МИКРОСХЕМА 1881ВЕ1Т

Руководство по эксплуатации

1. Введение.

К1881ВЕ1Т - однокристальный 16-разрядный КМОП RISC микроконтроллер. Микроконтроллер универсален, может использоваться в системах сбора и обработки данных, системах управления, цифровой обработки сигналов. Малая потребляемая мощность, полностью статический КМОП позволяет использовать микроконтроллер в автономных системах с ограниченным энергопотреблением. Архитектура К1881ВЕ1Т (далее МК) соответствует таким универсальным микроконтроллерам как 80С196, 80С186 фирмы Intel, 68300,68HC16 фирмы Motorola, SAB 80C16x фирмы Siemens, наличие аппаратного однотактного умножителя и поддержки операций ЦОС позволяют сравнивать его с процессорами ЦОС TMS320C25 фирмы TI, DSP561xx фирмы Motorola, ADSP21xx фирмы Analog Devices.

1.1. Функции.

- 16-разрядный КМОП RISC.
- Производительность 12,5 млн. команд в секунду на 25 МГц и Vcc=4.5В.
- Адресация 16М слов разделенных команд и данных.
- Целочисленный параллельный 16 разрядный умножитель с 32 разрядным результатом.
- Параллельность до 3-х операций: (арифметика,умножение,память).
- Цикл один или два такта.
- Гарвардская архитектура с параллельным обращением к кодам программ и данных внутренней памяти.
- 4 стека с аппаратной поддержкой смещения рекурсивных параметров .
- Поддержка 32 бит знаковых форматов чисел.
- Последовательное умножение - деление с результатом 38-разрядов.
- Статическая структура памяти, частота синхронизации от 0 мГц.
- Режим ТУРБО для удвоения производительности.
- 4К x 16 бит - ПЗУ программ (маска).
- 128 x 16 бит - ОЗУ программ/данных.
- 128 x 16 бит - ОЗУ данных.
- 16 и уровневая приоритетная система прерываний.
- Вывод селектора регисторов внешних устройств.
- Подключаемый кварцевый резонатор.
- 4-х 16-разрядных.универсальных таймера-счетчика, сторожевой таймер.
- Параллельный интерфейс совмещенныи для команд/данных с многорежимным арбитром и портами 24 бит.адреса, 16 бит.данных, 6 бит синхронизации.
- UART интерфейс с двумя автономными портами.
- SSI интерфейс с двумя автономными полнодуплексными портами.
- I2C интерфейс.
- Возможность построения многопроцессорных комплексов на всех интерфейсах.
- 8 битовый двунаправленный раздельно управляемый порт С .
- 27 разрядный совмещенный выходной параллельный порт А.

2. Архитектура.

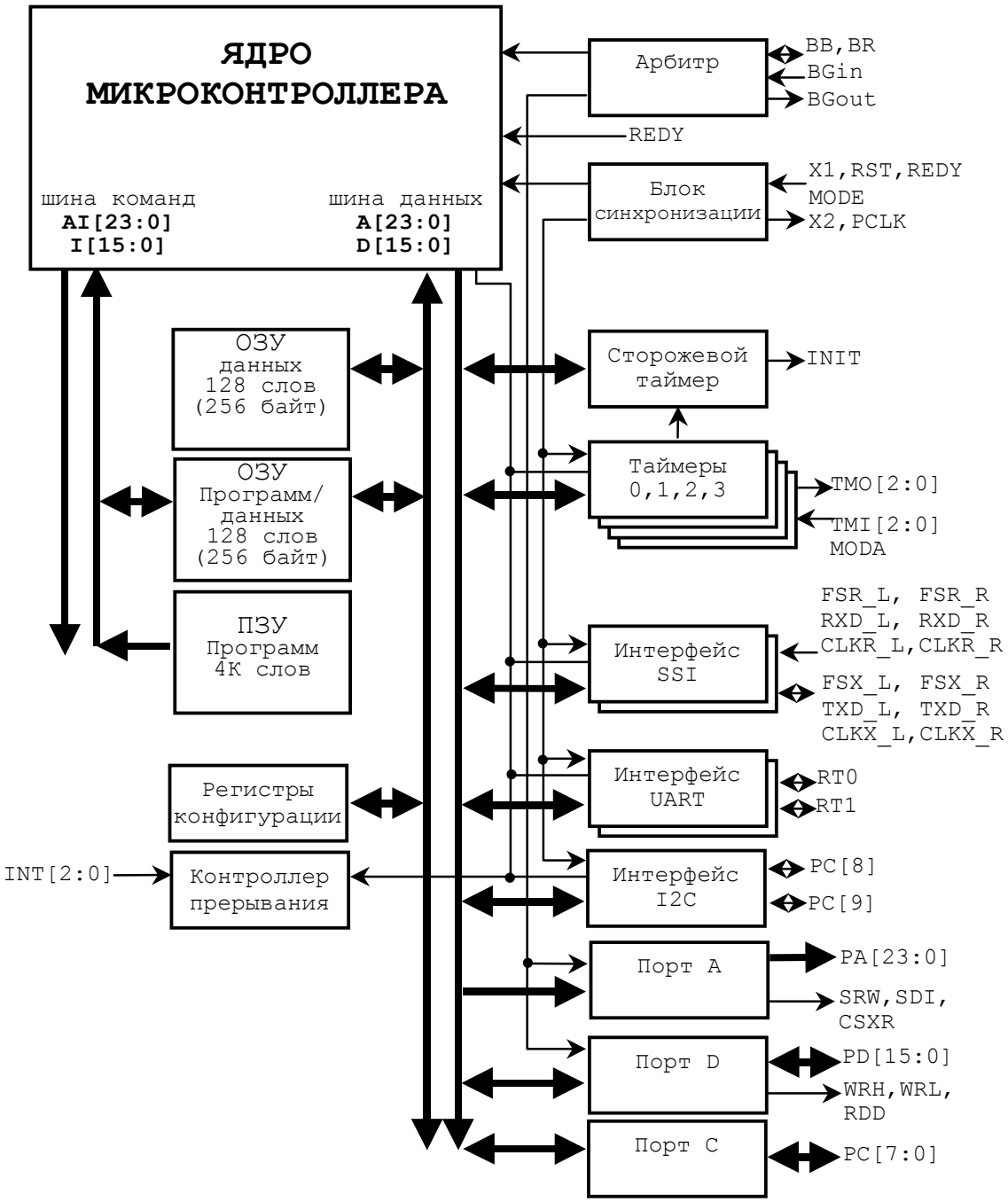


Рис.2.1 Структура микроконтроллера.

2.1. Микроконтроллер.

Структурная схема приведена на рис. 1. Микроконтроллер разработан с соблюдением правил модульности архитектуры по всем уровням, что позволяет получать компоновки кристалла для конкретных применений. В состав базовых блоков микроконтроллера входят:

- микропроцессорное ядро (МП);
- оперативное запоминающее устройство (ОЗУ);
- постоянное запоминающее устройство (ПЗУ);
- система таймеров и сторожевой таймер;
- последовательный асинхронный порт UART;
- последовательный синхронный порт SSI,
- I2C порт,
- двунаправленные битовые порты,
- однонаправленные битовые порты.

В перспективное архитектуры может быть дополнена электрически програмируемой памяти, увеличенными объемами ЗУПВ, блоками повышения производительности процессора и ускоренными аналого цифровыми каналами для связи и локации.

2.2. Микропроцессор.

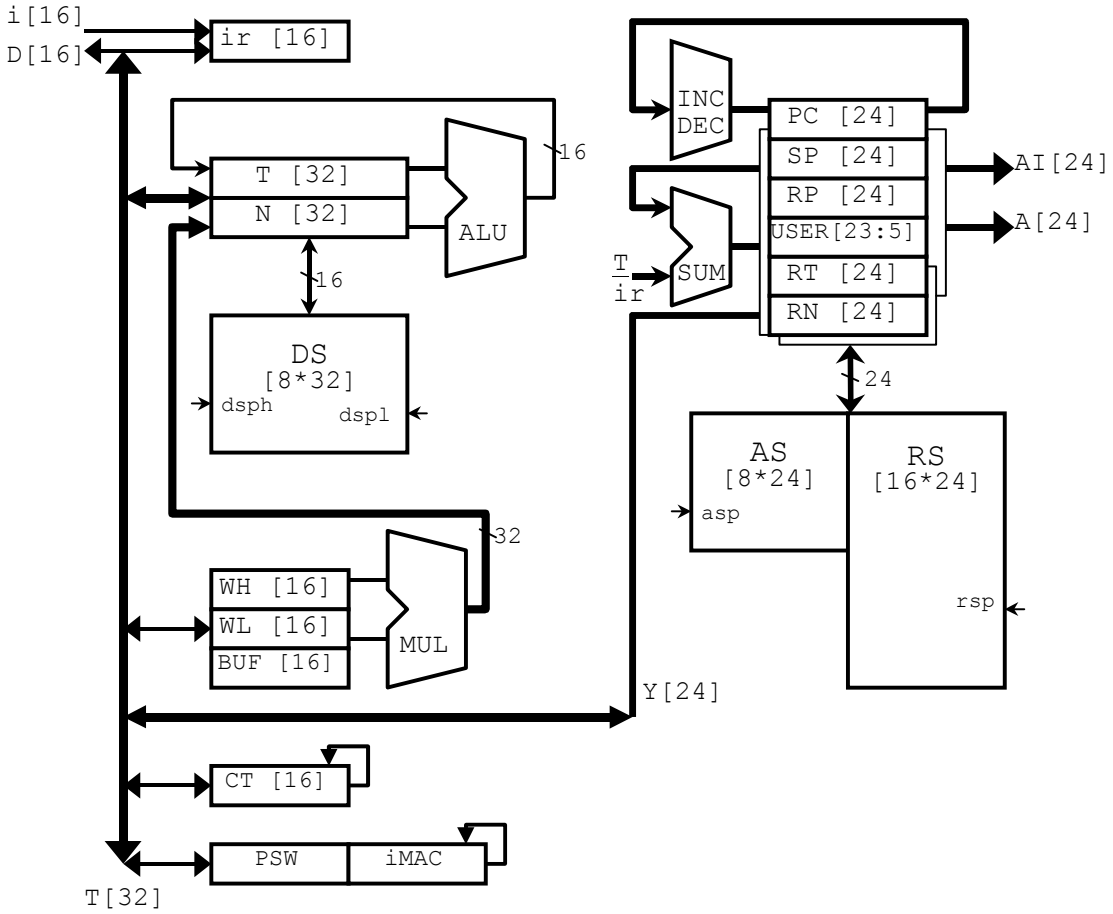


Рис.2.2.1 Структура процессорного ядра.

МП относится к вычислительным устройствам со стековой архитектурой. В таких устройствах обрабатываемые данные находятся на стеке и большинство команд использует в качестве операндов один или два верхних элемента стека. Особенность архитектуры МП состоит в том , что он содержит три стека (данных, возвратов, адресов) которые могут работать параллельно:

стек данных содержит обрабатываемые данные,

стек возвратов содержит адреса возвратов из подпрограмм или адреса переменных,

стек адресов содержит адреса переменных.

Еще одной особенностью архитектуры является наличие двух стеков данных , полностью симметричных т.е. с каждым из стеков возможен полный набор операций. Эта особенность позволяет рассматривать данный МП как 32-разрядный с 16-разрядной внешней шиной данных. Локальные 16 разрядные шины данных и команд нагружены запоминающими ячейками которые сохраняют состояние линии когда отключены драйвера.

Структурная схема приведена на рис.2.

В состав микропроцессора входят следующие блоки:

DS - стек данных состоит из двух блоков организации 8*16 бит соответствующих старшему и младшему слову 32 битовых регистров N и T.

T - 32-разрядный верхний регистр стека данных. Регистр доступен как 16-разрядный(t или %t) либо как 32-разрядный. T является одним из операндов АЛУ.

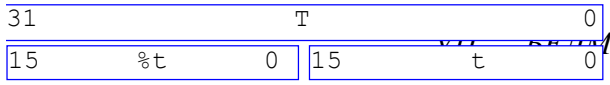
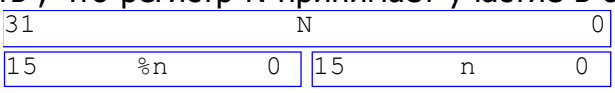


Рис.2.2.2 Формат операционного арифметического регистра T.

N - 32-разрядный следующий регистр стека данных.Он хранит второй сверху элемент стека данных, используется для арифметических операций и является вторым операндом АЛУ. Данные,записываемые в память стека, всегда берутся из регистра N. Данные , считываемые из памяти стека , записываются в N. N может пересылаться в T через АЛУ. При выгрузке в стек, T может непосредственно записываться в N. Операции пересылки данных между T и N , между N и стеком независимы и позволяют выполнить комбинированные действия, что увеличивает эффективность использования стека без дополнительных затрат времени. Необходимо отметить , что регистр N принимает участие в операциях только как 16-



разрядный т.е. **n** или **%n** .
Рис.2.2.3 Формат операционного арифметического регистра N.

RS - стек возвратов. Состоит из внутренней памяти RSм, организованной в виде одного банка регистров 16*24 бит и регистра RT. Все обращения к стеку возвратов осуществляются через регистр RT.

RT - 24-разрядный верхний регистр стека возврата, он хранит верхний элемент стека возвратов. При чтении регистра RT возможно продвижение стека возвратов вверх. При записи в RT стек возвратов может продвигается вниз. Регистр RT используется в командах вызова и возврата из подпрограмм , а также для хранения локальных переменных.

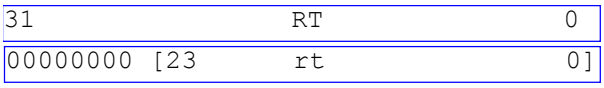


Рис.2.2.4 Формат адресного регистра RT

AS - стек адресов. Состоит из внутренней памяти AS, организованной в виде одного банка регистров 8*24 бит и регистра AT. Все обращения к стеку осуществляются через регистр AT.

AT - 24-разрядный верхний регистр стека адресов, он хранит адреса переменных. При чтении регистра AT возможно продвижение стека возвратов вверх. При записи в AT стек адресов может продвигается вниз.

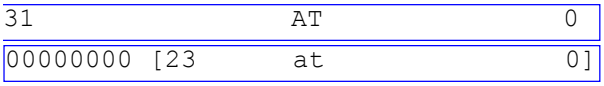


Рис.2.2.5 Формат адресного регистра AT

IR - 16-разрядный регистр команд.
PC - 24-разрядный счетчик команд.

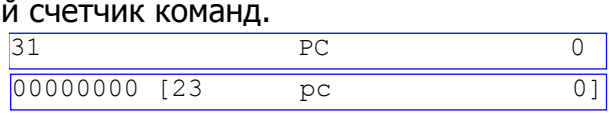
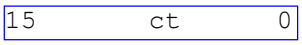


Рис.2.2.6 Формат регистра счетчик команд

CT - 16-разрядный оерационный регистр общего назначения. Особенностью данного регистра является то , что он может выполнять функции счетчика.



Содержимое регистра может анализироваться командами условных переходов, декрементироваться независимо от операции АЛУ. Регистр участвует в циклических командах в качестве счетчика цикла.

Рис.2.2.7 Формат регистра счетчик.

USER - 24-разрадный адресный регистр. Используется в качестве базового адреса для операций с внешними регистрами.

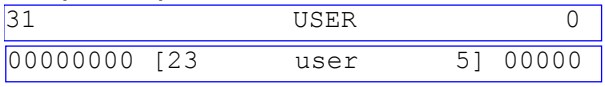


Рис.2.2.8 Формат регистра адресации USER.

SP - 24-разрядный регистр общего назначения. Кроме этого он может содержать адрес и использоваться для адресации внешней памяти.

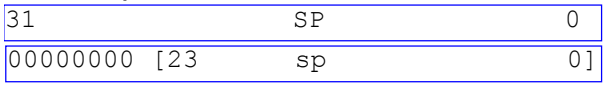
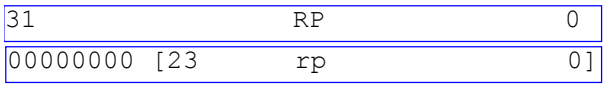


Рис.2.2.9 Формат регистра адресации SP.

RP **-** 24-разрядный регистр общего назначения. Кроме этого он может содержать адрес и



использоваться для адресации внешней памяти.

Рис.2.2.10 Формат буферного регистра адресации RP.

dsph,dspl,asp, rsp - указатели внутренних стеков данных старший, младший, адресов и возврата. Данные регистры программно недоступны.

WORK - 32-разрядный регистр общего назначения. Особенностью данного регистра является то , что он может участвовать в выполнении операций целочисленного умножения и умножения с накоплением.

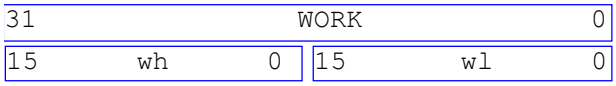


Рис.2.2.11 Формат буферного регистра адресации WORK.

ALU - 16-разрядное арифметико-логическое устройство с умножителем. АЛУ выполняет операцию над двумя операндами (T и N, T и непосредственный операнд) и записывает результат операции в T или N.

MUL - 16-разрядный регистр параллельный знаковый целочисленный умножитель с 32 битовым результатом

PSW - 16-разрядный регистр PSW управляет использованием ресурсов МП , а также отражает состояние некоторых внутренних признаков МП. Он не может быть использован как регистр общего назначения.

iMAC **-**6-разрядный оерационный регистр расширения для операций последовательных умножения деления, суммирует вычитает сигнал переноса ALU.

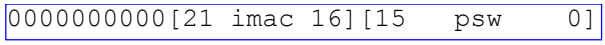


Рис.2.2.12 Формат регистров состояния процессора.

2.3. Память

Микропроцессор адресует 16 битовое слово в адресном пространстве посредством 24 разрядного адреса A[23:0]. Адресное пространство МК состоит из памяти данных и памяти программ. Адресация внутренней памяти данных и программ выполняется параллельно по разделенным локальным шинам. Для обмена с внешней памятью программ или данных задействуется общая шина данных. Внутренние области программной памяти в отличие от внешней не доступны для модификации адресными командами. Внешний сигнал SDI определяет выбираемое адресное пространство (данные/программа).

МК поддерживает возможность обращения к ст/мл байту слова при выполнении операции "запись". В системе команд МП имеются префиксные "BYTE_L", "BYTE_H" которые позволяют разделять сигналы WRL,WRH в следующей команде.

Адресное пространство МП:

данных	*программ*
000000 внутренние регистры 00007F	000000
000080 внешние регистры 0000FF	внешняя память
000100 внутреннее ОЗУ 00017F	FFEF7F
000180 внутреннее ОЗУ (en_RAMI=0) 0001FF	FFEF80 внутреннее ОЗУ (en_RAMI=1) FFEFFF
000200 внешняя память FFFFFF	FFF000 внутреннее ПЗУ (en_ROM=0) FFFFFF

Рис. 2.3.1 - Адресное пространство МП

3. Функционирование.

3.1. Синхронизация.

Для синхронизации микроконтроллера предусмотрены два внешних вывода X1 и X2 к которым может подключаться внешний кварцевый резонатор с ФНЧ. Все процессы в МК тактируются синхросигналом PCLK, который выдается и на вывод микросхемы. Частота PCLK равна половине частоты кварцевого резонатора (в режиме ТУРБО PCLK повторяет X1). В дальнейшем , при описании работы МК, под тактом подразумевается период PCLK.

3.2. Сброс МП.

Сброс устанавливает исходное состояние микроконтроллера. МП фиксируется на выборке команды с адресом FFFFFEh. После снятия сигнала в регистр команд читается слово по адресу FFFFFEh и начинается функционирование.

MODE=1 при сбросе подключит внутреннее ПЗУ (бит en_ROM=0 в регистре конфигурации). Адресация FFFFFEh вызывает чтение команды из внутреннего ПЗУ. В дальнейшем внутреннее ПЗУ можно программно отключить установкой бита en_ROM=1.

MODE=0 в момент сброса выключит ПЗУ (бит en_ROM=1 в регистре конфигурации). Адресация FFFFFEh вызывает чтение команды из внешней памяти программ. Включить внутреннее ПЗУ после этого невозможно.

При подаче питания вырабатывается внутренний сигнал сброса длительностью несколько микросекунд.

Сигнал RST=L сбрасывает 2 триггера задержки тактируемые PCLK и при отсутствии тактов на X1, МК будет находиться в сбросе и после установки RST=H. Вывод сигнала INIT=H
произойдет после второго L фронта PCLK от истечения задержки включения питания или снятия внешнего сброса RST=H.

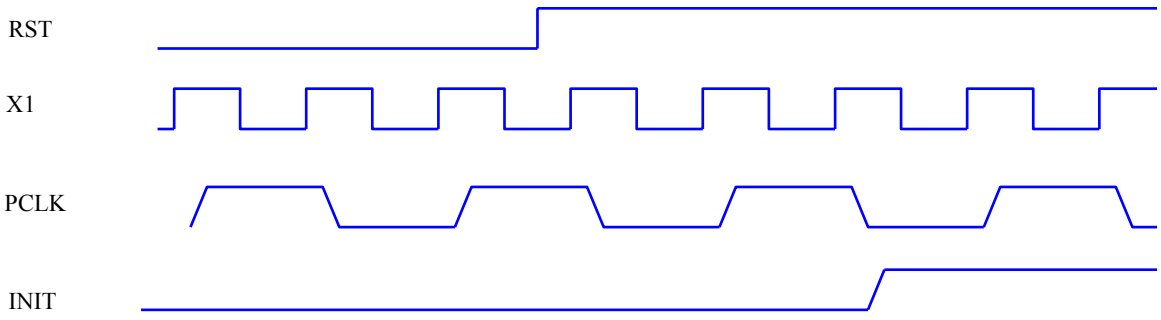


Рис. 3.2.1 - Временная диаграмма пуска микроконтроллера

3.3. Выполнение команды.

По положительному фронту сигнала PCLK в регистр IR записывается команда. Блок управления осуществляет дешифрацию формата команды, определяет код операции ALU и тип стековой операции. По следующему положительному фронту

PCLK в регистр команд записывается новая команда, а в рабочих регистрах микропроцессора фиксируется результат выполненной операции. Если необходим прием операнда из ЗУ, то параллельно с выполнением операции в АЛУ выполняется прием операнда. Гарвардская архитектура позволяет выбирать следующую команду параллельно с обращением к памяти за операндом. Адрес следующей команды берется из одного из трех источников , определяемого текущей выполняемой командой:

- регистра PC для последовательного исполнения,
- регистра RT для выхода из подпрограммы,
- регистра IR для команд с литеральными адресными константами.

Цикл обмена выполняется в течение одного такта PCLK, по положительному фронту сигнала PCLK на шину адреса с некоторой задержкой выставляется адрес , производиться чтение памяти и слово выдается на шину данных. По следующему положительному фронту эти данные защелкиваются во внутреннем регистре и на шине адреса устанавливаются новые сигналы. При записи в память одновременно с выдачей адреса МК устанавливает шину данных. Во втором полупериоде PCLK выполняется запись в память.

3.4. Управление прерываниями.

МК имеет программируемую 16 уровневую систему прерываний. Прерывания могут инициироваться :

а) внешними устройствами посредством сигналов INTx, TMx;
б) внутренними устройствами(таймеры,интерфейсы SSI,UART);
в) программно(команда SWI).

Для эффективного управления обработкой прерываний в МК предусмотрен контроллер прерываний , содержащий:

- регистр запросов прерываний. Он отражает текущее состояние триггеров запросов прерываний от каждого из возможных источников прерываний. Установка запроса на прерывание выполняется только аппаратно. Программно можно выполнить только сброс триггера.
- регистр маски прерываний , позволяющий заблокировать запрос на прерывание от любого источника прерываний.

3.5. Управление тактированием.

Отдельным команды исполняются или могут быть выполнены за несколько тактов PCLK. В случае необходимости повторного обращения к одному устройству исполнение команды автоматически происходит за два такта в первом выполняется обработка данных во втором читается следующая команда. Это требуется для команд с длинным 16 разрядным литералом или косвенной адресацией программной памяти через pc++ или irt++. Исполнение команд внешней памяти использующих адресацию данных требует разделения внутренней шины данных и соответственно двух тактов. Исполнение за один такт команд формата с адресацией возможно только при выборке команд из внутреннего ПЗУ и ОЗУ.

В определенных ситуациях есть возможность программного увеличения длительности такта на такт PCLK. Для команд с параллельным умножением запрос выполняется установкой бита [2] регистра PSW. Расширение при обращении к внешней памяти выполняется при CNFG[5]=0, для чтения из внутреннго ПЗУ при CNFG[6]=0, для чтения по

адресу при CNFG[1]=1 и CNFG[5]=0, для записи в адресную область внутренних регистров при CNFG[2]=0.

Остановка процессора до появления внешнего сигнала выполняется следующими способами. Команда STOP ожидает сигнал сброса от RST=0 или сторожевого таймера. Команда WAIT ”разбудит” процессор после разрешенных прерываний устройств или сигналов внешних прерываний. Сигнал READY=0 остановит исполнение команды выдавшей запрос внешней памяти или с операцией чтения по адресу при CNFG[1]=1. Остановка процессора арбитром внешней адресной магистрали выполняется при появлении запроса и отсутствии разрешающих сигналов устройств на магистрали. Она сопровождается отключением соответствующих выводов адреса PA, порта данных PD, выходов WRH,WRL,RDD,SRW,SDI,CSXR. Исполнение команд нарушающих режим защиты программной памяти остановит процессор в PCLK=0. Режим распостраняется на команды с косвенной адресацией программной памяти через pc++ или irt++ при CNFG[0]=1 и использованием внутреннего ПЗУ программ.

4. Внутренние устройства микроконтроллера.

4.1. ПЗУ.

Внутреннее ПЗУ обьемом 4К слов занимает старшие 4К адресного пространства МК(адреса $fff000-$ffffff). В МК предусмотрен режим защиты внутреннего ПЗУ от несанкцианированного считывания.

4.2. ОЗУ данных.

Внутреннее ОЗУ данных(RAM) обьемом 128 слов находиться в адресном пространстве от $100 до $17F. ОЗУ является статическим, что позволяет сохранять информацию во время выполнения команды WAIT.

4.3. ОЗУ данных/команд.

Внутреннее ОЗУ данных/команд (XRAM) обьемом 128 слов находиться в адресном пространстве от $180 до $1ff. ОЗУ является статическим, что позволяет сохранять информацию во время выполнения команды WAIT. Отличие XRAM от RAM состоит в том, что функции XRAM определяются битом en_RAMI регистра конфигурации. Если бит =0 то XRAM есть память данных в адресном пространстве от $180 до $1ff и по функциям эквивалентно RAM. Если бит en_RAM=1 то XRAM есть внутренняя память программ в адресном пространстве от $ffef80 до $ffefff. В этом случае область данных от $180 до $1ff есть внешняя память данных.

4.4. Внутренние регистры.

Внутренние регистры микроконтроллера находятся в адресном пространстве с 0 по $7f. Их назначение

00 - данные порта C
01 - направление порта C
интерфейс I2C
02 - управление
03 - данные
04 - управление CTRL
05 - порт PINS

06 - порт A
07 - порт AX
интерфейс SSI_R
08 - данные
09 - управление
интерфейс SSI_L
0a - данные
0b - управление
интерфейс UART_0
0c - управление
0d - данные
интерфейс UART_1
0e - управление
0f - данные
таймеры
10 - данные tmr0
11 - управление ctm0
12 - данные tmr1
13 - управление ctm1
14 - данные tmr2
15 - управление ctm2
16 - данные tmr3
17 - управление ctm3
контроллер прерываний
18 - маскирование MASK
19 - запросы прерываний FLAG
регистры-защелки таймеров
1a - cct0
1b - cct1
1c - cct2
1d - cct3
служебные регистры
1e - регистр конфигурации CNFG
1f - управление сторожевым таймером CL_WT

20 - 7f резервные

4.4.1. Регистр конфигурации CNFG.

Таблица 4.4.1.1 - Формат регистра конфигурации

Разряд	Название	Назначение	сброс
15	MDA	0 - отсутствие внешней памяти 1 - наличие внешней памяти	~mode
14	en_ROM	0 - наличие внутреннего ПЗУ 1 - отсутствие	~mode
13	RXB	0 - отсутствие 1 - запрос внешней шины у арбитра	~mode
12		не использованный триггер	0
11:10	HA_h, HA_l	управляют разрядностью шины адреса HL address port 00 - A[23:0] - 01 - A[15:0] A23-A16 10 - A[7:0] A23-A8 11 - - A23-A0,SRW,IDS,cs_XR	0
9	en_RAMI	0- ОЗУ $180-$1ff есть память данных 1- ОЗУ $180-$1ff есть память программ в адресном пространстве $ffef80-$ffefff, область $180-$1ff есть внешняя память данных	0
8	SMOD	0 - биты адреса 23-6 вектора прерывания равны нулю 1 - -"- единице	0
7	lock_T0	1 - блокировка таймера 0 по записи 0 - нет	0
6	ws_ROM	0 - один такт ожидания при обращении к внутр. ПЗУ 1 - нет	0
5	ws_XM	0 - один такт ожидания при обращении к внешней памяти 1 - нет	0
4	stop_mode		0
3	WOR	1 - выход порта С работает в режиме "активный ноль-активная единица" 0 - выход порта С работает в режиме "активный ноль-пассивная единица"	0
2	ws_RG	0 - один такт ожидания при обращении к внутренним регистрам 1 - нет	0
1	debug	1 - режим отладки 0 - рабочий режим	0
0	secur	1 - режим защиты внутреннего ПЗУ 0 - нет	1

4.4.2. Регистр управления CTRL.

Таблица 4.4.2.1 - Формат регистра управления синхронизацией

Разряд	Название	Назначение
15:14	CLK_SCI	синхронизация I2C, 00 - CLK/4 01 - /8 10 - /16 11 - таймер1
13:11	TMI[2:0]	управление полярностью сигнала 1 – инверсия,прерывание по фронту переключения в 0.
10:8	INT[2:0]	управление полярностью 1 – инверсия,прерывание по фронту в 0.
7:6	CLK_TM	синхронизация таймеров, 00 - CLK/2 01 - /4 10 - /8 11 - /16
5:4	CLK_SCI	синхронизация UART , 00 - CLK/4 01 - /8 10 - /16 11 - таймер3
3:2	CLK_SSI	синхронизация SSI , 00 - CLK/4 01 - /8 10 - /16 11 - таймер3
1	on_WTR	включение сторожевого таймера 1 - вкл. 0 - выкл.
0	TURBO	TURBO режим 1 - CLK=X1 0 - CLK=X1/2.

При сбросе все биты регистра сбрасываются в ноль. Биты 15,14,7-0 регистра могут быть заблокированы по записи (см.раздел 4.6 описания таймеров). Индексация таймеров: таймер0, таймер1, таймер2, таймер3. Сигналы таймеров 1 и 3 не могут маскироватся битами dpin.

4.4.3. Регистр запросов прерываний FLAG.

Регистр отражает запросы на прерывание со стороны внешних и внутренних устройств. Сброс не влияет на содержимое регистра. Установка определенного бита осуществляется только аппаратно. Сброс бита осуществляется программно посредством записи в него "1". Назначение бит регистра

Таблица 4.4.3.1 - Формат регистра установок флагов прерывания

Разряд	Название	Назначение	сброс
15	f_TMR0	запрос прерывания от таймера 0	x
14	f_TMR1	запрос прерывания от таймера 1	x
13	f_TMR2	запрос прерывания от таймера 2	x
12	f_TMR3	запрос прерывания от таймера 3	x
11	f_TM0	запрос прерывания по входу TM0	x
10	f_TM1	запрос прерывания по входу TM1	x
9	f_TM2	запрос прерывания по входу TM2	x
8	f_INT0	запрос прерывания по входу INT0	x
7	f_INT1	запрос прерывания по входу INT1	x
6	f_INT2	запрос прерывания по входу INT2	x
5	f_SSI_R_R	запрос прерывания от приемника интерфейса SSI_R	x
4	f_SSI_R_T	запрос прерывания от передатчика интерфейса SSI_R	x
3	f_SSI_L_R	запрос прерывания от приемника интерфейса SSI_L	x
2	f_SSI_L_T	запрос прерывания от передатчика интерфейса SSI_L	x
1	f_UART,I2C	запрос прерывания от UART,I2C	x
0	f_SWI	программное прерывание	x

4.4.4. Регистр маскирования запросов прерываний MASK.

Биты регистра вызывают маскирование (запрет/разрешение) соответствующих битов регистра запросов прерываний. При сбросе регистр очищается.

4.4.5. Регистр PINS.

Регистр PINS, доступен по чтению. Он позволяет читать состояния определенных выводов и внутренних признаков. Назначение разрядов приведены в таблице.

Таблица 4.4.5.1 - Формат регистра состояния выводов

Разряд	обозначение	назначение
15	master	режим арбитра МК 0 – master, 1 - slave
14	BGin	вход BGin
13	TMI2	вход TMI2
12	TMI1	вход TMI1
11	TMI0	вход TMI0
10	INT2	вход INT2
9	INT1	вход INT1
8	INT0	вход INT0
7	RXD_R	вход RXD_R
6	FSR_R	вход FSR_R
5	CLKR_R	вход CLKR_R
4	RXD_L	вход RXD_L
3	FSR_L	вход FSR_L
2	CLKR_L	вход CLKR_L
1	-	отключен
0	-	отключен

Указанные вывода МК через регистр PINS имеют альтернативное применение в качестве входного порта.Вход BGIN арбитра при централизованном режиме работы (см. п.4.9.) не используется и может быть входом общего назначения.

Регистр доступен только по чтению. Попытка записи по адресу регистра не оказывает влияния на работу МК.

4.4.6. Порт А.

27-разрядный параллельный программируемый выходной порт на выводах с двойным назначением. Используется для вывода адреса обращения к внешней памяти или вектора данных при записи по адресу 07 или 06 с сохранением в буферном регистре порта. Регистры доступны для чтения.

Младшие 16 бит порта выводят данные при записи по адресу 07 (*PA*), старшие разряды по адресу 06 (*PAX*). Функции порта зависят от режима работы МК. При работе с внешней памятью для адреса могут использоватся ограниченные поля порта сответственно функциям бит CNFG[11:10]. Незадействованные поля для адреса могут использоватся для вывода данных и не блокируются арбитром при запрете магистрали. *PA* и *PAX* типовые буферные регистры с программным чтением и записью, и возможностью вывода сигнала на контакты. При сбросе разряды регистров устанавливаются в единицу .

МК адресует до 16 Мбайт внешней памяти. В ряде применений с меньшей памятью остаются незадействованными старшие разряды адресной шины. Для их использования адресная шина масштабируется. Разряды CNFG[11:10] управляют разрядностью шины адреса a[23:0].

Таблица 4.4.6.1 - Упрвление конфигурацией адресного порта

Разряд порта A	Функция в зависимости от CNFG[11:10]			
	00	01	10	11
26	SRW	SRW	SRW	pax2
25	cs_XR	cs_XR	cs_XR	pax1
24	SDI	SDI	SDI	pax0
23	a23	pax15	pax15	pax15
22	a22	pax14	pax14	pax14
21	a21	pax13	pax13	pax13
20	a20	pax12	pax12	pax12
19	a19	pax11	pax11	pax11
18	a18	pax10	pax10	pax10
17	a17	pax9	pax9	pax9
16	a16	pax8	pax8	pax8
15	a15	a15	pa15	pa15
14	a14	a14	pa14	pa14
13	a13	a13	pa13	pa13
12	a12	a12	pa12	pa12
11	a11	a11	pa11	pa11
10	a10	a10	pa10	pa10
9	a9	a9	pa9	pa9
8	a8	a8	pa8	pa8
7	a7	a7	a7	pa7
6	a6	a6	a6	pa6
5	a5	a5	a5	pa5
4	a4	a4	a4	pa4
3	a3	a3	a3	pa3
2	a2	a2	a2	pa2

1	a1	a1	a1	pa1
0	a0	a0	a0	pa0

4.5 Асинхронный последовательный интерфейс UART

МК может выполнять обмен данными по однопроводной линии с протоколом UART. Посылка обмена состоит из:

- одного СТАРТ-бита;
- восьми или девяти бит данных (программируется);
- одного СТОП-бита. (см.рисунке 1.5).

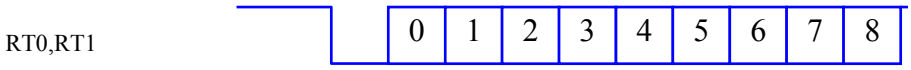


Рисунок .4.5.1 - Диаграмма сигнала UART интерфейса

Полнодуплексный режим обмена данными для одновременной передачи и приема реализован одновременной работе двух одинаковых блоков UART_0 UART_1 и выводов RT0 и RT1. Управляя коммутатором входа можно подключить приемник к любой линии. UART_0 передает по линии RT0, прием возможен по RT0 или RT1, UART_1 передает по RT1, а принимает по RT1 либо RT0. Сетевые функции работы с группой устройств подключенных к линии реализуется режимом “пробуждения” для приема только посылок с адресом содержащим 1 бите а[8] “бит А”.

Допуска рассогласование частот приемного и передающего устройств зависит от разрядности посылки для 8 бит превышение допустимо на 4,16% уменьшение до 5,5%, при обмене 9 разрядным словом допустимы превышение на 3,75% и уменьшение на 5%.

Вывода RT0, RT1 всегда находятся в одном из двух режимов: 1- передача с полными токами драйверов (установки CR[6]”TE”=1 -передача и CR[1]”WOR”=1 -сильная 1); 2-“резистивная 1”формируемая малым током (установки CR[6]”TE”=0 или CR[1]”WOR”=0).

Скорость обмена определяется битами регистра CTRL[5:4] которые подключают источник тактировния: интерфейса к выходам таймера 3 или деленных частот процессора. Длительность такта для обмена битом соответствует 32 периодам частоты интерфейса.

Программное управление контроллером интерфейса выполняется через регистры состояния CR (адрес 0C для UART_0 и 0E для UART_1), данных RD (адрес 0D для UART_0 и 0F для UART_1) и прерыванием последовательного интерфейса.

Таблица 4.5.1. - Назначение бит регистра управления интерфейсом UART

Разряд	Имя	Назначение	После сброса
15	FIRQ	Флаг – импульс запроса прерывания выдан	x
14:8		Отсутствуют	
7	M	Разрядность посылки: 0 - 8бит, 1 - 9бит.	0
6	TE	Включить передатчик: 1 - вкл	0
5	RE	Включить приемник: 1 - вкл	0
4:3		Не используется	
2	STR	Коммутировать приемник: 0–к RT0,1-RT1 для UART_0, 1–к RT0,0-RT1 для UART_1	0
1	WOR	Режим драйвера передатчика: 0 – малый ток формирования 1, 1 - полным током "активная 1"	0
0	RWU	1-режим "пробуждения" приемника включен	0

Запись в регистр данных выполняется только в режиме передачи. Сдвигатель принимает для вывода значение регистра данных , только после завершения вывода предыдущего слова или сброса сдвигателя (TE=RE=0), пересылать в RD следующее для передачи слово следует после загрузки сдвигателя. Принятая по интерфейсу посылка сохраняется в регистре данных до окончания следующего приема, так как показано в таблице 1.7.

Таблица 4.5.2 - Формат регистра данных интерфейсаUART

	Разряды											Назначение	Примеч.
	15	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**1**	**1**	**0**		
1	0	0r	1r	d7	d6	d5	d4	d3	d2	d1	d0	Вывод 8р.слова	1
2	0	1r	d8	d7	d6	d5	d4	d3	d2	d1	d0	Вывод 9р.слова	2
3	0	1r	A	d7	d6	d5	d4	d3	d2	d1	d0	Вывод 9р.слова с битом пробуждения А	3
4	1	M9	d8	d7	d6	d5	d4	d3	d2	d1	d0	Чтение принятого слова	4

Примечания
Последовательность вывода бит:15,0,1,2,3,4,5,6,7,8 – для 8р.посылки
15,0,1,2,3,4,5,6,7,8,9 – для 9 разрядной;
драйвера и входы на линиях d[14:10] отсутствуют.
1) разр.9 8 в записи не участвуют значения формируются автоматически;
2) разр.9 в записи не участвуют значения формируются автоматически;
3) бит А имеет значение при выводе слова для устройств находящихся в режиме пробуждения;
4) M9 соответствует значению только при вводе в режиме 9 разрядного слова.

По окончании операции генерируется импульс для запроса прерывания МК от последовательного интерфейса который поднимает флаг CR[15]=1. Автоматический сброс выполняется после записи, чтения регистра данных или записи в регистр состояния.

Для передачи информации необходимо записать в биты регистра управления:

TE=1, RE=0, режим передачи включен;
M=0(8бит)/1(9бит);
STR=x, RWU=x, незначимы для передачи;
WOR=0/1, ,вывод 1 при WOR=1 выполняется значительно быстрее.

По адресу регистра данных интерфейса записать передаваемую информацию. Причем биты передаваемых данных:

15 - старт_бит;
14:9 - не используются;
8 - бит данных при 9-бит пересылке;
7:0 - биты данных.

После вывода посылки рекомендуется выполнить операцию запись в регистр состояния и загрузить в регистр данных следующее слово.

Для приема информации необходимо записать в биты регистра управления:

TE=0; RE=1; режим приема включен;
M=0(8бит)/1(9бит);
STR=0(RT0)/1(RT1) для UART_0 или для UART_1 0(RT1)/1(RT0);
RWU=0; при RWU=1 – режим пробуждения: прием только слов с флагом 1 в 8 разряде, после чего сброс RWU=0 автоматически;
WOR=x, не значим для приема.

По адресу регистра данных интерфейса считать принятую информацию. Причем биты считываемых данных:

15 - стоп бит =1;
14:10 - не используются;
9:8 - биты данных при 9-бит приеме;
7:0 - биты данных.

Перед приемом следующего слова следует очистить сдвигатель, выполнив запись в регистр управления TE=0,RE=0.

Поледовательные интерфейсы используют один вектор прерывания. Для определения активного интерфейса, следует анализировать бит флага запроса FIRQ в соотвествующем регистре управления. Сбрасывать бит запроса в регистре флагов прерывания следует после обработки запросов от всех интерфейсов.

4.6 Блок таймеров

МК имеет четыре 16-разрядных таймера и 2 разрядный сторожевой таймер WTR. Все четыре таймера [3:0] эквивалентны. Каждый имеет:

регистр управления CTMi,
буферный регистр TMRi,
счетный регистр TMRi,
внешний вход TMI,
внешний выход TMO (кроме 3-го таймера),
регистр защелку CCi.

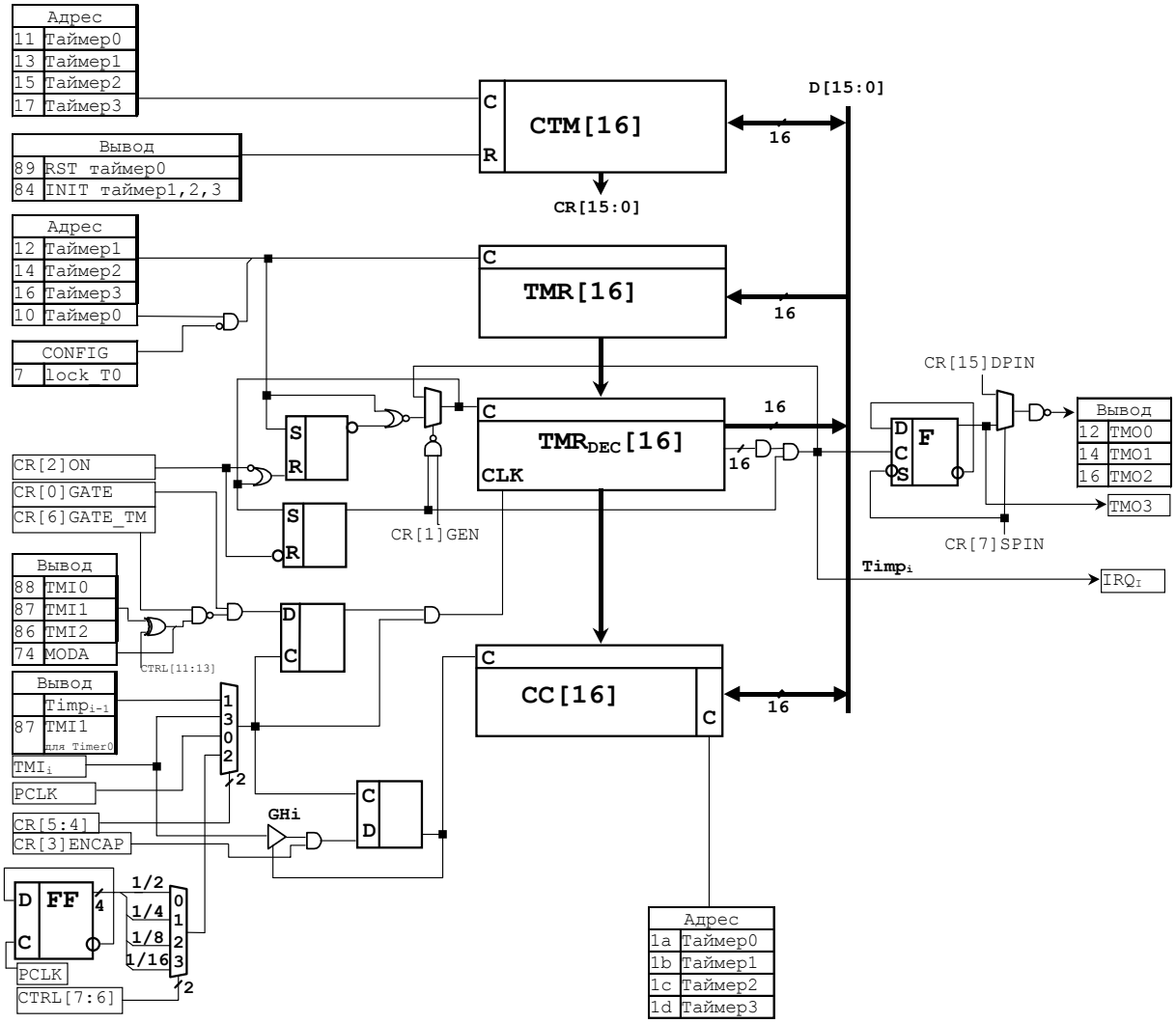
Каждый из таймеров по сигналу синхронизации осуществляет вычитание единицы из содержимого счетного регистра. При вычитании из нулевого значения таймер вырабатывает импульс Tout в половину периода частоты синхроизации. Он используется для прерывания (по обратному фронту) и тактовых частот последовательных интерфейсов. Частота импульса таймера делится на два перед выводом из МК. Программное конфигурирование позволяет соединять таймера подключая сигнал предыдущего, использовать для синхронизации вывод МК с модификацией инверсии, делать паузы и запоминать текущее значение в защелке используя внешний сигнал, автоматическую регенерацию начального значения счетчика. Формат регистра управления таймером показан в таблице 1.8.

Таблица 4.6.1 - Формат регистра управления таймером

Разряд	Название	Назначение разрядов регистра управления CTM
15	dpin	При spin=0 внешний выход TMO=dpin
14:8		Триггера общего назначения не используемые для управления
7	spin	Управление выдачей на внешний контакт TMO: 0 – dpin, 1 - Tout/2
6	gate_TM	Если бит =1 вход TMI используется для запрещения счета
5:4	scnt	Выбор источника тактирования таймера 00 - сигнал синхронизации PCLK. 01 – импульс Tout(i-1),(TMI1 для 0-го таймера) 10 – CLK_TM (см. рег. CTRL) делители частоты PCLK 11 – вход TMI
3	en_cap	1 – вход TMI используется для сохранения значения таймера в регистре CCi в момент 1 фронта модифицированного сигнала TMI .
2	on/off	вкл(1)/выкл(0) – блокирование счетчика для счета и записи
1	gen	Режим работы 0 - при достижении значения "ноль" таймер вырабатывает сигнал Tout и продолжает счет с максимального значения. При записи нового значения в буферный регистр, по следующему фронту синхросигнала новое значение переписывается в счетный регистр. 1 - при достижении значения "ноль" таймер вырабатывает сигнал Tout и выполняет загрузку нового значения из буферного регистра

0	gate	1 – разрешение продолжитьсчет, 0 – пауза

Функциональная схема для управления таймерами приведена на рисунке4.6.1



Вывод	Наименование
Адрес	Адрес в команде обращения к памяти
CTM	Регистр управления таймером
TMR	Регистр данных таймера
TMR_{DEC}	Регистр счетчика данных таймера
CC	Регистр защелка таймера
CTRL	Регистр управления синхронизацией МК
CONFIG	Регистр конфигурации МК
PCLK	Сигнал синхронизации процессора
GHi	Импульсный формирователь на Н фронт с синхронизированным сбросом

Рисунок 4.6.1 - Схема управления таймером

При записи данных для счета, данные попадают в буферный регистр. При чтении таймера, данные считываются из счетного регистра таймера. При чтении счетного регистра

могут возникнуть проблемы, связанные с возможностью считывания изменяемых данных. Для корректного считывания , при необходимости, можно сначала остановить счет (gate=0), а после считывания возобновить (gate=1). Импульс Tout вызывает установку соответствующего флага запроса прерывания.

В системе таймеров есть особенности. Так 0-ой таймер использоваться для тактирования сторожевого таймера. 3 и 1-й таймера может использоваться для синхронизации работы последовательных интерфейсов. В регистре конфигурации имеется бит для блокировки доступа к 0-му таймеру по записи. При сбросе регистры управления таймеров очищаются. Имеется еще одна особенность для таймера 0 - регистр управления CTM0 может быть сброшен только внешним сигналом. Сброс от сторожевого таймера не влияет на 0-й таймер. Сторожевой таймер представляет собой 2-бит счетчик тактируемый сигналом Tout 0-го таймера. При переполнении сторожевого таймера вырабатывается внутренний сигнал сброса. Пользователь имеет только косвенный доступ к сторожевому таймеру с возможностью его сброса.

Для сброса WTR в МК предусмотрен фиктивный регистр CL_WT. Для выполнения процедуры сброса WTR необходимо

а) записать по адресу CL_WT значение 5555h.

б) прочитать ячейку по адресу CL_WT.

В момент чтения происходит сброс сторожевого таймера. Если по адресу CL_WT записать значение отличное от 5555h, то в момент чтения CL_WT произойдет блокировка доступа по записи к битам регистра CNFG[15:0], CTRL[15,14] и CTRL[7:0]. Разблокирование выполнится только сигналом INIT=0 сброса МК.

4.6.1 Внешние сигналы управления таймером

Тракт сигналов внешнего тактирования таймеров:

TMI[2:0]-INV-XOR(CTRL[i])-INV-> TMI_in[2:0]

TMI[3] -INV----------> TMI_in[3]

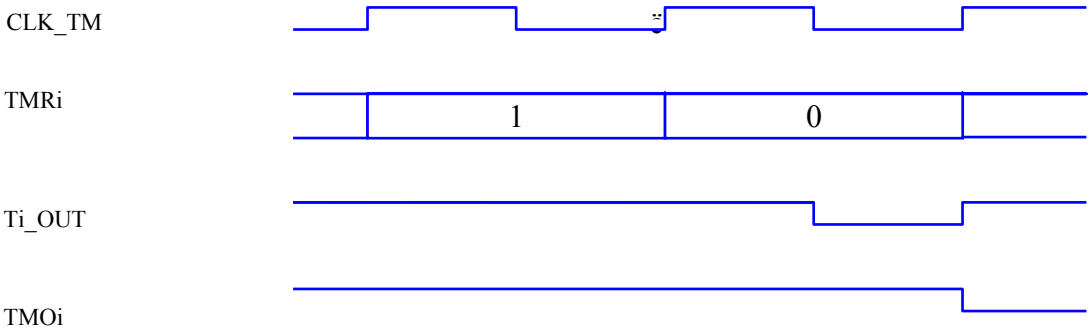
Временные диаграммы таймеров приведены на рисунке 1.7 .

Далее в описании, при упоминании сигналов TMI , следует иметь ввиду внутренний модифицированный сигнал TMI_in. Для TMI[2:0] это верно при CTRL[i] =0.

а) связь между битами 0 (gate) и 6 (gate_TM) регистра CTMi.

gate	gate_TM	TMI_in	счет
0	-	-	нет
1	0	-	да
1	1	0	да
1	1	1	нет

б) выходной сигнал TMO



запрос прерывания от входов TMI.

использование регистра CCi и входа TMIi в режиме "capture".

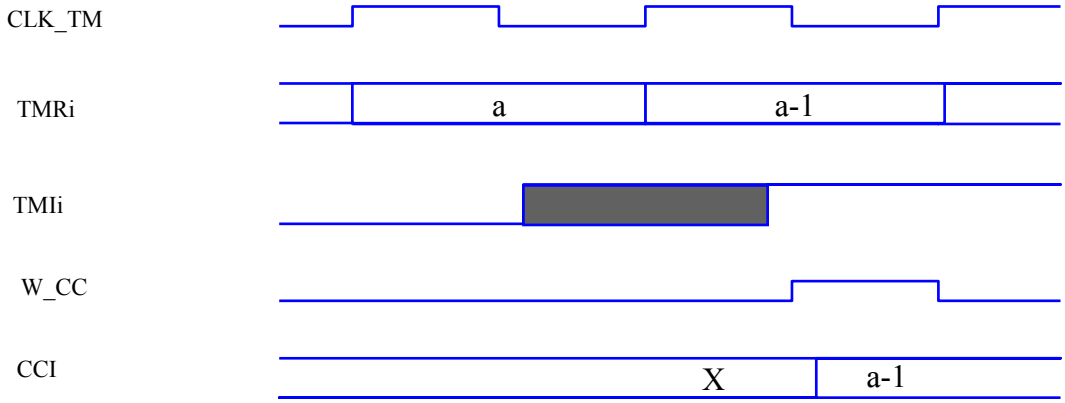


Рисунок 4.6.1.1 Диаграмма синхронизации таймеров

4.7 8-разрядный битовый порт С

Порт С - 8-разрядный универсальный битово управляемый порт ввода-вывода. Управление портом осуществляется регистром направления (*pcr*), вывод данных выполняется из регистра данных (pcd). Каждый бит может выполнять программно специфицированную функцию приема или выдачи сигнала соответствующего разряда порта. Если бит регистра направления (*pcr*) в состоянии единица, то выполняется вывод бита регистра данных на контактную площадку порта. Формат уровней выходных токов определяется битом регистра CNFG[3], (1-активный 0, активная 1, 0- активный 0, резистивная 1). При чтении регистра данных выгрузится значение регистра на d[7:0] и состояния выводов mode,moda,modb на d[15:13]. При чтении регистра данных в состоянии 0 бита регистра *pcr* на d[7:0] принимаются состояния соответствующих внешних выводов порта PC и сответственно mode, moda, modb на d[15:13]. Запись в регистры данных и напрвления выполнится всегда. При чтении регистров порта в неиспользуемых разрядах d[12:8] будет находиться предыдущее состояние локальной шины данных d. При сбросе регистр PCR принимает значение 0, значение регистра данных не изменяется. Формат регистра даных бит порта С показан в таблице 3.7.1.

Таблица 4.7.1 - Формат регистра даных бит порта С

N п/п	Разряд порта	Регистр данных	Регистр направления	Примечание
15	-	mode	-	Бит mode регистра данных доступен только по чтению
14	-	moda	-	Бит TMI3 регистра данных доступен только по чтению
13	-	modb	-	Бит modb регистра данных доступен только по чтению
12	-	-	-	
11	-	-	-	
10	-	-	-	
9				
8				
7	pc7	pcd7	pcr7	
6	pc6	pcd6	pcr6	
5	pc5	pcd5	pcr5	
4	pc4	pcd4	pcr4	
3	pc3	pcd3	pcr3	
2	pc2	pcd2	pcr2	
1	pc1	pcd1	pcr1	
0	pc0	pcd0	pcr0	

Биты регистра направления доступны по чтению и записи. Биты 0-9 регистра данных доступны по записи. При чтении регистра данных учитывается значение соответствующего бита регистра направления.

Если бит PCRi равен 1 - при чтении регистра данных считывается соответствующий бит регистра данных PCDi.

Если бит PCRi равен 0 - при чтении регистра данных считывается значение соответствующего внешнего контакта порта PCi.

4.8 Синхронный последовательный интерфейс SSI

Синхронный последовательный интерфейс аналогичен интерфейсу микросхемы TMS320C25. В микроконтроллере размещены два блока интерфейса. Каждый содержит приемную и передающую части с портами приема, передачи, регистрами буфера,сдвига данных. Программное управление блоком выполняется через 16 разрядные регистры управления CR, данных RD и прерываниями с векторами приема и передачи. Прием сигнала выполняется без задержек фильтрации одиночных импульсов, наиболее быстро. Скорость передачи соответствует тактовой частоте CLK источником которой может быть, как внешний, так и внутренний генератор. Универсальность обеспечена управляемой инверсией, возможностью тактирования только при выводе, отключением старт сигналов, 8 или 16 разрядным словом обмена, управлением последовательностью вывода бит.

Канал использует три линии: старт импульса, тактовой синхронизации и сигнала данных. В канале передачи соответственно: FSX – для старт импульса, CLKX - синхронизация, TXD - для данных. Для приема: FSR - для старт импульса, CLKR - синхронизация, RXD - для данных.

Временные диаграммы сигналов интерфейса представлены на рисунке 4.8.1

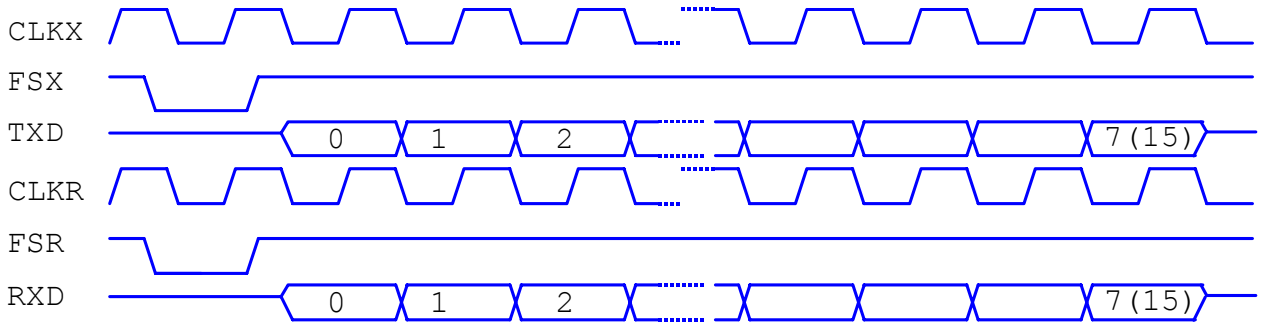


Рисунок .4.8.1 - Диаграмма сигналов SSI интерфейса

При отсутствии передачи на выходных линиях малым током поддерживается логическая 1 резистивного уровня.

В режиме формирования тактового синхросигнала управление частотой выполняется через соответствующие биты регистра CTRL. Коммутатором можно подключать сигналы деления PCLK:1/4, 1/8, 1/16, или выход таймера 3.

Состояние регистра буфера данных сбросом не определяется.

4.8.1 Регистр управления

Регистр управления - CSI определяет режим приема и передачи. Назначение бит регистра управлеия интерфейсом SSI показано в таблице 4.8.1.1.

Таблица 4.8.1.1 - Назначение бит регистра управлеия интерфейсом SSI

Разряд	Имя	Назначение
15	TE	Разрешить работу передатчика: 1 – вкл, 0 - откл.
14	CLKXD	Формировать сигнал тактирования передачи: 0 – использовать внешний сигнал синхронизации CLKX, 1 – выводить сигнал внутреннего формирования (CLK_SSI).
13	FSXD	Формировать старт сигнал по 1-й записи в регистр данных: 0 – передача запускается внешним импульсом с линии FSX. 1 – на FSX выдается строб внутреннего формирования.
12	X_RL	Формат передаваемой посылки: 1 - первым выдается младший 0-й бит, 0 – первым выдается старший бит.
11	FSI	Инвертировать строб FSX: 0 – импульс пуска 0 (инвертированный строб FSX), 1 – импульс 1 (нет инверсии).
10	CLI	Инверсия синхросигнала CLKX: 0 – выдача данных по фронту перехода в 0 (инверсия синхросигнала), 1 – по фронту в 1 (нет инверсии).
9:8		Триггера общего назначения
7	RE	Разрешить работу приемника: 1 – вкл, 0 - откл.
6	CLKR_S	Выбор линии тактирования приемника: 0 – линия CLKR, 1 – канал тактирования передатчика: с линии CLKX при CR[14] CLKXD=0 иначе от CLK_SSI - внутреннего источника.
5	FSR_S	Выбор линии старт сигнала приема: 0 – линия FSR, 1 – линия FSX, модификатор инверсии CR[11]FSI действует на обе линии.
4	R_RL	Формат принимаемой посылки: 0 – первым принимается старший бит 1 – первым принимается младший бит
3	p_GCLK	Инверсия внутреннего сигнала GCLK на выход CLKX: 0 –инверсия GCLK, 1 – нет инверсии.
2	en_GCLK	Включить режим GCLK("gated CLK"):формирование сигналов синхронизации CLKX только в момент передачи данных: 0 – обычный режим работы, 1 – выдача синхросигнала в режиме GCLK
1	FSM	Разрешить использование стартовых сигналов FSX и FSR: 0 – отсутствие сигнала, 1 – импульсная синхронизация по FSX и FSR включена.
0	format	Размер посылки: 0 – обмен с МК выполняется 16 бит данными, 1 – 8 бит.

При включении питания все биты регистра сбрасываются в 0.

4.8.2 Передача данных

Для вывода данных на интерфейс SSI следует:

а) в режиме формирования сигналов тактирования CLK определить скорость работы интерфейса установками в регистре CTRL;

б) установить в регистре управления

размер посылки (format);
источник тактирования (CLKXD=0 - внешний,CLKXD=1 - внутренний);
режим старт синхронизации (FSXD=0 – ожидание внешнего FSX,
FSXD=1 – формировать FSX после 1-й записи данных);
полярности сигналов CLKX,FSX(биты CLI,FSI);
порядок вывода бит (X_RL);
режим FSM=1;
переключится в en_GCLK=0;

в) включить передатчик (TE=1);

г) очистить флаг запроса прерывания от передатчика в регистре запросов прерываний МК и разрешить прерывание при необходимости;

д) записать по адресу регистра данных DSI передаваемые данные. Если выполняется байтовая пересылка - информация должна находится в младшем байте;

е) установка флага запроса прерывания от передатчика указывает на то, что данные переданы из буферного регистра передатчика в сдвиговый регистр для передачи;

ж) очищается флаг запроса прерывания;

з) новые данные(при необходимости) записываются в RD.

Пункты е, ж, з повторяются до окончания передачи.

4.8.3 Прием данных

Для организации приема информации посредством интерфейса SSI следует

а) посредством регистра CTRL определить скорость работы интерфейса, если предполагается внутренний источник тактирования;

б) определить биты

размер посылки (format=0/1);
источник тактирования (CLKR_S=0 - CLKR, CLKR_S=1 - CLKX);
источник старт сигнала (FSR_S=0 – с линии FSR, FSR_S=1 – с FSX);
полярности сигналов CLKR,FSR(биты CLI,FSI);
последовательность приема (бит R_RL=1 –первым принимается 0 бит);
режим FSM=1;

в) очистить флаг запроса прерывания от приемника в регистре запросов прерываний и разрешить прерывание при необходимости;

г) включить приемник (RE=1);

д) установка флага запроса прерывания от приемника указывает на то, что данные переданы из сдвигового регистра приемника в буферный регистр для чтения;

е) прочитать по адресу регистра данных DSI принятые данные. Если выполняется байтовая пересылка – информация будет находиться в младшем байте;

ж) очищается флаг запроса прерывания.

Пункты д, е, ж повторяются до окончания приема.

4.8.4 SPI интерфейс

Интерфейс SSI позволяет выполнить протокол синхронного интерфейса SPI (Motorola) в режиме master (см. рисунок .1.9). Данный интерфейс требует не менее трех линий для обмена

CLK - синхронизация
DI - вход данных
DO - выход данных

Линия MS, определяющая режим работы (master/slave), может отсутствовать (предполагается режим master).

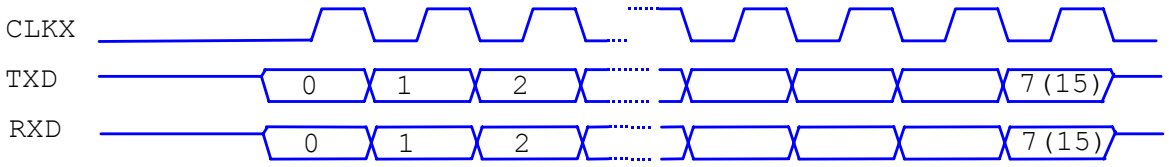


Рисунок .4.8.4.1 - Диаграмма сигналов SSI интерфейса в режиме SPI

Для реализации протокола SPI интерфейсом SSI следует:

а) в регистре CTRL определить скорость работы интерфейса;

б) определить биты:

размер посылки (format=0/1);
CLKXD=1,FSXD=1 - источники сигналов внутренние;
CLI=0,FSI=0 – отключить модификаторы;
формат посылки (биты X_RL,R_RL);
FSM=1;
en_GCLK=1 – режим тактирования только при выводе;
p_GCLK полярность GCLK;
CLKR_S=1, FSR_S=1;

в) включить передатчик и приемник (TE=1,RE=1);

г) очистить флаги запросов прерывания от передатчика и приемника в регистре запросов прерываний и разрешить прерывание при необходимости;

д) записать по адресу регистра данных RD передаваемые данные. Поскольку интерфейс SPI работает в синхронном режиме (одновременно с передачей по линии TXD (DO) идет прием данных по линии RXD (DI)) то в случае только приема эту операцию следует выполнить с незначимыми данными передатчика;

е) установка флага запроса прерывания от передатчика указывает на то, что данные переданы из буферного регистра передатчика, в сдвиговый регистр для передачи. Установка флага запроса прерывания от приемника указывает на то, что данные переданы из сдвигового регистра приемника в буферный регистр для чтения, он же указывает и на окончание передачи данных;

ж) очищаются флаги запросов прерывания;

з) новые данные RD записываются, принимаются.

В реализации интерфейса SPI сигналы CLKR,FSR,FSX не требуются.

4.9 I2C интерфейс

4.9.1 Программное управление

Управление интерфейсом выполняется через регистры: состояния CR (адрес 03) и данных RD (02), прерыванием по последовательным интерфейсам (таблицы 4.9.1.1(.2)) .

Таблица 4.9.1.1

Формат регистра управления контроллера																
Бит	15	-	-	-	-	-	9	8	7	6	5	4	3	2	1	0
Сброс	1	-	-	-	-	-	-	-	-	-	-	-	0	1	0	0
Доступ	R						R	R	R	R	R	R	RW	RW	RW	RW

разряды регистра				
Бит	Обозначение	Сигнал	Назначение	Примечание
0	open	1	Подсоединение линиям SDA SCL	
1	en_WASK	1	Остановка после приема 8 бит 1-го адресного байта	
2	b_ASK	0	Разрешение вывода ответ сигнала SDA=0 после 8 такта и прерывания по приему байта; отсуетствие ответа вызывает прекращение обмена с выводом стоп сигнала.	
3	MASTER	1	Режим контроллера - MASTER (0–SLAVE)	
4	R/W	1	Флаг: направление обмена - чтение	
5	LOAD _WASK	1	Флаг: регистр данных открыт для записи или ожидание окончания анализа адреса	
6	BUSY	1	Флаг: старт сигнал принят выполняю обмен	
7	wait _END	1	Флаг: ожидание сигнала ответа или стоп сигнал	
8	f_ASK	0	Флаг: в CR[9]=1 значением сигнала ответа считается CR[8]	1
9	c_ASK	0	Флаг: анализируется сигнал ответа по SDA	1
15	FIRQ	0	Флаг: запрос прерывания	

Примечания
1 - значения устанавливаются разрядами d[9,8] при записи в RD, для выбора режима окончания:
00 –продолжение, если принят ответ, 10- продолжение независимо от значения ответа,
01-после вывода байта ожидать адрес , 11- после вывода байта стоп сигнал.

Таблица 4.9.1.2 Формат регистра данных

N	Разряды										Назначение	Прим.
	9	8	7	6	5	4	3	2	1	0		
1	c	f	a6	a5	a4	a3	a2	a1	a0	RW	Запись адреса в *master* устройство	1
2				a6	a5	a4	a3	a2	a1	a0	Чтение адреса в *slave*	
3	c	f	d7	d6	d5	d4	d3	d2	d1	d0	Запись байта для вывода	2
4	0	d7	d6	d5	d4	d3	d2	d1	d0	ASK	Чтение принятого байта	3
В процессе обмена по интерфейсу												
5	d_i	...	d2	d1	d0	0	1	1	1	1	Чтение байта в процессе вывода	4
6	1	1	1	1	1	1	1	1	1	0	Регистр после чтения в режиме ожидания чтения принятого байта	4
7	1	1	1	0	d7	d6	d5	d4	...	d_i	Чтение байта в процессе ввода	4
8	d7	d6	d5	d4	d3	d2	d1	d0	0	1	Чтение записанного байта	4

Примечания
1 D[7:1] – адрес абонента, D[0]-бит направления, D[9,8]-биты с f устанавливают режимы f_ACK,c_ACK CR[9,8] регистра управления;
2 D[7:0] – байт для передачи, D[9,8]-биты с f определены в примечании 1;
3 D[8:1] - принятый байт, ACK – значение ответа;
4 Форматы 5,6,7,8 отражают одно из промежуточных состояний в процессе обмена.

4.9.2 Обмен данными в режиме *"master"*

Для работы контроллера в режиме "master" следует:

1) очистить флаг запроса прерывания от последовательных интерфейсов в регистре запросов прерывания МК. В регистре масок разрешить прерывание.

2) установить биты регистра управления

CR[0]"open"=1 - подключить интерфейс к линии,

CR[3]"master=1" - установить режим работы "управлющий",

CR[1]"en_WACK" - значение бита безразлично,

CR[2]"b_ACK" - не существенно при записи в управляемое устройство(RW=0 в адрес байте), при чтении, установка b_ACK=1 после чтения предпоследнего байта вызовет стоп сигнал после ввода последнего байта, установка b_ACK=0 – разрешит вывода сигнала ответа SDA=0 и продолжение обмена.

3) по установке бит, управление можно передать другой программе, ожидать прерывания по интерфейсу. Последующее прерывание подтверждает готовность интерфейса к загрузке в RD адресного байта и выхода сигналов на линию.

4) при обработке прерываний с последовательного интерфейса инициированные контроллером I2C анализируются разряды регистра управления:

Таблица 4.9.2.1 Состояния контроллера I2C при обработке прерывания

Идентификатор	CR[7] "wait/END"	CR[6] "busy"	CR[7] "load/WACK"	Состояние
m1	0	0	1	Готов к началу обмена
m3	0	1	1	Закончен вывод последнего байта, жду загрузки адрес байта
m4	1	0	0	Обмен закончен по стоп сигналу и нет разрешения загрузки в RD (SDA=SCL=1)
m5	1	0	1	Обмен закончен по стоп сигналу, жду загрузки адрес байта
m6	1	1	0	Прием байта закончен, жду чтения данные из RD
m7	1	1	1	вывод байта закончен, жду загрузки новых данных в RD

5) в зависимости от значений выполняется:
на состояниях m1="001" или m3="011" загрузка адресного байта в RD. При обмене адресом ”RW” бит A(0)автоматически переписывается в бит CR[4] регистра управления. После вывода адреса интерфейс приступает к обмену данными.

В состоянии m6="110" - определяется необходимость изменения бита b_ACK=1 для вывода стоп сигнала после приема следующего последнего байта, установку b_ACK=1 следует проводить после чтения предпоследнего байта.

В состоянии m7="111" - при подготовке следующего байта данных для передачи устанавливаются биты d[9],d[8] для управления окончанием обмена:

0,0 - после передачи байта продолжение возможно только при наличии сигнала ответа ACK=0 на линии SDA, иначе выводится "стоп" сигнал,

0,1 - после передачи байта контроллер перевести в состояние ожидание загрузки адресного байта другого абонента,

1,0 - после передачи байта обмен продолжить независимо от фактического значения ответного сигнала на линии SDA,

1,1 - после передачи байта выдать "стоп" сигнал.

Таблица 4.9.2.2 Состояние контроллера I2C после обмена байтом

	CR[9] c_ACK	CR[8] f_ACK	ответ ACK	CR[4] RW	Состояние
v1	0	0	0	0	Жду записи следующего байта для передачи
v2	0	0	0	1	Переключение в режим приема байта данных из "slave", прием байта выдача b_ASK, переход в состояние110" ожидания чтения принятых данных.
v3	0	0	1	-	Выдача "стоп" условия и переход в состояние "100" конца обмена (устройство с переданным адресом не ответило).
v4	1	0	-	0	см. v1
v5	1	0	-	1	см. v2
v6	1	1	-	-	см. v3

После определения необходимых бит слова выполняется запись по адресу регистра данных для передачи.

В состоянии m3="011" ожидается загрузка адресного байта для запроса связи с устройством без вывода стоп сигнала; этому предшествует загрузка в RD слова с d[9,8]=0,1 последним передаваемым байтом и получение ответа.

Состояние m4="100" после прекращение обмена со "стоп" сигналом и отсутствии разрешения записи в регистр данных, которое появляется после регистрации на линиях SDA=1 и SCL=1 и обозначется состоянием m5.

Прекращение обмена может инициироватся

а) управляющим MASTER устройством
после приема очередного байта в состоянии CR[2]"b_ACK"=1;
или при передаче с CR[9,8]=1,1 - выводом "стоп" сигнала;

б) управляемым SLAVE прибором после чтения предпоследнего байта установки состоянии CR[2]"b_ACK"=1 (блокировка вывода сигнала ответа) приема последнего байта.

При m5="101" - контроллер закончил обмен с выдачей "стоп" условия и готов к новому обмену.

4.9.3 Обмен данными в режиме *"slave"*

Для работы в режиме "slave"-управляемого устройства следует:

1) очистить флаг запроса прерывания МК от последовательного интерфейса в регистре запросов прерывания. В регистре масок разрешить прерывание.

2) разрядами регистра управления установить режимы:

CR[0]"open"=1 - подключение интерфейса к линии,

CR[3]"master"=0 - режим работы "управляемый", (в RD с SDA поступают данные, прерывания блокированы до приема старт сигнала);

CR[1]"en_WACK"=1 – включает режим остановки SCL линии на программный анализ 7 адресных бит после прерывания, при en_WACK=0 для продолжения обмена следует успеть определить значение бита b_ACK=0 в течении интервала опроса ACK сигнала с линии SDA. Если МК slave "не успевает", master контроллер выдает стоп сигнал а интерфейс будет формировать прерывания только после следующего "старт" сигнала. После записи en_WACK=0 линия SCL деблокируется с выдачей 0 на линию SDA.

CR[2]"b_ACK"=0 – для продолжения активного обмена данными – вывод ACK ответа SDA=0 и прерываний после приема байта данных. В режиме "slave" бит CR[2] имеет особенность - "старт" сигнал устанавливает его в 1.

3) после установок регистра управления МК может перейти к выполнению другой работы, предварительно разрешив прерывания от интерфейса. В этом случае МК реагировать прерываниями текущей программы на события I2C интерфейса.

4) после прерывания I2C интерфейса МК анализирует состояние регистра управления:

Таблица 4.9.3.1 Состояние контроллера I2C при обработке прерывании. в SLAVE режиме

Иденти-фикатор	CR[7] ”wait/END”	CR[6] ”busy”	CR[5] ”load/ WACK”	Состояние
s2	0	1	0	Приняты 7 бит адресного байта, следует определить бит b_ask
s3	0	1	1	Адресный байт принят,линия блокирована-SCL=0,жду сброса бита CR[1]en_WACK=0
s4	1	0	0	Обмен закончен по стоп сигналу от master
s6	1	1	0	Прием байта закончен, жду чтения данные из RD
s7	1	1	1	Вывод байта закончен, жду загрузки новых данных в RD.

5) в зависимости от значений регистра состояния МК может выполнить следующие действия:

при s2="010" – чтение регистра данных и анализ адреса, если это его адрес запись в регистр управления бит b_ACK=0 и en_WACK=0, если не его запись бит b_ACK=1 и en_WACK=0.

По приему адрес байта интерфейс автоматически переписывает бит A[0]”RW” (с инверсией) принятого байта в бит CR[4]”RW” регистра управления. После приема адреса slave интерфейс должен выполнять следующие действия

Таблица .4.9.3.2 Состояние контроллера I2C после приема адресного байта.

CR[2] b_ACK	CR[4] RW	Действия
0	0	Передача данных master устройству, в состоянии s7="111" запись данных для вывода в регистр RD.
0	1	Прием данных от master, в состоянии "110" необходимо прочитать принятый байт, принятый адрес байт также следует повторно прочесть
1		Контроллер потребует реакции только на следующий адресный байт.

При s3="011" - действия МК аналогичны состоянию "010"(см.выше). Отличие в том, что приняты все 8 бит адресного байта, а не 7. Это следует иметь в виду при анализе введенной информации.

При s6="110" - чтение принятого байта данных и принятие решения для завершения после ввода следующего байта, установка b_ACK=1 блокирует вывод ACK ответа и

стимулирует master устройство к выдаче стоп сигнала. В состоянии s6 интерфейс будет удерживать линию SCL=0 и блокировать продолжение обмена до тех пор пока МК не прочитает принятый байт.

При s7="111" - МК готовит очередной байт данных для передачи, вместе с битами d9 d8, которые могут инициировать прекращение обмена:

Таблица 4.9.3.3 Состояние контроллера I2C после приема байта данных .

d9	d8	**Действия**
0	0	После передачи байта обмен продолжится при наличии ответа SDA=0 от master.
	1	После передачи контроллер перейдет в состояние ожидания адресного байта, независимо от ответа master.
1	0	После вывода байта slave продолжит обмен независимо от ACK ответа master устройства.

После определения всех бит выполняется запись по адресу регистра данных для передачи. До тех пор пока МК не выполнит загрузку регистра контроллер будет удерживать линию SCL=0 и блокировать продолжение обмена.

При s4="100" slave может считать обмен законченным вследствии стоп сигнала. Обмен данными прекращается действиями:

а) master устройства, когда он не выдаст ACK ответ после приема байта, или выведет стоп сигнал независимо от значения ACK сигнала slave устройства.

б) slave устройством при приеме информации, установка b_ACK=1 после чтения предпоследнего байта блокирует вывод ACK ответа.

4.9.4 Прерывание последовательного интерфейса со стороны I2C

I2C интерфейс выдает запрос на прерывание МК при определенных условиях. Они определяются режимом работы. В режиме master интерфейс прерывает МК когда:

а) контроллер не участвует в обмене и I2C шина не занята (SCL=1, SDA=1);
б) он ожидает чтения принятых данных;
в) он ожидает загрузку новых данных для передачи;
г) он закончил обмен данными.

В slave режиме выдается прерывание когда:

а) приняты 7 бит адресного байта;
б) он ожидает чтения введенного байта;
в) он ожидает записи следующего байта для вывода;
г) после прекращения обмена данными.

Причина прерывания устанавливается анализом бит CR[5],CR[6],CR[7] регистра управления, конкретные значения которых определяются выше.
Наличие запроса прерывания от I2C контроллера определяется Флагом CR[15]”fIRQ”=0 в регистре управления. Снятие флага CR[15]=1 автоматическое после выполнения действий обусловленных причиной запроса.

Изменение частоты тактирования контроллера и частоты SCL выполняется программной установкой в регистре CTRL[15,14] управляющей коммутатором выходов делителя PCLK на 4,8,16 и выхода 1го таймера (режимы его синхронизации предоставляют возможности выбора любой частоты тактирования). Базовая частота SCL получается делением на 16 частоты тактирования.

4.10 Арбитр магистрали внешней адресации

МК содержит встроенный арбитр внешней шины данных/адреса для построения многоконтроллерных систем с общими ресурсами. Встроенный арбитр допускает подключение нескольких МК к общей шине адреса/данных и управляет процессом использования отдельным МК общих ресурсов системы. Арбитр может работать в двух режимах - распределенном или централизованном. В первом случае арбитр использует четыре внешних сигнала BB,BR,BGin,BGou для управления (захват,запрос,разрешение_вход, разрешение_выход). Этот режим не требует дополнительной внешней логики для свой реализации. Арбитр одного из МК работает в режиме "master", все другие в режиме "slave".

В централизованном режиме работы арбитр использует два внешних сигнала BR и BB для управления (запрос, предоставление). см. рис.4.10.1. Этот режим предполагает наличие дополнительного внешнего арбитра, управляющего предоставлением шины конкретному МК. В этом режиме сигнал BGin есть вход общего назначения, а выход BGou есть признак "останов/ожидание" микроконтроллера (см. бит CNFG[4]).

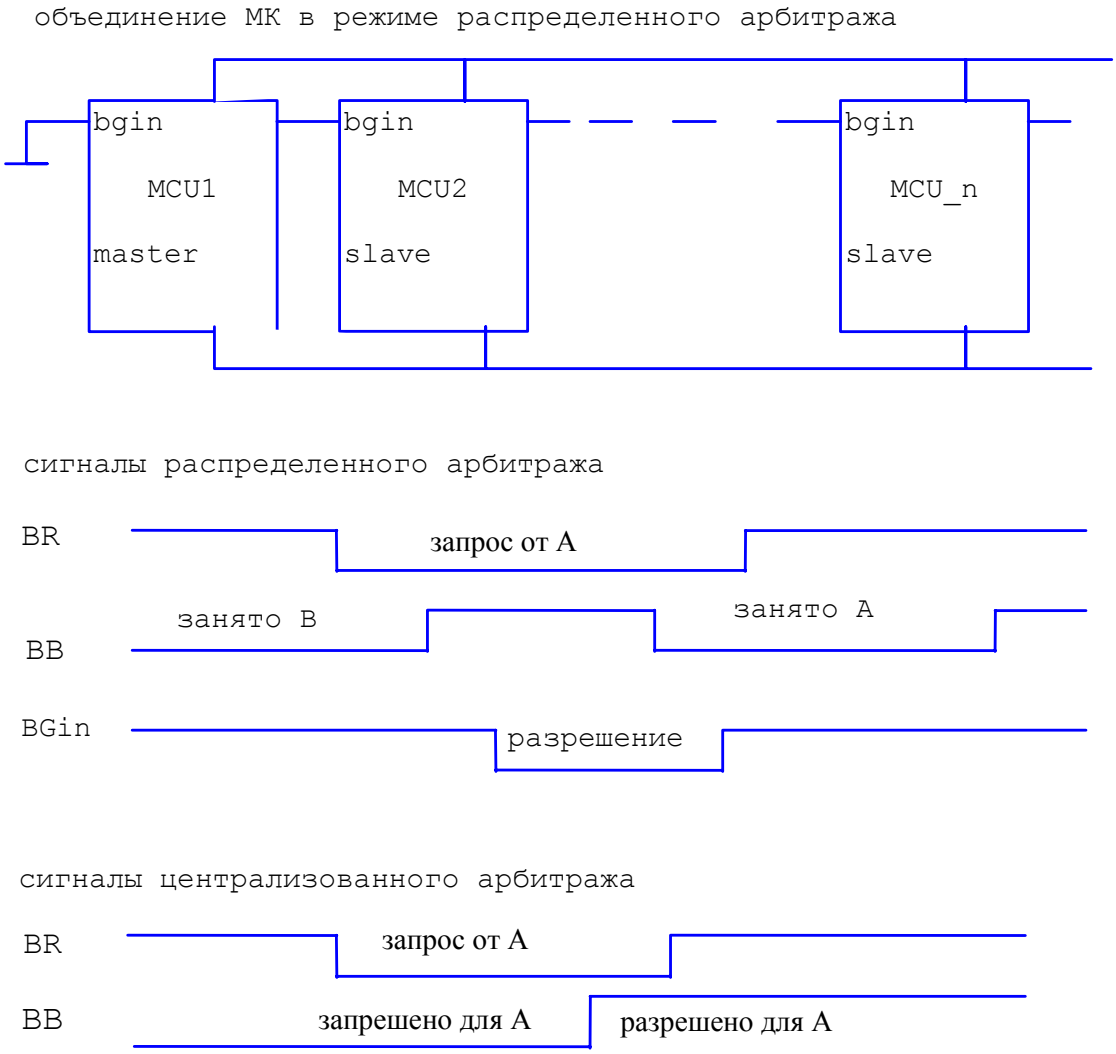


Рисунок .4.10.1. Управление арбитром в распределенном и централизованном режимах.

Выбор режима работы арбитра выполняется в момент сброса МК (кроме сброса от сторожевого таймера). Временные диаграммы приведены на рис.3.10.2. (CLR - внутренний сигнал сброса):

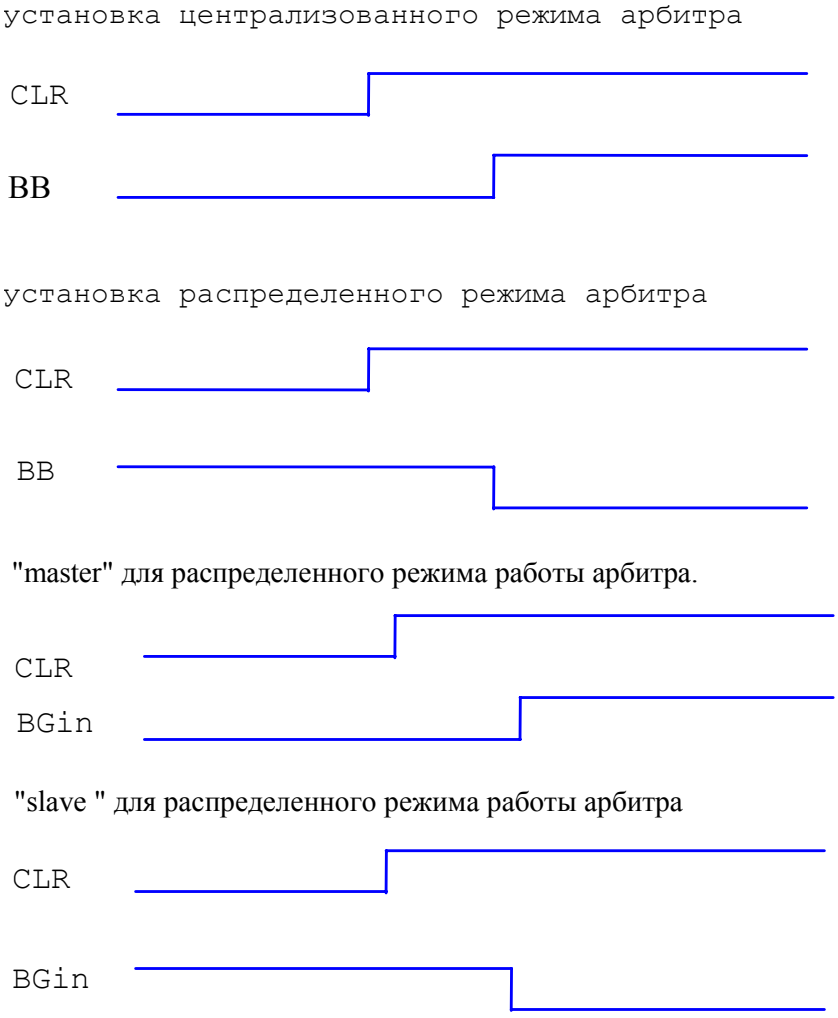


Рисунок .4.10.2. Установка режима рабты арбитра

4.11 Контроллер прерываний

МК имеет гибкую систему обработки прерываний. Источниками прерываний являются:

а) внешние сигналы INT0,INT1,INT2,TMI0,TMI1,TMI2;
б) внутренние устройства (таймеры, интерфейсы SSI,UART,I2C);
в) команда SWI.

Для обработки прерываний в МК предусмотрены:

- регистр запросов прерываний FLAG(см п. 4.4.3);
- регистр маски прерываний MASK (см п.4.4.4);
- регистр управления CTRL (см.4.4.2);
- бит разрешения прерываний PSW[1] и специальные команды для его установки или сброса.
- биты режимов обработки прерываний SMOD,VINT регистра CNFG.

Регистры FLAG и MASK были описаны выше. Каждый из 16-ти источников прерываний имеет свой 24-бит программный адрес прерывания. Биты 23-6 у всех кодов одинаковы и определяются битом SMOD регистра конфигурации. При SMOD=0 код прерываний находится в адресном пространстве команд 0 - 3f, а при SMOD=1 в адресном пространстве ffffc0 - ffffff. Младший и 5-тый биты кода преывания равны нулю. Таким образом биты 5-0 кодируют 16 источников прерывания:

3f-20 - резерв
00 - таймер 0
02 - таймер 1
04 - таймер 2
06 - таймер 3
08 - TMI0
0a - TMI1
0c - TMI2
0e - INT0
10 - INT1
12 - INT2
14 - приемник SSI_R
16 - передатчик SSI_R
18 - приемник SSI_L
1a - передатчик SSI_L
1c - UART, I2C
1e - SWI

Запросы на обслуживание прерваний имеют свои приоритеты. Наивысший приоритет имеет прерывание от таймера 0, низший приоритет - программное прерывание.

При появлении запроса на прерывание выполняется проверка следующих условий:

а) соответствующий бит регистра MASK имеет значение 1 т.е. разрешает прерывание;
б) общий бит разрешения прерываний PSW[1] = 1 т.е. прерывания разрешены;
в) текущая выполняемая команда не запрещает прерывание. Прерывания запрещены
- между первым и вторым тактами двухтактных команд,
- между специальными и следующими за ними любыми командами т.е. в случае программы

```
                round;
                nop;
                t+=n;
```

прерывание произойдет после команды "nop".

- между командой условного перехода и любой следующей за ней командой. Это связано с тем, что все переходы выполняются с задержкой на один такт и следующая за командой ветвления команда всегда выполняется. В случае

```
repeat: ifd(~ct--)      repeat;
                >t=*at++;
                nop;
```

прерывание произойдет только после выполнения команды >t=*at++.

- во время выполнения цикла DO(). В случае

```
                DO()            end_do;
                t=*sp++;
                *rp--=t;
end_do:         nop;
```

прерывание произойдет только после окончания выполнения цикла перед командой "nop".

Если нет условий запрещающих прерывание при его наличии, то в момент приема новой команды произойдет следующая последовательность действий

а) считанная из памяти программ новая команда игнорируется, но счетчик команд все равно увеличивается на 1. В регистр команд по положительному фронту сигнала PCLK защелкивается код

000 vvvv vvvv 0111

где биты v...v соответствуют значению бита SMOD регистра CNFG.

Для SMOD=0 - vvvv vvvv = 0000 0000,
SMOD=1 - vvvv vvvv = 1111 1111.

Данный код команды есть команда "длинного" вызова подпрограммы, которая выполняется за два такта. В первом такте происходит

б) очистка бита разрешения прерываний PSW<1>=0,

в) вычитание из значения счетчика команд единицы, т.е. --pc;

г) прием младших 16-ти бит адреса перехода (старшие восемь бит - это биты "vvvvvvvv" кода команды);

Прием младших бит адреса выполняется посредством чтения кода вектора прерывания

"vvvv vvvv vv0i iii0",

где бит "v"=SMOD , а биты "iiii" кодируют источник прерывания (см. выше).

Таким образом, в конце первого такта обработки прерывания мы имеем в

rt - адрес возврата и прерывания,
t - младшие 16-бит бит адреса перехода,
psw[1]=0 - запрещены прерывания.

Во втором такте выполняется обращение за командой по новому адресу перехода. Старое значение счетчика команд помещается на стек возврата, т.е. >rt=pc. В счетчик команд помещается адрес перехода + 1. С третьего такта начинается выполнение программы обработки прерывания.

Таким образом, в зависимости от значений бита SMOD регистра конфигурации , возможны следующие адреса вызова программ обработки прерываний

SMOD	адрес
0	0 - 1e (внешняя область программ)
1	ffffc0 - ffffde (внеш/внутр обл. программ)

Очевидно, что под программу обработки прерывания отводится только два слова программной памяти. Скорее всего, они могут быть использованы для размещения команды JMP на требуемую программу обработки прерывания. Значит, на вызов рабочей программы будет потрачено еще два такта. Если программа обработки прерывания изменяет значения некоторых регистров, то эти регистры должны быть предварительно сохранены , а перед выходом из программы - восстановлены. Сохранение регистров возлагается на программиста. Возврат из прерывания выполняется командой RTI (описана ниже).

5 Внешние контакты микроконтроллера

Конструктивно микросхема оформляется в пластмассовом 100-выводном корпусе с четырехсторонним расположением выводов, с шагом 0,65 мм, типа 4403Ю.100-А по ГОСТ 17467-88, которая показана на рис. 5.1.

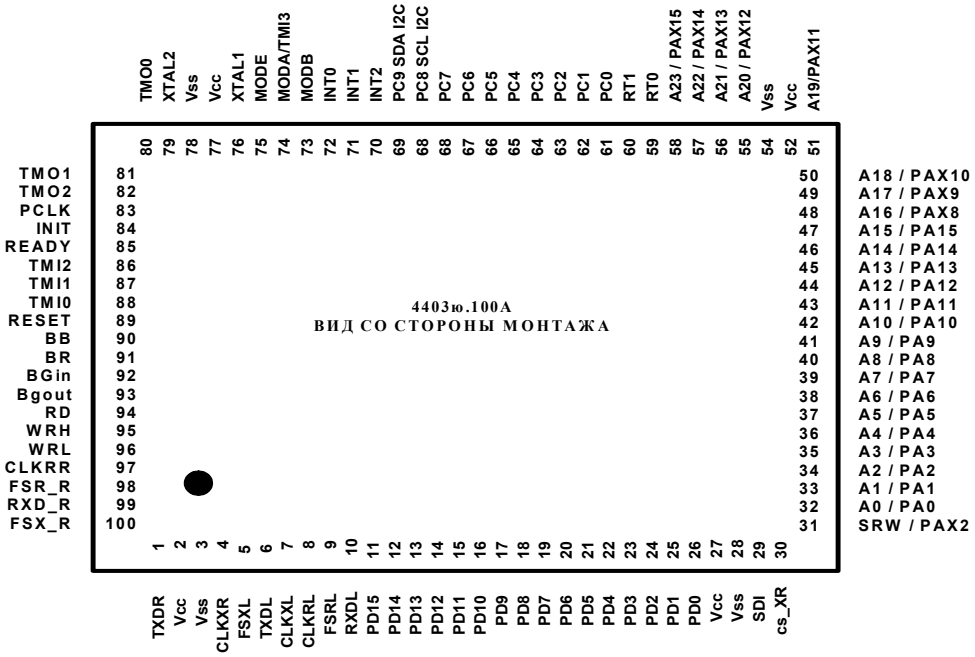


Рисунок 5.1 Корпус микросхемы

Таблица 5.1 Назначение выводов К1881ВЕ1Т.

Номер к.п.	Имя вывода	Вх/Вых	Назначение
1	TXD_R	Вх/вых	Данные передатчика SSI_R
2	VDD		Питание
3	VSS		Общий
4	CLKX_R	Вх/вых	Синхронизация передатчика SSI_R
5	FSX_L	Вх/вых	Строб передачи SSI_L
6	TXD_L	Вх/вых	Данные передатчика SSI_L
7	CLKX_L	Вх/вых	Синхронизация передатчика SSI_L
8	CLKR_L	Вх	Синхронизация приемника SSI_L
9	FSR_L	Вх	Строб приема SSI_L
10	RXD_L	Вх	Данные приемника SSI_L
11:26	PD[15:0]	Вх/вых	Порт D
27	VDD		Питание
28	VSS		Общий
29	SDI	Вых	Признак команда/данные
30	csXR	Вых	Выбор внешнего устройства
31	SRW	Вых	признак чтения/записи по порту D
32:51	PA[0:19]	Вых	разряды 0-19 порта А
52	VDD		питание
53	VSS		общий
54:57	PA[20:23]	Вых	разряды 20-23 порта А
58	RT0	Вх/Вых	Вх/выход передатчика SCI
59	RT1	Вх/Вых	Вх/выход приемника SCI
60:69	PC[0:9]	Вх/Вых	порт С
70	INT2	Вх	Запрос прерывания
71	INT1	Вх	Запрос прерывания
72	INT0	Вх	Запрос прерывания
73	MODB	Вх	Вход порта С
74	MODA	Вх	Управление таймером 3./ вход порта С.
75	MODE	Вх	Режим работы./ вход порта С
76	X1	Вх	Подключение внеш. кварц. резонатора
77	VDD		питание
78	VSS		общий
79	X2	Вх/Вых	Подключение внеш. кварц. резонатора
80	TMO0	Вых	Выход таймера 0
81	TMO1	Вых	Выход таймера 1
82	TMO2	Вых	Выход таймера 2
83	PCLK	Вых	Выходной синхросигнал МК
84	INIT	Вых	выход инициализации
85	READY	Вх	готовность данных
86	TMI2	Вх	управление таймером 2
87	TMI1	Вх	управление таймером 1
88	TMI0	Вх	управление таймером 0
89	RST	Вх	сброс
90	BB	Вх/вых	захват внешней шины
91	BR	Вх/вых	запрос внешней шины

92	BGIN	Вх	разрешение на захват шины
93	BGOU	Вых	разрешение на захват шины
94	RDD	Вых	строб чтения по порту D
95	WRH	Вых	строб записи старшего байта порта D
96	WRL	Вых	строб записи младшего байта порта D
97	CLKR_R	Вх	синхронизация приемника SSI_R
98	FSR_R	Вх	строб приема SSI_R
99	RXD_R	Вх	данные приемника SSI_R
100	FSX_R	Вх/вых	строб передачи SSI_R

Ниже дана характеристика основных выводов микросхемы.

5.1. Сигналы синхронизации и управления микроконтроллером.

X1 - вход кварцевого резонатора.

X2 - выход кварцевого резонатора. *

PCLK - выходной синхросигнал.

RST - вход сброса. Переводит МК в определенное состояние в котором сброшены некоторые регистры , все флаги , триггера в устройстве управления. Во время действия сигнала RST все двунаправленные выводы переводятся в состояние приема информации.

INIT - выход инициализации. Повторяет внутренний или внешний сигналы сброса(в том числе и от сторожевого таймера).

MODE - режим работы. Определяет стартовый адрес , использование портов и др.

5.2 Сигналы обмена с внешней памятью

A23:A00 - шина адреса. В каждом такте на шину адреса A[23:0] выдается адрес , определяющий нахождение требуемой в следующем такте информации. При сбросе шина адреса переводится в отключенное состояние.

D15:D00 - шина данных основной памяти. На данную шину помещается информация из ОЗУ при чтении в МК , или из МК при записи в ОЗУ. При сбросе шина данных переводится в отключенное состояние.

READY - вход готовности. Переводит МК в состояние , в котором входсинхронизации X1 не оказывает никакого воздействия на МП. Выходной сигнал PCLK при этом всегда имеет низкий уровень до окончания действия сигнала READY. Использование входа готовности позволяет синхронизировать работу МК с медленными устройствами , когда обмен с последними не может быть выполнен в течение одного такта.

RDD - строб чтения из внешнего ЗУ по шине D[15:0].

WRH,WRL - стробы записи из МК во внешнее ЗУ по шине D[15:0].

SRW - признак чтение/запись по шине данных

SDI - признак выборки команда/данные

cs_XR - обращение к внешней области данных $80 - $ff

При сбросе сигналы RDD,WRH,WRL,SRW,SDI,cs_XR переводятся в отключенное состояние.

5.3 Сигналы прерываний

INT[2:0] - запросы прерываний. Отрицательный/положительный (программируется регистром CTRL) фронт сигнала INT вызывает установку триггера запроса прерывания и если это прерывание не замаскировано ,то произойдет его обработка.

TMI[2:0] - аналогичны INT[2:0].

5.4 Сигналы последовательного интерфейса UART

TR0 - вход/выход данных UART_0
TR1 - вход/выход данных UART_1

5.5 Сигналы управления внутренними таймерами

TMI[2:0],MODA - входы управления внутренними таймерами (значение MODA может быть прочитано в 14 разряде порта C, без отмены функции как вход TMI3).

TMO[2:0] - выходные сигналы ,вырабатываемые внутренними таймерами

5.6 Порты ввода/вывода

PC[7:0] - универсальный порт ввода/вывода, каждый из выводов которого может быть индивидуально запрограммирован на ввод или вывод информации.

PA[15:0] - в зависимости от режима работы выход адреса A15:A0 либо выходной порт общего назначения

PAX[10:0] - в зависимости от режима работы выход адреса A23:A16, сигналы cs_XR,SDI,SRW либо выходной порт общего назначения.

5.7 Арбитр

BR - вход/выход запроса шины
BB - вход/выход занятости шины
BGin - вход предоставления шины либо вход общего назначения
BGou - выход предоставления шины либо выход ожидания

5.8 Последовательный интерфейс SSI

CLKX - вход/выход синхронизации передатчика
FSX - вход/выход строба начала передачи
TXD - выход данных передатчика
CLKR - вход синхронизации приемника
FSR - вход строба начала приема
RXD - вход данных приемника

6 Система команд микроконтроллера

6.1 Программная модель

В дальнейшем при описании процессов исполнения команды использованы следующие обозначения:

RS (или R) - стек возвратов состоит из
- Rsm - память 16*24 бит
- rt - 24-бит верхний регистр стека
- rsp - внутренний 4-бит указатель стека

AS (или A) - стек адресов состоит из
- Asm - память 8*24 бит
- at - 24-бит верхний регистр стека

asp - внутренний 3-бит указатель стека

Типовые стековые операции для AS и RS:

```
push / >R / >rt =>  RSm[++rsp]=rt; rt=?;
       >A / >at =>  ASm[++asp]=at; at=?;
pop / R> / rt> =>  ?=rt; rt=RSm[rsp--];
       A> / at> =>  ?=at; at=RSm[asp--];
```

DS (или D) - стек данных состоит из

Dsm - память 8*32 бит
N - 32_бит регистр
T - 32_бит регистр
dsp - внутренний 3-бит указатель стека

Поскольку АЛУ микропроцессора 16-разрядное, стек данных разбит на две симметричные 16-бит части - младшую (t,n,dsm,dsp) и старшую (%t, %n, %dsm, %dsp).

t,n - младшая часть T,N (15-0 биты)
%t,%n - старшая часть T,N (31-16 биты)

Старшая и младшая части стека данных независимы и большинство операций выполняются в данный момент времени только с одной половиной. Типовые стековые операции для DS:

```
push/ >D / >t =>       Dsm[++dsp]=n;     n=t;    t=?;
              >n =>  Dsm[++dsp]=n;       n=?;
pop/ D> / t> =>       ?=t;    t=n;    n=DSm[dsp--];
          n> =>       ?=n;    n=DSm[dsp--];
```

Внутренние регистры:

USER - 24-бит указатель банка регистров
PC - 24-бит счетчик команд
L - 16-бит регистр команды
work - 32-бит рабочий регистр
SP - 24-бит адресный регистр
RP - 24-бит адресный регистр
CT - 16-бит счетчик
PSW - 16-бит регистр состояния
BUF - 16-бит буферный регистр
XMAC - 6-бит спец регистр

Используемые обозначения:

% - выполняется операция над старшим словом стека данных DS

Пример:

```
/*  32-бит сложение
        t+n;
%;      t+n;    +c;
```

t - младшая часть регистра T(биты 15:0)
%t - старшая часть регистра T(биты 31:16)
T - 32-бит регистр

Для регистров-переменных t,n,at,rt, входящих в состав стеков, при записи выражений могут встречаться обозначения >t,>n,>at,>rt, t>,n>,at>,rt>. Выражение вида ">имя" означает запись информации на стек, а "имя>" - снятие информации со стека. Эти действия выполняются

параллельно и независимо от основной операции. Например,выражение "*at++" означает использование регистра at в качестве адреса и последующего его увеличение на 1. Приведенные выше обозначения означают выполнение следующих действий:

```
>t      => DSm[++dsp]=n; n=t;
>n      => DSm[++dsp]=n;
>at     => ASm[++asp]=at;
>rt     => RSm[++rsp]=rt;
t>      => t=n; n=DSm[dsp--];
n>      => n=DSm[dsp--];
at>     => at=ASm[asp--];
rt>     => rt=RSm[rsp--];
```

mem(A) - содержимое внешней памяти МК по адресу A
REG[i:j] - биты регистра REG с i по j

В коде команды символ "-" означает что данный бит может принимать любое значение. Для сокращения длины выражений м.б. использованы эквивалентные типы выражений. К примеру, записи

```
t+n;
t=t+n;
t+=n;
```

являются эквивалентными.

В приводимых примерах запись типа

D: 0 1 2 3 >

означает, что стек данных(младшая половина) содержит значения

3 - в регистре t,
2 - в регистре n,
1 - в ячейке памяти стека по адресу dsp.
0 - в ячейке памяти стека по адресу dsp-1.

%D: 0 1 2 3 >

означает, что стек данных(старшая половина) содержит значения

3 - в регистре %t,
2 - в регистре %n,
1 - в ячейке памяти стека по адресу %dsp.
0 - в ячейке памяти стека по адресу %dsp-1.

R: 1 2 3 >

означает, что стек возвратов содержит значения

3 - в регистре rt,
2 - в ячейке памяти стека по адресу rsp.
1 - в ячейке памяти стека по адресу rsp-1.

A: 1 2 3 >

означает, что стек адресов содержит значения

3 - в регистре at,
2 - в ячейке памяти стека данных по адресу asp.
1 - в ячейке памяти стека данных по адресу asp-1.

100: 7 >

означает , что в ячейке памяти с адресом 100 хранится значение 7.

6.2 Команды переходов

6.2.1 Вызов подпрограммы

а) локальный.

синтаксис : имя_п/п();

код команды: 0001 аааа аааа аааа

выполнение: >rt=pc; L=mem(a_sub); pc=a_sub+1;

где a_sub=PC[23:12]!!L[11:0]

слов : 1
циклов : 1

описание : осуществляется вызов п/п в пределах страницы обьемом 4К слов. Адрес возврата помещается на стек RS.

пример : fadd(): t+n; exit;

. . .
fadd();

б) глобальный.

синтаксис : имя_п/п();

код команды: 0000 аааа аааа 0111
аааа аааа аааа аааа

выполнение: 1. >t=mem(pc++);
2. >rt=pc; L=mem(a_sub); pc=a_sub+1; t>;

где a_sub=L[27:20] !! T[15:0]

слов : 2
циклов : 2

описание : осуществляется вызов п/п в пределах всего адресного пространства МК. Адрес возврата помещается на стек RS.

пример :

```
fadd();
. . .
fadd(): t+n; exit;
```

6.2.2 Возврат из подпрограммы

а) обычный

синтаксис : exit/rts/return;

код команды: 100- ---- ---- 1100

выполнение: pc=rt+1; L=mem(rt>);

слов : 1
циклов : 1

описание : осуществляется возврат из ранее вызванной п/п. Адрес возврата находится в регистре RT. Выход из п/п выполняется параллельно с выполнением операции в АЛУ,задаваемой неиспользуемыми при возврате битами(-) регистра команд.

пример :
```
fadd();
. . .
fadd(): t+n; exit;
```

б) возврат из прерывания

синтаксис : rti;

код команды: 100- ---- ---- 1101

выполнение: pc=rt+1; L=mem(rt>);
psw[1]=1; /* разрешение прерываний */

слов : 1
циклов : 1

описание : осуществляется возврат из ранее вызванной п/п обработки прерывания. Адрес возврата находится в регистре RT. Выход из п/п выполняется параллельно с выполнением операции в АЛУ,задаваемой неиспользуемыми при возврате битами(-) регистра команд. Одновременно с выходом из п/п устанавливается бит разрешения прерывания.

пример :
```
fadd();
. . .
fadd(): t+n;    rti;
```

6.2.3 Безусловный переход

синтаксис : jmp имя_метки;

код команды: 0000 aaaa aaaa 0110
aaaa aaaa aaaa aaaa

выполнение: 1. >t=mem(pc++);
2. L=mem(a_jmp); pc=a_jmp+1; t>;

где a_jmp=L[27:20] !! T[15:0]

слов : 2
циклов : 2

описание : осуществляется передача управления команде, определяемой меткой "имя_метки", в пределах всего адресного пространства МК.

пример :
```
        jmp   fadd;
        . . .
fadd:   t+n;
```

6.2.4 Условные переходы

синтаксис : if("условие перехода") метка_перехода;

код команды: 001c cccc aaaa aaaa

выполнение:
```
L=mem(pc);
if(ccccc is true)   pc=pc+offset;
        else        pc++;
```

где offset=aaaaaaaa

слов : 1
циклов : 1
описание : осуществляется передача управления команде, определяемой меткой "метка_перехода", если условие перехода выполняется. Поле "условие перехода" определяет условие проверяемое перед выполнением перехода. Возможные команды переходов:

c cccc	синтаксис	описание
h 0000	if(~n) метка;	знак операнда в регистре t равен нулю
h 1000	if(n) метка;	знак операнда в регистре t равен единице
h 0001	if(~z) метка;	регистр t не ноль
h 1001	if(z) метка;	регистр t равен нулю
h 0010	if(~v) метка;	значение v=(t[15] ^ psw[0]) равно нулю
h 1010	if(v) метка;	значение v=(t[15] ^ psw[0]) равно единице
h 0011	if(~c) метка;	бит переноса(c=psw[0]) равен нулю
h 1011	if(c) метка;	бит переноса(c=psw[0]) равен единице
h 0100	if(~z&c) метка;	для беззнаковых чисел A-B: A>B
h 1100	if(~c|z) метка;	для беззнаковых чисел A-B: A<=B
0 0101	if(ct) метка;	содержимое счетчика CT не равно нулю
0 1101	if(!ct) метка;	содержимое счетчика CT ноль
1 0101	if(ct--) метка;	содержимое счетчика CT не ноль и последующее вычитание из него 1

1 1101	if(!ct--) метка;	содержимое счетчика СТ ноль и последующее вычитание из него 1
h 0110	if(~nb) метка;	значение бита t[7] равно нулю
h 1110	if(nb) метка;	значение бита t[7] равно единице
- 0111	if() метка;	безусловный переход

Бит "h" определяет какой стек данных(ст/мл) используется при анализе условия перехода. Переход на метку задается в коде команды 8-разрядным знаковым смещением относительно текущего значения счетчика команд. Команда перехода выполняется следующим образом.

В текущем такте происходит параллельное выполнение следующих действий:

1. на адресном сумматоре происходит сложение текущего значения счетчика команд (указывает на адрес следующей команды) с 8-разрядным знаковым смещением , заданным в поле команды. Вычисляется адрес перехода.
2. выполняется выборка следующей команды по адресу в счетчике команд.
3. происходит наращивание счетчика команд на 1.
4. анализируется условие перехода, заданное в коде команды.

В конце такта, если условие перехода выполняется то PC=PC+смещение, в противном случае PC=PC+1. Независимо от результатов анализа - следующая за командой ветвления команда будет принята в регистр команд и выполнена всегда. Между командой ветвления и следующей за ней командой невозможно прерывание.

V1) Инструкция размещаемая за командой ветвления не должна быть длинной командой (т.е.должна быть в одно слово.)

Таким образом команда ветвления МП выполняется с задержкой. Если после команды ветвления нет возможности поместить полезную команду, то компилятор должен помещать команду NOP. В ассемблере МП имеются два варианта команды ветвлений (коды команд одинаковы):

а) if(?) метка;
б) ifd(?) метка;

В варианте а) компилятор сразу после команды ветвления размещает команду "нет операции". В варианте б) после команды ветвления компилируется следующая за ней команда из потока команд. Таким образом программист сам оптимизирует поток выполняемых команд.

пример :

```
        if(n)   exe_n1;
        . . .
exe_n1: t+n;
```

6.3 Операции со стеком данных

6.3.1 Операции АЛУ

а) арифметико-логические

синтаксис : t -*op*- n;
#literal -*op*- t;

код команды: 011- ffff ---- ----
1--- ffff --00 ----
1--- ffff --1- ----

выполнение: 011- ffff -0-- ---- #literal -*op*- t;
011- ffff -1-- ---- t -*op*- n;
1--- ffff --00 ---- t -*op*- n;
1--- ffff --1- ---- t -*op*- n;

ffff	синтаксис		описание
0000	nop;		пересылка t
0001	t+n;		сложение
0010	n-t;		обратное вычитание
0011	t-n;		вычитание
0100	t|n;		логическое поразрядное сложение
0101	t&n;		логическое поразрядное умножение
0110	t^n;		поразрядное "исключающее ИЛИ"
0111	n;		пересылка
1000	~t;		поразрядная инверсия
1001	t+n+c;		сложение с учетом переноса
1010	n+~t+c;		обратное вычитание с учетом переноса
1011	t+~n+c;		вычитание с учетом переноса
1100	u_t*n;	*	умножение кодов
1101	t*n;	*	знаковое умножение
1110	~(t^n);		инверсное "исключающее или"
1111	t=~n;		пересылка с инверсией

слов : 1
циклов : 1
описание : микропроцессор выполняет арифметические и логические операции над операндами находящимися в регистрах T и N (или литералом). Результат операции помещается в T или N.
*) Использование кода умножения u_t*n или t*n в формате команд с уможением wh*wl запрещено. Выполнение нестандартнай операции умножения %u_t*u_t или %t*t возможно только в некоторых форматах и определено в п.6.3.2.
пример : /* D: 1 2 3 >
/* %D: 5 8 7 >
t*n>;
/* D: 1 6 >
/* %D: 5 8 0 >

б) операции сдвига

синтаксис : "?">>t; 16-бит сдвиг вправо
"?">>T; 32-бит -"-
t<<"?"; 16-бит сдвиг влево
T<<"?"; 32-бит -"-

код команды: 1--h ffff --01 ----

выполнение: 1--h ffff --01 ----

h ffff	синтаксис	описание
h 0000	0>>T;	логический сдвиг вправо
h 0001	c>>T;	циклический с учетом переноса -"-
h 0010	T0>>T;	циклический сдвиг
h 0011	S>>T;	арифметический -"-
h 0100	0>>t;	логический сдвиг вправо
h 0101	c>>t;	циклический с учетом переноса -"-
h 0110	t0>>t;	циклический -"-
h 0111	s>>t;	арифметический -"-
h 1000	T<<0;	логический сдвиг влево
h 1001	T<<c;	циклический с учетом переноса -"-
h 1010	T<<T0;	
h 1011	T<<S;	циклический -"-
h 1100	t<<0;	логический сдвиг влево
h 1101	t<<c;	циклический с учетом переноса -"-
h 1110	t<<t0;	
h 1111	t<<s;	циклический -"-

знак "?" определяет бит информации который поступает в освобождаемый при сдвиге бит: 0 - ноль
c - значение триггера переноса (psw[0])
t0 - значение нулевого бита регистра t[15:0]
T0 - значение нулевого бита регистра T[31:0]
s - значение знакового 15 бита регистра t или %t
S - значение знакового 15 бита старшего слова регистра T

слов : 1
циклов : 1
описание : микропроцессор выполняет операцию сдвига над 16-бит регистром t (T[31:16] или T[15:0]), либо над 32-бит регистром T. Старшинство слов определяется битом h, при h=0 старшее T[31:16], при h=1 старшее T[15:0]. Выполнение 32 р.сдвига с загрузкой результата в регистр n имеет особенности, в n загружается результат из tl, при h=0 в nl, при h=1 в nh, th остается на месте а в регистр tl помещается результат второй параллельной операции. Выдвигаемый бит помещается в бит переноса.

Пример :
```
/*  D: 5 >
/* %D: 4 >
T0>>T;
/*  D:     2 >
/* %D: 0x1002 >
   T<<s;
/*  D: 5 >
```

```
/* %D: 4 >
```

в) операция умножения с использованием регистра WORK.

синтаксис *:* n=t -*op*- n; t=W*;

код команды: 1--h ffff --10 ----

выполнение: 1--0 ffff --10 ---- n=t-*op*-n; t=wh*wl;
1--1 ffff --10 ---- %; n=t-*op*-n; T=wh*wl, wl=buf;

слов *:* 1
циклов *:* 1
описание *:* микропроцессор выполняет арифметические и логические операции над операндами находящимися в регистрах t и n (в соответствии с битом h). Результат операции помещается в N. Параллельно с операцией АЛУ выполняется умножение старшей части регистра WORK (биты 31:16) на младшую часть(биты 15:0). Результат операции помещается в регистр T (в биты 15:0 если h=0 и если h=1 в биты 31:0 с wl=buf). Если данная команда использует дополнительно еще и обращение к внешней памяти за операндом, то считываемый операнд поступает в регистр WORK (в соответствии с п.п.6.6.1 и 6.6.2).
пример *:*

```
/*   D: 1 2 3 >
/*  %D: 3 2 1 >
/*  wh:    3 >
/*  wl:    2 >
/* buf:    0 >
/*   A:    3 >
/*   3:   55 >

n=t+n; t=W*;          *at++;

/*   D: 1 5 6 >
/*  %D: 3 2 1 >
/*  wh:    3 >
/*  wl:    2 >
/* buf:   55 >
/*   A:    4 >
/*   4:   77 >

%;    n=t+n; +c;    t=W*;          *at++;

/*   D: 1 5 6 >
/*  %D: 3 3 0 >
/*  wh:   77 >
/*  wl:   55 >
/* buf:   55 >
/*   A:    5 >
```

6.3.2 Операции со стеком и параллельные операции

синтаксис : "ALU_oper"; "gggg_oper";

код команды: 100h ffff psmm gggg

выполнение: ffff - поле кодирует операцию АЛУ (*-op-* либо *shift*)

```
psmm      тип операции
0000      t=t-op-n;
0100      t=t-op-n>;
1000      t=t-op-n;     n=t;
1100      >t=t-op-n;

0001      t=shift(t);
0101      t=shift(t);   n>;
1001      t=shift(t);   n=t;
1101      >t=shift(t);

-010      n=t-op-n;     t=wh*wl;  (исполнение соотв. p=0)
0110      n=t-op-n>;    t=wh*wl;
1110      >n=t-op-n;    t=wh*wl;

0011      t=t-op-n; (исполнение соответствует mm=00)
0111      t=t-op-n>;
1011      t=t-op-n;     n=t;
1111      >t=t-op-n;
```

* Стековые обмены (включая регистр N) с 32-разр.данными в операции сдвига осуществляется только 16-разр. словом определяемым битом h.
* Одновременно с операцией умножения %t=wh*wl (h=1)выполняется загрузка wl=buf.
* Умножения T=t*%t кодируется 100- 1101 0010 gggg и T=ut*%ut -100- 1100 0010 gggg. Операции wl=buf и wh*wl при этом не выполнятся.

Поле gggg кодирует параллельную операцию, которая выполняется независимо от операции АЛУ.

```
gggg      операция
0010      sp++;
0011      sp--;
0100      rp++;
0101      rp--;
0111      ct--;
1000      step_*;
1001      step_/;
101-      at>;
1100      exit;
1101      rti;
1110      user=rt>;
1111      rt>;
```

слов : 1
циклов : 1

описание *:* выполняется операция над содержимым регистров T и N стека данных. Результат операции поме щается в регистр T. Параллельно с операцией в АЛУ выполняется операция "gggg_oper" над адресными регистрами либо стеком адреса (возвратов) либо счетчиком CT. Операции step_*, step_/ в сочетании с соответствующими операциями АЛУ (сложение,вычитание) позволяют реализовать микрооперации операций умножения,деления.

6.4 Операции с внутренними регистрами

а) чтение

синтаксис : n=t-*op*-n; t="имя_регистра";
>n=t-*op*-n; t="имя_регистра";

код команды: 011h ffff p1g0 rrrr

Поле rrrr кодирует внутренний регистр микропроцессора

0000 - user
0001 - pc
0010 - work
0011 - sp
0100 - swt (r/o)
0101 - rp
0110 - rt
0111 - rt> загрузка rt выполнится с сохранением , чтение с восстановлением из стека
1000 - at
1001 - at> загрузка at выполнится с сохранением , чтение с восстановлением из стека
1010 - xmac (w/o)
1011 - psw(r/w) , xmac (r/o)
1100 - buf
1101 -
1110 - ct
1111 -

выполнение: n=t-*op*-n; t="имя_регистра"; (p==0)
>n=t-*op*-n; t="имя_регистра"; (p==1)

слов : 1
циклов : 1

описание : выполняется операция над содержимым регистров T и N стека данных. Результат операции помещается в регистр N. Параллельно с операцией в АЛУ выполняется операция чтения содержимого внутреннего регистра (код "rrrr"). Считываемый операнд помещается в регистр T старший байт T[31:24] при чтении адресного регистра заполняется 0.

```
/*  D: 1 2 3 >
/*  R:  3 4 >
        n=t+n; t=rt>;
/*  D: 1 5 4 >
```

```
/*  R:    3 >
```

Если необходимо сохранить содержимое регистра N на стеке используется код команды с битом "p"==1, что соответствует выражению >n=t-*op*-n; t=reg;

```
/*  D: 1 2 3 >
/*  R:  3 4 >
        n+=t;          t=rt>;
/*  D: 1 2 5 4 >
/*  R:     3 >
```

При операциях с внутренними регистрами регистр T рассматривается как 32-разрядный (бит "g"==1) либо как 16-разрядный(бит "g"==0). При g==0 учитывается бит "h".

011h ffff p1g0 rrrr

0	0	t=reg[15:0];
1	0	%t=reg[31:16];
-	1	T=reg[31:0];

Регистр SWT доступен только по чтению и позволяет реализовать перестаноку старшей и младшей частей регистра T.

при h==0 SWT=T[15:0] !! T[31:16]
при h==1 SWT=T[31:16] !! T[15:0]

пример:

```
/*  %D: 1 2 >
/*   D: 3 4 >
n+=t;          T=swt;
/*  %D: 1 4 >
/*   D: 7 2
```

б) запись

синтаксис : "имя_регистра"=t; t-*op*-n;
"имя_регистра"=t; t-*op*-n>;

код команды: 011h ffff p1-1 rrrr (кодирование "rrrr" приведено выше)

выполнение: "имя_регистра"=t; t=t-*op*-n; (p==0)
"имя_регистра"=t; t=t-*op*-n>; (p==1)

слов : 1
циклов : 1
описание : выполняется операция записи содержимого регистра T во внутренний регистр. Параллельно с операцией записи выполняется операция над содержимым регистров t и n стека данных. Результат операции помещается в регистр t. Если необходимо при этом загрузить в регистр n из стека, используется код команды с битом "p"==1, что соответствует выражению reg=t; t-op-n>;
В операции пересылки регистр T всегда 32-разрядный мультиплексированный между tl th соответственно биту h.

h == 0 => reg=%t !! t
h == 1 => reg= t !! %t

пример:
```
/* %D: 1 2 5 >
/*  D: 5 6 1 >
/*  R:  3 4 >
%;      >rt=t;          t+n>;
/* %D:  1 7 >
/*  D: 5 6 1 >
/*  R: 3 4 x10005 >
```

6.5 Операции с литералом

6.5.1 Короткий литерал

синтаксис : #-*op*- t;
#-*op*->t ;

код команды: 011h ffff p0ee dddd

выполнение: t=#-op-t; (p==0)
>t=#-op-t; (p==1)

непосредственный операнд – константа генерируется
dddd * 2**(ee) , где ee=0/1/2/3 -> 2**0/4/8/12
сдвигом 4-х бит литерала dddd по квадрантам на 4,8,12 бит соответственно ее, литерал с кодами ее=10, d=111- использовать запрещается для избежания перекрытия кода команды

слов : 1

циклов : 1

описание : выполняется операция над содержимым регистра T и непосредственным операндом, содержащемся в коде команды

пример :
```
/* D: 2 7 >
t+#1;
/* D: 2 8 >
>t-#2;
/* D: 2 8 6 >
```
испрользование констант ее=2 и dddd=111-, приведет к исполнению команды длинного литерала вследствии перекрытия кодов. Поэтому эту комбинацию использовать нельзя. Для загрузки константы в регистр следует кодировать АЛУ операцию =n (ffff=0111).

6.5.2 "Длинный" литерал

синтаксис : #-op-t;
#-op->t;

код команды:
```
011h ffff p010 111-
dddd dddd dddd dddd          16_бит операнд
```

выполнение:
```
1:    >t=mem(pc++);
2.    t-op-n;              (если p==0)
      t-op-n>;             (если p==1)
```

слов : 2

циклов : 2

описание : выполняется операция над содержимым регистра T и

непосредственным операндом, содержащемся в во втором слове кода команды. Для загрузки константы в регистр следует кодировать АЛУ операцию =t (ffff=0000).

пример :
```
/* D: 2 7 >
t+#1;
/* D: 2 8 >
>t-#2;
/* D: 2 8 6 >
```

6.6 Операции с внешней памятью

6.6.1 Операции с внешними регистрами

а) чтение

синтаксис : `n=t-op-n;  t=rg["имя_регистра"];`
`>n=t-op-n;  t=rg["имя_регистра"];`

код команды: 11rh ffff p0mm rrrr

выполнение: в зависимости от ключевых разрядов “h p mm” выполняются следующие действия

```
h    p0mm
-    0000    n=t-op-n;              t=rg[];
-    1000   >n=t-op-n;              t=rg[];
0    0010    n=t-op-n;   t=wh*wl;   buf=rg[];
0    1010   >n=t-op-n;   t=wh*wl;   buf=rg[];
1    0010 %; n=t-op-n;   T=wh*wl;   wh=rg[];  wl=buf;
1    1010 %;>n=t-op-n;   T=wh*wl;   wh=rg[];  wl=buf;
-    0001    n=shift(t);            t=rg[];
-    1001   >n=shift(t);            t=rg[];
0    0011    t=t-op-n;              buf=rg[];
0    1011    t=t-op-n>;             buf=rg[];
1    0011 %; t=t-op-n;              wh=rg[];  wl=buf;
1    1011 %; t=t-op-n>;             wh=rg[];  wl=buf;
```

слов : 1
циклов : 1

описание : выполняется операция над содержимым регистров T и N стека данных. Результат операции помещается в регистр N. Параллельно с операцией в АЛУ выполняется операция чтения содержимого внешней памяти по адресу *aop=user[23:5].rrrrr*
Считываемый операнд помещается в регистр T. Если необходимо сохранить содержимое регистра N на стеке используется код команды с битом "p"==1, что соответствует выражению >n=t-op-n; t=rg[];
Команда позволяет обращаться к 32-м словам в окне, задаваемом регистром базы USER.

пример :
```
/*   D: 2 7 >
/*  105: 5 >
/* user: 100 >
n+=t;          t=rg[5];
```

```
/*    D: 9 5 >
/*  105: 5 >
/* user: 100 >
```

* При использовании 32-р. регистра T в сдвигах загрузка результата результата всегда выполняется в th, nh или nl, nh при h=1 или nl при h=0, в tl выполняется чтение из памяти.Старшинство слов изменяется при h=1, T[15:0] старшее, T[31:16] младшее слово.

б) запись

синтаксис : rg["имя_регистра"]=t; t-op-n;
rg["имя_регистра"]=t; t-op-n>;

код команды: 11rh ffff p1mm rrrr

выполнение: в зависимости от ключевых разрядов “h p mm” выполняются следующие действия

```
h    p1mm
-    0100   rg[]=t; t=t-op-n;
-    1100   rg[]=t; t=t-op-n>;

0    0110   rg[]=t; n=t-op-n; t=wh*wl;
0    1110   rg[]=t;>n=t-op-n; t=wh*wl;
1    0110 %;rg[]=t; n=t-op-n; T=wh*wl;  wl=buf;
1    1110 %;rg[]=t;>n=t-op-n; T=wh*wl;  wl=buf;

-    0101   rg[]=t; t=shift(t);
-    1101   rg[]=t; t=shift(t);    n>;

-    0111   rg[]=t; t=t-op-n;
-    1111   rg[]=t; t=t-op-n>;
```

слов : 1
циклов : 1
описание : выполняется операция записи содержимого регистра T во внешнюю память по адресу *aop=user[23:5].rrrrr*

Параллельно с операцией записи выполняется операция над содержимым регистров T и N стека данных. Результат операции помещается в регистр T. Если необходимо при этом убрать содержимое регистра N со стека, используется код команды с битом "p"==1, что соответствует выражению rg[]=t; t-op-n>;

Данные команды позволяют прямо обращаться к 32-м 16-бит регистрам в текущем регистровом окне, задаваемом регистром базы USER. При выполнении 32-разрядных сдвигов результат помещается в %t !! n при h = 0 и в %n !! t при h = 1.

пример :

```
/*    D: 3 2 7 >
/*  105: 5 >
/* user: 100 >
rg[5]=t;        t+n>;
/*    D: 3 9 >
/*  105: 7 >
/* user: 100 >
```

6.6.2 Операции с использованием адресных регистров

а). Методы адресации.

В коде команды 101h ffff pwmm xxxxполе xxxx указывает тип адресации

0000	sp	0001	rp	0010	sp++	0011	sp--
0100	rp++	0101	rp--	0110	T	0111	pc++
1000	at++	1001	at--	1010	at	1011	at>
1100	rt++	1101	irt++	1110	rt	1111	rt>

* Обращаем внимание на метод адресации irt++. Он эквивалентен методу адресации rt++ с тем отличием , что обращение происходит к памяти программ.

б) чтение

синтаксис :
```
 n=t-op-n; t=*a_reg;
>n=t-op-n; t=*a_reg;
```

код команды:
```
101h ffff p0mm xxxx
```

выполнение: в зависимости от ключевых разрядов “h p mm” выполняются действия

```
h    p0mm
-    0000       n=t-op-n;                t=*a_reg;
-    1000      >n=t-op-n;                t=*a_reg;

0    0010       n=t-op-n; t=wh*wl;       buf=*a_reg;
0    1010      >n=t-op-n; t=wh*wl;       buf=*a_reg;
1    0010   %;  n=t-op-n; T=wh*wl;       wh=*a_reg; wl=buf;
1    1010   %; >n=t-op-n; T=wh*wl;       wh=*a_reg; wl=buf;

-    0001       n=shift(t);              t=*a_reg;
-    1001      >n=shift(t);              t=*a_reg;

0    0011       t=t-op-n;                buf=*a_reg;
0    1011       t=t-op-n>;               buf=*a_reg;
1    0011    %; t=t-op-n;                wh=*a_reg; wl=buf;
1    1011    %; t=t-op-n>;               wh=*a_reg; wl=buf;
```

слов : 1 (2 для pc++)
циклов : 1 (2 для pc++ и irt++)

описание : выполняется операция над содержимым регистров T и N стека данных. Результат операции помещается в регистр N. Параллельно с операцией в АЛУ выполняется операция чтения содержимого внешней памяти по адресу, находящемуся в адресном регистре "a_reg". Считываемый операнд помещается в регистр T. Если необходимо сохранить содержимое регистра N на стеке используется код команды с битом "p"==1, что соответствует выражению **>n=t-op-n; t=*a_reg.** При выполнении 32-разрядных сдвигов результат помещается в %t !! n при h = 0 и в %n !! t при h = 1.

пример :
```
/*   D: 2 7 >
/*  105: 5 >
```

```
/*    R: 8 9 105 >
n=t+n; t=*rt>;
/*    D: 9 5 >
/*  105: 5 >
/*    R: 8 9 >
```

в) запись

синтаксис : *a_reg=t; t-op-n;
*a_reg=t; t-op-n>;

код команды: 101h ffff p1mm xxxx

выполнение: в зависимости от ключевых разрядов “h p mm” выполняются действия

```
h     p1mm
-     0100        *a_reg=t;  t=t-op-n;
-     1100        *a_reg=t;  t=t-op-n>;

0     0110        *a_reg=t;  n=t-op-n; t=wh*wl;
0     1110        *a_reg=t; >n=t-op-n; t=wh*wl;
1     0110   %;   *a_reg=t;  n=t-op-n; T=wh*wl; wl=buf;
1     1110   %;   *a_reg=t; >n=t-op-n; T=wh*wl; wl=buf;

-     0101        *a_reg=t;  t=shift(t);
-     1101        *a_reg=t;  t=shift(t);  n>;

-     0111        *a_reg=t;  t=t-op-n;
-     1111        *a_reg=t;  t=t-op-n>;
```

слов : 1
циклов : 1

описание : выполняется операция записи содержимого регистра T во внешнюю память по адресу находящемуся в адресном регистре "a_reg". Параллельно с операцией записи выполняется операция над содержимым регистров T и N стека данных. Результат операции помещается в регистр T. Если необходимо при этом убрать содержимое регистра N со стека, используется код команды с битом "p"==1, что соответствует выражению

***a_reg=t; t-op-n>;**

пример :

```
/*    D: 3 2 7 >
/*  105: 5 >
/*    A: 105 >
*at++=t; t+=n>;
/*    D: 3 9 >
/*  105: 7 >
/*    A: 106 >
```

6.7 Операции с адресными регистрами

6.7.1 Загрузка адресных регистров

синтаксис : "a_reg"=#LA;
код команды: 0000 aaaa aaaa rrrr
aaaa aaaa aaaa aaaa
выполнение: 1. >t=mem(pc++);
2. a_reg=LA; t>; L=mem(pc++);

регистр **rrrr**

```
0000           user=LA;
0001 >rt=user; user=LA;
001-  sp=LA;
010-  rp=LA;
10-0  at=LA;
10-1 >at=LA;
11-0  rt=LA;
11-1 >rt=LA;
```

где LA=L[27:20] !! T[15:0]

слов : 2
циклов : 2

описание : осуществляется загрузка непосредственного операнда(адреса), находящегося в коде команды, в адресный регистр.

пример :

```
/*   A: 3 5 >
>at=#5a5a5a;
/*   A: 3 5 5a5a5a>
```

6.7.2. Сложение с 8-разрядным смещением.

синтаксис : a_reg1=a_reg2+#offset;
код команды: 010x xxxx 0xxx rrrd

выполнение: a_reg1=a_reg2+offset; (где offset= xxxxxxxx)
Бит “d” кодирует **a_reg1** - приемник результата сложения

d	a_reg1
0	rt
1	at

Биты “rrr” кодируют **a_reg2** и тип операции

```
001 > a_reg1=sp+offset;
010 > a_reg1=rp+offset;
011 > a_reg1= T+offset; ***
100   a_reg1=at+offset;
101 > a_reg1=at+offset;
110   a_reg1=rt+offset;
111 > a_reg1=rt+offset;
```

*** особенность выполнения операции **>_t= T+offset;** При выполении необходимо учитывать 12-й бит кода команды (бит h). В данном случае он выполняет функции старшего (знакового) бита смещения, но при работе с регистром T он влияет на значение регистра T. При h==0 (смещение положительное) регистр T имеет вид - T(31:0). При h==1 (отрицательное) регистр T имеет вид - T(15:0,31:16) т.е. старшая и младшая части регистра поменялись местами.

слов : 1

циклов : 1

описание : выполняется операция над содержимым адресного регистра a_reg2 (sp,rp,at,rt,T) и 8-разрядным знаковым смещением. Результат сложения помещается в адресный регистр a_reg1 (at или rt).

пример :
```
/*     A: 3 5 >
/*    sp: 100 >
      >at=sp+#5;
/*     A: 3 5 105>
/*    sp: 100 >
```

6.7.3 Сложение с 16-разрядным индексом в регистре T

синтаксис : a_reg1=a_reg2+t;
a_reg1=a_reg2+t>;

код команды: 010h ---- 1s00 rrrd *

выполнение: a_reg1=a_reg2+t; (для s==0)
a_reg1=a_reg2+t>; (для s==1)

Выполнение полностью аналогично п. 6.7.1. с той особенностью , что в данном случае **offset** равен значению регистра **t** (используются 16 бит , значение регистра рассматривается как знаковое, старшее или младшее слово выбирается в соответствии с битом h.

* особенность выполнения операции **>_t=T+t;** При выполении этой операции необходимо учитывать 12-й бит кода команды (бит h). Он влияет на значение регистра T и на значение индекса в регистре **t**. При h==0 регистр T имеет вид - T(31:0) , а значение индекса **t**=T(15:0). При h==1 регистр T имеет вид - T(15:0,31:16) , а значение индекса **t**=T(31:16).

слов : 1

циклов : 1

описание : выполняется операция над содержимым адресного регистра a_reg2 (sp,rp,at,rt,T) и 16-разрядным знаковым смещением,находящимся в регистре T. При h==0 это биты T[15:0], а при h==1 биты T[31:16]. Результат сложения помещается в адресный регистр a_reg1={at, >at, rt, >rt}.

пример :
```
/*   A: 3 4 >
/*   D: 6 5 >
/*  sp: 100 >
       >at=sp+t>;
/*   A: 3 4 105>
/*   D:   6 >
/*  sp: 100 >
```

6.7.4 Сложение с 16-разрядным смещением

синтаксис : a_reg1=a_reg2+#offset;

код команды: 010h ---- 1001 rrrd
010h ---- 1101 rrrd

выполнение: 1. >t=mem[pc++];
2. a_reg1=rrr+t; D>;

Для кода операции 010h ---- 1001 rrrd кодирование a_reg1, a_reg2 и выполнение определяется в разделе 6.7.2. т.е. a_reg1={at, >at, rt, >rt}, a_reg2={sp,rp,at,rt,T}.

Формат операции 010h ---- 1101 rrrd модифицирует адресные регистры a_reg(at,rt,sp,rp), для (sp,rp) a_reg1= a_reg2, значение d не существенно, а для a_reg2= (at,rt), a_reg1 определяется битом d .

Все адресные модификации регистров sp,rp учитывают ограничения адреса кодируемые в регистре PSW[15:12].

слов : 2
циклов : 2
описание :выполняется операция над содержимым адресного регистра a_reg2 (sp,rp,at,rt,T) и 16-разрядным знаковым смещением. Результат сложения помещается в адресный регистр a_reg1(at, rt, sp, rp).

пример :
```
/*   A: 3 5 >
/*  sp: 100 >
      >at=sp+#5;
/*   A: 3 5 105>
/*  sp: 100 >
```

6.7.5 Операции с регистрами стеков R и A

а) чтение/запись

синтаксис : R["номер_регистра"]=rt;
A["номер_регистра"]=at;

код команды: 010- iiii 1-10 rrrd

выполнение: В зависимости о значения разрядов
rrrd

```
0--0 R[rsp+iiii]=rt;
0--1 A[asp+iiii]=at;
1-00 rt=R[rsp+iiii];
1-01 at=A[asp+iiii];
1-10 R[rsp+iiii]=rt;
1-11 A[asp+iiii]=at;
```

слов : 1
циклов : 1

описание : выполняется операция пересылки(чтения/записи) между памятью стека A или R и соответствующим им верхним регистром at или rt. Необходимо учитывать, что сложение 4-бит индекса с соответствующим указателем стека осуществляется циклически, т.е. если asp=5, а iiii=7, то asp+iiii=5+7=4.

пример :
```
/*  A: 1 2 3 4 >
      at=A[7];
/*  A: 1 2 3 2 >
```

б) сложение с 16-разрядным смещением

синтаксис : a_reg1=a_reg2+#offset;

код команды: 010- iiii 1-11 rrrd

выполнение: 1. >t=mem[pc++];

```
     010- iiii 1-11 rrrd
                    10-- at=AS[asp+iiii];
                    11-- rt=RS[rsp+iiii];
2.   010- ---- 1-11 rrrd
                    1000     rt=at+t>;
                    1001     at=at+t>;
                    1010     >rt=at+t>;
                    1011     >at=at+t>;
                    1100     rt=rt+t>;
                    1101     at=rt+t>;
                    1110     >rt=rt+t>;
                    1111     >at=rt+t>;
```

слов : 2
циклов : 2
описание : выполняется операция над содержимым адресного регистра стека A или R и 16-разрядным знаковым смещением. Результат сложения помещается в адресный регистр at или rt. Индекс **iiii** задает смещение регистра стека относительно текущего указателя стека. (0-последний загруженный операнд, +i – смещение в сторону неиспользованных регистров)

пример :
```
/*  A: 1 2 3 4 >
/*  R: 5 6 >
      >rt=A[7]+#5;
/*  A: 1 2 3 2 >
/*  R: 5 6 7 >
```

6.9 Специальные команды

синтаксис : round/swi/di/ei/byte_l/byte_h/wait/stop;

код команды: 011- -000 -00c cccc

выполнение: В зависимости от ключевых разрядов **c cccc** выполняется

```
- 000-          округление
```

```
                - 001-          программное прерывание
                - 0100          запрещение  прерываний
                - 0111          разрешение  прерываний
                - 1000          выбор младшего байта
                - 1001          выбор старшего байта
                - 1010          STOP (останов)
                - 1011          wait (ожидание)
```

слов : 1
циклов : 1

описание : { round }

Команда округления выполняется над 32-бит регистром T. Действия команды заключаются в подсумировании бита T[15] к старшей половине T[31:16]. Младшая половина не изменяется.

пример :

```
/*   %D: 4 >
/*    D: 8000 >
round;
/*   %D: 5 >
/*    D: 8000 >
```

{ SWI }

Команда SWI вызывает установку 0-го бита регистра запросов прерываний и если в соответствующем бите регистра маски 1 , то произойдет прерывание с соответствующим адрес-вектором.

{ di/ei }

Команда запрещения/разрешения прерываний производит соответственно сброс/установку бита разрешения прерываний(psw[1]).

{ byte_l/byte_h }

Команда выбора младшего/старшего байта используется только для обращения к байту памяти по записи. Она может использоваться в сочетании с командой записи в память и влияет на формирование стробов WRL/WRH соответственно. Между этой командой и следующей за ней командой невозможно прерывание.

пример :

```
/*    D: 3355 >
/*  var: 0 >
byte_l;
*var=t;
/*    D: 3355 >
/*  var: 0055 >
```

{ stop/wait }

Команда останова/ожидания вызывает перевод МП в состояние ожидания запроса на прерывание. В этом состоянии возможно функционирование всех внутренних устройств МК(таймеров, последовательных интерфейсов). При появлении запроса на прерывание МП продолжает выполнение следующей команды если psw[1]=0, либо обрабатывает прерывание если psw[1]=1. Особенность команды останова в том, что на нее не действует сигнал прерывания и из состояния останова МП может выйти только посредством внешнего сброса либо сброса от сторожевого таймера.

6.10 Особенности МП

6.10.1 WORK регистр

WORK есть 32-бит регистр общего назначения. При чтении в регистр T он может быть доступен как младшая (WL) либо старшая (WH) половина, либо как 32-бит регистр WORK. При записи регистр всегда рассматривается как 32-разрядный. Особенности регистра WORK

а) он может участвовать в операции умножения, при этом выполняется операция знакового или беззнакового (определяется битом PSW[3]) умножения WH*WL и результат умножения помещается в регистр T.

б) он может участвовать в операции чтения операнда из внешней памяти в качестве приемника

6.10.2. 32-бит операции с регистром T.

Регистр стека данных T 32-разрядный. Он состоит из двух частей - младшей(t) и старшей (%t). Операция на АЛУ стека данных может выполнятся только над одной из половин регистра T. Однако есть случаи когда в качестве операнда (источник либо приемник) может выступать 32-бит регистр T. Это

а) операция чтения внутреннего регистра (см.6.4.) с битом g=1.

б) операция умножения (см. 6.10.5.)

в) операция двойного сдвига

6.10.3 XMAC регистр

Регистр XMAC может быть использован в операциях умножения с накоплением для отслеживания ситуации переполнения. В командах умножения с накоплением регистр XMAC образует вместе с регистром 38-разрядный аккумулятор. Регистр XMAC выполняет накопление переносов из основного АЛУ МП. Накопление(сложение или вычитание) инициируется выполнением операции сложения/вычитания на старшей половине стека данных

execute = op_add & i(12),
carry = execute & t_c;

Тип операции(сложение или вычитание переноса) определяется выполняемой в АЛУ операцией и знаковым битом регистра T.

```
if (i[9]==1)    -+ = ~t[15];        /*  oper  SUB
        else    -+ =  t[15];        /*  oper  ADD
```

Регистр XMAC доступен по чтению вместе с регистром PSW как старшая половина 32-бит операнда. Запись в регистр XMAC выполняется независимо, соответственно формату команды регистрового обмена.

6.10.4 PSW регистр

Биты:[15,14] - управление циклическим методом адресации регистра SP. Биты определяют размер циклического буфера, адресуемого регистром SP:

00 - 16 Мслов
10 - 64 Кслов
01 - 1 Кслов

11 - 128 слов

[13,12] - управление циклическим методом адресации регистра RP. Биты определяют размер циклического буфера, адресуемого регистром RP:

00 - 16 Мслов
10 - 64 Кслов
01 - 1 Кслов
11 - 128 слов

[11-4] - триггера общего назначения

[3] - тип умножения (1-знаковое/0-кодовое) в операциях с регистром WORK (wh*wl) .
[2] - дополнительный такт ожидания в операциях умножения t*n.
[1] - бит разрешения прерывания
[0] - бит переноса из АЛУ

Разряды [15:1] выполнены триггерами с возможность записи и сброса. Бит [0] – триггер значения переноса из старших разрядов. Сброс все биты устанавливает в [15:1] в 0.

6.10.5 Операция умножения

В отличие от большинства арифметико-логических операций МП, которые выполняются только над 16-бит операндами и дают 16-бит результат, результат операции умножения есть 32-бит число. Если выполняется операция умножения

t*n; t*#; t*%t; %; t*n;%; t*#;

то 32-бит результат помещается в 32-бит регистр T. Причем младшая часть результата в младшую половину T(регистр t), а старшая часть в старшую половину T(регистр %t). Тип умножения (знаковое или беззнаковое) задается кодом команды.

Если выполняется операция умножения над операндами регистра WORK и параллельно операция на младшей половине стека данных t-op-n; W*,
то только младшая часть результата умножения сохраняется в регистре t.

Если выполняется операция умножения над операндами регистра WORK и параллельно операция на старшей половине стека данных %; t-op-n; W*;
то 32-бит результат умножения сохраняется в регистре T.

Нестандартное умножения T=t*%t и T=ut*%ut кодируется форматами T=t*%t : 100- 1101 0010 gggg и T=ut*%ut : 100- 1100 0010 gggg.

6.10.6 BUF регистр

BUF - регистр общего назначения. В операциях с внутренними регистрами он доступен как 16-разрядный регистр через младшую половину регистра T.

Регистр BUF выполняет специфическую функцию в операциях с внешней памятью с участием регистра WORK. При чтении операнда из внешней памяти в младшую половину регистра WORK, считываемый операнд попадает не в регистр WL , а в регистр BUF. При чтении операнда из внешней памяти в старшую половину регистра WORK , операнд попадает в регистр WH, а в регистр WL поступает содержимое регистра BUF. Таким образом регистр BUF выполняет функции буфера при чтении 32-бит данных из внешней памяти в регистр WORK.

6.10.7 Шаг умножения step_*

В п.6.3.2. упоминалось о параллельных операциях gggg, которые выполняются одновременно с операцией на стеке данных. Большинство операций gggg выполняют

действия над адресными регистрами либо стеком адреса/возврата. Выполнение операции **step_*** имеет особенности. Она оказывает влияние на выполняемую в данный момент операцию на АЛУ и стеке данных. Действия ее заключаются в выполнении дополнительной операции двойного сдвига над старшей половиной регистра T и результатом операции на выходе АЛУ. Выполняется правый сдвиг. Сочетание операции сложения с командой **step_*** позволяет реализовать один шаг выполнения операции умножения.

Для корректного выполнения пошаговой операции умножения необходимо чтобы было исходное состояние регистров:

t - ноль
n - множимое
%t - множитель

Операция **t+=n; step_*;** выполняет действия:

а) на АЛУ выполняется сложение t+n;

б) если %t[0]==1 тогда result=t+n;
если %t[0]==0 тогда result=t;

в) выполняется сдвиг вправо регистра %t. Причем в бит 15 регистра %t помещается значение result[0].

г) выполняется сдвиг вправо результата result. Причем в бит 15 result помещается логическое произведение выхода переноса АЛУ и %t[0] (до сдвига). Полученный результат записывается в регистр t..

6.10.8 Шаг деления "step_/"

Аналогично п.6.10.7. **step_/** оказывает влияние на выполняемую в данный момент операцию на АЛУ стека данных. Действия ее заключаются в выполнении дополнительной операции двойного сдвига над старшей половиной регистра T и результатом операции на выходе АЛУ. Выполняется левый сдвиг. Сочетание операции вычитания с командой **step_/** позволяет реализовать один шаг выполнения операции деления. Для корректного выполнения пошаговой операции деления необходимо чтобы исходное состояние регистров было

t - ноль
n - делитель
%t - делимое

Операция **t-=n; step_/;** выполняет следующие действия:

а) на АЛУ выполняется вычитание **t-n** и определяется сигнал переноса из старшего разряда АЛУ (Cf)

б) если Cf==1 тогда result =t-n;
если Cf==0 тогда result =t;

в) выполняется сдвиг влево регистра %t. Причем в бит 0 регистра %t помещается значение Cf.

г) выполняется сдвиг влево результата result. Причем в бит 0 result помещается %t[15] (до сдвига). Полученный результат записывается в регистр t.

6.10.9 Встроенный цикл команд

синтаксис : DO() метка_конца_цикла;

код команды: 0011 1111 ---- -aaa

выполнение: L=mem(pc++);

слов : 1

циклов : 1

описание : команда DO() позволяет организовать циклическое выполнение группы команд (от 1 до 7), заключенной между командой DO() и меткой конца цикла. Особенности выполнения команды:

а) команда DO() должна располагаться в памяти программ по адресу с младшими тремя битами A[2:0] равными 111.

б) тело цикла начинается со следующего адреса где A[2:0]=000.

в) выполнение самой команды DO() соответствует операции NOP с сохранением значения "aaa" кода команды в специальном буфере. В этом случае содержимое буфера соответствует метке конца цикла и при достижении ситуации PC[2:0]=="aaa" происходит проверка значения счетчика CT на равенство нулю:

- если не ноль, выполняется CT-- и биты счетчика команд PC[2:0]=000, т.е. цикл выполняется заново.

- если ноль, сбрасывается признак цикла и продолжается нормальный ход программы

г) при выполнении цикла обслуживание прерывания невозможно.

д) тело цикла не должно содержать команд передачи управления, за исключением команд условного перехода.

е) к моменту выполнения команды DO() счетчик CT должен быть определен.

ж) тело цикла выполняется CT+1 раз.

* з) Последней командой цикла не должна быть if.

При обнаружении команды DO() компилятор автоматически производит выравнивание адресов для правильной компиляции тела цикла.

пример :
```
DO()
  t=*sp++;
  *rp++=t;
end_DO:  t>;
```

Приложение А.

А.Электрические характеристики

Таблица А1

Электрическические характеристики м/с 1881BE1T

V_{CC}=4,5 В, T_{NOM}=25 ±10˚C, T_{EMIN}=-45 ± 3˚C, T_{EMAX}=+85 ±3,

Нормы крайних температур приводены на нижней половине ячейки

Обозначение	Определение	Норма		Размерность	Примечание
		не менее	не более		
V_{CC}	Напряжение питания	3,3	5,5	В	
V_{IH}	Входное напряжение высокого уровня, группа 3.	Vcc-0,8	Vcc	В	*1
V_{IL}	Входное напряжение низкого уровня, группа 3.	0	0,8	В	*2
V_{OH}	Выходное напряжение высокого уровня,	3,5		В	*3
	I_{OL}=-4,0мА,группа 4.	3,2			
V_{OL}	Выходное напряжение низкого уровня,		0,4	В	*3
	I_{OL}=4,0мА, группа 4.		0,45		
I_{IL}	Входной ток низкого уровня ,V_{CC}=5,5В,V_{IL}=0В,		-10	мкА	
	группа 3.		-20		
I_{IH}	Входной ток высокого уровня, V_{CC}=5,5В,V_{IL}=5,5В,	10		мкА	
	группа 3.	20			
I_{OHR}	Ток высокого уровня на резисторе,	0,1	1	мА	
	V_{IL}=0,8В,группа 5.	0,05	1,5		
I_{OHR1}	Ток высокого уровня на резисторе коммутируемым	0,1	1	мА	
	режимом, V_{IL}=0,8В,группа 6.	0,05	1,5		
I_{CC}	Статический ток потребления,		250	мкА	
	V_{CC}=5,5 В;(группы 1,2)		500		
I_{CCO}	Динамический ток потребления		100	мА	
	V_{CC}=5,5 В, Fc= 25 МГц;(группы 1,2)		200		
Обозначение		Номер вывода			
Группа 1	Общие питания	02,27,52,77			
Группа 2	Общие земля	03,28,53,78			
Группа 3	Входы	08-10,70-76,85-89,92, 97-99			
Группа 4	Драйвера с полным выходным током	01,04-07,11-26,29-51, 54-67,80-84,93-96,100			
Группа 5	Ослабленные драйвера I_{OH}	68,69,90,91			
Группа 6	Программируемые драйвера высокого уровня I_{OH}	01,04-07,58-67,100			
Группа 7	Отключаемые.	11-26,32-51,54-57,94-96			

Примечания:
1) Технологическое обеспечение от 0,7Vcc до Vcc+0,3В.
2) Технологическое обеспечеие от -0,3В до 0,2Vcc.
3) Пиковый ток до 50 мА.

А2. Предельно допустимые и предельные режимы.

Таблица А2.1

Характеристики предельно - допустимых и предельных режимов м/с 1881ВЕ1Т

Наименование параметра, единица измерения	обозначение	Предельно допустимый режим		Предельный режим	
		Норма		Норма	
		Не менее	Не более	Не менее	Не более
Напряжение питания, В	V_{CC}	3,3	5,5	-0,5	6,0
Частота следования импульсов тактовых сигналов при V_{CC}=5,0 В, МГц	f_C	-	25	-	-
Частота следования импульсов тактовых сигналов при V_{CC}=3,3 В, МГц	f_{C1}	-	14	-	-
Входное напряжение высокого уровня, В	V_{IH}	Vcc- 0,8	Vcc	-	6,0
Входное напряжение низкого уровня, В	V_{IL}	0	0,8	-0,5	-
Выходной ток высокого уровня, мА	I_{OH}	-	|-4,0|	-	-
Выходной ток низкого уровня, мА	I_{OL}	-	4,0	-	-
Емкость нагрузки, пФ	C_L	-	50	-	-

Приложение Б.

Б. Временные диаграммы.

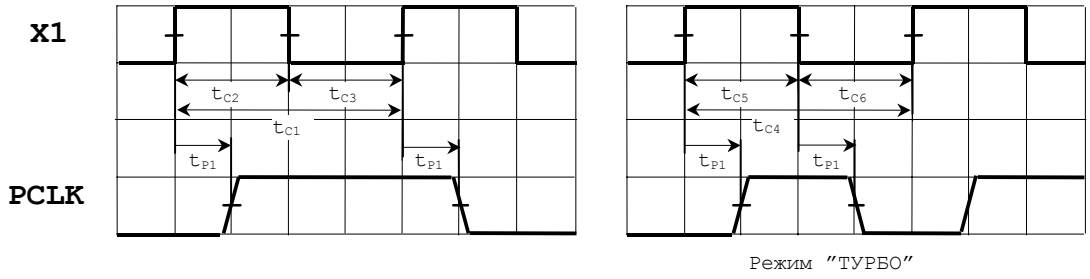


Рис.Б1.1 Тактовый синхросигнал.

Б1.Синхронизация микроконтроллера.

Таблица Б1.1

T_{MIN}=-45 , T_{MAX}=85 , Vcc=(4,5В - 5,5В), C_L=50пФ

Параметр	Назначение	не более	не менее
t_{C1}	Период частоты синхросигнала (F=25мГц),нс		40
t_{C2}	Длительность H импульса,нс		20
t_{C3}	Длительность L импульса,нс		20
t_{P1}	задержка относительно фронта X1,нс	35	25
t_{C4}	Период частоты синхросигнала в ТУРБО режиме,нс		80
t_{C5}	Длительность H импульса в ТУРБО режиме,нс		20
t_{C6}	Длительность L импульса в ТУРБО режиме,нс		40

наименование сигнала	назначение
X1	сигнал частоты тактового генератора
PCLK	сигнал синхронизации процессора

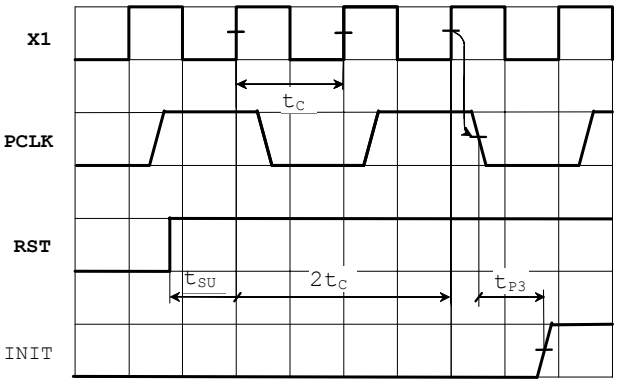


Рис. Б1.2 Пуск микроконтроллера.

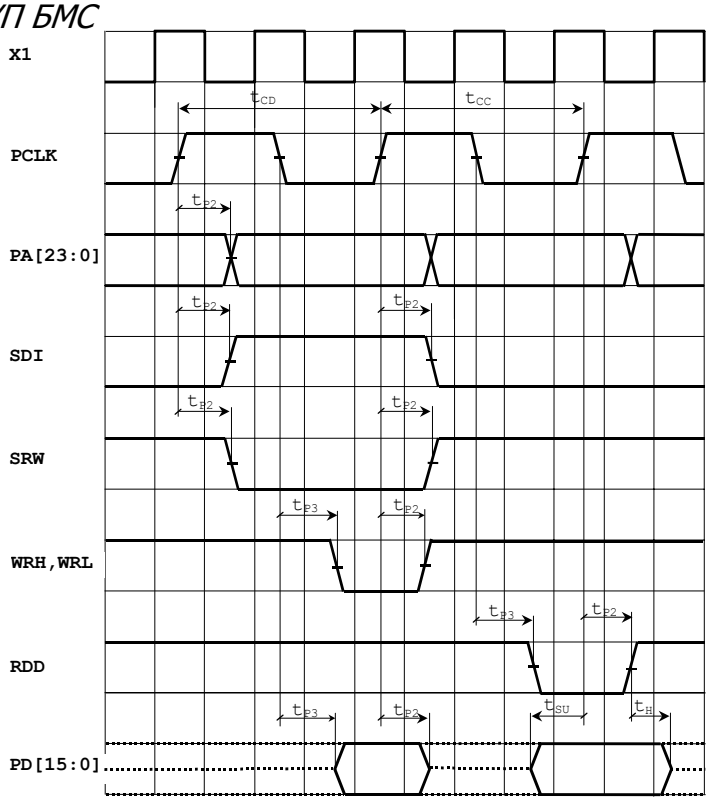


Рис. Б1.3. Исполнение команды с адресацией внешней памяти.

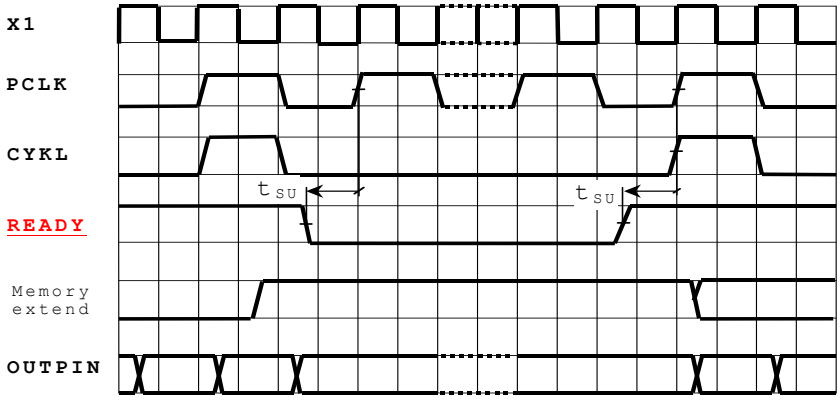


Рис. Б1.4. Остановка процессора сигналом READY при адресации внешней памяти.

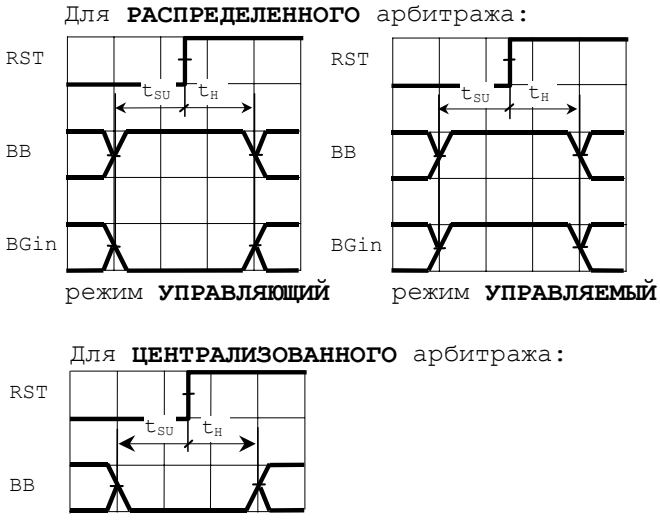
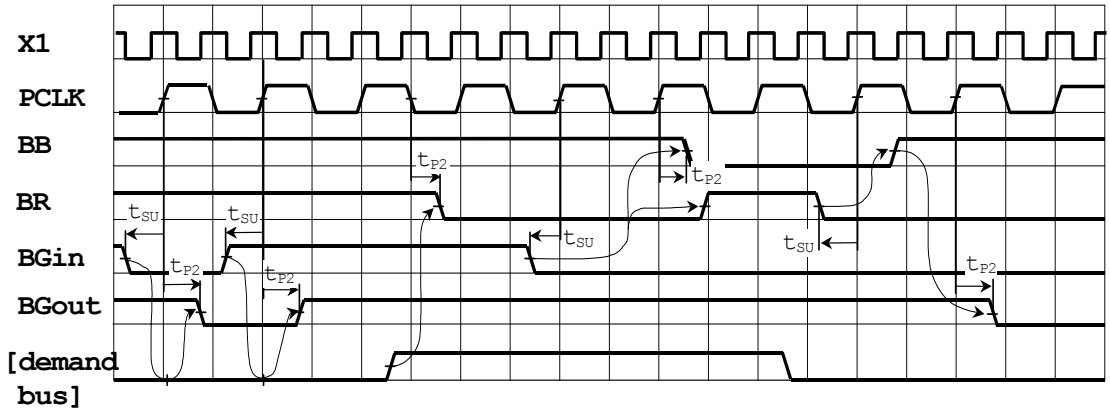


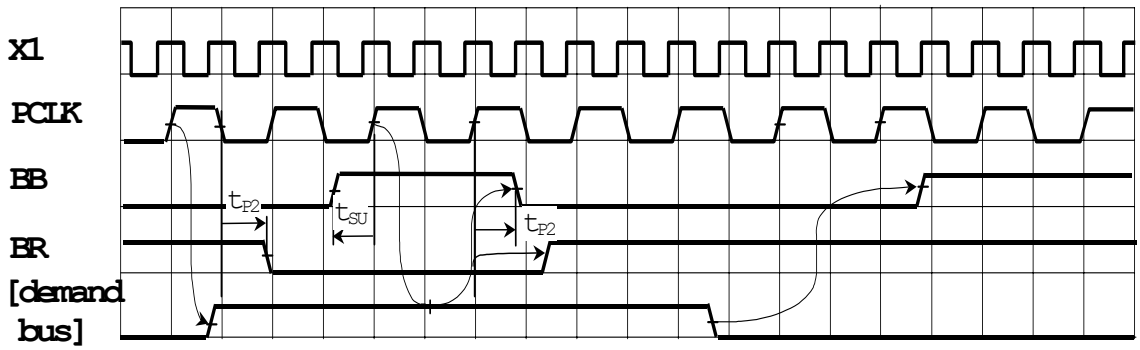
Рис. Б1.5. Установка режима арбитра адресого интерфейса.



demand bus – сигнал запроса внешней памяти

Рис. Б1.6 Распределённый арбитраж с передачей импульсного маркера разрешения
по каскадной цепочке МК (режим управляемого МК -***slave***).

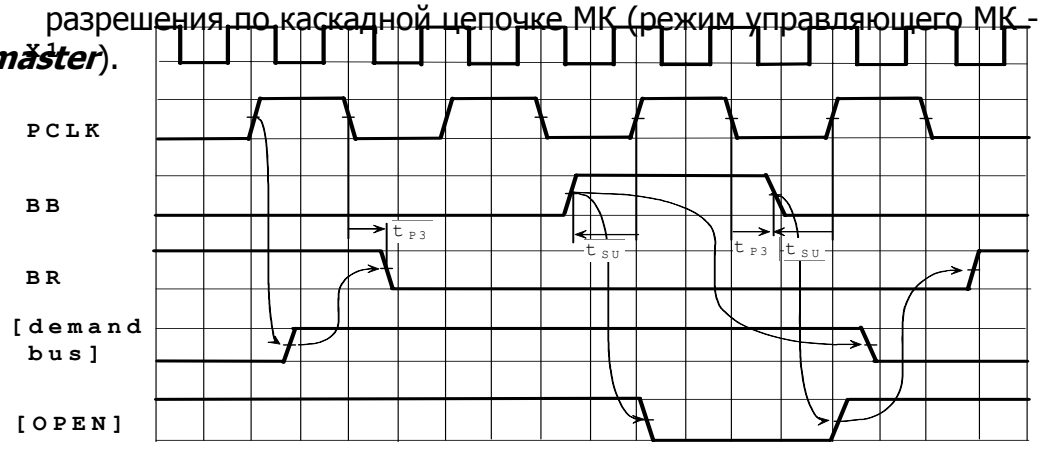
Примечание.
Задержка перехода в 1 на BB BR (управление арбитром адресного порта) определяется емкостью линии. Для ускорения сигнала выходной ток драйверов BB BR (200мкА) усиливается токами внешних элементов.



demand bus – сигнал запроса внешней памяти

Рис. Б1.7 Распределённый арбитраж с формированием импульсного маркера

разрешения по каскадной цепочке МК (режим управляющего МК - ***master***).



demand bus – сигнал запроса внешней памяти;
open – сигнал: порт внешней адресации открыть.

Рис. Б1.8 Централизованный арбитраж порта внешней адресации.

Таблица Б1.2

Обозначение на диаграммах Б1.2 - Б1.8.

имя	назначение
X1	сигнал частоты тактового генератора
PCLK	сигнал синхронизации процессора
RST	сброс МК
INIT	состояние МК
PA[23:0]	адресные линии магистрали внешней памяти
SDI	признак адресации памяти данных
SRW	признак чтение в МК
WRH,WRL	строб записи на внешнюю память
RDD	строб чтения из внешней памяти
PD[15:0]	линии порта данных
READY	останов процессора для ожидание готовности внешнего устройства
[CYKL]	внутренний сигнал синхронизации процессорного цикла
OUTPIN	выхода МК
BB	признак освобождения магистрали
BR	запрос на обращение к внешней памяти
BGin	входной сигнал разрешение активности МК
BGout	выходной сигнал разрешение активности МК
[demand bus]	внутренний сигнал запрос обращения к внешней памяти
OPEN	внутренний сигнал открыть порт для адресации внешней памяти

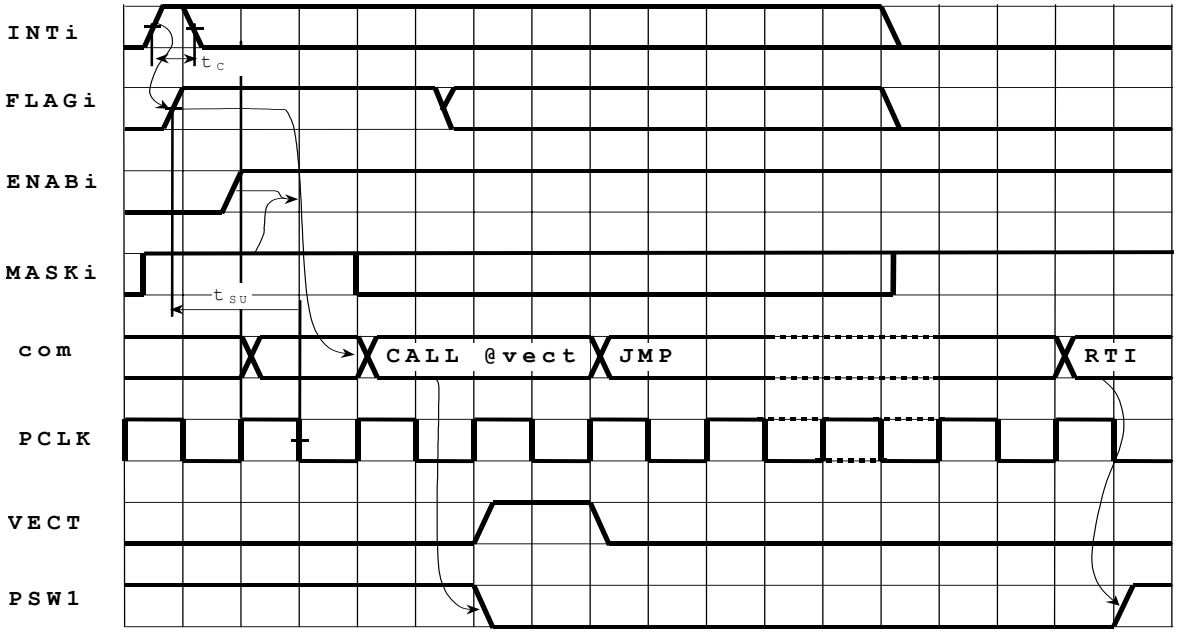


Рис. Б1.9 Сигналы прерывания.

Таблица Б1.3

Обозначение на диаграммах Б2.9

наименование	
INTi	фронт сигнала запрос прерывания (внешний *1, внутренний)
FLAGi	бит регистра флагов
MASKi	бит регистра маски
COM	команда
PCLK	внутренний сигнал синхронизации команды
VECT	внутренний синал чтения адрес вектора прерывания
PSW(1)	бит регистра PSW – разрешение прерывания

примечания:
полярность фронта определяется значениями бит [11-8] регистра CTRL, (1-прерывание инициируется фронтом переключения в 0).

Б2 UART интерфейс.

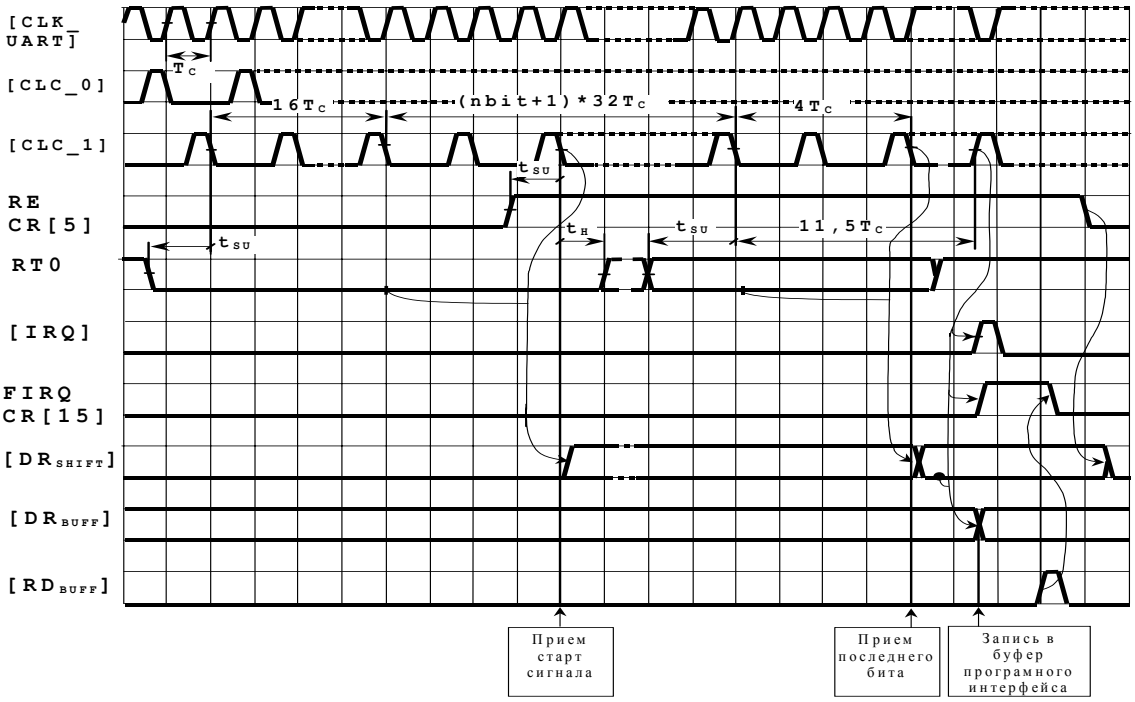


Рис. Б2.1 Прием данных.

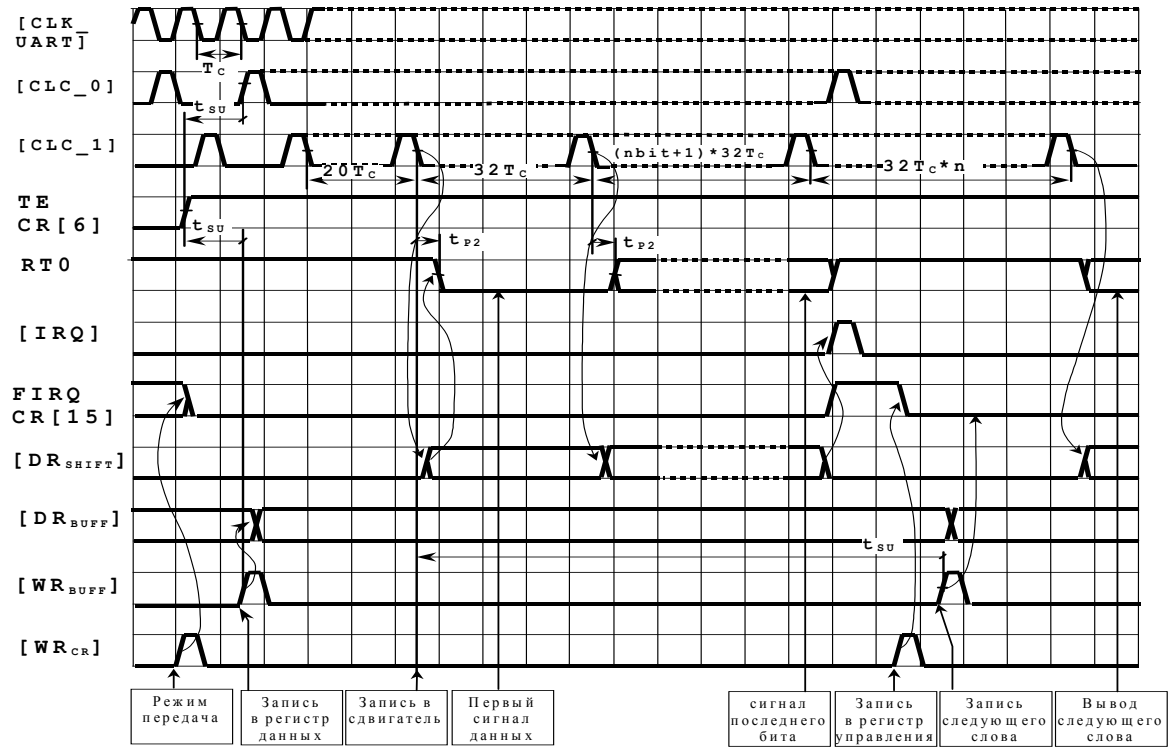


Рис. Б2.2 Передача данных.

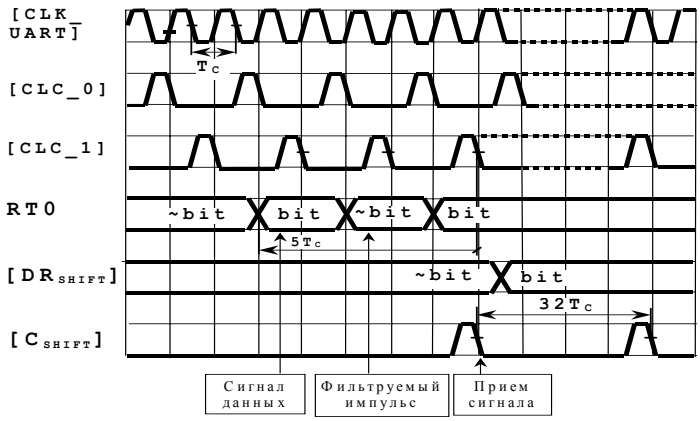


Рис. Б2.3 Фильтрация импульсов.

Таблица Б2.1

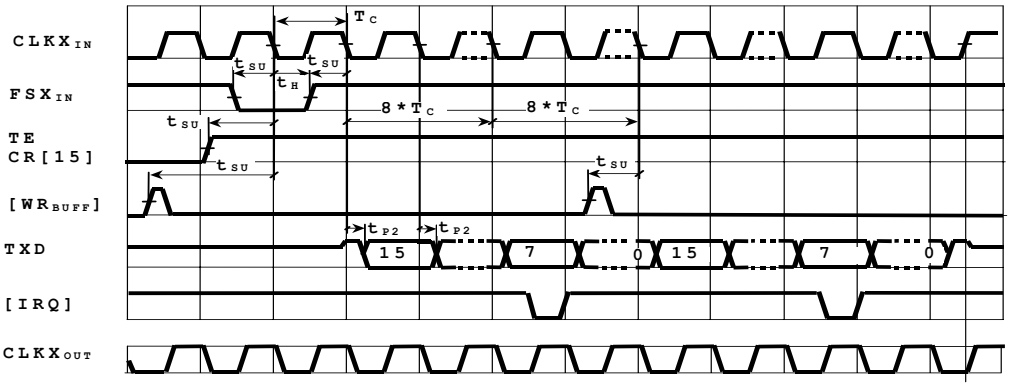
Обозначения диаграмм Б2.1 - Б2.3

Парам.	Назначение	не более	не менее	Прим.
T_C	Период интерфейса UART (устанавливается программно)	$T_{C1}+T_{C1}/18$	$T_{C1}-T_{C1}/24$	1
		$T_{C1}+T_{C1}/20$	$T_{C1}-3T_{C1}/88$	2

примечания:
T_{C1}-Период тактовой синхронизации UART абонента
1) 8 – разрядное слово
2) 9 – разрядное слово

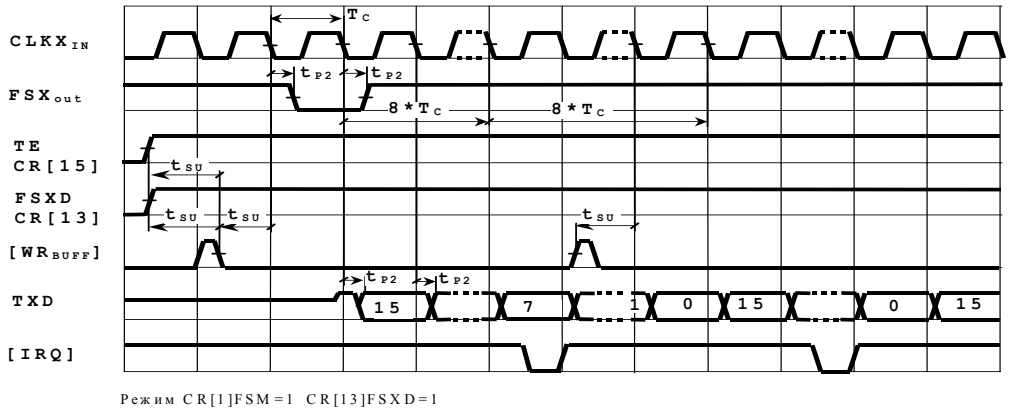
имя	назначение
CLK_UART	Тактирование интерфейса
CLC_0, CLC_1	Разделенные импульсы стробирования (внутренние)
RE,TE	Биты регистра управления. CR[5] прием CR[6] передача.
RT0	Вывод данных интерфейса.
IRQ	Импульс запроса прерывания (внутренний)
FIRQ	Флаг запроса. Бит [15] регистра управления
DR_{SHIFT}	Регистр сдвига данных(внутренний)
C_{SHIFT}	Строб регистр сдвига (внутренний)
DR_{BUFF}	Регистр данных
WR_{BUFF}	Сигнал записи в регистр данных (внутренний)
RD_{BUFF}	Сигнал чтения регистра данных (внутренний)
WR_{CR}	Сигнал записи в регистр управления (внутренний)

Б3. SSI интерфейс.



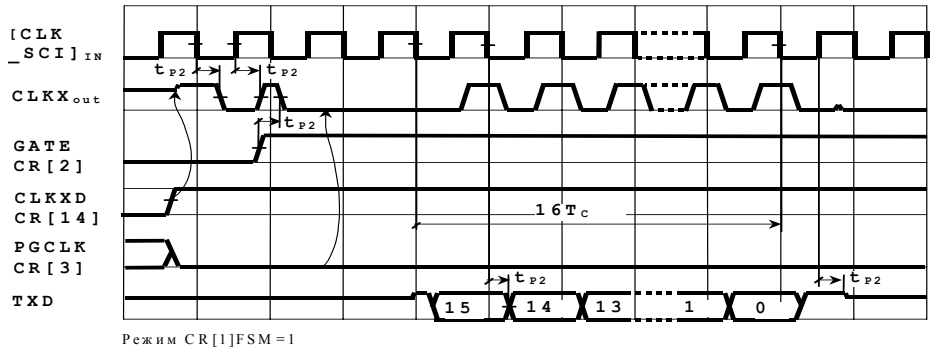
Режимы: CR[1]FSM=1

Рис.Б3.1. Вывод 16 бит в режиме запроса внешним импульсом по входу FSX.



Режим CR[1]FSM=1 CR[13]FSXD=1

Рис. Б3.2 Вывод 16 бит в режиме записи в регистр данных с выводом импульса на FSX.



Режим CR[1]FSM=1

Рис. Б3.3Режим формирования CLKX только при передаче данных.

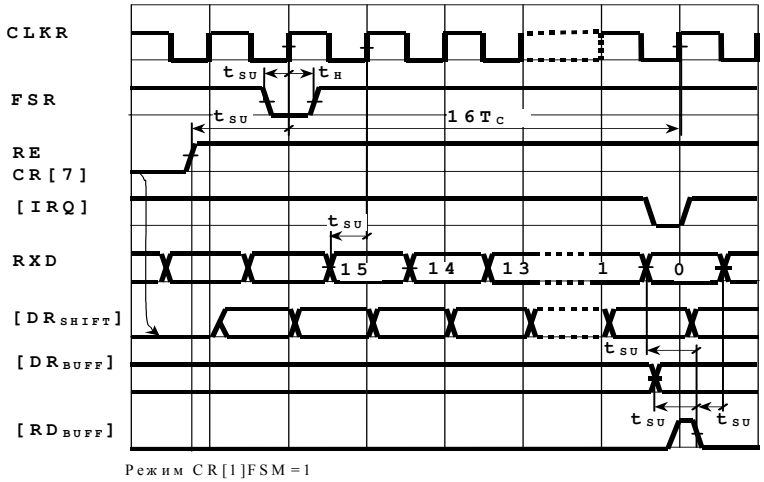


Рис. Б3.4 Прием данных.

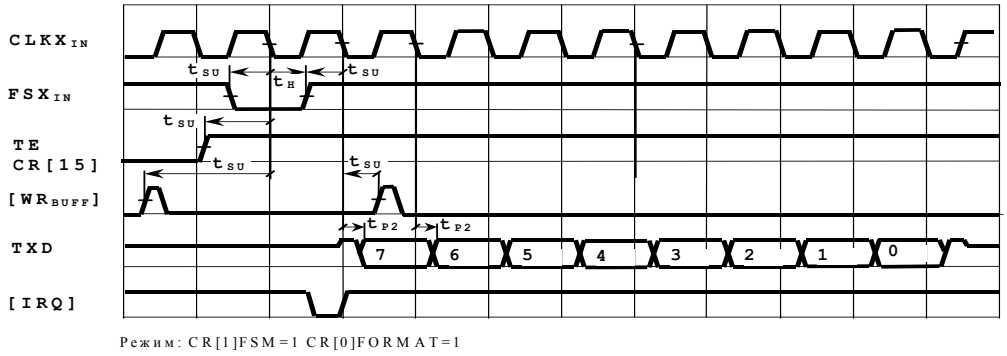


Рис. Б3.5 Вывод байта по запросу внешнего импульса по FSX.

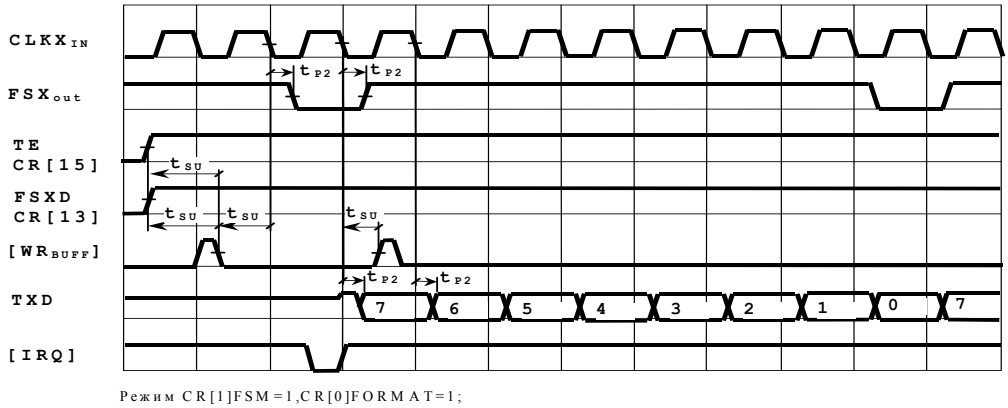


Рис. Б3.6 Вывод байта при записи в регистр данных с выводом импульса на FSX.

Таблица Б3.1

Обозначения диаграмм Б3.1 - Б3.6

имя	назначение
$CLKX_{IN}$, $CLKX_{OUT}$	сигнал тактирования передачи – внешнего формирования и внутреннего
[CLK_SCI]	тактовый сигнал МК
PCLK	сигнал синхронизации процессора
FSX_{IN}, FSX_{OUT}	сигнал пуска передачи - внешнего формирования и внутреннего
FSR	сигнал пуска приема
RXD,TXD	вывода для приема и передачи данных.
RE,TE,GATE CLKXD, PGCLK	биты регистра управления
[IRQ]	импульс запроса прерывания (внутренний)
$[DR_{SHIFT}]$	сдвигатель данных(внутренний)
$[DR_{BUFF}]$	регистр данных (внутренний)
$[WR_{BUFF}]$	сигнал программной загрузки регистра(внутренний)
$[RD_{BUFF}]$	сигнал программного чтения регистра (внутренний)

Б4 I2C интерфейс.

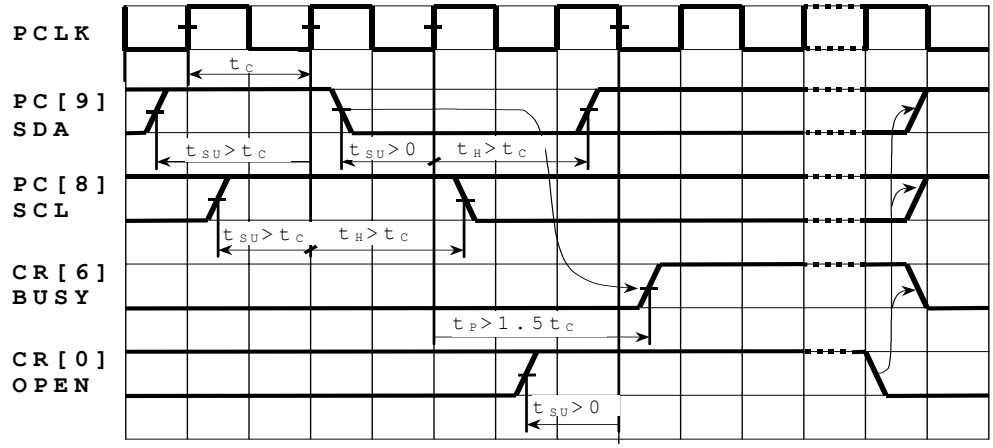


Рис.Б4.1 Прием старт сигнала.

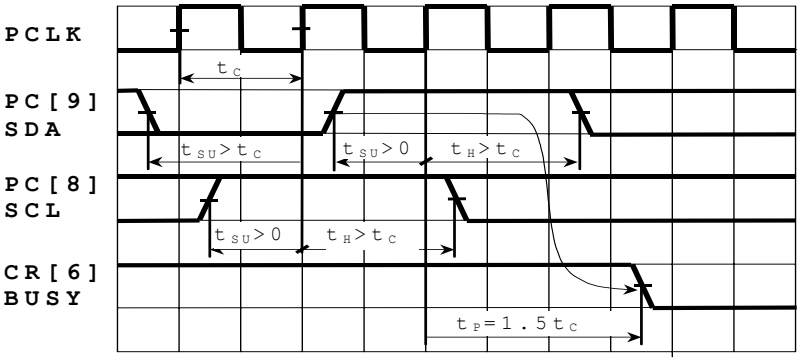


Рис. Б4.2 Прием стоп сигнала.

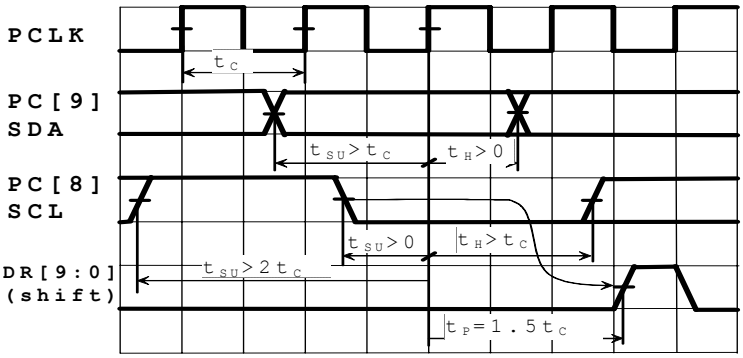


Рис. Б4.3 Прием сигнала данных.

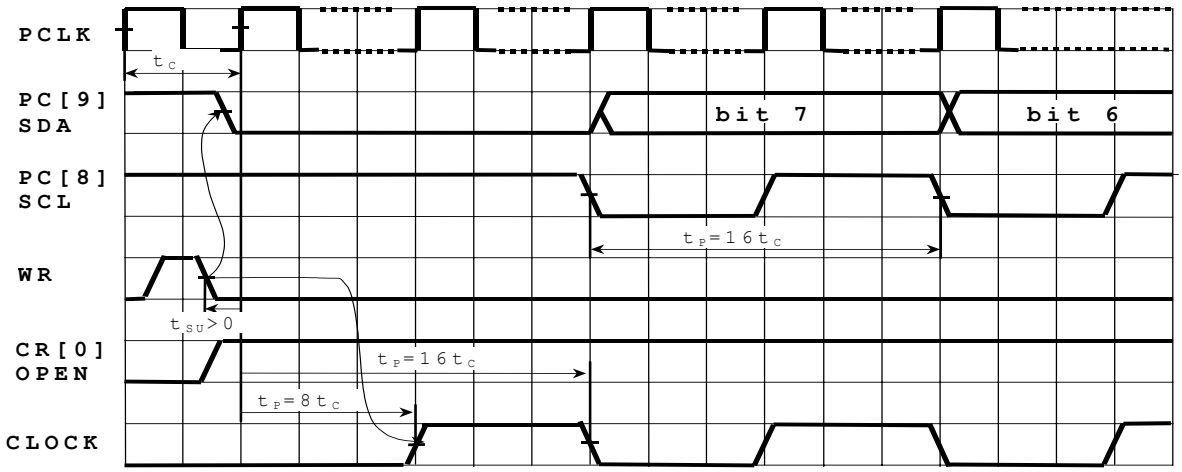


Рис. Б4.4 Старт сигнал в режиме запись байта в регистр данных.

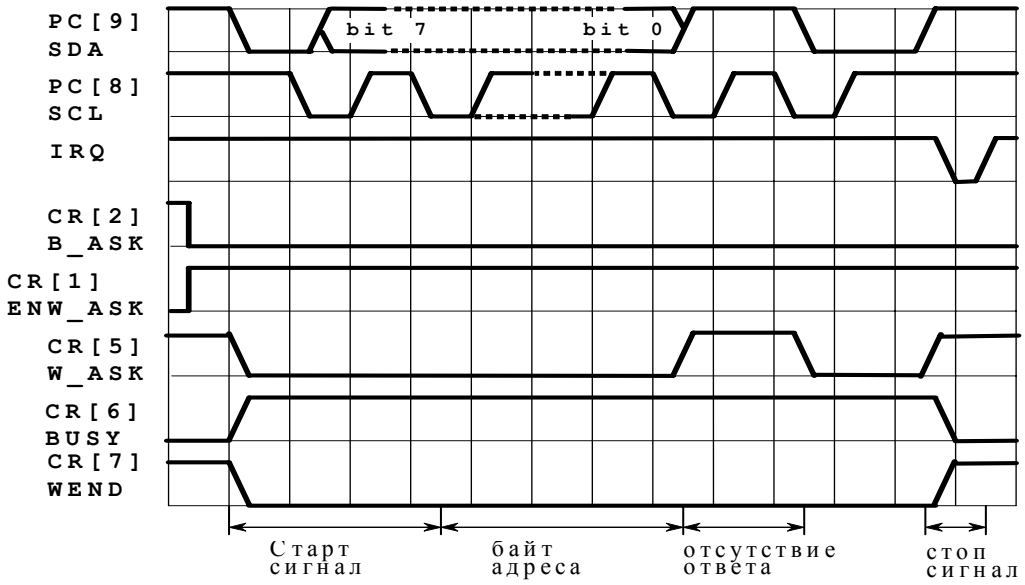


Рис. Б4.5 Формирование стоп сигнала при отсутствии ответа *slave* устройства

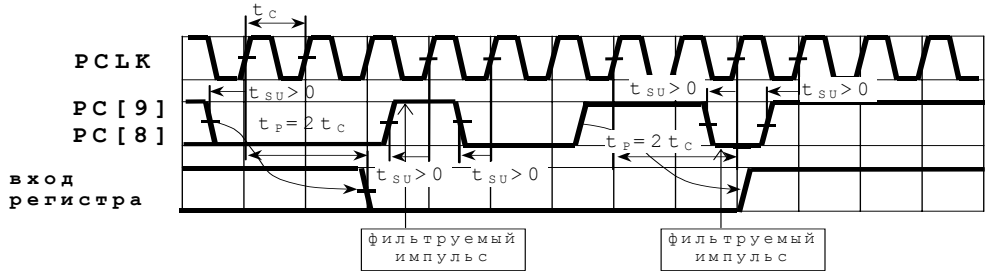


Рис. Б4.6 Фильтрация импульсов.

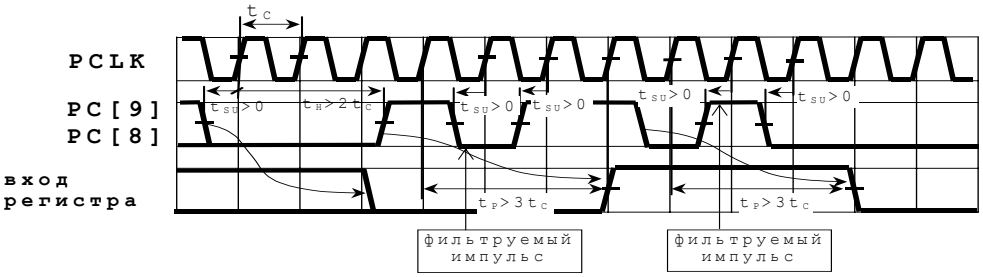


Рис. Б4.7 Фильтрация импульсов при переключении сигнала.

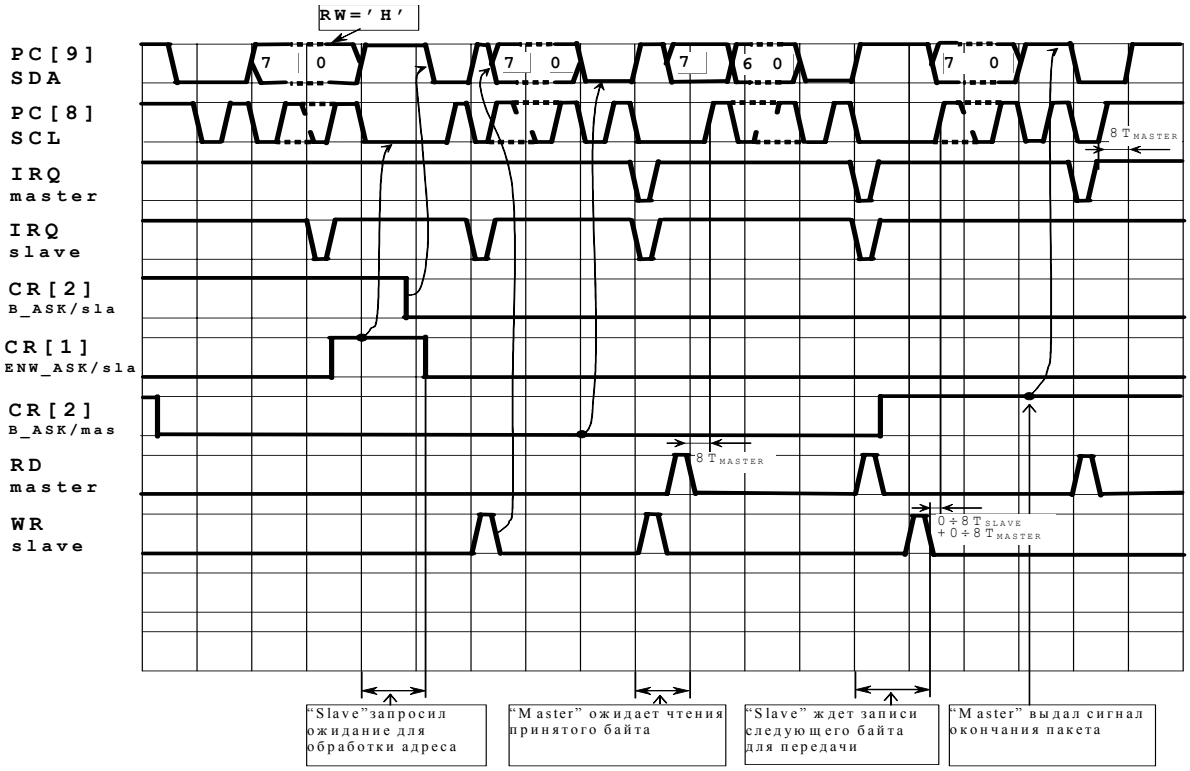


Рис. Б4.8 Прием данных *master* МК.

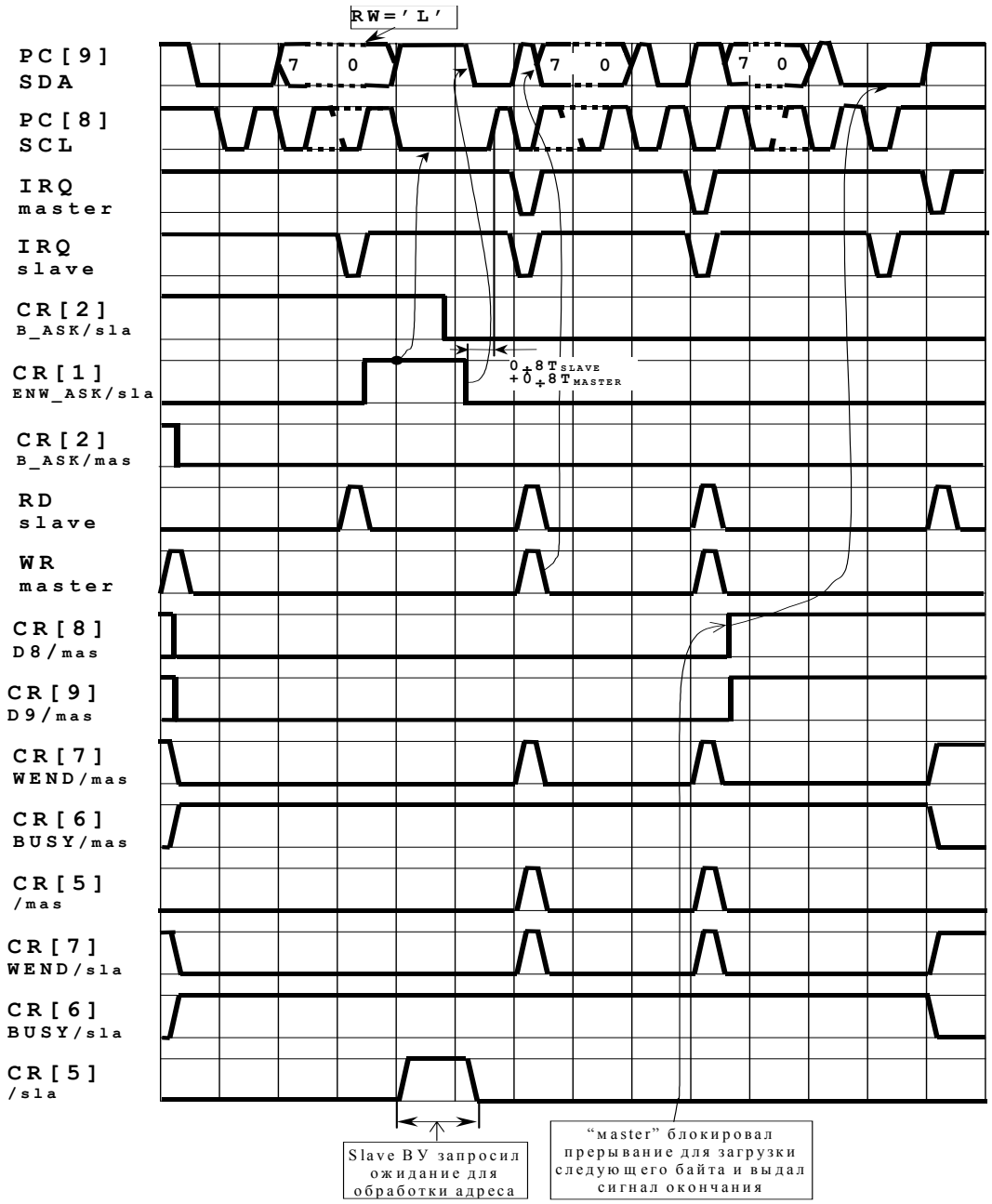


Рис. Б4.9 Передача на *slave* устройство.

Таблица Б4.1

Обозначения диаграмм Б4.1 - Б4.9	
Наименование	
t_C	Период PCLK частоты синхронизации процесора
t_{SU}	Время предустановки, указывается на диаграмме в единицах t_C
t_H	удержание, указывается на диаграмме в единицах t_C
t_P	транспортная задержка указана в единицах t_C, время вывода сигнала сответствует параметру t_{P2}
PCLK	сигнал синхронизации процессора
PC[9]SDA	вывод МК
PC[8]SCL	вывод МК
CR[0]-CR[15]	биты регистра управления
DR[9:0](shift)	внутренний регистр сдвигатель
IRQ	внутренний импульс запроса прерывания по интерфейсу
RD master	внутренний синал чтения регистра данных МК со статусом master
WR slave	внутренний синал записи в регистр данных МК со статусом slave
WR	сигнал записи в регистр данных МК
CLOCK	внутренний сигнал синхронизации обмена данными по интерфейсу.

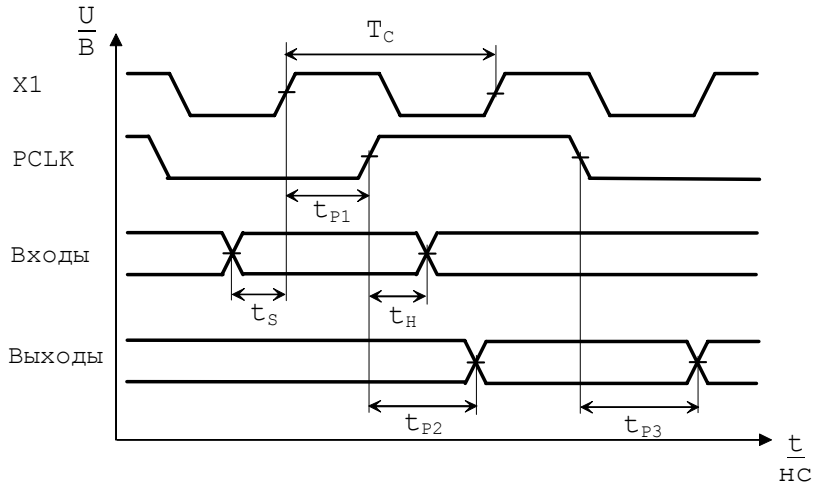


Рис. Б5.1 Динамические параметры.

Б5 Динамические параметры.

Таблица Б5.1.

Динамические параметры м/с 1881ВЕ1Т (“Двина-11”).
VCC=5.5В

Параметр	МИН	МАКС	СР	СКО	Норма	Температура [°С]
	31.67	32.07	31.87	0.28	≤ 40	-50

t_{P1} (нс)	31.67	32.07	31.87	0.28	≤ 40	-50
	33.35	33.55	33.45	0.14	≤ 40	90
t_{P2} (нс)	14.63	29.71	22.17	10.66	≤ 40	-50
	15.9	31.76	23.83	11.21	≤ 40	90
t_{P3} (нс)	7.69	31.72	19.7	17	≤ 40	-50
	8.11	23.45	15.78	10.85	≤ 40	90
t_S (нс)	7	8	7.5	0.7	≤ 10	-50
	7	8	7.5	0.7	≤ 10	90
t_H (нс)	0	0	0	0	≤ 5	-50
	0	0	0	0	≤ 5	90

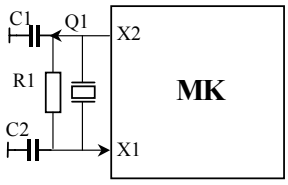
Таблица Б5.2.

Динамические параметры м/с 1881ВЕ1Т (“Двина-11”).
VCC=3.3В

Параметр	МИН	МАКС	СР	СКО	Норма ТЗ	Температура [°С]
t_{P1} (нс)	34.20	34.21	34.205	0.007	≤ 77	-50
	36.56	37.04	36.8	0.34	≤ 77	90
t_{P2} (нс)	18.32	19.66	18.99	0.95	≤ 77	-50
	35.92	38.21	37.065	1.62	≤ 77	90
t_{P3} (нс)	9.71	10.21	9.96	0.35	≤ 77	-50
	25.82	27.33	26.575	1.068	≤ 77	90
t_S (нс)	7	8	7.5	0.7	≤ 10	-50
	7	8	7.5	0.7	≤ 10	90
t_H (нс)	0	0	0	0	≤ 5	-50
	0	0	0	0	≤ 5	90

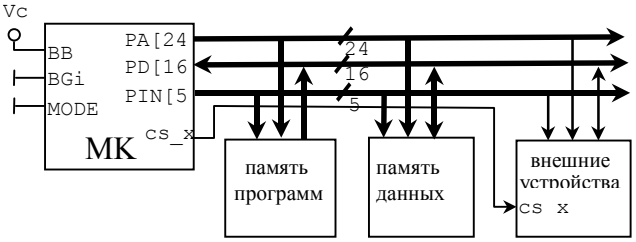
Приложение В.

В. Схемы подключения МК.



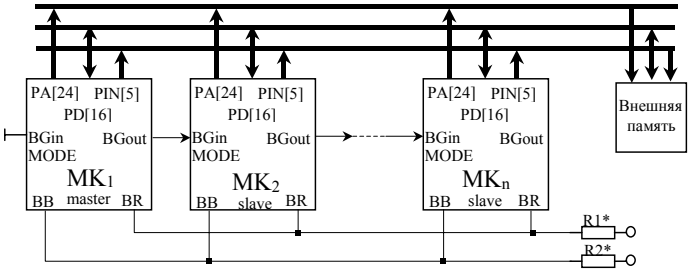
номиналы элементов:
R1=300кОм, C1=C2=10пФ

Рис.В.1.Подключение кварцевого резонатора.



PIN[5] - магистральные сигналы:
SDI – признак данные/команда
SRW – признак направления чтение/запись
WRH,WRL – стробы записи на внешнюю

Рис. В.2. Подключение внешней памяти и периферийных устройств.



PIN[5]- сигналы управления магистрали:
SDI – признак данные/команда
SRW – признак направления чтение/запись
WRH,WRL – стробы записи на внешнюю память
RDD – стробы чтения в МК.

R1,R2 - подстроечные резисторы для ускорения фронта переключения в 1.

Рис. В.3. Подключение МК к магистрали с режимом распределенного арбитра.

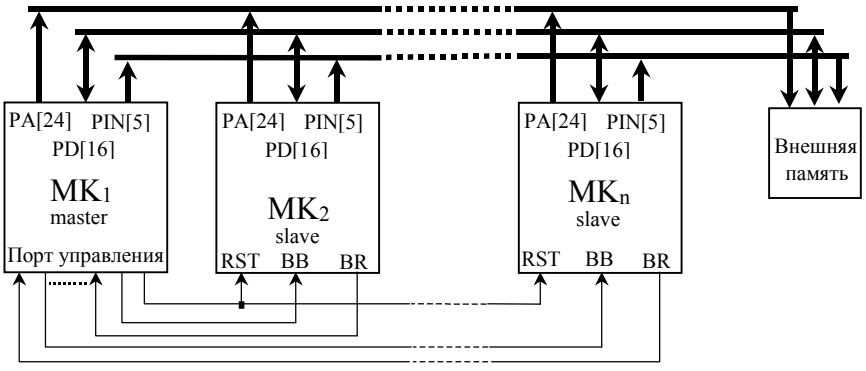
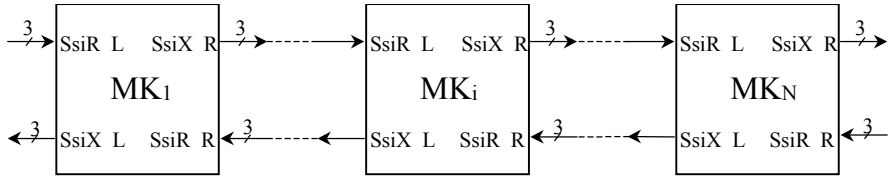
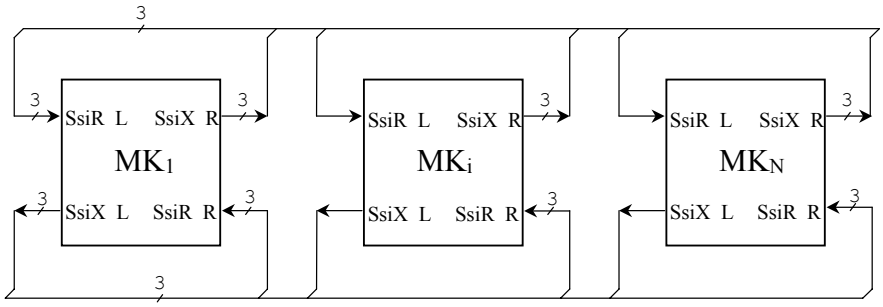


Рис. В.4. Подключение к магистрали с режимом централизованного арбитра.



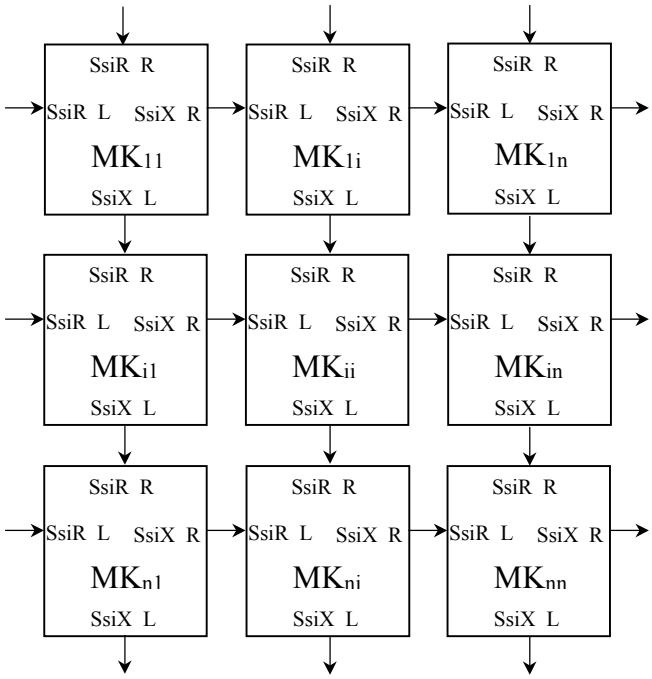
SsiR_L: приемный порт левый CLKR_L, RXD_L, FSR_L
SsiX_L: передающий порт левый CLKX_L, TXD_L, FSX_L
SsiR_R: приемный порт правый CLKR_R, RXD_R, FSR_R
SsiX_R: передающий порт правый CLKX_R, TXD_R, FSX_R

Рис. В.5. Последовательное подключение к SSI с полнодуплексной связью.



SsiR_L: приемный порт левый CLKR_L, RXD_L, FSR_L
SsiX_L: передающий порт левый CLKX_L, TXD_L, FSX_L
SsiR_R: приемный порт правый CLKR_R, RXD_R, FSR_R
SsiX_R: передающий порт правый CLKX_R, TXD_R, FSX_R

Рис. В.6. Параллельное подключение к SSI с полнодуплексной связью.



SsiR_L: приемный порт левый CLKR_L, RXD_L, FSR_L
SsiX_L: передающий порт левый CLKX_L, TXD_L, FSX_L
SsiR_R: приемный порт правый CLKR_R, RXD_R, FSR_R
SsiX_R: передающий порт правый CLKX_R, TXD_R, FSX_R

Рис. В.7. Матричное подключение к SSI.

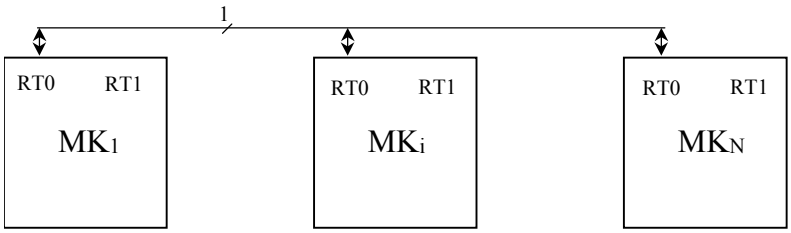


Рис. В.8. Подключение к UART с полудуплексной связью.

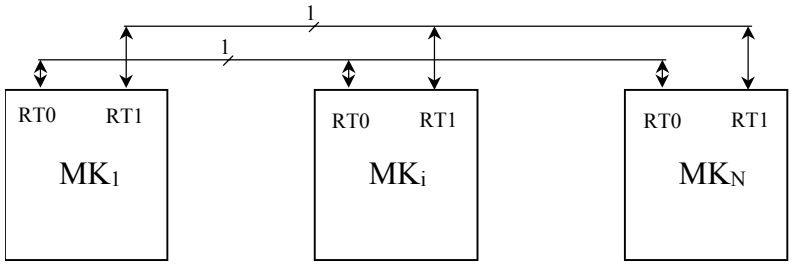


Рис. В.9. Подключение к UART с полнодуплексной связью.

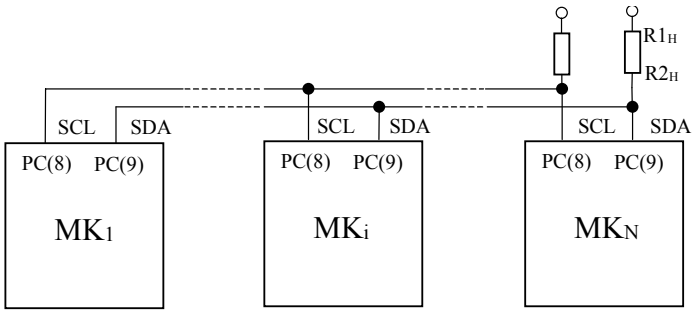


Рис. В.10. Подключение к I2C.

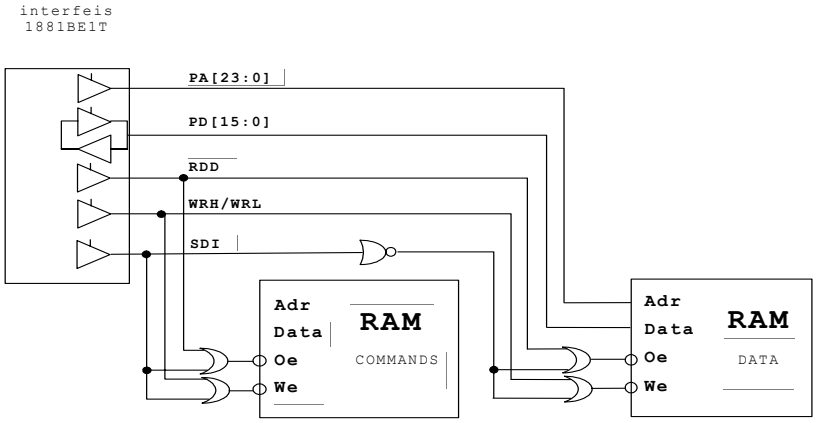


Рис. В11. Подключение микросхем внешней памяти 16 разр. команд и данных к интерфейсу параллельного порта К1881ВЕ1Т. Отключенное состояние выходов RDD, WRH или WRL, SDI, следует определять элементами монтажной логики к логической единице.