1 | 0 | Читается состояние контакта |
| 1 | 1 | Контакт в режиме вывода. Читаются данные из защелки. |

 \*R/W - внутренний сигнал, указывающий на чтение/запись данных.

Из рисунка и таблицы видно, что когда канал настраивается на вывод, чтение регистра данных приводит к чтению данных из выходного триггера, а не с вывода непосредственно. Регистры направления данных расположены по адресам 04Н, 05Н, 06Н для портов А, В и С, соответственно. Регистры данных расположены по адресам 00Н, 01Н, 02Н в той же последовательности.



1. Буфер вывода, разрешает вывод, когда соответствующий бит в регистре DDR установлен в 1 (вывод данных).

2. Буфер ввода, включен, когда соответствующий бит в регистре DDR сброшен на 0 (ввод данных).

3. Буфер ввода, включен, когда соответствующий бит в регистре DDR установлен в 1 (вывод данных).

Рис. 2-2 Структура канала параллельного порта

### 2.2 Система прерываний.

МС68НС705С8 имеет 5 типов прерываний, включающих 4 маскируемых аппаратных прерывания и одно немаскируемое программное прерывание. К аппаратным прерываниям относятся:

 -внешнее прерывание, поступающее через вывод IRQ/ микросхемы

 -прерывание от последовательного интерфейса связи

 -прерывание от последовательного периферийного интерфейса

 -прерывание от таймера

 При возникновении прерывания процессор сохраняет содержимое регистров в стеке, и затем устанавливает бит маски прерываний I в регистре признаков, запрещая последующие прерывания. Команда возврата из прерывания (RTI), которая является последней командой в программе обработки любого прерывания, восстанавливает содержимое регистров из стека. Порядок загрузки и выгрузки регистров из стека показан на рис. 2-3.

В отличие от RESET, прерывание не вызывает прекращения выполнения текущей команды, обработка прерывания начинается лишь после ее завершения.

Таймер и оба последовательных интерфейса имеют собственные регистр статуса и управляющий регистр. В регистре статуса расположены флаги прерываний, каждый из которых соответствует определенному действию, вызвавшему прерывание. Например, у таймера таких флагов три. Флаги в регистрах статуса предназначены только для чтения. В управляющем регистре каждому из этих флагов поставлен в соответствие управляющий бит, с помощью которого можно разрешить или запретить определенное прерывание установкой или сбросом соответствующего бита. Если управляющий бит сброшен, то прерывание блокируется, но это не мешает изменению состояния флага запроса прерывания. Во время начальной установки все управляющие биты сбрасываются, чтобы предотвратить прерывания в течение процедуры начальной установки.

Рис. 2-3 Порядок загрузки регистров в стек и

выгрузки их из стека.

На рисунке 2-4 показан процесс обработки прерывания, а в таблице 2-2 даны векторы всех прерываний. В первом столбце таблицы указан регистр статуса, в котором находится соответствующий флаг прерывания, во втором столбце даны названия флагов прерываний. В третьем столбце кратко описана причина возникновения прерывания, в четвертом столбце указано устройство-источник прерывания, и в пятом столбце приведен адрес по которому хранится вектор прерывания. Важно отметить, что бит маски прерываний (бит I в регистре признаков) будет сброшен только в том случае, если соответствующий бит в стеке хранит 0.

Рассмотрим подробнее каждый тип прерывания.

***Программное прерывание*** является результатом выполнения команды SWI (SoftWare Interrupt). Оно возникает вне зависимости от состояния бита I. Вектор прерывания содержится в ячейках памяти с адресами 1FFCH и 1FFDH.

***Внешнее прерывание*** происходит если бит маски прерываний (бит I) сброшен и на внешний вывод IRQ/ поступает сигнал активного уровня (режим срабатывания задается записью в бит IRQ регистра конфигурации, см. п. 1.3). Затем текущее состояние процессора запоминается в стеке, и бит I устанавливается в 1 для маскирования всех прерываний до тех пор, пока обслуживается внешнее.

***Прерывание от таймера*** может быть вызвано тремя флагами, как уже упоминалось ранее. Эти три флага расположены в регистре статуса таймера, и имеют один и тот же вектор (см. таблицу 2-2). Каждый флаг имеет соответствующий управляющий бит (ICIE, OCIE и TOIE) в регистре управления таймером. Установка этих битов разрешает соответствующее прерывание, сброс-запрещает. Подробнее назначение каждого флага описано в главе 2.3 "Таймер".



Рис. 2-4 Процесс обработки прерывания

***Прерывание от последовательного интерфейса связи*** происходит в том случае, если один из пяти флагов (TDRE, TC, RDRF, IDLE, OR) в регистре статуса интерфейса связи установлен, бит I в регистре признаков сброшен и управляющий бит во втором управляющем регистре установлен. При обслуживании прерывания приоритет обработки задается программно через проверку флагов в регистре статуса. Подробнее о регистрах интерфейса связи рассказывается в главе 2.4 "Последовательный интерфейс".

***Прерывание от последовательного периферийного интерфейса*** происходит при тех же условиях (установлен один из двух флагов в регистре статуса периферийного интерфейса (SPIF или MODF), бит I сброшен, и управляющий бит в управляющем регистре установлен). Подробнее о периферийном интерфейсе рассказывается в главе 2.4 "Последовательный интерфейс".

Функциональная схема обработки внешних прерываний приведена на рис. 2-5. Внешнее прерывание может определяться либо фронтом сигнала, либо его фронтом и уровнем. Задается это программно через регистр конфигурации (см. гл. 1.3). Если прерывание определяется фронтом сигнала, то следующее прерывание не должно возникать до тех пор, пока не будет обработано первое, т. е. пока не будет выполнена команда возврата из прерывания (RTI). Время между двумя прерываниями должно быть минимум на 21 машинный цикл больше, чем время обработки прерывания (не включая команду RTI). При определении прерывания по уровню сигнала, следующее прерывание будет определяться в том случае, если после обработки первого прерывания на выводе IRQ/ будет сохраняться сигнал низкого уровня. Этот метод обычно используется при работе с несколькими источниками прерываний, объединенными в одну линию. Оба режима реакции на внешнее прерывание наглядно демонстрируются на рис. 2-6 и 2-7, соответственно.

*Таблица 2-2. Векторы прерываний.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Регистр | Имя флага | Причина прерывания | Прерывание ЦПУ | Адрес вектора |
|  |  | RESET | RESET | 1FFE-1FFF |
|  |  | Программное | SWI | 1FFC-1FFD |
|  |  | Внешнее | IRQ | 1FFA-1FFB |
| Регистр статуса таймера | ICFOFC TOF  | Входная фиксацияВыходное сравнениеПереполнение | TIMER | 1FF8- 1FF9  |
| Регистр статуса интерфейса связи | TDRETCRDRFIDLEOR | Буфер передачи пустПередача завершенаПолон буфер приемаОбнаружена свободная линияПереполнение | SCI | 1FF6-1FF7 |
| Регистр статуса периферий-ного интерфейса | SPIFMODF | Передача завершенаРежим ошибочен | SPI | 1FF4-1FF5 |

Рис. 2-5 Функциональная схема обработки внешнего прерывания.



Рис. 2-6 Диаграмма, иллюстрирующая первый режим обнаружения внешнего прерывания.

Рис. 2-7 Диаграмма, иллюстрирующая второй режим обнаружения внешнего прерывания.

### 2.3 Таймер и связанные с ним устройства.

Блок таймера МС68НС705С8 имеет следующие характеристики:

 16-битный программируемый счетчик с предделителем тактовой частоты.

 Флаг переполнения таймера.

 16-битный регистр выходного сравнения.

 16-битный регистр входной фиксации.

 Три вида генерируемых прерываний.

Основу таймера составляет 16-битный программируемый счетчик, синхронизируемый внутренней тактовой частотой, деленной на 4. Функция переполнения таймера позволяет расширить возможность измерения промежутков времени путем фиксации момента переполнения счетчика и установки соответствующего флага в регистре статуса. Функция входной фиксации может использоваться для автоматической записи времени определенного процесса, например для измерения длительности внешнего сигнала. Функция выходного сравнения дает возможность реализовать генерацию сигналов различной частоты, формировать импульсы различной длительности, а так же задержки в программе и многое другое.

Таймер позволяет формировать прерывания по переполнению как основного счетчика, так и 16-битного регистра входной фиксации и 16-битного регистра выходного сравнения.

Блок-схема таймера представлена на рис. 2-8.

Поскольку таймер имеет 16-битовую архитектуру, каждая его функция представлена двумя регистрами, содержащими старший и младший байт. К регистрам таймера относятся в первую очередь ***счетчик и альтернативный счетчик***.



Рис. 2-8 Блок-схема таймера.

При начальной установке 16-битный счетчик таймера, расположенный по адресу 18Н-19Н, начинает работу со значения FFFCН. Далее значение счетчика увеличивается на единицу при поступлении синхроимпульса с предделителя частоты, дающего разрешение в 2 мкс при использовании кварца 4МГц. Когда значение счетчика достигает FFFFH, он сбрасывается в 0000H, устанавливая флаг переполнения. Значение счетчика дублируется в альтернативном счетчике, расположенном по адресу 1АН-1ВН. Единственное различие этих регистров состоит в том, что при чтении счетчика происходит сброс флага переполнения, а при обращении к альтернативному счетчику этого не происходит. Счетчик, как и альтернативный счетчик, может быть прочитан в любой момент, но не может быть записан. Если сначала читается старший байт счетчика (18Н), то младший байт запоминается в буфере. Это значение остается в буфере неизменным даже если старший байт читается несколько раз. Доступ к буферу осуществляется при чтении младшего байта счетчика. Если же считывать несколько раз только содержимое младшего байта счетчика, то каждый раз при этом будут считываться разные значения, соответствующие моменту считывания.

***Регистр входной фиксации*** (Input Capture Register) расположен по адресу 14Н-15Н и предназначен только для чтения из него времени обнаружения на выводе TCAD сигнала. Уровень сигнала, вызывающего фиксацию времени, задается программно через бит IEDG управляющего регистра. RESET не оказывает влияния на содержимое регистра входной фиксации. Значение в регистре ICR при фиксации будет на единицу больше значения счетчика в момент фиксации. Эта задержка необходима для внутренней синхронизации. Таким образом, счетчик и регистр входной фиксации имеют одинаковое разрешение, равное четырем циклам внутренней синхронизации.

При обнаружении сигнала на выводе ТСАP содержимое счетчика фиксируется в ICR вне зависимости от состояния флага ICF (см. регистр статуса). Поэтому ICR всегда содержит время последней фиксации.

При чтении старшего байта регистра входной фиксации (14Н) фиксация следующего значения запрещается до тех пор, пока не будет прочитан младший байт. Чтение же младшего байта не влияет на возможность очередной входной фиксации.

***Регистр выходного сравнения*** (Output Compare Register) расположен по адресу 16Н-17Н и используется для программирования какого-либо действия в определенное время (т. е. когда счетчик достигнет определенного значения). Все биты регистра могут быть прочитаны и записаны в любой момент времени, и не изменяют свое состояние при RESET. Если в функции выходного сравнения нет необходимости, эти ячейки памяти могут использоваться для других целей. Значение в регистре выходного сравнения сравнивается со счетчиком каждые 4 цикла. Когда значения совпадают, устанавливается флаг (OCF) и уровень бита OLVL (см. регистр статуса таймера) передается на вывод ТСМР. Так же может происходить прерывание, если установлен бит OCIE. Содержимое регистра выходного сравнения и бит OLVL должны изменяться после каждого удачного сравнения, для того чтобы начать новый цикл сравнения. При записи сначала в старший байт ОСR (16Н) сравнение не будет производиться до тех пор, пока не будет записан младший байт. При записи же сначала в младший байт, функция сравнения будет активирована. Таким образом, пользователь должен записать оба байта OCR, если сначала записывается старший.

***Управляющий регистр (TCR),*** находящийся по адресу 12Н, содержит 5 управляющих битов. Каждому из трех бит управления прерываниями соответствует флаг статуса таймера ICF, OCF и TOF.

 Управляющий регистр таймера

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название | ICIE | OCIE | TOIE | 0 | 0 | 0 | IEDG | OLVL |
| Установка после RESET | 0 | 0 | 0 | 0 | 0 | 0 | - | 0 |

ICIE - бит запрета/разрешения прерывания при входной фиксации,

 при ICIE=1-прерывание разрешено,

 при ICIE=0-прерывание запрещено,

OCIE - бит запрета/разрешения прерывания при выходном сравнении,

 при ОCIE=1-прерывание разрешено,

 при OCIE=0-прерывание запрещено,

TOIE - бит запрета/разрешения прерывания при переполнении таймера,

 при TOIE=1-прерывание разрешено,

 при TOIE=0-прерывание запрещено,

IEDG - бит определяет, что будет инициировать фиксацию счетчика в регистре входной фиксации - фронт или спад импульса на TCAP.

 при IEDG=1-фронт,

 при IEDG=0-спад

 RESET не оказывает влияния на этот бит.

OLVL - бит определяет, какой уровень будет устанавливаться на TCMP

при выходном сравнении.

 при OLVL=1-высокий уровень,

 при OLVL=0-изкий уровень.

***Регистр статуса таймера*** находится по адресу 13Н, предназначен только для чтения и содержит три флага, каждый из которых относится к определенному прерыванию.

 Регистр статуса

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Назначение | ICF | OCF | TOF | 0 | 0 | 0 | 0 | 0 |
| Установка после RESET | U | U | U | 0 | 0 | 0 | 0 | 0 |

ICF-флаг входной фиксации,

 ICF=1 - флаг устанавливается, когда входной сигнал соответствующей полярности зафиксирован,

 ICF=0 - флаг сбрасывается при чтении младшего байта входного регистра фиксации (15Н), после чтения TSR, пока установлен ICR.

OCF-флаг выходного сравнения,

 OCF=1 - флаг устанавливается, когда значение в регистре выходного сравнения совпадает со значением в счетчике.

 OCF=0 - флаг сбрасывается при записи в младший байт (17Н) регистра выходного сравнения, после чтения TSR пока установлен ОCF.

TOF-флаг переполнения таймера,

 TOF=1 - флаг устанавливается при переключении счетчика со значения FFFFH на 0000H.

 TOF=0 - флаг сбрасывается при чтении младшего байта (19Н) счетчика после чтения TSR пока установлен TOF.

Для того, чтобы сбросить бит статуса, необходимо произвести две операции: во-первых, прочитать регистр статуса, во-вторых, прочитать регистр, соответствующий определенному биту статуса (например, для бита ICF это регистр ICR).

При использовании функции переполнения таймера и обращениях к счетчику в различное время могут возникнуть проблемы, связанные со сбросом флага переполнения. Это может произойти в одном из двух случаев: либо регистр статуса был прочитан в тот момент, когда флаг TOF был установлен, либо младший бит счетчика был прочитан без цели воздействовать на флаг переполнения. Эти возможности необходимо отслеживать при написании программ.

Работа таймера в режимах STOP и WAIT описана в разделе 3.2.

### 2.4 Последовательный интерфейс

Порт D представляет собой 7-и битный (канал D6 не используется) однонаправленный порт (только для ввода данных, если он используется как порт общего назначения для параллельного обмена). Когда активизированы системы последовательного интерфейса связи (Serial Communication Interface), либо последовательного периферийного интерфейса (Serial Periferal Interface), управление портом D переходит к ним. Во время начальной установки порт D конфигурируется на ввод информации, поскольку последовательные интерфейсы блокированы.

Последовательный периферийный интерфейс использует 4 вывода порта D (MISO и MOSI-ввод-вывод информации, SCK-системная синхронизация, SS-ведущий/ведомый). Последовательный интерфейс связи использует 2 вывода (RDI-прием данных, TDO-передача).

Рассмотрим подробнее ***последовательный интерфейс связи.*** Полно-дуплексный асинхронный последовательный интерфейс связи работает в стандартном NRZ формате с широким выбором скорости приема/передачи информации. Передатчик и приемник функционально независимы, но используют одинаковый формат данных и скорость работы.

Общие особенности интерфейса:

 стандартный NRZ формат;

 усовершенствованная система выявления ошибок, включающая фильтрацию помех с длительностью до 1/16 времени передачи 1 бита;

 полно-дуплексное функционирование (одновременный прием и передача);

 программирование на одну из 32 возможных скоростей обмена;

 программно-изменяемая длина слова (8 либо 9 бит в слове);

 отдельно расположенные биты управления приемом и передачей;

 возможность управления по прерыванию;

 различных условия возникновения прерывания;

 Особенности приемника:

 наличие функции активации передатчика (свободная линия или разряд адреса);

 обнаружение свободной линии;

 обнаружение ошибок кадрирования (см. ниже);

 выявление шумов;

 обнаружение переполнения;

 флаг контроля за полнотой регистра данных;

 Особенности передатчика:

 флаг контроля за пустотой регистра данных;

 флаг завершения передачи;

 прерывание посылки;

Структура интерфейса представлена на рис. 2-9.

Пользователь имеет возможность задать режим работы интерфейса через ***первый управляющий регистр (SCCR1).*** ***Второй управляющий регистр (SCCR2)*** содержит биты контроля за приемом/передачей, разрешением/запретом прерываний и другие. Рассмотри работу SCI в различных режимах.

***Передача данных*** начинается с записи в регистр передачи данных (SCDAT),который физически расположен в одной и той же ячейке памяти с регистром приема данных. При условии, что работа передатчика разрешена, данные передаются в сдвиговый регистр. Эта передача данных устанавливает бит TDRE (пуст регистр передачи данных) в регистре SCSR и генерирует прерывание, если прерывания разрешены. Передача данных в сдвиговый регистр синхронизируется по внутреннему генератору скорости. Данные передаются младшим битом вперед.. При завершении передачи данных, бит TC устанавливается и генерируется прерывание. Если передатчик блокирован, и были посланы либо данные, либо код останова, бит ТС также будет установлен. Если передатчик блокируется во время передачи, то происходит завершение передачи текущего байта, и затем останов передатчика.

***Прием данных*** начинается с чтения из регистра SCDAT, содержащего последний полученный байт, при условии что приемник не блокирован. Бит RDRF в регистре SCSR устанавливается, если данные из сдвигового регистра перенесены в SCDAT, что может вызвать кроме того прерывание от периферийного интерфейса, если оно разрешено. Передача данных из сдвигового регистра в SCDAT синхронизируется по внутреннему генератору. Флаги переполнения (OR), обнаружения шумов (NF), либо FE устанавливаются, если происходит ошибка при приеме данных.

На рис. 2-10 представлена диаграмма обмена в упоминавшимся выше NRZ-формате, используемом в интерфейсе.



Рис. 2-9 Структурная схема последовательного интерфейса связи.

(1)-контрольный бит, задается программно.

Рис. 2-10 Диаграмма обмена по SCI.

Рассмотрим регистры, входящие в интерфейс.

***Регистр скорости передачи данных (BAUD)***, расположенный по адресу 0DH, используется для задания скорости обмена по интерфейсу. Из таблиц 2-3 и 2-4 можно определить скорость приема/передачи информации, исходя из частоты кварцевого генератора.

 Регистр скорости приема/передачи

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название | - | - | SCP1 | SCP0 | - | SCR2 | SCR1 | SCR0 |
| Установка после RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

 SCP1, SCP0-предварительное деление на 1, 3, 4, 13.

 SCR2, SCR1, SCR0-деление после предварительного делителя на 1, 2, 4, ... ,128.

*Таблица 2-3. Предделитель тактовой частоты.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SCP1 | SCP0 | Делитель тактовой частоты | 4.19430 | 4.0 | 2.4576 | 2.0 | 1.8432 |
| 0 | 0 | 1 | 131072 | 125000 | 76800 | 62500 | 57600 |
| 0 | 1 | 3 | 43691 | 41666 | 25600 | 20833 | 19200 |
| 1 | 0 | 4 | 32768 | 31250 | 19200 | 15625 | 14400 |
| 1 | 1 | 13 | 10082 | 9600 | 5907 | 4800 | 4430 |

*Таблица 2-4. Частота приема/передачи*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SCR2 | SCR1 | SCR0 | Делитель внутренней частоты | 131.072 | 32768 | 76800 | 19200 | 9600 |
| 0 | 0 | 0 | 1 | 131072 | 32768 | 76800 | 19200 | 9600 |
| 0 | 0 | 1 | 2 | 65536 | 16384 | 38400 | 9600 | 4800 |
| 0 | 1 | 0 | 4 | 32768 | 8192 | 19200 | 4800 | 2400 |
| 0 | 1 | 1 | 8 | 16384 | 4096 | 9600 | 2400 | 1200 |
| 1 | 0 | 0 | 16 | 8192 | 2048 | 4800 | 1200 | 600 |
| 1 | 0 | 1 | 32 | 4096 | 1024 | 2400 | 600 | 300 |
| 1 | 1 | 0 | 64 | 2048 | 512 | 1200 | 300 | 150 |
| 1 | 1 | 1 | 128 | 1024 | 256 | 600 | 150 | 75 |

***Первый управляющий регистр*** расположен по адресу 0ЕН (SCCR1) и содержит 3 бита, связанных с использованием 9-битного формата данных, а также один бит, определяющий способ активизации интерфейса.

 Первый управляющий регистр

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название | R8 | T8 | - | M | WAKE | - | - | - |
| Установка после RESET | 0 | 0 | - | 0 | 0 | - | - | - |

R8-принимаемый девятый бит.

T8-передаваемый девятый бит.

M-бит, определяющий длину посылки:

 при М=1- 9 бит данных;

 при М=0 -8 бит данных.

WAKE-метод активизации приемника:

 при WAKE=1-приемник активизируется при обнаружении разряда адреса (старшего бита);

 при WAKE=0-свободная линия вызывает активизацию приемника.

***Второй управляющий регистр (SCCR2)*** расположен по адресу 0FH и осуществляет управление такими индивидуальными функциями интерфейса, как прерывания, блокировка приемника/передатчика и другими.

Второй управляющий регистр

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  Название  | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK |
| Установка после RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

TIE-бит разрешения/запрета прерываний передатчика,

 при TIE=1-прерывания разрешены,

 при TIE=0-прерывания запрещены.

TCIE-бит разрешения/запрета прерываний при завершении передачи,

 при TCIE=1-прерывания разрешены,

 при TCIE=0-прерывания запрещены.

RIE-бит разрешения/запрета прерываний от приемника,

 при RIE=1-прерывания разрешены,

 при RIE=0-прерывания запрещены.

ILIE-бит разрешения/запрета прерывания при освобождении линии ,

 при ILIE=1-прерывание разрешено,

 при ILIE=0-прерывание запрещено.

TE-бит разрешения/запрета передачи,

 при TE=1-выход передающего сдвигового регистра подключен к линии TD0. В зависимости от бита M передаются 10 (M=0) или 11 (M=1) бит.

 при TE=0-передача запрещается после завершения передачи байта, который обрабатывался в момент сброса бита TE, линия TD0 переходит в высокоимпедансное состояние,

RE-бит запрета/разрешения приема,

 при RE=1-линия RDI соединяется со сдвиговым регистром приемника,

 при RE=0-прием запрещен.

RWU-метод активизации приемника,

 при RWU=1-приемник в режиме ожидания,

 при RWU=0-если бит WAKE=1, то бит RWU сбрасывается после приема слова данных. Если WAKE=0, бит RWU сбрасывается после приема 10 (M=0), либо 11 (M=1) бит.

SBK-посылка сигнала останов,

 при SBK=1-передатчик посылает блоки нулей (серии из 10 или 11) до очистки сдвигового регистра. Затем передатчик посылает один бит высокого уровня для определения старт-бита.

 при SBK=0-передатчик посылает 10 или 11 нулей и затем возвращается в свободное состояние или продолжает посылки информации.

Если сдвиговый регистр передатчика пуст и линия свободна, установка и очистка SBK может поставить в очередь до двух посылок сигнала останова, т.к. первый сигнал передается немедленно в сдвиговый регистр, а второй встает в очередь в параллельный буфер передатчика.

***Регистр статуса***(SCSR) находится по адресу 10Н и содержит два флага статуса передатчика, и 5 флагов, относящихся к приемнику.

Регистр статуса

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название | TDRE  | TC | RDRF | IDLE | OR | NF | FE | - |
| Установка после RESET | 1 | 1 | 0 | 0 | 0 | 0 | 0 | - |

TDRE-сообщает о наличии данных в регистре передачи (ТDR),

 при TDRE=1-регистр пуст и может принять новые данные,

 при TDRE=0-регистр содержит данные. Чтение SCSR сбрасывает TDRE, если оно следует за записью в TDR.

TC-флаг завершение передачи,

 при ТС=1-передатчик пуст (включая сдвиговый регистр),

 при ТС=0-передатчик активирован. Чтение SCSR сбрасывает TDRE, если оно следует за записью в TDR.

RDRF -флаг наличия данных в регистре приема данных (RDR),

 при RDRF=1-устанавливается, когда содержимое сдвигового регистра передано в RDR,

 при RDRF=0-данных в RDR нет. RDRF сбрасывается при чтении SCSR, следующие за чтением RDR.

IDLE-обнаружение свободной линии,

 при IDLE=1-Индицирует, что приемник обнаружил свободную линию,

 при IDLE=0-IDLE сбрасывается при чтении SCSR, следующим за чтением RDR. Если IDLE сброшен, то он не может быть установлен в 1 до тех пор, пока линия RDI не будет активной и свободной вновь.

OR-флаг ошибки переполнения,

 при OR=1-индицирует, что регистр приема данных переслан в RDR до считывания информации из него (RDRF=1). Данные, вызвавшие переполнение, теряются, а данные в RDR сохраняются,

 при OR=0-OR сброшен при чтении SCSR, следовавшим за чтением RDR.

NF-флаг помехи,

 при NF=1-индицирует наличие помехи на шине приема. NF не устанавливается, пока RDRF=1.

FE-флаг ошибки кадрирования,

 при FE=1-индицирует отсутствие стоп-бита в принятых данных. FE устанавливается одновременно с установкой RDRF. Если принятый байт вызывает и ошибку кадрирования, и ошибку переполнения, процессор опознает только переполнение. Дальнейшая передача данных в RDR запрещается до сброса FE.

 при FE=0-сброс бита FE произошел при чтении SCSR, следовавшим за чтением RDR.

***Регистр данных*** (SCDAT) расположен по адресу 11Н и используется для передачи и приема данных.

Регистр данных

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  SCD7 |  SCD6 |  SCD5 |  SCD4 |  SCD3 |  SCD2 |  SCD1 |  SCD0 |

Как уже упоминалось, SCDAT функционирует как два раздельных регистра: передающий и принимающий.

Перейдем к рассмотрению ***Последовательного периферийного интерфейса***, дающего возможность нескольким ОЭВМ и/или периферийным устройствам быть объединенными в одно целое. Периферийные устройства могут представлять собой как простой сдвиговый TTL регистр, так и сложные подсистемы, например драйверы жидкокристаллических дисплеев или системы ЦАП или АЦП.

В формате периферийного интерфейса сигнал синхронизации не включен в поток данных и выполнен в виде отдельного сигнала. Система на основе последовательного периферийного интерфейса должна состоять из одного ведущего устройства (ОЭВМ) и нескольких ведомых устройств, которые также могут представлять собой ОЭВМ (см. рис. 2-13), либо ОЭВМ, которые могут быть как ведущими, так и ведомыми.

 Особенности интерфейса:

 полно-дуплексная, трехпроводная синхронная передача данных;

 работа ОЭВМ в качестве ведущего или ведомого устройства;

 максимальная частота ведущего 1.05 МГц;

 максимальная частота ведомого 2.1 МГц;

 четыре программируемые скорости передачи данных ведущим;

 программируемая полярность синхронизации и фаза;

 флаг прерывания при окончании передачи;

 флаг защиты от конфликта при записи данных;

 возможность защиты от сбоя режима ведущий-ведомый.

Рис. 2-11 Последовательный периферийный интерфейс. Блок-схема.

Блок-схема интерфейса представлена на рис. 2-11. При работе с периферийным интерфейсом нет необходимости указывать направление передачи данных, поскольку "ведущий" одновременно производит передачу и прием по раздельным линиям. Сдвиговые регистры ведущего и ведомого устройств удобно рассматривать как один 16-битный регистр, замкнутый в кольцо. Когда происходит передача данных, содержимое регистра сдвигается на 8 позиций, и устройства обмениваются данными.

Как видно из рис. 2-12, в состав периферийного интерфейса входят 4 линии (MOSI, MISO, SCK и SS), назначение которых приведено ниже:

**MOSI (Master Out, Slave In)**-линия, предназначенная для ввода последовательной информации, если устройство находится в режиме "ведомый", или для вывода информации, если устройство находится в режиме "ведущий". Данные по линии передаются в одном направлении старшим битом вперед.

**MISO (Master In, Slave Out)**-линия, предназначенная для ввода последовательной информации, если устройство находится в режиме "ведущий", или для ввода информации, если устройство находится в режиме "ведомый". Данные по линии также передаются старшим битом вперед. Линия MISO ведомого устройства находится в высокоимпедансном состоянии, если режим "ведомый" для него не выбран (SS\=1). Линия MISO ведомого устройства находится в состоянии высокого импеданса, если SS=1.

**SCK (Serial CloK)**-вывод синхронизации. Поскольку синхронизация происходит по ведущему устройству, линия SCK ведомого устройства работает на ввод. Как показано на рисунке, возможны четыре различных типа синхронизации, которые могут быть выбраны через биты CPOL и CPHA регистра управления. Режим синхронизации задается одинаковым для ведущего и ведомого устройств. Биты SPR0 и SPR1 в регистре управления ведущего устройства задают скорость обмена информацией. Значение этих бит у ведомого прибора не имеет значения.

**SS\ (Slave Select)**-выбор ведущий/ведомый. Линия SS\ имеет высокий уровень у ведущего устройства. Если вывод SS\ ведущего устройства переходит в состояние логического нуля, в регистре статуса SPSR устанавливается флаг MODF. Если ОЭВМ работает как ведущее устройство, генератор ведущего устройства является источником сигнала синхронизации для ведомых устройств. Данные параллельно загружаются в 8-битный сдвиговый регистр из внутренней шины (в ходе цикла записи) и затем последовательно поступают через вывод MOSI на ведомое устройство. В ходе цикла чтения данные поступают последовательно через вывод MISO от ведомого устройства в 8-битный сдвиговый регистр. Затем данные параллельно переносятся в буфер чтения и становятся доступными для чтения с внутренней шины. Если ОЭВМ работает как ведомое устройство, то на внешний вывод SS\ должен поступать сигнал низкого уровня, а на вывод SCK-сигнал синхронизации. Данные от ведущего устройства поступают последовательно на вывод MOSI и передаются в 8-битный сдвиговый регистр для параллельного переноса в буфер чтения. У ведомого в ходе цикла записи, данные параллельно загружаются в 8-битный сдвиговый регистр с внутренней шины данных, ожидая сигнала синхронизации от ведущего, для того чтобы последовательно передать данные через вывод MISO.

Рис. 2-12. Структурная схема SPI.

На рис. 2-13 представлена типовая схема подключения пяти микропроцессорных устройств, из которых одно является ведущим, а четыре-ведомыми. В этой системе у каждого устройства обозначены основные контакты: MISO, MOSI, SCK и SS\.

Рис. 2-13 Схема подключения устройств с использованием периферийного интерфейса.

Рис. 2-14 Диаграмма работы SPI.

На рисунке 2-14 представлена временная диаграмма, показывающая соотношение данных с сигналом синхронизации. Как видно из рисунка, возможны 4 различных варианта, которые могут быть выбраны через биты CPOL и CPHA управляющего регистра. Различные протоколы работы SPI позволяют осуществлять обмен данными с любыми последовательными синхронными устройствами.

В состав периферийного интерфейса входят три регистра, которые осуществляют управление, отражают статус, а так же выполняют другие функции.

***Управляющий регистр*** (SPSR), расположенный по адресу 0АН, контролирует индивидуальные функции интерфейса: прерывания, разрешение/запрет доступа, выбор ведущего/ведомого устройства и другими.

Управляющий регистр

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название | SPIE | SPE | - | MSTR | CPOL | CPHA | SPR1 | SPR0 |
| Установка после RESET | 0 | 0 | - | 0 | 0 | 0 | 0 | 0 |

SPIE-бит разрешения/запрета прерываний от последовательного интерфейса связи,

 при SPIE=1 прерывания разрешены,

 при SPIE=0 прерывания запрещены,

SPE-активация/блокировка интерфейса,

 при SPE=1-система периферийного интерфейса активирована,

 при SPE=0-система блокирована,

MSTR-бит выбора режима ведущий/ведомый,

 при MSTR=1-режим "ведущий",

 при MSTR=0-режим "ведомый",

CPOL-бит полярности синхронизации (см. рис. 17),

 при CPOL=1-SCK-линия считается нерабочей в высоком состоянии,

 при CPOL=0-SCK-линия считается нерабочей в низком состоянии,

CPHA-фаза синхронизации (см. рис. 17),

 при CPHA=1 вывод SS\ является управляющим выводом доступа к данным

 при CPHA=0-сдвиг синхронизации является логическим "ИЛИ" SCK и SS. Когда SS в состоянии 0 первый фронт SCK является сигналом начала передачи.

Биты SPR0,SPR1 задают скорость передачи данных, которую можно выбрать, руководствуясь таблицей:

|  |  |  |
| --- | --- | --- |
| SPSR1 | SPSR0 | Внутренняя синхронизация делится на |
| 0 | 0 | 2 |
| 0 | 1 | 4 |
| 1 | 0 | 16 |
| 1 | 1 | 32 |

***Регистр статуса*** расположен по адресу 0ВН и содержит три бита статуса

Регистр статуса

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название | SPIF | WCOL | - | MODF | - | - | - | - |
| Установка после RESET | 0 | 0 | - | 0 | - | - | - | - |

SPIF-флаг пересылки данных,

 при SPIF=1-индицирует завершение обмена данными между ОЭВМ и внешним устройством. Передача данных начинается при записи "ведущим" данных в регистр данных. (если SPIF=1 и SPIE=1 прерывания разрешены),

 при SPIF=0-сброс происходит при чтении SPSR (где SPIF=1), следующим за доступом к SPDR.

WCOL-флаг конфликта записи,

 при WCOL=1-индицирует попытку записи данных в SPDR в тот момент, когда процесс обмена данными еще не завершен, передача данных прекращается,

 при WCOL=0-сброс происходит при чтении SPSR (где WCOL=1), следующим за доступом к SPDR.

MODF-флаг сбоя режима,

 при MODF=1-индицирует конфликт системы из-за наличия нескольких ведущих устройств. Флаг устанавливается, если cигнал на линии SS/ переходит в низкий уровень, в то время, как ОЭВМ конфигурируется ведущим устройством,

 при MODF=0- сброс достигается при чтении SPSR (где MODF=1), следующим за записью в SPSR.

***Регистр данных*** расположен по адресу 0СН и предназначен для обмена данными. Запись в этот регистр у ведущего устройства инициирует начало приема/передачи данных. Затем данные передаются прямо в сдвиговый регистр, который обменивается данными с ведомым устройством. По окончание обмена устанавливается флаг SPIF.

Регистр данных

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  SPD7 |  SPD6 |  SPD5 |  SPD4 |  SPD3 |  SPD2 |  SPD1 |  SPD0 |

## 3. Специальные режимы работы.

### 3.1 RESET.

RESET, или начальная установка, по-существу означает запуск программы, хранящейся в ППЗУ ОЭВМ, с определенного адреса с установкой периферийных систем, управляющих регистров и флагов в определенное состояние. Начальная установка происходит в одном из 4-х случаев:

1) Внешний RESET при подаче активного сигнала низкого уровня на вывод RESET/ микросхемы.

2) Внутренний RESET при включении питания.

3) Внутренний RESET при неправильном функционировании программы.

4) Внутренний RESET при отсутствии сигнала синхронизации.

При установке ОЭВМ в исходное состояние происходит следующее:

 все регистры направления передачи данных сбрасываются (настройка на ввод информации);

 в указатель стека загружается значение 00FFH;

 бит I в регистре признаков устанавливается в "1";

 триггер внешнего прерывания сбрасывается;

 запрещается доступ к последовательному интерфейсу связи (биты ТЕ=0 и RE=0). Остальные биты регистров интерфейса связи: TIE, TCIE, RIE, ILIE, RWU, SBK, RDRF, IDLE, OR, и FE-сбрасываются.

 биты статуса последовательного интерфейса TDRE и TC устанавливаются в 1.

 предварительный делитель частоты и биты управления скоростью обмена информацией SCP0 и SCP1 сбрасываются.

 последовательный периферийный интерфейс блокирован (бит SPE=0). Остальные биты сбрасываются: SPIE, MSTR, SPIF, WCOL, и MODF.

 все биты разрешения прерываний от устройств последовательного ввода-вывода сбрасываются (SPIE, TIE, TCIE).

 последовательный периферийный интерфейс конфигурируется как подчиненный (ведомый) компонент системы (MSTR=0).

 предварительный делитель частоты таймера сбрасывается. В счетчик таймера загружается значение FFFCH. Бит TCMP сбрасывается. Все биты разрешения прерываний от таймера сбрасываются (ICIE, OCIE, и TOIE), запрещая прерывания. Бит OLVL сбрасывается.

 триггер STOP сбрасывается.

 триггер WAIT сбрасывается.

 внутренняя шина адреса загружается исходным вектором (в старший байт программного счетчика загружается значение из ячейки с адресом 1FFEH, в младший-из ячейки 1FFFH).

Перейдем к более подробному рассмотрению режима RESET.

***RESET по включению питания*** происходит при подаче напряжения питания на вывод Vdd микросхемы, и предназначен для стабилизации работы тактового генератора. Данный вид режима RESET предусмотрен исключительно для момента включения питания, и не должен использоваться для определения падения питающего напряжения. Схема задержки включения предусматривает прохождение 4064 импульсов с момента запуска генератора. Если же и после этого периода на выводе RESET/ сохраняется сигнал низкого уровня, то процессор остается в состоянии начальной установки до тех пор, пока на вывод RESET/ не будет подан сигнал высокого уровня.

***RESET по внешнему сигналу*** происходит при условии присутствия уровня логического 0 на выводе RESET/ микросхемы в течение 1.5 машинных циклов.

***RESET по времени выполнения программы*** происходит при переполнении специального таймера COP (Computer Operating Properly, дословно-компьютер работающий правильно, далее просто таймер COP), следящего за правильностью работы программы. Если таймер СОР активизирован, то его переполнение индицирует неправильность работы прикладной программы, что, в свою очередь, вызывает установку ОЭВМ в начальное состояние.

Поскольку при работе таймер использует внутреннюю шину синхронизации, диспетчер сигнала синхронизации должен быть включен одновременно с таймером, т. к. сам СОР таймер не может определить наличие/отсутствие сигнала синхронизации. Управление таймером и диспетчером синхронизации осуществляется с помощью регистров сброса (COPRST 1DH) и управления (COPCR 1EH).

***Регистр сброса*** (COPRST) расположен по адресу 1DН.

Регистр сброса

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Бит | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Установка после RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Чтобы осуществить сброс таймера, необходимо проделать следующие операции:

-записать 55Н в COPRST

-записать ААН в COPRST

Эти операции должны производиться в описанном порядке, но не обязательно сразу друг за другом. Время между программными сбросами не должно превышать время переполнения таймера, в противном случае система будет приведена в исходное состояние. Чтение COPRST не оказывает на него влияния.

Регистр управления (COPCR) расположен по адресу 1E и содержит биты управления СОР таймером и диспетчером синхронизации (см. ниже).

Регистр управления

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название | 0 | 0 | 0 | COPF | CME | COPE | CM1 | CM0 |
| Установка после RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

COPF-флаг нормального функционирования

 при COPF=1- произошел сброс по следящему таймеру или диспетчеру синхронизации.

 при COPF=0- сброс не произошел.

 Чтение COPCR сбрасывает COPF.

CME-бит доступа к диспетчеру синхронизации.

 при CME=1- диспетчер синхронизации разблокирован.

 при CME=0- диспетчер блокирован.

 Ьит CME может быть прочитан и записан в любой момент.

COPE-бит управления СОР таймером.

 при COPE=1- COP-система разрешена.

 при COPE=0- COP-система блокирована.

 COPE может быть прочитан в любой момент, но записан только один раз.

CM1, СМ0-биты, задающие период переполнения СОР таймера (см. таблицу 3-1). Эти биты могут быть прочитаны и установлены в "1" в любой момент, но сброшены лишь при начальной установке ОЭВМ.

*Таблица 3-1 Период переполнения таймера.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CM1 | CM0 | Fp/2E15 деленная на | Fг=4.0МГцFр=2.0МГц | Fг=3.5795МГцFp=1.7897МГц | Fг=2.0МГцFр=1.0МГц | Fг=1.0МГцFр=0.5МГц |
| 0 | 0 | 1 | 16.38мс | 18.31мс | 32.77мс | 65.54мс |
| 0 | 1 | 4 | 65.54мс | 73.24мс | 131.07мс | 262.14мс |
| 1 | 0 | 16 | 262.14мс | 292.95мс | 524.29мс | 1.048с |
| 1 | 1 | 64 | 1.048с | 1.172с | 2.097с | 4.194с |

***RESET по отсутствию сигнала синхронизации*** происходит при условии, что бит CME в регистре COPCR установлен. В этом случае диспетчер синхронизации контролирует наличие/отсутствие сигнала синхронизации в течение определенного промежутка времени. Этот период колеблется от 5 до 100 мс и зависит от параметров работы процессора. Например, при рабочей частоте синхронизации 200 КГц и менее, диспетчер синхронизации использоваться не может.

Если определено отсутствие синхронизации, либо ее низкая частота, то происходит сброс, сигнал которого поступает на внешние устройства через двунаправленный вывод RESET/ в течение четырех циклов.

### 3.2 Режимы с пониженным энергопотреблением.

ОЭВМ предусматривает 3 режима с пониженным энергопотреблением, которые могут использоваться в схемах, питающихся от батарей и в автомобильной технике. Команды STOP и WAIT оказывают влияние на системы SPI, SCI и таймера.

***Режим STOP.*** В этом режиме ОЭВМ потребляет минимально возможную энергию, поскольку внутренний тактовый генератор выключен, вызывая тем самым прекращение всех внутренних процессов. В режиме останова бит I сбрасывается, разрешая все внешние прерывания. Все остальные регистры и память остаются без изменения. Без изменения остаются и линии ввода-вывода. Это состояние сохраняется до тех пор, пока не появиться сигнал IRQ либо RESET. В этот момент внутренний генератор возобновит работу. Вход в режим осуществляется программно командой STOP. ***Последовательный интерфейс связи*** в этот момент прекращает работу. Если в этот момент происходила передача информации, то она прекращается, и возобновляется при подаче сигнала низкого уровня на вывод IRQ\ микросхемы. Если же интерфейс принимал информацию, то данные теряются. Поэтому передатчик должен находиться в состоянии ожидания в ходе режима STOP. ***Последовательный периферийный интерфейс*** в ходе режима STOP продолжает прием и передачу информации если он был конфигурирован как ведомый. Единственное отличие состоит в том, что ни один флаг не будет установлен либо сброшен до тех пор, пока сигнал IRQ\ не поступит на внешний вывод. Однако при работе в режиме STOP необходимо соблюдать осторожность, поскольку схема защиты (биты WCOL, MODF и др.) не работает. Если же интерфейс был конфигурирован как ведущий, то его работа прекращается и может быть продолжена только после сигнала IRQ\. ***Таймер*** в ходе режима STOP сохраняет в счетчике последнее значение. Если же на выводе ТСАР появляется сигнал, то схема входной фиксации срабатывает, и после окончания режима STOP устанавливается соответствующий флаг (во время режима никаких действий не производится).

***Команда WAIT*** также помещает микропроцессор в режим с пониженным потреблением энергии (режим WAIT), но с несколько большим потреблением, чем при STOP-режиме. При режиме WAIT все процессы в блоке центрального процессора остановлены, но внутренний генератор, таймер и интерфейсы (связи и периферийный) в том случае, если они активизированы, работают. Прерывание от любого из этих устройств вызовет выход из режима. В ходе режима WAIT бит I сброшен, допуская внешние прерывания. Все остальные регистры-память и линии ввода-вывода остаются без изменения. то состояние сохраняется до тех пор, пока не возникнет прерывание либо RESET.

Последним из режимов с пониженным энергопотреблением является ***режим сохранения данных.*** Содержимое ОЗУ и регистров ЦП сохраняется при напряжении питания 2.0В. При таком питающем напряжении системы ОЭВМ уже не могут гарантированно работать. В ходе режима пониженного энергопотребления ОЭВМ должна находиться в состоянии RESET.

На приведенной на рис. 2-15 блок-схеме можно наглядно проследить процесс входа в режимы STOP и WAIT и условия выхода из них.

Рис. 2-15 Блок-схема режимов STOP и WAIT.

## Приложение 1. Расположение и назначение выводов МС68НС705С8.

Поскольку МС68НС705С8 является КМОП устройством, неиспользуемые входные выводы должны быть защищены от наводок и перенапряжений. Наиболее предпочтительным методом защиты является включение резистора между каждым неиспользуемым выводом и напряжением питания.

Цоколевка и наименование выводов для различных типов корпусов представлены на рисунках п1а и п1б.



Рис.п1а. Цоколевка и наименование выводов для DIP корпуса.



Рис.п1б. Цоколевка и наименование выводов для PLCC корпуса.

Рассмотрим подробнее назначение выводов ОЭВМ.

Выводы ***Vdd и Vss*** используются для подключения источника питания. На Vdd подается напряжение питания +5В, а вывод Vss соединяется с общей шиной.

Вывод ***Vpp*** используется при программировании ПЗУ или ППЗУ. Напряжение программирования (+14.75В) подается на этот вывод, когда программируется ППЗУ. В других случаях на этот вывод подается напряжение питания.

ПРЕДУПРЕЖДЕНИЕ: не допускается соединять Vpp c Vss. Это приведет к повреждению ОЭВМ.

Вход внешнего запроса на прерывание ***IRQ*** функционирует в одном из двух режимов. В первом режиме входная логика срабатывает по отрицательному фронту, во втором-по отрицательному фронту и низкому уровню . Во втором случае как отрицательный, так и низкий уровень, подаваемый на IRQ, будут приводить к прерыванию. При возникновении на входе IRQ отрицательного фронта, возникает запрос прерывания. Когда ОЭВМ заканчивает выполнение текущей команды, проверяется наличие запроса прерывания. Если запрос обнаружен и бит маски прерывания (1-ый бит в регистре флагов) сброшен в ноль, то ОЭВМ переходит на процедуру обработки прерывания. Если выбран режим срабатывания входа IRQ по уровню, то этот вывод необходимо соединить через резистор с напряжением питания +5В для обеспечения возможности операции "монтажное или".

Вывод ***RESET*** является двунаправленным, с низким активным уровнем. При подаче на вывод RESET логического нуля, происходит инициализация ОЭВМ. В случае сбоя, определяемого системами СОР-таймера или диспетчера частоты синхронизации, на выходе RESET устанавливается логический ноль для инициализации внешних устройств, подключенных к ОЭВМ.

Выводы ***OSC1 и OSC2*** используются для подачи тактовой частоты для ОЭВМ. Тактовые сигналы могут формироваться с помощью кварцевого резонатора, или подаваться от внешнего генератора импульсов. Внутренняя тактовая частота получается делением частоты генератора на 2. Схема, показанная на рисунке п2а, рекомендуется в случае использования кварцевого резонатора. Внутренний генератор разработан для работы с кварцевым резонатором или керамическим резонатором с частотой 4MHz. Кварцевый резонатор и дополнительные элементы этой схемы должны быть установлены как можно ближе к соответствующим выводам ОЭВМ для того, чтобы минимизировать искажения сигнала и время запуска генератора. Схема на рисунке п2б рекомендуется, когда используется внешний генератор тактовых импульсов. Тактовые импульсы подаются на вход OSC1, а вывод OSC2 остается незадействованным.

Рис.20а Схема подключения кварцевого резонатора.

Рис.20б Схема подключения внешнего генератора.

Порт A использует выводы ***РА0-РА7*** . Каждый вывод порта А может быть запрограммирован на ввод или вывод информации.

Порт B использует выводы ***РВ0-РВ7***. Каждый из выводов порта В может быть запрограммирован на ввод или вывод информации.

Порт C использует выводы ***РС0-РС7***. Каждый из выводов порта С может быть запрограммирован на ввод или вывод информации.

Порт D использует выводы ***РD0-PD5, PD7***. После включения или перезапуска все выводы порта конфигурируются как входы. Когда используется SPI (последовательный периферийный интерфейс) четыре вывода MISO/PD2, MOSI/PD3, SCK/PD4 и SS/PD5 используются SPI-системой. Когда осуществляется обмен данными по каналу SCI, вывод PD0/RDI используется для приема данных, передача данных осуществляется через SCI, вывод PD1/TDO используется для передачи данных.

## Приложение 2. Типовая схема подключения МС68НС705С8.

На рисунке п3 показана схема включения ОЭВМ, которая может использоваться как базовая для любых применений MC68HC705C8. Способы построения тактового генератора и подключения ОЭВМ к источнику питания, примененные в этой схеме, рекомендуются при разработке устройств на основе микроЭВМ семейства MC68HC05. Обратите внимание, что неиспользованные входы подключаются через резисторы к источнику питающего напряжения +5В.

Рис. п3 Стандартная схема включения MC68HC05705B8.