МИКРОСХЕМА ИНТЕГРАЛЬНАЯ

1867ВЦ10Т

Руководство пользователя

2015

Содержание

1 Введение

1.1 Краткие характеристики ИС 1867ВЦ10Т

Интегральная схема 1867ВЦ10Т 16-разрядного DSP-микроконтроллера для управления электроприводом разработана по высокопроизводительной статической КМОП технологии с 0,18 мкм нормами. ИС 1867ВЦ10Т конструктивно реализована в 132-выводном металлокерамическом четырехстороннем плоском корпусе типа 4229.132-3.

ИС 1867ВЦ10Т включает центральный процессор ЦП, который имеет систему команд и систему адресации, ориентированную на цифровую обработку сигналов, внутрикристальную память программ и данных с произвольным доступом, электрически перепрограммируемую память EEPROM (Flash), менеджер событий, два аналого-цифровых преобразователя, сторожевой таймер, последовательные интерфейсы SPI и SCI, JTAG (IEEE1149.1), CAN интерфейс, I2C интерфейс, тестовый и отладочный интерфейсы.

Ниже даны краткие характеристики модулей, входящих в состав ИС 1867ВЦ10Т, и характеристики ИС в целом:

1 Центральный процессор с 8,33 нс циклом выполнения команд. Команды ЦП совместимы по исходному коду с командами ИС 1867ВМ2 и совместимы вверх с ИС 1867ВЦ2Т, 1867ВЦ5Т.

2 Память 1867ВЦ10Т реализует три отдельных адресных пространства: адресное пространство программной памяти, памяти данных и I/O (ввода-вывода). Адресное пространство общей памяти ИС 1867ВЦ10Т представлено в разделе 4.1 «Карта памяти».

3 Модуль менеджера событий, который содержит 12 каналов-компараторов или 12 каналов широтно-импульсной модуляции (PWM), три 16-битных таймера общего назначения с шестью режимами, включая непрерывный режим, режимы вверх и вверх/вниз (Continuous, Up, Up/Down), три 16-битных полных сравнивающих устройства с обходом «мертвой» зоны, три 16-битных простых устройства сравнения, четыре устройства сбора данных, два из которых с возможностью интерфейса к квадратурно-кодирующему импульсному устройству (QEP).

4 Два 12-битных аналого-цифровых преобразователя, осуществляющих аналого-цифровое преобразование за 1,25 мкс.

5 Двадцать восемь индивидуально программируемых мультиплексированных вывода I/O.

6 Модуль сторожевого таймера (с сигналом прерывания в реальном времени).

7 Модуль последовательного коммуникационного интерфейса SCI.

8 Модуль периферийного последовательного интерфейса SPI.

9 Шесть внешних сигналов прерывания: Power Drive Protect, Reset, NMI и три маскируемых сигнала прерывания.

10 Четыре режима выключения питания для работы с пониженным энергопотреблением.

11 Модуль эмуляции, основанный на JTAG скан-цепочках.

12 Модуль фазосинхронизирующей подстройки частоты (PLL).

13 Последовательный двухпроводной интерфейс (I2C).

14 Модуль последовательного интерфейса связи CAN.

Характеристики ИС 1867ВЦ10Т приведены в таблице 1.

ИС 1867ВЦ10Т поддерживается инструментальными средствами для разработки систем на ее основе от фирмы Texas Instruments (TI) и третьих фирм. Эти средства включают:

- ANSI C компилятор, Assembler/Linker (ассемблер/редактор связей) и отладчик С-кода (Source Debugger) от TI.
- Эмулятор, основанный на скан-цепочках (Эмулятор-ВЦ10Т).
- Code Composer от TI, поддерживающий TMS320C/F2xx.
- Библиотеку программ для цифрового управления моторами от третьих фирм и поддержку разработки на базе нечеткой логики (Fuzzy-Logic).

Таблица 1 – Характеристики ИС 1867ВЦ10Т ПЦОС контроллера

Наименование параметра, единица измерения	
Формат представления данных	фиксированная запятая
Аппаратный умножитель, бит	16 × 16
Разрядность АЛУ, бит	32
ОЗУ данных, бит	47376 × 16
ОЗУ данных/программы, бит	512 × 16
Объем ПЗУ программ (Flash типа), байт	128 К
Таймер 16-разрядный	3
Сторожевой таймер	1
JTAG-интерфейс для тестирования и отладки	1
Последовательный асинхронный порт (SCI)	1
Последовательный синхронный порт (SPI)	1
CAN интерфейс (мультиплексированный с SPI)	2
I2C интерфейс	1
12-битный АЦП	2
Менеджер событий	1
Число каналов ШИМ	12
Интерфейс внешней памяти	1
Таймер реального времени	1
Число каналов ввода-вывода	28
16-битное устройство полного сравнения	3
16-битное устройство простого сравнения	3
Устройство сбора данных	4
Количество режимов пониженного энергопотребления	4
Внешних сигналов прерывания	6
Производительность, MIPS	120
Напряжение питания ядра, В	1,8 ± 10 %
Напряжение питания периферии, В	3,3 ± 10 %
Значения температуры, ºС	от минус 60 до плюс 85
Тип корпуса	4229.132-3
Количество выводов	132

1.2 Описание ИС 1867ВЦ10Т

ИС 1867ВЦ10Т принадлежит к группе контроллеров цифровой обработки сигналов, основанных на поколении 16-битных процессоров цифровой обработки сигналов с фиксированной запятой (ПЦОС) М1867ВМ1, Л1867ВМ2, 1867ВЦ2Т, 1867ВЦ5Т. Контроллер оптимизирован для применения в системах цифрового управления двигателями.

1867ВЦ10Т сочетает ЦП с модернизированной архитектурой и высокопроизводительными возможностями по обработке данных и набором улучшенных периферийных устройств, которые оптимизированы для управления мотором/двигателем. Периферийные устройства включают в себя модуль менеджера событий, который с таймерами общего назначения и регистрами сравнения способны обслуживать свыше 12 выводов данных формата ШИМ (PWM), а два 12-битных аналого-цифровых преобразователя позволяют осуществлять два одновременных преобразования в течение 1,25 мкс.

На рисунке 1 приведено условное графическое обозначение ИС 1867ВЦ10Т.

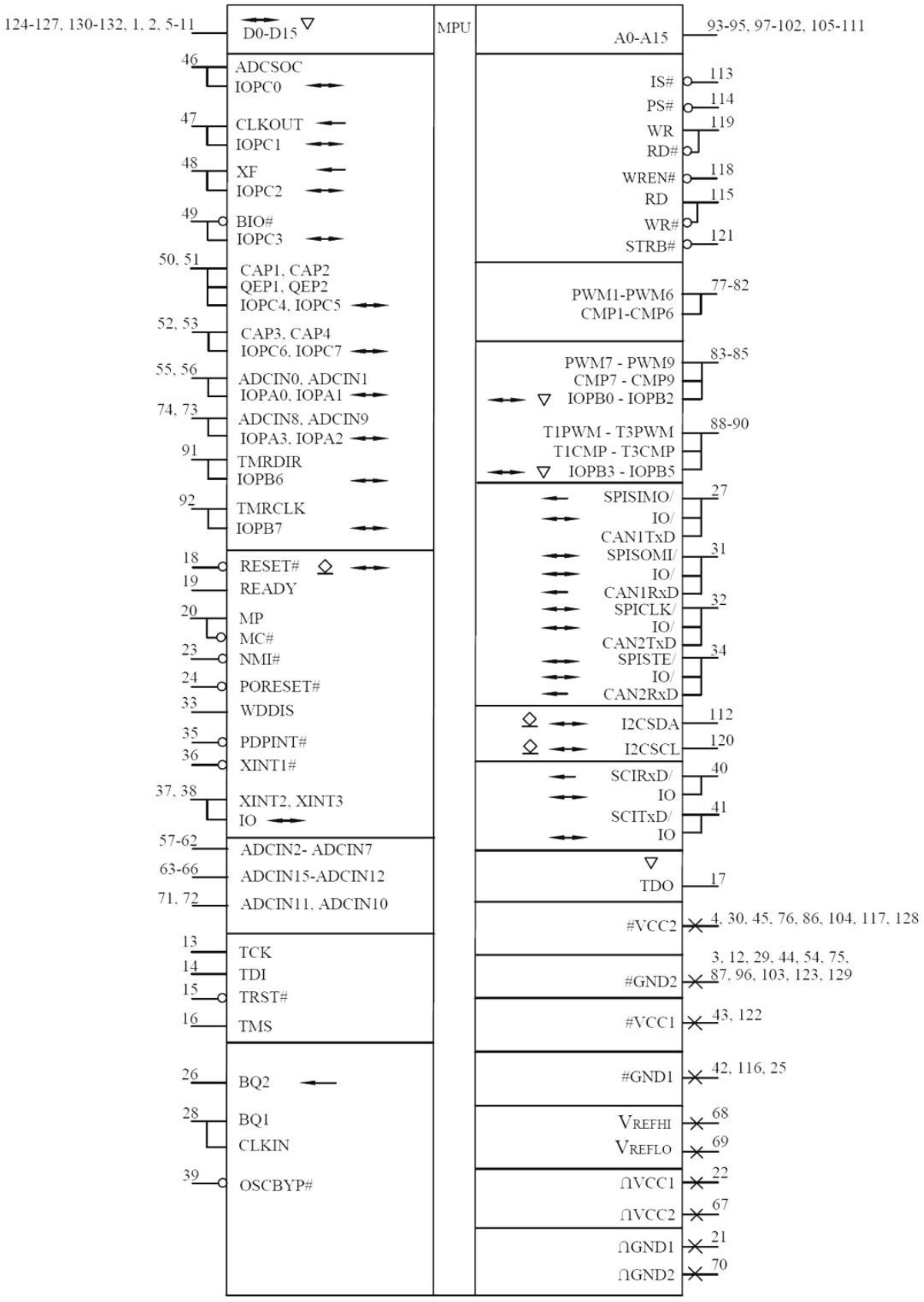


Рисунок 1 – Условное графическое обозначение ИС 1867ВЦ10Т

2 Корпус ИС 1867ВЦ10Т

Микросхема выполняется в металлокерамическом 132-выводном корпусе типа 4229.132-3. Масса микросхем должна быть не более 8,0 г. Конструктивное исполнение ИС 1867ВЦ10Т приведено на рисунке 2.

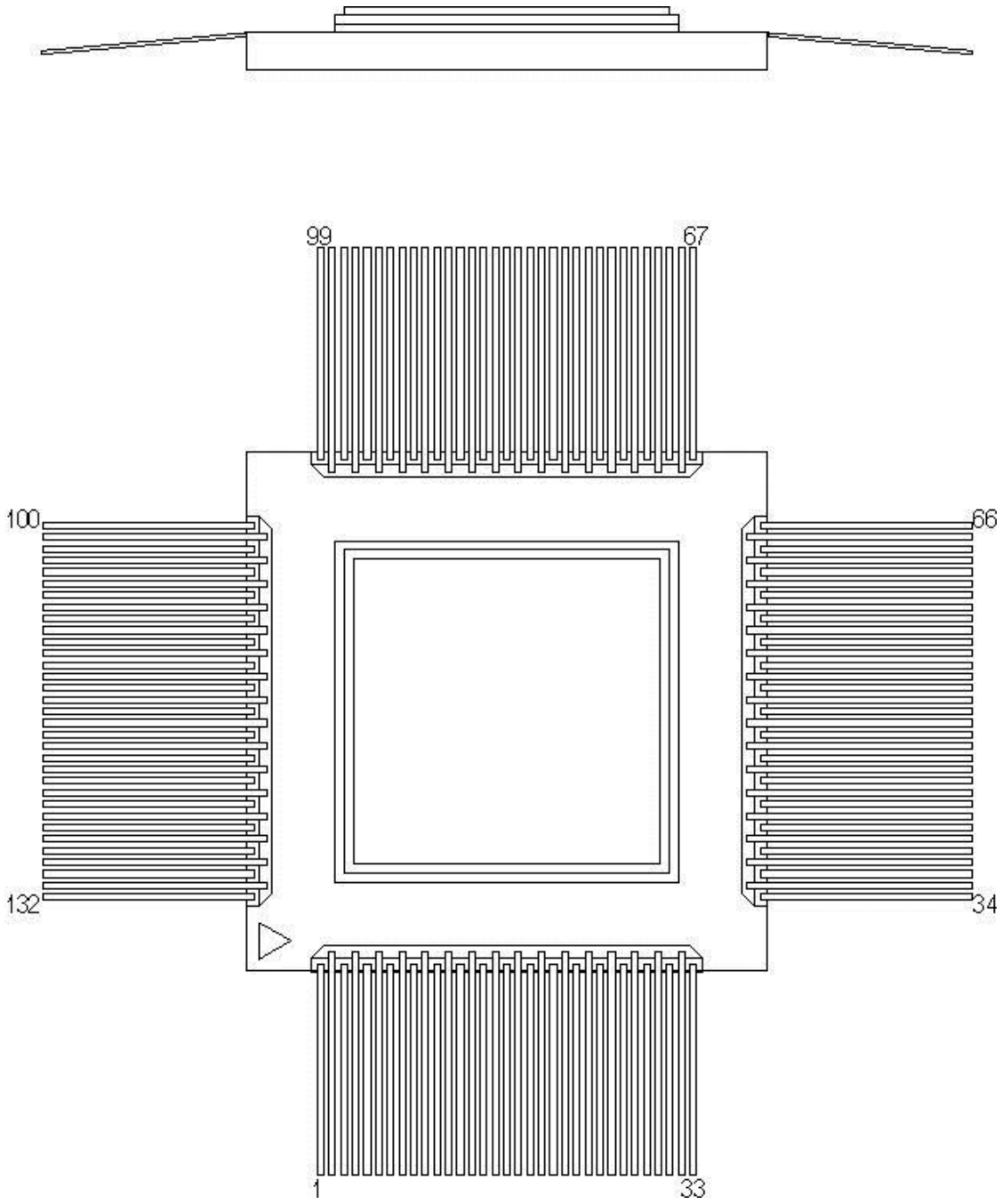


Рисунок 2 – Конструктивное исполнение микросхемы

3 Функциональное назначение выводов ИС 1867ВЦ10Т

Функциональное назначение выводов приведено в таблице 2.

Таблица 2 – Функциональное назначение выводов ИС 1867ВЦ10Т

Номер вывода корпуса	Обозначение вывода	Тип вывода	Функциональное назначение
93 - 95, 97 - 102, 105 - 111	A0 – A15	O	Выводы параллельной шины адреса, разряд 0 - 15
124 -127, 130 -132, 1, 2, 5-11	D0 – D15	I/O/Z	Выводы параллельной шины данных, разряд 0 - 15
13	TCK	I	Вывод типа «pull-up» сигнала тестового тактового входа JTAG
14	TDI	I	Вывод типа «pull-up» сигнала входа тестовых данных JTAG
15	TRST#	I	Вывод типа «pull-down» сброса логики порта тестового доступа JTAG
16	TMS	I	Вывод типа «pull-up» сигнала выбора тестового режима JTAG
17	TDO	O/Z	Выход тестовых данных JTAG
18	RESET#	I/O/2	Сигнал сброса. Двунаправленный цифровой вход/выход с открытым стоком
19	READY	I	Вывод сигнала готовности данных
20	MP/MC#	I	Вывод сигнала выбора режима микропроцессор/микроконтроллер
23	NMI#	I	Вывод типа «pull-up» сигнала входа немаскируемого прерывания
24	PORESET#	I	Вывод сигнала сброса при включении питания
33	WDDIS	I	Вход запрета сторожевого таймера
35	PDPINT#	I	Вывод типа «pull-up» сигнала маскируемого прерывания защиты по питанию
36	XINT1#	I	Вывод типа «pull-up» сигнала внешнего прерывания номер 1
37	XINT2/ IO	I I/O	Вывод типа «pull-up» сигнала внешнего прерывания номер 2/ Вывод двунаправленного входа/выхода общего назначения
38	XINT3/ IO	I I/O	Вывод типа «pull-up» сигнала внешнего прерывания номер 3/ Вывод двунаправленного входа/выхода общего назначения
40	SCIRxD/ IO	I I/O	Вывод асинхронного последовательного входа приема данных/ Вывод универсального двунаправленного входа/выхода
41	SCITxD/ IO	O I/O	Вывод асинхронного последовательного выхода передачи данных/ Вывод универсального двунаправленного входа/выхода
27	SPISIMO/ IO/ CAN1TxD	I I/O O	Вывод SPI подчиненного входа/ Вывод универсального двунаправленного входа/выхода/ Выход передатчика данных интерфейса CAN1
31	SPISOMI/ IO/ CAN1RxD	I/O I/O I	Вывод SPI подчиненного выхода, основной вход/ Вывод универсального двунаправленного входа/выхода/ Вход приёмника данных интерфейса CAN1
32	SPICLK/ IO/ CAN2TxD	I/O I/O O	Вывод SPI тактового сигнала/ Вывод универсального двунаправленного входа/выхода/ Выход передатчика данных интерфейса CAN2
34	SPISTE/ IO/ CAN2RxD	I/O I/O I	Вывод SPI подчиненного сигнала разрешения передачи Вывод универсального двунаправленного входа/выхода/ Вход приёмника данных интерфейса CAN2

Продолжение таблицы 2

Номер вывода корпуса	Обозначение вывода	Тип вывода	Функциональное назначение
112	I2CSDA	I/O/2	Вывод последовательной линии данных. Двунаправленный цифровой вход/выход с открытым стоком
120	I2CSCL	I/O/2	Вывод сигнала запроса шины. Двунаправленный цифровой вход/выход с открытым стоком
39	OSCBYP#	I	Вывод тактового сигнала отключения кварцевого осциллятора
26	BQ2	O	Выход генератора PLL
28	BQ1/CLKIN	I	Вход генератора PLL
46	ADCSOC/ IOPC0	I I/O	Вывод внешнего входа сигнала начала преобразования для АЦП/ Вывод двунаправленного цифрового входа/выхода
47	CLKOUT/ IOPC1	O I/O	Выход тактового сигнала/ Вывод двунаправленного цифрового входа/выхода
48	XF/ IOPC2	O I/O	Выход внешнего флага/ Вывод двунаправленного цифрового входа/выхода
49	BIO#/ IOPC3	I I/O	Вывод типа «pull-up» управления условным переходом Вывод двунаправленного цифрового входа/выхода
50	CAP1/ QEP1/ IOPC4	I I I/O	Вход блока захвата 1/ Вход квадратурного кодирующего устройства 1/ Вывод двунаправленного цифрового входа/выхода
51	CAP2/ QEP2/ IOPC5	I I I/O	Вход блока захвата 2/ Вход квадратурного кодирующего устройства 2/ Вывод двунаправленного цифрового входа/выхода
52	CAP3/ IOPC6	I I/O	Вход блока захвата 3/ Вывод двунаправленного цифрового входа/выхода
53	CAP4/ IOPC7	I I/O	Вход блока захвата 4/ Вывод двунаправленного цифрового входа/выхода
55	ADCIN0/ IOPA0	I I/O	Вывод аналогового входа первого АЦП/ Вывод двунаправленного цифрового входа/выхода
56	ADCIN1/ IOPA1	I I/O	Вывод аналогового входа первого АЦП/ Вывод двунаправленного цифрового входа/выхода
73	ADCIN9/ IOPA2	I I/O	Вывод аналогового входа второго АЦП/ Вывод двунаправленного цифрового входа/выхода
74	ADCIN8/ IOPA3	I I/O	Вывод аналогового входа второго АЦП/ Вывод двунаправленного цифрового входа/выхода
91	TMRDIR/ IOPB6	I I/O	Вывод сигнала управления направлением счета таймеров/ Вывод двунаправленного цифрового входа/выхода
92	TMRCLK/ IOPB7	I I/O	Вывод внешнего тактового входа для таймеров общего назначения/ Вывод двунаправленного цифрового входа/выхода
57 – 62, 63 – 66, 71, 72	ADCIN2-ADCIN7, ADCIN15-ADCIN12 ADCIN11, ADCIN10	I I I I	Выводы аналоговых входов первого АЦП Выводы аналоговых входов второго АЦП Вывод аналоговых входов второго АЦП Вывод аналоговых входов второго АЦП
77 - 82	PWM1-PWM6 CMP1 – CMP6	O O	Выходы ШИМ Выходы компаратора
83	PWM7/ CMP7/ IOPB0	O O I/O/Z	Выход 1 ШИМ/ Выход простого компаратора/ Вывод двунаправленного цифрового входа/выхода

Окончание таблицы 2

Номер вывода корпуса	Обозначение вывода	Тип вывода	Функциональное назначение
84	PWM8/ CMP8/ IOPB1	O O I/O/Z	Выход 2 ШИМ/ Выход простого компаратора/ Вывод двунаправленного цифрового входа/выхода
85	PWM9/ CMP9/ IOPB2	O O I/O/Z	Выход 3 ШИМ/ Выход простого компаратора/ Вывод двунаправленного цифрового входа/выхода
88	T1PWM/ T1CMP/ IOPB3	O O I/O/Z	Выход сравнения первого таймера/ Выход компаратора/ Вывод двунаправленного цифрового входа/выхода
89	T2PWM/ T2CMP/ IOPB4	O O I/O/Z	Выход сравнения второго таймера/ Выход компаратора/ Вывод двунаправленного цифрового входа/выхода
90	T3PWM/ T3CMP/ IOPB5	O O I/O/Z	Выход сравнения третьего таймера/ Выход компаратора/ Вывод двунаправленного цифрового входа/выхода
113	IS#	O	Вывод сигнал выбора пространства ввода-вывода
114	PS#	O	Сигнал выбора программ
119	WR/RD#	O	Сигнал записи-чтения
118	WREN#	O	Сигнал разрешения записи
115	RD/WR#	O	Сигнал чтения/записи
121	STRB#	O	Сигнал стробирующего импульса
4, 30, 45, 76, 86, 104, 117, 128	#VCC2	-	Выводы питания буферов ввода-вывода, 3,3 В
3, 12, 29, 44, 54, 75, 87, 96, 103, 123, 129	#GND2	-	Общие выводы буферов ввода-вывода
43, 122	#VCC1	-	Выводы питания процессорного ядра 1,8 В
42, 116, 25	#GND1	-	Общие выводы процессорного ядра
68	VREFHI	-	Вывод питания опорного напряжения высокого уровня АЦП, (0 - 3,3) В
69	VREFLO	-	Вывод питания опорного напряжения низкого уровня АЦП, (0 - 3,3) В
67	∩VCC2	-	Вывод питания АЦП, 3,3 В
70	∩GND2	-	Общий вывод АЦП
21	∩GND1	-	Общий вывод PLL
22	∩VCC1	-	Питание PLL, 1,8 В

Примечание – В графе “Тип вывода” условно обозначены: I – вход, O – выход, Z – третье состояние, 2 – открытый сток.

4 Краткое описание архитектуры

Структурная блок-схема ИС 1867ВЦ10Т приведена на рисунке 3.

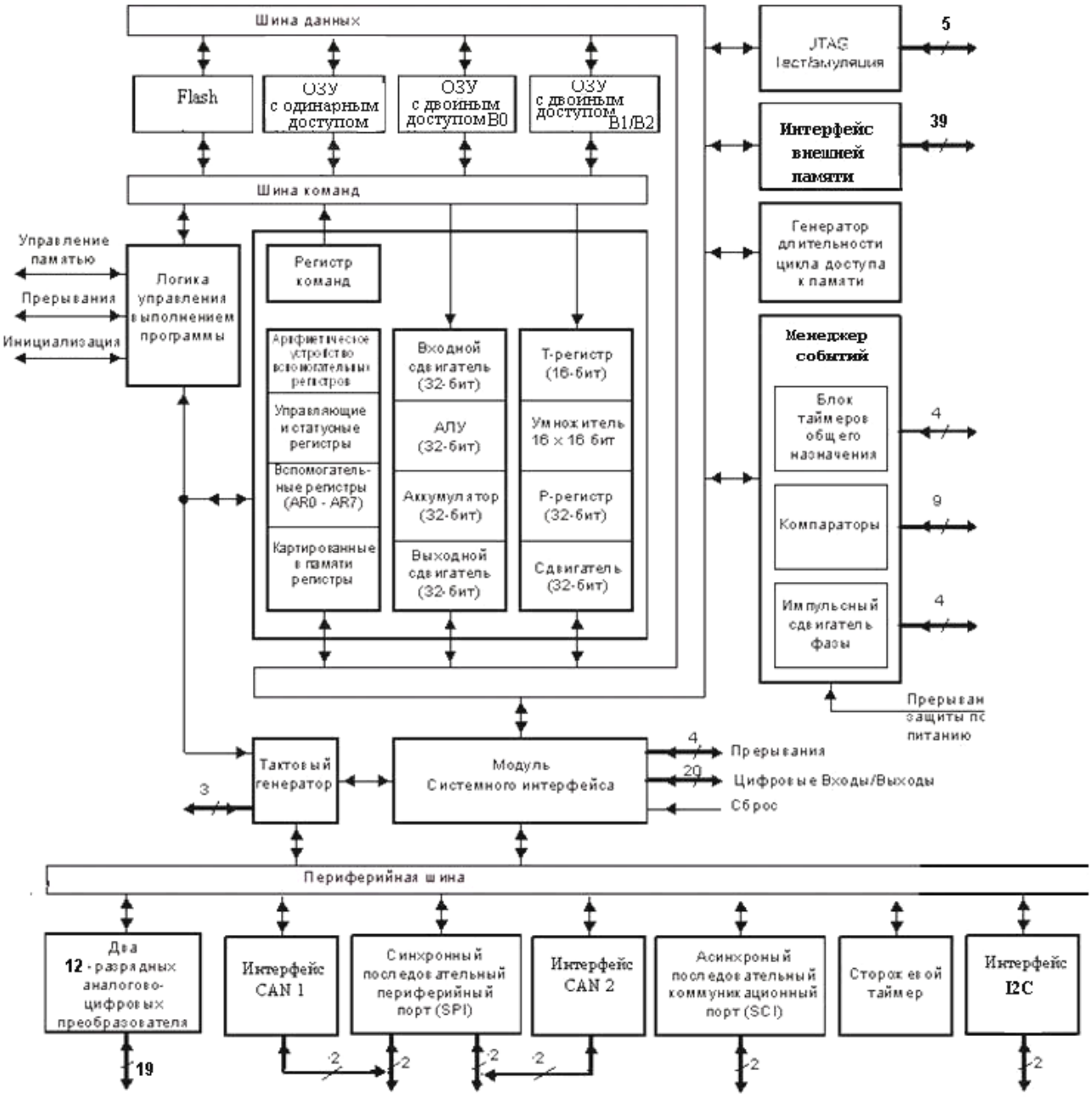


Рисунок 3 – Функциональная схема ИС 1867ВЦ10Т

В функциональной схеме описывается архитектура ИС 1867ВЦ10Т на высоком уровне. Микросхема состоит из трех основных функциональных устройств:

- центрального процессора;
- внутренней памяти;
- периферийных устройств.

В дополнение к этим трем основным функциональным единицам имеется несколько устройств и архитектурных особенностей, описывающих ИС на системном уровне. Они включают в себя: карту памяти, устройство сброса, сигналы прерывания, устройство цифрового ввода-вывода (I/O), генератор синхронизации и энергосберегающие режимы функционирования.

4.1 Карта памяти

ИС 1867ВЦ10Т реализует три отдельных адресных пространства: адресное пространство программной памяти, памяти данных и I/O (ввода-вывода).

ИС 1867ВЦ10Т может адресовать память программ:

- 64К слов внутрикристальной Flash памяти при MP/MC=0;
- 64К слов внешней памяти при MP/MC=1;
- 0,5К слов внутрикристальной DARAM при CNF=1.

ИС 1867ВЦ10Т может адресовать память данных:

- 45К слов ×16 бит внутрикристальной SRAM;
- 1824 слов ×16 бит внутрикристальной DARAM при CNF=0;
- 1312 слов ×16 бит внутрикристальной DARAM при CNF=1.

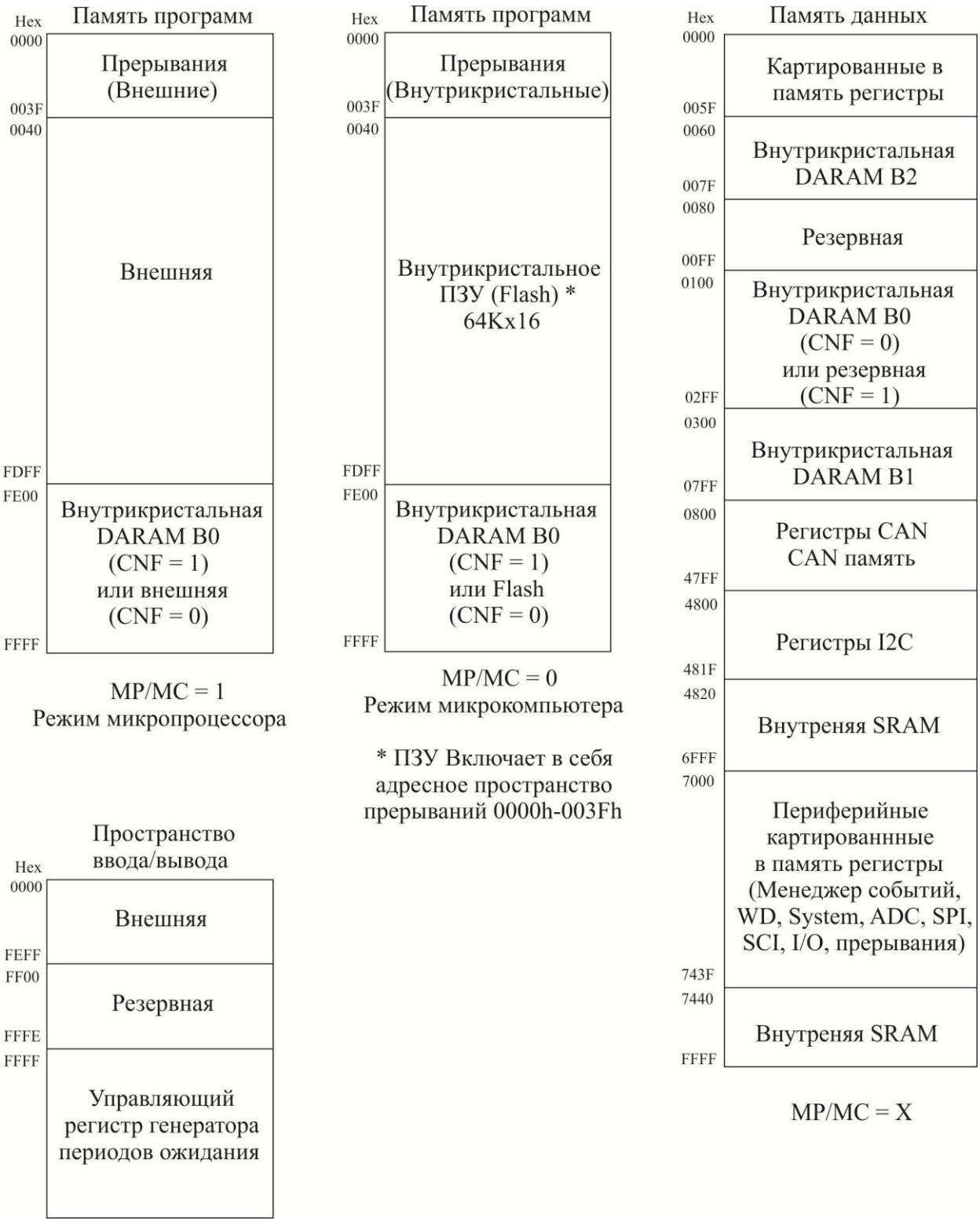
ИС 1867ВЦ10Т может адресовать память пространства ввода-вывода:

- 64К слов внешней памяти.

Первые 96 слов памяти данных (адреса 0–5Fh) предназначены для размещения регистров, картируемых (мапируемых) в памяти данных, или могут находиться в резерве. Это пространство картируемых регистров в памяти данных содержит различные управляющие и статусные регистры, включая регистры ЦП.

Примечание – Термин «картируемый (мапируемый) регистр в памяти данных» означает, что адрес этого регистра находится в области адресов памяти данных.

Все внутрикристальные периферийные устройства ИС 1867ВЦ10Т внесены в карту пространства памяти данных. Доступ к этим регистрам осуществляется командами ЦП, которые адресуют их как любые данные в картируемой памяти данных. На рисунке 4 показана карта памяти ИС 1867ВЦ10Т. Из этого рисунка видно, что карта памяти ИС 1867ВЦ10Т состоит из трех областей памяти – пространства памяти программ, пространства памяти данных и пространства ввода-вывода (I/O).



Hex – шестнадцатеричное значение

Рисунок 4 – Карта памяти устройств ИС 1867ВЦ10Т

4.2 Карта памяти периферийных устройств

Блок периферийных управляющих регистров ИС 1867ВЦ10Т содержит все необходимые данные, которые отражают информацию о состоянии периферийных устройств, и управляющие биты, которые необходимы для выполнения операций с системой и периферийными модулями ИС (включая менеджер событий). На рисунке 5 приведена карта памяти периферийных устройств.

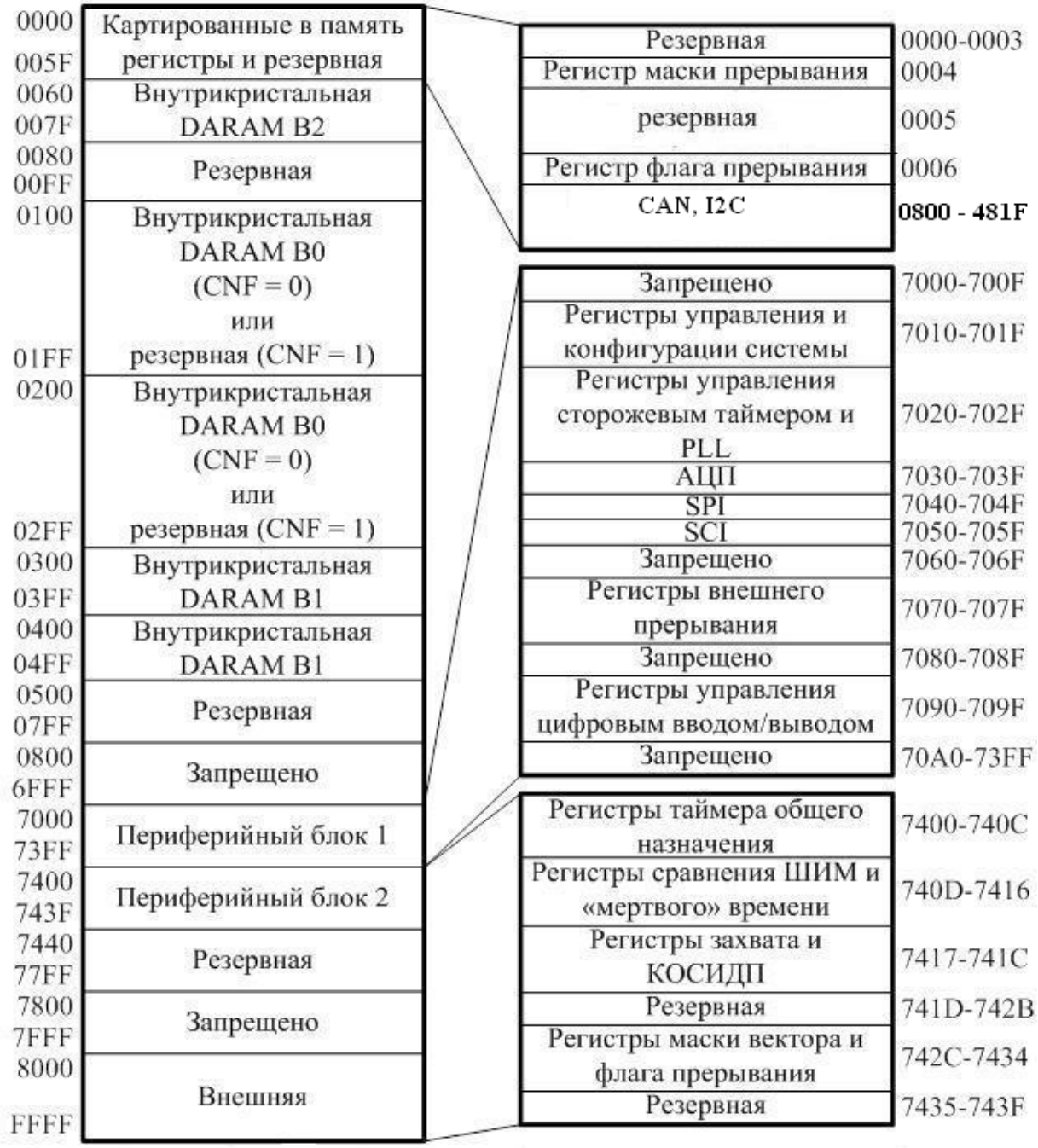


Рисунок 5 – Карта памяти периферийных устройств

4.3 Цифровой ввод-вывод и функции совместно используемых выводов

Двадцать восемь выводов ИС 1867ВЦ10Т могут выполнять как первичные функции (функции периферийных устройств, которые к ним относятся), так и функции цифрового I/O. Эти выводы разделены на 2 группы:

- Группа 1 включает выводы, совместно используемые периферийными устройствами (первичные функции) и модулем цифрового ввода-вывода (цифровой I/O), которые принадлежат портам цифрового ввода-вывода (порту A, порту B и порту C).

- Группа 2 используется периферийными модулями для первичной функции, а также для цифрового ввода-вывода как вторичной функции этих модулей (например, выводы модулей SCI, SPI, выводы внешних сигналов прерывания и выводы модуля синхронизации PLL).

4.3.1 Описание выводов группы 1

Совместно используемая конфигурация выводов группы 1 показана на рисунке 6. Единственным исключением в этой группе является вывод CLKOUT/IOPC1. Каждый вывод управляется тремя битами, которые определяют его работу:

- mux control выбирает функцию вывода между первичной функцией (mux control = 1) и функцией цифрового I/O (mux control = 0);

- I/O direction устанавливает, является ли вывод входом (I/O direction = 0) или выходом (I/O direction = 1), если выбрана функция для вывода цифрового I/O (mux control = 0);

- I/O data – если выбрана функция для вывода цифрового I/O (mux control бит установлен на 0) и выбранное направление является входом для вывода (I/O direction = 0), то данные считываются из этого бита; если же направление для вывода выбрано как выход (I/O direction = 1), то данные пишутся в этот бит.

Mux control бит, I/O direction бит и I/O data бит находятся в управляющих регистрах.

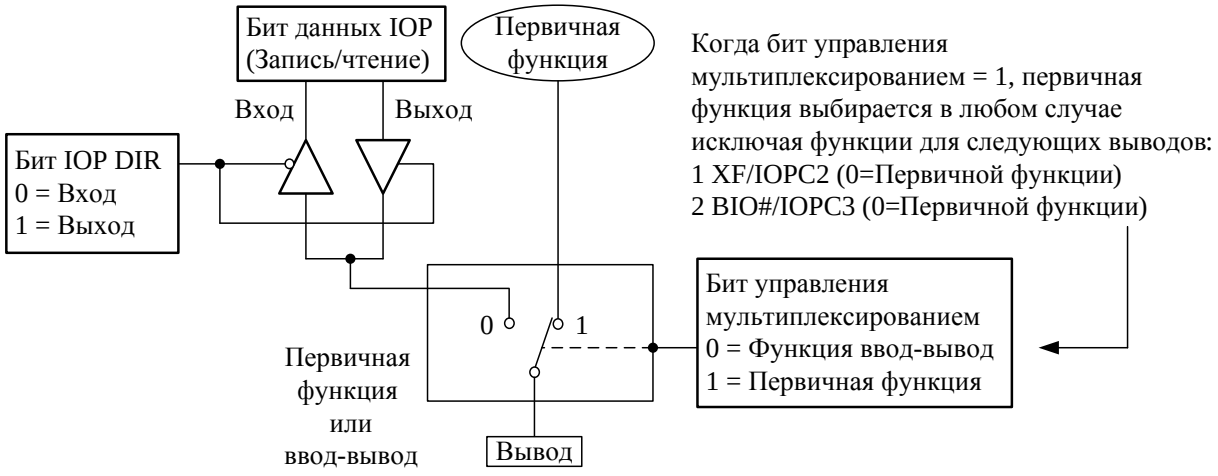


Рисунок 6 – Совместно используемая конфигурация выводов группы 1

Краткое описание конфигурации выводов группы 1 и взаимосвязанных с ней бит показано в таблице 3.

Таблица 3 – Группа 1. Совместно используемые конфигурации выводов

Номер вывода	MUX control регистр	Выбираемая функция вывода		Порт I/O данных*		
		CRx.n=1	CRx.n=0	Регистр	Бит данных	Бит направления
55	OCRA.0	ADCIN0	IOPA0	PADATADIR	0	8
56	OCRA.1	ADCIN1	IOPA1		1	9
73	OCRA.2	ADCIN9	IOPA2		2	10
74	OCRA.3	ADCIN8	IOPA3		3	11
83	OCRA.8	PWM7/CMP7	IOPB0	PBDATADIR	0	8
84	OCRA.9	PWM8/CMP8	IOPB1		1	9
85	OCRA.10	PWM9/CMP9	IOPB2		2	10
88	OCRA.11	T1PWM/T1CMP	IOPB3		3	11
89	OCRA.12	T2PWM/T2CMP	IOPB4		4	12
90	OCRA.13	T3PWM/T3CMP	IOPB5		5	13
91	OCRA.14	TMRDIR	IOPB6		6	14
92	OCRA.15	TMRCLK	IOPB7		7	15
46	OCRB.0	ADCSOC	IOPC0	PCDATADIR	0	8
47	SYSCR.7-6					
	00	IOPC1		PCDATADIR	1	9
	01	WDCLK		–		
	10	SYSCLK		–		
	11	CPCLK		–		
48	OCRB.2	IOPC2	XF	PCDATADIR	2	10
49	OCRB.3	IOPC3	BIO#		3	11
50	OCRB.4	CAP1/QEP1	IOPC4		4	12
51	OCRB.5	CAP2/QEP2	IOPC5		5	13
52	OCRB.6	CAP3	IOPC6		6	14
53	OCRB.7	CAP4	IOPC7		7	15

* Действительно, только если функция I/O выбирается как выход.

4.3.2 Описание выводов группы 2

Краткое описание конфигураций выводов группы 2 показано в таблице 4.

Выводы группы 2 содержат выводы, принадлежащие периферийным устройствам, и используются для первичной функции и функции цифрового ввода-вывода. Управление и конфигурация этих выводов задается установкой подхо-

дящих бит управления и конфигурирования в регистрах периферийных устройств. Таблица 4 содержит список совместно используемых выводов группы 2.

Таблица 4 – Группа 2. Конфигурации совместно используемых выводов

Номер вывода	Первичная функция	Регистр	Адрес	Периферийный модуль
40	SCIRXD	SCIPC2	705Eh	SCI
41	SCITXD	SCIPC2	705Eh	SCI
27	SPISIMO	SPIPC2	704Eh	SPI
31	SPISOMI	SPIPC2	704Dh	SPI
32	SPICLK	SPIPC1	704Dh	SPI
34	SPISTE	SPIPC1	704Dh	SPI
37	XINT2	XINT2CR	7078h	внешний сигнал прерывания
38	XINT3	XINT3CR	707Ah	внешний сигнал прерывания

4.4 Управляющие регистры цифрового ввода-вывода

В таблице 5 приведен перечень регистров, которые содержатся в модуле цифрового входа-выхода. Так же как и у других периферийных устройств, эти регистры находятся в памяти данных.

Таблица 5 – Адреса управляющих регистров цифровых I/O

Адрес	Регистр	Наименование
7090h	OCRA	I/O MUX управляющий регистр A
7092h	OCRB	I/O MUX управляющий регистр B
7098h	PADATADIR	Регистр порта I/O A и направления передачи
709Ah	PBDATADIR	Регистр порта I/O B и направления передачи
709Ch	PCDATADIR	Регистр порта I/O C и направления передачи

4.5 Устройство сброса и сигналы прерывания

Программно управляемая структура прерываний ИС 1867ВЦ10Т поддерживает гибкую структуру внутрикристальных и внешних сигналов прерывания для обеспечения требований реального времени при их использовании. ИС 1867ВЦ10Т различает три вида прерываний:

- Сброс (Reset). Этот вид прерывания определяет аппаратную или программную инициализацию. Он не управляется от ЦП и имеет наивысший приоритет перед другими выполняемыми функциями. Все замаскированные сигналы прерывания блокируются до тех пор, пока обслуживающая программа «Сброса» не разблокирует их.

- Сигналы прерывания от аппаратуры. Этот вид прерывания вызывается внешними выводами или внутрикристальными периферийными устройствами.

- Внешние сигналы прерывания. Этот вид прерываний генерируется одним из пяти внешних выводов, которые соответствуют сигналам прерывания XINT1, XINT2, XINT3, PDPINT и NMI.

Первые четыре прерывания могут быть маскированы как выделенными маскирующими битами, так и через регистр маски ЦП (IMR), который может блокировать любую из маскированных линий сигналов прерывания ЦП.

Прерывание от NMI, которое не маскируется, имеет более высокий приоритет перед сигналами прерывания от периферийных устройств и сигналами прерывания от программы. Оно может быть блокировано только уже существующим NMI или прерыванием «Сброс».

Сигналы прерывания от периферийных устройств - это внутренние сигналы, которые инициируются внутрикристальными периферийными модулями, такими как менеджер событий, SPI, SCI, I2C, CAN, сигналы прерывания от сторожевого таймера и таймера реального времени (таймеры WD/RTI) и АЦП.

Они могут быть маскированы либо через маскирующие биты для каждого события в каждом из периферийных устройств, либо через регистр IMR, который может маскировать каждую маскированную линию сигнала прерывания ядра ИС.

4.6 Программно-генерируемые прерывания

4.6.1 Команда INTR

Эта команда позволяет инициировать любое прерывание ИС 1867ВЦ10Т программным обеспечением. Ее операнд указывает, к какой ячейке вектора прерывания переходит ЦП. Эта команда глобально блокирует маскируемые сигналы прерывания (устанавливает INTM бит в 1).

4.6.2 Команда NMI

Эта команда заставляет перейти ЦП к вектору с адресом 24h. Эта же ячейка используется для немаскируемого аппаратного сигнала прерывания NMI.

Прерывание NMI может инициализироваться через вывод NMI# (низкий уровень) или выполнением команды NMI. Эта команда глобально блокирует маскируемые сигналы прерывания.

4.6.3 Команда TRAP

Эта команда заставляет ЦП перейти к ячейке с 22h. Команда TRAP не блокирует маскируемые сигналы прерывания (INTM не устанавливается в 1); поэтому, когда ЦП переходит к стандартной программе обработки этого прерывания, то эта стандартная программа может быть прервана маскируемыми аппаратными сигналами прерывания.

4.6.4 Системное прерывание для целей эмуляции или программной «ловушки»

Этот сигнал прерывания может быть сгенерирован командами INTR или TRAP.

4.6.5 Сброс

Операция сброса обеспечивает упорядоченную последовательность запуска ИС. Существует пять возможных событий, вызывающих сброс (см. рисунок 7). Три из этих событий сформированы внутри ИС, а два других события – это внешние сигналы RESET# и PORESET# на соответствующих выводах.

Рисунок 7 – Блок-схема формирования сигнала «Сброс»

Существует пять возможных сигналов сброса:

- Сброс от сторожевого таймера. Сброс от сторожевого таймера происходит, если сторожевой таймер переполнился или было записано неправильное значение либо в регистр ключа, либо в управляющий регистр сторожевого таймера. Когда ИС включается, то сторожевой таймер автоматически активизируется.

- Сброс, генерируемый программным обеспечением. Он формируется от системного управляющего регистра (SYSCR). Очистка/установка в исходное состояние RESET0 бит (бит 14) или установка RESET1 бит (бит 15) является причиной формирования системного сброса.

- Сброс по обращению к нелегальному адресу. Система и блок периферийных управляющих регистров содержит нереализованное адресное пространство, маркируемое как нелегальное. Любое обращение к ячейке, расположенной в нелегальных адресах, вызывает сброс по нелегальному адресу.

- Сброс по выводу RESET#. Для создания внешнего импульса сброса по выводу RESET# (импульс низкого уровня) этот импульс должен быть не менее одного цикла SYSCLK. Это необходимо для обеспечения того, чтобы ИС распознало сигнал сброса по этому выводу.

- Формирование сброса по выводу PORESET#. Для формирования сброса по выводу PORESET# (импульс низкого уровня) должно быть не менее одного цикла SYSCLK. Это необходимо для обеспечения того, чтобы ИС распознало сигнал сброса по этому выводу.

Как только источник сброса активизирован, внешний вывод RESET# устанавливается активным как минимум для восьми циклов SYSCLK. Это позволяет ИС 1867ВЦ10Т сбросить внешние системные компоненты. В случае, если на ИС уменьшилось напряжение питания меньше допустимого ($U_{CC} < U_{CC}$ min) на несколько микросекунд, это должно явиться причиной перехода сигнала PORESET# в низкий уровень. При возникновении условий сброса логическая схема сброса держит ИС в состоянии сброса до тех пор, пока эти условия сохраняются.

Наличие условия сброса является причиной того, что ИС заканчивает выполнение программы и изменяет содержимое различных регистров и состояние бит.

Во время сброса память внутрикристального ОЗУ не изменяется, а все управляющие биты, которые затрагиваются сбросом, приводятся в их исходное

состояние. В случае включения сброса управляющие регистры PLL устанавливаются в исходное состояние, которое равно нулю.

Программа должна определять причину сброса и устанавливать конфигурацию PLL для продолжения выполнения правильной операции.

После сброса программа может проверить причину сброса, проверив флаг включения сброса (PORST флаг/SYSSR.15), флаг нелегального адреса (ILLADR флаг/SYSSR. 12), программный сброс флага (SWRST флаг/SYSSR.10) и флаг сброса от сторожевого таймера (WDRST флаг/SYSSR.9) для определения источника сброса. Сброс не устанавливает в исходное состояние эти флаги.

Сигналы RESET# и PORESET# должны удерживаться низким уровнем до тех пор, пока сигнал синхронизации не станет нормальным и напряжение питания $U_{\#VCC2}$ не установится в пределах рабочего диапазона. Кроме этого, PORESET# должен быть переведен в низкое состояние, когда напряжение питания падает ниже минимума требуемого электрического напряжения.

4.6.6 Сигналы аппаратного прерывания

Все линии сигналов аппаратно-генерируемых прерываний процессора ИС нумеруются в списке от 1 до 10 (прерывание с номером 1 имеет самый высокий приоритет). Когда больше чем один из этих аппаратных сигналов прерываний активен в ожидании подтверждения, то сигнал прерывания более высокого уровня получает первым подтверждение на обслуживание. Другие сигналы получают это подтверждение после этого по их приоритетному порядку. Из этих десяти линий прерывания, шесть – это для маскируемых линий сигналов прерывания INT1-INT6 и один сигнал для немаскируемой линии прерывания NMI. INT1-INT6 и NMI имеют приоритеты, которые показаны в таблице 6.

Таблица 6 – Приоритеты сигналов прерывания на уровне центрального процессора ИС 1867ВЦ10Т

Сигнал прерывания	Приоритет на уровне ЦП
RESET	1
TI RESERVED	2
NMI	3
INT1	4
INT2	5
INT3	6
INT4	7
INT5	8
INT6	9
Резервируется	10

Эти линии управляются системным модулем и менеджером событий, как показано в таблице 7 и на рисунке 8.

Таблица 7 – Линии сигналов прерывания, управляемые системным модулем и менеджером событий

Периферийное устройство	Линии сигнала прерывания
Системный модуль	INT 1 INT 5, NMI INT 6
Менеджер событий	INT 2 INT 3 INT 4

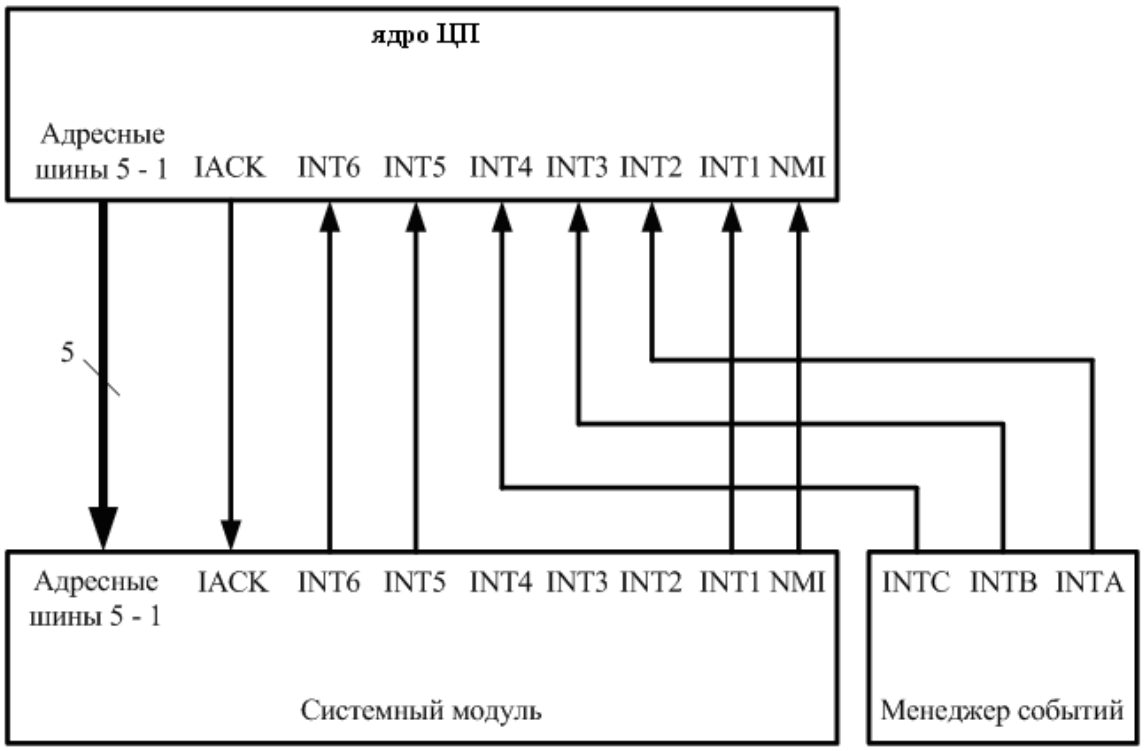


Рисунок 8 – Структура прерываний ядра ЦП

На уровне системного модуля и менеджера событий каждая из маскируемых линий сигналов прерывания INT1-INT6 соединена с многочисленными маскируемыми источниками сигналов прерывания. Источники, связанные с линией сигналов прерывания INT1, вызываются сигналами прерывания «Уровень 1»; источники, связанные с линией сигналов прерывания INT2, вызываются сигналами прерывания «Уровень 2» и т. д. В источнике, который имеет несколько входов, также имеется упорядоченный приоритет для каждого входа сигнала прерывания. Источник с самым высоким приоритетом имеет свой сигнал запроса прерывания. На этот источник сначала приходит подтверждение от ядра ЦП.

На рисунке 9 показаны источники сигналов прерывания, упорядоченные по приоритету, которые управляются системным модулем. Для каждой цепочки сиг-

налов прерывания источник сигналов прерывания, имеющий самый высокий приоритет, находится на вершине. Приоритет уменьшается с вершины цепочки до основания.

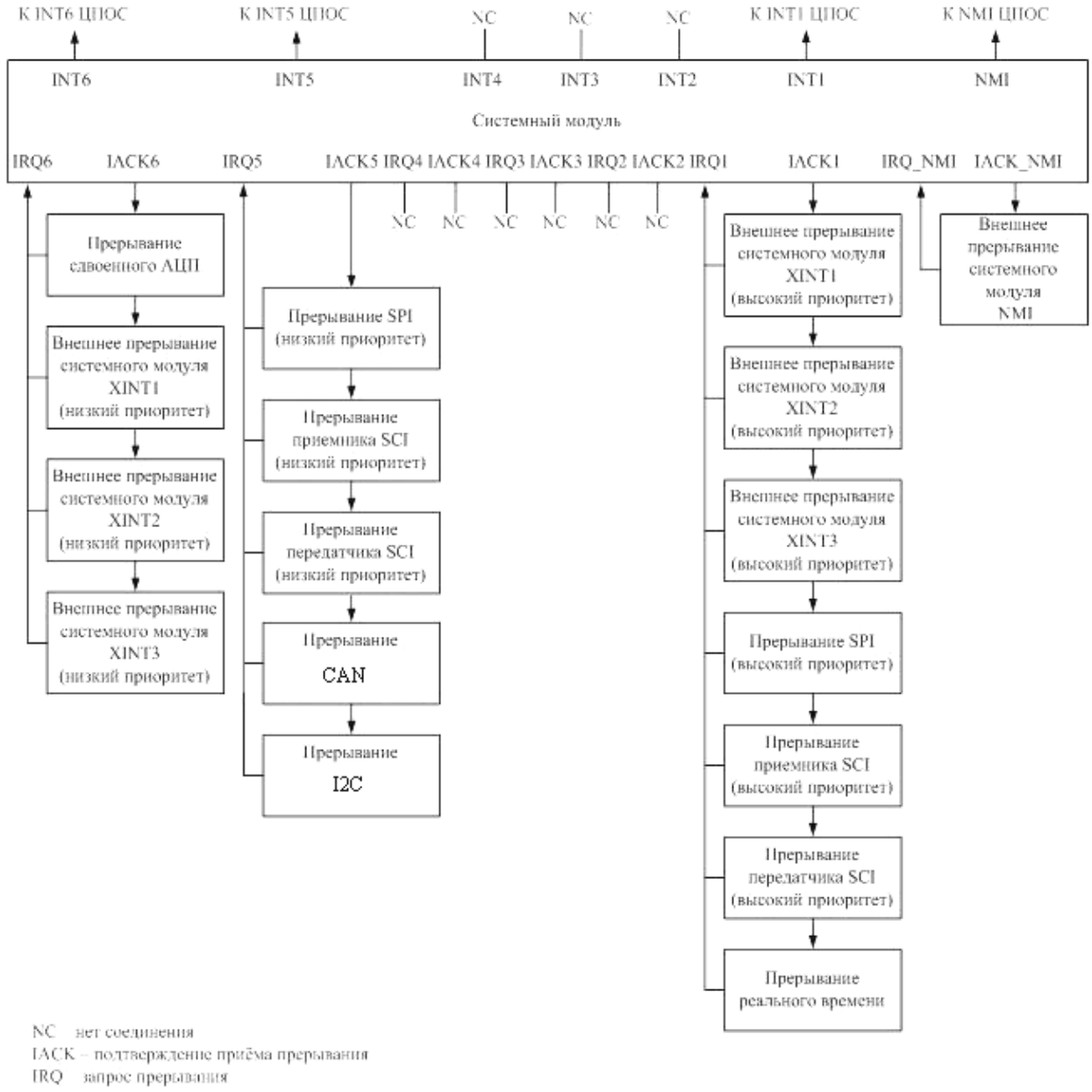


Рисунок 9 – Структура прерываний системного модуля

На рисунке 10 показаны источники сигналов прерывания и их приоритет для менеджера событий.

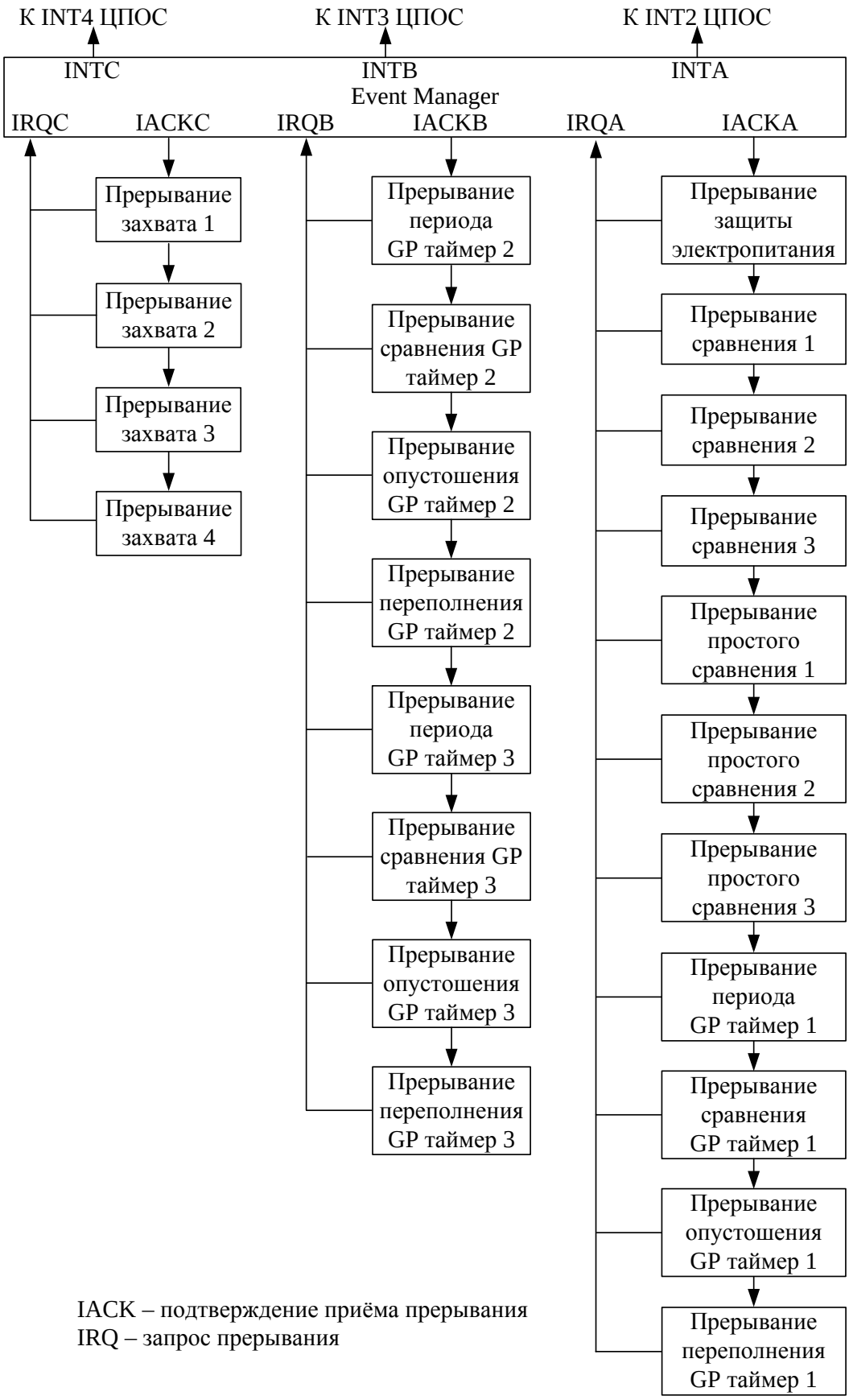


Рисунок 10 – Структура прерываний менеджера событий

Каждый из источников сигнала прерывания имеет собственный управляющий регистр с битом флага и битом маскирования. Когда запрос сигнала прерывания получен, бит флага устанавливается в единицу в соответствующем управляющем регистре.

Если бит маскирования установлен, то сигнал посылается на логическую схему управления доступа к общей шине. Эта схема может одновременно получать сигналы прерывания от одного или более управляющих регистров.

Логическая схема управления доступом к общей шине сравнивает приоритетный уровень конкурирующих запросов сигналов прерывания и пропускает к ЦП сигнал прерывания с наиболее высоким приоритетом. Соответствующий флаг устанавливается в регистре флагов прерывания IFR, указывая на то, что сигнал прерывания еще не обслужен. После этого ЦП должен решить, подтвердить запрос прерывания или нет. Маскируемые сигналы прерывания от аппаратуры подтверждаются только после выполнения некоторых условий:

- Приоритет является самым высоким. Когда больше чем один сигнал прерывания от аппаратуры запрашивается в одно и то же время, то ЦП обслуживает их в соответствии с системой ранжирования приоритета.

- INTM бит равен 0. Режим прерывания (INTM) бит, бит 9 статусного регистра STO разблокирует или блокирует все маскируемые сигналы прерывания (когда INTM = 0, то все маскируемые сигналы прерывания разблокированы, когда INTM = 1, то все маскируемые сигналы прерывания заблокированы). INTM устанавливается в 1 автоматически, когда ЦП подтверждает прием сигнала прерывания (за исключением, когда они инициированы командой TRAP) и при сбросе. Он может быть установлен и переведен в исходное состояние программными средствами.

- Маскирующий бит в регистре IMR равен 1. Каждая из линий сигналов прерывания имеет маскирующий бит в регистре маскирования сигналов прерывания IMR. Чтобы размаскировать линию сигналов прерывания, нужно установить соответствующий IMR бит в 1.

Когда ЦП подтверждает сигнал прерывания от аппаратуры, то он приостанавливает выполнение команды INTR. Эта команда устанавливает программный счетчик на соответствующий адрес, с которого ЦП вызывает вектор прерывания. Этот вектор ведет к стандартной программе обслуживания сигнала прерывания.

Обычно, стандартная программа обслуживания сигнала прерывания читает смещение периферийного векторного адреса из периферийного регистра адреса вектора (см. таблицу 7) для перехода к системе кодирования, которая предназначена для определения источника сигнала прерывания, инициировавшего запрос сигнала прерывания.

Микросхема реализует искусственный сигнал прерывания со смещением вектора (0000h), позволяющий управлять выходом из неправильной последовательности прерывания. Это происходит, если ЦП подтверждает запрос для периферийного устройства, тогда как в действительности ни одно из периферийных устройств не сформировало сигнал прерывания. В этом случае искусственный вектор прерывания читается из векторного регистра прерывания.

В таблице 8 приведены источники сигналов прерывания, общий приоритет, смещение адреса вектора, источник и функция каждого сигнала прерывания.

Таблица 8 – Источники сигналов прерывания

Прерывание	Общий приоритет	Прерывание DSP/адрес	Адрес вектора периферии	Смещение	Маскируемое?	Источник периферийного модуля	Функция прерывания
RESET#	1	RESET# /0000h	нет		Нет	Ядро, SD	Внешняя, системный сброс (RESET)
RESERVED	2	INT7/0026h	нет	Нет	Нет	Ядро	Эмулятор TRAP
NMI	3	NMI/0024h	нет	0002h	Нет	Ядро, SD	Внешнее пользовательское прерывание
XINT1 XINT2 XINT3	4 5 6	INT1/0002h (Система)	SYSIVR/ 7010Eh	0001h 0011h 001Fh	Нет	SD	Высоко приоритетные внешние пользовательские прерывания
SPIINT	7			0005h	Да	SPI	Высоко приоритетное прерывание от SPI
RXINT	8			0006h	Да	SCI	Прерывание приемника SCI (высокий приоритет)
TXINT	9			0007h	Да	SCI	Прерывание передатчика SCI (высокий приоритет)
WDINT	10			0010h	Да	WDT	Прерывание от WD
PDPINT	11	INT2/0004h (Менеджер событий Группа A)	EVIVRA/ 7432h	0020h	Да	Внешний	Прерывание от схемы управления питанием
CMP1INT	12			0021h	Да	EV.CMP1	Прерывание при полном совпадении 1
CMP2INT	13			0022h	Да	EV.CMP2	Прерывание при полном совпадении 2
CMP3INT	14			0023h	Да	EV.CMP3	Прерывание при полном совпадении 3
SCMP1INT	15			0024h	Да	EV.CMP4	Прерывание при простом совпадении 1
SCMP2INT	16			0025h	Да	EV.CMP5	Прерывание при простом совпадении 2
SCMP3INT	17			0026h	Да	EV.CMP6	Прерывание при простом совпадении 3
TPINT1	18			0027h	Да	EV.GPT1	Прерывание по периоду Таймер 1
TCINT1	19			0028h	Да	EV.GPT1	Прерывание по совпадению Таймер 1
TUFINT1	20			0029h	Да	EV.GPT1	Прерывание по заему таймер 1
TOFINT1	21			003Ah	Да	EV.GPT1	Прерывание по переполнению таймер 1

Окончание таблицы 8

Прерывание	Общий приоритет	Прерывание DSP/адрес	Адрес вектора периферии	Смещение	Маскируемое?	Источник периферийного модуля	Функция прерывания
TPINT2	22	INT3/0008h (Менеджер событий Группа B)	EVIVRB/ 7432h	002Bh	Да	EV.GPT2	Прерывание по периоду таймер 2
TCINT2	23			002Ch	Да	EV.GPT2	Прерывание по совпадению таймер 2
TUFINT2	24			002Dh	Да	EV.GPT2	Прерывание по заему таймер 2
TOFINT2	25			002Eh	Да	EV.GPT2	Прерывание по переполнению таймер 2
TPINT3	26			002Fh	Да	EV.GPT3	Прерывание по периоду таймер 3
TCINT3	27			0030h	Да	EV.GPT3	Прерывание по совпадению таймер 3
TUFINT3	28			0031h	Да	EV.GPT3	Прерывание по заему таймер 3
TOFINT3	29			0032h	Да	EV.GPT3	Прерывание по переполнению таймер 3
CAPINT1	30	INT4/0008h (Менеджер событий Группа C)	EVIVRC/ 7432h	0033h	Да	EV.CAP1	Прерывание по Захвату 1
CAPINT2	31			0034h	Да	EV.CAP2	Прерывание по Захвату 2
CAPINT3	32			0035h	Да	EV.CAP3	Прерывание по Захвату 3
CAPINT4	33			0036h	Да	EV.CAP4	Прерывание по Захвату 4
SPIINT	34	INT5 000Ah (Система)	SYSIVR/ 701Eh	0005h	Да	SPI	Низко приоритетное прерывание от SPI
RXINT	35			0006h	Да	SCI	Прерывание от приемника SCI (низкий приоритет)
TXINT	36			0007h	Да	SCI	Прерывание от передатчика SCI (низкий приоритет)
CANINT	37			0008h	Да	CAN	Прерывание от CAN
I2CINT	38			0009h	Да	I2C	Прерывание от I2C
ADCINT XINT1 XINT2 XINT3	39 40 41 42	INT6/000Ch (Система)		0004h 0001h 0011h 001Fh	Да	ADC Внешний вывод	Прерывание от АЦП
RESERVED	43	000Eh	Нет		Да	Ядро	Используется для анализа
TRAP	нет	0022h	Нет		Да		Инструкция TRAP

4.6.7 Внешние сигналы прерывания

ИС 1867ВЦ10Т имеет пять внешних сигналов прерывания. Эти сигналы прерывания включают в себя:

- XINT1. Прерывание типа A. Управляющий регистр XINT1 (7070h) обеспечивает управление и статус для этого сигнала прерывания. XINT1 может использоваться как вход маскируемого сигнала прерывания высокого уровня (уровень 1) или низкого уровня прерывания (уровень 6) или как вывод входа общего назначения. Вывод XINT1# может быть запрограммирован для формирования сигнала прерывания по положительному или отрицательному фронту входного сигнала на этом выводе.

- NMI. Прерывание типа A. Управляющий регистр NMI (7072h) обеспечивает управление и статус для этого сигнала прерывания. NMI – это немаскируемый внешний сигнал прерывания или вывод входа общего назначения. Вывод NMI# может быть запрограммирован для формирования сигнала прерывания по положительному или отрицательному фронту входного сигнала на этом выводе.

- XINT2. Прерывание типа C. Управляющий регистр XINT2 (7078h) обеспечивает управление и статус для этого сигнала прерывания. XINT2 может использоваться как маскируемый сигнал прерывания на высоком приоритете (уровень 1) или низком приоритете (уровень 6), или как вывод I/O общего назначения. Вывод XINT2 может быть запрограммирован для формирования сигнала прерывания по положительному или отрицательному фронту входного сигнала на этом выводе.

- XINT3. Прерывание типа C. Управляющий регистр XINT3 (707Ah) обеспечивает управление и статус для этого сигнала прерывания. XINT3 может использоваться как маскируемый сигнал прерывания на высоком приоритете (уровень 1) или низком приоритете (уровень 6) или как вывод I/O общего назначения. Вывод XINT3 может быть запрограммирован для формирования сигнала прерывания по положительному или отрицательному фронту входного сигнала на этом выводе.

- PDPINT. Этот сигнал прерывания предусмотрен для осуществления правильной работы преобразователя мощности или привода двигателя в случае нарушения питания. Маскируемый сигнал прерывания может установить таймеры и выходные выводы ШИМ в положение высокоимпедансного состояния и информировать ЦП о ненормальной работе привода двигателя при повышении напряжения или превышении величины тока, чрезмерном повышении температуры, других подобных событиях. PDPINT является сигналом прерывания уровня 2.

В таблице 9 обобщаются характеристики внешних прерываний ИС 1867ВЦ10Т.

Таблица 9 - Внешние сигналы прерывания, типы и функции

Внешний сигнал прерывания	Название управляющего регистра	Адрес управляющего регистра	Тип сигнала прерывания	Можно применить NMI?	Цифровой вывод I/O	Маскируемый
XINT1	XINT1CR	7070h	A	Нет	Только ввод	Да (уровень от 1 до 6)
NMI	NMICR	7072h	A	Да	Только ввод	Нет
XINT2	XINT2CR	7078h	C	Нет	I/O	Да (уровень от 1 до 6)
XINT3	XINT3CR	707Ah	C	Нет	I/O	Да (уровень от 1 до 6)
PDPINT	EVIMRA	742Ch	N/A	Да/нет	N/A	Да (уровень от 1 до 2)

4.7 Генератор синхросигналов

ИС 1867ВЦ10Т имеет внутрикристальный модуль синхронизации с фазовой подстройкой частоты (PLL). Этот модуль обеспечивает все необходимые синхронизирующие сигналы для ИС, так же как и управление режимом пониженного энергопотребления. Единственный внешний компонент, необходимый для этого модуля, – это внешний кварцевый резонатор или генератор.

Базовый модуль синхронизации PLL обеспечивает два основных режима функционирования – это режим генератора и режим «включенной» синхронизации (clockin).

4.8 Режим внутреннего генератора

Этот режим позволяет использовать четырех, шести или восьми МГц внешние кристаллы кварцевого резонатора для обеспечения временной диаграммы всей микросхемы.

Внутренняя схема генератора устанавливается программным обеспечением в соответствующее состояние для выбора требуемой частоты ЦП, которая может быть равной входной частоте синхронизации, частоте входной синхронизации, поделенной на два (по умолчанию) или частоте входной синхронизации, определенной PLL.

4.9 Режим синхронизации от внешнего генератора

Этот режим позволяет шунтировать схему внутрикристального генератора. В этом режиме модуль синхронизирующих импульсов запускается от входа внешнего источника синхронизации на выводе BQ1/CLKIN. Этот модуль может быть сконфигурирован программным обеспечением (для функционирования на частоте входной синхронизации, частоте синхронизации, поделенной на два или частоте входной синхронизации, определенной через PLL).

4.10 Система синхронизации в ИС 1867ВЦ10Т

ИС функционирует на двух частотах синхронизации: синхронизации ЦП (частота CPUCLK) и системной синхронизации (частота SYSCLK). ЦП, устройства памяти, внешний интерфейс памяти и менеджер событий «работают» на частоте CPUCLK. Все другие периферийные устройства «работают» на частоте SYSCLK. Частота CPUCLK равна 2× или 4× частоты SYSCLK; например, для 2×: CPUCLK = 20 МГц и SYSCLK= 10 МГц. Есть также синхронизация для сторожевого таймера (частота WDCLK). Эта синхронизация имеет номинальную частоту 16384 Гц.

Модуль синхронизации включает три внешних вывода:

- BQ1/CLKIN – источник синхронизации кристалла;
- BQ2 – вывод к кристаллу кварцевого резонатора;
- OSCBYP# – шунтирование внутреннего генератора.

Если OSCBYP# $\geq U_{IH}$ (высокий уровень), тогда внутренний генератор разблокирован и, если OSCBYP# $\leq U_{IL}$ (низкий уровень), тогда внутренний генератор шунтируется и устройство находится в режиме синхронизации от внешнего генератора.

В режиме синхронизации от внешнего генератора, последний должен иметь TTL уровни для подключения к выводу BQ1/CLKIN. Вывод BQ2 может быть оставлен без подключения. Блок-схема модуля синхронизации представлена на рисунке 11.

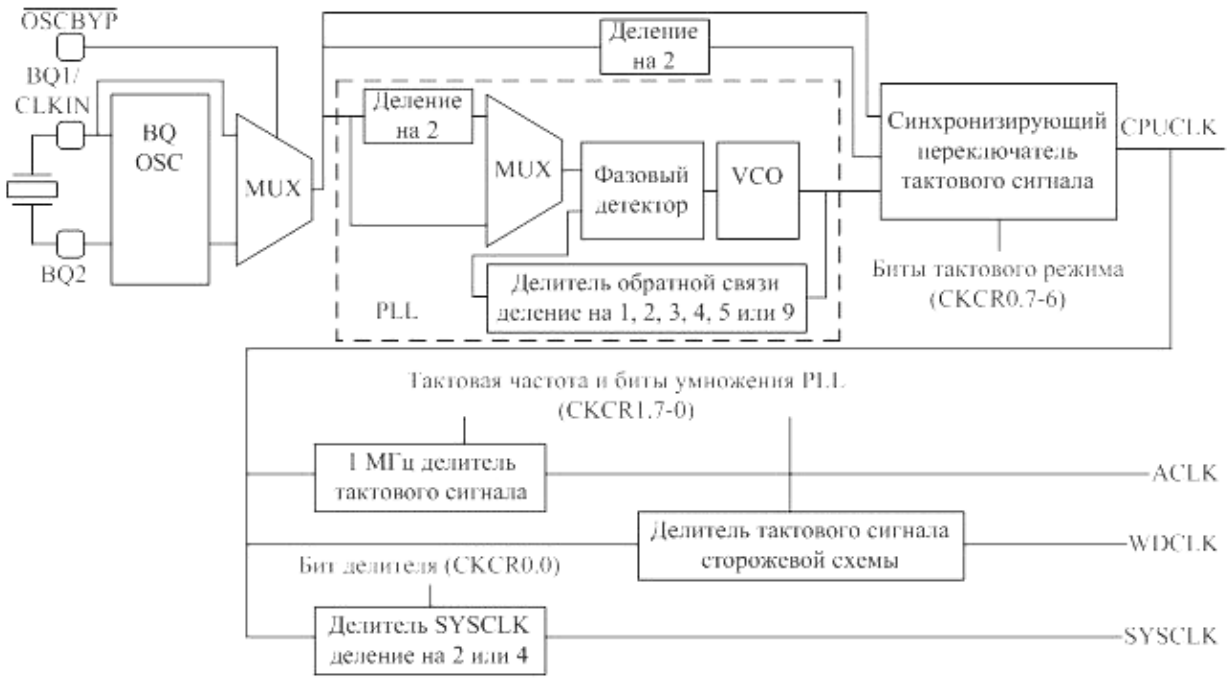


Рисунок 11 – Блок-схема модуля синхронизации PLL

4.11 Режимы низкого энергопотребления

ИС 1867ВЦ10Т имеет четыре низкопотребляемых режима (IDLE1, IDLE2, PLL со снижением мощности и осциллятор со снижением мощности потребления), которые уменьшают энергопотребление схемы во время работы путем уменьшения или прекращения работы различных модулей (путем блокировки их синхронизирующих импульсов). Два PLL бита модуля синхронизации управляющего регистра CKCR0 выбирают, какой из низкопотребляемых режимов ИС выбирается во время выполнения команды IDLE. Сброс или немаскируемый сигнал прерывания от любого источника вызывают выход ИС из режима пониженного энергопотребления IDLE1.

Сигнал прерывания в режиме реального времени от модуля сторожевого таймера вызывает выход ИС из всех низкоэнергопотребляемых режимов, кроме режима генератора со снижением потребляемой мощности. Он является «будящим» сигналом прерывания. Разблокирование, сброс или любое из четырех внешних сигналов прерывания (NMI, XINT1, XINT2 или XINT3) вызывают выход ИС из любого низкоэнергопотребляемого режима (IDLE1, IDLE2, PLL со снижением мощности потребления, генератор со снижением мощности потребления и осциллятор со снижением мощности потребления). Все внешние сигналы прерывания являются «будящими» сигналами прерывания. Маскируемые внешние сигналы прерывания (XINT1, XINT2 и XINT3) должны быть разблокированы индивидуально и глобально для правильного определения низкоэнергопотребляемого режима микросхемы. Это важно для обеспечения того, чтобы желаемый путь выхода из низкоэнергопотребляемого режима был разблокирован до его введения.

На рисунке 12 показана «будящая» последовательность после включения режима низкого потребления (power down).

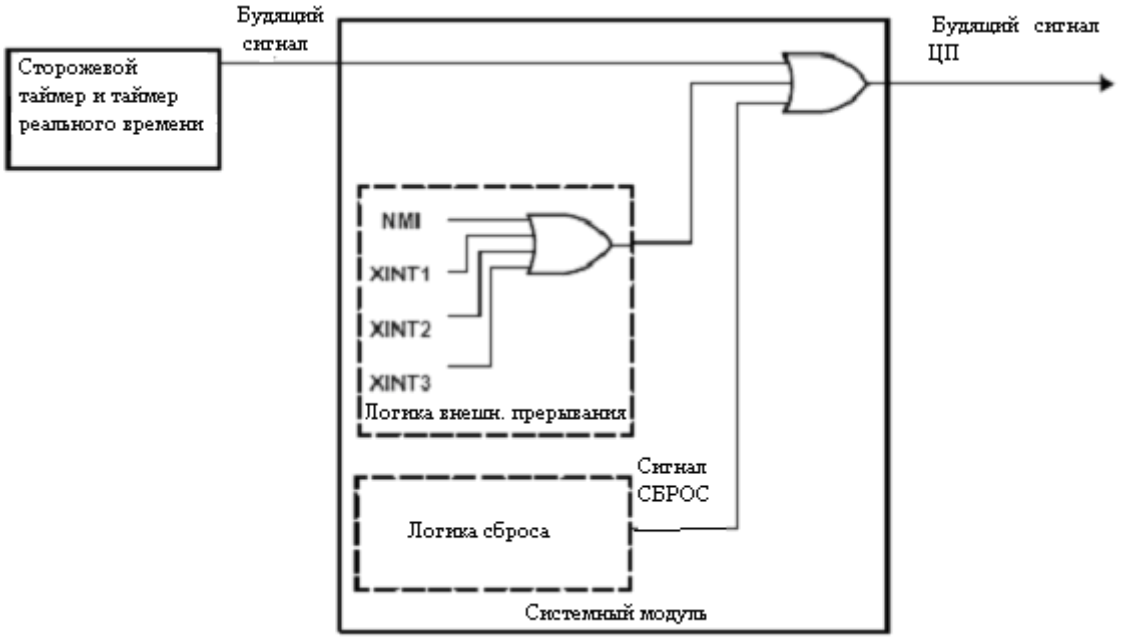


Рисунок 12 – Блок-схема выхода из режима низкого потребления

В таблице 10 указаны все низкопотребляемые режимы и их характеристики.

Таблица 10 – Низкопотребляемые режимы

Режим энергопотребления	Биты PDM(x)	Статус CPCLK	Статус SYSCLK	Статус WDCLK	Статус OSC	Условие выхода	Типовой ток потребления
Работа	XX	Вкл.	Вкл.	Вкл.	Вкл.	–	240 мА
IDLE1	00	Выкл.	Вкл.	Вкл.	Вкл.	Любое прерывание или сброс	120 мА
IDLE2	01	Выкл.	Выкл.	Вкл.	Вкл.	«Будящее» прерывание или сброс	20 мА
Выключен PLL	10	Выкл.	Выкл.	Вкл.	Вкл.	«Будящее» прерывание или сброс	10 мА
Выключен OSC	11	Выкл.	Выкл.	Выкл.	Выкл.	«Будящее» прерывание или сброс	Не более 1,6 мА

5 Структурная блок-схема ЦП 1867ВЦ10Т

Структурная блок-схема приведена на рисунке 3.

Описание основных элементов ИС 1867ВЦ10Т приведено в таблице 11.

Таблица 11 – Основные элементы ИС 1867ВЦ10Т

Название	Описание
Аккумулятор	32-битный регистр, хранящий результат и обеспечивающий вход для последовательных операций CALU. Также включает в себя возможность обычного и циклического сдвига
Арифметическое устройство вспомогательных регистров	Необозначенное 16-битное арифметическое устройство для вычисления косвенных адресов, использующее вспомогательные регистры как источник и приемник
Вспомогательные регистры 0-7	Эти 16-битные регистры используются как указатели адресного пространства в области данных. Они управляются через ARAU и выбираются через вспомогательный регистр указателя (ARP). AR0 может также использоваться как индексное значение для изменения AR и как значение сравнения с AR
Сигнал канала запроса	READY устанавливается к ИС, когда глобальная память данных доступна для обмена данными.
Перенос	Выход переноса регистра из CALU. Перенос передается в CALU для расширенной арифметической операции. “C” бит постоянно хранится в статусном регистре ST1 и может проверяться в условных командах. “C” бит также используется в аккумуляторе для сдвигов и циклических сдвигов

Продолжение таблицы 11

Название	Описание
Центральное арифметическое логическое устройство	32-разрядное основное арифметическое и логическое устройство для ядра ИС 1867ВЦ10Т. CALU выполняет 32-битные операции за один машинный цикл. CALU оперирует данными, выходящими от ISCALE или PSCALE, и с данными от ACC. CALU обеспечивает результирующий статус для PCTRL
ОЗУ двойного доступа	Если управляющий бит CNF установлен в 0, то внутрикристальная память - блок B0 картирован в пространство данных; в противном случае B0 картирован в программное пространство. Блоки B1 и B2 картированы только в пространство памяти данных по адресам 0300-03FF и 0060-007F, соответственно. Блоки B0 и B1 содержат по 256 слов, блок B2 содержит 32 слова
ОЗУ одинарного доступа	SRAM - это память с одинарным доступом за машинный цикл с объемом 46064 слов ×16 бит. SRAM состоит из двух блоков: SRAM_B1 (1088 x 16 бит) и SRAM_B2 (35768 x 16 бит).
Указатель страницы памяти данных	9-битный DP регистр, связанный с семью младшими битами LSB для формирования 16-битного прямого адреса памяти. DP может изменяться командами LST и LDP
Регистр маски прерываний	IMR индивидуально маскирует или разблокирует семь сигналов прерывания
Регистр флага прерываний	7-битный IFR указывает, что ИС зафиксировал (защелкнул) сигнал прерывания от одного из семи маскируемых сигналов прерывания
Перехват прерывания	Общее количество 32 аппаратных/программных прерываний
Ввод масштабирования данных сдвигового регистра	16/32-битное устройство циклического сдвига влево. ISCALE сдвигает входящие 16-битные данные от 0 до 16 позиций влево относительно 32-битного выхода в течение цикла выборки. Не требуется дополнительного цикла для реализации операции масштабирования
Устройство умножения	16×16-битное устройство умножения для получения 32-битного произведения. MPY производит умножение в одном цикле. MPY оперирует со знаковым или без знаковым числом с дополнением до двух
Микростек	MSTACK обеспечивает временное хранение адреса следующей команды для того, чтобы быть выбранным, после того как логическая схема формирования адреса использована для генерации последовательных адресов в пространстве данных
Мультиплексор	Мультиплексор шины
Регистр следующего программного адреса	NPAR хранит адрес программы для управления PAB в следующем цикле

Окончание таблицы 11

Название	Описание
Выход сдвигателя масштабирования данных	16/32-битное устройство циклического сдвига влево. OSCALE сдвигает 32-битный выход аккумулятора от 0 до 7 бит влево для управления квантизацией и подает либо 16 старших бит, либо 16 младших бит 32-битных данных на шину данных для записи (Data Write Data Bus - DWEB)
Регистр программного адреса	PAR хранит адрес, который является текущим на PAB во время стольких циклов, сколько это необходимо для завершения всех операций с памятью, планируемых для текущего цикла шины
Программный счетчик	PC увеличивает значение от NPAR для обеспечения последовательности адресов для выборки инструкции и последовательных операций по передаче данных
Программный контроллер	Декодирует инструкцию и управляет конвейером, сохраняет статус операции и декодирует условные операции
Регистр произведения	32-битный регистр, который хранит результат умножения 16×16
Сдвиговый регистр для масштабирования	0-, 1-, или 4-битный сдвигатель влево, или 6-битный сдвигатель вправо устройства умножения. Варианты левого сдвига используются для управления дополнительными знаковыми битами, являющимися результатом умножения чисел с дополнением до двух. Вариант правого сдвига используется для масштабирования вниз числа, для устранения переполнения произведения, накопленного в CALU. PSCALE находится на линии 32-битного сдвигателя произведения или CALU или шины данных для записи (Data Write Data Bus - DWEB) и не требует никакого дополнительного цикла для сдвига
Стековая память	STACK - это блок памяти, используемый для хранения адресов возврата для подпрограмм и обслуживания сигналов прерывания или для накапливаемых данных. Стековая память имеет 16-битную ширину и восьмиуровневую глубину
Временный регистр	16-битный регистр, который хранит один из операндов для операций умножения. TREG хранит динамический счетчик сдвига для команд LACT, ADDT и SUBT, а также динамичную позицию бита для команды BITT

6 Центральный процессор

Центральный процессор ИС 1867ВЦ10Т использует улучшенную Гарвардскую архитектуру процессора, которая максимизирует эффективность обработки данных путем использования двух отдельных структур шин памяти (программ и данных) для выполнения инструкций на максимальной скорости. Эта множественная структура шин позволяет данным и командам считываться одновременно. Эта архитектура позволяет программным коэффициентам, которые хранятся в программной памяти, быть считанными из памяти, что устраняет необходимость иметь отдельное ПЗУ коэффициентов. Все это, а также четырехуровневый конвейер, позволяет микросхеме выполнять большинство команд за один цикл.

6.1 Статусные и управляющие регистры

Два статусных регистра, ST0 и ST1, содержат состояние различных условий и режимов (см. таблицы 12 и 13). Эти регистры могут сохраняться в памяти данных и также могут загружаться из нее, позволяя таким образом сохранить состояние ЦП и восстановить это состояние для выполнения подпрограмм. Команда загрузки статусного регистра (LST) используется для записи в ST0 и ST1 кроме INTM бита, который не затрагивается командой LST. Команда сохранения статусного регистра (SST) используется для чтения из ST0 и ST1.

Индивидуальные биты этих регистров могут быть установлены или приведены в исходное состояние при использовании команд SETC и CLRC. В таблицах 12 и 13 показана организация статусных регистров ST0 и ST1, указывая все статусные биты, содержащиеся в каждом из них. Несколько бит в статусных регистрах зарезервированы и читаются как логическая единица.

Таблица 12 – Назначение бит ST0

15 13	12	11	10	9	8 0
ARP	OV	OVM	1	INTM	DP

Таблица 13 – Назначение бит ST1

15 13	12	11	10	9	8	7	6	5	4	3	2	1 0
ARB	CNF	TC	SXM	C	1	1	1	1	XF	1	1	PM

В таблице 14 описываются поля и биты статусных регистров.

Таблица 14 – Описание полей и битов статусных регистров

Поле	Функция
ARB	Буфер указателя вспомогательного регистра. Когда ARP загружается в ST0, то старое значение ARP копируется в ARB кроме инструкции LST. Когда ARB загружается командой LST #1, то тоже самое значение копируется в ARP
ARP	Указатель вспомогательного регистра AR. ARP выбирает AR для использования косвенной адресации. Когда ARP загружается, то старое значение ARP копируется в ARB. ARP может модифицироваться инструкциями, использующими косвенную адресацию, и инструкциями LARP, MAR и LST. ARP загружается тем же значением, что и ARB, когда выполняется инструкция LST #1

Продолжение таблицы 14

Поле	Функция
C	Бит переноса «C» устанавливается в 1, если результат сложения генерирует перенос или сбрасывается в 0, если результат вычитания генерирует заем. Иначе бит «C» сбрасывается в 0 после сложения или устанавливается после вычитания кроме случая, если инструкции ADD/SUB имеют 16-битный сдвиг. В этом случае ADD может только установить, а SUB только сбросить бит «C». Одиночный сдвиг или циклический сдвиг также влияют на бит «C», так же, как инструкции SETC, CLRC и LST #1. Инструкции перехода используют состояние этого бита. При сбросе бит «C» устанавливается в 1
CNF	Бит конфигурирования внутрикристальной памяти. Если CNF=0, то DARAM (блок B0) картируется в пространство данных, в противном случае B0 картируется в пространство программы. CNF модифицируется инструкциями SETC CNF, CLRC CNF и LST #1. Сброс устанавливает CNF в 0
DP	Указатель страницы данных. Это 9-битное поле соединяется с семью младшими битами слова инструкции для формирования 16-битного адреса при прямой адресации. DP модифицируется инструкциями LDP и LST
INTM	Бит режима прерывания. Когда INTM=0, то все маскируемые прерывания разрешены, в противном случае, все маскируемые прерывания запрещены. INTM устанавливается и сбрасывается инструкциями SETC INTM и CLRC INTM. Сигналы RESET# и IACK# также устанавливают INTM в 1. INTM не влияет на немаскируемые прерывания RESET и NMI. На INTM не влияет инструкция LST. Сигнал сброс устанавливает INTM в 1
OVM	Бит режима переполнения. Когда OVM установлен в 0, то результаты переполнения не меняют аккумулятор. Когда установлен в 1, то аккумулятор устанавливается или в наибольшее положительное, или в наименьшее отрицательное число в зависимости от направления переполнения. Инструкции SETC и CLRC устанавливают и сбрасывают этот бит соответственно. Инструкция LST также модифицирует этот бит
PM	Режим сдвига произведения. Если PM=00, то умножитель загружает произведение в АЛУ с 0-ым сдвигом. Если PM=01, то выход PREG сдвигается влево на 1 бит и после загружается в АЛУ с заполнением 0 младшего бита. Если PM=10, то выход PREG сдвигается влево на 4 бита и после загружается в АЛУ с заполнением 0 младших битов. Если PM=11, то выход PREG сдвигается вправо на 6 бит с расширением знака. PREG остается неизменным. Сдвиг происходит, когда содержимое PREG передается в АЛУ. PM загружается инструкциями SPM и LST #1. PM сбрасывается сбросом
SXM	Бит расширения знака. Если SXM=1, то генерируется расширение знака во время передачи данных в аккумулятор масштабирующим сдвигателем

Окончание таблицы 14

Поле	Функция
TC	Тестовый/управляющий бит. TC изменяется инструкциями BIT, BITT, CMPR, LST и NORM. TC устанавливается в 1, если тестируемый бит инструкциями BIT и BITT находится в 1, если при сравнении AR(ARP) и AR0 условие сравнения существует или если функция «исключающего – ИЛИ» двух старших битов аккумулятора истинна при их тестировании инструкцией NORM. Условные инструкции, а также инструкции вызова перехода и возврата могут выполняться, основываясь на состоянии бита TC
XF	Статусный бит вывода XF. Этот бит показывает состояние вывода XF, используемого как вывод общего назначения. Инструкции SETC XF и CLRC XF соответственно устанавливают и сбрасывают этот бит. Сброс устанавливает этот бит в 1

6.2 Центральное процессорное устройство

Центральное процессорное устройство **(**ЦПУ) ИС 1867ВЦ10Т содержит 16-битный масштабирующий сдвигатель, (16×16)-разрядное параллельное устройство умножения, 32-разрядное центральное арифметическое логическое устройство (CALU), 32-битный аккумулятор и дополнительные сдвигатели на выходах аккумулятора и умножителя.

Этот подраздел описывает компоненты ЦПУ и их функции.

6.2.1 Вход масштабируемого сдвигателя

ИС 1867ВЦ10Т имеет масштабирующий сдвигатель с 16-битным входом, связанным с шиной данных и 32-битным выходом, который связан с CALU. Сдвигатель управляет направлением прохождения данных, который идет от пространства программы или данных к CALU и не требует дополнительного цикла. Он используется для того, чтобы выравнить 16-битные данные, выходящие из памяти к 32-битному CALU. Это необходимо для масштабируемой арифметики так же, как выравненные маски для логических операций.

Масштабирующий сдвигатель производит левый сдвиг от 0 до 16 на входе данных. LSB выводы заполняются нулями, а MSB может заполниться или нулями или знаково расшириться в зависимости от значения SXM бита (режим знакового расширения – sign extention mode) статусного регистра ST1. Количество сдвигов указывается в слове команды или значением регистра TREG.

Количество сдвигов в команде позволяет указать специальное масштабирование или выравнивание для операций специального масштабирования или выравнивания. Сдвигатель TREG позволяет адаптировать показатель масштабирования к характеристике системы.

6.2.2 Умножитель

ЦПУ микросхемы 1867ВЦ10Т использует 16×16 аппаратный умножитель, который может осуществлять обработку данных со знаком или без знака для получения 32-битного произведения в одном машинном цикле. Все команды умножения, кроме команды MPYU (умножение без знака), выполняют операцию умножения со знаком (signed). Таким образом, два числа, будучи умноженными, рассматриваются как числа с дополнением до двух, и результатом умножения является 32-битное число с дополнением до двух. Два регистра связаны с устройством умножения: 16-битный временный регистр TREG, который хранит один из операндов для устройства умножения; 32-битный регистр произведения PREG, который хранит это произведение.

Четыре режима сдвигателя произведения (определяются полем PM) возможны на PREG выходе (PSCALE). Эти режимы сдвигателя полезны для выполнения операций при умножении/накоплении, при выполнении дробных арифметических операций или выполнении выравнивания дробных произведений.

Поле PM статусного регистра ST1 устанавливает режим сдвига, как показано в таблице 15.

Таблица 15 – Описание значений поля PM статусного регистра ST1

PM	Сдвиг	Описание
00	Нет сдвига	Передача произведения в CALU или шину данных без сдвига
01	Левый сдвиг на 1	Удаляет дополнительный знаковый двоичный разряд, вычисленный в устройстве умножения с дополнением до двух (Q31)
10	Левый сдвиг на 4	Удаляет дополнительные четыре знаковых разряда, полученные в 16×13 устройстве умножения с дополнением до двух для получения произведения Q31, когда используется умножение с 13-битной константой
11	Правый сдвиг на 6	Масштабирует произведение для получения свыше 128 суммирований произведений, для исключения возможности переполнения аккумулятора

Произведение может быть сдвинуто на один бит для компенсации дополнительного знакового (двоичного) разряда, полученного умножением двух 16-битных чисел с дополнением до двух (команда MPY). Четырехбитный сдвиг используется в сочетании с командой MPY, с коротким непосредственным значением (13 битов или меньше) для удаления четырех дополнительных знаковых (двоичных) разрядов, полученных умножением 16-битного числа на 13-битное число. Вывод PREG может быть сдвинут вправо на 6 бит для выполнения 128 последовательных умножений с накоплением без возможного переполнения.

Команда LT (загрузка TREG) обычно загружает TREG для обеспечения одного операнда (с шины данных), а команда MPY (умножение) обеспечивает второй операнд (также с шины данных). Умножение может быть также осуществлено с 13-битным непосредственным операндом во время использования команды MPY. Операция умножения в этом случае использует каждые два машинных цик-

ла для своего выполнения. Когда код выполняет многократные операции умножения и суммирует произведения, ЦП поддерживает конвейерный режим TREG запуском операций с CALU операциями, использующих предыдущее произведение. Операции конвейерного режима проходят параллельно с загрузкой TREG, включая загрузку ACC из PREG (LTP), добавление PREG к ACC (LTA), добавление PREG к ACC и сдвиг данных входа PREG (DMOV) к следующему адресу в памяти данных (LTD), вычитание PREG из ACC (LTS).

Две команды умножения с накоплением (MAC и MACD) используют в полном объеме вычислительную мощность устройства умножения, позволяя обоим операндам обрабатываться одновременно. Операнды этих операций могут быть переданы устройству умножения в каждом цикле через программную шину или шину данных. Это уменьшает число циклов умножения/накопления, когда они используются с командой повтора (RPT). В этих командах адрес коэффициента получается из программного адресного пространства логической части (PAGEN), в то время как адреса данных образуются логической частью формирования адреса данных (DAGEN). Это позволяет повторяемой команде последовательно получать значения из таблицы коэффициентов и получать данные в любом из косвенных адресных режимов.

Команда MACD при повторе поддерживает конструкции для реализации фильтров таким образом, что вычисляемая сумма произведений выборки данных сдвигается в память и освобождает участок памяти для следующей выборки, выталкивая наиболее старую выборку.

Команда MPYU выполняет умножение без знака. Она значительно облегчает получение расширенной точности арифметических операций. Число без знака TREG умножается на число без знака, размещенного в адресном пространстве памяти данных, с результатом, расположенным в PREG. Это дает возможность умножать операнды с большей разрядностью чем 16 бит, разбивая их на два 16-битных слова и обрабатывая их отдельно для получения произведения разрядностью больше чем 32 бита. Команды SQRA (квадрат числа/суммирование) и SQRS (квадрат числа/вычитание) подают одно и то же значение операнда к обоим входам устройства умножения для получения квадрата значения числа из памяти данных.

После умножения двух 16-битных чисел 32-битное произведение загружается в 32-битный регистр произведения (PREG). Произведение из PREG может быть передано в CALU или память данных инструкциями SPH (сохранение старших разрядов произведения) и SPL (сохранение младших разрядов произведения).

Примечание – Данные переходят из PREG к CALU или шине данных через сдвигатель PSCALE, и поэтому они могут быть изменены в зависимости от режима сдвига произведения, определенный в PM.

Это важно, когда происходит сохранение PREG в контексте стандартной программы обработки сигналов прерывания (interrupt-service-routine context), так как действия сдвигателя PSCALE не могут быть смоделированы при операции восстановления контекста. PREG может быть очищен выполнением команды MPY# 0. Регистр произведения может быть восстановлен через загрузку сохраненной младшей половины в TREG, выполнением команды MPY# 0. Тогда как старшая половина загружается, используя команду LPH.

6.2.3 Центральное арифметическое логическое устройство

Центральное арифметическое логическое устройство ИС 1867ВЦ10Т (CALU) реализует широкий набор арифметических и логических операций, большинство из которых выполняются в одном машинном цикле. Это АЛУ относится к центральному устройству для отличия его от второго АЛУ, которое используется для создания косвенного адреса, названного арифметическим устройством вспомогательного регистра (ARAU). Как только операция в CALU завершена, то результат из него передается в аккумулятор (ACC), где могут произойти дополнительные операции, такие как сдвиг. Входные данные в CALU могут быть масштабированы через ISCALE во время передачи от одной шины данных к другой шине данных (DRDB или PRDB) или масштабированы через PSCALE во время передачи от устройства умножения.

CALU представляет собой арифметическое логическое устройство общего назначения, которое оперирует 16-битными словами, взятыми из памяти данных или полученными непосредственно из команд. В добавление к обычным арифметическим командам CALU может выполнять логические операции, облегчая возможность управления отдельным битом, что требуется для высокоскоростного контроллера. Один вход в CALU всегда идет из аккумулятора, другой вход может идти из регистра произведения (PREG) устройства умножения или выхода масштабирующего сдвигателя (читается из памяти данных или с ACC). После того как CALU выполнил арифметическую или логическую операцию результат сохраняется в аккумуляторе.

ИС 1867ВЦ10Т поддерживает операции с плавающей точкой/запятой для функций, требующих широкого динамичного диапазона. Команда NORM (нормализация) используется для приведения к норме чисел с фиксированной точкой (fixed-point numbers), которые содержатся в аккумуляторе, путем выполнения левых сдвигов. Четыре бита TREG определяют переменный сдвиг через масштабирующий сдвигатель для команд LACT/ADDT/ SUBT (загрузка/добавить к/вычесть из аккумулятора со сдвигом, определенным TREG). Эти команды полезны в арифметике с плавающей точкой, где число должно быть денормализовано - это требуется для преобразования числа с плавающей точкой в число с фиксированной точкой. Эти команды также полезны при выполнении автоматического управления усилением (AGC) сигнала, проходящего в фильтр. Команда BITT (тест бита) обеспечивает тестирование одиночного бита в слове из памяти данных, который определяется четырьмя младшими битами (LSB) регистра TREG.

Режим насыщения переполнением CALU может быть разблокирован/заблокирован через установку бита OVM в ST0. Когда CALU находится в режиме насыщения переполнением и происходит переполнение (флаг переполнения установлен), то аккумулятор загружается наибольшим положительным или наименьшим отрицательным значением, представленным в аккумуляторе, в зависимости от направления переполнения. Значение аккумулятора при насыщении – 07FFFFFFh (положительное) или 080000000h (отрицательное). Если бит статусного регистра OVM (режим переполнения) сброшен и происходит переполнение (overflow), то этот результат загружается в аккумулятор без изменения (логические операции не могут вызвать переполнение).

CALU может осуществлять множество команд перехода, которые зависят от статуса CALU и аккумулятора. Эти команды могут выполняться с условием, которое основано на любой комбинации статусных битов. Для управления по переполнению эти условия включают в себя OV (ветвление по переполнению) и EQ (ветвление при аккумуляторе равным нулю). Кроме этого команда BACC (ветвление по адресу аккумулятора) обеспечивает возможность переходить на адрес, определенный аккумулятором (вычисленный goto). Команды теста бита (BIT и BITT), которые не затрагивают аккумулятор, позволяют тестировать специфицированный бит в слове памяти данных.

CALU также имеет бит переноса, который устанавливается в зависимости от разных операций в пределах устройства. Бит переноса изменяется большинством арифметических команд, так же как и командами сдвига на один бит и командами циклического сдвига. Он не изменяется загрузкой аккумулятора, логическими операциями, другими неарифметическими или управляющими командами.

Команды ADDC (добавление к аккумулятору с переносом) и SUBB (вычитание из аккумулятора с заемом) используют предыдущее значение переноса в их операциях по сложению/вычитанию.

Единственным исключением является операция формирования разряда переноса при выполнении команды ADD с числом сдвигов равным 16 битам (прибавить к старшему слову аккумулятора) и SUB с числом сдвигов равным 16 битам (вычитание из старшего слова аккумулятора). Команда ADD устанавливает бит переноса, если перенос произошел, а команда SUB может сбросить бит переноса, если произошел заем; в другом случае никакая другая команда не изменяет его.

Два условных операнда C и NC предназначены для перехода, вызова, возвращения и условного выполнения команд, которые основаны на статусе бита переноса. Команды SETC, CLRC и LST #1 могут также использоваться для изменения бита переноса. Бит переноса устанавливается в 1 при аппаратном сбросе.

6.2.4 Аккумулятор

32-битный аккумулятор является выходом CALU. Он может быть разделен на два 16-битных сегмента для сохранения их в памяти данных. Сдвиговый регистр на выходе аккумулятора обеспечивает левый сдвиг от 0 до 7 позиций. Этот сдвиг выполняется во время передачи данных к шине данных для сохранения в памяти. Содержание аккумулятора при этом остается неизменным. Когда масштабирующий сдвигатель используется для «старшего слова» аккумулятора (биты 16-31), то MSB теряются, а LSB заполняются битами, смещенными туда из младшего слова (биты 0-15). Когда масштабирующий сдвигатель используется для младшего слова, то LSB заполняются нулями.

Команды SFL и SFR для однобитного сдвига влево/вправо, а также команды ROL и ROR (вращение влево/вправо) включают схему сдвига содержимого аккумулятора через разряд переноса. SXM бит уточняет команду SFR (правый сдвиг регистра аккумулятора). Когда SXM =1, SFR выполняет арифметический правый сдвиг, поддерживая знак данных аккумулятора. Когда SXM =0, то SFR выполняет логическую сдвиговую схему, сдвигая вне LSB и вдвигая нуль в MSB. SFL (левый сдвиговый аккумулятор) не затрагивается битом SXM и одинаково выполняется в обоих случаях, сдвигая вне MSB и вдвигая ноль в LSB. Команды повтора (RPT)

могут использоваться с командами сдвига и циклического сдвига для сдвига на несколько бит.

6.2.5 Вспомогательные регистры и арифметическое устройство вспомогательных регистров ARAU

ИС 1867ВЦ10Т имеет блок регистров, содержащий восемь вспомогательных регистров (AR0-AR7). Вспомогательные регистры используются для косвенной адресации памяти данных или временного хранения данных. Косвенная адресация через вспомогательный регистр позволяет разместить адрес операнда в памяти данных в одном из вспомогательных регистров. Эти регистры доступны через 3-битный указатель вспомогательного регистра (ARP), который загружается значением от 0 до 7 (обозначенные от AR0 до AR7 соответственно). Вспомогательные регистры и ARP могут быть загружены из памяти данных, ACC, регистра произведения или непосредственного операнда, который определен в команде. Содержимое этих регистров может быть сохранено в памяти данных или использовано как входы в CALU.

Блок вспомогательных регистров (AR0-AR7) связан с ARAU. ARAU может автоматически индексировать текущий вспомогательный регистр во время адресации ячейки памяти данных. Индексация регистра может выполняться на +/-1 или содержимым AR0. Для вычисления адреса в памяти данных не требуется участие CALU; таким образом, CALU освобождено для выполнения других операций параллельно с вычислением адреса.

6.2.6 Внутрикристальная память

Внутрикристальная память ИС 1867ВЦ10Т состоит из
- ОЗУ двойного доступа за машинный цикл (DARAM);
- ОЗУ однократного доступа за машинный цикл (SARAM);
- перепрограмируемой постоянной памяти (FLASH).

ОЗУ двойного доступа DARAM

Память DARAM позволяет записывать и считывать данные в/из неё в одном и том же машинном цикле, что обеспечивает параллельное обращение на чтение и запись на полной скорости без состояний ожидания ЦПУ.

Объём DARAM составляет 1824 слова ×16 бит. DARAM состоит из трех блоков: блок 0 (B0), блок 1 (B1) и блок 2 (B2).

Блок 0 содержит 512 слов и может быть сконфигурирован либо в пространство данных, либо в пространство программ. Команды SETC CNF (конфигурирует B0 как память данных) и CLRC CNF (конфигурирует B0 как память программ) позволяют изменять конфигурацию карты памяти программным обеспечением. Во время использования блока B0 как память программ, его содержимое может быть загружено из внешней программной памяти, а затем загруженные команды могут быть выполнены.

Блок 1 содержит 1280 слов, блок 2 содержит 32 слова, оба блока расположены только в пространстве памяти данных.

ОЗУ однократного доступа SARAM

Память SARAM позволяет либо записывать, либо считывать данные в/из неё в одном машинном цикле. В случае одновременных запросов на чтение и запись к одному блоку SARAM, запросы обрабатываются последовательно (два машинных цикла) с приоритетом запроса на запись, ЦПУ переводится в состояние ожидания. Обслуживание однократных запросов (только чтение или только запись) к блоку SARAM или одновременных запросов к разным блокам SARAM производится на полной скорости без состояний ожидания ЦПУ.

Объём SARAM составляет 46056 слов ×16 бит. Все блоки SARAM расположены только в пространстве памяти данных.

SARAM состоит из следующих блоков:

- S1_0: 2096 слов, диапазон адресов: 4820h — 4FFFh;
- S1_1: 8192 слов, диапазон адресов: 5000h — 6FFFh;
- S2_0: 2096 слов, диапазон адресов: 7440h — 7FFFh;
- S2_1: 8192 слов, диапазон адресов: 8000h — 9FFFh;
- S2_2: 8192 слов, диапазон адресов: A000h — BFFFh;
- S2_3: 8192 слов, диапазон адресов: C000h — DFFFh;
- S2_4: 8192 слов, диапазон адресов: E000h — FFFFh.

FLASH память

Внутрикристальная FLASH память расположена только в пространстве памяти программ.

Уровнем сигнала на выводе MP/MC# производится выбор использования в качестве памяти программ либо внешней памяти, либо внутрикристальной FLASH памяти. При низком уровне (режим «мирокомпьютер») используется внутрикристальная FLASH память.

Объём FLASH составляет 64K слов ×16 бит.

Содержимое внутрикристальной FLASH памяти может многократно быть стерто и перезаписано при отладке или обновлении версии программного обеспечения микроконтроллера. Изменение содержимого FLASH памяти (запись, стирание) осуществляется внешней специальной программой через стандартный IEEE 1149.1 (JTAG) порт микроконтроллера.

7 Периферийные устройства

Интегрированные в ИС 1867ВЦ10Т периферийные устройства описываются в следующих подразделах:

- Интерфейс внешней памяти.
- Модуль менеджера событий EV.
- Модуль двойного аналого-цифрового преобразователя ADC.
- Модуль последовательного периферийного интерфейса SPI.
- Модуль последовательного коммуникационного интерфейса SCI.
- Сторожевой таймер WD и модуль сигналов прерывания в реальном времени.

- Модуль интерфейса CAN.
- Модуль интерфейса I2C.

7.1 Интерфейс внешней памяти

ИС 1867ВЦ10Т может адресовать свыше 64 К слов памяти или регистров в пространстве программы, данных и пространстве I/O. Внутрикристальная память, когда она разрешена, исключает некоторое пространство адресов из внекристального диапазона.

ЦП ИС 1867ВЦ10Т выполняет выборку кода команды, считывание данных и запись данных в одном и том же машинном цикле. Поэтому из внутрикристальной памяти ЦП может производить все три эти операции в одном и том же машинном цикле.

Внешний интерфейс упорядочивает эти операции сначала для завершения записи данных, а потом для считывания данных в конце чтения программы. Внешний интерфейс мультиплексирует внутренние шины адреса и данных в одну внешнюю адресную шину и одну внешнюю шину данных.

ИС удовлетворяет широкому набору требований к внешнему интерфейсу. Адресное пространство программы, данных и I/O обеспечивают интерфейс к памяти программ, данных и I/O, максимизируя производительность системы. Полный 16-битный адрес и шина данных с сигналами PS#, IS# для выбора пространства позволяют адресовать до 64 К слов в программном пространстве и пространстве I/O.

Из-за внутрикристальной периферии внешнее пространство данных адресует до 32 К слов.

Структура I/O упрощена благодаря тому, что обращение к пространству I/O осуществляется тем же способом, что и к памяти данных.

Устройства I/O картрированы в пространство адресов I/O, используя внешние адресные шины процессора и шины данных тем же способом, что и устройства, картрированные в памяти данных.

Внешний параллельный интерфейс ИС обеспечивает управляющим сигналам упрощение интерфейса к устройствам. Сигнал WR/RD# указывает, пишутся ли или читаются данные в текущем цикле в память.

Сигнал STRB# обеспечивает временные соотношения для всех внешних циклов.

Доступ к памяти устройств I/O может выполняться с различной скоростью, используя вывод READY.

Когда обмен выполняется с медленными устройствами, то процессор ИС ждет до того момента, когда устройство завершит свою функцию, посылая сигнал процессору через вход READY. Как только внешнее устройство установило готовность (высокий уровень READY), то выполнение программы продолжается.

В ИС 1867ВЦ10Т ввод READY должен управляться высоким уровнем для завершения записи или считывания на внутренние шины данных, картируемых регистров I/O и всех внешних адресов.

ИС 1867ВЦ10Т поддерживает считывание с нулевым состоянием ожидания на внешнем интерфейсе.

Запись выполняется за два цикла, чтобы избежать конфликтов на шине. Это позволяет ИС буферизовать переход шины данных от ввода к выводу (или от вы-

вода к вводу) за половину цикла. В большинстве систем с ИС 1867ВЦ10Т соотношение от считывания до записи довольно широкое для минимизации дополнительного цикла записи.

Режимы ожидания могут быть выполнены для доступа к внешним медленным устройствам. Режимы ожидания управляются на границе машинного цикла и инициируются через использование сигнала READY или с использованием программного обеспечения, которое управляет генератором циклов ожидания. Сигнал READY может использоваться для реализации любого количества режимов ожидания.

7.2 Модуль менеджера событий EV

Модуль менеджера событий включает общецелевые (GP) таймеры, устройства полного сравнения, устройства сбора данных (захвата) и схемы квадратурного кодирующего устройства. На рисунке 13 приведена структура модуля менеджера событий.

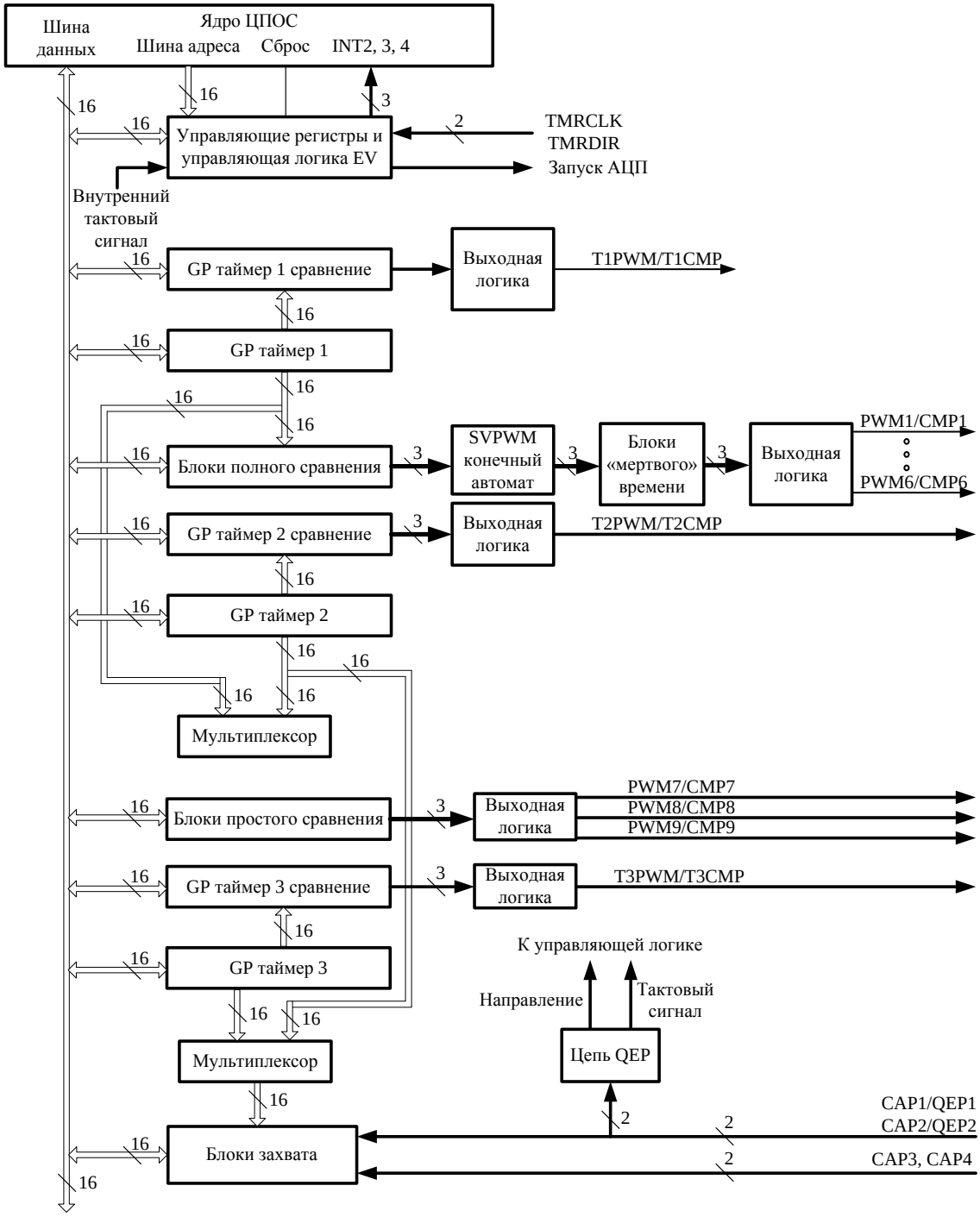


Рисунок 13 – Блок-схема модуля менеджера событий

7.2.1 Общецелевые таймеры GP

На ИС 1867ВЦ10Т есть три GP таймера. GP таймер x (где x = 1, 2, 3) включает:

- 16-битный таймер со счетчиком вверх, вверх/вниз, регистр TxCNT для считывания и записи;

- 16-битный регистр таймера-сравнения (с двойной буферизацией теневым регистром), регистр TxCMPR для считывания и записи;

- 16-битный регистр таймера-периода (с двойной буферизацией теневым регистром), регистр TxPR для считывания и записи;

- 16-битный регистр управления таймером, регистр TxCON для считывания и записи;

- выбираемые внешние или внутренние синхронизирующие импульсы;

- программируемый предварительный делитель (prescalar) для внутренних или внешних синхронизирующих импульсов;

- управляющую и логическую схемы прерывания для четырех маскируемых сигналов прерывания: потеря значимости/заем, переполнение, сравнение и окончание периода;

- вывод выхода таймера сравнения с перестраиваемой конфигурацией режимов (активно-низкий или активно-высокий уровень выходного сигнала), а также режимов форсирования состояний вверх-вниз;

- выбираемое направление (TMRDIR) через вывод входа (счет вверх или вниз, когда выбран режим отсчета по направлению вверх-вниз).

GP таймеры могут управляться независимо или быть синхронизированы один с другим. 32-битный GP таймер может быть конфигурирован соответствующим использованием GP таймера 2 или 3.

Регистр сравнения, включенный в каждый GP таймер, может быть использован для функции сравнения при создании сигнала ШИМ (PWM)-формы. Имеется два однократных и три непрерывных способа функционирования для каждого таймера.

Внутренний или внешний вход синхронизирующих импульсов с программируемым предварительным делителем используются для каждого GP таймера. Режим каждого GP таймера или вывода сравнения конфигурируется через управляющий регистр общецелевыми таймерами (GPTCON). GP таймеры также обеспечивают временную основу для каждого субмодуля менеджера событий: GP таймер 1 для всех компараторов и PWM схем, GP таймер 1 или 2 для простых компараторов и для создания дополнительного компаратора или PWM, GP таймеров 2 или 3 для устройств сбора данных и вычислительных операций с квадратурным импульсом.

Двойная буферизация регистров периода и сравнения позволяет производить программируемое изменение периода таймера (PWM) и длительности импульса PWM во время их работы, если это необходимо.

7.2.2 Устройства полного сравнения

В ИС 1867ВЦ10Т существуют три полных сравнивающих устройства. Эти сравнивающие устройства используют GP таймер 1 как временную базу PWM и реализуют шесть выводов для образования сигнала сравнения и сигнала PWM – формы, используя программную схему обхода «мертвой» зоны. Состояние каждого из этих шести выходов конфигурируется независимо. Сравнивающие регистры устройств сравнения имеют двойную буферизацию, позволяя производить программируемое изменение сравнивающего устройства/длительностей импульса PWM, если это необходимо в процессе работы устройства.

7.2.3 Программируемый генератор обхода «мертвой» зоны

Схема генератора обхода «мертвой» зоны включает три 8-битных счетчика и 8-битный регистр сравнения. Желаемое значение обхода «мертвой» зоны (от 0 до 81,92 мкс) может программироваться в сравнивающем регистре для выводов трех сравнивающих устройств. Образование обхода «мертвой» зоны может быть разблокировано/блокировано для каждого вывода сравнивающего устройства индивидуально. Схема генератора обхода мертвой зоны реализует два вывода (с/без обхода «мертвой» зоны) для каждого сигнала вывода устройства сравнения. Состояния вывода генератора обхода «мертвой» зоны имеют перестраиваемую конфигурацию и изменяются при необходимости через регистр ACTR с двойной буферизацией.

7.2.4 Простые сравнивающие устройства

ИС 1867ВЦ10Т имеет три устройства простого сравнения, которые могут использоваться для реализации трех дополнительных независимых устройств сравнения или сигнала высокой точности PWM – формы. GP таймеры 1 или 2 могут быть выбраны временной основой для трех устройств простого сравнения. Состояния выводов для трех устройств простого сравнения конфигурируются как активно низкие, активно высокие, низко-силовые или высоко-силовые независимо. Регистры устройств простого сравнения имеют двойную буферизацию, позволяя производить программируемое изменение сравнивающего устройства/длительностей импульса PWM, если это необходимо.

Состояния выводов устройств простого сравнения имеют перестраиваемую конфигурацию и изменяются при необходимости через регистр с двойной буферизацией SACTR.

7.2.5 Сравнение или генерация сигнала PWM формы

Свыше 12 выводов устройств сравнения и/или PWM формы могут быть реализованы в ИС 1867ВЦ10Т одновременно. Это три независимые пары (шесть выводов), формируемые тремя устройствами полного сравнения с программируемым обходом «мертвой» зоны, три независимых вывода сравнения или PWM формы, формируемые устройствами простого сравнения, три вывода сравнения и PWM формы, формируемые GP -таймером сравнения.

7.2.6 Характеристики устройства сравнения/формирования сигнала PWM формы

Характеристики устройства сравнения/формирования сигнала PWM-формы следующие:

- 16-битное и 8 нс разрешение;
- программируемый обход «мертвой» зоны для пар выводов PWM от 0 до 81,92 мкс;
- минимум длительности обхода «мертвой» зоны в 120 нс;
- выбор источника частоты PWM, при необходимости;

- изменение длительностей импульса PWM во время и после каждого PWM периода, при необходимости;

- внешние маскируемые сигналы прерывания для защиты по питанию;

- генератор временной диаграммы следования импульсов, для программируемой генерации асимметричного, симметричного и четырехпространственного вектора PWM формы;

- минимизированная нагрузка на ЦП за счет использования автоперезагрузки регистров устройства сравнения и устройства формирования периода.

7.2.7 Устройство сбора данных

Устройство сбора данных обеспечивает функцию регистрации различных событий или переходов. Значения счетчика GP-таймера 2 и/или GP-таймера 3 собираются в данные и запоминаются в двухуровневой памяти стекового обратного (магазинного) типа FIFO, когда выбранные переходы детектированы на выводе входа сбора данных, CAPx для x = 1, 2, 3 или 4. Устройство сбора данных ИС 1867ВЦ10Т состоит из четырех схем сбора данных.

7.2.8 Особенности устройства сбора данных

Устройство сбора данных имеет следующие особенности:

- один 16-битный регистр управления устройством сбора данных, CAPCON, для считывания и записи;

- один 16-битный статусный регистр FIFO устройства сбора данных, CAPFIFO с восемью MSB для операций только для чтения и восемь LSB для операций только для записи;

- возможность выбора GP таймера 2 и/или GP таймера 3 через два 16-битных мультиплексора (MUX). Один MUX выбирает GP таймеры 3 и 4 для схем сбора данных, а другой MUX выбирает GP таймеры 1 и 2 для схем сбора данных;

- четыре 16-битных FIFO стек регистров, один или двух уровневый FIFO стек регистр на каждое устройство сбора данных. Самый верхний регистр каждого стека является только регистром для чтения (FIFOx), где x = 1, 2, 3 или 4;

- четыре входных вывода по сбору данных (CAPx, x = 1 до 4) с одним входным выводом на каждое устройство сбора данных;

- входные выводы CAP1 или CAP2 также могут использоваться как входы в QEP схему;

- режим специфицированного пользователем детектирования фронта импульса на входных выводах;

- четыре маскируемых сигнала прерывания/флагов, CAPINTx, где x = 1, 2, 3, 4.

7.2.9 Схема квадратурного кодирующего устройства QEP

Два входа устройства сбора данных (CAP1 и CAP2) могут использоваться для согласования внутрикристальной схемы QEP с квадратурно-кодированным

импульсом. Полная синхронизация этих входов выполняется внутрикристально. Направление или последовательность подачи кодированного импульса определяется, а GP таймеры 2 или 3 дают положительное или отрицательное приращение положительным и отрицательным фронтам импульса с двух входных сигналов (четыре частоты входного импульса).

7.3 Модуль аналогово-цифрового преобразователя

Упрощенная структурная блок-схема модуля АЦП показана на рисунке 14.

Модуль АЦП состоит их двух 12-разрядных АЦП с двумя встроенными схемами выборки-хранения (sample – and – hold S/H). ИС 1867ВЦ10Т имеет 16 каналов ввода аналоговых сигналов. Восемь аналоговых входов предусмотрены для каждого устройства АЦП через аналоговый мультиплексор 8/1. Минимальное общее время преобразования для каждого АЦП – 1,25 мкс. Общая точность для каждого АЦП ± 1,5 LSB. Опорное напряжение для модуля АЦП должно быть обеспечено внешним источником питания через два вывода, U_{REFHI} и U_{REFLO}.

Результат цифрового преобразования выражается как:

$$\text{Цифровой результат} = 1023 \times \frac{\text{Входное аналоговое напряжение} - U_{REFLO}}{U_{REFHI} - U_{REFLO}}.$$

Модуль АЦП включает:

- два канала ввода (один на каждое устройство АЦП), которые могут участвовать в осуществлении выборки и хранения (S/H операции) и преобразовании одновременно (каждое устройство АЦП может выполнять как одиночные, так и непрерывные S/H операции или операции по преобразованию);
- два двухуровневых по глубине FIFO регистра для устройства АЦП1 или АЦП2;
- каждое устройство АЦП может начать операцию командой программного обеспечения, внешней подачей сигнала на выход устройства или через менеджер событий при событии на каждом из GP таймеров/компараторов или на блоке захвата четырех выходов;
- управляющий регистр АЦП имеет двойную буферизацию (с теневым регистром) и может быть записан в любое время. Новое преобразование АЦП может начаться немедленно или когда предыдущий процесс преобразования завершен в соответствии с битами управляющего регистра;
- в конце каждого преобразования устанавливается флаг сигнала прерывания и прерывание выполняется, если оно не маскировано или разблокировано.

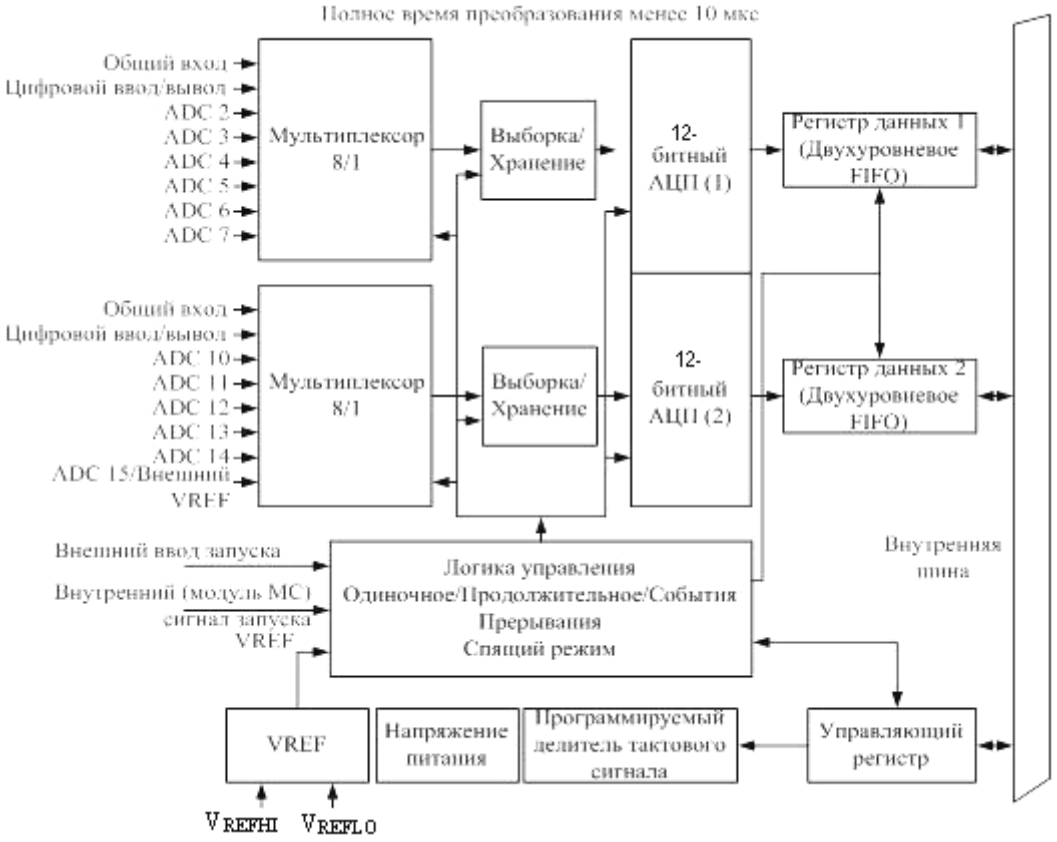


Рисунок 14 – Структурная схема модуля аналого-цифрового преобразователя

7.4 Модуль последовательного периферийного интерфейса SPI

ИС 1867ВЦ10Т содержит модуль последовательного периферийного интерфейса с четырьмя выводами. SPI – это высокоскоростной синхронный последовательный порт I/O, который осуществляет прием/передачу последовательного потока битов программируемой длины (от одного до восьми битов) в/из SPI на программируемой скорости передачи.

Обычно SPI используется для коммуникаций между ПЦОС контроллером и внешними периферийными устройствами или другим процессором. Типичные применения SPI включают внешний I/O или расширение периферии ПЦОС, таких как сдвиговые регистры, драйверы дисплея и дополнительные АЦП и ЦАП. Сообщения от множества устройств поддерживаются через операции SPI ведущий/ведомый или главный/подчиненный.

Особенностями модуля SPI являются:

- Четыре внешних вывода:

SPISOMI: SPI slave-output/master-input (ведомый-выход/ведущий-вход) вывод или общецелевой двунаправленный вывод I/O;

SPISIMI: SPI slave-input/master-output (ведомый - вход/ведущий- выход) вывод или общецелевой двунаправленный вывод I/O;

SPISTE: SPI slave- transmit- enable (ведомый -передача-разрешение) вывод или общецелевой двунаправленный вывод I/O;

SPICLK: SPI вывод синхронизации или общецелевой двунаправленный вывод I/O.

- Два программируемых режима работы: ведущий/ведомый.
- Программируемая скорость передачи в бодах: 125 разных программируемых скоростей.
- Программируемая длина слова: от одного до восьми бит данных.
- Четыре синхронизирующих схемы, управляющие полярностью синхронизации и фазой синхроимпульса:

1) На спаде фронта синхроимпульса без фазовой задержки (SPICLK – активный высокий). В этом случае SPI передает данные на спаде фронта синхроимпульса SPICLK и получает данные на подъеме фронта синхроимпульса SPICLK.

2) На спаде фронта синхроимпульса с фазовой задержкой (SPICLK – активный высокий). В этом случае SPI передает данные на одной половине цикла впереди снижающегося фронта синхроимпульса SPICLK и получает данные на снижающемся фронте синхроимпульса SPICLK.

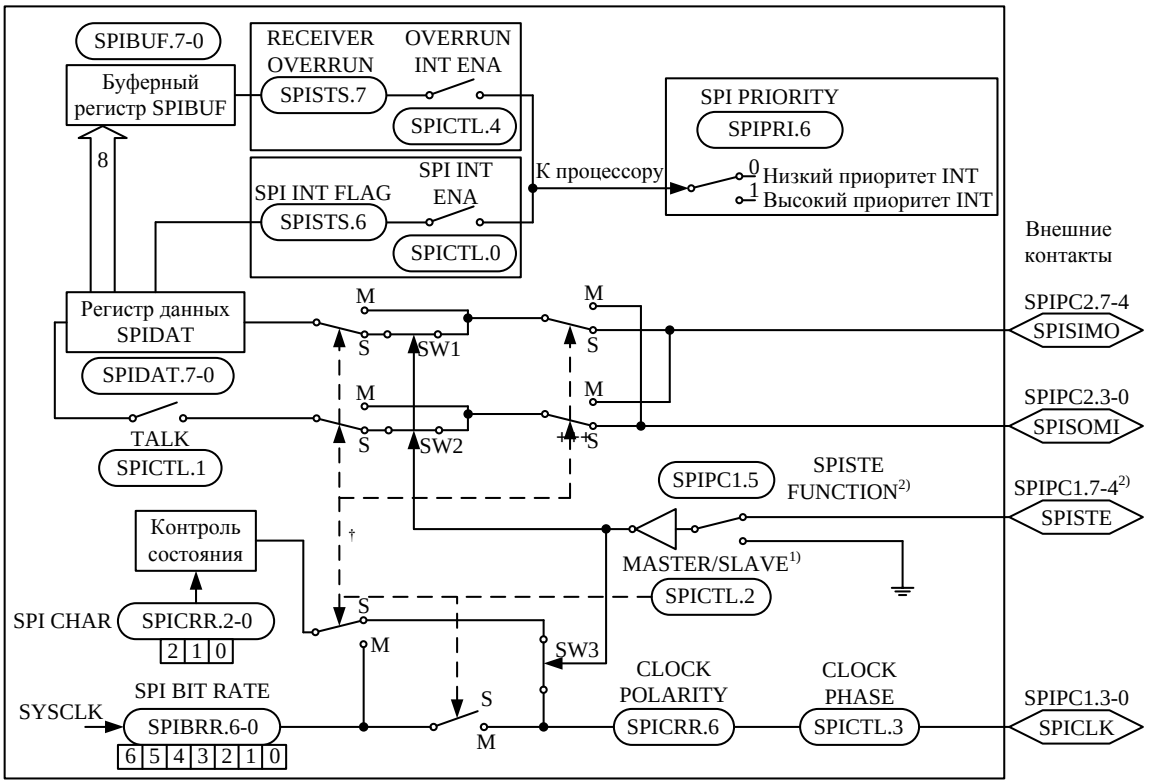
3) На подъеме фронта синхроимпульса без фазовой задержки: (SPICLK – неактивный низкий). В этом случае SPI передает данные на подъеме фронта синхроимпульса SPICLK и получает данные на спаде фронта синхроимпульса SPICLK.

4) На подъеме фронта синхроимпульса с фазовой задержкой (SPICLK – неактивный низкий). В этом случае SPI передает данные на одной половине цикла перед спадом фронта синхроимпульса SPICLK и получает данные на подъеме фронта SPICLK.

- Одновременные операции по получению и передаче (функция передачи может быть блокирована программным обеспечением).
- Операции передачи и приема выполняются через обработку прерывания или путем алгоритмов опроса соответствующих бит.
- Десять управляющих регистров SPI, расположенных в блоке управляющего регистра, начинающегося с адреса 7040h.

Примечание – В этом модуле все регистры 8-битные, подключены к 16-битной периферийной шине. В режиме чтения из регистра данные из регистра находятся в нижнем байте (7-0), а в верхнем байте (15-8) данные считываются как нули. Запись в верхний байт не влияет на работу устройства.

Блок-схема четырехвыводного модуля последовательного периферийного интерфейса представлена на рисунке 15.



1) Структурная схема приведена в режиме ведомого (Slave).

2) Ввод SPISTE запрещен, это значит, что данные могут быть переданы или приняты в этом режиме. Следует отметить, что ключи SW1, SW2 и SW3 закрыты в этой конфигурации.

Рисунок 15 – Блок-схема четырехвыводного модуля последовательного периферийного интерфейса

7.5 Модуль последовательного коммуникационного интерфейса SCI

ИС 1867ВЦ10Т включает модуль последовательного коммуникационного интерфейса (SCI). SCI модуль обеспечивает цифровое взаимодействие между ЦП и другими асинхронными периферийными устройствами, которые используют стандартный, без возвращения к нулю, формат NRZ. SCI приемник и передатчик имеют двойную буферизацию, каждый из которых имеет свои собственные отдельные биты маскирования сигналов прерывания. Оба они могут управляться независимо или одновременно в полностью дуплексном режиме. Для обеспечения целостности данных SCI проверяет полученные данные на предмет разрыва линии, равенства, переполнения и ошибок кадровой синхронизации. Скорость передачи бита (бод) является программируемой на более чем 65000 различных скоростях, которые выбираются через 16-битный регистр выбора скорости.

Особенности SCI модуля:

- Два внешних вывода (SCITXD – это SCI вывод передаваемых данных или двунаправленный общецелевой вывод I/O. SCIRXD – это SCI вывод принимаемых данных или двунаправленный общецелевой вывод I/O).

- Программируемая скорость передачи данных в бодах (до 64 К различных скоростей).

- Скорость передачи свыше 625 Кбит/с при 10 МГц SYSCLK.

- Программируемый формат слова (один стартовый бит, длина слова данных от одного до восьми бит, бит выбора четный/нечетный, один или два стоповых бита).

- Четыре флага обнаружения ошибки (паритета, разрыва, переполнения и ошибок кадровой синхронизации).

- Два запускающих мультипроцессорных режима (пустой промежуток и адресный бит).

- Полудуплексная или полностью дуплексная операция.

- Функции приемника и передатчика с двойной буферизацией.

- Операции приемника и передатчика могут быть обработаны через управляемые сигналы прерывания или опросом соответствующих флагов.

Передатчик: флаг TXRDY – буферный регистр передатчика готов для получения другого символа и флаг TXEMPTY – сдвиговый регистр передатчика пуст.

Приемник: флаг RXRDY – буферный регистр приемника готов для приема другого символа. Флаг BRKDT – произошел разрыв и RXERROR - мониторинг четырех условий прерывания.

- Раздельные биты маскирования для сигналов прерывания приемника и передатчика (кроме BRKDT).

- Формат NRZ (без возвращения к нулю).

- Одиннадцать управляющих регистров SCI, расположенных в блоке данных управляющего регистра, начинающегося с адреса 7050h.

Примечание - Все регистры в этом модуле 8-битные, соединены с 16-битной периферийной шиной. Когда регистр читается, то данные регистра находятся в младшем байте (7-0), а старшие биты (15-8) считываются как нули.

На рисунке 16 приведена блок-схема SCI модуля.

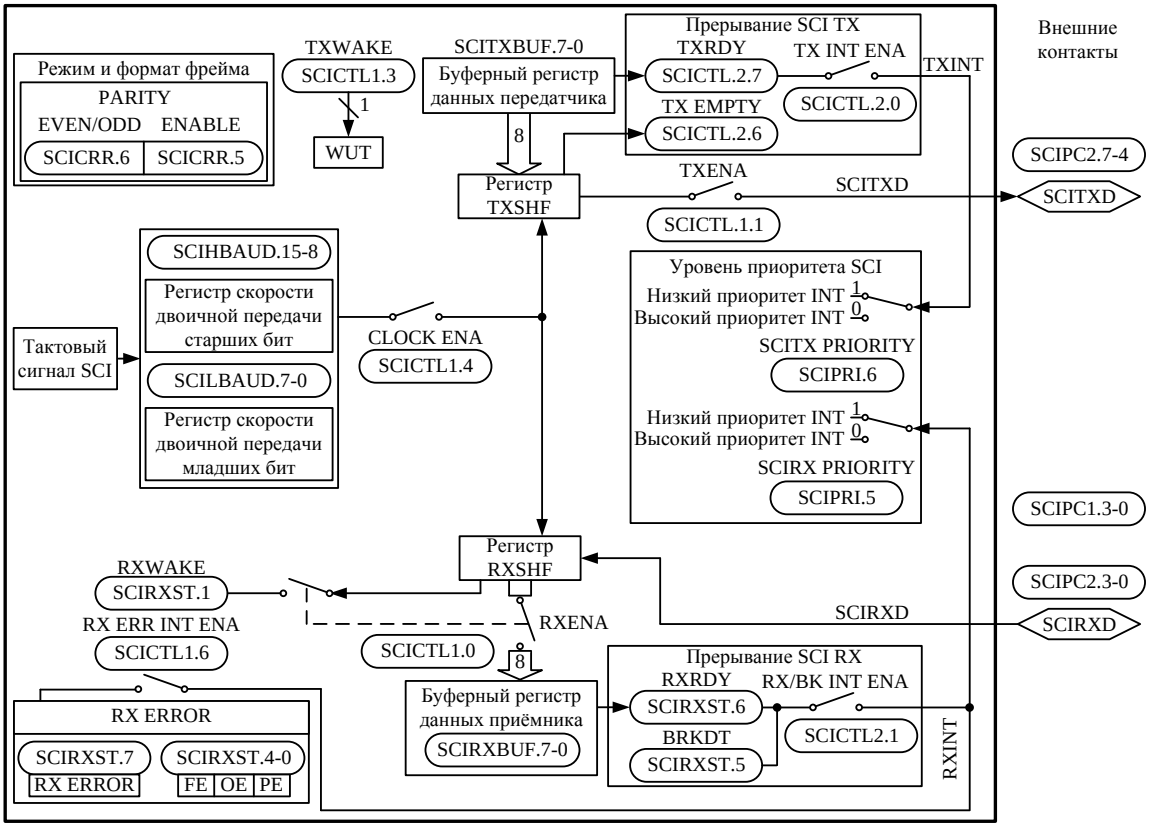


Рисунок 16 – Блок-схема модуля интерфейса последовательной коммуникации (SCI)

7.6 Сторожевой таймер и модуль сигналов прерывания в реальном времени

ИС 1867ВЦ10Т включает сторожевой таймер (WD) и модуль сигналов прерывания в реальном времени (RTI). WD-функция этого модуля выполняет текущий контроль программного и технического обеспечения через сброс системы, если сброс периодически не обслуживается программным обеспечением путем записи правильного ключа. Функция RTI обеспечивает сигналы прерывания на программируемых интервалах. На рисунке 17 показана блок-схема модуля WD/RTI.

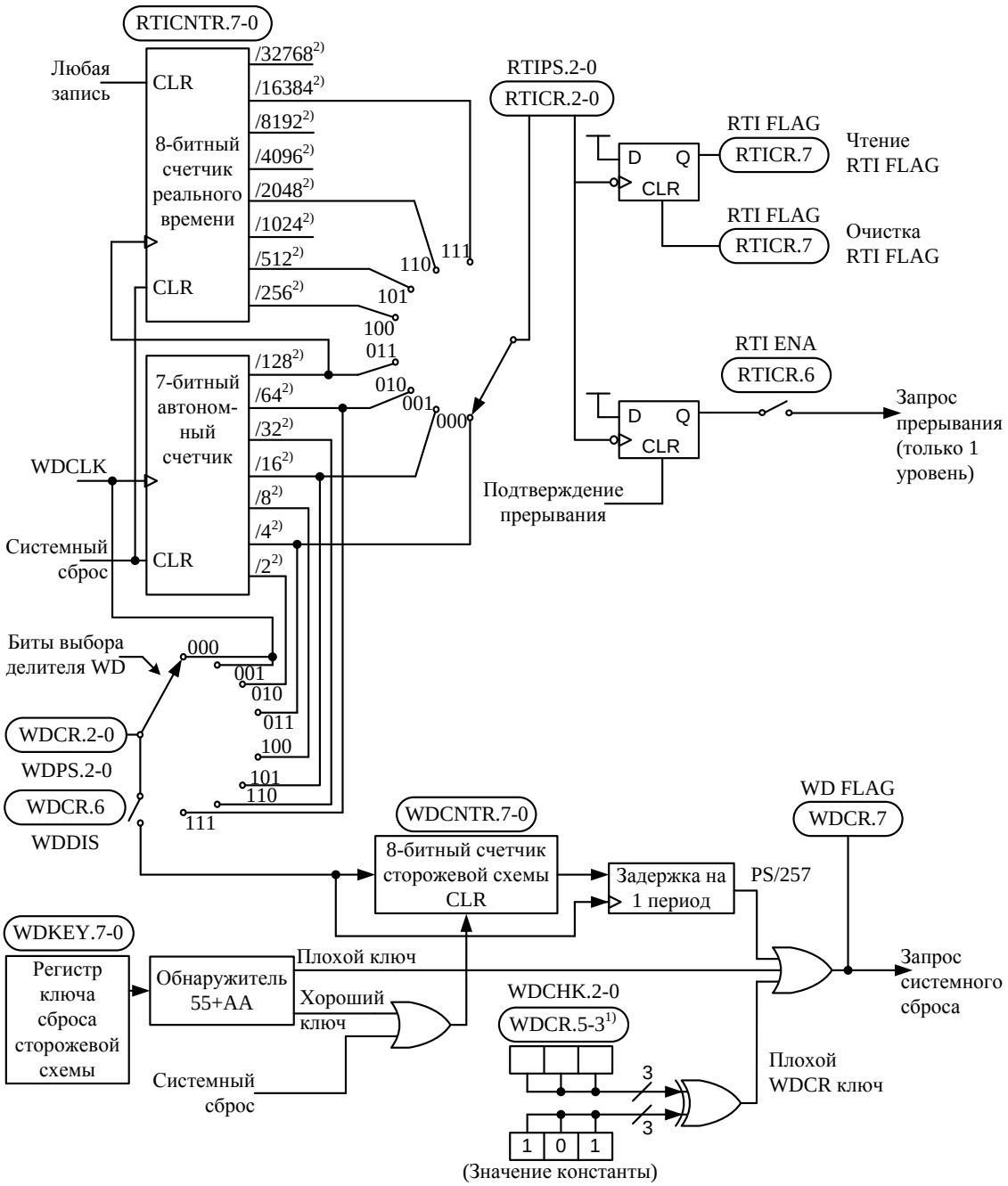
WD таймер имеет следующие особенности:

- Семь различных степеней переполнения WD от 15,63 мс до 1 с.
- Регистр WD для сброса ключом (WDKEY). Он очищает WD счетчик, когда правильное значение записано, формирует системный сброс, если записано неправильное значение в этот регистр.
- WD флаг (WD FLAG), который указывает, произвел ли WD таймер системный сброс.
- Проверка комбинации бит WD, которые инициируют системный сброс, если неправильное значение записано в управляющем регистре WD (WDCR).
- Автоматическая активация WD таймера, как только системный сброс снят.
- Три управляющих регистра WD расположены в блоке управляющего регистра, начинающегося с адреса 7020h.

Модуль сигналов прерывания в реальном времени (RTI) имеет следующие особенности:

- Формирование сигналов прерывания осуществляется на программируемой частоте от 1 до 4096 сигналов прерывания в секунду.
- Инициирование обслуживания возможно как от сигналов прерывания, так и путем опроса флага.
- Два управляющих регистра RTI расположены в блоке данных управляющего регистра, начинающегося с адреса 7020h.

Все регистры в этом модуле являются 8-битными и соединены с 16-битной периферийной шиной. Когда регистр выбирается, то данные регистра находятся в младшем байте (7-0), а в старшем байте биты (15-8) читаются как нули. Запись в старший байт не влияет на работу устройства.



[1)] Запись в биты WDCR.5-3 в конфигурации, отличающейся от нормальной (101) формирует системный сброс.
[2)] Эти значения делителя взяты относительно сигнала WDCLK.

Рисунок 17 – Блок-схема модуля WD/RTI

7.7 Модуль интерфейса I2C

ИС 1867ВЦ10Т включает модуль интерфейса I2C. Модуль интерфейса I2C (далее – I2C) обеспечивает полную поддержку двухпроводного последовательного синхронного интерфейса I2C/SMBUS. Он также совместим с шинами ACCESS.Bus и Philips' I2C. Результат такой совместимости – легкое соединение со многими запоминающими устройствами и устройствами ввода-вывода, включая EEPROM, SRAM, счетчики, АЦП, ЦАП, периферийные устройства.

Функциональные возможности модуля:

- Совместимость с SMBus (Version 1.1 и 2.0), ACCESS.Bus, I2C (Version 2.1).
- Поддержка стандартного (Standard), скоростного (F/S) и высокоскоростного (Hs) режимов.
- Программирование интерфейса для работы в режиме ведущего (Master) или ведомого (Slave).
- Возможность подключения к шине нескольких ведущих устройств (режим Multi-Master).
-Один программно определяемый 7/10-битный адрес ведомого (Slave) устройства.
- Возможность одновременного обращения ко всем устройствам шины, так называемый «общий вызов» (global call).
- Особые возможности SMBus:
  - отслеживание времени ожидания линии SCL;
  - наличие функции PEC (Packet Error Checking – отслеживание ошибок в пакетах данных) с использованием стандартного полинома SMBus;
  - возможность обращения одного или нескольких ведомых (Slave) к ведущему (Master) с получением отклика от последнего.
- Возможность последовательного опроса.
- Функционирование под управлением прерываний.

I2C использует двухпроводной интерфейс для двусторонней связи между устройствами, подключенными к шине. Двунаправленная шина состоит из двух линий: линии данных (Serial Data Line – SDA) и линии тактового сигнала (Serial Clock Line – SCL). Эти линии подключены к источнику питания через подтягивающие резисторы. Шинные формирователи любых устройств, подключаемых к шине, выполняются по схеме с открытым стоком. Устройства могут выставить только низкий уровень на соответствующей линии. Следовательно, обе линии SDA и SCL реализуют функцию «монтажное И».

Структурная схема модуля изображена на рисунке 18

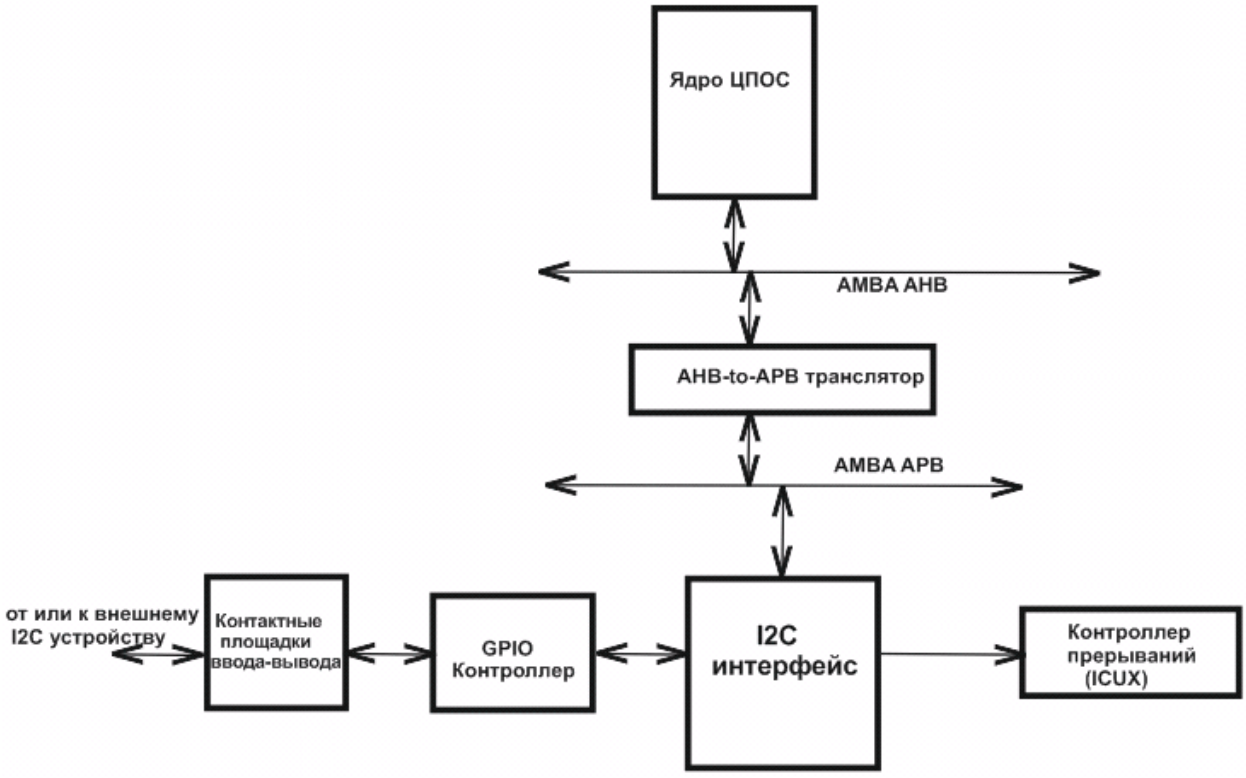


Рисунок 18 – Структурная схема модуля интерфейса I2C

7.8 Модуль последовательного интерфейса связи CAN

ИС 1867ВЦ10Т включает последовательный интерфейс связи (далее – модуль CAN) эффективно поддерживающий распределенное управление в реальном масштабе времени с высокой помехозащищенностью.

Модуль CAN выполняет все функции для приема и передачи сообщений на шине. Данные, предназначенные для передачи, записываются в соответствующие регистры. Состояние модуля CAN и возникшие ошибки проверяются чтением регистров состояния. Любое сообщение, передаваемое по шине, проверяется на наличие ошибок, согласуется по фильтрам и сохраняется в регистрах приемного буфера при выполнении всех условий.

CAN модуль поддерживает следующие типы сообщений:

- сообщения данных;
- удаленный запрос данных;
- сообщение об ошибке;
- сообщение о перезагрузке.

Существует два типа сообщений данных:

- стандартное сообщение;
- расширенное сообщение.

В микроконтроллере 1867ВЦ10Т находится модуль интерфейса CAN, который представлен на рисунке 19, включает в себя два идентичных независимых

блока CAN (в дальнейшем CAN узел 0 и CAN узел 1), с общим ОЗУ, между которыми есть отличие лишь в подключении к выводам.

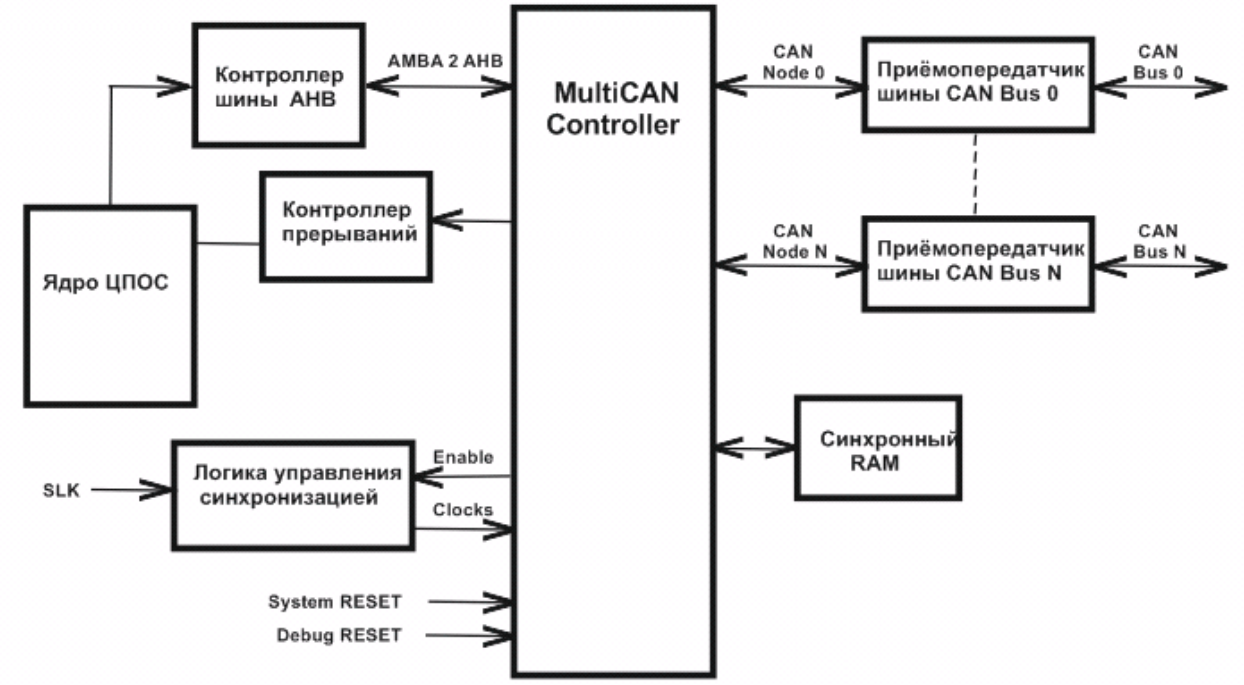


Рисунок 19 – Модуль интерфейса CAN

Помимо стандартной передачи сообщений, когда каждый узел передает на свои выходы, в контроллере реализовано логическое «И» передач CAN узлов (вывод результата сложения сигналов передачи обоих узлов на один выход).

8 Эмуляция, основанная на скан-цепочках

ИС 1867ВЦ10Т использует эмуляцию, основанную на скан-цепочках для поддержки аппаратной и программной отладки. Последовательный интерфейс сканирования обеспечен портом тестового доступа (test-access port). Эмуляция, основанная на скан-цепочках, позволяет эмулятору контролировать и управлять процессором в системе без использования сложных кабелей для обеспечения соединения со всеми выводами от ИС.

9 Инструкции ИС 1867ВЦ10Т

ИС 1867ВЦ10Т реализует понятную систему команд, которая поддерживает цифровые операции обработки сигналов и операции общецелевого назначения, такие как мультиобработка (multiprocessing) и высокоскоростное управление. Исходный код для ПЦОС М1867ВМ1, 1867ВМ2, 1867ВЦ5Т совместим снизу вверх с кодом ИС 1867ВЦ10Т.

Для получения максимальной пропускной способности следующая команда предварительно вызывается во время выполнения текущей операции. Так как те же линии данных используются для получения внешних данных, программы или пространства I/O, то число циклов, которое требует команда для своего выполне-

ния, зависит от того, происходит выборка следующего операнда из внутренней или из внешней памяти. Наиболее высокая пропускная способность ИС достигается путем использования внутрикристальной памяти данных и программы, внутренней и внешней памяти программ.

9.1 Режимы адресации

Система команд 1867ВЦ10Т использует четыре основных режима адресации памяти:

- прямую;
- косвенную;
- непосредственную;
- регистровую.

В случае прямой адресации слово команды содержит младшие семь бит адреса памяти данных. Это поле соединяется девятью битами указателя страницы памяти данных (DP) для образования 16-битного адреса памяти данных. Таким образом, в режиме прямой адресации память данных разбивается на страницы общим числом 512 страниц, при этом каждая страница содержит 128 слов.

Косвенная адресация имеет доступ к памяти данных через вспомогательные регистры. В этом режиме адресации адрес операнда содержится во вспомогательном регистре, выбранном как текущий. Восемь вспомогательных регистров (AR0-AR7) обеспечивают гибкую и эффективную косвенную адресацию.

Для выбора специфицированного вспомогательного регистра указатель вспомогательного регистра ARP загружается значением от 0 до 7 для AR0 - AR7, соответственно.

Существует семь видов косвенной адресации: положительное приращение или отрицательное приращение, постиндексация через сложение или вычитание из значения текущего регистра содержимым AR0, одиночная косвенная адресация без положительных/отрицательных приращений, бит-реверсная адресация (bit-reversed), которая используется в быстром преобразовании Фурье (БПФ) с положительным/отрицательным приращением. Все операции выполняются на текущем вспомогательном регистре в том же цикле, что и команда-оригинал, следя за тем, какой текущий вспомогательный регистр и ARP могут быть изменены.

В непосредственной адресации данные операнда содержатся в слове команды или словах команды. Есть два типа непосредственной адресации - длинная и короткая. При короткой непосредственной адресации данные содержатся в битах однословной команды. При длинной непосредственной адресации данные содержатся во втором слове двухсловной команды.

Режим непосредственной адресации используется для данных, которые не нуждаются в хранении или используются более, чем один раз во время выполнения программы, например, инициализация (установка в исходное состояние) значений или постоянных величин.

Режим регистровой адресации использует операнды, находящиеся в регистрах ЦП, или явно с прямым обращением к специальному регистру или опосредованно, командами, которые обращаются к некоторым регистрам. Во втором случае ссылка операнда упрощается, так как 16-битное значение может использо-

ваться без указания полного 16-битного адреса операнда или непосредственного значения.

9.2 Особенности повторения команд (repeat)

Функция повторения команд может использоваться с командами (как указано в таблице 17) такими как умножение/суммирование (MAC и MACD), обмен данными между блоками (BLDD и BLDP), пересылки I/O (IN/OUT), считывания/записей таблицы (TBLR/TBLW). Эти команды, обычно представленные как мультицикл, находятся в конвейерном режиме. Когда используется их повторение, то они становятся одноцикловыми командами. Например, команда чтения таблицы может занять три или больше циклов для однократного выполнения, но при повторении команды ячейки таблицы могут читаться в каждом цикле.

Счетчик повтора (RPTC) загружается из ячейки памяти, расположенной по адресу, определяемому прямой или косвенной адресацией или 8-битным непосредственным значением, если используется короткая непосредственная адресация. Регистр RPTC загружается командой RPT. Это позволяет выполнить эту команду максимум N+1. RPTC очищается через сброс. Как только команда повтора RPT декодируется, все сигналы прерывания, включая NMI (но исключая сброс), блокируются до завершения цикла RPT.

9.3 Система команд

В этом разделе обобщены коды операций (опкоды) в таблице инструкций для процессора цифровой обработки сигналов 1867ВЦ10Т. Эта система команд является расширением системы команд М1867ВМ1, 1867ВМ2, 1867ВЦ5Т. Команды расположены в соответствии с их функциями и упорядочены по мнемонике в пределах каждой категории. В таблице 16 приведены символы, которые используются в таблице итоговой системы команд (см. таблицу 17).

Число слов, которое команда занимает в программной памяти, приведено в графе “Слов/Циклов” таблицы 17. Несколько команд имеют два значения количества слов, они разделены косой чертой (/). В этих случаях различные форматы команды занимают разное количество слов. Например, ADD команда занимает одно слово, если операнд является коротким непосредственным, или два слова, если операнд является длинным непосредственным.

Количество циклов, которое требует команда для выполнения, находится также в графе “Слов/Циклов” таблицы 17. Предполагается, что все команды выполняются из внутренней программной памяти (ОЗУ) и с данными из внутренней памяти DARAM (двойного доступа).

В таблице 16 приведены данные о символах (сокращениях), используемых в таблице 17.

Таблица 16 – Символы и сокращения

Символ	Описание
ACC	Аккумулятор
AR	Вспомогательный регистр
ARX	3-битное значение в инструкциях LAR и SAR для определения

	вспомогательного регистра, который будет загружаться инструкцией (LAR) или сохраняться инструкцией (SAR)

Продолжение таблицы 16

Символ	Описание
BITX	4-битное значение (битный код), которое определяет, какой бит определенного значения памяти данных будет тестироваться (проверяться) инструкцией BIT
CM	2-битное значение. Инструкция CMPR выполняет сравнение, указанное значением CM: Если CM = 00, проверяет (текущий AR = AR0) Если CM = 01, проверяет (текущий AR < AR0) Если CM = 10, проверяет (текущий AR > AR0) Если CM = 11, проверяет (текущий AR = AR0)
IAAA AAAA	(За I следует семь A). I бит отражает способ адресации (I=0 – прямая адресация) или (I=1 – косвенная адресация). Когда используется прямая адресация, то AAA AAAA это последние значащие биты (LSB) адреса в странице памяти данных. Для косвенной адресации AAA AAAA – это биты, определяющие режим косвенной адресации, т. е. способ манипуляции вспомогательными регистрами и номер регистра
IIII IIII	(Восемь I) 8-битная константа, используется в короткой непосредственной адресации
I IIII IIII	(Девять I) 9-битная константа используется для короткой непосредственной адресации в инструкции LDP
I IIII IIII IIII	(Тринадцать I) 13-битная константа, используемая в короткой непосредственной адресации для инструкции MPY
I NTR#	5-битное значение представляет число от 0 до 31. Инструкция INTR использует это число для изменения программного управления к одному из 32-х адресов векторов прерывания
PM	2-битное значение, копируемое в PM статусного регистра ST1 инструкцией SPM
SHF	3-битное значение левого сдвига
SHFT	4-битное значение левого сдвига
TP	2-битное значение, используемое условными инструкциями, для указания типа условия TP = 00 – вывод BIO низкий TP = 01 – TC = 1 TP = 10 – TC=0 TP = 11 – без условий

Окончание таблицы 16

Символ	Описание
ZLVC ZLVC	Два 4-битных поля, каждое из которых представляет следующие условия: ACC = 0 Z ACC < 0 L Переполнение V Перенос C Инструкция с условиями содержит два 4-битных поля. Поле четырех младших битов (биты 0-3) инструкции – это поле маски. 1 в соответствующем бите маски указывает, какое из условий будет проверяться. Например, для проверки ACC ≥ 0 необходимо установить биты Z и L в 1 и биты B и C установить в 0. Поле Z устанавливается для тестирования условия ACC = 0, а L поле сбрасывается для тестирования условия ACC ≥ 0. Второе 4-битное поле (биты 4-7) указывают состояние условий для тестирования. Возможные условия, задаваемые этими 8 битами, показаны в описании для инструкций BCND, CC и RETC
+ 1 слово	Второе слово опкода инструкции. Это второе слово содержит константу. В зависимости от инструкции эта константа имеет или длинное непосредственное значение, или адрес программной памяти, или адрес порта I/O, или картируемый регистр I/O
УВ	Условие выполнено
УНВ	Условие не выполнено

Таблица 17 – Итоговая таблица инструкций ИС 1867ВЦ10Т

Мнемоника	Описание	Слов/ Циклов	Опкод
ABS	Абсолютное значение ACC	1/1	1011 1110 0000 0000
ADD	Прибавляет к ACC со сдвигом от 0 до 15 операнд с прямой/косвенной адресацией	1/1	0010 SHFT IAAA AAAA
ADD	Прибавляет к ACC со сдвигом от 0 до 15 операнд с длинной непосредственной адресацией	2/2	1011 1111 1001 SHFT +1 слово
ADD	Прибавляет к ACC со сдвигом 16 операнд с прямой/косвенной адресацией	1/1	0110 0001 IAAA AAAA
ADD	Прибавляет к ACC операнд с короткой непосредственной адресацией	1/1	1011 1000 IIII IIII
ADDC	Прибавляет к ACC операнд с прямой/косвенной адресацией и перенос	1/1	0110 0010 IAAA AAAA
ADDT	Прибавляет к ACC операнд с прямой/косвенной адресацией и сдвигом от 0 до 15, указанным в TREG	1/1	0110 0011 IAAA AAAA
ADRK	Прибавление константы с короткой непосредственной адресацией к текущему AR	1/1	0111 1000 IIII IIII
AND	Логическое «И» с ACC операнда со сдвигом от 0 до 15 с прямой/косвенной адресацией	1/1	0110 1110 IAAA AAAA
AND	Логическое «И» с ACC операнда со сдвигом от 0 до 15 с длинной непосредственной адресацией	2/2	1011 1111 1011 SHFT
AND	Логическое «И» с ACC операнда со сдвигом от 16 с длинной непосредственной адресацией	2/2	1011 1110 1000 0001 +1 слово
APAC	Прибавить PREG к ACC	1/1	1011 1110 0000 0100
B	Безусловный переход	2/4	0111 1001 1AAA AAAA
BACC	Переход по адресу, указанному в ACC	1/4	1011 1110 0010 0000
BANZ	Переход, если текущий AR не равен 0, косвенная адресация	2 /4(УВ) /2(УНВ)	0111 1011 1AAA AAAA +1 слово
BCND	Условный переход	2 /4(УВ) /2(УНВ)	1110 00TP ZLVC ZLVC +1 слово
BIT	Проверка бита	1/1	0100 BITX IAAA AAAA
BITT	Проверка бита в операнде с прямой/косвенной адресацией, указанного в TREG	1/1	0110 1111 IAAA AAAA
BLDD	Пересылка блока из ячеек памяти данных, адресуемых длинной непосредственной адресацией, в ячейки память данных, адресуемых прямо/косвенно	2/3	1010 1000 IAAA AAAA +1 слово
BLDD	Пересылка блока из ячеек памяти данных, адресуемых прямо/косвенно, в ячейки памяти данных, адресуемых длинной непосредственной адресацией	2/3	1010 1001 IAAA AAAA +1 слово
BLPD	Пересылка блока из ячеек памяти программ, адресуемых вторым словом инструкции, в ячейки памяти данных, адресуемых прямо/косвенно	2	1010 0101 IAAA AAAA +1 слово
CAL	Вызов подпрограммы. Косвенная адресация	2/4	0111 1010 1AAA AAAA +1 слово
CALA	Вызов подпрограммы с адресом, указанным в ACC	1/4	1011 1110 0011 0000
CLRC INTM	Очистка бита INTM	1/1	1011 1110 0100 0000
CLRC OVM	Очистка бита OVM	1/1	1011 1110 0100 0010
CLRC SXM	Очистка бита SXM	1/1	1011 1110 0100 0110

Продолжение таблицы 17

Мнемоника	Описание	Слов/ Циклов	Опкод
CLRC TC	Очистка бита TC	1/1	1011 1110 0100 1010
CLRC XF	Очистка бита XF	1/1	1011 1110 0100 1100
CLRC C	Очистка бита C	1/1	1011 1110 0100 1110
CLRC CNF	Очистка бита CNF	1/1	1011 1110 0100 0100
CMPL	Дополнение ACC	1/1	1011 1110 0000 0001
CMPR	Сравнение текущего AR с AR0	1/1	1011 1111 0100 01CM
DMOV	Пересылка данных из ячейки памяти данных, адресуемой прямо/косвенно в следующую ячейку памяти данных	1/1	0111 0111 IAAA AAAA
IDLE	Ожидание прерывания	1/1	1011 1110 0010 0010
IN	Чтение данных из ячейки пространства ввода-вывода, адресуемой 2-ым словом инструкции, в ячейку, адресуемую прямо/косвенно	2/2	1010 1111 IAAA AAAA +1 слово
INTR	Программное прерывание	1/4	1011 1110 011I NTR#
LACC	Загружает в ACC со сдвигом от 0 до 15 операнд с прямой/ косвенной адресацией	1/1	0001 SHFT IAAA AAAA
LACC	Загружает в ACC со сдвигом от 0 до 15 операнд с длинной непосредственной адресацией	2/2	1011 1111 1000 SHFT +1 слово
LACC	Загружает в ACC со сдвигом 16 операнд с прямой/ косвенной адресацией	1/1	0110 1010 IAAA AAAA
LACL	Загружает в ACC операнд с прямой/косвенной адресацией	1/1	0110 1001 IAAA AAAA
LACL	Загружает в ACC операнд с короткой непосредственной адресацией	1/1	1011 1001 IIII IIII
LACT	Загружает в ACC со сдвигом от 0 до 15, указанным TREG, в операнд с прямой/ косвенной адресацией	1/1	0110 1011 IAAA AAAA
LAR	Загрузка указанного AR операндом с прямой/косвенной адресацией	1/2	0000 0ARX IAAA AAAA
LAR	Загрузка указанного AR операндом с короткой непосредственной адресацией	1/2	1011 0ARX IIII IIII
LAR	Загрузка указанного AR операндом с длинной непосредственной адресацией	2/2	1011 1111 0000 1ARX +1 слово
LDP	Загрузка указателя страниц операндом с прямой/косвенной адресацией	1/2	0000 1101 IAAA AAAA
LDP	Загрузка указателя страниц операндом с короткой непосредственной адресацией	1/2	1011 110I IIII IIII
LPH	Загрузка старшего слова PREG операндом с прямой/косвенной адресацией	1/1	0111 0101 IAAA AAAA
LST	Загрузка регистра ST0 операндом с короткой непосредственной адресацией	1/2	0000 1110 IAAA AAAA
LST	Загрузка регистра ST1 операндом с короткой непосредственной адресацией	1/2	0000 1111 IAAA AAAA
LT	Загрузка PREG операндом с прямой/косвенной адресацией	1/1	0111 0011 IAAA AAAA
LTA	Загрузка TREG операндом с прямой/косвенной адресацией и сложение ACC с PREG	1/1	0111 0000 IAAA AAAA
LTD	Загрузка TREG операндом с прямой/косвенной адресацией и сложение ACC с PREG и пересылка операнда с адресом операнда + 1	1/1	0111 0010 IAAA AAAA

Продолжение таблицы 17

Мнемоника	Описание	Слов/ Циклов	Опкод
LTP	Загрузка TREG операндом с прямой/косвенной адресацией и сохранение PREG в ACC	1/1	0111 0001 IAAA AAAA
LTS	Загрузка TREG операндом с прямой/косвенной адресацией и вычитание PREG из ACC	1/1	0111 0100 IAAA AAAA
MAC	Умножение и накопление с прямой/косвенной адресацией	2/3	1010 0010 IAAA AAAA +1 слово
MACD	Умножение операнда с прямой/косвенной адресацией с накоплением и пересылка операнда по адресу операнда + 1	2/3	1010 0011 IAAA AAAA +1 слово
MAR	Модификация текущего AR и/или ARP операндом с косвенной адресацией (NOP с прямой адресацией)	1/1	1000 1011 IAAA AAAA
MPY	Умножение TREG с операндом с прямой/косвенной адресацией	1/1	0101 0100 IAAA AAAA
MPY	Умножение TREG с операндом (13-битная константа) с короткой непосредственной адресацией	1/1	110I IIII IIII IIII
MPYA	Умножение с операндом с прямой/косвенной адресацией и накопление	1/1	0101 0000 IAAA AAAA
MPYS	Умножение с операндом с прямой/косвенной адресацией и вычитание из предыдущего произведения	1/1	0101 0001 IAAA AAAA
MPYU	Беззнаковое умножение с операндом с прямой/косвенной адресацией	1/1	0101 0101 IAAA AAAA
NEG	Изменение знака ACC	1/1	1011 1110 0000 0010
NMI	Немаскируемое программное прерывание	1/4	1011 1110 0101 0010
NOP	Нет операции	1/1	1000 1011 0000 0000
NORM	Нормализация содержимого ACC, косвенная адресация	1/1	1010 0000 IAAA AAAA
OR	Логическое «ИЛИ» ACC и операнда с прямой/косвенной адресацией	1/1	0110 1101 IAAA AAAA
OR	Логическое «ИЛИ» ACC и операнда со сдвигом от 0 до 15 с длинной непосредственной адресацией	2/2	1011 1111 1100 SHFT +1 слово
OR	Логическое «ИЛИ» ACC и операнда со сдвигом 16 с длинной непосредственной адресацией	2/2	1011 1110 1000 0010 +1 слово
OUT	Запись данных в ячейку пространства ввода-вывода, адресуемую вторым словом инструкции из ячейки памяти данных, адресуемой прямо/косвенно	2/3	0000 1100 IAAA AAAA +1 слово
PAC	Загрузка ACC PREG	1/1	1011 1110 0000 0011
POP	Запись верхушки стека в младшее слово ACC	1/1	1000 1011 0000 0000
POP	Запись верхушки стека в ячейку памяти данных с прямой/косвенной адресацией	1/1	1000 1010 IAAA AAAA
PSHD	Запись значения ячейки памяти данных с прямой/косвенной адресацией в стек	1/1	0111 0110 IAAA AAAA
PUSH	Запись ACC в стек	1/1	1011 1110 0011 1100
RET	Возврат из подпрограммы	1/4	1110 1111 0000 0000
RETC	Условный возврат из прерывания	1 /4(УВ) /2(УНВ)	1110 11TP ZLVC ZLVC
ROL	Циклический сдвиг ACC влево	1/1	1011 1110 0000 1100
ROR	Циклический сдвиг ACC вправо	1/1	1011 1110 0000 1101

Продолжение таблицы 17

Мнемоника	Описание	Слов/ Циклов	Опкод
RPT	Повторение следующей инструкции с количеством раз, указанным в операнде с прямой/косвенной адресацией минус 1	1/1	0000 1011 IAAA AAAA
RPT	Повторение следующей инструкции с количеством раз, указанным в операнде с короткой непосредственной адресацией минус 1	1/1	1011 1011 IIII IIII
SACH	Сохранение старшего слова ACC со сдвигом от 0 до 7 с прямой/косвенной адресацией	1/1	1001 1SHF IAAA AAAA
SACL	Сохранение младшего слова ACC со сдвигом от 0 до 7 с прямой/косвенной адресацией	1/1	1001 0SHF IAAA AAAA
SAR	Сохранение указанного AR в ячейку с прямой/ косвенной адресацией	1/1	1000 0ARX IAAA AAAA
SBRK	Вычитание операнда с короткой непосредственной адресацией из текущего AR	1/1	0111 1100 IIII IIII
SETC C	Установка бита C	1/1	1011 1110 0100 1111
SETC CNF	Установка бита CNF	1/1	1011 1110 0100 0101
SETC INTM	Установка бита INTM	1/1	1011 1110 0100 0001
SETC OVM	Установка бита OVM	1/1	1011 1110 0100 0011
SETC SXM	Установка бита SXM	1/1	1011 1110 0100 0111
SETC TC	Установка бита TC	1/1	1011 1110 0100 1011
SETC XF	Установка бита XF	1/1	1011 1110 0100 1101
SFL	Сдвиг ACC влево	1/1	1011 1110 0000 1001
SFR	Сдвиг ACC вправо	1/1	1011 1110 0000 1010
SPAC	Вычитание PREG из ACC	1/1	1011 1110 0000 0101
SPH	Сохранение старшего слова PREG в ячейке с прямой/косвенной адресацией	1/1	1000 1101 IAAA AAAA
SPL	Сохранение старшего слова PREG в ячейке с прямой/косвенной адресацией	1/1	1000 1100 IAAA AAAA
SPLK	Сохранение константы (второе слово инструкции) в ячейке памяти данных, адресуемой прямо/косвенно	2/2	1010 1110 IAAA AAAA +1 слово
SPM	Установка режима сдвига произведения	1/1	1011 1111 0000 00PM
SQRA	Квадрат операнда с прямой/косвенной адресацией и накопление в ACC предыдущего произведения	1/1	0101 0010 IAAA AAAA
SST	Сохранение ST0 в ячейке, адресуемой прямо/косвенно	1/1	1000 1110 IAAA AAAA
SST	Сохранение ST1 в ячейке, адресуемой прямо/косвенно	1/1	1000 1111 IAAA AAAA
SUB	Вычитание из ACC операнда с прямой/косвенной адресацией со сдвигом от 0 до 15	1/1	0011 SHFT IAAA AAAA
SUB	Вычитание из ACC операнда с длинной непосредственной адресацией со сдвигом от 0 до 15	2/2	1011 1111 1010 SHFT +1 слово
SUB	Вычитание из ACC операнда с прямой/косвенной адресацией со сдвигом 16	1/1	0110 0101 IAAA AAAA
SUB	Вычитание из ACC операнда с короткой непосредственной адресацией	1/1	1011 1010 IIII IIII
SUBB	Вычитание из ACC операнда с прямой/косвенной адресацией с заемом	1/1	0110 0100 IAAA AAAA

Окончание таблицы 17

Мнемоника	Описание	Слов/ Циклов	Опкод
SUBC	Условное вычитание операнда с прямой/косвенной адресацией	1/1	0110 0100 IAAA AAAA
SUBS	Вычитание из ACC операнда с прямой/косвенной адресацией с запрещением расширения знака	1/1	0110 0110 IAAA AAAA
SUBT	Вычитание из ACC операнда с прямой/косвенной адресацией со сдвигом, указанным в TREG	1/1	0110 0111 IAAA AAAA
TBLR	Передача слова из ячейки памяти программ, адресуемой вторым словом инструкции, в память данных, адресуемую прямо/косвенно	1/3	1010 0110 IAAA AAAA
TBLW	Передача слова из ячейки памяти данных, адресуемой прямо/косвенно в ячейку памяти программ, адресуемую вторым словом инструкции	1/3	1010 0111 IAAA AAAA
TRAP	Программное прерывание	1/4	1011 1110 0101 0001

10 Поддержка при разработке систем на основе ИС 1867ВЦ10Т

ИС 1867ВЦ10Т поддерживается аппаратными (эмулятор-ВЦ10Т и модуль TMS320LF2407A EVM) и программными средствами, включая средства для генерации кода, реализации разработки алгоритмов и полную интеграцию отладки программного обеспечения и модулей технического обеспечения.

10.1 Средства разработки программного обеспечения

Нижеследующий перечень описывает средства разработки для ИС 1867ВЦ10Т:

- Ассемблер/Редактор связей от фирмы Texas Instrument (TI).
- Оптимизирующий ANSI C компилятор (TI).
- Алгоритмы приложений (TI) и третьих фирм.
- Библиотека стандартных программ (TI).
- Отладчик C/Ассемблер и просмотрщик кода (TI).
- Code Composer, поддерживающий TMS320C/F2xx.

10.2 Средства разработки технического обеспечения

Эмулятор-ВЦ10Т поддерживает мультипроцессорную систему отладки ИС 1867ВЦ10Т, а модуль TMS320LF2407A EVM поддерживают разработку аппаратуры на базе ИС 1867ВЦ10Т.

В таблице 18 приведен полный перечень средств, которые могут быть использованы при разработке приложения на ИС 1867ВЦ10Т.

Таблица 18 – Инструментальные средства для ИС 1867ВЦ10Т

Средство разработки	Платформа	Индекс
Программное обеспечение – Средства для генерации кода		
Assembler/Linker	PC , Windows 95	TMDS3242850-02
C Compiler/Assembler/Linker	PC, Windows 95	TMDS3242855-02

Окончание таблицы 18

Программное обеспечение – Средства для эмуляторной отладки		
LF2407 eZdsp	PC	TMDS3P761119
Code Composer 4.12, Code Generation 7.0	PC	TMDS324012xx
Аппаратное обеспечение – Средства для эмуляторной отладки		
XDS510XL Board (ISA card), w/JTAG cable	PC	TMDS00510
XDS510PP Pod (Parallel Port) w/JTAG cable	PC	TMDS00510PP
Специальные средства поддержки по разработке 1867ВЦ10Т		
Аппаратное обеспечение – Оценочный/Стартовый набор		
TMS320LF2407A EVM	PC, Windows 95, Windows 98	TMDX3P701016

XDS510XL, XDS510PP и TMS320 являются торговыми марками Texas Inctruments.

PC является торговой маркой International Business Machines Corp. Windows является зарегистрированной торговой маркой Microsoft Corporation. eZdsp является торговой маркой Spectrum Digital, Inc.

11 Электрические и временные характеристики ИС 1867ВЦ10Т

11.1 Предельные значения параметров при работе в диапазоне температур окружающей среды (если иные условия не указаны)

Номинальные значения напряжений питания микросхемы: ядра ($U_{\#VCC1}$) – 1,8 В; выходных каскадов ($U_{\#VCC2}$) - 3,3 В.

Допустимые отклонения напряжений питания от номинальных значений: ±10 %.

Диапазон температур окружающей среды, $T_{окр}$ от минус 60 ºС до плюс 85 ºС.

Температура хранения $T_{хр}$ от минус 55 ºС до плюс 150 ºС.

Примечание - Воздействия за пределами оговоренных значений «предельные значения параметров» могут привести к постепенному разрушению микросхемы.

В таблице 19 приведены рекомендуемые условия эксплуатации. Эксплуатация микросхем за пределами указанных значений не предполагается. Незащищенность изделий от повышенных нагрузок на протяжении длительного промежутка времени может привести к нарушению безотказности работы.

Примечание – Все значения напряжения приводятся относительно направления «Земля».

Таблица 19 – Рекомендуемые условия эксплуатации

Наименование параметра режима, единица измерения	Буквенное обозначение	Значение параметра	
		мин.	макс.
Напряжение питания ядра, В	$U_{\#VCC1}$	1,6	2,0
Напряжение питания цифровой части выходных каскадов, В	$U_{\#VCC2}$	3,0	3,6

Окончание таблицы 19

Наименование параметра режима, единица измерения		Буквенное обозначение	Значение параметра	
			мин.	макс.
Входное напряжение низкого уровня, В	BQ1/CLKIN	U_{IL}	–0,3	–0,4
	Все остальные входы		–0,3	–0,6
Входное напряжение ысокого уровня, В	BQ1/CLKIN	U_{IH}	2,4	$U_{\#VCC2}$+0,3
	PORESET#, NMI#, RESET#,TRST#		2,2	$U_{\#VCC2}$+0,3
	Все остальные входы		2,0	$U_{\#VCC2}$+0,3
Выходной ток низкого уровня, мА	RESET#	I_{OL}	–	2
	*		–	2
	Все остальные выходы		–	4
Выходной ток высокого уровня, мА	*	I_{OH}	–2	–
	Все остальные выходы, кроме RESET#		–4	–
Рабочая температура, ºС		$T_{окр}$	-60	85

* Выводы ADCIN0/IOPA0, ADCIN1/IOPA1, ADCIN9/IOPA2, ADCIN8/IOPA3, SCIRXD/IO, SCITXD/IO, XINT2/IO, XINT3/IO, SPISOMI/IO, SPISTE/IO.

11.2 Зависимость тока от выходного напряжения (результаты моделирования)

Условия зависимости тока от выходного напряжения:

- температура 85 ºС;
- напряжение 3,0 В.

Типовые значения выходных токов при высоком и низком уровнях напряжения на выходе приведены в таблице 20.

Микросхема должна быть стойкой к воздействию механических, климатических, биологических факторов и специальных сред со значениями характеристик, соответствующими группе унифицированного исполнения 4У по ГОСТ РВ 20.39.414.1-97 и ОСТ В 11 0998-99 с уточнениями, приведенными в таблице 19.

Электрические параметры микросхемы при приемке и поставке в диапазоне рабочих температур окружающей среды (от минус 60 до плюс 85 °С) приведены в таблице 21.

Предельно допустимые и предельные режимы эксплуатации микросхем приведены в таблице 22.

Таблица 20 – Типовые значения выходных токов при высоком (2,9 В) и низком (0,3 В) уровнях напряжения на выходе

Вывод	Напряжение 2,9 В	Напряжение 0,3 В
*	–2,0 мА	-
Все остальные входы	–4,0 мА	-
RESET#	-	4 мА
*	-	2 мА
Все остальные	-	4,0 мА

_____

* Выводы ADCIN0/IOPA0, ADCIN1/IOPA1, ADCIN9/IOPA2, ADCIN8/IOPA3, SCIRXD/IO,SCITXD/IO, XINT2/IO, XINT3/IO, SPISOMI/IO, SPISTE/IO.

Электрические параметры микросхемы при приемке и поставке в диапазоне рабочих температур окружающей среды (от минус 60 до плюс 85 °С) приведены в таблице 21.

Таблица 21 - Электрические параметры микросхемы при приемке и поставке в диапазоне рабочих температур окружающей среды

Наименование параметра, единица измерения, режим измерения			Буквенное обозначение параметра	Норма параметра		Температура среды, °С
				не менее	не более	
1			2	3	4	5
1 Выходное напряжение низкого уровня, В, $U_{CC1}=U_{CC3}$ = 1,8 В, $U_{CC2}=U_{CC4}$ = 3,0 В	Выходы	I_{OL}, мА	U_{OL}			
	RESET#	2				
	ADCIN0/IOPA0, ADCIN1/IOPA1, ADCIN9/IOPA2, ADCIN8/IOPA3, SCIRxD/IO, SCITxD/IO, XINT2/IO, XINT3/IO, SPISOMI/IO/CAN1RxD, SPISTE/IO/CAN2RxD	2		–	0,4	
	все остальные выходы	4				–60 ± 3 25 ± 10 85 ± 3
2 Выходное напряжение высокого уровня, В, $U_{CC1}=U_{CC3}$ = 1,8 В, $U_{CC2}=U_{CC4}$ = 3,0 В	Выходы	I_{OH}, мА	U_{OH}			
	ADCIN0/IOPA0, ADCIN1/IOPA1, ADCIN9/IOPA2, ADCIN8/IOPA3, SCIRxD/IO, SCITxD/IO, XINT2/IO, XINT3/IO, SPISOMI/IO/CAN1RxD, SPISTE/IO/CAN2RxD	–2		2,4	–	
	все остальные выходы, кроме RESET#, I2CSDA, I2CSCL	–4				

Окончание таблицы 21

1		2	3	4	5
3 Входной ток низкого уровня, мкА, $U_{CC1} = U_{CC3} = 1{,}8$ В, $U_{CC2}=U_{CC4} = 3{,}6$ В, $U_{IL} = 0$ В	вход TRST#	I_{IL}	–10	10	
	входы TMS, TCK, TDI, NMI#, PDPINT#, XINT1#, XINT2/IO, XINT3/IO, BIO#/IOPC3		–500	–10	
	все остальные входы		–10	10	
4 Входной ток высокого уровня, мкА, $U_{CC1} = U_{CC3} = 1{,}8$ В, $U_{CC2} = U_{CC4} = 3{,}6$ В, $U_{IH} = 3{,}6$ В	вход TRST#	I_{IH}	10	500	
	входы TMS, TCK, TDI, NMI#, PDPINT#, XINT1#, XINT2/IO, XINT3/IO, BIO#/IOPC3		–10	10	
	все остальные входы		–10	10	–60 ± 3 25 ± 10 85 ± 3
5 Выходной ток низкого уровня буфера в состоянии «Выключено», мкА, $U_{CC1} = U_{CC3} = 1{,}8$ В, $U_{CC2} = U_{CC4} = 3{,}6$ В, $U_{OZL} = 0$ В		I_{OZL}	–5	5	
6 Выходной ток высокого уровня буфера в состоянии «Выключено», мкА, $U_{CC1} = U_{CC3} = 1{,}8$ В, $U_{CC2} = U_{CC4} = 3{,}6$ В, $U_{OZH} = 3{,}6$ В		I_{OZH}	–5	5	
7 Динамический ток потребления ядра, мА, $U_{CC1} = U_{CC3} = 2{,}0$ В, $U_{CC2} = U_{CC4} = 3{,}6$ В, $f_{CI} = 60$ МГц		I_{OCC1}	–	240	
8 Функциональный контроль $U_{CC1} = U_{CC3} = (1{,}6; 2{,}0)$ В, $U_{CC2} = U_{CC4} = U_{REFH} = (3{,}0; 3{,}6)$ В, $f_{CI} = (1; 60; 120)$ МГц, $U_{REFL} = 0$ В		ФК	–	–	

Примечания

1 Параметры I_{IL}, I_{IH}, I_{OZL}, I_{OZH} при температуре минус 60 ºС не измеряются, а гарантируются нормами при температуре (25 ± 10) ºС.

2 Выводы RESET#, I2CSDA, I2CSCL типа I/O (вход/выход). У этих выводов выходной буфер выполнен по схеме с открытым стоком.

3 Вход TRST# через резистор (типа «pull-down») соединен с шиной земля.

4 Входы TMS, TCK, TDI, NMI#, PDPINT#, XINT1#, XINT2/IO, XINT3/IO, BIO#/IOPC3 через резистор (типа «pull-up») соединены с шиной питания.

Таблица 22

Наименование параметра режима, единица измерения		Буквенное обозначение параметра	Предельно допустимый режим		Предельный режим	
			не менее	не более	не менее	не более
1 Напряжение питания буферов ввода-вывода, В		U_{CC2}	3,0	3,6	–0,3	4,2
2 Напряжение питания ядра и периферийных устройств, В		U_{CC1}	1,6	2,0	–0,3	2,3
3 Напряжение питания PLL, В		U_{CC3}	1,6	2,0	–0,3	2,3
4 Напряжение питания АЦП, В		U_{CC4}	3,0	3,6	–0,3	4,2
5 Входное напряжение низкого уровня, В	BQ1/CLKIN	U_{IL}	–0,3	0,4	–0,5	–
	все остальные входы		–0,3	0,6	–0,5	–
6 Входное напряжение высокого уровня, В	BQ1/CLKIN	U_{IH}	2,4	U_{CC1}+0,3	–	U_{CC1}+0,5
	PORESET#, NMI#, RESET#, TRST#		2,2	U_{CC1}+0,3	–	U_{CC1}+0,5
	все остальные входы		2,0	U_{CC1}+0,3	–	U_{CC1}+0,5
7 Напряжение, подаваемое на выход микросхемы в состоянии «Выключено», В		U_{OZ}	–0,3	U_{CC1}+0,3	–0,5	U_{CC1}+0,5
8 Выходной ток низкого уровня, мА	RESET#	I_{OL}	–	2	–	4
	ADCIN0/IOPA0, ADCIN1/IOPA1, ADCIN9/IOPA2, ADCIN8/IOPA3, SCIRxD/IO, SCITxD/IO, XINT2/IO, XINT3/IO, SPISOMI/IO/CAN1RxD, SPISTE/IO/CAN2RxD		–	2	–	4
	остальные выходы		–	4	–	8
9 Выходной ток высокого уровня, мА	ADCIN0/IOPA0, ADCIN1/IOPA1, ADCIN9/IOPA2, ADCIN8/IOPA3, SCIRxD/IO, SCITxD/IO, XINT2/IO, XINT3/IO, SPISOMI/IO/CAN1RxD, SPISTE/IO/CAN2RxD	I_{OH}	–2	–	–4	–
	остальные выходы, кроме RESET#		–4	–	–8	–
10 Частота следования импульсов тактового сигнала, МГц		f_{CI}	1	120	–	–
11 Емкость нагрузки, пФ		C_L	–	50	–	
12 Потребляемая мощность, Вт		P_{CC}	–	1	–	–

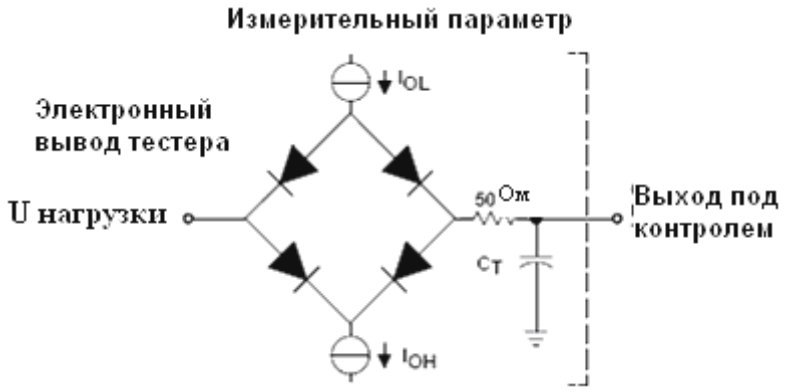
Примечания

1 Время работы в одном из предельных режимов должно быть не более 5 с.

2 Между напряжениями источников питания должны выполняться соотношения $|U_{CC1} - U_{CC3}| \le 0{,}3$ В и $|U_{CC2} - U_{CC4}| \le 0{,}2$ В.

11.3 Информация по измерительным параметрам

Схема тестовой нагрузки показана на рисунке 20.



I_{OL} = 1,5 мА (все выходы);
I_{OH} = 300 мкА (все выходы);
U_{LOAD} = 1,5 В;
C_T = 40 пФ (типовая емкость нагрузки)

Рисунок 20 – Схема тестовой нагрузки

11.4 Уровни перехода сигнала

Некоторые из сигналов используют различную ссылку по электрическому напряжению, смотри рекомендованную таблицу 19 рабочих условий.

TTL-уровни выхода приведены для минимального высокого уровня логической единицы 2,4 В и для максимального низкого уровня логического нуля 0,7 В. На рисунке 21 показан TTL-уровень выводов.

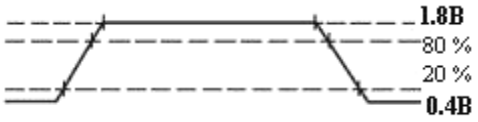


Рисунок 21 – TTL-уровень выводов

Время перехода TTL-совместимого выхода специфицировано ниже:

- Для перехода с высшего значения к низшему значению уровня, на котором вышеупомянутый выход рассматривается как больше не высокий, устанавливается на 80 % или ниже от общего диапазона электрического напряжения, уровень, на котором вышеупомянутый выход рассматривается как низкий, имеет 20 % от общего диапазона электрического напряжения и ниже.

- Для перехода с низшего значения к высшему значению уровня, на котором вышеупомянутый вывод рассматривается как больше не низкий, устанавливается на 20 % от общего диапазона электрического напряжения и выше, уровень, на котором вышеупомянутый вывод рассматривается как высокий, имеет 80 % от общего диапазона электрического напряжения и выше.

11.5 Символы параметров временных диаграмм

Символы параметров временных диаграмм приведены в таблице 23.

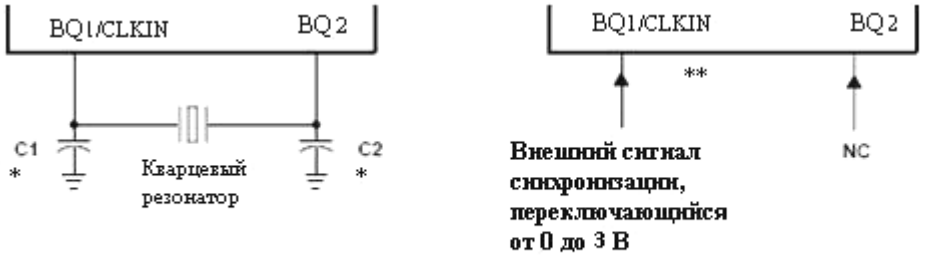
Таблица 23 – Сокращения, используемые на временных диаграммах

Символ	Описание
A	A[15:0]
Cl	BQ1/CLKIN
CO	CLKOUT/IOPC1
D	D[15:0]
INT	NMI, XINT1, XINT2/IO и XINT3/IO
MS	Выводы строба памяти IS, PS
READY	READY
RD	Цикл чтения или RD/WR#
RESET#	RESET# или PORESET#
WR	Цикл записи или WREN#
Подстрочные индексы и их значения	
t_a (access time)	Время доступа
t_c (cycle time period)	Время цикла (периода)
t_d (delay time)	Время задержки
t_f (fall time)	Время падения
t_h (hold time)	Время удержания
t_r (rise time)	Время повышения
t_{su} (setup time)	Время установки
t_t (transition time)	Время перехода
t_v (valid time)	Время достоверного значения
t_w (pulse duration/width)	Длительность импульса (ширина)
H (High)	Высокий
L (Low)	Низкий
B (Valid)	Достоверный
X	Неизвестный, измененный или не имеющий значения
Z	Высокий импеданс

11.6 Основные замечания по временным параметрам

Все выходные сигналы от устройства 1867ВЦ10Т (включая CLKOUT) происходят от внутренней синхронизации так, что все переходы выходов для данной половины цикла происходят с минимальным сдвигом друг относительно друга.

Комбинация сигналов, показанная в следующих временных диаграммах, не обязательно представляет действительные циклы. Для ознакомления с примерами действительных циклов смотреть соответствующее описание цикла в разделе 11 КДФЛ.431299.030ТО. Рекомендованная связь цепи синхронизации с кварцевым резонатором представлена на рисунке 22.



* Значения C1 и C2 выбираются исходя из специфики используемого кварцевого резонатора.

** Используется эта конфигурация в сочетании с OSCBYP выводом.

Рисунок 22 – Рекомендованная связь цепи синхронизации с кварцевым резонатором

11.7 Выбор синхронизации

Варианты подачи тактовых импульсов представлены в таблице 24. В таблице 25 показывается тактирование при подключенном генераторе внешнего сигнала. В таблице 26 указаны характеристики времен переключений в рекомендованных условиях эксплуатации (см. таблицу 19).

Таблица 24 Варианты подачи тактовых импульсов

Параметр	CLKMD[1:0]
Режим внешнего тактирования, деление на 2	Режим внешнего тактирования, деление на 2
Режим внешнего тактирования, умножение на 1	Режим внешнего тактирования, умножение на 1
Режим кварцевого резонатора, деление на 2	Режим кварцевого резонатора, деление на 2
Режим кварцевого резонатора, умножение на 1	Режим кварцевого резонатора, умножение на 1

Таблица 25 – Тактирование при подключенном генераторе внешнего сигнала

Параметр	Описание параметра	Тестовые условия, T_a, °C	Мин.	Макс.	Ед. изм.
f_x	Частота входного импульса, режим деления на два	от –60 до 85	0	120	МГц
	Частота входного импульса, режим деления на единицу	от –60 до 85	0	60	МГц
Примечание – Микросхема разработана как полностью статическое устройство и, следовательно, может работать при длительности входного тактового импульса t_c(CI), стремящейся к бесконечности. Работа схемы описывается на частоте, близкой к 0 Гц.					

Таблица 26 – Характеристики времен переключений в рекомендованных условиях эксплуатации [H = 0,5 t_c(CO)]

Параметр	Описание параметра	Тактовый режим	Мин.	Тип.	Макс.	Ед. изм.
t_c(CP)	Длительность импульса CPUCLK	CLKIN деление на 2	–	$2t_c$(CI)	0[1)]	нс
		CLKIN деление на 1	–	t_c(CI)	–	
t_c(SYS)	Длительность импульса SYSCLK	CPUCLK деление на 2	–	$2t_c$(CP)	0[1)]	нс
		CPUCLK деление на 4 [2)]	–	$4t_c$(CP)	–	
t_c(CO)	Длительность импульса CLKOUT	CLKIN деление на 2	–	$2t_c$(CI)	0[1)]	нс
		CLKIN деление на	–	t_c(CI)	0[1)]	
t_d(CIH-CO)	Время задержки от BQ1/CLKIN «высокий» до CLKOUT «высокий»/«низкий»			11.6		нс
$t_{f(CO)}$	Время спада импульса CLKOUT		–	2	–	нс
$t_{r(CO)}$	Время нарастания импульса CLKOUT		–	2	–	нс
t_w(COL)	Время нахождения CLKOUT в состоянии низкого уровня		H–10	H–6	H–1	нс
t_w(COH)	Время нахождения CLKOUT в состоянии высокого уровня		H+0	H+4	H+8	нс

[1)] Микросхема разработана как полностью статическое устройство и, следовательно, может работать при длительности входного тактового импульса t_c(CI), стремящейся к бесконечности. Работа схемы описывается на частоте, близкой к 0 Гц.

[2)] SYSCLK устанавливается в режим «деление на 4» при всех вариантах сброса.

Временные диаграммы предполагают, что CLKOUT установлен как CPUCLK. CLKOUT инициализируется как CPUCLK сбросом по питанию.

В таблице 27 приведены требования к тактирующим импульсам при работе в рекомендуемых условиях. Временные характеристики представлены в таблицах 28, 29, 30.

Таблица 27 – Требования к тактирующим импульсам при работе в рекомендуемых условиях

Параметр	Описание параметра	Режим входного такта	Мин.	Макс.	Ед. изм.
t_c(CI)	Длительность импульса BQ1/CLKIN	Деление на 2	4	0[1)]	нс
		Деление на 1	8	0[1)]	
t_f(CI)	Время спада фронта BQ1/CLKIN			1	нс
t_r(CI)	Время нарастания фронта BQ1/CLKIN			1	нс

Окончание таблицы 27

Параметр	Описание параметра	Режим входного такта	Мин.	Макс.	Ед. изм.
t_w(CIL)	Длительность пребывания BQ1/CLKIN в состоянии «низкого» уровня, в процентах от t_c(Cl)			50	%
t_w(CIH)	Длительность пребывания BQ1/CLKIN в состоянии «высокого» уровня, в процентах от t_c(CI)			50	%

[1] Микросхема разработана как полностью статическое устройство и, следовательно, может работать при длительности входного тактового импульса t_c(CI), стремящейся к бесконечности. Работа схемы описывается на частоте, близкой к 0 Гц.

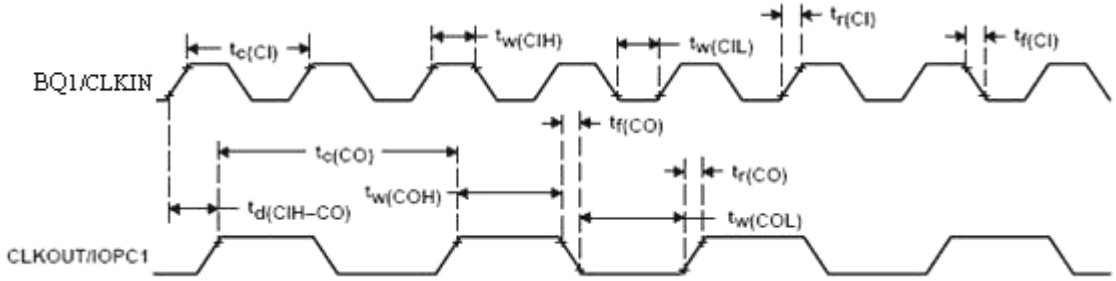


Рисунок 23 – Внешняя синхронизация с делением на два

Кварцевый резонатор запускается подключением OSCBYP# к U_{CC1} и подсоединением через выводы BQ1/CLKIN и BQ2, как показано на рисунке 22. Кварцевый резонатор должен находиться либо в основном, либо в параллельно-резонансном режиме с эффективным последовательным сопротивлением 30 Ом, рассеиваемой мощностью 1 мВт, емкостью нагрузки 20 пФ.

Таблица 28 – Временные характеристики при подключенном кварцевом резонаторе

Параметр	Описание параметра	Внешний кварцевый резонатор, МГц	Мин.	Тип.	Макс.	Ед. изм.
f_x	Частота входного тактового импульса	20				МГц

Таблица 29 – Характеристика времен переключения в рекомендованных условиях эксплуатации (H = 0,5 t_c (CO))

Параметр	Описание параметра	Режим тактирования (CLOCK MODE)	Мин.	Тип.	Макс.	Ед. изм.
t_c(CPU)	Длительность импульса CPUCLK	До захвата частоты, CLKIN деленная на 2	–	2t_c(CI)	0[1)]	нс
		До захвата частоты, CLKIN деленная на 1	–	t_c(CI)	–	
		После захвата внутреннего генератора	–	4	–	

Окончание таблицы 29

Параметр	Описание параметра	Режим тактирования (CLOCK MODE)	Мин.	Тип.	Макс.	Ед. изм.
t_c(SYS)	Длительность импульса SYSCLK	CPUCLK деление на 2	$2t_c$(CPU)		0[1)]	нс
		CPUCLK деление на 4[2)]	$4t_c$(CPU)		–	
t_c(CO)	Длительность импульса CLKOUT		4		0[1)]	нс
t_f(CO)	Время спада фронта CLKOUT		–	1	–	нс
t_r(CO)	Время нарастания фронта CLKOUT		–	1	–	нс
t_w(COL)	Длительность импульса CLKOUT в состоянии «низкого» уровня		H–10	H–6	H–1	нс
t_w(COH)	Длительность импульса CLKOUT в состоянии «высокого» уровня		H+0	H+4	H+8	нс
t_p	Переходное время до установления синхронизации после разрешения PLL	До захвата частоты, CLKIN деление на 2		300		мкс
		До захвата частоты, CLKIN деление на 1				

[1)] Микросхема разработана как полностью статическое устройство и, следовательно, может работать при длительности входного тактового импульса t_c(CI), стремящейся к бесконечности. Работа схемы описывается на частоте, близкой к 0 Гц.

[2)] SYSCLK инициализируется в режиме «деление на четыре» всеми вариантами сбросов микросхемы.

Таблица 30 – Временные характеристики в рекомендованных условиях эксплуатации

Параметр	Описание	Внешний кварц	Мин.	Макс.	Ед. изм.
t_c(CI)	Продолжительность цикла BQ1/CLKIN Cycle time, BQ1/CLKIN Cycle time, BQ1/CLKIN	20 МГц	50	–	нс
t_f(CI)	Время спада фронта, BQ1/CLKIN			1	нс
t_r(CI)	Время нарастания фронта, BQ1/CLKIN			1	нс
t_w(CIL)	Длительность импульса, BQ1/CLKIN «низкий», в процентах от t_c(CI)			60	%
t_w(CIH)	Длительность импульса, BQ1/CLKIN «высокий», в процентах от t_c(CI)			60	%

Окончание таблицы 30

Примечание – Временные диаграммы предполагают, что CLKOUT установлен как CPUCLK. CLKOUT инициализируется как CPUCLK сброс при включении питания.

[1)] Микросхема разработана как полностью статическое устройство и, следовательно, может работать при длительности входного тактового импульса t_c(CI), стремящейся к бесконечности. Работа схемы описывается на частоте, близкой к 0 Гц.

Временные соотношения от CLKIN до CLKOUT для режима генератора PLL изображены на рисунке 24.

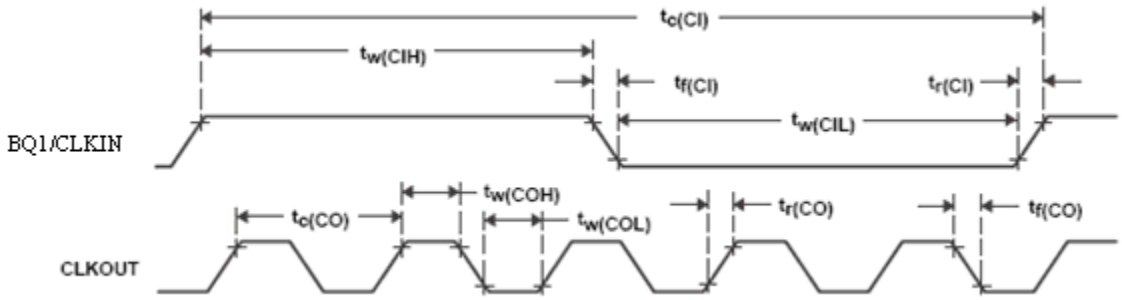


Рисунок 24 – Временные соотношения от CLKIN до CLKOUT для режима генератора PLL, опция умножение на пять с 4 МГц кристаллом

11.8 Временные диаграммы в режимах пониженного энергопотребления

Характеристики времен переключений в рекомендованных условиях эксплуатации представлены в таблице 31, см. также рисунки 25 – 27.

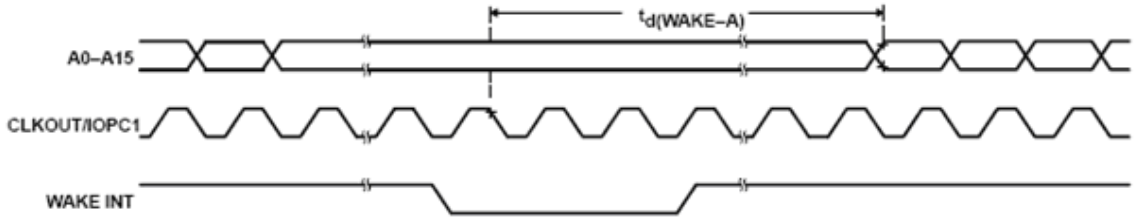


Рисунок 25 – Временные диаграммы входа и выхода из IDLE 1

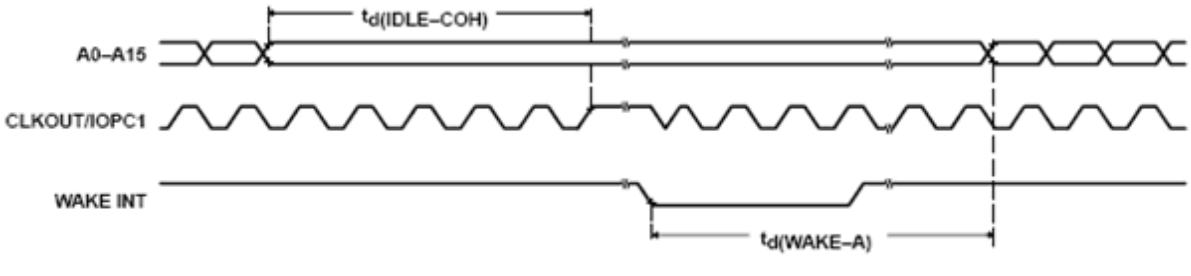


Рисунок 26 – Временные диаграммы входа и выхода из IDLE 2

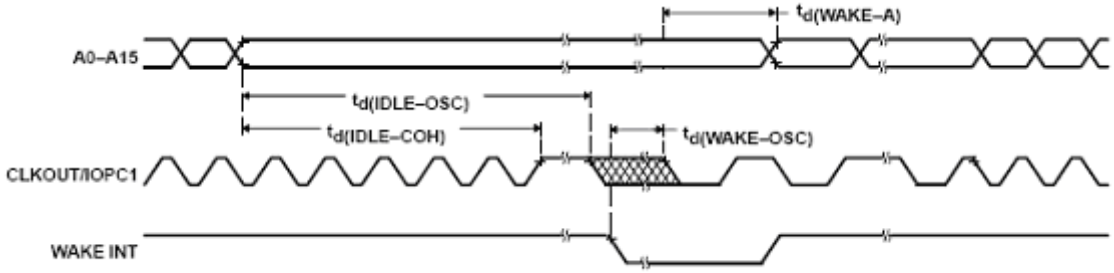


Рисунок 27 – Вход OSC в режим выключения питания и временные характеристики выхода

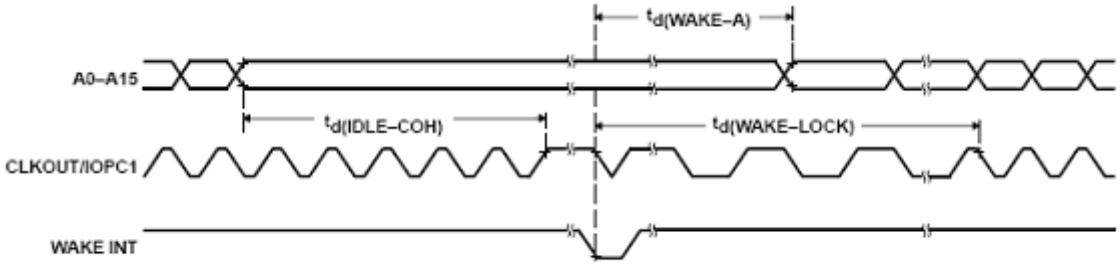


Рисунок 28 - Вход PLL в режим выключения питания и временные характеристики выхода

Таблица 31 – Характеристики времен переключений в рекомендованных условиях эксплуатации

Параметр	Описание параметра	Режимы пониженного потребления	Мин.	Тип.	Макс.	Ед. изм.
t_d(WAKE-A)	Время задержки от отключения CLKOUT до возобновления рабочего режима схемы (см. примечание)	Режимы IDLE 1 и IDLE 2	–	15 × t_c(CO)		нс
		Режим отключения питания PLL или OSC	–	15 × t_c(CI)		
t_d(IDLE-COH)	Время задержки с момента старта IDLE инструкции до установления CLKOUT в «высокий» уровень (см. примечание)	Режим IDLE 2, режим отключения питания PLL, режим отключения питания OSC	–	+500		нс
t_d(WAKE-LOCK)	Время задержки от момента отключения CLKOUT до начала синхронизации PLL (см. примечание)	Режим отключения питания PLL или OSC	–	300		мкс

Окончание таблицы 31

Параметр	Описание параметра	Режимы пониженного потребления	Мин.	Тип.	Макс.	Ед. изм.
t_d(WAKE-OSC)	Время задержки с момента начала прерывания до запуска кварцевого резонатора	Режим отключения питания OSC	–	10		мс
t_d(IDLE-OSC)	Время задержки с момента старта инструкции Idle до отключения питания кварцевого резонатора	Режим отключения питания OSC	–	60		мкс
Примечание – Временные диаграммы предполагают, что CLKOUT установлен как CPCLK. CLKOUT инициализируется CPCLK сбросом при включении питания.						

11.9 Временные характеристики устройств памяти и параллельного интерфейса ввода-вывода при выполнении «чтения»

Характеристики времен переключений в рекомендованных условиях эксплуатации для устройств памяти в процессе «чтения» при 3,3 В представлены в таблице 32, см. также рисунок 29.

Таблица 32 – Характеристики времен переключений в рекомендованных условиях эксплуатации

Параметр	Описание	Мин.	Макс.	Ед. изм.
t_d(CO-A)RD	Время задержки от момента установления низкого уровня на CLKOUT/IOPC1 до достоверного адреса	–	4,5	нс
t_d(CO-SL)RD	Время задержки от момента установления низкого уровня на CLKOUT/IOPC1 до низкого уровня STRB#	–	4	нс
t_d(CO-SH)RD	Время задержки от момента установления низкого уровня на CLKOUT/IOPC1 до высокого уровня STRB#	–	4	нс
t_d(CO-ACTL)RD	Время задержки от момента установления низкого уровня на CLKOUT/IOPC1 до переключения PS#, IS# в низкий уровень	–	4,5	нс
t_d(CO-ACTH)RD	Время задержки от момента установления низкого уровня на CLKOUT/IOPC1 до переключения PS#, IS# в высокий уровень	–	4,5	нс

Требования к временным характеристикам при работе в рекомендованных условиях 3,3 В [H = 0,5 t_c(CO)] для устройств памяти при «чтении» представлены в таблице 33, см. рисунок 30.

Таблица 33 – Требования к временным характеристикам

Параметр	Описание параметра		Мин.	Макс.	Ед. изм.
t_a(A)	Время доступа к чтению данных с начала разрешения адресации	0 состояние ожидания (0 wait state)	–	2H –32	нс
		1 состояние ожидания (1 wait state)	–	4H –32	
t_{su}(DCOL)RD	Время до установления на CLKOUT/IOPC1 низкого уровня напряжения с момента начала чтения		5	–	нс
t_h(COL-D)RD	Время продолжения чтения данных с момента установления на CLKOUT/IOPC1 низкого уровня		2	–	нс

Примечание – На всех временных диаграммах, относящихся к CLKOUT/IOPC1, предполагается, что биты [1:0] CLKSRC установлены таким образом, что CPUCLK работает как выход.

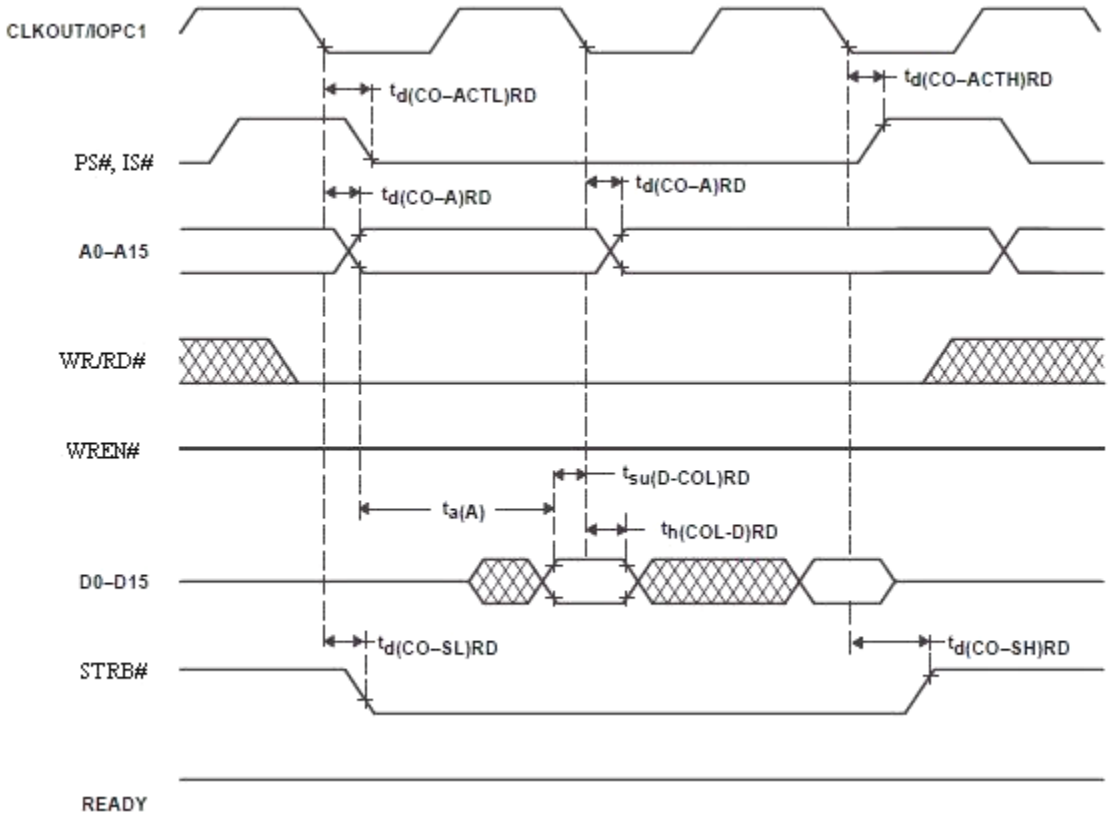


Рисунок 29 – Временные диаграммы операции «чтения» интерфейса памяти

11.10 Временные характеристики устройств памяти и параллельного интерфейса ввода-вывода при выполнении «записи»

Характеристики времен переключений в рекомендованных условиях эксплуатации 3,3 В [H = 0,5t_c(CO)] для устройств памяти в процессе «записи» приведены в таблице 34, диаграммы – на рисунке 30.

На всех временных диаграммах, относящихся к CLKOUT/IOPC1, предполагается, что биты [1:0] CLKSRC установлены таким образом, что CPCLK работает как выход.

Таблица 34 – Характеристики времен переключений

Параметр	Описание параметра	Мин.	Макс.	Ед. изм.
t_d(CO-A)W	Время задержки от установления высокого уровня на CLKOUT/IOPC1 до достоверного адреса	–	4,5	нс
t_d(CO-D)	Время задержки с момента установления низкого уровня на CLKOUT/IOPC1 до управления шиной данных	–	4	нс
t_h(WH-A)	Время сохранения адреса с момента установления высокого уровня на WREN#	H –8	–	нс
t_w(WH)	Время длительности WREN# в состоянии высокого уровня	2H –11	–	нс
t_w(WL)	Время длительности WREN# в состоянии низкого уровня	2H –11	–	
t_d(CO-WL)	Время задержки с момента установления низкого уровня на CLKOUT/IOPC1 до переключения WREN# в «0»	–	4	нс
t_d(CO-WH)	Время задержки с момента установления низкого уровня на CLKOUT/IOPC1 до переключения WREN# в «1»	–	4	нс
t_{su}(D-WH)	Время установки достоверных данных для записи до установления высокого уровня на WREN#	2H –8	–	нс
t_{hz}(WH-D)	Время высокого импеданса с момента переключения WREN# в «1» до установления третьего состояния на шине данных	0	4	нс

Окончание таблицы 34

Параметр	Описание параметра	Мин.	Макс.	Ед. изм.
t_d(CO-SL)W	Время задержки с момента установления низкого уровня на CLKOUT/IOPC1 до переключения STRB в «0»	–	4	нс
t_d(CO-SH)W	Время задержки с момента установления низкого уровня на CLKOUT/IOPC1 до переключения STRB в «1»	–	4	нс
t_d(CO-ACTL)W	Время задержки с момента установления высокого уровня на CLKOUT/IOPC1 до переключения PS#, IS# и STRB# в «0	–	4	нс
t_d(CO-ACTH)W	Время задержки с момента установления высокого уровня на CLKOUT/IOPC1 до переключения PS#, IS# и STRB# в «1»	–	4	нс
t_d(CO-RWL)	Время задержки с момента установления высокого уровня на CLKOUT/IOPC1 до переключения WR/RD# в «0»	–	4	нс
t_d(CO-RWH)	Время задержки с момента установления высокого уровня на CLKOUT/IOPC1 до переключения WR/RD# в «1»	–	4	нс

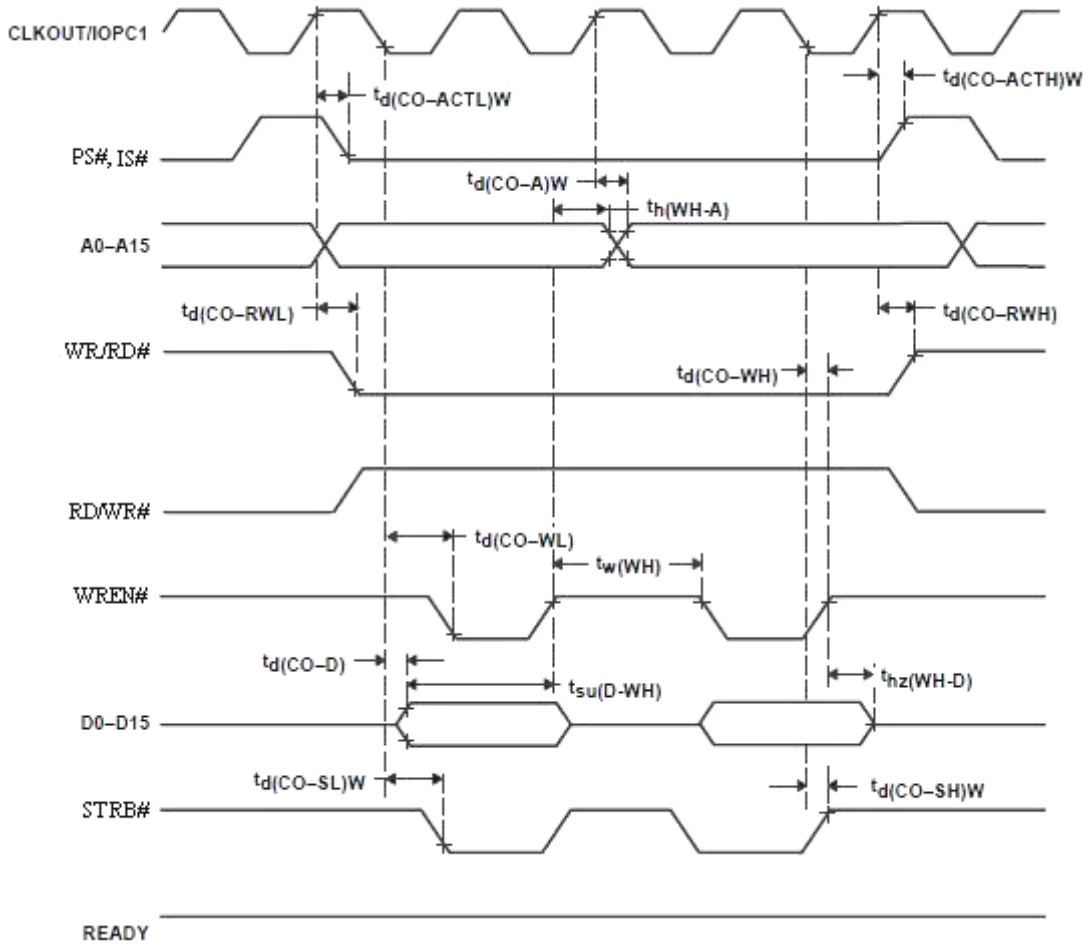


Рисунок 30 – Временные диаграммы операции «записи» интерфейса памяти

11.11 Изменения временных характеристик при различных емкостях нагрузки входов/выходов: результаты моделирования на SPICE

Временная диаграмма нарастания и спада фронта импульса представлена на рисунке 31. Условия: температура от минус 60 до плюс 85 ºС, емкость от 5 до 50 пФ, напряжение 3,3 В.

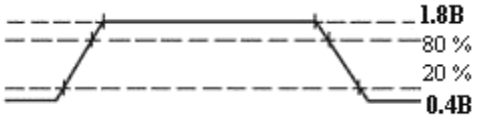


Рисунок 31 – Временная диаграмма нарастания и спада фронта импульса

Изменение параметров импульсов в зависимости от емкости нагрузки (Ucc = 3,3 В, U_{OH} = 2,2 В, U_{OL} = 0,8 В) приведено в таблице 35.

Таблица 35 – Изменение параметров импульсов в зависимости от напряжения питания на выходе CLKOUT/IOPC1

Напряжение питания, В	Нарастающий фронт, нс			Спадающий фронт, нс		
	–60 °C	25 °C	85 °C	–60 °C	25 °C	85 °C
3,0	1,04	1,21	1,32	0,7	0,81	0,9
3,1	0,94	1,1	1,2	6,5	0,74	0,83
3,2	0,84	0,98	1,08	0,59	0,68	0,74
3,3	0,76	0,89	0,99	0,53	0,62	0,69
3,4	0,7	0,8	0,89	0,5	0,57	0,64
3,5	0,67	0,76	0,83	0,48	0,55	0,6
3,6	0,61	0,7	0,78	0,47	0,52	0,56

11.12 Временная синхронизация сигнала READY

Временные диаграммы в рекомендованных условиях эксплуатации [H = 0,5t_c(CO)] приведены в таблице 36 и на рисунке 32.

Таблица 36 – Временные характеристики в рекомендованных условиях эксплуатации*

Параметр	Описание параметра	Мин.	Макс.	Ед. изм.
t_{su}(R-CO)	Время нахождения READY в «0» перед тем, как CLKOUT/IOPC1 перейдет в высокий уровень	4	–	нс
t_h(CO-R)	Время нахождения READY в «0» после того, как CLKOUT/IOPC1 приходит в высокий уровень	0	–	нс
t_v(R)ARD	Разрешенное время готовности READY после разрешения адресов на чтение	–	3H –31	нс
t_v(R)AW	Разрешенное время готовности READY	–	4H –31	нс

*[H = 0,5t_c(CO)] Синхронизация сигнала READY основывается на одном состоянии ожидания. На полной рабочей частоте не допускают ни одного состояния ожидания для сигнала READY.

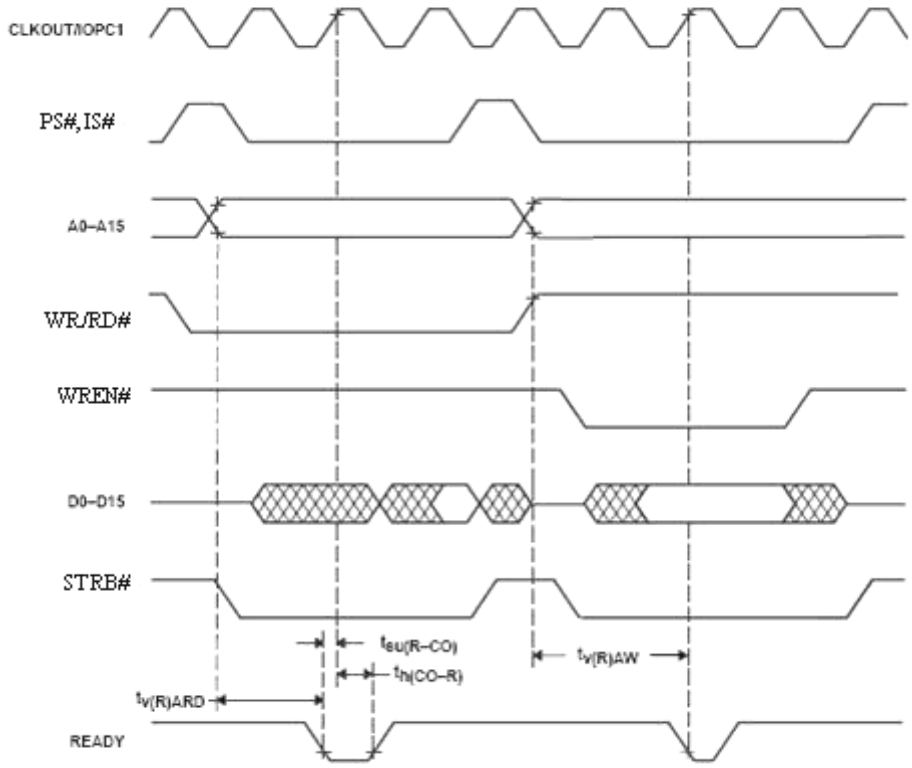


Рисунок 32 – Временная диаграмма сигнала READY

11.13 Временные согласования сигналов RESET# и PORESET#

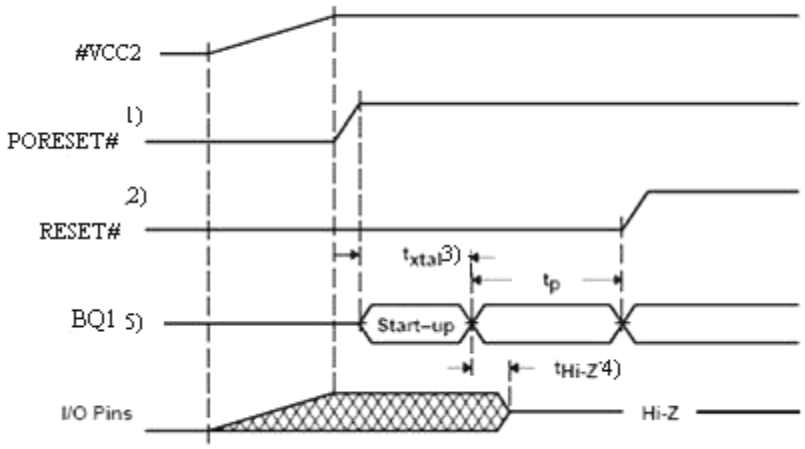
В таблице 37 приведены временные характеристики переключений сигнала RESET# в рекомендованных условиях эксплуатации [H = 0,5t_c(CO)], см. также рисунки 33 – 35.

Таблица 37 – Временные характеристики переключений RESET#

Параметр		Мин.	Макс.	Ед. изм.
t_w(RSL1)	Время нахождения RESET# в низком уровне [1)]	8t_c(SYS)	–	нс
t_d(RS)	Время задержки переключения адреса программ по вектору сброса после перехода RESET# в низкий уровень	4H	–	нс
t_d(EX)	Время задержки срабатывания вектора сброса после перехода RESET# в высокий уровень	32H	–	нс

[1)] Параметр t_w(RSL1) относится к периоду, когда RESET# является выходом. [H = 0,5t_c(CO)]

Временные диаграммы для PORESET# в рекомендованных условиях эксплуатации приведены в таблице 38, см. также на рисунках 33 – 36.



_____

[1)] PORESET# должен изменяться медленно во время включения питания для обеспечения сброса всей синхронизации/PLL регистров к известному состоянию.

[2)] RESET# является двунаправленным выводом и может быть по желанию подтянут к «нулю» через открытый сток или через 2,7 кОм резистор. Если сигнал RESET# остается неуправляемым, то используется подтянутый вверх резистор 20 кОм.

[3)] Время включения однокристального генератора зависит от параметров кристалла, обратной связи конденсатора и схемы расположения его на плате. Типичное время включения около 10 мс.

[4)] После того как сигнал PORESET# станет высоким уровнем и включится генератор, это займет несколько фронтов синхронизирующего импульса (обычно 4–8 циклов генератора) для I/O и для обеспечения состояния высокого напряжения.

[5)] CLKOUT использует внутрикристальный генератор.

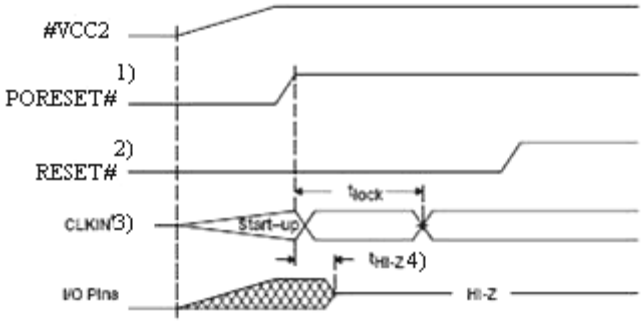
Рисунок 33 – Случай с кварцевым резонаторм

Таблица 38 – Временные характеристики для сигнала RESET#

Параметр	Описание	Мин.	Макс.	Ед. изм.
t_w(RSL)	Длительность сигналов RESET# или PORESET# в низком уровне [1)]	100	–	нс

_____

[1)] Параметр t_w(RSL) относится к периоду, когда RESET# является входом.



---

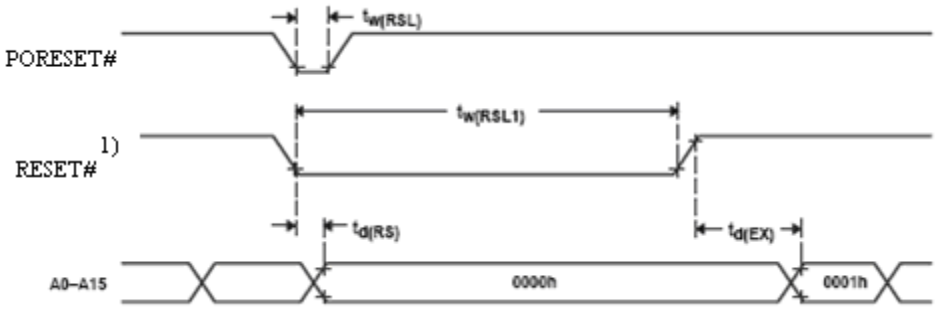
1) PORESET# должен изменяться медленно во время включения пита ния для обеспечения сброса всей синхронизации к известному состоянию.

2) RESET# является двунаправленным выводом и может быть по желанию подтянут к «нулю» через открытый сток или через 2,7 кОм резистор. Если сигнал RESET# остается неуправляемым, то используется подтянутый вверх резистор 20 кОм.

3) CLKOUT использует внешний генератор.

4) Если используется внешнее тактирование и после того как сигнал PORESET# станет высоким уровнем, это займет несколько фронтов управляюще-го синхронизирующего импульса (обычно 4–8 циклов генератора) для I/O и для обеспечения состояния высокого напряжения.

Рисунок 34 – Случай с внешним генератором



---

1) RESET# изменяется медленно через любой сброс ИС, который включает следующие сигналы:

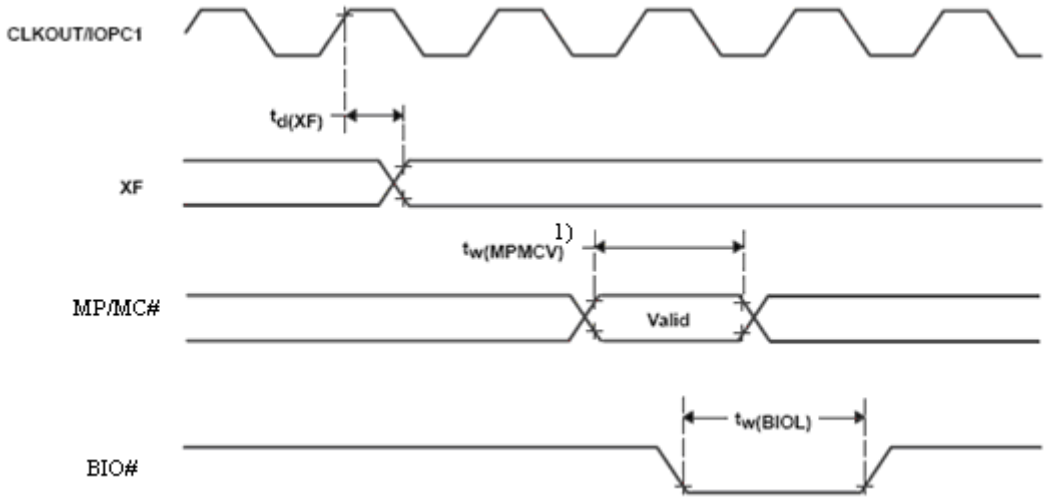
- PORESET#;
- RESET#;
- доступ к нелегальному адресу;
- выполнение сброса программой;
- сброс от сторожевого таймера.

Рисунок 35 – Временные диаграммы включения сброса

Таблица 39 – Временные характеристики для XF, BIO# и MP/MC#

Параметр	Описание параметра	Мин.	Макс.	Ед. изм.
t_d(XF)	Время задержки переключения XF (высокий/низкий) после перехода CLKOUT в «1»	–	4	нс
t_w(BIOL)	Время нахождения BIO# в низком уровне	2H + 16	–	нс
t_w(MPMCV)	Время нахождения MP/MC#*	2H + 24	–	нс

*Указано минимальное время пребывания вывода MP/MC# в стабильном состоянии, необходимом для установления связи с внутренними логическими схемами. Однако для правильного функционирования ИС необходимо поддерживать заданный уровень на протяжении всего цикла доступа (или доступов) для внутрикристальной или внекристальной памяти. [H = 0,5t_c(CO)].



1) Указано минимальное время пребывания вывода MP/MC# в стабильном состоянии, необходимом для установления связи с внутренними логическими устройствами. Однако для правильного функционирования пользователь должен поддерживать заданный уровень на протяжении всего цикла доступа (или доступов) для внутрикристальной или внекристальной памяти

Рисунок 36 – Временные диаграммы для сигналов XF, BIO# и MP/MC#

11.14 Интерфейс временной диаграммы менеджера событий

11.14.1 Временные диаграммы для выводов PWM/CMP

Вывод PWM подсоединен к выводам PWM1/CMP1, PWM2/CMP2, PWM3/CMP3, PWM4/CMP4, PWM5/CMP5, PWM6/CMP6, T1PWM/T1CMP, T2PWM/T2CMP, T3PWM/T3CMP, PWM7/CMP7, PWM8/CMP8 и PWM9/CMP9.

Временные характеристики переключений в рекомендованных условиях эксплуатации для вывода PWM представлены в таблице 40. На рисунке 37 представлена временная диаграмма для вывода PWM.

Таблица 40 – Временные характеристики переключений [H = 0,5t_c(CO)]

Параметр	Описание	Мин.	Макс.	Ед. изм.
t_d(PWM)CO	Время задержки переключения выхода PWM после прихода «1» на CLKOUT	–	4	нс
t_w(TMRDIR)	Время нахождения TMRDIR в низком/высоком уровне	4H + 12	–	нс
t_w(TMRCLKL)	Время нахождения TMRCLK в низком уровне в процентном отношении от длительности	40	60	%
t_w(TMRCLKH)	Время нахождения TMRCLK в высоком уровне в процентном отношении от длительности	40	60	%
t_c(TMRCLK)	Длительность импульса TMRCLK	4 t_c(CPCLK)	–	нс

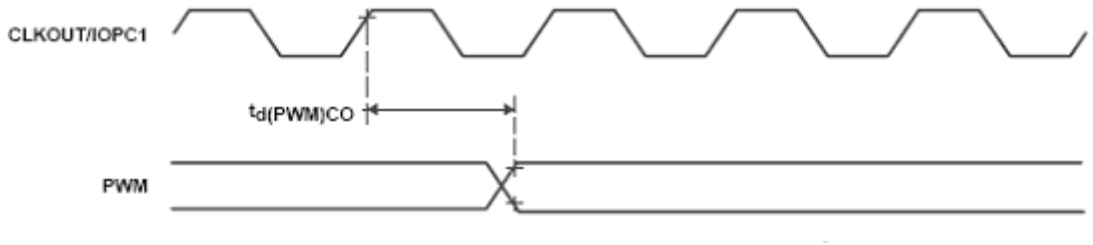


Рисунок 37 – Временные диаграммы для выхода PWM и устройства сравнения

Временные диаграммы для PWM в рекомендованных условиях эксплуатации изображены на рисунках 38 и 39.

Рисунок 38 – Временная диаграмма входа внешнего тактирующего сигнала GP таймеров

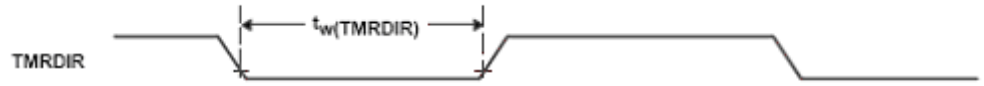


Рисунок 39 – Временная диаграммы для входа внешнего выбора направления GP таймеров

11.14.2 Временные диаграммы для режимов Захват и QEP

Вывод CAP подсоединен к выводам CAP1/QEP1/IOPC4, CAP2/QEP2/IOPC5, CAP3/IOPC6 и CAP4/IOPC7. Временные диаграммы для вывода CAP в рекомендованных условиях эксплуатации [H = 0,5tc(CO)], см. рисунок 40.

Таблица 41 – Описание параметра t_w(CAP) [H = 0,5t_c(CO)]

Параметр	Описание параметра	Мин.	Ед. изм.
t_w(CAP)	Время нахождения входа CAP в низком/высоком уровне (Pulse duration, CAP input low/high)	4H + 12	нс

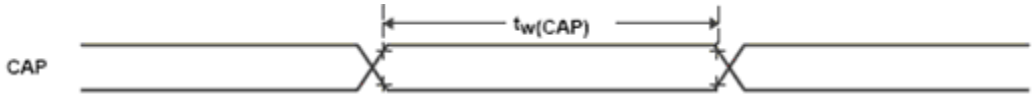


Рисунок 40 – Временная диаграмма для входов режимов Захват и QEP

11.14.3 Временные согласования прерываний

Вывод PWM подсоединен к выводам PWM1/CMP1, PWM2/CMP2, PWM3/CMP3, PWM4/CMP4, PWM5/CMP5, PWM6/CMP6, T1PWM/T1CMP, T2PWM/T2CMP, T3PWM/T3CMP, PWM7/CMP7, PWM8/CMP8 и PWM9/CMP9.

Вывод INT подсоединен к выводам NMI#, XINT1#, XINT2/IO и XINT3/IO.

Вывод PDP подсоединен к выводу PDPINT#.

Временные диаграммы для режимов прерываний в рекомендованных условиях эксплуатации [H = 0,5tc(CO)] изображены на рисунках 41, 42. В таблице 42 описаны параметры временных характеристик переключений.

Таблица 42 – Характеристики времен переключений [H = 0,5tc(CO)]

Параметр	Описание	Мин.	Макс.	Ед. изм.
t_d(PWM)PDP	Время задержки с момента установления низкого уровня на выводе PDPINT# до переключения вывода PWM в третье состояние	0	4	нс

Окончание таблицы 42

Параметр	Описание	Мин.	Макс.	Ед. изм.
t_w(INT)	Время нахождения входа INT в низком/высоком уровне	t_c(SYS) + 12		нс
t_w(PDP)	Время нахождения входа PDPINT# в низком уровне	2H + 18		нс
t_d(INT)	Время задержки с момента переключения INT (низкий/высокий) до инициализации вектора прерывания	$2t_c$(SYS) + $4t_c$(CP)		нс

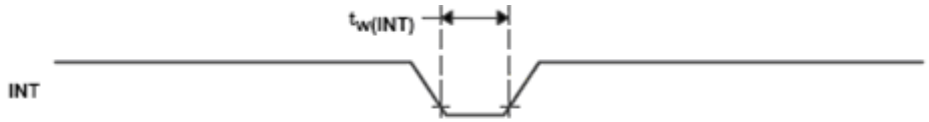


Рисунок 41 – Временная диаграмма внешних сигналов прерывания

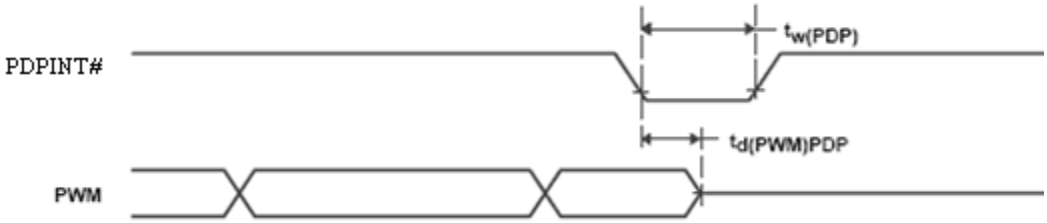


Рисунок 42 – Временная диаграмма прерывания от схемы защиты по питанию

11.14.4 Временные характеристики для входов-выходов общего назначения

Вывод GPO подключен к многофункциональным выходам IOPA0 - IOPA3, IOPB0 - IOPB7, IOPC0 - IOPC7, XINT2/IO и XINT3/IO.

Описание параметров, характеризующих время переключения, приведено в таблице 43. Характеристики времен переключений в рекомендованных условиях эксплуатации для выходов общего назначения изображены на рисунке 43. Временные диаграммы для входов общего назначения в рекомендованных условиях изображены на рисунке 44.

Таблица 43 – Характеристики времен переключений

Параметр	Описание параметра		Мин.	Макс.	Ед. изм.
t_d(GPO)CO	Время задержки переключения GPO (низкий/высокий) после прихода низкого уровня на CLKOUT	XINT2/IO, XINT3/IO, IOPB6, IOPB7 и IOPC0	–	4	нс
		Все остальные GP выходы	–	4	
t_w(GPI)	Время нахождения GP в высоком/низком уровне		t_c(SYS) + 12	–	нс

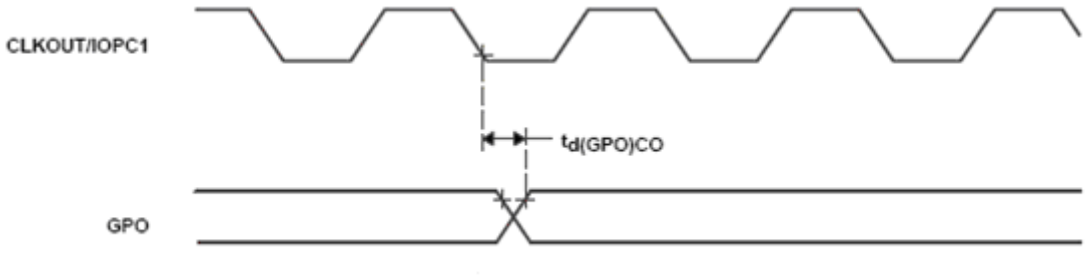


Рисунок 43 – Временная диаграмма для выходов общего назначения

Рисунок 44 – Временная диаграмма для входов общего назначения

11.14.5 Временные характеристики для входов/выходов последовательного интерфейса связи SCI

Временная диаграмма последовательного интерфейса связи SCI изображена на рисунке 45. В таблице 44 описаны временные параметры диаграммы SCI.

Таблица 44 – Временные характеристики для последовательного интерфейса связи SCI

Параметр	Описание	(BRR + 1) четные и BRR = 0		(BRR + 1) нечетные и BRR ≠ 0		Ед. изм.
		Мин.	Макс.	Мин.	Макс.	
t_c(SCC)	Длительность импульса SCICLK	$16t_c$	$65536t_c$	$24t_c$	$65535t_c$	нс
t_v(TXD)	Разрешенное время доступа к данным SCITXD	t_c(SCC)-70	t_c(SCC)+70	t_c(SCC)-70	t_c(SCC)+70	нс
t_v(RXD)	Разрешенное время доступа к данным SCIRXD	$16t_c$	–	$24t_c$	–	нс
Примечание – t_c – длительность импульса тактового сигнала. t_c = 1/SYSCLK.						

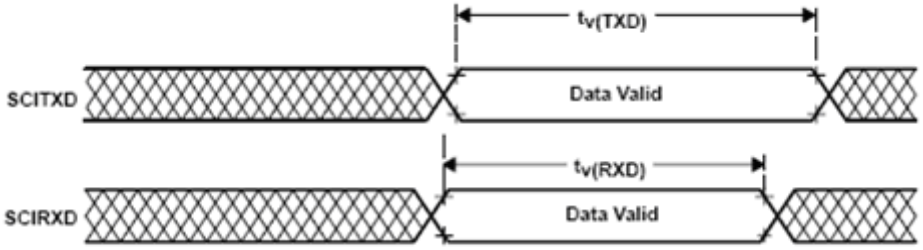


Рисунок 45 – Временные диаграммы для SCI

11.14. 6 Временные параметры в режиме SPI MASTER

Временные параметры режима SPI MASTER (ведущий) представлены в таблицах 45 и 46.

Временные параметры внешних сигналов в режиме работы SPI MASTER (фаза синхроимпульса равна нулю) изображены на рисунке 46.

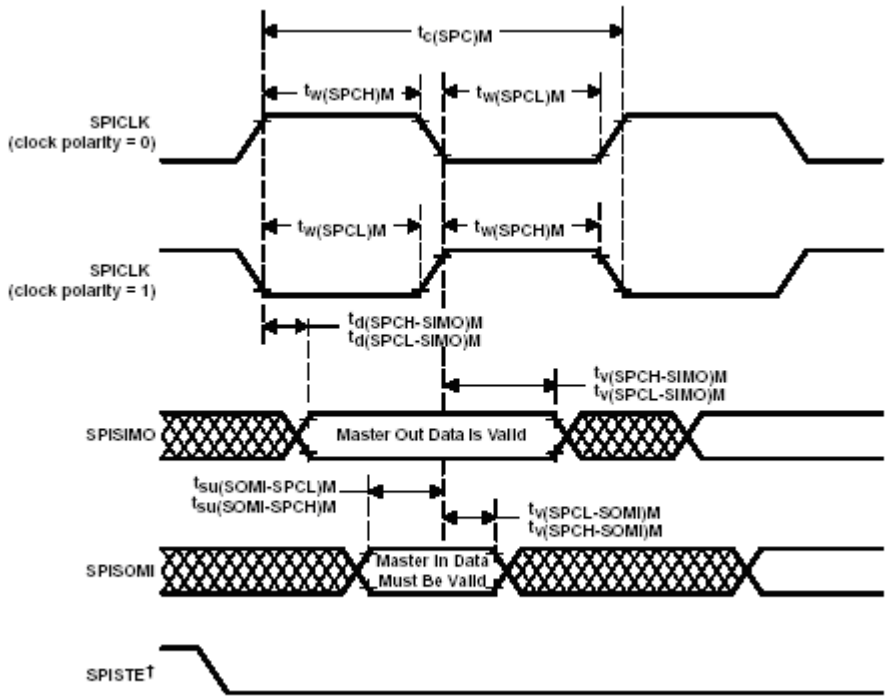


Рисунок 46 – Временные диаграммы режима SPI MASTER (ведущий), фаза тактирующего сигнала равна нулю

Примечание – Сигнал SPISTE должен быть активным перед началом коммуникационного потока и оставаться активным до полного завершения коммуникационного потока.

Таблица 45 – Временные параметры внешних сигналов в режиме работы SPI MASTER (фаза синхроимпульса равна 0)

Параметр	Описание параметра	(SPIBRR + 1) четные или SPIBRR = 0, или 2		(SPIBRR + 1) нечетные и SPIBRR > 3		Ед. изм.
		Мин.	Макс.	Мин.	Макс.	
t_c(SPC)M	Длительность импульса SPICLK	$4t_c$ *	$128t_c$ *	$5t_c$ *	$127t_c$ *	нс
t_w(SPCH)M**	Время нахождения SPICLK в высоком уровне (полярность тактового сигнала = 0)	$0,5t_c$(SPC)M–70	$0,5t_c$(SPC)M	$0,5t_c$(SPC)M–$0,5t_c$ –70	$0,5t_c$(SPC)M –$0,5t_c$	нс
t_w(SPCL)M**	Время нахождения SPICLK в низком уровне (полярность тактового сигнала = 1)	$0,5t_c$(SPC)M–70	$0,5t_c$(SPC)M	$0,5t_c$(SPC)M–$0,5t_c$ –70	$0,5t_c$(SPC)M –$0,5t_c$	нс
t_w(SPCL)M**	Время нахождения SPICLK в низком уровне (полярность тактового сигнала = 0)	$0,t_c$(SPC)M–70	$0,5t_c$(SPC)M	$0,5t_c$(SPC)M–$0,5t_c$ –70	$0,5t_c$(SPC)M + $0,5t_c$	нс
t_w(SPCH)M**	Время нахождения SPICLK в высоком уровне (полярность тактового сигнала = 1)	$0,5t_c$(SPC)M–70	$0,5t_c$(SPC)M	$0,5t_c$(SPC)M–$0,5t_c$ –70	$0,5t_c$(SPC)M + $0,5t_c$	нс
t_d(SPCH-SIMO)M**	Время задержки с момента прихода высокого уровня на SPICLK (полярность тактового сигнала = 0) до разрешения	–10	10	–10	10	нс
t_d(SPCL-SIMO)M**	Время задержки с момента прихода низкого уровня на SPICLK (полярность тактового сигнала = 1) до разрешения	–10	10	–10	10	нс
t_v(SPCL-SIMO)M**	Разрешенное время доступа SPISIMO к данным после прихода на SPICLK низкого уровня (полярность тактового сигнала = 0)	$0,5t_c$(SPC)M–70	–	$0,5t_c$(SPC)M+$0,5t_c$ –70	–	нс
t_v(SPCH-SIMO)M**	Разрешенное время доступа SPISIMO к данным после прихода на SPICLK высокого уровня (полярность тактового сигнала = 1)	$0,t_c$(SPC)M–70	–	$0,t_c$(SPC)M+$0,5t_c$ –70	–	нс
t_{su}(SOMI-SPCL)M**	Время установления постоянного уровня на SPISOMI до прихода низкого уровня на SPICLK (полярность тактового сигнала = 0)	0	–	0	–	нс
t_{su}(SOMI-SPCH)M**	Время установления постоянного уровня на SPISOMI до прихода высокого уровня на SPICLK (полярность тактового сигнала = 1)	0	–	0	–	нс
t_v(SPCL-SOMI)M**	Время разрешения доступа SPISOMI к данным после прихода на SPICLK низкого уровня (полярность тактового сигнала = 0)	$0,25t_c$(SPC)M–70	–	$0,5t_c$(SPC)M–$0,5t_c$ –70	–	нс
t_v(SPCH-SOMI)M**	Время разрешения доступа SPISOMI к данным после прихода на SPICLK высокого уровня (полярность тактового сигнала = 1)	$0,25t_c$(SPC)M–70	–	$0,5t_c$(SPC)M–$0,5t_c$ –70	–	нс

_____

* t_c – длительность синхроимпульса, равна 1/SYSCLK = t_c(SYS).

** За полярность активного фронта тактового сигнала SPICLK отвечает бит полярности (SPICCR.6).

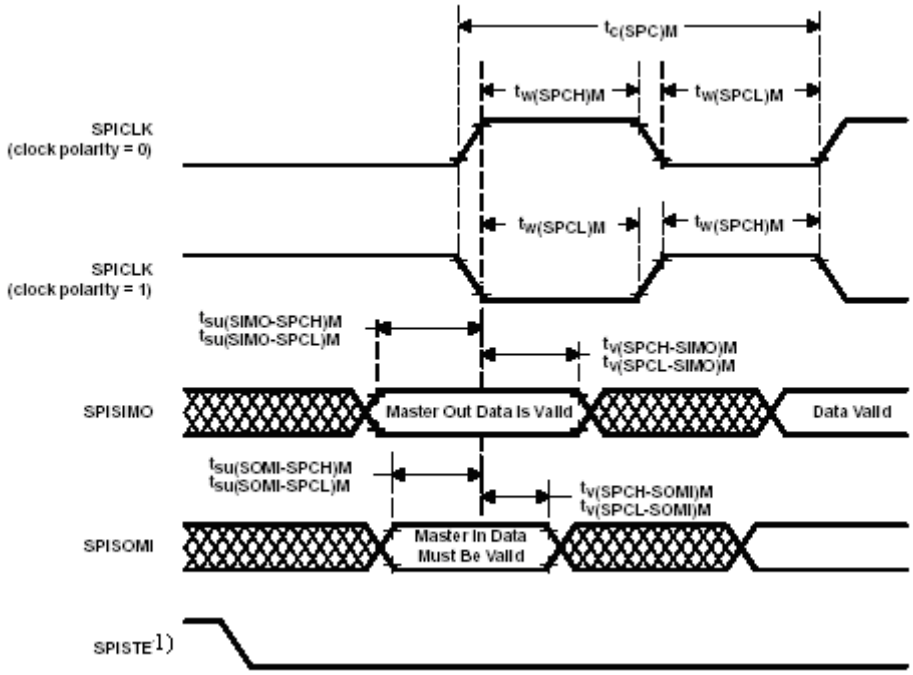
Таблица 46 – Временные параметры внешних сигналов в режиме работы SPI MASTER (фаза синхроимпульса равна 1)

Параметр	Описание параметра	(SPIBRR + 1) четные или SPIBRR = 0 или 2		(SPIBRR + 1) нечетные и SPIBRR > 3		Ед. изм.
		Мин.	Макс.	Мин.	Макс.	
t_c(SPC)M	Длительность импульса SPICLK	$4t_c$ *	$128t_c$ *	$5t_c$ *	$127t_c$ *	нс
t_w(SPCH)M**	Время нахождения SPICLK в высоком уровне (полярность тактового сигнала = 0)	0,5t_c(SPC)M–70	0,5t_c(SPC)M	0,5t_c(SPC)M–0,5t_c –70	0,5t_c(SPC)M –0,5t_c	нс
t_w(SPCL)M**	Время нахождения SPICLK в низком уровне (полярность тактового сигнала = 1)	0,5t_c(SPC)M–70	0,5t_c(SPC)M	0,5t_c(SPC)M–0,5t_c –70	0,5t_c(SPC)M –0,5t_c	нс
t_w(SPCL)M**	Время нахождения SPICLK в низком уровне (полярность тактового сигнала = 0)	0,5t_c(SPC)M–70	0,5t_c(SPC)M	0,5t_c(SPC)M–0,5t_c –70	0,5t_c(SPC)M –0,5t_c	нс
t_w(SPCH)M**	Время нахождения SPICLK в высоком уровне (полярность тактового сигнала = 1)	0,5t_c(SPC)M–70	0,5t_c(SPC)M	0,5t_c(SPC)M–0,5t_c –70	0,5t_c(SPC)M –0,5t_c	нс
t_{su}(SOMI-SPCL)M**	Время установления постоянного уровня на SPISIMO до прихода высокого уровня на SPICLK (полярность тактового сигнала = 0)	0,5t_c(SPC)M–70	–	0,5t_c(SPC)M–70	–	нс
t_{su}(SOMI-SPCL)M**	Время установления постоянного уровня на SPISIMO до прихода низкого уровня на SPICLK (полярность тактового сигнала = 1)	0,5t_c(SPC)M–70	–	0,5t_c(SPC)M–70	–	нс
t_v(SPCL-SIMO)M**	Разрешенное время доступа SPISIMO к данным после прихода на SPICLK высокого уровня (полярность тактового сигнала = 0)	0,5t_c(SPC)M–70	–	0,5t_c(SPC)M+0,5t_c –70	–	нс
t_v(SPCH-SIMO)M**	Разрешенное время доступа SPISIMO к данным после прихода на SPICLK низкого уровня (полярность тактового сигнала = 1)	0,5t_c(SPC)M–70	–	0,5t_c(SPC)M+0,5t_c –70	–	нс
t_{su}(SOMI-SPCL)M**	Время установления постоянного уровня на SPISOMI до прихода высокого уровня на SPICLK (полярность тактового сигнала = 0)	0	–	0	–	нс
t_{su}(SOMI-SPCH)M**	Время установления постоянного уровня на SPISOMI до прихода низкого уровня на SPICLK (полярность тактового сигнала = 1)	0	–	0	–	нс
t_v(SPCL-SOMI)M**	Время разрешения доступа SPISOMI к данным после прихода на SPICLK высокого уровня (полярность тактового сигнала = 0)	0,25t_c(SPC)M–70	–	0,5t_c(SPC)M–70	–	нс
t_v(SPCH-SOMI)M**	Время разрешения доступа SPISOMI к данным после прихода на SPICLK низкого уровня (полярность тактового сигнала = 1)	0,25t_c(SPC)M–70	–	0,5t_c(SPC)M–70	–	нс

_____

* t_c – длительность синхроимпульса, равна 1/SYSCLK = t_c(SYS).

** За полярность активного фронта тактового сигнала SPICLK отвечает бит полярности (SPICCR.6).



Условные обозначения:
Clock polarity=0 – полярность тактового сигнала равна нулю.
Clock polarity=1 – полярность тактового сигнала равна единице.

[1)] Сигнал SPISTE должен быть активным перед началом коммуникационного потока и оставаться активным до полного завершения коммуникационного потока

Рисунок 47 – Временная диаграмма режима SPI MASTER.
Фаза синхроимпульса равна 1

11.14.7 Временные параметры в режиме SPI SLAVE

Временные параметры режима SPI SLAVE представлены в таблицах 47, 48, на рисунках 48, 49.

Таблица 47 – Временные параметры внешних сигналов в режиме работы SPI SLAVE (фаза синхроимпульса равна нулю)

Параметр	Описание	Мин.	Макс.	Ед. изм.
t_c(SPC)S	Длительность импульса SPICLK	8t_c*	–	нс
t_w(SPCH)S**	Время нахождения SPICLK в высоком уровне (полярность тактового сигнала = 0)	0,5t_c(SPC)S–70	0,5t_c(SPC)S	нс
t_w(SPCL)S**	Время нахождения SPICLK в низком уровне (полярность тактового сигнала = 1)	0,5t_c(SPC)S–70	0,5t_c(SPC)S	

Продолжение таблицы 47

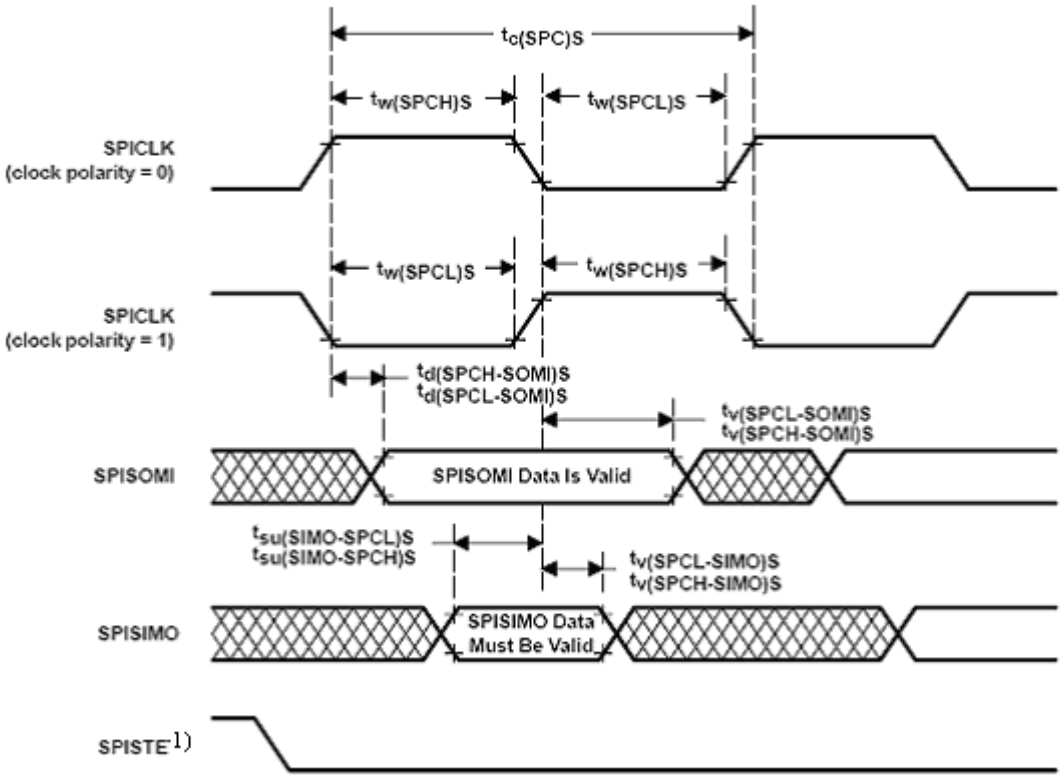
Параметр	Описание	Мин.	Макс.	Ед. изм.
t_w(SPCL)S**	Время нахождения SPICLK в низком уровне (полярность тактового сигнала = 0)	0,5t_c(SPC)S–70	0,5t_c(SPC)S	нс
t_w(SPCH)S**	Время нахождения SPICLK в высоком уровне (полярность тактового сигнала = 1)	0,5t_c(SPC)S–70	0,5t_c(SPC)S	
t_d(SPCH-SOMI)S**	Время задержки с момента прихода высокого уровня на SPICLK (полярность тактового сигнала = 0) до разрешения SPISOMI	0,375t_c(SPC)S–70	–	нс
t_d(SPCL-SOMI)S**	Время задержки с момента прихода низкого уровня на SPICLK (полярность тактового сигнала = 1) до разрешения SPISOMI	0,375t_c(SPC)S–70	–	
t_v(SPCL-SOMI)S**	Разрешенное время доступа SPISOMI к данным после прихода на SPICLK низкого уровня (полярность тактового сигнала = 0)	0,75t_c(SPC)S	–	нс
Ot_v(SPCH-SOMI)S**	Разрешенное время доступа SPISOMI к данным после прихода на SPICLK высокого уровня (полярность тактового сигнала = 1)	0,75t_c(SPC)S	–	нс
t_{su}(SIMO-SPCL)S**	Время установления постоянного уровня на SPISIMO до прихода низкого уровня на SPICLK (полярность тактового сигнала = 0)	0	–	нс
t_{su}(SIMO-SPCH)S**	Время установления постоянного уровня на SPISIMO до прихода высокого уровня на SPICLK (полярность тактового сигнала = 1	0	–	нс

Окончание таблицы 47

Параметр	Описание	Мин.	Макс.	Ед. изм.
t_v(SPCL-SIMO)S**	Время разрешения доступа SPISIMO к данным после прихода на SPICLK низкого уровня (полярность тактового сигнала = 0)	0,5t_c(SPC)S	–	нс
t_v(SPCH-SIMO)S**	Время разрешения доступа SPISIMO к данным после прихода на SPICLK высокого уровня (полярность тактового сигнала = 1)	0,5t_c(SPC)S	–	нс

* t_c – длительность синхроимпульса равна 1/SYSCLK = t_c(SYS).

** За полярность активного фронта тактового сигнала SPICLK отвечает бит полярности (SPICCR.6).



Условные обозначения:

Clock polarity=0 – полярность тактового сигнала равна нулю.

Clock polarity=1 – полярность тактового сигнала равна единице

[1)] Сигнал SPISTE должен быть активным до старта коммуникационного потока SPI и сигнал SPISTE должен оставаться активным до полного завершения коммуникационного потока SPI.

Рисунок 48 – Временные внешние параметры в режиме SPI SLAVE (фаза синхроимпульса равна нулю)

Временные параметры внешних сигналов в режиме работы SPI SLAVE (фаза синхроимпульса равна единице) представлены в таблице 48 и на рисунке 49.

Таблица 48 – Временные параметры внешних сигналов в режиме работы SPI SLAVE (фаза синхроимпульса равна единице)

Параметр	Описание	Мин.	Макс.	Ед. изм.
t_c(SPC)S	Длительность импульса SPI-CLK	8tc*	–	нс
t_w(SPCH)S**	Время нахождения SPICLK в высоком уровне (полярность тактового сигнала = 0)	0,5t_c(SPC)S–70	0,5t_c(SPC)S	нс
t_w(SPCL)S**	Время нахождения SPICLK в низком уровне (полярность тактового сигнала = 1)	0,5t_c(SPC)S–70	0,5t_c(SPC)S	
t_w(SPCL)S**	Время нахождения SPICLK в низком уровне (полярность тактового сигнала = 0)	0,5t_c(SPC)S–70	0,5t_c(SPC)S	нс
t_w(SPCH)S**	Время нахождения SPICLK в высоком уровне (полярность тактового сигнала = 1)	0,5t_c(SPC)S–70	0,5t_c(SPC)S	
t_{su}(SOMI-SPCH)S**	Время установления постоянного уровня на SPISIMO до прихода высокого уровня на SPICLK (полярность тактового сигнала = 0)	0,125t_c(SPC)S	–	нс
t_{su}(SOMI-SPCL)S**	Время установления постоянного уровня на SPISIMO до прихода низкого уровня на SPICLK (полярность тактового сигнала = 1)	0,125t_c(SPC)S	–	
t_v(SPCH-SOMI)S**	Разрешенное время доступа SPISOMI к данным после прихода на SPICLK высокого уровня (полярность тактового сигнала = 0)	0,75t_c(SPC)S	–	нс
t_v(SPCL-SOMI)S**	Разрешенное время доступа SPISOMI к данным после прихода на SPICLK низкого уровня (полярность тактового сигнала = 1)	0,75t_c(SPC)S	–	
t_{su}(SIMO-SPCH)S**	Время установления постоянного уровня на SPISIMO до прихода высокого уровня на SPICLK (полярность тактового сигнала = 0)	0	–	нс

Окончание таблицы 48

Параметр	Описание	Мин.	Макс.	Ед. изм.
t_{su}(SIMO-SPCL)S**	Время установления постоянного уровня на SPISIMO до прихода низкого уровня на SPICLK (полярность тактового сигнала = 1)	0	–	нс
t_v(SPCH-SIMO)S**	Время разрешения доступа SPISIMO к данным после прихода на SPICLK высокого уровня (полярность тактового сигнала = 0)	0,5t_c(SPC)S	–	нс
t_v(SPCL-SIMO)S**	Время разрешения доступа SPISIMO к данным после прихода на SPICLK низкого уровня (полярность тактового сигнала = 1)	0,5t_c(SPC)S	–	

_____

*t_c – длительность синхроимпульса = 1/SYSCLK = t_c(SYS).

**За полярность активного фронта тактового сигнала SPICLK отвечает бит полярности (SPICCR.6).

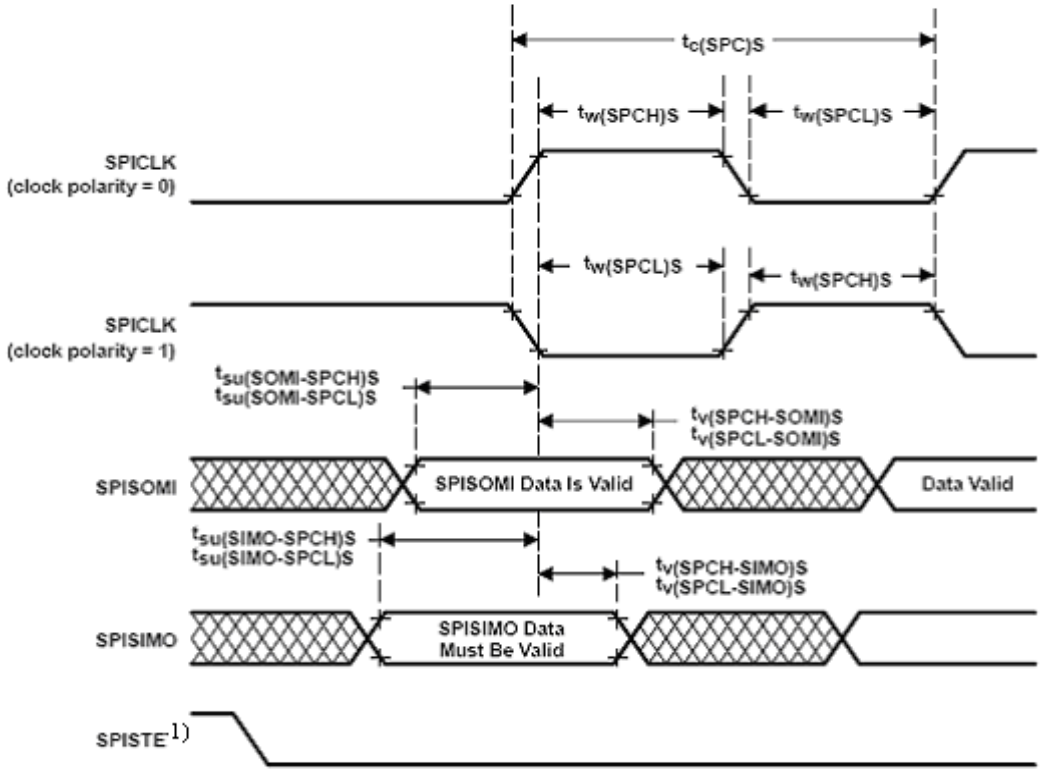


Рисунок 49 – Временные диаграммы в режиме SPI SLAVE
(фаза синхроимпульса равна 1)

Условные обозначения на рисунке 49:

Clock polarity = 0 – полярность тактового сигнала равна 0.

Clock polarity = 1 – полярность тактового сигнала равна 1.

Сигнал SPISTE должен быть активным до старта коммуникационного потока SPI и оставаться активным до полного завершения коммуникационного потока SPI.

11.15 12-битный аналого-цифровой преобразователь АЦП

12-битный АЦП имеет отдельный вход питания для аналоговой схемы. Выводы аналогового напряжения питания и аналоговой земли обозначаются как ∩VCC2 и ∩GND2 соответственно. Цель разделения питания цифровой и аналоговой части – улучшить исполнение АЦП путем исключения цифровых помех от переключения логических схем, что может происходить на выводах $U_{\cap GND2}$ и $U_{\cap VCC2}$ при их подключении к выводам питания цифровой части схемы.

Количество АЦП – два.

Все характеристики АЦП даются с учетом $U_{\cap GND2}$.

Разрешение – 12-бит (1024 значений).

Линейность обеспечена.

Режим преобразования входа: от 000h до 3FFh (000h для $U_{VI} \leq U_{\cap GND}$; 3FFh для $U_{VI} \geq U_{\cap VCC2}$), где U_{VI} – напряжение входного сигнала. Рекомендуемые условия эксплуатации представлены в таблице 49.

Таблица 49 – Рекомендованные условия эксплуатации

Параметр	Значение параметра, В		
	Мин.	Ном.	Макс.
$U_{\cap VCC2}$ – аналоговое питание	3,0	3,3	3,6
$U_{\cap GND2}$ – аналоговая «земля»	0	0	0
U_{VREFHI} – аналоговый источник смещения	U_{VREFLO}	–	$U_{\cap VCC2}$
U_{VREFLO} – аналоговая «земля» источника смещения	$U_{\cap GND2}$	–	U_{VREFHI}
$U_{\cap I}$ – входное аналоговое напряжения, выводы ADCIN0-ADCIN15	$U_{\cap GND2}$	–	$U_{\cap VCC2}$

Рабочие характеристики сверх рекомендованного диапазона условий эксплуатации представлены в таблице 50.

Таблица 50 - Рабочие характеристики сверх рекомендованного диапазона условий эксплуатации

Параметр	Описание		Макс.	Ед. изм.
$I_{\cap CC}$- аналоговый ток	$U_{\cap VCC2}$= 3,3 В		3	мА
C_{AI} емкость аналогового входа	Типичная емкостная нагрузка на аналоговый вход	нет выборки	6	пФ
		выборка	8	
Z_{AI} - полное сопротивление аналогового источника	Полное сопротивление аналогового источника для преобразований, чтобы оставаться в пределах спецификаций		22,4	кОм
E_{DNL}- дифференциальная нелинейная ошибка/погрешность	Различие между действительным шагом в ширину и идеальным значением		1,5	LSB
E_{INL} интегральная нелинейная ошибка/погрешность	Максимальное отклонение от оптимальной прямой линии через передаточные характеристики ADC, включая погрешность квантования		±1,5	LSB
$T_{d(PU)}$ - время задержки, включение питания до достоверного ADC	Время для стабилизации аналоговой части после включения питания		10	мкс

U_{VREFHI}=3,3 В и U_{VREFLO}=0 В составляет один LSB. Когда U_{VREFHI} уменьшается, а U_{VREFLO} увеличивается, то размер LSB уменьшается. Поэтому, абсолютная точность и дифференциальная/интегральная линейная погрешность в отношении LSB увеличивается.

Модуль АЦП предоставляет полную свободу в конструкции источников аналоговых входов. Период времени выборки независим от полного сопротивления источника. Период времени удержания происходит в первом такте АЦП после установки в 1 бита ADCIMSTART или бита ADCSOC управляющего регистра АЦП. (ADCTRL1, биты 13 и 0, соответственно). После этого преобразование происходит во время следующих шести циклов синхронизации АЦП. Цифровые регистры результата обновляются и корректируются на следующем цикле синхронизации АЦП после того, как преобразование завершено.

11.15.1 Схема входного вывода АЦП

Одно из наиболее распространенных прикладных ошибок аналогового преобразования является неподходящее полное сопротивление источника. На практике источник с минимальным полным сопротивлением должен использоваться для ограничения погрешности, так же как и для минимизации требуемого времени выборки; так полное сопротивление источника должно быть меньше, чем Z_{AI}. Типичная схема вывода входа АЦП показана на рисунке 50.

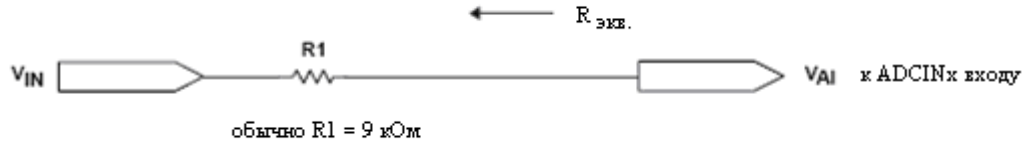


Рисунок 50 – Типичная схема вывода входа АЦП

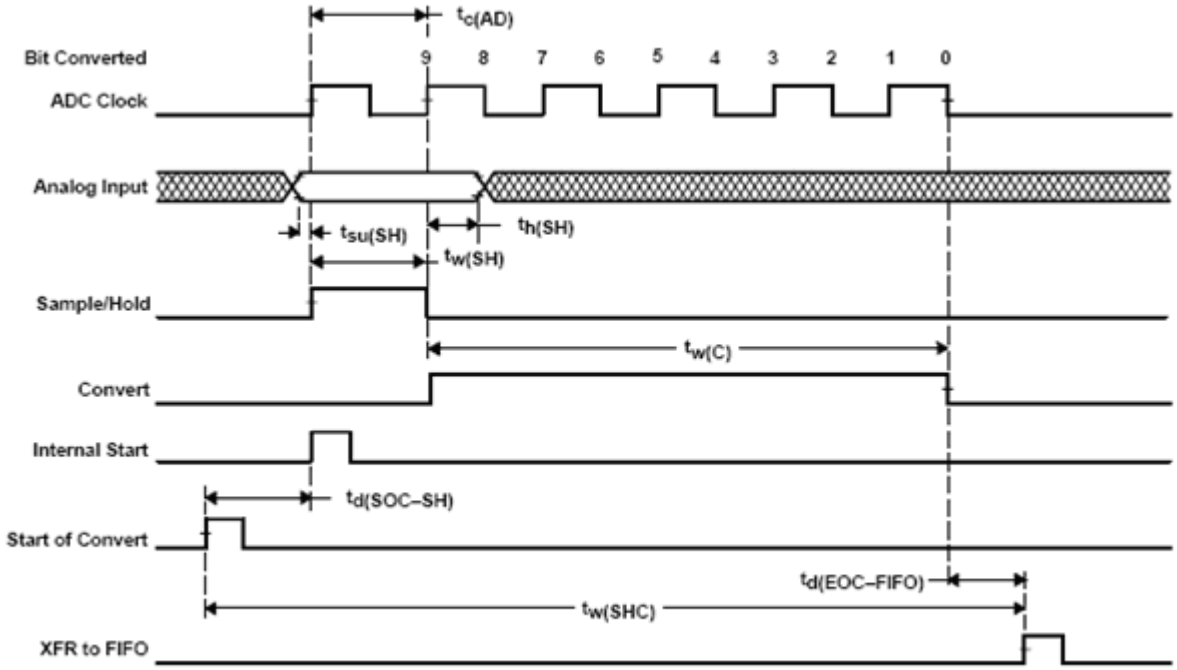
Требования к временам АЦП указаны в таблице 51 и на рисунке 51.

Таблица 51 – Требования к временам АЦП

Параметр	Мин.	Макс.	Ед. изм.
$t_{c(AD)}$ - цикл синхронизации, такт делителя частоты ADC	83	–	нс
$t_{w(SHC)}$ - длительность импульса, общее время выборки/сохранения напряжения на входе и время преобразования (см. примечание)	1,25	–	мкс
$t_{w(SH)}$ - длительность импульса, время выборки и сохранения напряжения на входе	$t_{c(AD)}$	–	мкс
$t_{su(SH)}$ - время установки, аналоговый ввод стабилен до начала выборки/сохранения напряжения на входе	0	–	нс
$t_{h(SH)}$ - время сохранения напряжения на входе, аналоговый ввод стабилен после завершения выборки/сохранения напряжения на входе	0	–	нс
$t_{w(C)}$ - длительность импульса, полное время преобразования	4,5 $t_{c(AD)}$	–	мкс
$t_{w(SOC-SH)}$ - время задержки, начало преобразования* до начала выборки и сохранения напряжения на входе	$3t_{c(SYS)}$	–	нс
$t_{d\ (EOC-FIFO)}$ - время задержки, конец преобразования до данных, загруженных в результирующее FIFO	$3t_{c(SYS)}$	–	нс

Примечание – Общее время выборки и сохранения напряжения на входе и преобразования определяется суммой $t_{d(SOC-SH)}$, $t_{w(SH)}$, $t_{w(C)}$ и $t_{d\ (EOC-FIFO)}$.

* Начало преобразования управляется через бит ADCIMSTART или бит ADCSOC, установленные программой, внешним активным стартовым сигналом (ADCSOC) или внутренним активным сигнал EVSOC.



Условные обозначения:
Bit Converted – преобразованный бит.
ADC Clock – синхронизация ADC.
Analog Input – аналоговый вход.
Sample/Hold – выборка/хранение напряжения на входе.
Convert – преобразователь.
Internal Start – внутренний старт.
Start of Convert – начало преобразования.
XFR to FIFO – флаг загрузки преобразования в FIFO

Рисунок 51 – Временная диаграмма аналого-цифрового преобразования

12 Файл регистров

В таблице 52 представлена совокупность всех программируемых регистров ИС 1867ВЦ10Т (предусмотрена для получения быстрой информации).

Таблица 52 - Регистры ИС 1867ВЦ10Т

ADDR	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	REG
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
	пространство памяти данных								
	статусные регистры ЦП								
	ARP			OV	OVM	1	INTM	DP(8)	ST0
	DP(7)	DP(6)	DP(5)	DP(4)	DP(3)	DP(2)	DP(1)	DP(0)	
	ARB			CNF	TC	SXM	C	1	ST1
	1	1	1	XF	1	1	PM		
	глобальная память и регистры WG сигналов прерывания								
00004h	—	—	—	—	—	—	—	—	IMR
	—	—	INT6 MASK	INT5 MASK	INT4 MASK	INT3 MASK	INT2 MASK	INT1 MASK	
00005h	—	—	—	—	—	—	—	—	GREG
	конфигурационные биты глобальной памяти (7-0)								
00006h	—	—	—	—	—	—	—	—	IFR
	—	—	INT6 FLAG	INT5 FLAG	INT4 FLAG	INT3 FLAG	INT2 FLAG	INT1 FLAG	
	регистры системной конфигурации								
07018h	RESET1	RESET0	—	—	—	—	—	—	SYSCR
	CLKSRC1	CLKSRC0	—	—	—	—	—	—	
07019h	Зарезервированы								
0701Ah	PORST	—	—	ILLADR	—	SWRST	WDRST	—	SYSSR
	—	—	HPO	—	VCCAOR	—	—	VECRD	
0701Bh to 0701Dh	Зарезервированы								
0701Eh	0	0	0	0	0	0	0	0	SYSIVR
	D7	D6	D5	D4	D3	D2	D1	D0	
0701Fh	Зарезервированы								
	WD/RTI управляющие регистры								
07020h	Зарезервированы								
07021h	D7	D6	D5	D4	D3	D2	D1	D0	RTICNTI
07022h	Зарезервированы								
07023h	D7	D6	D5	D4	D3	D2	D1	D0	WDCNTI
07024h	Зарезервированы								
07025h	D7	D6	D5	D4	D3	D2	D1	D0	WDKEY
07026h	Зарезервированы								
07027h	RTI FLAG	RTI ENA	—	—	—	RTIPS2	RTIPS1	RTIPS0	RTICR
07028h	Зарезервированы								
07029h	WD FLAG	WDDIS	WDCHK2	WDCHK1	WDCHK0	WDPS2	WDPS1	WDPS0	WDCR
	управляющие регистры модуля синхронизации								
0702Ah	Зарезервированы								
0702Bh	CLKMD(1)	CLKMD(0)	LOCK(1)	LOCK(0)	PDM(1)	PDM(0)	ACLKENA	PS	CKCR0
0702Ch	Зарезервированы								

Продолжение таблицы 52

ADDR	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	REG
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
	управляющие регистры модуля синхронизации (продолжение)								
0702Dh	CKINF(3)	CKINF(2)	CKINF(1)	CKINF(0)	DIV2	FB(2)	FB(1)	FB(0)	CKCR1
0702Eh to 07031h	Зарезервированы								
	A-к-D управляющие регистры								
07032h	SUSPEND-SOFT	SUSPEND-FREE	ADCIM-START	ADC2EN	ADC1EN	ADCCON-RUN	ADCINTEN	ADCINTFLAG	ADCTRL1
	ADCEOC	ADC2CHSEL			ADC1CHSEL			ADCSOC	
07033h	Зарезервированы								
07034h	—	—	—	—	—	ADCEVSOC	ADCEXTSOC	—	ADCTRL2
	ADCFIFO2		—	ADCFIFO1		ADCPSCALE			
07035h	Зарезервированы								
07036h	D9	D8	D7	D6	D5	D4	D3	D2	ADCFIFO
	D1	D0	0	0	0	0	0	0	
07037h	Зарезервированы								
07038h	D9	D8	D7	D6	D5	D4	D3	D2	ADCFIFO
	D1	D0	0	0	0	0	0	0	
07039h to 0703Fh	Зарезервированы								
	регистры системной конфигурации последовательного периферийного интерфейса (SPI)								
07040h	SPI SW RESET	CLOCK POLARITY	—	—	—	SPI CHAR2	SPI CHAR1	SPI CHAR0	SPICCR
07041h	—	—	—	OVERRUN INT ENA	CLOCK PHASE	MASTER/ SLAVE	TALK	SPI INT ENA	SPICTL
07042h	RECEIVER OVERRUN	SPI INT FLAG	—	—	—	—	—	—	SPISTS
07043h	Зарезервированы								
07044h	—	SPI BIT RATE 6	SPI BIT RATE 5	SPI BIT RATE 4	SPI BIT RATE 3	SPI BIT RATE 2	SPI BIT RATE 1	SPI BIT RATE 0	SPIBRR
07045h	Зарезервированы								
07046h	ERCVD7	ERCVD6	ERCVD5	ERCVD4	ERCVD3	ERCVD2	ERCVD1	ERCVD0	SPIEMU
07047h	RCVD7	RCVD6	RCVD5	RCVD4	RCVD3	RCVD2	RCVD1	RCVD0	SPIBUF
07048h	Зарезервированы								
07049h	SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0	SPIDAT
0704Ah to 0704Ch	Зарезервированы								
0704Dh	SPISTE DATA IN	SPISTE DATA OUT	SPISTE FUNCTION	SPISTE DATA DIR	SPICLK DATA IN	SPICLK DATA OUT	SPICLK FUNCTION	SPICLK DATA DIR	SPIPC1
0704Eh	SPISIMO DATA IN	SPISIMO DATA OUT	SPISIMO FUNCTION	SPISIMO DATA DIR	SPISOMI DATA IN	SPISOMI DATA OUT	SPISOMI FUNCTION	SPISOMI DATA DIR	SPIPC2
0704Fh	—	SPI PRIORITY	SPI ESPEN	—	—	—	—	—	SPIPRI

Продолжение таблицы 52

ADDR	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	REG
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
	Регистры управления конфигурацией последовательного коммуникационного интерфейса SCI								
07050h	STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	SCI ENA	ADDR/IDLE MODE	SCI CHAR2	SCI CHAR1	SCI CHAR0	SCICCR
07051h	—	RX ERR INT ENA	SW RESET	CLOCK ENA	TXWAKE	SLEEP	TXENA	RXENA	SCICTL1
07052h	BAUD15 (MSB)	BAUD14	BAUD13	BAUD12	BAUD11	BAUD10	BAUD9	BAUD8	SCIHBAUD
07053h	BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0 (LSB)	SCILBAUD
07054h	TXRDY	TX EMPTY	—	—	—	—	RX/BK INT ENA	TX INT ENA	SCICTL2
07055h	RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	—	SCIRXST
07056h	ERXDT7	ERXDT6	ERXDT5	ERXDT4	ERXDT3	ERXDT2	ERXDT1	ERXDT0	SCIRXEMU
07057h	RXDT7	RXDT6	RXDT5	RXDT4	RXDT3	RXDT2	RXDT1	RXDT0	SCIRXBUF
07058h	Зарезервированы								
07059h	TXDT7	TXDT6	TXDT5	TXDT4	TXDT3	TXDT2	TXDT1	TXDT0	SCITXBUF
0705Ah to 0705Dh	Зарезервированы								
0705Eh	SCITXD DATA IN	SCITXD DATA OUT	SCITXD FUNCTION	SCITXD DATA DIR	SCIRXD DATA IN	SCIRXD DATA OUT	SCIRXD FUNCTION	SCIRXD DATA DIR	SCIPC2
0705Fh	—	SCITX PRIORITY	SCIRX PRIORITY	SCI ESPEN	—	—	—	—	SCIPRI
07060h to 0706Fh	Зарезервированы								
	регистры управления внешними прерываниями								
07070h	XINT1 FLAG	—	—	—	—	—	—	—	XINT1CR
	—	XINT1 PIN DATA	0	—	—	XINT1 POLARITY	XINT1 PRIORITY	XINT1 ENA	
07071h	Зарезервированы								
07072h	NMI FLAG	—	—	—	—	—	—	—	NMICR
	—	NMI PIN DATA	1	—	—	NMI POLARITY	—	—	
07073h to 07077h	Зарезервированы								
07078h	XINT2 FLAG	—	—	—	—	—	—	—	XINT2CR
	—	XINT2 PIN DATA	—	XINT2 DATA DIR	XINT2 DATA OUT	XINT2 POLARITY	XINT2 PRIORITY	XINT2 ENA	
07079h	Зарезервированы								

Продолжение таблицы 52

ADDR	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	REG
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
	регистры управления внешними прерываниями (продолжение)								
0707Ah	XINT3 FLAG	—	—	—	—	—	—	—	XINT3CR
	—	XINT3 PIN DATA	—	XINT3 DATA DIR	XINT3 DATA OUT	XINT3 POLARITY	XINT3 PRIORITY	XINT3 ENA	
0707Bh to 0708Fh	Зарезервированы								
	регистры управления цифровым I/O								
07090h	CRA.15	CRA.14	CRA.13	CRA.12	CRA.11	CRA.10	CRA.9	CRA.8	OCRA
	—	—	—	—	CRA.3	CRA.2	CRA.1	CRA.0	
07091h	Зарезервированы								
07092h	—	—	—	—	—	—	—	—	OCRB
	CRB.7	CRB.6	CRB.5	CRB.4	CRB.3	CRB.2	CRB.1	CRB.0	
07093h to 07097h	Зарезервированы								
07098h	—	—	—	—	A3DIR	A2DIR	A1DIR	A0DIR	PADATDIR
	—	—	—	—	IOPA3	IOPA2	IOPA1	IOPA0	
07099h	Зарезервированы								
0709Ah	B7DIR	B6DIR	B5DIR	B4DIR	B3DIR	B2DIR	B1DIR	B0DIR	PBDATDIR
	IOPB7	IOPB6	IOPB5	IOPB4	IOPB3	IOPB2	IOPB1	IOPB0	
0709Bh	Зарезервированы								
0709Ch	C7DIR	C6DIR	C5DIR	C4DIR	C3DIR	C2DIR	C1DIR	C0DIR	PCDATDIR
	IOPC7	IOPC6	IOPC5	IOPC4	IOPC3	IOPC2	IOPC1	IOPC0	
0709Dh to 073FFh	Зарезервированы								
	регистры управления конфигурацией обще-целевых таймеров								
07400h	T3STAT	T2STAT	T1STAT	T3TOADC		T2TOADC		T1TOADC(1)	GPTCON
	T1TOADC(0)	TCOMPOE	T3PIN		T2PIN		T1PIN		
07401h	D15	D14	D13	D12	D11	D10	D9	D8	T1CNT
	D7	D6	D5	D4	D3	D2	D1	D0	
07402h	D15	D14	D13	D12	D11	D10	D9	D8	T1CMPR
	D7	D6	D5	D4	D3	D2	D1	D0	
07403h	D15	D14	D13	D12	D11	D10	D9	D8	T1PR
	D7	D6	D5	D4	D3	D2	D1	D0	
07404h	FREE	SOFT	TMODE2	TMODE1	TMODE0	TPS2	TPS1	TPS0	T1CON
	TSWT1	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR	
07405h	D15	D14	D13	D12	D11	D10	D9	D8	T2CNT
	D7	D6	D5	D4	D3	D2	D1	D0	
07406h	D15	D14	D13	D12	D11	D10	D9	D8	T2CMPR
	D7	D6	D5	D4	D3	D2	D1	D0	

Продолжение таблицы 52

ADDR	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	REG
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
	регистры управления конфигурацией обще-целевых таймеров (продолжение)								
07407h	D15	D14	D13	D12	D11	D10	D9	D8	T2PR
	D7	D6	D5	D4	D3	D2	D1	D0	
07408h	FREE	SOFT	TMODE2	TMODE1	TMODE0	TPS2	TPS1	TPS0	T2CON
	TSWT1	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR	
07409h	D15	D14	D13	D12	D11	D10	D9	D8	T3CNT
	D7	D6	D5	D4	D3	D2	D1	D0	
0740Ah	D15	D14	D13	D12	D11	D10	D9	D8	T3CMPR
	D7	D6	D5	D4	D3	D2	D1	D0	
0740Bh	D15	D14	D13	D12	D11	D10	D9	D8	T3PR
	D7	D6	D5	D4	D3	D2	D1	D0	
0740Ch	FREE	SOFT	TMODE2	TMODE1	TMODE0	TPS2	TPS1	TPS0	T3CON
	TSWT1	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR	
0740Dh to 07410h	Зарезервированы								
	регистры полного и простого устройства сравнения								
07411h	CENABLE	CLD1	CLD0	SVENABLE	ACTRLD1	ACTRLD0	FCOMPOE	SCOMPOE	COMCON
	SELTMR	SCLD1	SCLD0	SACTRLD1	SACTRLD0	SELCMP3	SELCMP2	SELCMP1	
07412h	Зарезервированы								
07413h	SVRDIR	D2	D1	D0	CMP6ACT1	CMP6ACT0	CMP5ACT1	CMP5ACT0	ACTR
	CMP4ACT1	CMP4ACT0	CMP3ACT1	CMP3ACT0	CMP2ACT1	CMP2ACT0	CMP1ACT1	CMP1ACT0	
07414h	—	—	—	—	—	—	—	—	SACTR
	—	—	SCMP3-ACT1	SCMP3-ACT0	SCMP2-ACT1	SCMP2-ACT0	SCMP1-ACT1	SCMP1-ACT0	
07415h	DBT7	DBT6	DBT5	DBT4	DBT3	DBT2	DBT1	DBT0	DBTCON
	EDBT3	EDBT2	EDBT1	DBTPS1	DBTPS0	—	—	—	
07416h	Зарезервированы								
07417h	D15	D14	D13	D12	D11	D10	D9	D8	CMPR1
	D7	D6	D5	D4	D3	D2	D1	D0	
07418h	D15	D14	D13	D12	D11	D10	D9	D8	CMPR2
	D7	D6	D5	D4	D3	D2	D1	D0	
07419h	D15	D14	D13	D12	D11	D10	D9	D8	CMPR3
	D7	D6	D5	D4	D3	D2	D1	D0	
0741Ah	D15	D14	D13	D12	D11	D10	D9	D8	SCMPR1
	D7	D6	D5	D4	D3	D2	D1	D0	
0741Bh	D15	D14	D13	D12	D11	D10	D9	D8	SCMPR2
	D7	D6	D5	D4	D3	D2	D1	D0	
0741Ch	D15	D14	D13	D12	D11	D10	D9	D8	SCMPR3
	D7	D6	D5	D4	D3	D2	D1	D0	
0741Dh to 0741Fh	Зарезервированы								

Окончание таблицы 52

ADDR	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	REG
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
	регистр устройства захвата								
07420h	CAPRES	CAPQEPN		CAP3EN	CAP4EN	CAP34TSEL	CAP12TSEL	CAP4TOADC	CAPCON
	CAP1EDGE		CAP2EDGE		CAP3EDGE		CAP4EDGE		
07421h	Зарезервированы								
07422h	CAP4FIFO		CAP3FIFO		CAP2FIFO		CAP1FIFO		CAPFIFO
	CAPFIFO15	CAPFIFO14	CAPFIFO13	CAPFIFO12	CAPFIFO11	CAPFIFO10	CAPFIFO9	CAPFIFO8	
07423h	D15	D14	D13	D12	D11	D10	D9	D8	CAP1FIFO
	D7	D6	D5	D4	D3	D2	D1	D0	
07424h	D15	D14	D13	D12	D11	D10	D9	D8	CAP2FIFO
	D7	D6	D5	D4	D3	D2	D1	D0	
07425h	D15	D14	D13	D12	D11	D10	D9	D8	CAP3FIFO
	D7	D6	D5	D4	D3	D2	D1	D0	
07426h	D15	D14	D13	D12	D11	D10	D9	D8	CAP4FIFO
	D7	D6	D5	D4	D3	D2	D1	D0	
07427h to 0742Bh	Зарезервированы								
	Регистры менеджера событий (EV), управляющие прерываниями								
0742Ch	—	—	—	—	—	T1OFINT ENA	T1UFINT ENA	T1CINT ENA	EVIMRA
	T1PINT ENA	SCMP3INT ENA	SCMP2INT ENA	SCMP1INT ENA	CMP3INT ENA	CMP2INT ENA	CMP1INT ENA	PDPINT ENA	
0742Dh	—	—	—	—	—	—	—	—	EVIMRB
	T3OFINT ENA	T3UFINT ENA	T3CINT ENA	T3PINT ENA	T2OFINT ENA	T2UFINT ENA	T2CINT ENA	T2PINT ENA	
0742Eh	—	—	—	—	—	—	—	—	EVIMRC
	—	—	—	—	CAP4INT ENA	CAP3INT ENA	CAP2INT ENA	CAP1INT ENA	
0742Fh	—	—	—	—	—	T1OFINT FLAG	T1UFINT FLAG	T1CINT FLAG	EVIFRA
	T1PINT FLAG	SCMP3INT FLAG	SCMP2INT FLAG	SCMP1INT FLAG	CMP3INT FLAG	CMP2INT FLAG	CMP1INT FLAG	PDPINT FLAG	
07430h	—	—	—	—	—	—	—	—	EVIFRB
	T3OFINT FLAG	T3UFINT FLAG	T3CINT FLAG	T3PINT FLAG	T2OFINT FLAG	T2UFINT FLAG	T2CINT FLAG	T2PINT FLAG	
07431h	—	—	—	—	—	—	—	—	EVIFRC
	—	—	—	—	CAP4INT FLAG	CAP3INT FLAG	CAP2INT FLAG	CAP1INT FLAG	
07432h	0	0	0	0	0	0	0	0	EVIVRA
	0	0	D5	D4	D3	D2	D1	D0	
07433h	0	0	0	0	0	0	0	0	EVIVRB
	0	0	D5	D4	D3	D2	D1	D0	
07434h	0	0	0	0	0	0	0	0	EVIVRC
	0	0	D5	D4	D3	D2	D1	D0	
07435h to 0743Fh	Зарезервированы								

ADDR	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	REG
	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	
	пространство I/O								
	регистры, управляющие генератором состояний ожидания								
0FFFFh	—	—	—	—	—	—	—	—	WSGR
	—	—	—	—	AVIS	ISWS	DSWS	PSWS	

13 Описание периферийных устройств

13.1 Обзор ИС 1867ВЦ10Т

ИС 1867ВЦ10Т является некоторым стандартом для однокристальных цифровых регуляторов частоты вращения двигателя. ИС 1867ВЦ10Т может выполнять 120 миллионов операций в секунду. Почти все команды выполняются за один цикл в 8,33 нс. Такая высокая эффективность позволяет выполнять сложные алгоритмы управления в реальном времени, такие как адаптивное управление и фильтры Калмана. Очень высокие частоты дискретизации могут так же использоваться для уменьшения задержек в циклах.

ИС 1867ВЦ10Т имеет архитектуру, которая необходима для высокоскоростной обработки сигналов и функций цифрового управления, и имеет периферийные устройства, требуемые для обеспечения однокристальной реализации приложений для регулирования частоты вращения двигателя. ИС 1867ВЦ10Т изготовлена с использованием субмикронной КМОП технологии, с достижением малой мощности рассеяния. Также с включением нескольких режимов пониженного потребления питания для экономии энергопотребления.

Приложения, использующие преимущества улучшенной производительности ИС 1867ВЦ10Т, включают:

- промышленные электродвигатели;
- контроллеры и усилители мощности;
- автомобильные системы, такие как электронное усиление рулевого управления, АБС тормозов и климат-контроль;
- устройства вентиляции и охлаждения двигателя;
- принтеры, копиры и другое офисное оборудование;
- устройства хранения информации;
- роботы и программно-управляемые станки.

Для функционирования в качестве главного контроллера системы ЦПОС должны иметь устойчивую к ошибкам внутрипроцессорную систему ввода-вывода и другие внешние устройства. Менеджер событий в ИС 1867ВЦ10Т отличается от подобных на ЦПОС. Этот оптимизированный периферийный блок, связанный с высокоэффективным ядром ЦПОС, позволяет использовать усовершенствованную технику управления для высокоточного и высокоэффективного регулирования скорости для всех типов двигателей. Включенные в модуль менеджера событий специальные функции формирования широтно-импульсной модуляции (PWM), такие как программируемая функция «мертвого» времени и конечный автомат пространственного вектора PWM для трёхфазных двигателей, обеспечивают достижение максимальной эффективности при переключении на мощных транзисторах. Три независимых таймера, каждый из которых имеет свой регистр сравнения, поддерживают формирование как асимметричных, так и симметричных PWM сигналов. Два из четырёх входов захвата получают напрямую сигналы со схемы «квадратурной» обработки сигналов импульсного датчика положения от оптического кодирующего устройства.

Особенности ИС 1867ВЦ10Т:

– Ядро процессора ИС 1867ВЦ10Т:

– 32-битное центральное арифметическо-логическое устройство АЛУ;

– 32-битный аккумулятор;

– 16-бит × 16-бит параллельный умножитель с 32-битной вычислительной способностью;

– три счетных сдвиговых регистра;

– восемь 16-битных вспомогательных регистра со специализированным арифметическим блоком для косвенной адресации памяти данных.

– Память ИС 1867ВЦ10Т реализует три отдельных адресных пространства: адресное пространство программной памяти, памяти данных и I/O (ввода-вывода).

ИС 1867ВЦ10Т может адресовать память программ:

– 64К слов внутрикристальной Flash памяти при MP/MC=0;

– 64К слов внешней памяти при MP/MC=1;

– 0.5К слов внутрикристальной DARAM при CNF=1.

ИС 1867ВЦ10Т может адресовать память данных:

– 45К слов ×16 бит внутрикристальной SRAM;

– 1824 слов ×16 бит внутрикристальной DARAM при CNF=0;

– 1312 слов ×16 бит внутрикристальной DARAM при CNF=1.

ИС 1867ВЦ10Т может адресовать память пространства ввода-вывода:

– 64К слов внешней памяти.

– Программное управление:

– четырёхуровневая работа в конвейерном режиме;

– восьмиуровневый аппаратный стек;

– шесть внешних прерываний: прерывание защиты электропитания, сброс, NMI и три маскируемых прерывания.

– Система команд:

– исходный код, совместимый с М1867ВМ1, 1867ВМ2, 1867ВЦ2Т, 1867ВЦ5Т;

– однокомандная операция повтора;

– одноцикловые команды умножения/накопления;

– команды перемещения блока памяти для управления командами/данными;

– возможность индексной адресации;

– бит-реверсная возможность индексной адресации для быстрого преобразования Фурье по основанию 2.

– Производительность:

– статическая КМОП технология;

– четыре режима пониженного потребления питания для снижения потребляемой мощности.

– Эмуляция: интерфейс тестового JTAG порта по стандарту IEEE 1149.1 с внутрикристальной сканирующей логикой эмуляции.

– Скорость: 8,33 нс (120 MIPS) время команд цикла с большинством команд, выполняемых в одном цикле.

– Модуль менеджера событий (Event manager EV):

– 12 каналов с широтно-импульсной модуляцией PWM (9 независимых);

– три 16-битных универсальных таймера с шестью режимами, включающие непрерывный счет в прямом направлении и непрерывный счет в прямом/обратном направлении;

– три 16-битных блока полного сравнения с возможностью задания «мертвого» времени;

– три 16-битных блока простого сравнения;

– четыре блока захвата, два из которых имеют интерфейс «квадратурной» обработки сигналов импульсного датчика положения.

– Сдвоенный 12-битный аналогово-цифровой преобразователь.

– 28 индивидуально-программируемых, мультиплексированных ввода-вывода.

– Модуль фазовой автоподстройки частоты тактового сигнала PLL.

– Модуль сторожевого таймера WD и блока прерываний реального времени RTI.

– Последовательный коммуникационный интерфейс SCI.

– Последовательный периферийный интерфейс SPI.

– последовательный интерфейс связи CAN.

– последовательный двухпроводной I2C интерфейс.

13.2 Модуль менеджера событий

13.2.1 Обзор модуля EV

Модуль менеджера событий (EV) обеспечивает широкий спектр функций и возможностей, которые особенно полезны для контроля перемещения и приложений управления двигателем.

Функциональные блоки модуля EV

На рисунке 52 показана структурная схема модуля EV. Модуль EV содержит следующие функциональные блоки:

– Три таймера общего назначения (General Purpose (GP) таймеры).

– Три блока полного сравнения.

– Три блока простого сравнения.

– Цепи широтно-импульсной модуляции (PWM), которые содержат цепи пространственного вектора PWM, блоки генерации «мертвого» времени и выходную логику.

– Четыре блока захвата.

– Цепи «квадратурной» обработки сигналов импульсного датчика положения (Quadrature encoder pulse (QEP)).

– Логику прерывания.

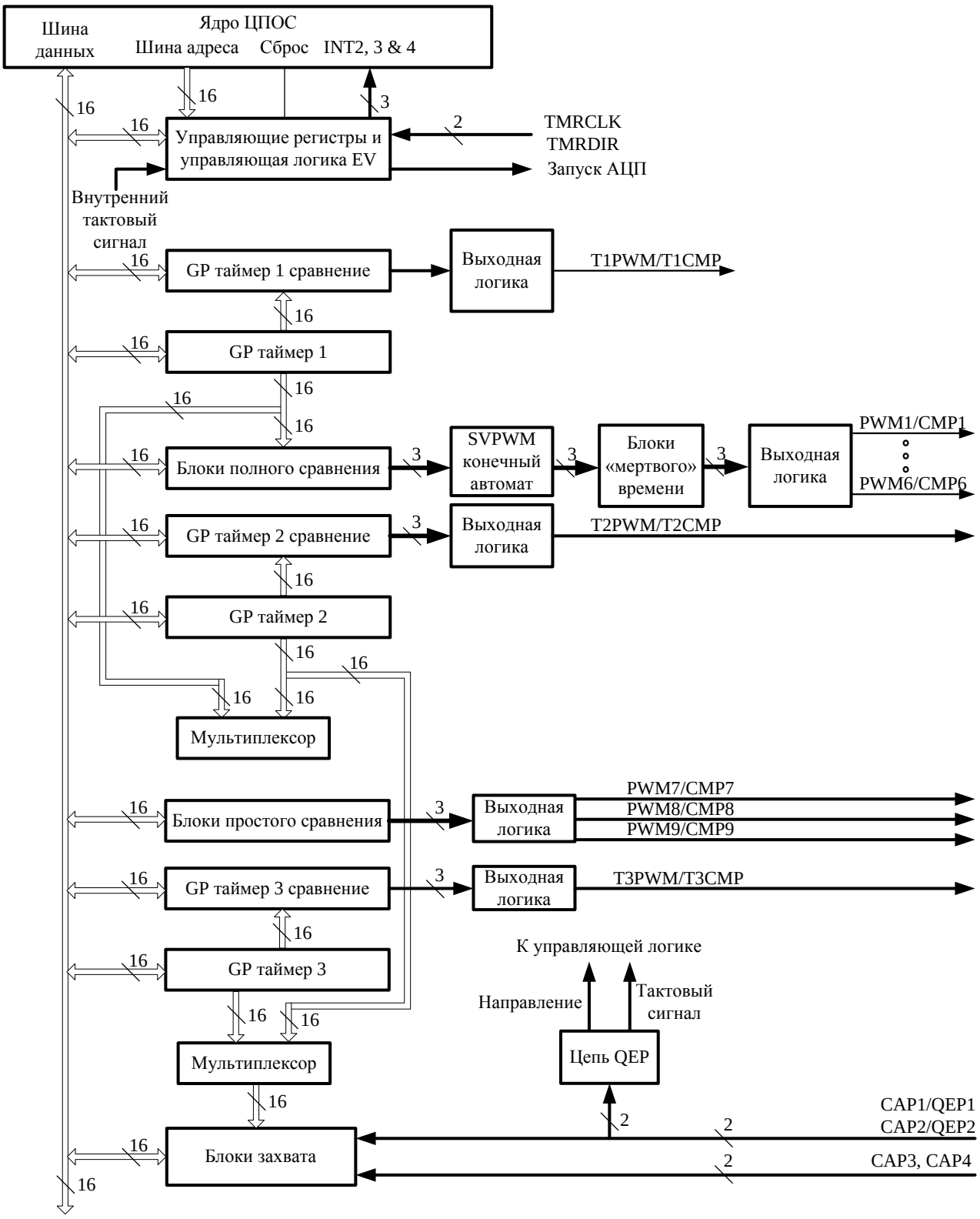


Рисунок 52 – Структурная схема модуля менеджера событий

Выводы модуля EV

Модуль EV использует 12 выводов для выходов сравнения (compare) и PWM:

– Три выходных вывода сравнения(compare)/PWM и таймера общего назначения:
  – T1PWM/T1CMP;
  – T2PWM/T2CMP;
  – T3PWM/T3CMP.
– Шесть выходных выводов полного сравнения (compare) и PWM:
  – PWM1/CMP1;
  – PWM2/CMP2;
  – PWM3/CMP3;
  – PWM4/CMP4;
  – PWM5/CMP5;
  – PWM6/CMP6.
– Три выходных вывода простого сравнения (compare) и PWM:
  – PWM7/CMP7;
  – PWM8/CMP8;
  – PWM9/CMP9.

Модуль EV использует четыре вывода CAP1/QEP1, CAP2/QEP2, CAP3 и CAP4 как входы от импульсного датчика положения.

Таймеры общего назначения в модуле EV могут быть запрограммированы для работы как от внешнего синхросигнала, так и от внутреннего тактового сигнала процессора. Вывод TMRCLK поддерживает вход внешнего тактового сигнала.

Контакт TMRDIR используется для определения направления счета, когда таймер общего назначения находится в режиме сложения/вычитания.

Все входы EV модуля синхронизированы с внутренним тактовым сигналом процессора. Изменение на входах EV модуля должно удерживаться, пока два передних фронта тактового сигнала (то есть, два передних фронта CLKOUT, если тактовый импульс процессора был выбран как источник выхода CLKOUT) не появятся перед их распознанием в модуле EV. Поэтому рекомендуется, чтобы любой входной сигнал был задержан более чем на два цикла тактового сигнала.

Выводы модуля EV объединены в таблице 53.

Таблица 53 – Выводы модуля EV

Обозначение вывода	Описание
CAP1/QEP1	Вход блока захвата 1 или QEP1
CAP2/QEP2	Вход блока захвата 2 или QEP2
CAP3	Вход блока захвата 3
CAP4	Вход блока захвата 4
PWM1/CMP1	Выход 1 блока полного сравнения(compare)/PWM выход 1
PWM2/CMP2	Выход 1 блока полного сравнения(compare)/PWM выход 2
PWM3/CMP3	Выход 2 блока полного сравнения(compare)/PWM выход 1
PWM4/CMP4	Выход 2 блока полного сравнения(compare)/PWM выход 2

Окончание таблицы 53

Обозначение вывода	Описание
PWM5/CMP5	Выход 3 блока полного сравнения (compare)/PWM выход 1
PWM6/CMP6	Выход 3 блока полного сравнения (compare)/PWM выход 2
PWM7/CMP7	Выход 1 блока простого сравнения (compare)/PWM выход 1
PWM8/CMP8	Выход 2 блока простого сравнения (compare)/PWM выход 2
PWM9/CMP9	Выход 3 блока простого сравнения (compare)/PWM выход 3
T1PWM/T1CMP	Выход сравнения первого таймера (compare)/PWM выход
T2PWM/T2CMP	Выход сравнения второго таймера (compare)/PWM выход
T3PWM/T3CMP	Выход сравнения третьего таймера (compare)/PWM выход
TMRCLK	Вход внешнего тактового сигнала для универсальных таймеров
TMRDIR	Сигнал направления счета таймеров

Защита электропитания

Внешнее прерывание формируется, когда сигнал PDPINT установлен в 0. Это прерывание предназначено для безопасного функционирования таких устройств, как силовые преобразователи и двигатели. Если PDPINT не маскирован, все выводы модуля EV устанавливаются в высокоимпедансное состояние аппаратно, сразу после того, как PDPINT установлен в 0. Флаг прерывания соответствующий PDPINT также должен быть установлен при этих условиях; однако, установления не произойдёт пока переключение PDPINT не будет уточнено и синхронизировано с внутренним тактовым сигналом процессора. Уточнение и синхронизация имеют задержку от трёх до четырёх тактовых периодов. Если PDPINT не маскирован, флаг удерживает выводы модуля EV в высокоимпедансном состоянии и формирует запрос прерывания к ядру ЦПОС. Поэтому для поддержания выводов в высокоимпедансном состоянии PDPINT должен находиться в низком состоянии более чем четыре периода тактового сигнала. Установление флага не зависит от маскирования PDPINT. Высокоимпедансное состояние выводов устройства сравнения будет оставаться до тех пор пока на выводе PDPINT не произойдёт необходимого переключения. PDPINT используется для информирования программы мониторинга двигателя о нарушениях в работе, таких как: выход питающего напряжения за допустимые границы, перегрузка по току, избыточное увеличение температуры и т.п.

Регистры модуля EV

Все регистры модуля EV картированы в память данных. Их адреса занимают 64 16-битных слова из 64К слов диапазона адресов памяти данных от 7400h до 743Fh. Регистры обрабатываются программой как ячейки памяти данных и позволяют получать доступ для широкого набора команд ЦПОС. Неопределённые биты регистров модуля EV считываются нулями.

Прерывания модуля EV

Прерывания, формируемые модулем EV, распределены на три группы. В модуле EV имеются три регистра EVIFRA, EVIFRB и EVIFRC для флагов трёх групп

прерываний. Три группы прерываний связаны с тремя входами прерываний процессора. Каждая группа прерываний модуля EV имеет соответствующий регистр вектора прерывания, EVIFVRx (x = A, B и C), который может быть считан пользовательской программой для получения идентификатора вектора источника прерывания. Каждый источник прерывания имеет уникальный идентификатор вектора.

Флаг прерывания устанавливается, когда событие прерывания (например, совпадение регистра сравнения и GP таймера) сформировано и разрешено. Существует регистр маски прерывания, EVIMRx (x = A, B и C), соответствующий каждой группе прерываний модуля EV. Прерывание маскировано, если соответствующий бит в EVIMRA, EVIMRB или EVIMRC установлен в 0, и не маскировано, если соответствующий бит установлен в 1. Запрос прерывания будет сформирован флагом, если флаг установлен, не маскирован и не выполняются никакие другие, немаскированные прерывания с более высоким приоритетом в той же группе.

Установка и сброс флагов прерываний, тем не менее, не зависят от маскирования соответствующих прерываний. Поэтому флаг прерывания может быть проверен программно для контроля события, когда соответствующее прерывание маскировано. Флаг прерывания может быть сброшен в 0 двумя способами:

– пользовательская программа записывает 1 в соответствующий бит в EVIMRA, EVIMRB или EVIMRC;

– пользовательская программа считывает идентификатор вектора после того, как запрос прерывания, сформированный его группой, был принят.

Идентификатор вектора флага, который имеет высший приоритет среди флагов, установленных в группе, будет загружен в аккумулятор, когда вектор прерывания считывается после того, как запрос прерывания, сформированный его группой, был принят. Нулевое значение будет возвращено, когда регистр вектора прерывания группы прерываний считан и в группе не установлен флаг прерывания. Это предотвращает распознавание потерянных прерываний как прерываний модуля EV.

13.2.2 Адреса регистров модуля EV

В таблицах 54 – 57 приведены адреса регистров модуля EV.

Таблица 54 – Адреса регистров GP таймера

Адрес	Регистр	Описание
7400h	GPTCON	Управляющий регистр таймера общего назначения
7401h	T1CNT	GP таймер 1 счетный регистр
7402h	T1CMPR	GP таймер 1 регистр сравнения
7403h	T1PR	GP таймер 1 регистр периода
7404h	T1CON	GP таймер 1 управляющий регистр
7405h	T2CNT	GP таймер 2 счетный регистр
7406h	T2CMPR	GP таймер 2 регистр сравнения
7407h	T2PR	GP таймер 2 регистр периода
7408h	T2CON	GP таймер 2 управляющий регистр
7409h	T3CNT	GP таймер 3 счетный регистр
740Ah	T3CMPR	GP таймер 3 регистр сравнения
740Bh	T3PR	GP таймер 3 регистр периода
740Ch	T3CON	GP таймер 3 управляющий регистр

Таблица 55 – Адреса регистров блоков полного и простого сравнения

Адрес	Регистр	Описание
7411h	COMCON	Управляющий регистр сравнения
7413h	ACTR	Операционный управляющий регистр полного сравнения
7414h	SACTR	Операционный управляющий регистр простого сравнения
7415h	DBTCON	Управляющий регистр таймера «мертвого» времени
7417h	CMPR1	Регистры сравнения блока полного сравнения
7418h	CMPR2	
7419h	CMPR3	
741Ah	SCMPR1	Регистры сравнения блока простого сравнения
741Bh	SCMPR2	
741Ch	SCMPR3	

Таблица 56 – Адреса регистров блока захвата и цепи «квадратурной» обработки сигналов импульсного датчика положения

Адрес	Регистр	Описание
7420h	CAPCON	Управляющий регистр захвата
7422h	CAPFIFO	Статусный регистр захвата FIFO
7423h	CAP1FIFO	Двухуровневые FIFO стеки
7424h	CAP2FIFO	
7425h	CAP3FIFO	
7426h	CAP4FIFO	

Таблица 57 – Адреса регистров прерываний модуля EV

Адрес	Регистр	Описание
742Ch	EVIMRA	Регистры масок прерываний
742Dh	EVIMRB	
742Eh	EVIMRC	
742Fh	EVIFRA	Регистры флагов прерываний
7430h	EVIFRB	
7431h	EVIFRC	
7432h	EVIVRA	Регистры векторов прерываний
7433h	EVIVRB	
7434h	EVIVRC	

13.2.3 Таймер общего назначения (GP таймер)

Обзор GP таймера общего назначения

В ИС предусмотрено использование трёх GP таймеров общего назначения в модуле EV. Эти таймеры используются как независимые временные оси в таких приложениях как:

– формирование периода дискретизации/квантования в управляющей системе;

– обеспечение временного масштаба для функционирования блока захвата и цепи «квадратурной» обработки сигналов импульсного датчика положения;

– обеспечение временного масштаба для функционирования блоков полного и простого сравнения и соответствующих PWM цепей для формирования выводов сравнения (compare)/PWM.

Функциональные блоки GP таймера

На рисунке 53 представлена структурная схема GP таймера. Каждый GP таймер включает в себя:

– Один 16-битный счетчик сложения и вычитания с возможностью чтения и записи (R/W), TxCNT (x = 1, 2, 3).

– Один 16-битный регистр сравнения (теневой) с возможностью чтения и записи (R/W), TxCMPR (x = 1, 2, 3).

– Один 16-битный регистр периода таймера (теневой) с возможностью чтения и записи (R/W), TxPR (x = 1, 2, 3).

– 16-битный управляющий регистр с возможностью чтения и записи (R/W), TxCON (x = 1, 2, 3).

– Программируемый делитель частоты, применяемый для деления как внутреннего, так внешнего тактового сигнала.

– Логику управления и прерывания.

– Один вывод сравнения GP таймера, TxPWM/TxCMP.

– Выходную логику.

Другой регистр управления GPTCON определяет операции, производимые GP таймером при различных событиях таймера и указывает направление счёта всех трёх таймеров. GPTCON имеет возможность чтения и записи, тем не менее, запись трёх статусных бит не дает эффекта.

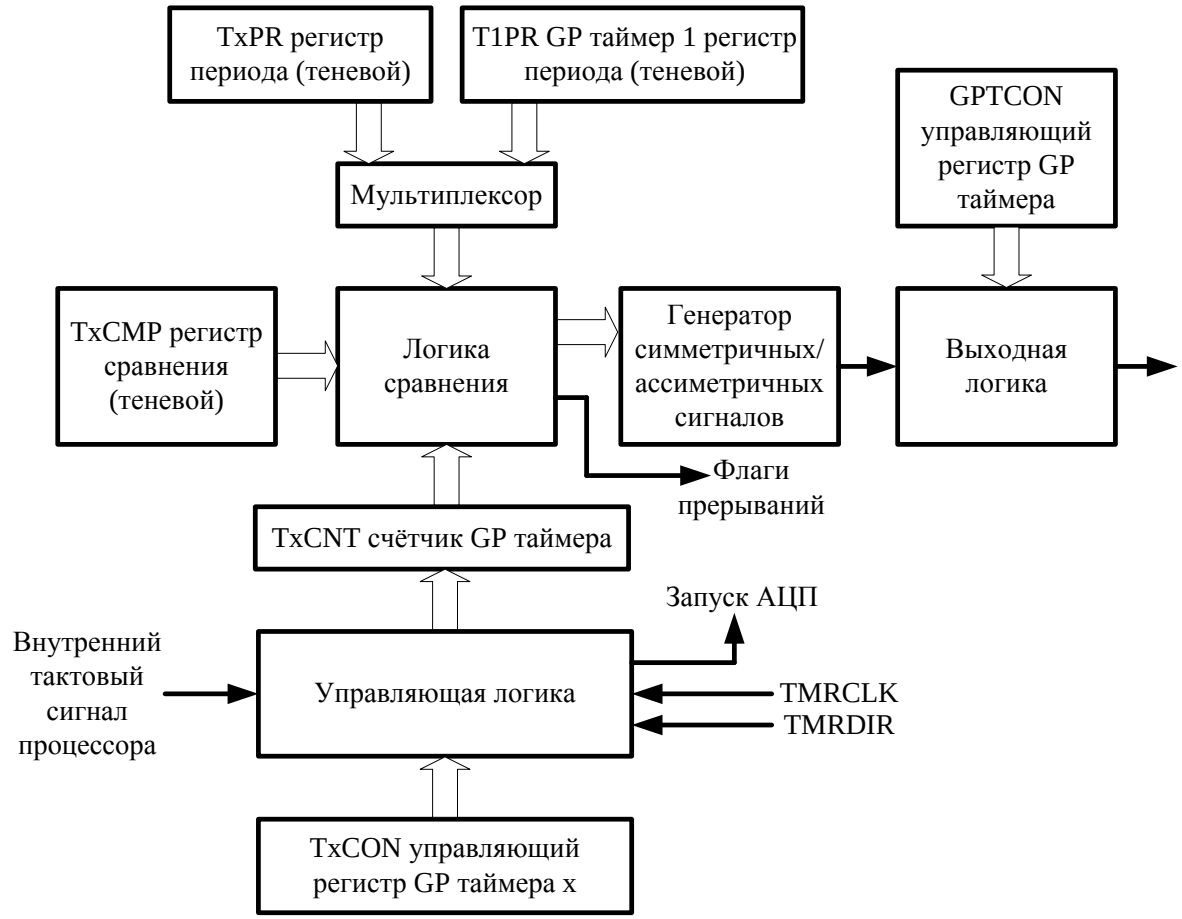


Рисунок 53 – Структурная схема GP таймера (x = 1, 2, 3)

Входы GP таймера

GP таймер имеет следующие входы:

– Внутренний тактовый сигнал, который поступает непосредственно от ядра и имеющий такую же частоту, как и тактовый сигнал процессора.

– Внешний тактовый сигнал TMRCLK, который имеет максимальную частоту в четверть тактового сигнала процессора.

– Вход направления TMRDIR, используемый GP таймером для задания направления счета в прямом/обратном счетном режиме.

– Сигнал сброса RESET.

К тому же, GP таймер 3 принимает переполнение GP таймера 2 как внутренний тактовый сигнал, когда GP таймер 2 и GP таймер 3 соединены последовательно в 32-битный таймер. Когда GP таймер используется вместе с цепью QEP, то цепь QEP формирует направление счета и тактовый сигнал таймера.

Выходы GP таймера

GP таймер имеет следующие выходы:

– Выходы сравнения GP таймер TxPWM/TxCMP, x=1, 2, 3.

– Сигнал запуска модуля АЦП.

– Сигналы обнуления, переполнения, совпадения при сравнении и совпадении периода к его собственной логике сравнения и к блокам полного и простого сравнения.

– Биты указателя направления счета.

Управление работой GP таймером

Управление режимом работы GP таймера осуществляется управляющим регистром TxCON. Биты TxCON регистра определяют:

– Какой из шести режимов счета GP таймера используется.

– Используется внешний или внутренний тактовый сигнал.

– Какой из шести масштабирующих множителей для входного тактового сигнала (в диапазоне от 1 до 1/128) используется.

– При каких условиях перезагружается регистр сравнения таймера.

– Разрешено или запрещено использование таймера.

– Разрешена или запрещена операция сравнения GP таймера.

– Определяет ли регистр периода GP таймер 1 или его собственный регистр периода собственный период (только для T2CON и T3CON).

– Должно ли использоваться переполнение GP таймера 2 как собственный тактовый сигнал (только для T3CON).

Управляющий регистр GP таймером (GPTCON)

Управляющий регистр GPTCON определяет операции, производимые GP таймером при различных событиях таймера, и определяет направление счёта.

Регистры сравнения GP таймера

Регистр сравнения, связанный с GP таймером, содержит значение, которое постоянно сравнивается со счетчиком GP таймера. Когда происходит совпадение, то на соответствующем выходе сравнения происходит переключение в соответствии с комбинацией битов GPTCON, устанавливается соответствующий флаг прерывания и формируется запрос прерывания к ядру, если прерывание не маскировано и не выполняются никакие другие немаскированные прерывания с более высоким приоритетом в той же группе. Операция сравнения GP таймера может быть разрешена или запрещена соответствующим битом TxCON.

Регистр периода GP таймера

Значение в регистре периода GP таймера определяет период таймера. Функционирование GP таймера прекращается и удерживается в текущем состоянии, сбрасывается в 0 или начинается счет в обратном направлении, когда происходит совпадение между регистром периода и счетчиком таймера, зависящее от того, какой используется счётный режим.

Двойная буферизация регистров периода и сравнения GP таймера

Регистры периода и сравнения (TxCMPR и TxPR) GP таймера скрыты. Новое значение может быть записано в один из этих регистров в любое время в течение периода. Однако, новое значение записывается в соответствующий теневой регистр. Для регистра сравнения содержимое теневого регистра загружается в работающий (активный) регистр только когда происходит точное событие таймера, определяемое TxCON. Для регистра периода работающий регистр перезагружен со значением, содержащимся в теневом регистре только когда значение счетного регистра TxCNT равно 0. Условия, при которых регистр сравнения перезагружается, следующие:

– Сразу после того как записан теневой регистр.

– При обнулении, то есть когда значение счётчика GP таймера равно 0.

– При опустошении или совпадении периода, то есть когда значение счётчика равно 0 или когда значение счётчика равно значению регистра периода.

Свойство двойной буферизации регистров периода и сравнения позволяет коду приложения обновлять регистры периода и сравнения в любое время в течение периода для смены периода таймера и ширины PWM для этого периода. Быстрое изменение значения периода таймера в отношении формирования PWM означает быстрое изменение несущей частоты PWM колебаний.

ПРЕДУПРЕЖДЕНИЕ! Регистр периода GP таймера должен быть инициализирован перед тем как счетчик установится в отличное от нуля значение, иначе значение регистра периода останется неизменным до следующего опустошения.

Примечание – Регистр сравнения прозрачен (только что загруженное значение поступает прямо в активный регистр), когда соответствующая операция сравнения запрещена. Это относится ко всем регистрам сравнения в модуля EV.

Выходы сравнения (compare) GP таймера

Выходы сравнения (compare) GP таймера могут быть определены как установленные в активный высокий уровень, активный низкий уровень, принудительный высокий уровень, принудительный низкий уровень в зависимости от конфигурации бит GPTCON. Выходы изменяются от низкого до высокого (высокого до низкого) уровня при первом совпадении при сравнении, когда оно в активном высоком (низком) уровне. Далее изменяются от высокого до низкого (от низкого до высокого) уровня при втором совпадении сравнения, если GP таймер находится в режимах прямого/обратного счёта или при совпадении периодов, если GP таймер находится в режимах прямого счёта. Вывод сравнения (compare) становится в высокий (низкий) уровень сразу после указания на принудительный высокий (низкий) уровень.

Направление счета GP таймера

Направление счета всех трёх GP таймеров отражено в соответствующих битах GPTCON во время всех операций таймера:

– 1 отображает прямое направление счета.

– 0 отображает обратное направление счета.

Ввод TMRDIR определяет направление счета, когда GP таймер находится в режиме прямого/обратного счёта. Когда TMRDIR имеет высокий уровень, определён прямой счет; когда TMRDIR имеет низкий уровень, определён обратный счет.

Тактовый сигнал GP таймера

Источником тактового сигнала GP таймера может быть как внутренний тактовый сигнал процессора, так и внешний сигнал на входе TMRCLK. Частота внешнего тактового сигнала должна быть меньше либо равной четверти частоты тактового сигнала процессора. GP таймер 2 и 3 могут вместе работать как один 32-битный таймер, а также могут использоваться в цепях QEP в режиме прямого/обратного счёта. В этом случае цепи QEP подают на таймер и тактовый сигнал, и направление счета.

Для тактового входа каждого GP таймера предусмотрен широкий диапазон масштабирующих множителей.

32-битный таймер

Сигнал переполнения GP таймера 2 используется как тактовый вход GP таймера 3, когда GP таймер 2 и GP таймер 3 последовательно соединены в 32-битный таймер. В этом случае счетчик GP таймера 2 будет работать как младшие 16 разрядов 32-битного счетчика. 32-битный таймер, полученный таким образом, может функционировать только в режиме прямого/обратного счёта (см. режимы работы GP таймера) с внешним или с внутренним тактовым сигналом и внешними входами направления. Цепь QEP может также быть выбрана для формирования счетного тактового сигнала и направления счета 32-битного таймера. Регистры периода GP таймера 2 и GP таймера 3 в этом случае соединены последовательно для получения 32-битного регистра периода для 32-битного таймера, пока операция сравнения основана на индивидуальных регистрах сравнения и происходят индивидуальные 16-битные совпадения при сравнении. Опустошение и переполнение 32-битного таймера так же 32-битные.

Вход тактового сигнала от QEP

Когда выбрана цепь QEP, то она может формировать входной тактовый сигнал и направление счёта для GP таймера 2, GP таймера 3 или GP таймера 2 и GP таймера 3, объединенных вместе как 32-битный таймер в режиме прямого/обратного счёта. Этот входной тактовый сигнал не может быть масштабирован цепью предмасштабирования GP таймера (то есть делитель, выбранного GP таймера, всегда равен 1, если цепь QEP выбрана как источник входного тактового сигнала). К тому же частота сигнала формируемого цепями QEP вчетверо больше частоты каждого входного канала QEP, потому что и передний и задний фронт обоих входных каналов QEP считаются выбранными таймером. Частота входа QEP должна быть меньше либо равной четверти частоты тактового сигнала процессора.

Синхронизация GP таймера

GP таймер 2 и GP таймер 3 могут быть индивидуально синхронизированы с GP таймером 1 посредством корректной конфигурации T2CON и T3CON следующими способами:

– Запуск GP таймера 2 или GP таймера 3 осуществляется тем же разрядом управления T1CON, который запускает GP таймер 1.

– Инициализация счетчиков GP таймера 2 и GP таймер 3 различными значениями перед синхронизированным запуском.

– Установление для GP таймера 2 или GP таймера 3 регистров периода через GP таймер 1, как их собственных регистров периода (игнорируя их собственные регистры периода).

Это позволяет получить требуемую синхронизацию между событиями GP таймеров. Как только каждый GP таймер начинает операцию счета со своего текущего значения в счетном регистре, то GP таймер может быть запрограммирован для запуска с известной задержкой после других GP таймеров. Следует отметить, что для синхронизации GP таймера 1 и GP таймера 2 или GP таймера 2 и GP таймера 3 необходимо произвести две записи в T1CON.

Запуск модуля АЦП с помощью GP таймера

Биты GPTCON могут задавать формирование сигнала запуска модуля АЦП по таким событиям GP таймера как обнуление, совпадение при сравнении или совпадение периода. Это свойство обеспечивает синхронизацию между событиями GP таймера и запуском модуля АЦП без вмешательства ядра процессора.

Работа GP таймера при его приостановке эмулятором

Биты управляющего регистра GP таймера так же определяют работу GP таймера при его приостановке эмулятором. Эти биты могут быть установлены для разрешения продолжения работы GP таймера, делая возможной внутрисхемную эмуляцию при прерывании от эмулятора. Так же они могут быть установлены для прекращения работы GP таймера немедленно или после завершения текущего счётного периода при появлении прерывания эмулятора.

Приостановка работы ИС от внутрикристального эмулятора возникает, когда тактовый сигнал процессора остановлен эмулятором, например, когда эмулятор встречает точку останова.

Прерывания GP таймера

Существует 12 флагов прерываний в EVIFRA и EVIFRB для трёх GP таймеров. Каждый GP таймер может формировать четыре прерывания по следующим событиям:

– Переполнение: TxOFINT (x=1, 2, 3).
– Опустошение: TxUFINT (x=1, 2, 3).
– Совпадение при сравнении: TxCINT (x=1, 2, 3).
– Совпадение периода: TxPINT (x=1, 2, 3).

Событие сравнения (совпадение) таймера происходит, когда содержимое счетчика GP таймера одинаково с содержимым регистра сравнения. Соответству-

ющий флаг прерывания сравнения устанавливается через два периода тактового сигнала после совпадения, если разрешена операция сравнения.

Событие переполнения возникает, когда значение счетчика таймера достигает FFFFh. Событие опустошения возникает, когда значение счетчика таймера достигает 0000h. Так же событие совпадения периода происходит, когда значение счетчика таймера совпадает со значением в регистре периода. Флаги прерываний переполнения, опустошения и совпадения периода таймера устанавливаются через два периода тактового сигнала после появления соответствующего события. Следует отметить, что определения переполнения и опустошения отличаются от общепринятого определения.

Операции счета GP таймера

Каждый GP таймер имеет шесть выбираемых режимов работы:

- Режим остановки/удержания.
- Режим одиночного прямого счета.
- Режим продолжительного прямого счета.
- Режим направленного прямого/обратного счета.
- Режим одиночного прямого/обратного счета.
- Режим продолжительного прямого/обратного счета.

Битовая комбинация соответствующего регистра управления таймером TxCON определяет счётный режим GP таймера. Бит разрешения таймера TxCON[6] разрешает или запрещает счетную операцию таймера. Когда таймер запрещен, счетная операция останавливается и делитель таймера сбрасывается до x/1. Когда таймер разрешен, таймер начинает считать в соответствии со счётным режимом, выбранным другими битами TxCON.

Режим остановки/удержания

Работа GP таймера останавливается, и удерживается текущее состояние. Значения счетчика таймера, выхода сравнения (compare) и масштабирующего счетчика остаются неизменными в этом режиме.

Режим одиночного прямого счета

В этом режиме GP таймер считает в прямом направлении в соответствии с масштабированным входным тактовым сигналом, пока значение счетчика таймера не совпадет со значением в регистре периода. При следующем переднем фронте входного тактового сигнала после совпадения, GP таймер будет сброшен в 0 и заблокирует счетные операции, сбросом бита запрещения таймера, TxCON[6].

Флаг прерывания периода таймера устанавливается через два периода тактового сигнала после совпадения между счетчиком таймера и регистром периода. Флагом формируется запрос прерывания, если прерывание не маскировано и не выполняются никакие другие, немаскированные прерывания с более высоким приоритетом в той же группе. Сигнал запуска модуля АЦП посылается в модуль модуля АЦП в тоже время, когда устанавливается флаг, если прерывание периода этого

таймера было выбрано соответствующими битами в GPTCON для запуска модуля АЦП.

Через два такта после установления GP таймера в 0, устанавливается флаг прерывания опустошения таймера. Флагом формируется запрос прерывания, если прерывание немаскировано и не выполняются никакие другие, немаскированные прерывания с более высоким приоритетом в той же группе. Сигнал запуска модуля АЦП посылается в тоже время, когда флаг прерывания опустошения этого таймера был выбран соответствующими битами в GPTCON для запуска модуля АЦП.

Флаг прерывания переполнения устанавливается через два периода тактового сигнала после достижения TxCNT значения FFFFh. Флагом формируется запрос прерывания, если прерывание немаскировано и не выполняются никакие другие, немаскированные прерывания с более высоким приоритетом в той же группе.

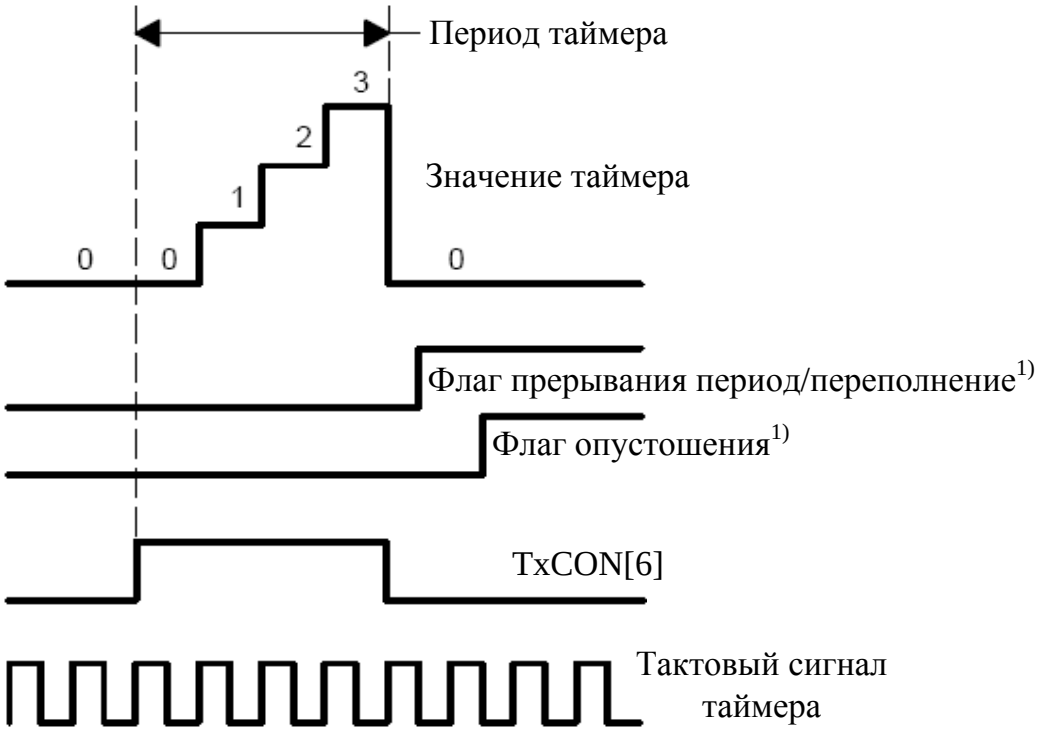
Длительность счетного периода в этом режиме составляет (TxPER)+1 периодов масштабированного входного тактового сигнала, если счетчик таймера содержит 0 в начале периода.

Начальное значение GP таймера может быть любым значением между 0h и FFFFh. Когда начальное значение больше значения регистра периода, таймер досчитает до FFFFh, сбросится до 0, будет считать в прямом направлении опять, чтобы закончить период так, как если бы начальное значение счётчика было 0. Когда начальное значение счетчика таймера одинаково со значением регистра периода, таймер установит флаг прерывания периода, сбросится в 0, установит флаг прерывания опустошения и немедленно завершит период. Если начальное значение таймера находится между 0 и содержимым регистра периода, таймер будет считать в прямом направлении до значения периода и продолжит завершать период так, как если бы начальное значение счётчика было равно значению регистра периода.

После окончания периода, работа GP таймера может быть заново начата только программной записью бита разрешения в таймер, TxCON[6].

Бит направления счёта в GPTCON в этом режиме работы должен быть равен 1 для выбранного таймера. В качестве входного тактового сигнала могут использоваться как внешний сигнал, так и внутренний тактовый сигнал процессора. В этом режиме работы GP таймера вход TMRDIR игнорируется. На рисунке 54 показан режим одиночного прямого счета GP таймера.

Следует отметить, что GP таймер начинает счет сразу же после установления TxCON[6]. Это справедливо для каждого счетного режима.



[1)] Синхронизация флагов прерывания одинакова во всех счётных режимах

Рисунок 54 – Режим одиночного прямого счета GP таймера (TxPR = 4 – 1 = 3)

Операции счета GP таймера

Каждый GP таймер имеет шесть выбираемых режимов работы:

- Режим остановки/удержания.
- Режим одиночного прямого счета.
- Режим продолжительного прямого счета.
- Режим направленного прямого/обратного счета.
- Режим одиночного прямого/обратного счета.
- Режим продолжительного прямого/обратного счета.

Битовая комбинация соответствующего регистра управления таймером TxCON определяет счётный режим GP таймера. Бит разрешения таймера TxCON[6] разрешает или запрещает счетную операцию таймера. Когда таймер запрещен, счетная операция останавливается и делитель таймера сбрасывается до x/1. Когда таймер разрешен, таймер начинает считать в соответствии со счётным режимом, выбранным другими битами TxCON.

Режим остановки/удержания

Работа GP таймера останавливается, и удерживается текущее состояние. Значения счетчика таймера, выхода сравнения (compare) и масштабирующего счетчика остаются неизменными в этом режиме.

В примере 1 показывается программа инициализации режима одиночного прямого счета GP таймера 1.

Пример 1 – Программа инициализации режима одиночного прямого счета GP таймера 1.

```
;*****************************************************************
; Эта часть кода инициализирует GP таймер 1 для запуска в режиме
* одиночного прямого счета. Функция сравнения (compare) GP тай-
мера  также разрешена.
;*****************************************************************
    LDPK #232                    ; DP => Регистры модуля EV
;*****************************************************************
; Настройка GPTCON
;*****************************************************************

    SPLK #0000000001000001b, GPTCON    ;Установка управления GP
                                       ;таймера
;       ||||||||||||||||
;       FEDCBA9876543210
;
* bits 0-1     01:  GP таймер 1 - выход сравнения (compare)
с активным низким уровнем
* bits 2-3     00:  GP таймер 2 - выход сравнения (compare)
с принудительным низким уровнем
* bits 4-5     00:  GP таймер 3 - выход сравнения (compare)
с принудительным низким уровнем
* bit 6        1:   Разрешение выводов сравнения (compare)
                    GP  таймера
* bits 7-8     00:  Нет  событий  запускающих  модуля  АЦП  от  GP
таймера 1
* bits 9-9     00:  Нет  событий  запускающих  модуля  АЦП  от  GP
таймера 2
* bits 10-11   00:  Нет  событий  запускающих  модуля  АЦП  от  GP
таймера 3
;*****************************************************************
; Программирование T1PER и T1CMP
*
;*****************************************************************
    SPLK #05, T1PER     ; Установка периода GP таймера
    SPLK #03, T1CMP     ; Установка сравнения (compare) GP
                        ;таймера
;*****************************************************************
; Настройка T1CON и запуск GP таймер 1                          *
;*****************************************************************
  SPLK #1000100101000010b, T1CON ;Установка управления GP тай-
мер3
;       ||||||||||||||||
;       FEDCBA9876543210
;
* bit 0        0:   Использование собственного периода (PR)
* bit 1        1:   Разрешение сравнения (compare) GP таймера
```

```
* bits 2-3     00:  Загрузка регистра сравнения GP таймера
при опустошении
* bits 4-5     00:  Выбор внутреннего тактового сигнала (CLK)
* bit 6        1:   Разрешение операций счета
* bit 7        0:   Использование собственного разрешения (ENA-
BLE)
* bits 8-10    001: Делитель = /2
* bits 11-13   001: Одиночный прямой счет
* bit 14       0:   SOFT = 0
* bit 15       1:   FREE = 1
```

Режим продолжительного прямого счета

Работа GP таймера в этом режиме похожа на режим одиночного прямого счета, повторяющийся каждый раз, когда таймер сбрасывается до 0. В этом режиме GP таймер считает в прямом направлении, в соответствии с масштабированным входным тактовым сигналом, пока значение счетчика таймера не совпадет со значением в регистре периода. Далее происходит сброс до 0 и начинается другой счётный период.

Длительность счетного периода в этом режиме составляет (TxPER)+1 периодов масштабированного входного тактового сигнала, исключая первый период. Длительность первого периода такова, как если бы счетчик таймера был в 0 при начале счёта.

Начальное значение GP таймера может быть любым значением между 0h и FFFFh. Когда начальное значение больше значения регистра периода, таймер досчитает до FFFFh, сбросится до 0, будет считать в прямом направлении опять, чтобы закончить период, так как если бы начальное значение счётчика было 0. Когда начальное значение счетчика таймера одинаково со значением регистра периода, таймер установит флаг прерывания периода, сбросится до 0, установит флаг прерывания опустошения, а затем продолжит операцию, так как если бы начальное значение счётчика было 0. Если начальное значение таймера находится между 0 и содержимым регистра периода, таймер будет считать в прямом направлении до значения периода и продолжит завершать период, так как если бы начальное значение счётчика было равно значению регистра периода.

Флаги прерываний периода, опустошения и переполнения, прерывания и соответствующие действия формируются по соответственным совпадениям, так же как они формируются в режиме одиночного прямого счета.

Бит направления счёта в GPTCON равен 1 для таймера в этом режиме работы. В качестве входного тактового сигнала может использоваться внешний сигнал или внутренний тактовый сигнал процессора. В этом режиме работы GP таймера вход TMRDIR игнорируется. На рисунке 55 показан режим продолжительного прямого счета GP таймера.

Режим непрерывного прямого счета GP таймера очень важен для формирования запускаемых фронтом или асинхронных сигналов PWM и выборки периодов во многих системах управления перемещением и двигателями.

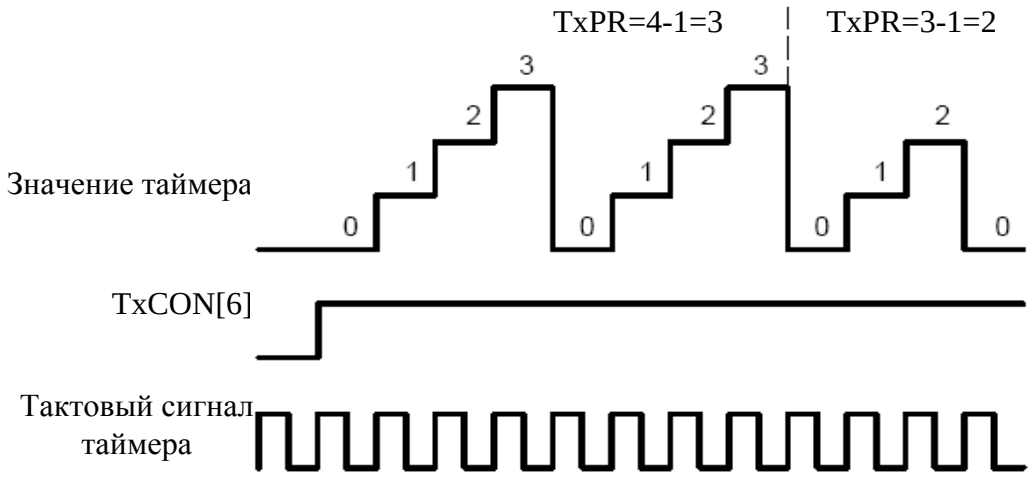


Рисунок 55 – Режим непрерывного прямого счета GP таймера (TxPR = 2 или 3)

Как показано на рисунке 55, ни один тактовый период не потерян с момента достижения таймером значения регистра периода до времени начала следующего счетного периода.

В примере 2 показывается программа инициализации режима продолжительного прямого счета GP таймера 1.

Пример 2 – Программа инициализации режима продолжительного прямого счета GP таймера 1.

```
;*************************************************************
*
; Эта часть кода инициализирует GP таймер 1 для запуска в режиме
продолжительного прямого счета. Функция сравнения (compare) GP
таймера так же разрешена. Начальное значение счетчика FFFEh. *
;*************************************************************

;*************************************************************
    LDPK #232              ; DP => Регистры модуля EV
;*************************************************************
; Программирование T1CON без запуска GP таймера 1 *
;*************************************************************
    SPLK #1000000101000010b, T1CON     ; Установка управления GP
                                       ;таймера 3
;     ||||||||||||||||
;     FEDCBA9876543210
;
* bit 0      0:   Использование собственного периода PR
* bit 1      1:   Разрешение сравнения (compare) GP таймера
* bits 2-3   00:  Загрузка регистра сравнения GP таймера при
                  опустошении
* bits 4-5   00:  Выбор внутреннего тактового сигнала CLK
* bit 6      1:   Разрешение операций счета
* bit 7      0:   Использование собственного разрешения ENABLE
* bits 8-10  001: Делитель = /2
* bits 11-13 000: Режим остановки и удержания
```

```
* bit 14      0:   SOFT = 0
* bit 15      1:   FREE = 1
;*****************************************************************
; Программирование GPTCON
;*****************************************************************
     SPLK #0000000001101010b, GPTCON  ;Установка управления GP
                                      ;таймера
;          ||||||||||||||||
;          FEDCBA9876543210
;
* bits 0-1    10:  GP таймера 1 выход сравнения (compare) с ак-
тивным высоким уровнем
* bits 2-3    10:  GP таймера 2 выход сравнения (compare) с ак-
тивным высоким уровнем
* bits 4-5    10:  GP таймера 3 выход сравнения (compare) с ак-
тивным высоким уровнем
* bit 6       1:   Разрешение выводов сравнения (compare) GP тай-
мера
* bits 7-8    00:   Нет GP таймера 1 событий запускающих модуля
АЦП
* bits 9-10   00:   Нет GP таймера 2 событий запускающих модуля
АЦП
* bits 11-12  00:   Нет GP таймера 3 событий запускающих модуля
АЦП
;*****************************************************************
; Формирование T1PER T1CMP и T1CNT *
;*****************************************************************
    SPLK #05, T1PER    ;Установление периода GP таймера
    SPLK #03, T1CMP    ;Установление сравнения (compare) GP
                       ;таймера
SPLK #0FFFEh, T1CNT     ;Установление счетчика GP таймера
;*****************************************************************
; Запуск GP таймера *
;*****************************************************************

;*****************************************************************
SPLK #1001000101000010b, T1CON  ;Установка управления GP таймера
3
;   ||||||||||||||||
;   FEDCBA9876543210
;
* bit 0       0:   Использование собственного периода PR
* bit 1       1:   Разрешение сравнения (compare) GP таймера
* bits 2-3    00:  Загрузка регистра сравнения GP таймера при
                   опустошении
* bits 4-5    00:  Выбор внутреннего тактового сигнала CLK
* bit 6       1:   Разрешение операций счета
* bit 7       0:   Использование собственного разрешения ENABLE
* bits 8-10   001: Делитель = /2
* bits 11-13  010: Продолжительный прямой счет
* bit 14      0:   SOFT = 0
* bit 15      1:   FREE = 1
```

Режим направленного прямого/обратного счета

В режиме направленного прямого/обратного счета GP-таймер будет считать в прямом или в обратном направлении в соответствии с масштабированным тактовым сигналом и входом TMRDIR. Когда вход TMRDIR удерживается в высоком состоянии, то GP таймер будет считать в прямом направлении, до тех пор пока его значение не достигнет значения, равного значению в регистре периода или FFFFh. Когда значение таймера сравняется со значением его регистра периода или FFFFh, и если вход TMRDIR удерживается в высоком состоянии, то таймер будет удерживаться в этом состоянии. Когда вывод TMRDIR удерживается в низком состоянии, то GP таймер будет считать в обратном направлении до тех пор, пока его значение не достигнет значения 0. Когда значение таймера равно 0, если вход TMRDIR удерживается в низком состоянии, таймер будет удерживаться в состоянии 0.

Начальное значение таймера может быть любым значением между 0000h и FFFFh. Когда начальное значение счетчика таймера больше значения регистра периода, и TMRDIR удерживается в высоком состоянии, таймер досчитает в прямом направлении до FFFFh и будет удерживаться в этом состоянии. Или, если TMRDIR удерживается в низком состоянии, то таймер начнет счет в обратном направлении до тех пор, пока его значение не достигнет значения регистра периода, и продолжит счёт так, как если бы начальное значение счётчика было равно значению регистра периода. Когда начальное значение счетчика таймера одинаково со значением регистра периода, таймер будет удерживаться в этом значении при высоком уровне TMRDIR или будет считать в обратном направлении с этого значения – при низком TMRDIR.

Если флаги прерываний периода, обнуления и переполнения, прерывания и соответствующие действия формируются по соответствующим событиям, так же как они формируются в режиме одиночного прямого счета.

Задержка между изменением TMRDIR и изменением направления счета составляет два периода тактового сигнала процессора после окончания текущего вычисления (то есть после окончания текущего предмасштабированного счетного периода).

Направление счета таймера в этом режиме работы отражается значением соответствующего бита (бит индикации направления) в GPTCON: 1 означает счёт в прямом направлении, 0 означает счёт в обратном направлении. В качестве входного тактового сигнала может использоваться как внешний сигнал, так и внутренний тактовый сигнал процессора. На рисунке 56 показан режим направленного прямого/обратного счета GP таймера.

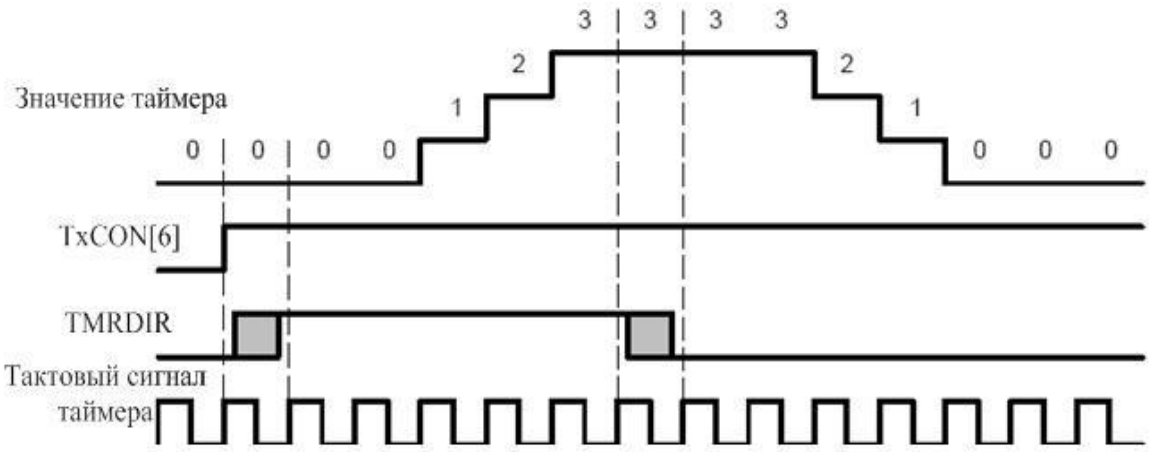


Рисунок 56 – Режим направленного прямого/обратного счета с коэффициентом масштабирования 1 и TxPR = 3

Режим направленного прямого/обратного счета может использоваться с цепями QEP в модуле EV. В этом случае цепи QEP обеспечивают и счётный тактовый сигнал, и задают направление счёта для таймера. Этот режим работы также может быть использован для хронометража появления внешних событий в системах управления перемещением/двигателями и электроникой для управления силовыми приводами. Однако, не произойдёт никакого переключения на выходах сравнения (compare) соответствующих модулей сравнения (включая сравнение GP таймера, полное и простое сравнение), которые используют GP таймер в этом режиме как свой временной масштаб.

На примере 3 показана программа инициализации режима направленного прямого/обратного счета GP таймера 1.

Пример 3 – Программа инициализации режима направленного прямого/обратного счета GP таймера 1.

```
;*****************************************************************
; Эта часть кода инициализирует GP таймер 1 для запуска в режиме
направленного прямого/обратного счета. Функция сравнения
(compare) GP таймера так же разрешена, однако это не оказывает
никакого действия на выводы сравнения (compare). *
;*****************************************************************
LDPK #232                          ; DP => Регистры модуля EV
;*****************************************************************
; Программирование GPTCON
;*****************************************************************
;*****************************************************************
    SPLK #0000000001101010b, GPTCON  ;Установка управления GP
                                          ;таймера
    ;          ||||||||||||||||
    ;          FEDCBA9876543210
;
* bits 0-1    10: GP таймер 1 - выход сравнения (compare) с ак-
тивным высоким уровнем
```

```
* bits 2-3    10: GP таймер 2 - выход сравнения (compare) с активным высоким уровнем
* bits 4-5    10: GP таймер 3 - выход сравнения (compare) с активным высоким уровнем
* bit 6       1:  Разрешение выводов сравнения (compare) GP таймера
* bits 7-8    00: Нет событий запускающих модуля АЦП от GP таймера 1
* bits 9-10   00: Нет событий запускающих модуля АЦП от GP таймера 2
* bits 11-12 00: Нет событий запускающих модуля АЦП от GP таймера 3
;*****************************************************************
; Программирование T1PER и T1CMP *
;*****************************************************************
    SPLK #05, T1PER ; Установка периода GP таймера
    SPLK #03, T1CMP ; Установка сравнения (compare) GP таймера
;*****************************************************************
; Запуск GP таймера *
;*****************************************************************
   SPLK #1001100001010010b, T1CON  ;Установка управления GP
                                                ;таймера 3
   ;          ||||||||||||||||
   ;          FEDCBA9876543210
;
* bit 0      0:   Использование собственного периода PR
* bit 1      1:   Разрешение сравнения (compare) GP таймера
* bits 2-3   00:  Загрузка регистра сравнения GP таймера при
                  опустошении
* bits 4-5   00:  Выбор внутреннего тактового сигнала CLK
* bit 6      1:   Разрешение операций счета
* bit 7      0:   Использование собственного разрешения ENABLE
* bits 8-10  001: Делитель = /1
* bits 11-13 010: Направленный прямой/обратный счёт
* bit 14     0:   SOFT = 0
* bit 15     1:   FREE = 1
```

Режим одиночного прямого/обратного счета

В этом режиме GP таймер считает в прямом направлении в соответствии с масштабированным входным тактовым сигналом до значения в его регистре периода. Затем он изменит своё направление счёта на обратное, пока его значение не достигнет 0. Когда таймер достигает 0, он сбрасывает TxCON[6] и свой масштабирующий счетчик, останавливает счёт и удерживается в текущем состоянии.

Длительность периода в этом режиме составляет 2×(TxPR) периодов масштабированного входного тактового сигнала, если значение счетчика таймера равно 0.

Начальное значение GP таймера может быть любым значением между 0h и FFFFh. Когда начальное значение больше значения регистра периода, таймер досчитает до FFFFh, сбросится до 0 и продолжит счёт, так как если бы начальное значение счётчика было 0. Когда начальное значение счетчика таймера одинаково со

значением регистра периода, таймер досчитает до 0 и завершит период. Если начальное значение таймера находится между 0 и содержимым регистра периода, таймер будет считать в прямом направлении до значения периода и продолжит завершать период, так как если бы начальное значение счётчика было равно значению регистра периода.

Флаги прерываний периода, обнуления и переполнения устанавливаются, а прерывания и связанные с ними действия формируются по соответствующим событиям, подобные тому, как они формируются в режиме одиночного прямого счета. Следует отметить, однако, что событие периода возникает, когда происходит совпадение между счетчиком таймера и регистром периода, т. е. в середине счетного периода.

После завершения работы в этом режиме GP таймера его очередной запуск может быть осуществлен только программной записью 1 в TxCON[6]. Бит индикации направления в GPTCON равен 1 при счёте в прямом направлении и равен 0 при счёте в обратном направлении. В качестве входного тактового сигнала может использоваться как внешний сигнал, так и внутренний тактовый сигнал процессора. В этом режиме работы GP таймера вход TMRDIR игнорируется.

На рисунке 57 показан режим одиночного прямого/обратного счета GP таймера. На примере 4 показывается программа инициализации режима одиночного прямого/обратного счета GP таймера 1.

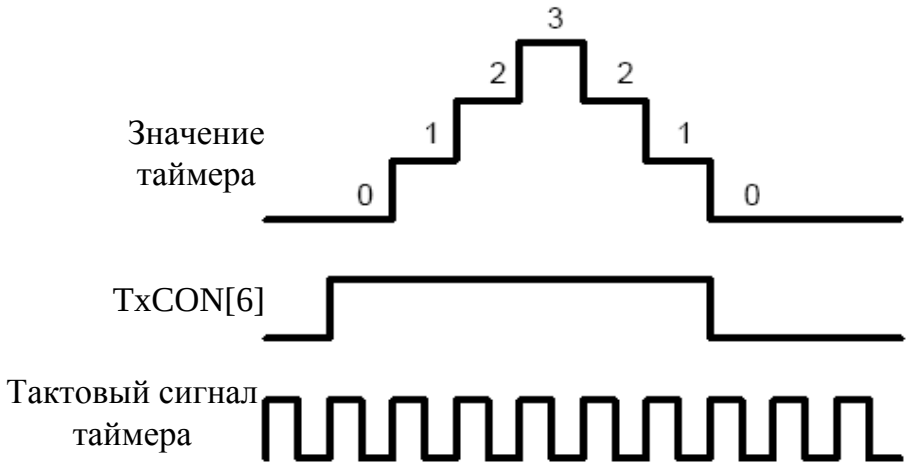


Рисунок 57 – Режим одиночного прямого/обратного счета (TxPR = 3)

Пример 4 – Программа инициализации режима одиночного прямого/обратного счета GP таймера 1.

```
;*************************************************************
; Эта часть кода инициализирует GP таймер 1 для запуска в режиме
одиночного прямого/обратного счета. Функция сравнения (compare)
GP таймера так же разрешена. *
;*************************************************************
LDPK #232              ; DP => Регистры модуля EV
;*************************************************************

;*************************************************************
```

```
; Формирование GPTCON
;*************************************************************
  SPLK #0000000001000001b, GPTCON ;Установка управления GP тай-
мера
;       ||||||||||||||||
;       FEDCBA9876543210
;
* bits 0-1    01: GP таймер 1 выход сравнения (compare) с актив-
ным низким уровнем
* bits 2-3    00: GP таймер 2 выход сравнения (compare) с прину-
дительным низким уровнем
* bits 4-5    00: GP таймер 3 выход сравнения (compar)с  принуди-
тельным низким уровнем
* bit 6        1:  Разрешение выводов сравнения (compare) GP тай-
мера
* bits 7-8    00: Нет событий запускающих модуля АЦП от GP тайме-
ра 1
* bits 9-9    00: Нет событий запускающих модуля АЦП от GP тайме-
ра 2
* bits 10-11 00: Нет событий запускающих модуля АЦП от GP тайме-
ра 3
;*************************************************************
; Формирование T1PER и T1CMP *
;*************************************************************
    SPLK #05, T1PER      ;Установление периода GP таймера
    SPLK #03, T1CMP      ;Установление сравнения (compare) GP
                         ;таймера
;*************************************************************
; Формирование T1CON и запуск GP таймера 1 *
;*************************************************************
    SPLK #1000100101000010b, T1CON  ;Установка управления GP
                                          ;таймера 3
    ;          ||||||||||||||||
    ;          FEDCBA9876543210
;
* bit 0      0:   Использование собственного периода PR
* bit 1      1:   Разрешение сравнения (compare) GP таймера
* bits 2-3   00:  Загрузка регистра сравнения GP таймера при
                  опустошении
* bits 4-5   00:  Выбор внутреннего тактового сигнала CLK
* bit 6      1:   Разрешение операций счета
* bit 7      0:   Использование собственного разрешения ENABLE
* bits 8-10  001: Делитель = /2
* bits 11-13 001: Одиночный прямой счет
* bit 14     0:   SOFT = 0
* bit 15     1:   FREE = 1
```

Режим продолжительного прямого/обратного счета

Работа GP таймера в этом режиме похожа на режим одиночного прямого/обратного счета, повторяющийся каждый раз, когда таймер сбрасывается до 0.

Как только запущена работа в этом режиме, не требуется никакого аппаратного или программного вмешательства для повторения счетного периода.

Длительность периода в этом режиме составляет 2×(TxPR) периодов масштабированного входного тактового сигнала, исключая первый период. Длительность первого периода такова, как если бы счетчик таймера был в 0 при начале счёта.

Начальное значение GP таймера может быть любым значением между 0h и FFFFh. Когда начальное значение больше значения регистра периода, таймер досчитает до FFFFh, сбросится до 0, будет считать в прямом направлении опять, чтобы закончить период, так как если бы начальное значение счётчика было 0. Когда начальное значение счетчика таймера одинаково со значением регистра периода, таймер досчитает до 0, а затем продолжит операцию, так как если бы начальное значение счётчика было 0. Если начальное значение таймера находится между 0 и содержимым регистра периода, таймер будет считать в прямом направлении до значения периода и продолжит завершать период, так как если бы начальное значение счётчика было равно значению регистра периода.

Флаги прерываний периода, обнуления и переполнения, а также прерывания и связанные с ними действия формируются по соответствующим событиям подобно тому, как они формируются в режиме одиночного прямого счета.

Для этого режима работы таймера бит направления счёта в GPTCON равен 1 при прямом направлении счета и 0 при обратном направлении счета. В качестве входного тактового сигнала может использоваться как внешний сигнал, так и внутренний тактовый сигнал процессора. В этом режиме работы таймера ввод TMRDIR игнорируется. На рисунке 58 показан режим непрерывного прямого/обратного счета GP таймера.

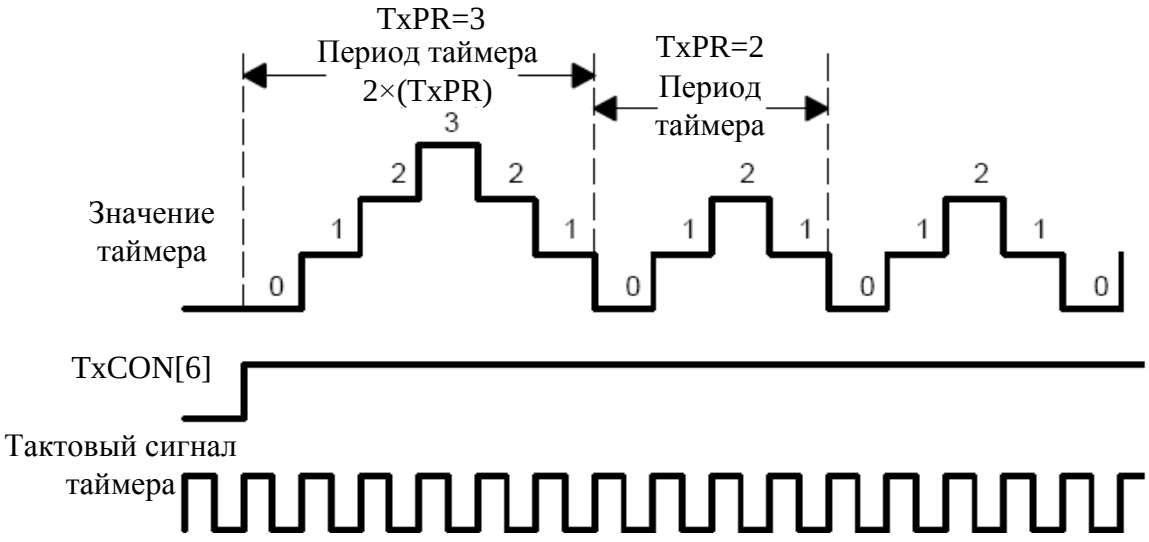


Рисунок 58 – Режим продолжительного прямого/обратного счета GP таймера (TxPR = 3 или 2)

Режим продолжительного прямого/обратного счета очень важен для формирования центрированных или симметричных сигналов PWM, получаемых во многих системах управления перемещением/двигателями и электроникой для управления силовыми приводами.

В примере 5 показывается программа инициализации режима непрерывного прямого/обратного счета GP таймера 1.

Пример 5 – Программа инициализации режима продолжительного прямого/обратного счета GP таймера 1.

```
;*************************************************************
; Эта часть кода инициализирует GP таймера 1 для запуска в режи-
ме продолжительного прямого/обратного счета. Функция сравнения
GP таймера так же разрешена. *
;*************************************************************
    LDPK #232              ; DP => Регистры модуля EV
;*************************************************************
; Формирование GPTCON
;*************************************************************
   SPLK #0000000000111111b, GPTCON ;Установка управления GP тай-
мера
    ;       ||||||||||||||
    ;       FEDCBA9876543210
;
* bits 0-1   01: GP таймера 1 выход сравнения (compare) с
                принудительным высоким уровнем
* bits 2-3   00: GP таймера 2 выход сравнения (compare) с
                принудительным высоким уровнем
* bits 4-5   00: GP таймер 3 выход сравнения (compare) с
                принудительным высоким уровнем
* bit 6       1:  Разрешение выводов сравнения (compare) GP тай-
мера
* bits 7-8   00: Нет событий запускающих модуля АЦП от GP тайме-
ра 1
* bits 9-9   00: Нет событий запускающих модуля АЦП от GP тайме-
ра 2
* bits 10-11 00: Нет событий запускающих модуля АЦП от GP тайме-
ра 3
;*************************************************************
; Формирование T1PER и T1CMP *
;*************************************************************
    SPLK #05, T1PER     ; Установка периода GP таймера
    SPLK #03, T1CMP     ; Установление сравнения (compare) GP
                        ;таймера
;*************************************************************
; Формирование T1CON и запуск GP таймера 1 *
;*************************************************************
    SPLK #1010100001000000b, T1CON  ;Установка управления GP
                                          ;таймера 3
    ;          ||||||||||||||||
    ;          FEDCBA9876543210
;
* bit 0  0:   Использование собственного периода PR
* bit 1  0:   Запрещение сравнения(compare) GP таймером
* bits 2-3   00: Загрузка регистра сравнения GP таймера при об-
нулении
```

```
* bits 4-5   00:  Выбор внутреннего тактового сигнала CLK
* bit 6  1:   Разрешение операций счета
* bit 7  0:   Использование собственного разрешения ENABLE
* bits 8-10  000: Делитель = /1
* bits 11-13  010: Продолжительный прямой/обратный счет
* bit 14 0:   SOFT = 0
* bit 15 1:   FREE = 1
```

Операции сравнения (compare) GP таймером

Каждый GP таймер имеет соответствующий регистр сравнения TxCMPR и вывод сравнения (compare)/PWM TxPWM/TxCMP. Значение счетчика GP таймера постоянно сравнивается со значением в соответствующем регистре сравнения.

Совпадение при сравнении происходит, когда значение счетчика таймера совпадает со значением в регистре сравнения. Операция сравнения разрешается установкой в 1 бита TxCON[1]. Если операция разрешена, то при совпадении при сравнении происходит следующее:

– флаг прерывания сравнения этого таймера устанавливается через два тактовых периода после совпадения;

– если GP таймер находится не в режиме прямого/обратного счёта, переключение происходит на соответствующем выходе сравнения(compare)/PWM в соответствии с конфигурацией бит в GPTCON, через один тактовый период после совпадения;

– если флаг прерывания сравнения был выбран соответствующими битами GPTCON для запуска модуля АЦП, сигнал запуска модуля АЦП формируется одновременно с установлением флага прерывания.

Запрос прерывания к ядру формируется флагом прерывания сравнения, если прерывание не маскировано и не выполняются никакие другие, немаскированные прерывания с более высоким приоритетом в той же группе.

Переключение сравнения (compare)/PWM

Выход переключения сравнения (compare)/PWM управляется генератором асимметричных и симметричных сигналов и соответствующей выходной логикой, и зависит от следующего:

– состояния битов в GPTCON;

– в каком счетном режиме находится таймер;

– направления счёта, когда используется режим одиночного или продолжительного прямого/обратного счета.

Генератор асимметричных/симметричных сигналов

Генератор асимметричных/симметричных сигналов формирует асимметричные или симметричные сигналы сравнения (compare)/PWM, основанные на том, в каком режиме счёта находится GP таймер.

Формирование асимметричных сигналов

Асимметричные сигналы (см. рисунок 59) формируются, когда GP таймер находится в режимах одиночного или продолжительного прямого/обратного счета. Когда GP таймер находится в этих двух режимах, выход генератора сигналов изменяется в следующем порядке:

– 0 перед началом операции счёта;

– остается неизменным пока не произойдет совпадение при сравнении;

– переключается в момент совпадения при сравнении;

– остается неизменным до конца периода;

– сбрасывается до 0 в конце периода при совпадении периодов, если новое значение сравнения для этого периода не равно 0.

Выход будет равен 1 в течение всего периода, если значение сравнения 0 в начале периода. Выход не будет сброшен до 0, если новое значение сравнения для этого периода равно 0. Это важно, потому что это позволяет формировать PWM сигналы с (0 - 100) % коэффициентом заполнения без сбоев. Выход равен 0 в течение всего периода, если значение сравнения больше значения регистра периода. Выход будет равен 1 в течение одного периода масштабированного входного тактового сигнала, если компарируемая величина равна значению в регистре периода.

Характеристикой асимметричных сигналов сравнения(compare)/PWM является то, что изменения в значении регистра сравнения оказывают влияние только на один фронт сигнала сравнения (compare)/PWM.

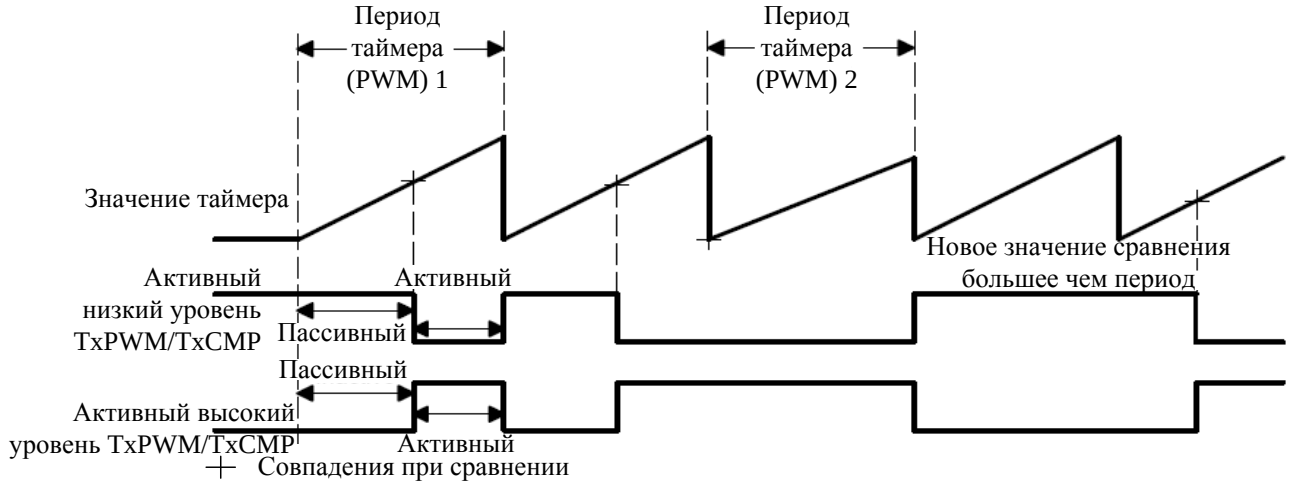


Рисунок 59 – Выход сравнения (compare)/PWM в режиме прямого счёта GP таймера

Формирование симметричных сигналов

Симметричные сигналы (см. рисунок 60) формируются, когда GP таймер находится в режимах одиночного или продолжительного прямого/обратного счета. Когда GP таймер находится в этих двух режимах, выход генератора сигналов изменяется в следующем порядке:

– 0 перед началом операции счёта;

– остается неизменным пока не произойдет первое совпадение при сравнении;

– переключается в момент первого совпадения при сравнении;

– остается неизменным до следующего совпадения при сравнении;

– переключается в момент следующего совпадения при сравнении;

– остается неизменным до конца периода;

– сбрасывается до 0 в конце периода при совпадении периодов, если новое значение сравнения для этого периода не равно 0.

Выход устанавливается в 1 в начале периода и будет оставаться 1 до следующего совпадения при сравнении, если значение сравнения было 0 в начале периода. После первого переключения выход будет оставаться неизменным до конца периода, если значение сравнения будет 0 до следующей половины периода. Когда это происходит, выход не будет сброшен до 0, если новое значение сравнения для этого периода всё ещё равно 0. Это делается для обеспечения формирования PWM сигналов с (0 – 100) % коэффициентом заполнения без сбоев. Первое переключение не произойдёт, если значение сравнения больше или равно значению в регистре периода в первую половину периода. Эта ошибка выходного переключения, часто являющаяся результатом ошибки вычислений в подпрограмме, будет исправлена в конце периода потому, что выход будет сброшен до 0, пока новое значение сравнения для этого периода не станет 0; в этом случае выход останется 1, что снова установит генератор сигналов в корректное состояние.

Примечание – Выходная логика определяет полярность всех выходов.

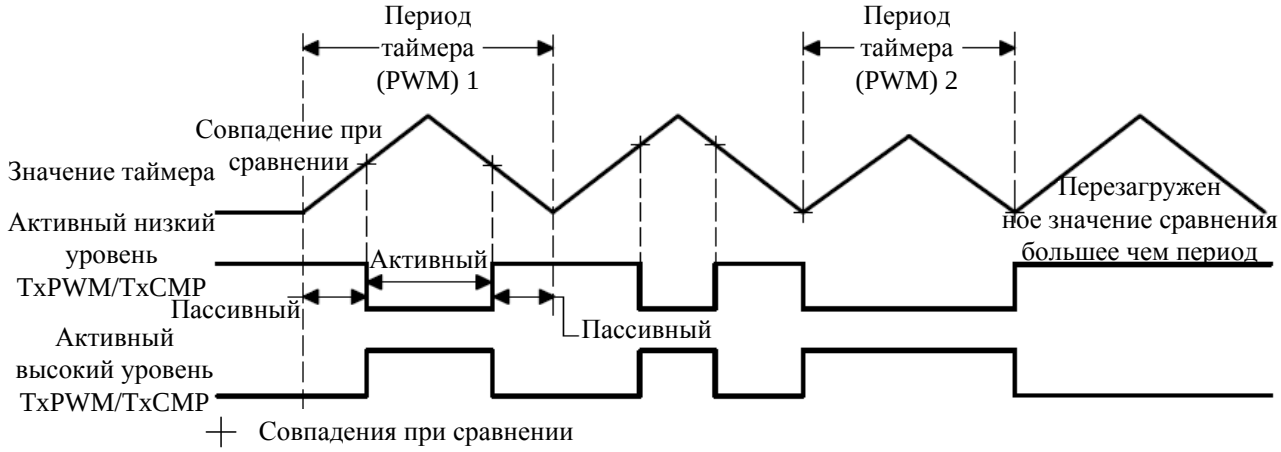


Рисунок 60 – Выход сравнения (compare)/PWM в режиме прямого/обратного счёта GP таймера

Выходная логика

Наличие выходной логики является дальнейшим условием для формирования конечных выходов сравнения (compare)/PWM в генераторе сигналов для управления различными типами силовых устройств. Выходы сравнения (compare)/PWM могут быть определены как установленные в активный высокий

уровень, активный низкий уровень, принудительный высокий уровень, принудительный низкий уровень в зависимости от состояния битов в GPTCON.

Полярность выходов сравнения (compare)/PWM одинакова с полярностью соответствующих выходов генератора асимметричных/симметричных сигналов, когда выход сравнения (compare)/PWM установлен в активный высокий уровень.

Полярность выходов сравнения (compare)/PWM противоположна полярности на соответствующих выходах генератора асимметричных/симметричных сигналов, когда выход сравнения (compare)/PWM установлен в активный низкий уровень.

Выход сравнения (compare)/PWM устанавливается в 1 (или 0) сразу после того, как установятся соответствующие биты в GPTCON, и последовательность бит определит принудительный высокий (низкий) уровень выхода сравнения (compare)/PWM.

В итоге, при нормальном режиме счёта (при условии, что сравнение разрешено) переключение выхода сравнения (compare)/PWM GP таймера для одиночного и непрерывного вариантов прямого счета происходит в соответствии с таблицей 58. Для одиночного и непрерывного прямого/обратного вариантов – в соответствии с таблицей 59.

Установка высокого активного уровня – устанавливает активный уровень высоким, а установка низкого уровня – устанавливает активный уровень низкий. Установка низкого активного уровня – устанавливает активные уровни с точностью до наоборот.

Формирование асимметричных/симметричных сигналов, основанное на счётном режиме таймера и выходной логике, так же применимо для блоков полного и простого сравнения.

Таблица 58 – Выход сравнения (compare) GP таймера в режимах одиночного прямого и продолжительного прямого счета

Время в периоде	Состояние выхода сравнения (compare)
Перед совпадением при сравнении	Пассивный уровень
При совпадении при сравнении	Установление активного уровня
При совпадении периода	Установление пассивного уровня

Таблица 59 – Выход сравнения (compare) GP таймера в режимах одиночного прямого/обратного и продолжительного прямого/обратного счета

Время в периоде	Состояние выхода сравнения (compare)
Перед первым совпадением при сравнении	Пассивный уровень
При первом совпадении при сравнении	Установление активного уровня
При втором совпадении при сравнении	Установление пассивного уровня
После второго совпадения при сравнении	Пассивный уровень

Все выходы сравнения (compare)/PWM устанавливаются в высокоимпедансное состояние, когда происходит одно из следующих событий:

- GPTCON[6] программно установлен в 0;
- PDPINT в низком уровне и не маскировано;
- происходит какое либо событие сброса.

Выход сравнения в режиме направленного прямого/обратного счета

Когда GP таймер находится в режиме направленного прямого/обратного счета, на его выходе сравнения не происходит никаких переключений. Также не происходит переключений на выходах сравнения (compare), связанных с блоками полного сравнения, когда GP таймер 1 в режиме направленного прямого/обратного счета; не происходит переключений на выходах сравнения (compare) связанных с блоками простого сравнения, когда GP таймер выбран как временная ось для блоков простого сравнения в режиме направленного прямого/обратного счета. Установление флагов прерывания сравнения и формирование запросов прерывания сравнения не влияют на то, в каком счётном режиме находится GP таймер.

Вычисление активного/пассивного счёта времени

Для режимов прямого счёта значение регистра сравнения представляет собой время, прошедшее между началом периода и появлением первого совпадения при сравнении, что является длиной пассивной фазы. Это время равно периоду масштабированного тактового сигнала с коэффициентом, содержащимся в TxCMPR. Поэтому длина активной фазы (ширина выходного сигнала) получается из (TxPR) - (TxCMPR) + 1 циклов масштабированного входного тактового сигнала.

Для режимов прямого/обратного счёта регистр сравнения может иметь различное значение - от значения при обратном счете, до значения при прямом счете. Длина активной фазы, то есть, ширина выходного импульса для режимов прямого/обратного счёта, таким образом, получается из (TxPR) - $(TxCMPR)_{up}$ + (TxPR) - $(TxCMPR)_{dn}$ циклов масштабированного входного тактового сигнала, где $(TxCMPR)_{up}$ значение сравнения при прямом счете и $(TxCMPR)_{dn}$ значение сравнения при обратном счете.

Когда значение в TxCMPR равно 0, выход сравнения (compare) GP таймера будет активным в течение всего периода, если таймер находится в прямом счётном режиме. Для режимов прямого/обратного счёта выход сравнения (compare) будет активным в начале периода, если $(TxCMPR)_{up}$ равен 0. Выход останется активным до конца периода, если $(TxCMPR)_{dn}$ так же равен 0.

Длина активной фазы (ширина выходного импульса) будет равна 0, когда значение TxCMPR больше, чем значение TxPR, для прямых счётных режимов. Для режимов прямого/обратного счёта, первое переключение будет потеряно, когда значение $(TxCMPR)_{up}$ больше или равно значению (TxPR). Также, следующее переключение будет потеряно, когда значение $(TxCMPR)_{dn}$ больше или равно значению (TxPR). Выход сравнения (compare) GP таймера будет пассивным в течение всего периода, если оба значения $(TxCMPR)_{up}$ и $(TxCMPR)_{dn}$ больше или равны значению (TxPR) для режимов прямого/обратного счёта.

На рисунке 59 показана операция сравнения GP таймера в режимах прямого счёта. На рисунке 60 показана операция сравнения GP таймера в режиме прямого/обратного счёта.

Управляющие регистры GP таймера (TxCON и GPTCON)

Адреса регистров даны в таблице 54. Описание битов управляющих регистров GP таймера TxCON и GPTCON показано на рисунках 61 и 62 соответственно.

Управляющий регистр GP таймера TxCON (x = 1, 2 и 3)

15	14	13	12	11	10	9	8
Free	Soft	TMODE2	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
TSWT1	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 61 – Управляющий регистр GP таймера TxCON (x = 1, 2 и 3)

Биты 15-14 Free, Soft. Биты контроля эмуляции
00 = Немедленная остановка при временной остановке для эмуляции
01 = Остановка после завершения текущего периода таймера при временной остановке для эмуляции
10 = Временная остановка для эмуляции не влияет на работу
11 = Временная остановка для эмуляции не влияет на работу

Биты 13-11 TMODE2 – TMODE0. Выбор режима счёта
000 = Остановка/удержание
001 = Режим одиночного прямого счета
010 = Режим продолжительного прямого счета
011 = Режим направленного прямого/обратного счета
100 = Режим одиночного прямого/обратного счета
101 = Режим продолжительного прямого/обратного счета
110 = Зарезервировано. Результат непредсказуем
111 = Зарезервировано. Результат непредсказуем

Биты 10-8 TPS2 – TPS0. Делитель частоты входного тактового сигнала
000 = x/1
001 = x/2
010 = x/4
011 = x/8
100 = x/16
101 = x/32
110 = x/64
111 = x/128
x = частота тактового сигнала процессора

Бит 7 TSWT1. Запуск таймера с помощью бита разрешения GP таймера 1. Этот бит зарезервирован в T1CON.
0 = Использование собственного бита TENABLE.
1 = Использование бита TENABLE, содержащегося в T1CON, для

разрешения и запрещения работы, игнорируя собственный бит TENABLE.

Бит 6 TENABLE. Разрешение таймера. В режимах одиночного прямого и прямого/обратного счета этот бит устанавливается таймером в 0 после завершения периода работы.
0 = Запрещение работы таймера, таймер устанавливается в режим удержания и счетчик делителя сбрасывается.
1 = Разрешение работы таймера.

Биты 5-4 TCLKS1, TCLKS0. Выбор источника тактового сигнала
00 = Внутренний
01 = Внешний
10 = Переполнение от GP таймера 2. Применимо только для T3CON, когда GP таймер 2 и GP таймер 3 соединены каскадно в 32-битный таймер; зарезервировано в T1CON и T2CON; ILLEGAL, если SELT1PR=1.
11 = Цепь QEP. Применимо только для T2CON и T3CON; зарезервировано в T1CON; ILLEGAL, если SELT1PR=1.
ILLEGAL означает непредсказуемость результата.

Биты 3-2 TCLD1, TCLD0. Условия перезагрузки активного регистра сравнения таймера
00 = Когда значение счетчика равно 0
01 = Когда значение счетчика равно 0 или равно значению регистра периода
10 = Немедленно
11 = Зарезервировано

Бит 1 TECMPR. Разрешение сравнения таймера
0 = Запрещение операций сравнения таймера
1 = Разрешение операций сравнения таймера

Бит 0 SELT1PR. Выбор регистра периода. Этот бит зарезервирован в T1CON.
0 = Использование собственного регистра периода
1 = Использование T1PR как регистр периода, игнорируя собственный регистр периода.

Примечание – Синхронизация нескольких GP таймеров. Требуются две последовательные записи в T1CON для обеспечения синхронизации нескольких GP таймеров, когда T1CON[6] используется для разрешения GP таймера 2 или таймера 3:

1 Формирование всех остальных бит установкой T1CON[6] в 0.

2 Разрешение GP таймер1 и, соответственно, разрешение GP таймера 2 или GP таймера 2 и GP таймера 3 установкой T1CON[6] в 1.

Управляющий регистр GP таймера (GPTCON)

15	14	13	12-11	10-9	8-7
T3STAT	T2STAT	T1STAT	T3TOADC	T2TOADC	T1TOADC
R-1	R-1	R-1	RW-0	RW-0	RW-0

6	5-4	3-2	1-0
TCOMPOE	T3PIN	T2PIN	T1PIN
RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -n – значение после сброса

Рисунок 62 – Управляющий регистр GP таймера (GPTCON) – адрес 7400h

Бит 15 T3STAT. Состояние GP таймера 3. Только чтение
0 = Счет в обратном направлении
1 = Счет в прямом направлении

Бит 14 T2STAT. Состояние GP таймера 2. Только чтение
0 = Счет в обратном направлении
1 = Счет в прямом направлении

Бит 13 T1STAT. Состояние GP таймера 1. Только чтение
0 = Счет в обратном направлении
1 = Счет в прямом направлении

Биты 12-11 T3TOADC. Запуск модуля АЦП событием GP таймера 3.
00 = Нет событий запускающих модуля АЦП
01 = Установка флага прерывания при обнулении. Запуск модуля АЦП.
10 = Установка флага прерывания для периода. Запуск модуля АЦП.
11 = Установка флага прерывания при сравнении. Запуск модуля АЦП.

Биты 10-9 T2TOADC. Запуск модуля АЦП событием GP таймера 2.
00 = Нет событий запускающих модуля АЦП
01 = Установка флага прерывания при обнулении. Запуск модуля АЦП.
10 = Установка флага прерывания для периода. Запуск модуля АЦП.
11 = Установка флага прерывания при сравнении. Запуск модуля АЦП.

Биты 8-7 T1TOADC. Запуск модуля АЦП событием GP таймера 1.
00 = Нет событий запускающих модуля АЦП
01 = Установка флага прерывания при обнулении. Запуск модуля АЦП.
10 = Установка флага прерывания для периода. Запуск модуля АЦП.
11 = Установка флага прерывания при сравнении. Запуск модуля АЦП.

Бит 6 TCOMPOE. Разрешение выхода сравнения (compare). Активный PDPINT записывает 0 в этот бит

0 = Запрещение выходов сравнения (compare) всех трёх GP таймеров (все три выхода сравнения (compare) устанавливаются в высокоимпедансное состояние).
1 = Разрешение выходов сравнения (compare) всех трёх GP таймеров

Биты 5-4 T3PIN. Полярность выходов сравнения (compare) GP таймера 3.
00 = Принудительный низкий уровень
01 = Активный низкий уровень
10 = Активный высокий уровень
11 = Принудительный высокий уровень

Биты 3-2 T2PIN. Полярность выходов сравнения (compare) GP таймера 2.
00 = Принудительный низкий уровень
01 = Активный низкий уровень
10 = Активный высокий уровень
11 = Принудительный высокий уровень

Биты 1-0 T1PIN. Полярность выходов сравнения (compare) GP таймера 1.
00 = Принудительный низкий уровень
01 = Активный низкий уровень
10 = Активный высокий уровень
11 = Принудительный высокий уровень

Формирование выходов сравнения (compare) и PWM с помощью GP таймера

Каждый GP таймер может быть независимо использован для обеспечения выходного канала сравнения (compare) или PWM. Таким образом, GP таймер может формировать до трёх выходов сравнения (compare) или PWM.

Операция сравнения (compare)

Для формирования выхода сравнения (compare) должен быть выбран соответствующий режим работы GP таймера. Эта процедура заключается в следующем:

– Установление TxCMPR в соответствии с тем, когда ожидается возникновение события сравнения.

– Установление GPTCON из условия, чтобы произошло требуемое переключение на выходе сравнения (compare) при совпадении при сравнении.

– Загрузка TxPR с требуемым значением периода, если нужно.

– Загрузка TxCNT с требуемым начальным значением счетчика, если нужно.

– Установление TxCON для определения счётного режима и источника тактового сигнала и запуска работы.

Операции PWM

Для формирования выхода PWM с помощью GP таймера может быть выбран продолжительный прямой или прямой/обратный режим счета. Запускаемые фронтом или асимметричные PWM сигналы формируются, когда выбран режим продолжительного прямого счёта. Центрированные или симметричные PWM сигналы формируются, когда выбран режим продолжительного прямого/обратного счёта. Для установки GP таймера для этой операции нужно произвести следующие операции:

– Установка TxPR в соответствии с требуемым периодом (несущей) PWM.

– Установка TxCON для задания счетного режима и источника тактового сигнала, и запуск работы.

– Загрузка TxCMPR значениями, соответствующими рассчитываемой в режиме онлайн ширине (коэффициенту заполнения) PWM сигналов.

Значение периода получено делением требуемого периода PWM на период входного тактового сигнала GP таймера, и вычитанием единицы из полученного значения, когда для формирования асимметричных PWM сигналов выбран режим продолжительного прямого счёта. Когда для формирования симметричных PWM сигналов выбран режим продолжительного прямого/обратного счёта, это значение получается делением дважды требуемого периода PWM на период входного тактового сигнала GP .

Во время выполнения этих операций регистр сравнения GP таймера постоянно обновляется вновь определенными значениями сравнения в соответствии с вновь определенными коэффициентами заполнения.

На рисунках 63 и 64 показаны примеры асимметричных и симметричных PWM сигналов, которые могут формироваться GP таймером.

Сброс GP таймера

Когда появляется событие сброса, происходит следующее:

– Все биты регистра GP таймера, исключая бит индикации направления счёта GPTCON, сбрасываются в 0; таким образом, работа всех GP таймеров запрещена. Все биты индикации направления счёта устанавливаются в 1.

– Все флаги прерываний сбрасываются в 0.

– Все биты масок прерываний таймера сбрасываются в 0; таким образом, все прерывания от GP таймера маскированы.

– Все выходы сравнения (compare) GP таймера устанавливаются в высокоимпедансное состояние.

13.2.4 Блоки сравнения

В модуле модуля EV существуют три блока полного сравнения и три блока простого сравнения. Каждый блок полного сравнения имеет два соответствующих выхода сравнения(compare)/PWM. Каждый блок простого сравнения имеет один соответствующий выход сравнения(compare)/PWM. Временная ось для блоков полного сравнения обеспечивается GP таймером 1. Временная ось для блоков простого сравнения обеспечивается GP таймером 1 или GP таймером 2.

Блоки простого сравнения

Три блока простого сравнения содержат:

– Три 16-битных регистра сравнения (SCMPRx, x = 1, 2, 3), все с соответствующим теневым регистром, (R/W).

– Один регистр управления сравнением (COMCON), общий с блоками полного сравнения, (R/W).

– Один 16-битный операционный регистр управления (SACTR), с соответствующим теневым регистром, (R/W).

– Три генератора асимметричных/симметричных сигналов.

– Три выхода сравнения (compare)/PWM (третье состояние), PWMy/CMPy (y = 7, 8, 9) один на каждый блок простого сравнения, с программируемой полярностью.

– Логику сравнения и прерывания.

Работа трёх блоков простого сравнения одинакова с операцией сравнения GP таймера, исключая следующее:

– Временная ось для блоков простого сравнения обеспечивается GP таймер 1 или GP таймером 2.

– Разрешение и запрещение операции простого сравнения, разрешение и запрещение выходов простого сравнения, состояние, когда (активный) регистр простого сравнения обновлён, и выбор временной оси для операции простого сравнения управляются соответствующими битами в COMCON.

– Поведение выходов сравнения блоков простого сравнения описывается индивидуально соответственными битами в операционном управляющем регистре простого сравнения SACTR.

На рисунке 63 показана структурная схема блоков простого сравнения.

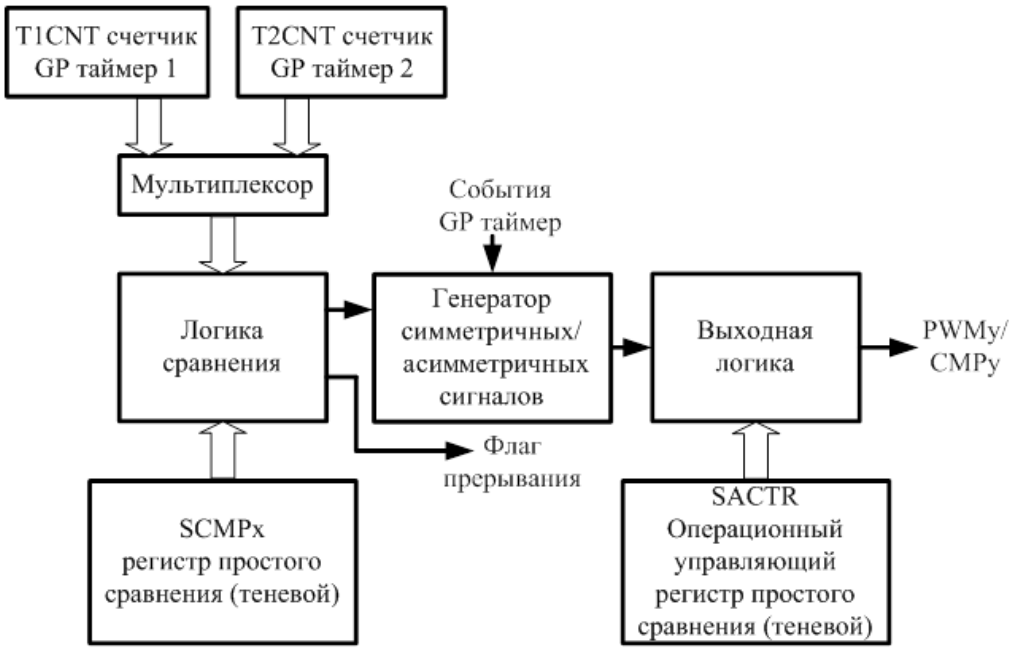


Рисунок 63– Структурная схема блоков простого сравнения
(x = 1, 2, 3; y = 7, 8, 9)

Синхронизация выходов простого сравнения (compare) одинакова с синхронизацией выходов сравнения (compare) GP таймера. Существует флаг прерывания

для каждого блока простого сравнения. Установка флагов прерываний простого сравнения и формирование запросов прерываний простого сравнения так же одинакова с операциями сравнения GP таймера.

Блоки полного сравнения

Три блока полного сравнения включают в себя:

– Три 16-битных регистра сравнения (CMPRx, x = 1, 2, 3), все с соответствующим теневым регистром, (R/W).

– Один регистр управления сравнением (COMCON), (R/W).

– Один 16-битный операционный регистр управления (ACTR), с соответствующим теневым регистром, (R/W).

– Шесть выходов сравнения (compare)/PWM (третье состояние), PWMy/CMPy (y = 1, 2, 3, 4, 5, 6), один на каждый блок простого сравнения, с программируемой полярностью.

– Логику управления и прерывания.

Структурная схема блока полного сравнения показана на рисунке 64.

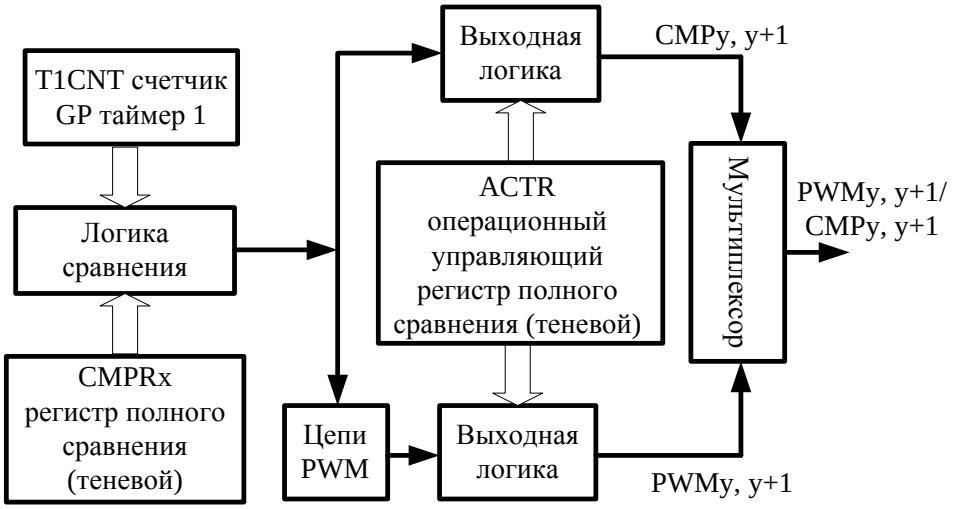


Рисунок 64 – Структурная схема блоков полного сравнения
(x = 1, 2, 3; y = 1, 3, 5)

Временная ось для блоков полного сравнения и соответствующих цепей PWM обеспечивается GP таймером 1. GP таймер 1 может находиться в любом из своих шести счётных режимов, когда разрешена операция сравнения. Однако, на выходах сравнения (compare) не происходит переключения, когда GP таймер 1 находится в прямом/обратном счётном режиме.

Входы/выходы полного сравнения (compare)

Входами блока полного сравнения (compare) являются:

– Управляющие сигналы от регистров управления.

– Сигналы GP таймера 1 (T1CNT), его опустошения и совпадения периода.

– Сигнал сброса.

Выходом блоков полного сравнения (compare) является сигнал совпадения при сравнении. Если операция сравнения разрешена, этот сигнал совпадения устанавливает флаг прерывания и обуславливает переключения на двух выводах, соответствующих блоку полного сравнения.

Режимы работы полного сравнения

Режим работы блоков полного сравнения описывается битами в COMCON. Эти биты определяют:

– Разрешена ли операция сравнения.

– Разрешены ли выходы полного сравнения.

– Условие, при котором регистры полного сравнения обновляются значениями, содержащимися в их теневых регистрах.

– Условие, при котором операционный управляющий регистр обновляется значением, содержащимся в его теневом регистре.

– Разрешен ли режим пространственного вектора PWM.

– Находится ли каждый блок полного сравнения в режиме сравнения (compare) или PWM.

Три блока полного сравнения могут находиться в одном из двух режимов работы: режим сравнения (compare) или режим PWM. Биты в COMCON определяют, в каком режиме работы находится каждый блок полного сравнения.

Режим сравнения (compare)

Когда выбран режим сравнения (compare) и сравнение разрешено, значение счетчика GP таймера 1 постоянно сравнивается со значением, содержащееся в регистре сравнения. Когда происходит совпадение, на двух выходах блока сравнения (compare) возникнет переключение в соответствии с битами в операционном управляющем регистре (ACTR). Биты в ACTR могут индивидуально определять каждый выход как удержание, сброс, установление или переключение в случае совпадения при сравнении. Флаг прерывания сравнения, соответствующий блоку полного сравнения, устанавливается, когда происходит совпадение при сравнении между GP таймером 1 и регистром сравнения этого блока, и сравнение разрешено. Запрос прерывания к ядру формируется флагом прерывания сравнения, если прерывание не маскировано и не выполняются никакие другие, немаскированные прерывания с более высоким приоритетом в той же группе. Синхронизация выходных переключений, установка флагов прерывания и формирование запросов прерываний одинаковы с теми же операциями в режиме сравнения (compare) GP таймера. Выходы блоков полного сравнения в режиме сравнения (compare) являются объектами преобразования выходной логикой.

Режим PWM

Каждый блок полного сравнения может быть индивидуально установлен в режим PWM. Работа блоков полного сравнения в этом режиме одинакова работой в режиме сравнения (compare) GP таймера за исключением того, что блоки полного сравнения управляются различными управляющими регистрами, и выходы полного

сравнения (compare)/PWM являются объектами преобразования блоками «мертвого» времени и логикой пространственного вектора PWM.

Установка регистра для операции полного сравнения

Порядок установки регистра для операции полного сравнения следующий:
- установка T1PR;
- установка ACTR;
- инициализация CMPRx;
- установка COMCON;
- установка T1CON.

Следует отметить, что во многих случаях COMCON требует двух записей для обеспечения полярности PWM выходов.

Регистры блоков сравнения

Адреса регистров, соответствующих блокам полного и простого сравнения и цепям PWM, даны в таблице 55.

Управляющий регистр сравнения (COMCON)

Функционирование блоков полного и простого сравнения управляется 16-битным управляющим регистром сравнения (COMCON). Описание бит COMCON показано на рисунке 65. COMCON имеет доступ чтения и записи и картирован в память данных.

15	14	13	12	11	10	9	8
CENABLE	CLD1	CLD0	SVENABLE	ACTRDL1	ACTRDL0	FCOMPOE	SCOMPOE
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
SELTMR	SCLD1	SCLD0	SACTRDL1	SACTRDL0	SELCMP3	SELCMP2	SELCMP1
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 65 – Управляющий регистр (COMCON) – адрес 7411h

Бит 15 CENABLE. Разрешение сравнения.
0 = Запрещение операции сравнения. Все теневые регистры (CMPRx, SCMPRx, ACTR, SACTR) становятся прозрачными.
1 = Разрешение операции сравнения

Биты 14-13 CLD1, CLD0. Условие перезагрузки регистра полного сравнения CMPRx
00 = Когда T1CON = 0 (то есть, при опустошении)
01 = Когда T1CON = 0 или T1CON = T1PR (то есть, при опустошении или совпадении периода).

10 = Немедленно
11 = Зарезервировано. Результат непредсказуем.

Бит 12 SVENABLE. Разрешение режима пространственного вектора PWM. В режиме пространственного вектора PWM все шесть выходов полного сравнения (compare) являются выходами PWM. SVENABLE = 1 подменяет биты 0, 1 и 2 в COMCON.
0 = Запрещение режима пространственного вектора PWM.
1 = Разрешение режима пространственного вектора PWM.

Биты 11-10 ACTRDL1, ACTRDL0. Условие перезагрузки операционного регистра полного сравнения ACTR.
00 = Когда T1CNT = 0 (то есть, при опустошении)
01 = Когда T1CNT = 0 или T1CNT = T1PR (то есть, при опустошении или совпадении периода).
10 = Немедленно
11 = Зарезервировано. Результат непредсказуем.

Бит 9 FCOMPOE. Выход разрешения полного сравнения. Активный PDPINT очищает этот бит до 0.
0 = Выводы полного сравнения становятся в высокоимпедансное состояние, таким образом, они запрещены.
1 = Выводы полного сравнения не становятся в высокоимпедансное состояние, таким образом, они разрешены.

Бит 8 SCOMPOE. Выход разрешения простого сравнения. Активный PDPINT очищает этот бит до 0.
0 = Выводы простого сравнения становятся в высокоимпедансное состояние, таким образом, они запрещены.
1 = Выводы простого сравнения не становятся в высокоимпедансное состояние, таким образом, они разрешены.

Бит 7 SELTMR. Выбор временной базы простого сравнения.
0 = GP таймер1
1 = GP таймер2

Биты 6-5 SCLD1, SCLD0. Условие перезагрузки регистра простого сравнения SCMPRx
00 = Когда TyCNT = 0 (y = 1 или 2 в соответствии с SELTMR).
01 = Когда TyCNT = 0 или TyCNT = T1PR.
10 = Немедленно.
11 = Зарезервировано. Результат непредсказуем.

Биты 4-3 SACTRDL1, SACTRDL0. Условие перезагрузки операционного регистра простого сравнения SACTR.
00 = Когда TyCNT = 0 (y = 1 или 2 в соответствии с SELTMR).
01 = Когда TyCNT = 0 или TyCNT = T1PR.

10 = Немедленно.
11 = Зарезервировано. Результат непредсказуем.

Бит 2 SELCMP3. Выбор режима для PWM6/CMP6 и PWM5/CMP5 (для блока полного сравнения 3)
0 = Режим сравнения (compare)
1 = Режим PWM

Бит 1 SELCMP2. Выбор режима для PWM4/CMP4 и PWM3/CMP3 (для блока полного сравнения 2)
0 = Режим сравнения (compare)
1 = Режим PWM

Бит 0 SELCMP1. Выбор режима для PWM2/CMP2 и PWM1/CMP1 (для блока полного сравнения 1)
0 = Режим сравнения (compare)
1 = Режим PWM

Примечание – Требуются две последовательные записи в COMCON для обеспечения правильной работы блоков полного сравнения в режиме PWM:

1 Разрешение режима PWM без разрешения операции сравнения (compare).

2 Разрешение операции сравнения (compare) установкой COMCON[15] в 1 без изменения других разрядов.

В примере 6 показывается программирование COMCON.

Пример 6 – Блоки полного сравнения в режиме PWM

```
;*************************************************************
; Формирование регистра COMCON *
;*************************************************************
    SPLK #0100101101010111B, COMCON ; Нужны две записи в COMCON
    SPLK #1100101101010111B, COMCON ; для правильной работы.
          ||||||||||||||||
          FEDCBA9876543210
; bit 15    1:  Разрешение PWM.
; bit 14-13 10: Загрузка сравнения (compare) немедленно.
; bit 12    0:  Запрещение пространственного вектора.
; bit 11-10 10: Загрузка ACTR немедленно.
; bit 9     1:  PWM определяется ACTR.
; bit 8     1:  PWM определяется SACTR.
; bit 7     0:  Простое сравнение соответствующее GP таймер 2.
; bit 6-5   10: Загрузка SCMPR немедленно.
; bit 4-3   10: Загрузка SACTR немедленно.
; bit 2     1:  CMP6/5 разрешен.
; bit 1     1:  CMP4/3 разрешен.
; bit 0     1:  CMP2/1 разрешен.
```

Операционный управляющий регистр полного сравнения (ACTR)

Биты в операционном управляющем регистре полного сравнения (ACTR) управляют действиями, которые происходят в каждом из шести выводов сравнения (compare) (PWMx/CMPx, x = 1–6) при событии сравнения, в обоих режимах сравнения (compare) и PWM, если операция сравнения (compare) разрешена COMCON[15]. ACTR имеет двойную буферизацию. Условие, при котором ACTR перезагружается, определяется битами в COMCON. ACTR так же содержит биты SVRDIR, D2, D1, и D0, необходимые для операции пространственного вектора PWM. Описание бит ACTR показано на рисунке 66.

15	14	13	12	11	10	9	8
SVRDIR	D2	D1	D0	CMP6ACT1	CMP6ACT0	CMP5ACT1	CMP5ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
CMP4ACT1	CMP4ACT0	CMP3ACT1	CMP3ACT0	CMP2ACT1	CMP2ACT0	CMP1ACT1	CMP1ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 66 – Операционный управляющий регистр полного сравнения (ACTR) – адрес 7413h

Бит 15 SVRDIR. Направление вращения пространственного вектора PWM. Используется только при формировании выхода пространственного вектора PWM
0 = Положительное (против часовой стрелки)
1 = Отрицательное (по часовой стрелке)

Биты 14-12 D2 – D0. Основные биты пространственного вектора. Используются только при формировании выхода пространственного вектора PWM

Биты 11-10 CMP6ACT1, CMP6ACT0. Действие на выводе полного сравнения (compare) 6, PWM6/CMP6.

	Режим сравнения (compare)	Режим PWM
00	Удержание	Принудительный низкий уровень
01	Сброс	Активный низкий уровень
10	Установка	Активный высокий уровень
11	Переключение	Принудительный высокий уровень

Биты 9-8 CMP5ACT1, CMP5ACT0. Действие на выходе полного сравнения (compare) 5, PWM5/CMP5.

	Режим сравнения (compare)	Режим PWM
00	Удержание	Принудительный низкий уровень
01	Сброс	Активный низкий уровень
10	Установка	Активный высокий уровень
11	Переключение	Принудительный высокий уровень

Биты 7-6 CMP4ACT1, CMP4ACT0. Действие на выходе полного сравнения (compare) 4, PWM4/CMP4.

	Режим сравнения (compare)	Режим PWM
00	Удержание	Принудительный низкий уровень
01	Сброс	Активный низкий уровень
10	Установка	Активный высокий уровень
11	Переключение	Принудительный высокий уровень

Биты 5-4 CMP3ACT1, CMP3ACT0. Действие на выходе полного сравнения (compare) 3, PWM3/CMP3.

	Режим сравнения (compare)	Режим PWM
00	Удержание	Принудительный низкий уровень
01	Сброс	Активный низкий уровень
10	Установка	Активный высокий уровень
11	Переключение	Принудительный высокий уровень

Биты 3-2 CMP2ACT1, CMP2ACT0. Действие на выходе полного сравнения (compare) 2, PWM2/CMP2.

	Режим сравнения (compare)	Режим PWM
00	Удержание	Принудительный низкий уровень
01	Сброс	Активный низкий уровень
10	Установка	Активный высокий уровень
11	Переключение	Принудительный высокий уровень

Биты 1-0 CMP1ACT1, CMP1ACT0. Действие на выходе полного сравнения (compare) 1, PWM1/CMP1.

	Режим сравнения (compare)	Режим PWM
00	Удержание	Принудительный низкий уровень
01	Сброс	Активный низкий уровень
10	Установка	Активный высокий уровень
11	Переключение	Принудительный высокий уровень

Операционный управляющий регистр простого сравнения (SACTR)

Действия выводов простого сравнения (compare) при событиях сравнения описываются 16-битным операционным управляющий регистром простого сравнения (SACTR). Описание бит SACTR показано на рисунке 67. SACTR имеет двойную буферизацию. Условие, при котором SACTR перезагружается, определяется битами в COMCON.

15-8						
Зарезервировано						
7-6	5	4	3	2	1	0
	SCMP3ACT1	SCMP3ACT0	SCMP2ACT1	SCMP2ACT0	SCMP1ACT1	SCMP1ACT0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 67 – Операционный управляющий регистр простого сравнения (SACTR) – адрес 7414h

Биты 15-6 Зарезервированы. Читаются как 0, запись не возможна.

Биты 5-4 SCMP3ACT1, SCMP3ACT0. Действие на выводе 3 простого сравнения (compare), PWM9/CMP9.
00 = Принудительный низкий уровень
01 = Активный низкий уровень
10 = Активный высокий уровень
11 = Принудительный высокий уровень

Биты 3-2 SCMP2ACT1, SCMP2ACT0. Действие на выводе 2 простого сравнения (compare), PWM8/CMP8.
00 = Принудительный низкий уровень
01 = Активный низкий уровень
10 = Активный высокий уровень
11 = Принудительный высокий уровень

Биты 1-0 SCMP1ACT1, SCMP1ACT0. Действие на выводе 1 простого сравнения (compare), PWM7/CMP7.
00 = Принудительный низкий уровень
01 = Активный низкий уровень
10 = Активный высокий уровень
11 = Принудительный высокий уровень

Прерывания блока сравнения

Существует маскируемый флаг прерывания для каждого блока сравнения в EVIFRA и EVIFRC. Флаг прерывания блока сравнения устанавливается через два такта после совпадения при сравнении, если разрешена операция сравнения (compare). Запрос прерывания будет сформирован флагом, если флаг установлен, не маскирован, и не выполняются никакие другие, немаскированные прерывания с более высоким приоритетом в той же группе.

Сброс блока сравнения

Когда происходит какое-либо событие сброса, все регистры, связанные с блоками сравнения, сбрасываются в 0, и все выводы сравнения (compare) переходят в высокоимпедансное состояние.

13.2.5 Цепи PWM, связанные с блоками полного сравнения

Цепи PWM, связанные с блоками полного сравнения, позволяют им формировать 6 выходных каналов PWM с программируемым «мертвым» временем и выходной полярностью. Структурная схема цепей PWM показана на рисунке 68. Она состоит из следующих функциональных блоков:

- Генераторы асимметричных/симметричных сигналов.
- Программируемый блок «мертвого» времени.
- Выходная логика.
- Пространственный вектор (Space vector SV) конечного автомата PWM.

Генераторы асимметричных/симметричных сигналов одинаковы с генераторами GP таймера и блоков простого сравнения, поэтому здесь они не будут описаны. Блоки «мертвого» времени и выходная логика описаны ниже. Пространственный вектор конечного автомата PWM и способы формирования пространственного вектора PWM описаны далее в этом подразделе.

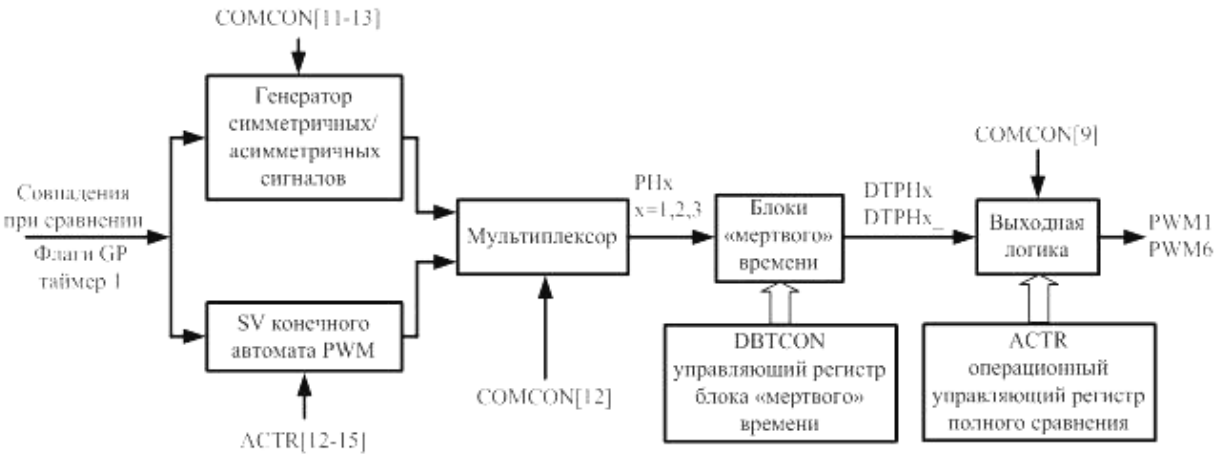


Рисунок 68 – Структурная схема цепей PWM

Цепи PWM предназначены для минимизации издержек производительности процессора и вмешательства пользователя в формирование PWM сигналов, используемых в приложениях управления движением и двигателями. Формирование PWM с помощью блоков полного сравнения и связанных цепей PWM управляется следующими управляющими регистрами: T1CON, COMCON, ACTR и DBTCON.

Возможность формирования PWM с помощью модуля EV

Возможность формирования PWM с помощью модуля EV заключается в следующем:

- Наличие девяти независимых выводов PWM.

– Программируемое «мертвое» время для выходных пар PWM, связанных с блоками полного сравнения, от 0 до 2048 тактовых периодов, 81,92 мкс, если период тактового сигнала равен 8 нс.

– Минимальная длительность «мертвого» времени в одном тактовом периоде.

– Минимальная ширина и минимальный инкремент/декремент ширины сигнала PWM в одном тактовом периоде.

– 16-битное максимальное разрешение PWM.

– Оперативное изменение несущей частоты PWM (регистры периода с двойной буферизацией).

– Оперативное изменение ширины импульса PWM (регистры сравнения с двойной буферизацией).

– Прерывание защиты электропитания.

– Программируемое формирование асимметричных, симметричных сигналов и сигналов SV PWM.

– Минимальные издержки производительности процессора благодаря автоперезагрузке регистров периода и сравнения.

Программируемый блок «мертвого» времени

Особенности программируемого блока «мертвого» времени:

– один 16-битный управляющий регистр «мертвого» времени, DBTCON (R/W);

– один делитель входного тактового сигнала: x/1, x/2, x/4, x/8;

– входной тактовый сигнал процессора;

– три 8-битных счетчика в обратном направлении;

– логика управления.

Управляющий регистр таймера «мертвого» времени (DBTCON)

Функционирование блока «мертвого» времени управляется регистром таймера «мертвого» времени (DBTCON). Описание бит DBTCON показано на рисунке 69.

15	14	13	12	11	10	9	8
DBT7	DBT6	DBT5	DBT4	DBT3	DBT2	DBT1	DBT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2-0		
EDBT3	EDBT2	EDBT1	DBTPS1	DBTPS0			
RW-0	RW-0	RW-0	RW-0	RW-0			

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 69 – Управляющий регистр таймера «мертвого» времени (DBTCON) – адрес 7415h

Биты 15-8 DBT7 (MSB)–DBT0 (LSB). Период таймера «мертвого» времени. Эти биты описывают значение периода трёх 8-битных таймеров «мертвого» времени.

Бит 7 EDBT3. Разрешение таймера «мертвого» времени 3 (для выводов PWM5/CMP5 и PWM6/CMP6 блока полного сравнения 3).
0 = Запрещено
1 = Разрешено

Бит 6 EDBT2. Разрешение таймера «мертвого» времени 2 (для выводов PWM3/CMP3 и PWM4/CMP4 блока полного сравнения 2).
0 = Запрещено
1 = Разрешено

Бит 5 EDBT1. Разрешение таймера «мертвого» времени 1 (для выводов PWM1/CMP1 и PWM2/CMP2 блока полного сравнения 1).
0 = Запрещено
1 = Разрешено

Биты 4-3 DBTPS1, DBTPS0. Делитель таймера «мертвого» времени.
000 = x/1
001 = x/2
010 = x/4
011 = x/8
x = частота тактового сигнала процессора

Биты 2-0 Зарезервированы. При чтении – нули; запись не имеет эффекта.

Входами блока «мертвого» времени являются PH1, PH2, и PH3 от генераторов асимметричных/симметричных сигналов блоков полного сравнения (compare) 1, 2 и 3 соответственно.

Выходами блока «мертвого» времени являются DTPH1, DTPH1_, DTPH2, DTPH2_, DTPH3, и DTPH3_, соответствующие PH1, PH2, и PH3.

Формирование «мертвого» времени

Для каждого входного сигнала PHx формируются два выходных сигнала DTPHx и DTPHx_. Когда блок «мертвого» времени не разрешен для блока сравнения и его связанного выхода, два сигнала будут одинаковыми. Когда блок «мертвого» времени разрешен для блока сравнения, фронты переключения двух сигналов будут разделены временным интервалом, называемым «мертвым» временем. Этот временной интервал определяется битами DBTCON. Примем значение в DBTCON[15-8] за m и значение в DBTCON[4-3] соответственно делителю x/p. Таким образом, значение «мертвого» времени будет равно p*m тактового сигнала процессора.

В таблице 60 показано «мертвое» время, формируемое типичной комбинацией бит в DBTCON. Значения основаны на периоде работы в 8 нс. На рисунке 70 представлена структурная схема логики «мертвого» времени для одного блока полного сравнения. На рисунке 71 представлены сигналы, формирующие «мертвое» время.

Входы и выходы блока «мертвого» времени

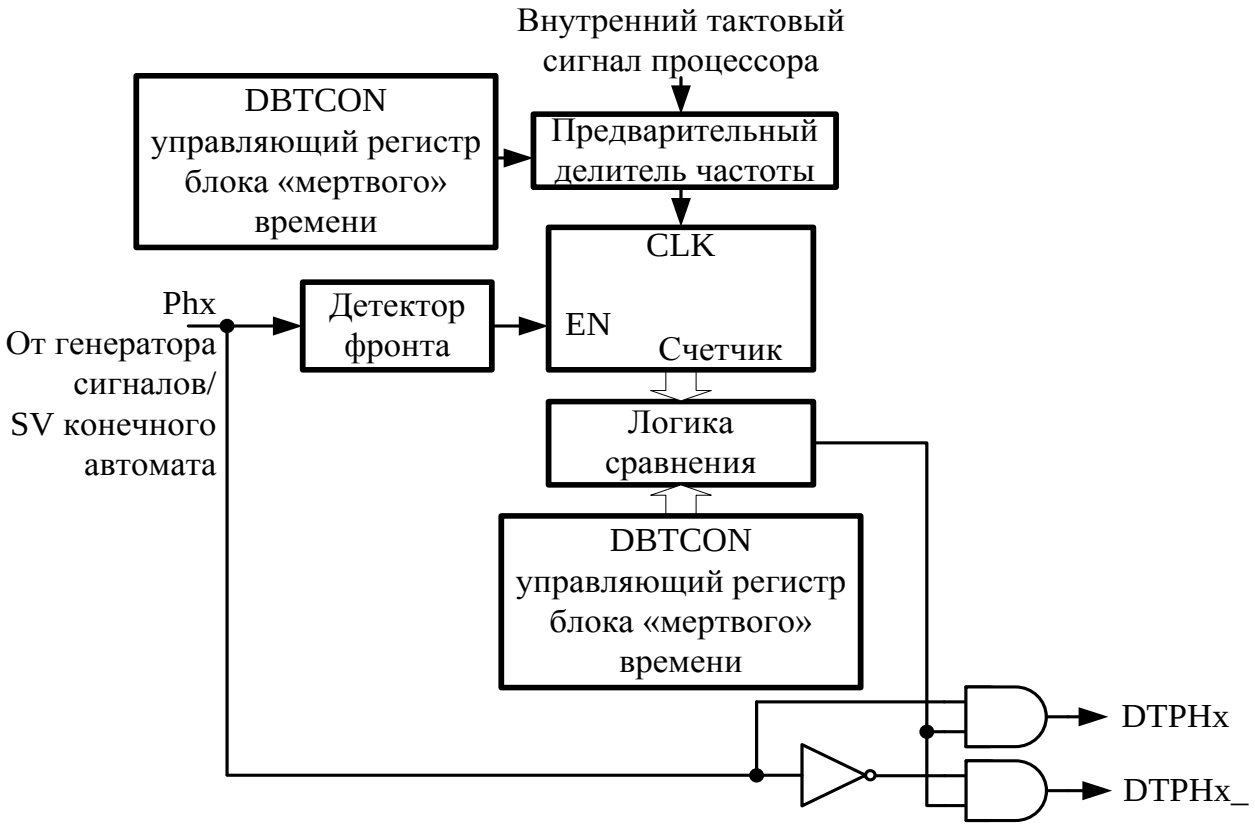


Рисунок 70 – Структурная схема блока «мертвого» времени (x = 1, 2 или 3)

Таблица 60 – Пример формирования «мертвого» времени

DBT7–DBT0 (m) (DBTCON[15–8])	DBTPS1–DBTPS0 (p) (DBTCON[4–3])			
	11(P = 8), мкс	10(P = 4), мкс	01(P = 2), мкс	00(P = 1), мкс
00h	0	0	0	0
01h	0,32	0,16	0,08	0,04
02h	0,64	0,32	0,16	0,08
03h	0,96	0,48	0,24	0,12
04h	1,28	0,64	0,32	0,16
05h	1,60	0,8	0,4	0,2
06h	1,92	0,96	0,48	0,24
07h	2,24	1,12	0,56	0,28
08h	2,56	1,28	0,64	0,32

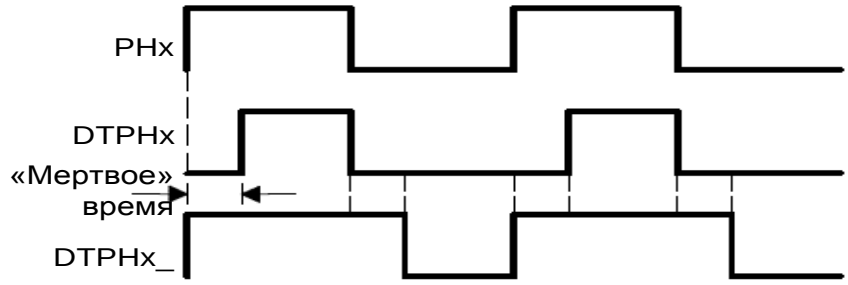


Рисунок 71 – Сигналы, формирующие «мертвое» время

Другие важные особенности блоков «мертвого» времени

Блок «мертвого» времени предназначен для предотвращения перекрытия между периодами включения устройств верхнего и нижнего уровней, управляемых двумя выходами сравнения (compare)/PWM, связанными с каждым блоком полного сравнения, во время любой рабочей ситуации, включая ситуации, когда пользователь загружает значение «мертвого» времени больше, чем рабочий цикл, и когда коэффициент заполнения 100 % или 0 %. Как результат, выходы сравнения (compare) /PWM, связанные с блоком полного сравнения, не сбрасываются до пассивного состояния в конце периода, когда «мертвое» время разрешено для блока полного сравнения.

«Мертвое» время запрещено, когда блок полного сравнения находится в режиме сравнения (compare).

В примере 7 показывается программа инициализации для блоков полного сравнения для симметричных PWM сигналов при активированном блоке «мертвого» времени.

Пример 7 – Программа инициализации для формирования «мертвого» времени.

```
;*****************************************************************
; Эта часть кода инициализирует симметричные PWM с «мертвым» временем
;*****************************************************************
    LDPK #232                  ; DP => Регистры модуля EV
;*****************************************************************
; Формирование ACTR
;*****************************************************************
    SPLK #0000011001100110b, ACTR     ; Управление GP таймером
    ;          ||||||||||||||||
    ;          FEDCBA9876543210
;
* bit 15     0   Направление вращения SV (здесь не используется)
* bits 12-14 000 Биты SV (здесь не используется)
* bits 10-11 01  PWM6 активный низкий уровень
* bits 8-9   10  PWM5 активный высокий уровень
* bits 6-7   01  PWM4 активный низкий уровень
* bits 4-5   10  PWM3 активный высокий уровень
* bits 2-3   01  PWM2 активный низкий уровень
```

```
* bits 0-1   10  PWM1 активный высокий уровень
;*************************************************************
;*************************************************************
; Формирование DBTCON
;*************************************************************
    SPLK #0000010111100000b, DBTCON     ; Управление таймером
    ;          ||||||||||||||||
    ;          FEDCBA9876543210
;
* bits 8-15 101: 5 периодов «мертвого» времени
* bit 7     1:   Разрешение «мертвого» времени для PWM 5/6
* bit 6     1:   Разрешение «мертвого» времени для PWM 3/4
* bit 5     1:   Разрешение «мертвого» времени для PWM 1/2
* bits 3-4  00:  Делитель таймера для блока «мертвого» времени
* bits 0-2  000: Зарезервировано
;*************************************************************
; Инициализация регистров сравнения
;*************************************************************
    SPLK #0008h, CMPR1
    SPLK #000Ch, CMPR2
    SPLK #0011h, CMPR3
;*************************************************************
; Инициализация регистра T1PER
;*************************************************************
    SPLK #0014h, T1PER
```

Выходная логика

Схемы выходной логики определяют полярность и/или действие, которое должно быть произведено в случае совпадения при сравнении для выходов PWMx/CMPx, где x = 1–9. Выходы, связанные с каждым блоком полного сравнения, могут быть определены как установленные в активный низкий уровень, активный высокий уровень, принудительный низкий уровень, принудительный высокий уровень, когда блок полного сравнения находится в режиме PWM. Они могут быть установлены в удержание, установку, сброс или переключение, когда блок полного сравнения находится в режиме сравнения (compare). Полярность и/или действие выходов полного и простого сравнения (compare)/PWM могут быть запрограммированы необходимым значением бита в ACTR и SACTR. Шесть выводов полного сравнения (compare)/PWM и три вывода простого сравнения (compare)/PWM могут быть установлены в высокоимпедансное состояние любым из следующих вариантов:

– Программный сброс битов COMCON[9] и COMCON[8] соответственно.

– Аппаратная установка PDPINT в низкое состояние, когда PDPINT не маскирован.

– Появление какого-либо события сброса.

Активный PDPINT (когда разрешен) и системный сброс подменяют биты в COMCON, ACTR и SACTR.

На рисунке 72 представлена структурная схема цепей выходной логики. Входами выходной логики для блоков полного и простого сравнения являются:

– DTPH1, DTPH1_, DTPH2, DTPH2_, DTPH3 и DTPH3_ от блока «мертвого» времени и сигналов совпадения при полном сравнении.

– Выходы от генератора асимметричных/симметричных сигналов блоков простого сравнения.

– Биты ACTR и SACTR.

– PDPINT и сброс.

Выходами выходной логики для блоков полного и простого сравнения являются PWMx/CMPx, где x = 1 – 9.

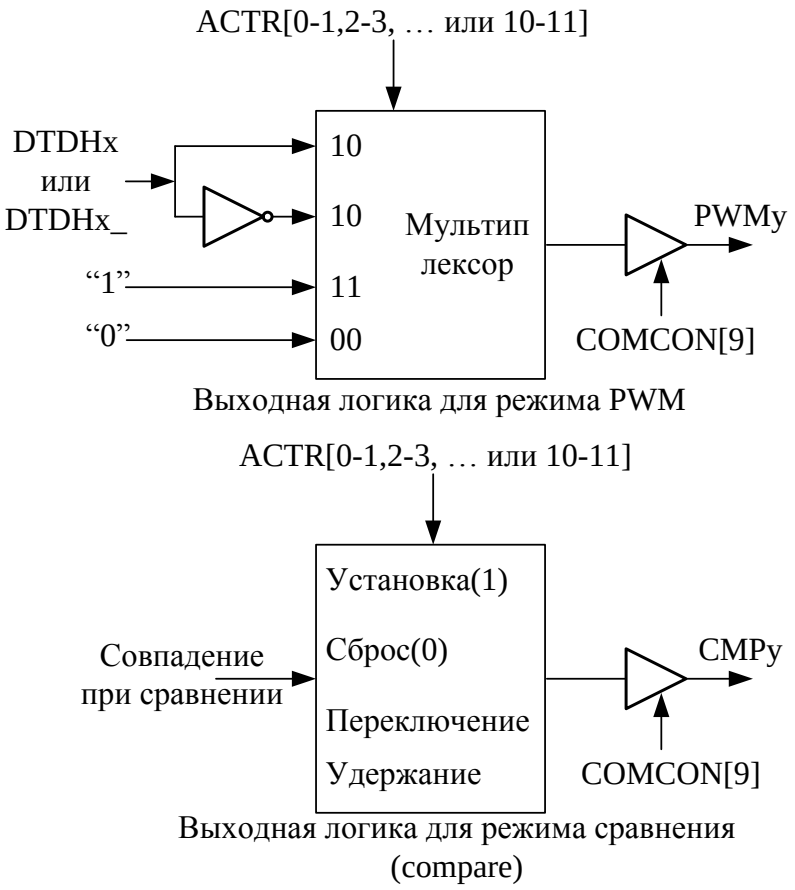


Рисунок 72 – Структурная схема выходной логики
(x = 1, 2 или 3; y = 1, 2, 3, 4, 5 или 6)

13.2.6 Формирование PWM сигналов с помощью блоков сравнения и цепей PWM

Сигналы PWM

Сигналы с широтно-импульсной модуляцией (PWM) представляют собой последовательность импульсов с изменяющейся длительностью. Импульсы распределены на определенном количестве периодов фиксированной длины таким образом, что на каждый период приходится один импульс. Фиксированный период называется периодом PWM (несущей), его обратное значение - частотой PWM (несущей). Ширина импульсов PWM определяется или модулируется от импульса к

импульсу в соответствии с другой последовательностью требуемых значений, модулирующим сигналом.

В системах управления двигателем PWM сигналы используются для управления временем включения и выключения устройств переключения питания, которые переносят требуемый ток и энергию на обмотки двигателя (см. рисунок 72). Форма и частота фазного тока и количество энергии, переносимой на обмотки двигателя, управляют требуемой скоростью и крутящим моментом двигателя. В этом случае управляющее напряжение или ток, приложенный к двигателю, являются модулирующими сигналами. Частота модулирующего сигнала обычно намного меньше несущей частоты PWM.

Формирование PWM сигналов

Для формирования PWM сигнала требуется соответствующий таймер для повторения счетного периода, который одинаков с периодом PWM. Регистр сравнения используется для удержания модулирующих значений. Значение регистра сравнения постоянно сравнивается со значением счетчика таймера. Когда значения совпадают, происходит переключение (от низкого до высокого или от высокого до низкого) на соответствующем выходе. Когда происходит второе совпадение между значениями или при достижении конца периода таймера, происходит другое переключение (от высокого до низкого или от низкого до высокого) на соответствующем выходе. В этом случае выходной сигнал формируется тем, чья длительность включения (или выключения) пропорциональна значению в регистре сравнения. Этот процесс повторяется для каждого периода таймера с различными (модулирующими) значениями в регистре сравнения. В результате на соответствующем выходе формируется PWM сигнал.

«Мертвое» время

Во многих приложениях управления движением/двигателем и силовой электроники два (верхнее и/или нижнее) силовых устройства соединены последовательно на одной шине преобразователя питания, и периоды включения этих двух устройств не должны пересекаться друг с другом для предотвращения сквозного короткого замыкания. Таким образом, пара непересекающихся PWM выходов требуется для правильного включения или выключения устройств. «Мертвое» время часто вставляется между выключением одного транзистора и включением другого. Эта задержка позволяет завершить выключение одного транзистора перед включением другого. Требуемая временная задержка определяется характеристиками включения и выключения силовых транзисторов и характеристиками загрузки в специальном приложении.

Формирование выходов PWM с помощью модуля EV

Каждый из трёх блоков полного сравнения вместе с GP таймером 1, блоком «мертвого» времени и выходной логикой в модуле модуля EV может быть использован для формирования пары PWM выходов с программируемым «мертвым» временем и выходной полярностью на двух специализированных выводах устройства.

Существует шесть таких специализированных выводов PWM, связанных с тремя блоками полного сравнения в модуле модуля EV. Эти шесть специализированных выходов могут быть использованы для простого управления двигателями трёхфазного переменного тока или бесщёточными двигателями постоянного тока. Гибкость выходного режима управляется операционным регистром полного сравнения (ACTR), который так же позволяет упростить управление двигателями с регулируемым магнитным сопротивлением и синхронизированными реактивными двигателями в широком диапазоне приложений. Цепи PWM так же могут быть использованы для упрощенного управления других типов двигателей таких, как щёточный двигатель постоянного тока и шаговый двигатель в одно или многоосных управляющих приложениях. Три блока простого сравнения с GP таймером 1 или GP таймером 2 могут быть использованы для формирования других трёх выходов PWM, в этом случае «мертвое» время не требуется или формируется цепями вне кристалла. Каждый блок сравнения GP таймера, если нужно, может формировать выход PWM, основываясь на собственном таймере.

Формирование асимметричных и симметричных PWM

Асимметричные и симметричные PWM сигналы могут быть сформированы каждым блоком сравнения в модуле модуля EV. Кроме того, три блока полного сравнения вместе могут быть использованы для формирования выходов трёхфазного симметричного пространственного вектора PWM. Формирование PWM с помощью блоков сравнения GP таймера описано в разделе «GP таймер». Формирование PWM сигнала с помощью блоков простого сравнения одинаково с формированием блоками сравнения GP таймера, исключая тех, которые используют различные управляющие регистры, и того, что любой из GP таймеров 1 и 2 может быть выбран как временная ось.

Установка регистров для формирования PWM

Все три вида формирования PWM сигналов с помощью блоков полного сравнения и соответствующих схем требуют определенную конфигурацию соответствующих регистров модуля EV. Процесс установки для формирования PWM сигнала включает в себя следующие шаги:

– Установка и загрузка ACTR.

– Установка и загрузка DBTCON, если будет использовано «мертвое» время.

– Инициализация CMPRx.

– Установка и загрузка COMCON без разрешения операции сравнения.

– Установка и загрузка COMCON для разрешения операции сравнения.

– Установка и загрузка T1CON для запуска работы.

– Перезапись CMPRx новым определённым значением.

Примечание – Перед записью в T1CON для запуска работы COMCON должен быть записан дважды, чтобы обеспечить поднятие выходов полного сравнения в корректное (пассивное) состояние.

Формирование асимметричных PWM сигналов

Запускаемый фронтом или асимметричный PWM сигнал характеризуется модулированными импульсами, которые не центрированы с учетом периода PWM, как показано на рисунке 73. Ширина каждого импульса может быть изменена только с одной стороны импульса.

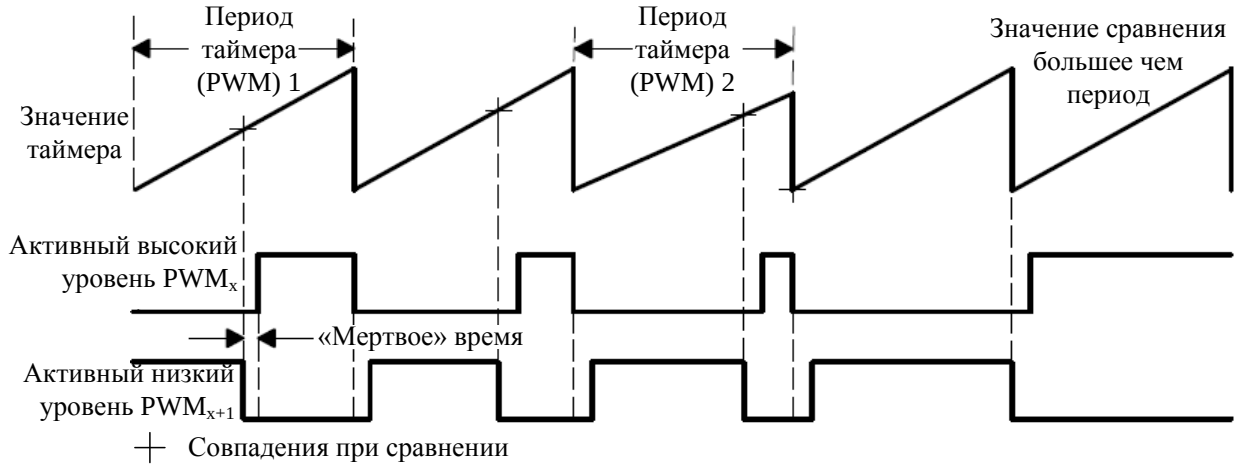


Рисунок 73 – Формирование асимметричных PWM сигналов с помощью блоков полного сравнения и цепей PWM (x = 1, 3 или 5)

Для формирования асимметричного PWM сигнала, GP таймер 1 устанавливается в непрерывный прямой режим счёта. Его регистр периода загружается значением соответствующим требуемому периоду несущей PWM. Регистр COMCON конфигурируется для разрешения операции сравнения, выбранные выходы устанавливаются как выходы PWM и разрешаются выходы. Если «мертвое» время разрешено, требуемое значение, соответствующее блоку «мертвого» времени, должно быть записано в 8 старших бит DBTCON как период для 8-битного таймера «мертвого» времени. Одно значение «мертвого» времени используется для всех выходных каналов PWM.

При помощи правильной программной конфигурации ACTR, на одном выходе, связанном с блоком полного сравнения, может быть сформирован нормальный PWM сигнал, пока другой удерживается в низком (или выключен) или высоком (или включен) состоянии в начале, середине или в конце периода PWM. Такая программно управляемая гибкость выходов PWM особенно полезна для двигателей с регулируемым магнитным сопротивлением.

После запуска GP таймера 1, регистры сравнения перезаписываются каждый период PWM вновь определёнными значениями сравнения для корректировки ширины (коэффициента заполнения) выходов PWM, которые управляют продолжительностью включения и выключения силовых устройств. Как только регистры сравнения становятся скрытыми, новые значения могут быть записаны в них в любое время в течение периода. Новые значения могут быть записаны в операционные регистры и регистры периода в любое время в течение периода для изменения периода PWM или для усиления изменений в определении выхода PWM.

Формирование симметричных PWM сигналов

Центрированный или симметричный PWM сигнал характеризуется модулированными импульсами, которые центрированы с учетом каждого периода PWM. Преимуществом симметричного PWM сигнала над асимметричным является то, что он имеет две пассивные зоны с одинаковой длительностью – в начале и в конце каждого периода PWM. Эта симметрия призвана обеспечить меньшие гармоники, чем у асимметричного PWM сигнала для фазных токов двигателей переменного тока, таких как асинхронный двигатель и бесщёточный двигатель постоянного тока, когда используется синусоидальная модуляция. На рисунке 74 представлены два примера симметричных PWM сигналов.

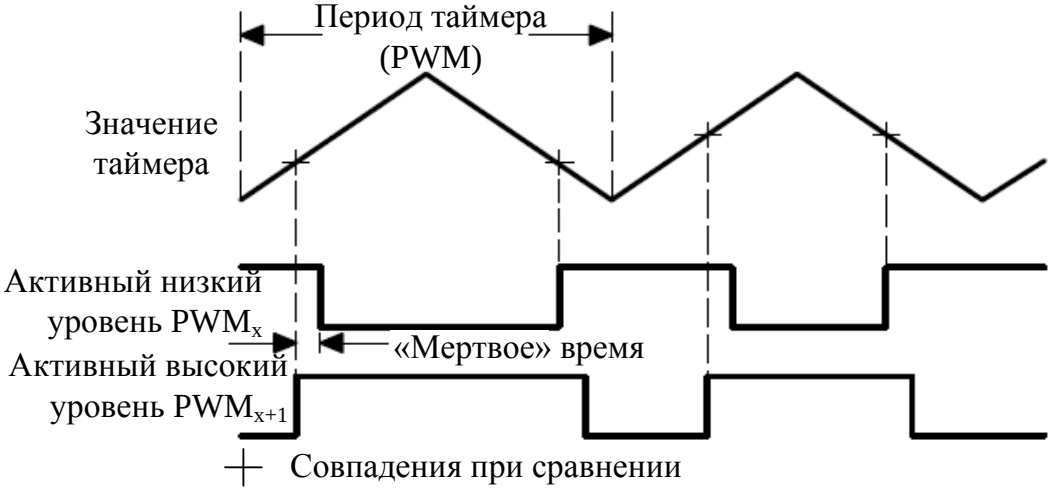


Рисунок 74 – Формирование симметричных PWM сигналов с помощью блоков полного сравнения и цепей PWM (x = 1, 3 или 5)

Формирование симметричных PWM сигналов с помощью блоков полного сравнения одинаково с формированием асимметричных PWM сигналов. Единственным исключением является то, что GP таймер 1 не требует установки в непрерывный прямой/обратный режим счёта.

При формировании симметричных PWM сигналов обычно происходит два совпадения при сравнении в течение периода PWM: одно во время прямого счёта перед совпадением периода и другое во время обратного счёта после совпадения периода. Новое значение сравнения может стать эффективным после совпадения периода (перезагрузка на периоде), что позволяет опережать или задерживать следующий фронт PWM импульса. Эта особенность используется с целью модификации PWM сигнала для компенсации текущих ошибок, вызванных «мертвым» временем при управлении двигателями переменного тока.

Как только регистры сравнения становятся теневыми, новые значения могут быть записаны в них в любое время в течение периода. Новые значения могут быть записаны в операционные регистры и регистры периода в любое время в течение периода для изменения периода PWM или для усиления изменений в определении выхода PWM.

13.2.7 Пространственный вектор PWM

Обзор теории пространственного вектора (Space Vector (SV)) PWM

Пространственный вектор PWM относится к специальной схеме включения шести силовых транзисторов трёхфазного силового преобразователя. Она формирует минимальное искажение гармоники токов на обмотках трёхфазных двигателей переменного тока. Она так же обеспечивает более эффективное использование питающего напряжения по сравнению с синусоидальной модуляцией.

Трёхфазный инвертирующий усилитель мощности

Структура обычного трёхфазного инвертирующего усилителя мощности показана на рисунке 75, где U_a, U_b и U_c – напряжения, прикладываемые к обмоткам двигателя. Шесть силовых транзисторов управляются с помощью $DTPH_x$ и $DTPH_{x\_}$ (x = a, b и c). Когда верхний транзистор включен ($DTPH_x = 1$), нижний транзистор выключен ($DTPH_{x\_} = 0$). Таким образом, состояния включения и выключения верхних транзисторов (Q1, Q3 и Q5) или состояние $DTPH_x$ (x = a, b и c) достаточны для нахождения приложенного к двигателю напряжения U_{out}.

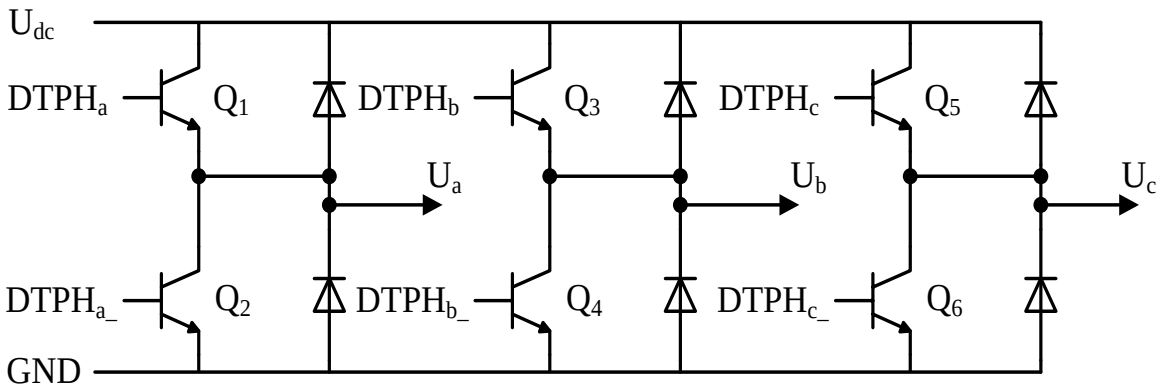


Рисунок 75 – Принципиальная схема трёхфазного инвертирующего усилителя мощности

Последовательность включения инвертирующего усилителя мощности и базовых пространственных векторов

Когда верхний транзистор шины питания включен, напряжение U_x (x = a, b или c), прикладываемое шиной питания к соответствующим обмоткам двигателя, равно напряжению питания U_{dc}. Когда он выключен, прикладываемое напряжение равно 0. Включение и выключение верхних транзисторов ($DTPH_x$, x = a, b или c) имеет восемь возможных комбинаций. Восемь возможных комбинаций и производимое двигателем межфазное и фазовое напряжение в единицах источника постоянного тока U_{dc} показаны в таблице 61, где a, b и c представляют собой значения $DTPH_a$, $DTPH_b$ и $DTPH_c$, соответственно.

Таблица 61 – Последовательность включения трёхфазного инвертирующего усилителя мощности

a	b	c	$U_{a0}(U_{dc})$	$U_{b0}(U_{dc})$	$U_{c0}(U_{dc})$	$U_{ab}(U_{dc})$	$U_{bc}(U_{dc})$	$U_{ca}(U_{dc})$
0	0	0	0	0	0	0	0	0
0	0	1	–1/3	–1/3	2/3	0	–1	1
0	1	0	–1/3	2/3	–1/3	–1	1	0
0	1	1	–2/3	1/3	1/3	–1	0	1
1	0	0	2/3	–1/3	–1/3	1	0	–1
1	0	1	1/3	–2/3	1/3	1	–1	0
1	1	0	1/3	1/3	–2/3	0	1	–1
1	1	1	0	0	0	0	0	0
Примечание – 0 = выключено, 1 = включено.								

Если произвести d-q трансформацию (что эквивалентно ортогональной проекции трёх векторов (a, b и c) на двухмерную плоскость, перпендикулярную вектору (1, 1, 1), d-q плоскость), то в результате получится распределение фазовых напряжений в соответствии с восемью комбинациями на d-q плоскости, что дает шесть ненулевых векторов и два нулевых вектора. Ненулевые векторы формируют оси шестиугольником. Угол между двумя смежными векторами составляет 60 градусов. Два нулевых вектора находятся в начале координат. Эти 8 векторов называются базовыми пространственными векторами и обозначаются U_0, U_{60}, U_{120}, U_{180}, U_{240}, U_{300}, O_{000} и O_{111}. Такая же трансформация может быть применима к требуемому вектору напряжения, прилагаемому к двигателю U_{out}. На рисунке 76 представлены построенные векторы и построенный требуемый вектор напряжения двигателя U_{out}.

Оси d и q d-q плоскости здесь соответствуют горизонтальной и вертикальной геометрическим осям статора машины переменного тока.

Целью метода SV PWM является аппроксимация вектора напряжения двигателя U_{out} комбинацией этих восьми схем включения шести силовых транзисторов.

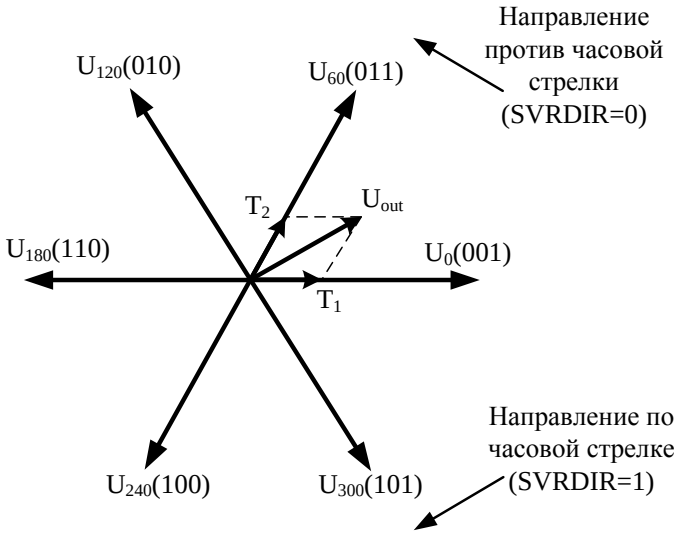


Рисунок 76– Базовые пространственные векторы и схемы включения

Если произвести d-q трансформацию (что эквивалентно ортогональной проекции трёх векторов (a, b и c) на двухмерную плоскость, перпендикулярную вектору (1, 1, 1), d-q плоскость), то в результате получится распределение фазовых напряжений в соответствии с восемью комбинациями на d-q плоскости, что дает шесть ненулевых векторов и два нулевых вектора. Ненулевые векторы формируют оси шестиугольником. Угол между двумя смежными векторами составляет 60 градусов. Два нулевых вектора находятся в начале координат. Эти 8 векторов называются базовыми пространственными векторами и обозначаются U_0, U_{60}, U_{120}, U_{180}, U_{240}, U_{300}, O_{000} и O_{111}. Такая же трансформация может быть применима к требуемому вектору напряжения, прилагаемому к двигателю U_{out}. На рисунке 76 представлены построенные векторы и построенный требуемый вектор напряжения двигателя U_{out}.

Оси d и q d-q плоскости здесь соответствуют горизонтальной и вертикальной геометрическим осям статора машины переменного тока.

Бинарное представление двух смежных базовых векторов различается только на один бит. Таким образом, только один из верхних транзисторов переключается, когда схема переключения включается от U_x до U_{x+60} или от U_{x+60} до U_x. Так же нулевые векторы O_{000} и O_{111} не подают напряжения на двигатель.

Аппроксимация вектора напряжения двигателя с помощью базовых пространственных векторов

Проецируемый вектор напряжения двигателя U_{out}, в любое заданное время, опускается в одном из шести секторов. Таким образом, для каждого периода PWM U_{out} может быть аппроксимировано векторной суммой двух компонентов векторов лежащих на двух смежных базовых векторах:

$$U_{out} = \frac{T_1}{T_p} U_x + \frac{T_2}{T_p} U_{x+60} + \frac{T_0}{T_p} (O_{000} \text{или} O_{111}),$$

где T_0 задается как T_p-T_1-T_2, и T_p является периодом несущей PWM.

Третье слагаемое в этом выражении не влияет на векторную сумму U_{out}.

Эта аппроксимация означает, что верхние транзисторы имеют схемы включения и выключения в соответствии с U_x и U_{x+60} для длительности T_1 и T_2, соответственно, чтобы правильно подать напряжения U_{out} на двигатель. Наличие нулевых базовых векторов помогает уравнивать периоды включения и выключения транзисторов и, тем самым, рассеиваемую мощность.

Формирование сигнала SV PWM с помощью модуля EV

Программные средства

Аппаратно встроенный модуль EV позволяет значительно упростить формирование симметричных сигналов SV PWM. Для формирования выходов SV PWM пользовательская программа должна:

– Сконфигурировать ACTR для определения полярности выводов полного сравнения (compare).

– Сконфигурировать COMCON для разрешения операции сравнения и режима SV PWM и установить условие перезагрузки для ACTR и CMPRx при опустошении.

– Установить GP таймер 1 в режим непрерывного прямого/обратного счета для запуска работы.

Примечание – Разрешение режима SV PWM автоматически устанавливает все выводы полного сравнения (compare) как выходы PWM.

Далее пользовательской программе нужно определить напряжение U_{out}, которое будет приложено к фазам двигателя в двухмерной d-q плоскости, разложить U_{out} на составляющие и определить для каждого PWM периода:

– Два смежных вектора, U_x и U_{x+60}.

– Параметры T_1, T_2 и T_0.

– Записать схему переключения соответствующую U_x в ACTR[14-12] и 1 в ACTR[15] или схему переключения U_{x+60} в ACTR[14-12] и 0 в ACTR[15].

– Установить (1/2 T1) в CMPR1 и (1/2 T1 + 1/2 T2) в CMPR2.

Аппаратный SV PWM

Аппаратный SV PWM в модуле модуля EV выполняет следующие процедуры для завершения периода SV PWM:

– В начале каждого периода, устанавливает выходы PWM на (новое) значение U_y, определяемое ACTR[14-12].

– При первом совпадении при сравнении во время прямого счёта между CMPR1 и GP таймером 1 при (1/2 T1), переключает выходы PWM к схеме U_{y+60}, если ACTR[15] в 1, или в схему U_y, если ACTR[15] в 0. ($U_{0-60} = U_{300}$, $U_{360+60} = U_{60}$).

– При втором совпадении при сравнении во время прямого счёта между CMPR2 и GP таймером 1 при (1/2 T1 + 1/2 T2), переключает выходы PWM к схеме (000) или (111), которые отличаются от следующей схемы на один бит.

– При первом совпадении при сравнении во время обратного счёта между CMPR2 и GP таймером 1 при (1/2 T1 + 1/2 T2), переключает выходы PWM обратно ко второй выходной схеме.

– При втором совпадении при сравнении во время обратного счёта между CMPR1 и GP таймером 1 при (1/2 T1), переключает выходы PWM обратно к первой схеме.

Сигналы SV PWM

Сигналы SV PWM формируются одинаково по отношению к середине каждого периода PWM. Поэтому, это называется методом формирования симметричного SV PWM. На рисунке 77 представлены сигналы симметричного SV PWM.

Неиспользуемый регистр полного сравнения

Только два регистра полного сравнения используются при формировании SV PWM. Третий регистр полного сравнения, тем не менее, продолжает постоянно сравниваться с GP таймером 1. Когда происходит совпадение при сравнении, соот-

ветственный флаг прерывания сравнения будет установлен и будет сформирован запрос прерывания, если флаг не маскирован, и не выполняются никакие другие немаскированные прерывания с более высоким приоритетом в той же группе. Поэтому регистр сравнения, который не используется при формировании выходов SV PWM, может быть использован для временных событий, происходящих в специальных приложениях. Также, из-за дополнительной задержки, вносимой конечным автоматом, выходные транзисторы полного сравнения задерживаются на два тактовых периода тактового сигнала в режиме SV PWM.

Граничные условия SV PWM

Все три выхода полного сравнения (compare) становятся пассивными, когда оба регистра полного сравнения (CMPR1 и CMPR2) загружены нулевым значением в режиме SV PWM. Обязанностью пользователя является обеспечение выполнения (CMPR1) ≤ (CMPR2) ≤ (T1PR) в режиме SV PWM. Иначе возможно непредсказуемое поведение двигателя.

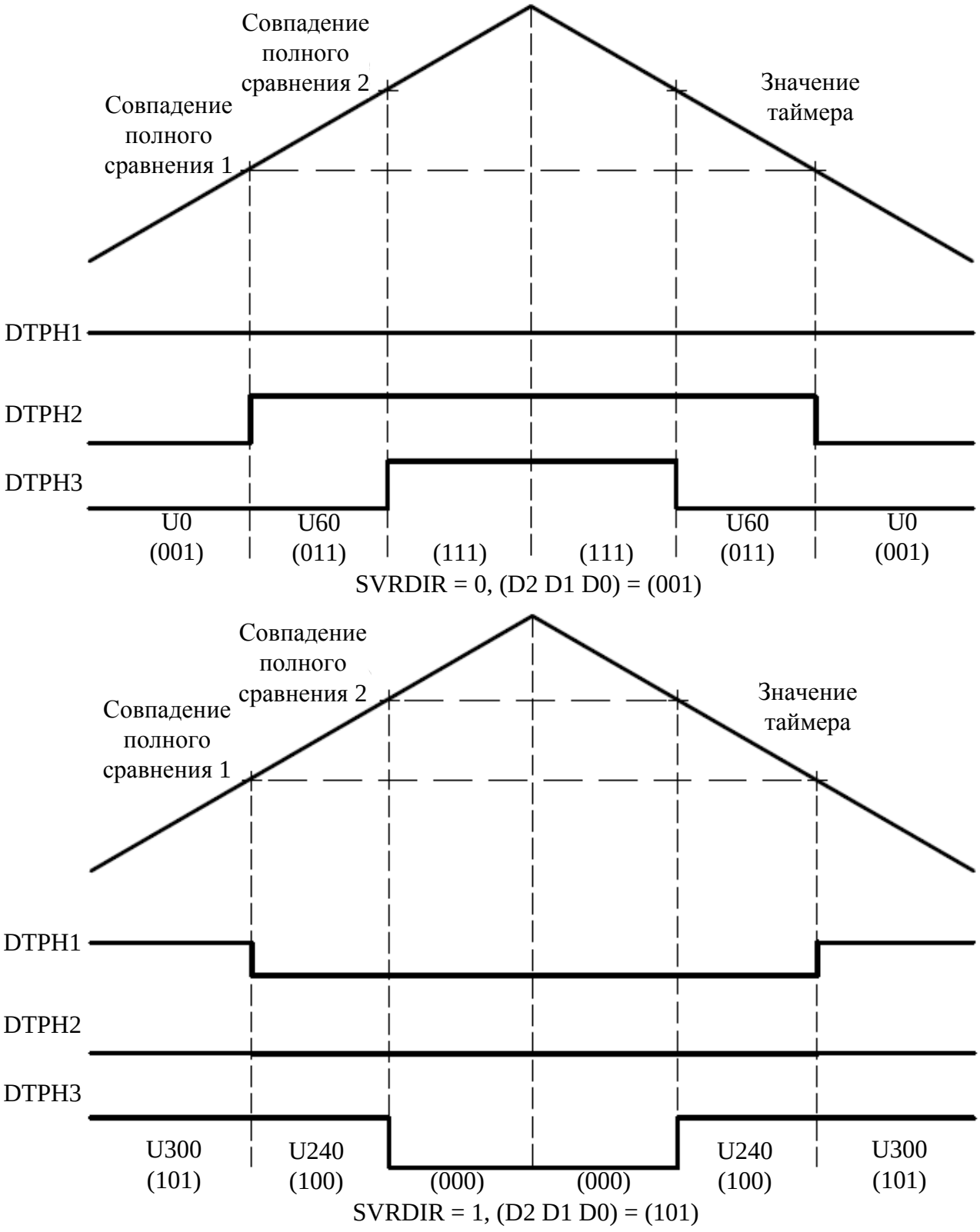


Рисунок 77 –Сигналы симметричного SV PWM

13.2.8 Блоки захвата

Блоки захвата позволяют записывать переключения на входах захвата. Предусмотрено четыре блока захвата (БЗ): БЗ 1, 2, 3 и 4. Каждый БЗ соединён с

входом захвата. Каждый БЗ может выбирать GP таймер 2 или GP таймер 3 как свою временную ось. Значение GP таймера 2 или GP таймера 3 захватывается и хранится в соответствующем двухуровневом стеке FIFO, когда заданное переключение определено на входе захвата CAPx. На рисунке 78 представлена структурная схема блока захвата.

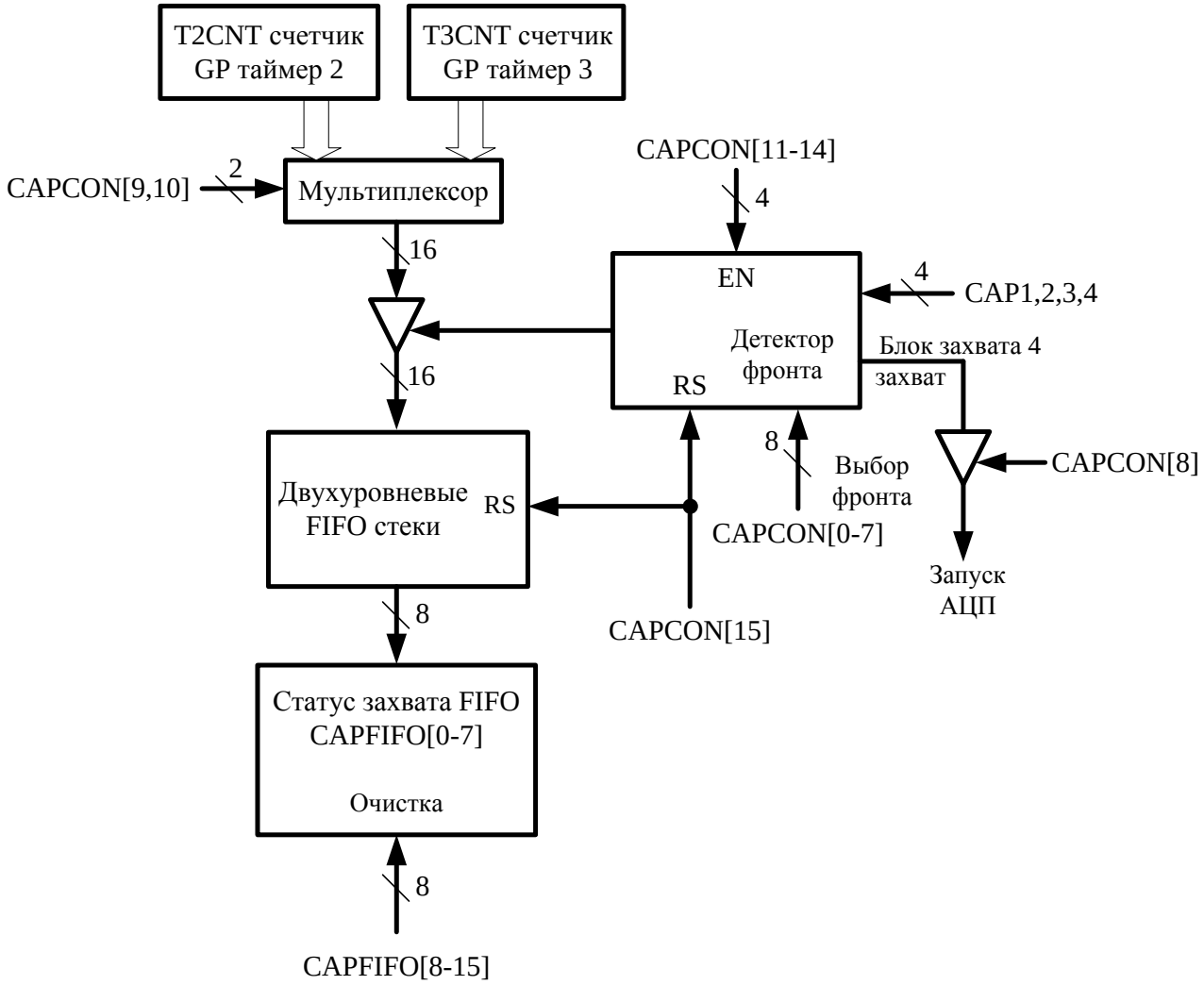


Рисунок 78 – Структурная схема блока захвата

Особенности блоков захвата

Блоки захвата имеют следующие особенности:

– Один 16-битный управляющий регистр захвата, CAPCON (R/W).

– Один 16-битный управляющий статусный FIFO регистр захвата, CAPFIFO (8 старших разрядов доступны только для чтения, 8 младших разрядов доступны только для записи).

– Выбор GP таймера 2 или GP таймера 3 в качестве временной оси.

– Четыре 16-битных двухуровневых FIFO стека (один для каждого блока захвата).

– Четыре входа захвата с триггерами Шмитта: CAP1, CAP2, CAP3 и CAP4 (один вход на каждый блок захвата). Все входы синхронизированы с тактовым сиг-

налом процессора. Для требуемого захвата переключения вход должен удерживаться в текущем уровне в течение двух передних фронтов тактового сигнала процессора. Входы CAP1 и CAP2 так же могут быть использованы как входы для цепей QEP.

– Задаваемый пользователем вид детектирования переключения (передний фронт, задний фронт или оба фронта).

– Четыре маскируемых флага (один для каждого блока захвата).

Функционирование блоков захвата

Выбор временной оси блока захвата

Любой из GP таймера 2 или GP таймера 3 может быть выбран БЗ 1 и БЗ 2 или БЗ 3 и БЗ 4. Таким образом, два различных GP таймера могут быть использованы в одно и то же время: один для БЗ 1 и БЗ 2 и другой для БЗ 3 и БЗ 4.

Операция захвата не оказывает влияния на работу любого GP таймера или на операции сравнения (compare)/PWM, связанные с любым GP таймером.

Операция захвата

После разрешения блока захвата, требуемое переключение на соответствующем вводе приводит к тому, что значение счетчика выбранного GP таймера будет записано в соответствующем стеке FIFO. В то же время устанавливается соответствующий флаг прерывания и будет сформирован запрос прерывания, если флаг не маскирован, и не выполняются никакие другие, немаскированные прерывания с более высоким приоритетом в той же группе. Соответствующие статусные биты в CAPFIFO установлены для отображения нового состояния стека FIFO каждый раз, когда новое значение счётчика захватывается в стеке FIFO. Длительность задержки между моментом переключения на входе захвата и моментом фиксации значения счетчика выбранного GP таймера составляет 3,5 – 4,5 периодов тактового сигнала.

Все регистры блока захвата сбрасываются в 0, когда вход сброса устанавливается в низкое состояние.

Настройка блока захвата

Для корректной работы блока захвата требуется следующая настройка регистров:

1 Инициализация CAPFIFO. Очистка соответственных статусных разрядов.

2 Установка выбранного GP таймера в одном из его режимов работы.

3 Установка связанного регистра сравнения GP таймера или регистра периода GP таймера, если необходимо.

4 Установка CAPCON.

Регистры блока захвата

Работа блоков захвата управляется двумя 16-битными регистрами CAPCON и CAPFIFO. Регистры T2CON и T3CON также используются, если для цепей захвата используется временная ось GP таймера 2 или GP таймера 3. CAPCON также используется для управления работой цепей QEP. Как и остальные регистры модуля EV, все эти регистры картированы в памяти данных и, следовательно, могут обрабатываться программой пользователя как ячейки памяти данных. В таблице 56 показаны адреса этих регистров.

Регистр управления захвата (CAPCON)

Формат CAPCON приведен на рисунке 79.

15	14-13	12	11	10	9	8
CAPRES	CAPQEPN	CAP3EN	CAP4EN	CAP34TSEL	CAP12TSEL	CAP4TOADC
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7-6	5-4	3-2	1-0
CAP1EDGE	CAP2EDGE	CAP3EDGE	CAP4EDGE
RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 79 – Регистр управления захвата (CAPCON) – адрес 7420h

Бит 15 CAPRES. Сброс захвата.
0 = Очистка всех регистров блоков сравнения и цепи QEP до 0.
1 = Нет событий.

Биты 14-13 CAPQEPN. Управление БЗ 1, БЗ 2 и цепью QEP.
00 = Запрещение БЗ 1, БЗ 2 и цепи QEP. FIFO стек сохраняет свое содержимое.
01 = Разрешение БЗ 1 и БЗ 2. Запрещение цепи QEP.
10 = Зарезервировано.
11 = Разрешение цепи QEP. Запрещение БЗ 1 и БЗ 2. Биты 4-7 и 9 игнорируются.

Бит 12 CAP3EN. Управление БЗ 3.
0 = Запрещение БЗ 3. FIFO стек БЗ 3 сохраняет его содержимое.
1 = Разрешение БЗ 3.

Бит 11 CAP4EN. Управление БЗ 4.
0 = Запрещение БЗ 4. FIFO стек БЗ 4 сохраняет его содержимое.
1 = Разрешение БЗ 4.

Бит 10 CAP34TSEL. Выбор GP таймера для БЗ 3 и БЗ 4.
0 = Выбор GP таймер 2.
1 = Выбор GP таймер 3.

Бит 9 CAP12TSEL. Выбор GP таймера для БЗ 1 и БЗ 2.
0 = Выбор GP таймер 2.
1 = Выбор GP таймер 3.

Бит 8 CAP4TOADC. Событие БЗ 4 запускающее модуля АЦП
0 = Нет событий.
1 = Запуск модуля АЦП при установке флага CAP4INT.

Биты 7-6 CAP1EDGE. Управление видом детектирования фронта для БЗ 1.
00 = Нет детектирования.
01 = Детектирование переднего фронта.
10 = Детектирование заднего фронта.
11 = Детектирование обоих фронтов.

Биты 5-4 CAP2EDGE. Управление видом детектирования фронта для БЗ 2.
00 = Нет детектирования.
01 = Детектирование переднего фронта.
10 = Детектирование заднего фронта.
11 = Детектирование обоих фронтов.

Биты 3-2 CAP3EDGE. Управление видом детектирования фронта для БЗ 3.
00 = Нет детектирования.
01 = Детектирование переднего фронта.
10 = Детектирование заднего фронта.
11 = Детектирование обоих фронтов.

Биты 1-0 CAP4EDGE. Управление видом детектирования фронта для БЗ 4.
00 = Нет детектирования.
01 = Детектирование переднего фронта.
10 = Детектирование заднего фронта.
11 = Детектирование обоих фронтов.

Статусный регистр захвата FIFO (CAPFIFO)

CAPFIFO содержит статусные биты для каждого из четырёх FIFO стеков блоков захвата. Формат CAPFIFO приведен на рисунке 80. 8 младших разрядов CAPFIFO имеют однозначное соответствие 8 старшим разрядам CAPFIFO. Одна запись в CAPFIFO[x] (при x = 0, 1, … или 7) очищает биты CAPFIFO[x+8]. CAPFIFO[x] при x = 0, 1, … или 7 имеют доступ только для записи и CAPFIFO[y] при y = 8, 9, … или 15 имеют доступ только для чтения.

15-14	13-12	11-10	9-8
CAP4FIFO	CAP3FIFO	CAP2FIFO	CAP1FIFO
R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
CAPFIFO15	CAPFIFO14	CAPFIFO13	CAPFIFO12	CAPFIFO11	CAPFIFO10	CAPFIFO9	CAPFIFO8
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 80 – Статусный регистр захвата FIFO (CAPFIFO) – адрес 7422h

Биты 15-14 CAP4FIFO. Статус CAP4FIFO. Только чтение.
00 = Пусто.
01 = Была одна запись.
10 = Было две записи.
11 = Было две записи и произошла ещё одна; первая запись утеряна.

Биты 13-12 CAP3FIFO. Статус CAP3FIFO. Только чтение.
00 = Пусто.
01 = Была одна запись.
10 = Было две записи.
11 = Было две записи и произошла ещё одна; первая запись утеряна.

Биты 11-10 CAP2FIFO. Статус CAP2FIFO. Только чтение.
00 = Пусто.
01 = Была одна запись.
10 = Было две записи.
11 = Было две записи и произошла ещё одна; первая запись утеряна.

Биты 9-8 CAP1FIFO. Статус CAP1FIFO. Только чтение.
00 = Пусто.
01 = Была одна запись.
10 = Было две записи.
11 = Было две записи и произошла ещё одна; первая запись утеряна.

Бит 7 CAPFIFO15. Очистка 15 бита CAP1FIFO. Только чтение.
0 = Нет действия
1 = Очистка 15 бита

Бит 6 CAPFIFO14. Очистка 14 бита CAP1FIFO. Только чтение.
0 = Нет действия
1 = Очистка 14 бита

Бит 5 CAPFIFO13. Очистка 13 бита CAP1FIFO. Только чтение.
0 = Нет действия
1 = Очистка 13 бита

Бит 4 CAPFIFO12. Очистка 12 бита CAP1FIFO. Только чтение.
0 = Нет действия
1 = Очистка 12 бита

Бит 3 CAPFIFO11. Очистка 11 бита CAP1FIFO. Только чтение.
0 = Нет действия
1 = Очистка 11 бита

Бит 2 CAPFIFO10. Очистка 10 бита CAP1FIFO. Только чтение.
0 = Нет действия
1 = Очистка 10 бита

Бит 1 CAPFIFO9. Очистка 9 бита CAP1FIFO. Только чтение.
0 = Нет действия
1 = Очистка 9 бита

Бит 0 CAPFIFO8. Очистка 8 бита CAP1FIFO. Только чтение.
0 = Нет действия
1 = Очистка 8 бита

FIFO стеки блока захвата

FIFO стек, связанный с каждым блоком захвата

Каждый блок захвата имеет специализированный двухуровневый FIFO стек. Регистр верхнего уровня FIFO стеков имеет доступ только на чтение, это регистр, который содержит старое значение счётчика, захваченное соответствующим блоком захвата. Поэтому, при чтении FIFO стека блока захвата всегда считывает старое значение счётчика, захваченное в стек. Когда считывается старое значение счётчика в верхнем регистре FIFO стека, новое значение счётчика в нижнем регистре стека, если оно есть, помещается в верхний регистр.

Первый захват

Значение счётчика выбранного GP таймера, захваченное блоком захвата, когда на его входе происходит переключение, будет записано в верхний регистр стека, если стек пуст. Одновременно, соответствующие статусные биты установятся в (01). Статусные биты сбрасываются до (00), если производилось чтение FIFO стека перед совершением следующего захвата.

Второй захват

Если следующий захват происходит до чтения предыдущего захваченного значения счётчика, новое захваченное значение счётчика поступает в нижний регистр. Между тем, соответствующие статусные биты установятся в (10). Если FIFO стек был считан до возникновения другого захвата, старое значение счётчика в верхнем регистре будет считано, новое значение счётчика в нижнем регистре стека переместится в верхний регистр и установится соответствующий статусный бит в (01).

Третий захват

Если захват происходит когда в FIFO стеке уже есть два значения счётчика, старое значение счётчика в верхнем регистре будет вытеснено и потеряно. Значение счётчика в нижнем регистре стека переместится в верхний регистр, новое значение счётчика будет записано в нижний регистр стека, и произойдет установка статусных разрядов в (11) для указания на то, что одно или более старых значений счётчика были потеряны.

В примере 8 показывается программа инициализации для блоков захвата.

Пример 8 – Программа инициализации для блоков захвата.

```
;*********************************************************
; Инициализация счётных регистров                        *
;*********************************************************
    SPLK #00000H, T1CNT ; Счетчик GP таймера 1
    SPLK #00000H, T2CNT ; Счетчик GP таймера 2
    SPLK #00000H, T3CNT ; Счетчик GP таймера 3
;*********************************************************
; Инициализация регистров периода                        *
;*********************************************************
    SPLK #00011H, T1PER ; Период GP таймера 1
    SPLK #07fffH, T2PER ; Период GP таймера 2
    SPLK #00011H, T3PER ; Период GP таймера 3
;*********************************************************
; Инициализация регистров сравнения                      *
;*********************************************************
    SPLK #00004H, T1CMP ; Значение сравнения (compare) ; GP тай-
мера 1
    SPLK #00006H, T2CMP ; Значение сравнения (compare) ; GP тай-
мера 2
    SPLK #00008H, T3CMP ; Значение сравнения (compare) ; GP тай-
мера 3
;*********************************************************
; Программирование GPTCON                                *
;*********************************************************
    SPLK #0000000001010101b, GPTCON ;Установка управления GP
                                     ;таймера 1
* bits 12-11 00: Нет событий запускающих модуля АЦП от GP тайме-
ра 3
* bits 10-9  00: Нет событий запускающих модуля АЦП от GP тайме-
ра 2
* bits 8-7   00: Нет событий запускающих модуля АЦП от GP тайме-
ра 1
* bit 6       1:  Разрешение выводов сравнения (compare) GP тай-
мер
* bits 5-4    01: GP таймер 3 - выход сравнения (compare) с ак-
тивным низким уровнем
* bits 3-2    01: GP таймер 2 - выход сравнения (compare) с ак-
тивным низким уровнем
* bits 1-0    01: GP таймер 1 выход сравнения (compare) с актив-
ным низким уровнем
;*********************************************************
; Очистка всех прерываний модуля EV пред запуском работы
     *
;*********************************************************
;*********************************************************
    SPLK #0ffffh, IFRA ; Очистка всех флагов прерываний группы A
    SPLK #0ffffh, IFRB ; Очистка всех флагов прерываний группы B
    SPLK #0ffffh, IFRC ; Очистка всех флагов прерываний группы C
;*********************************************************
; Конфигурирование T2CON и запуск GP таймера 2           *
```

```
;*********************************************************
    SPLK #1001000001000010b, T2CON ;Установка управления GP
                                   ;таймер 2
* bit 15     1:   FREE = 1
* bit 14     0:   SOFT = 0
* bits 13-11 010: Продолжительный прямой счётный режим
* bits 10-8  000: Делитель = /1
* bit 7      0:   Использование собственного разрешения ENABLE
* bit 6      1:   Разрешение операций счета
* bits 5-4   00:  Выбор внутреннего тактового сигнала CLK
* bits 3-2   00:  Загрузка регистра сравнения GP таймера при
                  обнулении
* bit 1      1:   Разрешение сравнения (compare) GP таймера
* bit 0      0:   Использование собственного периода PR
    NOP
    NOP
    NOP
    NOP
;*********************************************************
; Конфигурирование CAPCON и разрешение операции захвата    *
;*********************************************************
SPLK #1011110001010101b, CAPCON ; Управление захвата
* bit 15    1:  Нет действия
* bit 14-13 01: Разрешение БЗ 1 и БЗ 2 и запрещение QEP
* bit 12    1:  Разрешение БЗ 3
* bit 11    1:  Разрешение БЗ 4
* bit 10    1:  GP таймер 3 – временная ось для БЗ 3 и БЗ 4
* bit 9     0:  GP таймер 2 – временная ось для БЗ 1 и БЗ 2
* bit 8     0:  Нет БЗ 4 событий запускающих модуля АЦП
* bits 7-6  01: БЗ 1 детектирует передний фронт
* bits 5-4  01: БЗ 2 детектирует передний фронт
* bits 3-2  01: БЗ 3 детектирует передний фронт
* bits 1-0  01: БЗ 4 детектирует передний фронт
;*********************************************************
; Конфигурирование T3CON                                   *
;*********************************************************
    SPLK #1001000011000010b, T3CON ; Установка управления GP
                                   ;таймера 3
* bit 15     1:   FREE = 1
* bit 14     0:   SOFT = 0
* bits 13-12 010: Продолжительный прямой счётный режим
* bits 10-8  000: Делитель = /1
* bit 7       1:   Использование разрешения ENABLE от GP таймера
1
* bit 6      1:   Разрешение операций счета
* bits 5-4   00:  Выбор внутреннего тактового сигнала CLK
* bits 3-2   00:  Загрузка регистра сравнения GP таймера при
                  опустошении
* bit 1      1:   Разрешение сравнения (compare) GP таймера
* bit 0      0:   Использование собственного периода PR
;*********************************************************
; Конфигурирование T1CON и запуск GP таймера 1 и GP таймера 3
```

```
;*********************************************************
    SPLK #1001000001000010b, T1CON ; Установка управления GP
                                    ;таймер 1.
                          ;Запуск GP таймера 1 и GP таймера 2
* bit 15     1:   FREE = 1
* bit 14     0:   SOFT = 0
* bits 13-12 010: Продолжительный прямой счётный режим
* bits 10-8  000: Делитель = /1
* bit 7      0:   Зарезервировано
* bit 6      1:   Разрешение операций счета
* bits 5-4   00:  Выбор внутреннего тактового сигнала CLK
* bits 3-2   00:  Загрузка регистра сравнения GP таймера при
                  обнулении
* bit 1      1:   Разрешение сравнения (compare) GP таймера
* bit 0      0:   Зарезервировано
;*********************************************************
; Чтение захваченных значений и вычисление разницы         *
;*********************************************************
   LAR AR2, #01eh ; Зациклить 16 раз
   MAR *, AR1     ; ARP<=AR1 указание в результирующий отчет
LOOP
   LACC CAPFIFO   ; Чтение статуса захвата FIFO
   AND #0000001000000000b
                  ; Получено >=2 запись в FIFO 1
   BZ LOOP        ; Нет, обойти
   LACC FIFO1     ; Чтение первой записи захвата FIFO 1
   SACL *+        ; Сохранить
   LACC FIFO1     ; Первая запись-Вторая запись
   SACL *-        ; Сохранить
   SUB *+         ; Вычисление разницы
   MAR *+, AR3     ; Настроить указатель и указать в отчете раз-
личий
   SACL *+, AR2   ; Сохранить различия и указания в управлении
                  ; циклом
DLOOP B  DLOOP
```

Прерывания захвата

Когда происходит захват, соответствующий флаг прерывания сравнения будет установлен и будет сформирован запрос прерывания к ядру ЦПОС, если флаг не маскирован, и не выполняются никакие другие, немаскированные прерывания с более высоким приоритетом в той же группе. Захваченное значение счётчика, таким образом, может считываться с обслуживающей программы прерывания, если используется прерывание. Если прерывание не требуется, то или флаг прерывания или статусный бит может быть опрошен для выяснения совершения события захвата и, таким образом, может быть считано значение счётчика.

13.2.9 Цепи «квадратурной» обработки сигналов импульсного датчика положения QEP

Модуль EV имеет в своём составе цепь «квадратурной» обработки сигналов импульсного датчика положения (QEP). Цепь QEP, когда разрешена, считает входные «квадратурно» кодированные импульсы на вводах CAP1/QEP1 и CAP2/QEP2.

Цепь QEP может быть использована для связи с оптическим датчиком положения для получения информации о позиции и скорости вращающегося механизма.

Вводы QEP

Два ввода QEP являются общими для БЗ 1, БЗ 2 и цепи QEP. Для разрешения цепи QEP и запрещения БЗ 1 и БЗ 2 необходима правильная конфигурация битов в CAPCON, которая, таким образом, приписывает двум связанным входам использование цепью QEP.

Временная ось цепи QEP

Временная ось цепи QEP может быть обеспечена GP таймером 2, GP таймером 3 или GP таймером 2 и GP таймером 3, соединённых как 32-битный таймер. Выбор осуществляется формированием бит в T2CON или T3CON. Выбранный GP таймер или 32-битный таймер должен быть установлен в режим направленного прямого/обратного счета с цепью QEP в качестве источника тактового сигнала. На рисунке 81 представлена структурная схема цепи QEP.

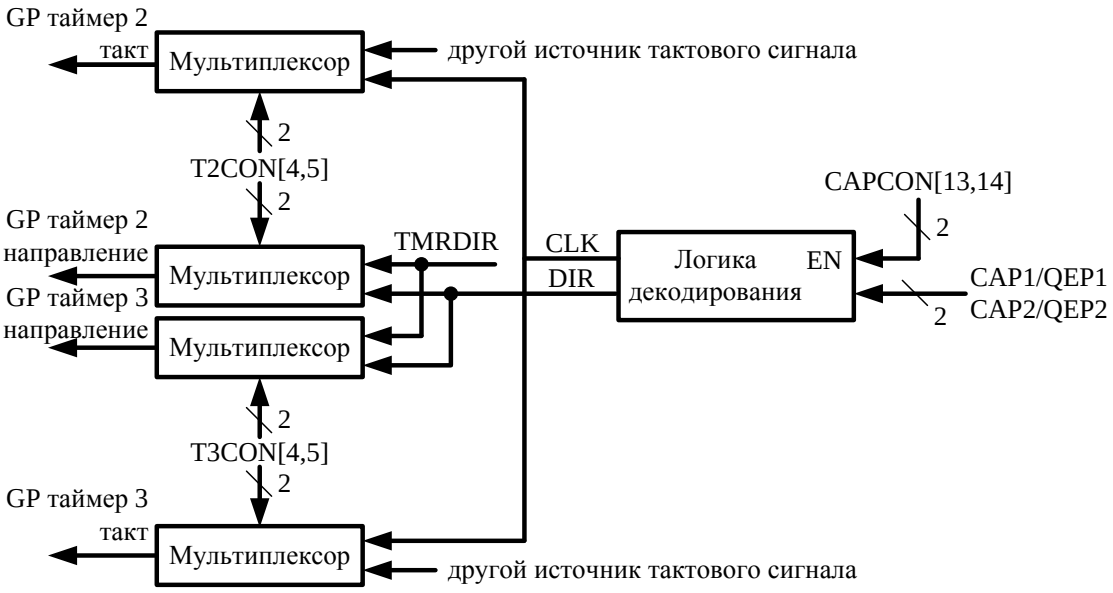


Рисунок 81 – Структурная схема цепи QEP

Декодирование QEP

«Квадратурно» кодированные импульсы – это две последовательности импульсов с переменной частотой и фиксированным фазовым сдвигом на четверть периода (90 градусов). Когда они формируются оптическим датчиком положения на валу электродвигателя, то направление вращения двигателя может быть установлено определением того, какая из последовательностей является лидирующей последовательностью, а дуговая координата и скорость могут быть установлены отсчетом и частотой импульсов.

Цепь QEP

Логика распознавания направления в цепи QEP модуля модуля EV определяет, какая из последовательностей является лидирующей последовательностью. Затем она формирует сигнал направления, как вход направления для выбранного таймера. Выбранный таймер считает в прямом направлении, если вход CAP1/QEP1 является лидирующей последовательностью, и считает в обратном направлении, если лидирующей последовательностью является вход CAP2/QEP2.

Цепь QEP подсчитывает импульсы двух «квадратурно» кодированных входов по обоим фронтам. Поэтому частота формируемого тактового сигнала для GP таймера, вчетверо больше, чем у каждой входной последовательности. Этот формируемый тактовый сигнал соединён с тактовым входом выбранного GP таймера или 32-битного таймера.

Пример декодирования «квадратурно» обработанных импульсов

На рисунке 82 представлен пример «квадратурно» кодированных импульсов, выявленное прямое и обратное направление счёта и тактовый сигнал.

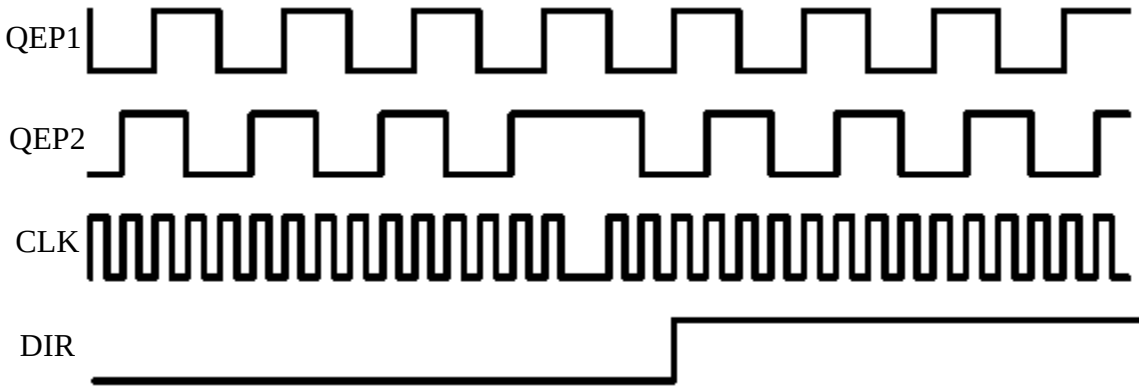


Рисунок 82 – «Квадратурно» кодированные импульсы и декодированные тактовый сигнал и направление

Счёт QEP

GP таймер используемый с цепью QEP

Выбранный GP таймер всегда начинает счёт со своего текущего значения, содержащегося в счётчике. Требуемое значение может быть загружено в выбранный счетчик GP таймер перед разрешением операции QEP. Когда цепь QEP выбрана как источник тактового сигнала, выбранный таймер будет игнорировать вводы TMRDIR и TMRCLK.

Режим счёта GP таймера при работе QEP

Важно отметить, что режим направленного прямого/обратного счёта выбранного GP таймера с источником тактового сигнала от цепи QEP отличается от нормального режима направленного прямого/обратного счёта. Когда таймер, выбранный для операций QEP, считает в прямом направлении до значения периода,

GP таймер не останавливается, вместо этого он будет считать в прямом направлении, пока не изменится направление счёта. Если выбрано прямое направление счёта и начальное значение счетчика GP таймера больше значения в регистре периода, таймер будет считать в прямом направлении до значения FFFFh (или FFFF FFFFh, при использовании 32-битного таймера) и перебирать значения до 0. Когда выбрано обратное направление счёта, и таймер считает в обратном направлении до 0, GP таймер будет перебирать значения до FFFFh (или FFFF FFFFh, при использовании 32-битного таймера).

Прерывания GP таймера и связанные выходы сравнения (compare) при работе QEP

Если цепь QEP является источником тактового сигнала, флаги прерывания периода, опустошения, переполнения и сравнения для GP таймера формируются при соответствующих совпадениях, даже если не происходит никакого переключения на выходе сравнения (compare) выбранного GP таймера или любого другого блока сравнения, использующего этот таймер как свою ось времени. Запрос прерывания может быть сформирован флагом, если флаг установлен, не маскирован, и не выполняются никакие другие, немаскированные прерывания с более высоким приоритетом в той же группе.

Установка регистров для цепи QEP

Для запуска работы цепи QEP требуется:

1 Загрузить регистры счётчика, периода и сравнения выбранного GP таймера требуемыми значениями, если необходимо.

2 Сформировать T2CON или T3CON для установки GP таймера 2, GP таймера 3 или GP таймера 2 и GP таймера 3 в режим направленного прямого/обратного счёта или 32-битный режим с цепью QEP, как источником тактового сигнала и разрешить выбранный таймер.

3 Сформировать CAPCON для разрешения работы QEP.

На примере 9 показана программа инициализации для работы QEP.

Пример 9 – Установка регистров для цепи QEP.

```
;*************************************************************
; Инициализация работы QEP *
;*************************************************************
     LDPK #232                ; DP => Регистры модуля EV
;*************************************************************
; Формирование GPTCON
;*************************************************************
    SPLK #1110001011110000b, CAPCON     ; Установка управления
                                        ; GP таймера
    ;      ||||||||||||||||
    ;      FEDCBA9876543210
* bits 0-1   00: Нет определения для БЗ 4
```

```
* bits 2-3   00: Нет определения для БЗ 3
* bits 4-5   11: БЗ 2 определяет оба фронта
* bits 6-7   11: БЗ 1 определяет оба фронта
* bit 8      0:  Нет БЗ 4 событий запускающих модуля АЦП
* bit 9      1:  GP таймер 3 - временная ось для БЗ 1 и БЗ 2
* bit 10     0:  GP таймер 2 - временная ось для БЗ 3 и БЗ 4
* bit 11     0:  Запрещение БЗ 4
* bit 12     0:  Запрещение БЗ 3
* bits 13-14 11: Разрешение QEP
* bit 15     1:  Нет действия
;*****************************************************************
; Формирование регистра счетчика T3CNT *
;*****************************************************************
    SPLK #0000h, T3CNT
;*****************************************************************
; Формирование регистра периода T3PER *
;*****************************************************************
    SPLK #00FFh, T3PER
;*****************************************************************
; Формирование T3CON и запуск GP таймера 3 *
;*****************************************************************
    SPLK #1001100001110000b, T1CON    ; Установка управления GP
                                      ; таймер 3
    ;     ||||||||||||||||
    ;     FEDCBA9876543210
;
* bit 0      0:   Использование собственного периода PR
* bit 1      0:   Запрещение сравнения (compare) GP таймера
* bits 2-3   00:  Загрузка регистра сравнения GP таймера при
                  обнулении
* bits 4-5   11:  Выбор QEP
* bit 6      1:   Разрешение операций счета
* bit 7      0:   Использование собственного разрешения ENABLE
* bits 8-10  000: Делитель = /1 (При QEP, делитель всегда 1)
* bits 11-13 011: Направленный прямой/обратный счётный режим для
QEP
* bit 14     0:   SOFT = 0
* bit 15     1:   FREE = 1
```

13.2.10 Прерывания модуля EV

Организация прерываний 1867ВЦ10Т

Организация прерываний ядра 1867ВЦ10Т

Прерывание NMI имеет наивысший приоритет; следующее по приоритету – INT1, затем приоритет убывает от INT1 до INT6. INT6 имеет низший приоритет. Каждое прерывание соответствует биту в регистре флагов прерываний ядра (IFR), и каждое маскируемое прерывание соответствует биту в регистре маски прерываний ядра (IMR). Маскируемое прерывание маскируется (не формирует прерывание

к ядру), когда соответствующий бит в IMR установлен в 0. К тому же ядро имеет глобальный бит маски прерывания (INTM) в статусном регистре ST0. Когда INTM установлен в 1, все маскируемые прерывания маскируются.

Обработка прерываний ядром 1867ВЦ10Т

Когда на входе прерывания к ядру происходит переключение с верхнего уровня на нижний, соответствующий бит флага прерывания в IFR устанавливается в 1. Прерывание к ядру формируется этим флагом, если он не маскирован, прерывания глобально разрешены (INTM = 0), и не выполняются никакие другие, немаскированные прерывания с более высоким приоритетом (то есть, не выставлены флаги никаких других немаскированных прерываний с более высоким приоритетом). Флаг очищается аппаратно, сразу после того, как запрос прерывания принят ядром. Флаг прерывания также может быть очищен пользовательской программой записью 1 в соответствующий разряд.

Запрос и служба прерываний модуля EV

Группы прерываний

Прерывания, формируемые модулем EV, распределены на три группы: A, B и C. Группа прерываний модуля EV A формирует запросы прерывания к ядру на INT2. Группы прерываний модуля EV B и C формируют запросы прерывания к ядру на INT3 и INT4, соответственно. В таблице 71 приведены все прерывания модуля EV их приоритет и группирование. Имеется регистр флага прерывания и регистр маски прерывания для каждой группы прерываний модуля EV: EVIFRA, EVIFRB, и EVIFRC, и EVIMRA, EVIMRB или EVIMRC. Флаг в EVIFRx (x = A, B и C) маскирован (не формирует прерывание к ядру), если соответствующий бит в EVIMRx установлен в 0.

Существует 8-битный регистр вектора прерывания EVIVRx (x = A, B и C), связанный с каждой группой прерываний. Соответствующий регистр вектора прерывания может быть считан с обслуживающей программы прерывания (ISR), когда запрос прерывания, формируемый группой прерываний, принят ядром. Значение (вектор) в регистре вектора прерывания определяет, какое незавершенное прерывание в группе имеет высший приоритет.

Таблица 62 – Прерывания модуля EV

Группа	Прерывание	Приоритет внутри группы	Вектор	Описание/источник
A	PDPINT	1(высший)	0020h	Прерывание защиты электропитания
	CMP1INT	2	0021h	Прерывание сравнения блока полного сравнения 1
	CMP2INT	3	0022h	Прерывание сравнения блока полного сравнения 2
	CMP3INT	4	0023h	Прерывание сравнения блока полного сравнения 3
	SCMP1INT	5	0024h	Прерывание сравнения блока простого сравнения 1
	SCMP2INT	6	0025h	Прерывание сравнения блока простого сравнения 2
	SCMP3INT	7	0026h	Прерывание сравнения блока простого сравнения 3
	T1PINT	8	0027h	Прерывание периода GP таймера 1
	T1CINT	9	0028h	Прерывание сравнения GP таймера 1
	T1UFINT	10	0029h	Прерывание обнуления GP таймера 1
	T1OFINT	11(низший)	002Ah	Прерывание переполнения GP таймера 1
B	T2PINT	1(высший)	002Bh	Прерывание периода GP таймера 2
	T2CINT	2	002Ch	Прерывание сравнения GP таймера 2
	T2UFINT	3	002Dh	Прерывание обнуления GP таймера 2
	T2OFINT	4	002Eh	Прерывание переполнения GP таймера 2
	T3PINT	5	002Fh	Прерывание периода GP таймера 3
	T3CINT	6	0030h	Прерывание сравнения GP таймера 3
	T3UFINT	7	0031h	Прерывание обнуления GP таймера 3
	T3OFINT	8(низший)	0032h	Прерывание переполнения GP таймера 3
C	CAP1INT	1(высший)	0033h	Прерывание блока захвата 1
	CAP2INT	2	0034h	Прерывание блока захвата 2
	CAP3INT	3	0035h	Прерывание блока захвата 3
	CAP4INT	4(низший)	0036h	Прерывание блока захвата 4

Формирование прерываний

Когда в модуле EV происходит событие прерывания, соответствующий флаг прерывания в одном из регистров флага прерывания модуля EV устанавливается в 1. Запрос прерывания формируется на соответствующем INTx, если флаг локально не маскирован (соответствующий бит в EVIMRx установлен в 1) и не выполняются никакие другие немаскированные прерывания с более высоким приоритетом в той же группе.

Вектор прерываний

Вектор прерывания, соответствующий флагу прерывания, имеющий более высокий приоритет среди установленных флагов, загружается в аккумулятор, когда вектор прерывания группы прерываний модуля EV считывается после того, как за-

прос прерывания сформирован группой. При считывании вектора прерывания флаг очищается. Флаг прерывания так же может быть очищен пользовательской программой прямой записью 1 в соответствующий бит.

Нулевое значение будет возвращено, когда регистр вектора прерывания группы прерываний считан, и в группе не установлен флаг прерывания. Это предотвращает распознавание потерянных прерываний как прерываний модуля EV.

Обработка прерываний

После получения запроса прерывания модуля EV, EVIVRx должен быть считан в аккумулятор и сдвинут влево на один или более бит. Далее, относительный адрес (начало таблицы записи прерываний) добавляется к аккумулятору. Инструкция BACC используется для ветвления к правильной записи в массиве. Другая ветвь идет на ISR для получения специальных источников. Этот процесс приводит к типичной задержке прерываний на 20 периодов тактового сигнала процессора (25, если требуется сохранение минимального контекста) от времени формирования прерывания до поступления в специальный источник первой инструкции от ISR. Эта задержка может быть уменьшена до восьми периодов тактового сигнала процессора, если разрешено только одно прерывание в группе прерываний модуля EV. Если пространство памяти не важно, задержка может быть уменьшена до 16 периодов тактового сигнала процессора без требования к разрешению только одного прерывания в группе прерываний модуля EV.

Регистры флагов прерываний модуля EV

Адреса регистров прерываний модуля EV показаны в таблице 57. Все регистры рассматриваются как 16-битные картированные в память регистры. Все неиспользуемые биты считываются нулями. Запись в неиспользуемые биты не имеет эффекта. Как только EVIFRx становятся доступными для чтения, возникновение событий прерывания может быть отслежено программой с помощью опроса соответствующего бита в EVIFRx, когда прерывание маскировано.

Регистр флага прерывания модуля EV A (EVIFRA)

Описание бит EVIFRA показано на рисунке 83.

15-11	10	9	8
Зарезервировано	T1OFINT	T1UFINT	T1CINT
	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
T1PINT	SCMP3INT	SCMP2INT	SCMP1INT	CMP3INT	CMP2INT	CMP1INT	PDPINT
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 83 – Регистр флага прерывания модуля EV A (EVIFRA) — адрес 742Fh

Биты 15-11 Зарезервированы. При чтении – нули; запись не имеет эффекта.

Бит 10 T1OFINT. Прерывание переполнения GP таймера 1
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 9 T1UFINT. Прерывание обнуления GP таймера 1
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 8 T1CINT. Прерывание сравнения GP таймера 1
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 7 T1PINT. Прерывание периода GP таймера 1
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 6 SCMP3INT. Прерывание простого сравнения 3
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 5 SCMP2INT. Прерывание простого сравнения 2
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 4 SCMP1INT. Прерывание простого сравнения 1
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 3 CMP3INT. Прерывание полного сравнения 3

Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 2 CMP2INT. Прерывание полного сравнения 2
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.

Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 1 CMP1INT. Прерывание полного сравнения 1
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 0 PDPINT. Прерывание защиты электропитания
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Регистр флага прерывания модуля EV B (EVIFRB)

Описание бит EVIFRB показано на рисунке 84.

15-8	7	6	5	4	3	2	1	0
Зарезервировано	T3OFINT	T3UFINT	T3CINT	T3PINT	T2OFINT	T2UFINT	T2CINT	T2PINT
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 84 – Регистр флага прерывания модуля EV B (EVIFRB) — адрес 7430h

Биты 15-8 Зарезервированы. При чтении – нули; запись не имеет эффекта.

Бит 7 T3OFINT. Прерывание переполнения GP таймера 3
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 6 T3UFINT. Прерывание обнуления GP таймера 3
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта

1 = Сброс флага

Бит 5 T3CINT. Прерывание сравнения GP таймера 3
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 4 T3PINT. Прерывание периода GP таймера 3
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 3 T2OFINT. Прерывание переполнения GP таймера 2
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 2 T2UFINT. Прерывание опустошения GP таймера 2
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 1 T2CINT. Прерывание сравнения GP таймера 2
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 0 T2PINT. Прерывание периода GP таймера 2
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Регистр флага прерывания модуля EV C (EVIFRC)

Описание бит EVIFRC показано на рисунке 85.

15-4	3	2	1	0
Зарезервировано	CAP4INT	CAP3INT	CAP2INT	CAP1INT
	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 85 – Регистр флага прерывания модуля EV C (EVIFRC) – адрес 7431h

Биты 15-4 Зарезервированы. При чтении – нули; запись не имеет эффекта.

Бит 3 CAP4INT. Прерывание захвата 4
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 2 CAP3INT. Прерывание захвата 3
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 1 CAP2INT. Прерывание захвата 2
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Бит 0 CAP1INT. Прерывание захвата 1
Чтение: 0 = Флаг сброшен.
1 = Флаг установлен.
Запись: 0 = Нет эффекта
1 = Сброс флага

Регистр маски прерывания модуля EV A (EVIMRA)

Описание бит модуля EVIMRA показано на рисунке 86.

15-11	10	9	8
Зарезервировано	T1OFINT ENABLE	T1UFINT ENABLE	T1CINT ENABLE
	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
T1PINT ENABLE	SCMP3INT ENABLE	SCMP2INT ENABLE	SCMP1INT ENABLE	CMP3INT ENABLE	CMP2INT ENABLE	CMP1INT ENABLE	PDPINT ENABLE
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 86 – Регистр маски прерывания модуля EV A (EVIMRA) — адрес 742Ch

Биты 15-11 Зарезервированы. Читаются как 0, запись возможна
Бит 10 T1OFINT ENABLE

0 = Запрещено
1 = Разрешено

Бит 9 T1UFINT ENABLE
0 = Запрещено
1 = Разрешено

Бит 8 T1CINT ENABLE
0 = Запрещено
1 = Разрешено

Бит 7 T1PINT ENABLE
0 = Запрещено
1 = Разрешено

Бит 6 SCMP3INT ENABLE
0 = Запрещено
1 = Разрешено

Бит 5 SCMP2INT ENABLE
0 = Запрещено
1 = Разрешено

Бит 4 SCMP1INT ENABLE
0 = Запрещено
1 = Разрешено

Бит 3 CMP3INT ENABLE
0 = Запрещено
1 = Разрешено

Бит 2 CMP2INT ENABLE
0 = Запрещено
1 = Разрешено

Бит 1 CMP1INT ENABLE
0 = Запрещено
1 = Разрешено

Бит 0 PDPINT ENABLE
0 = Запрещено
1 = Разрешено

Регистр маски прерывания модуля EV B (EVIMRB)

Описание бит EVIMRB показано на рисунке 87.

15-8	7	6	5	4	3	2	1	0
Зарезервировано	T3OFINT ENABLE	T3UFINT ENABLE	T3CINT ENABLE	T3PINT ENABLE	T2OFINT ENABLE	T2UFINT ENABLE	T2CINT ENABLE	T2PINT ENABLE
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 87 – Регистр маски прерывания модуля EV B (EVIMRB) — адрес 742Dh

Биты 15-8 Зарезервированы. Читаются как 0, запись не возможна.

Бит 7 T3OFINT ENABLE
0 = Запрещено
1 = Разрешено

Бит 6 T3UFINT ENABLE
0 = Запрещено
1 = Разрешено

Бит 5 T3CINT ENABLE
0 = Запрещено
1 = Разрешено

Бит 4 T3PINT ENABLE
0 = Запрещено
1 = Разрешено

Бит 3 T2OFINT ENABLE
0 = Запрещено
1 = Разрешено

Бит 2 T2UFINT ENABLE
0 = Запрещено
1 = Разрешено

Бит 1 T2CINT ENABLE
0 = Запрещено
1 = Разрешено

Бит 0 T2PINT ENABLE
0 = Запрещено
1 = Разрешено

Регистр маски прерывания модуля EV C (EVIMRC)

Описание бит EVIMRC показано на рисунке 88.

15-4	3	2	1	0
Зарезервировано	CAP4INT ENABLE	CAP3INT ENABLE	CAP2INT ENABLE	CAP1INT ENABLE
	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 88 – Регистр маски прерывания модуля EV C (EVIMRC) — адрес 742Eh

Биты 15-4 Зарезервированы. При чтении – нули; запись не имеет эффекта.

Бит 3 CAP4INT ENABLE
0 = Запрещено
1 = Разрешено

Бит 2 CAP3INT ENABLE
0 = Запрещено
1 = Разрешено

Бит 1 CAP2INT ENABLE
0 = Запрещено
1 = Разрешено

Бит 0 CAP1INT ENABLE
0 = Запрещено
1 = Разрешено

Регистр вектора прерывания модуля EV A (EVIVRA)

Описание бит EVIVRA показано на рисунке 89.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R – доступ чтения, -0 – значение после сброса

Рисунок 89 – Регистр вектора прерывания модуля EV A (EVIVRA) — адрес 7432h

Биты 15-6 Зарезервированы. При чтении – нули; запись не имеет эффекта.

Бит 5-0 D5 – D0. Вектор флага прерывания, который имеет более высокий приоритет среди установленных флагов в EVIFRA; равен 0, если нет установленных флагов прерывания в EVIFRA.

Регистр вектора прерывания модуля EV B (EVIVRB)

Описание бит EVIVRB показано на рисунке 90.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R – доступ чтения, -0 – значение после сброса

Рисунок 90 – Регистр вектора прерывания модуля EV B (EVIVRB) — адрес 7433h

Биты 15-6 Зарезервированы. При чтении – нули; запись не имеет эффекта.

Бит 5-0 D5 – D0. Вектор флага прерывания, который имеет более высокий приоритет среди установленных флагов в EVIFRB; равен 0, если нет установленных флагов прерывания в EVIFRB.

Регистр вектора прерывания модуля EV C (EVIVRC)

Описание бит EVIVRC показано на рисунке 91.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R – доступ чтения, -0 – значение после сброса

Рисунок 91 – Регистр вектора прерывания модуля EV C (EVIVRC) — адрес 7434h

Биты 15-6 Зарезервированы. При чтении – нули; запись не имеет эффекта.

Бит 5-0 D5 – D0. Вектор флага прерывания, который имеет более высокий приоритет среди установленных флагов в EVIFRC; равен 0, если нет установленных флагов прерывания в EVIFRC.

13.3 Модуль сдвоенного 12-битного аналогово-цифрового преобразователя АЦП

13.3.1 Обзор 12-битного модуля АЦП

Аналогово-цифровой преобразователь (АЦП) – это 12-битный преобразователь с внутренней схемой выборки и хранения. Модуль АЦП состоит из двух 12-битных аналогово-цифровых преобразователей с двумя встроенными схемами выборки и хранения. Все 16 аналоговых входных каналов доступны в ИС 1867ВЦ10Т.

Восемь аналоговых входов предназначены для каждого 8/1 аналогового мультиплексора – блока ввода модуля АЦП. Максимальное суммарное время преобразования модуля АЦП составляет 1,25 мкс. Опорное напряжение модуля АЦП подаётся с внешнего источника напряжения. Верхний и нижний уровни могут быть установлены любым напряжением меньшим или равным 3,3 В, присоединением V_{REFHI} и V_{REFLO} к соответствующим вводам. Вводы ∩VCC2 и ∩GND2 должны быть соединены с 3,3 В и аналоговой «землей», соответственно.

Модуль АЦП, показанный на рисунке 92, состоит из следующего:

– 8 аналоговых входов для каждого блока модуля АЦП, дающих в сумме 16 аналоговых входов (ADC0 – ADC15).

– Одновременное измерение двух аналоговых входов с использованием двух блоков модуля АЦП.

– Одиночное преобразование, непрерывное преобразование.

– Преобразование может быть запущено программно, внутренним событием, и/или внешним событием.

– Выводы опорного напряжения V_{REFHI} и V_{REFLO} (высокое и низкое напряжение).

– Блок аналогово-цифрового преобразования.

– Два двухуровневых регистра цифрового результата, которые содержат цифровые значения завершенных преобразований.

– Два программируемых управляющих регистра модуля АЦП.

– Программируемый выбор делителя.

– Операция прерывания или опроса.

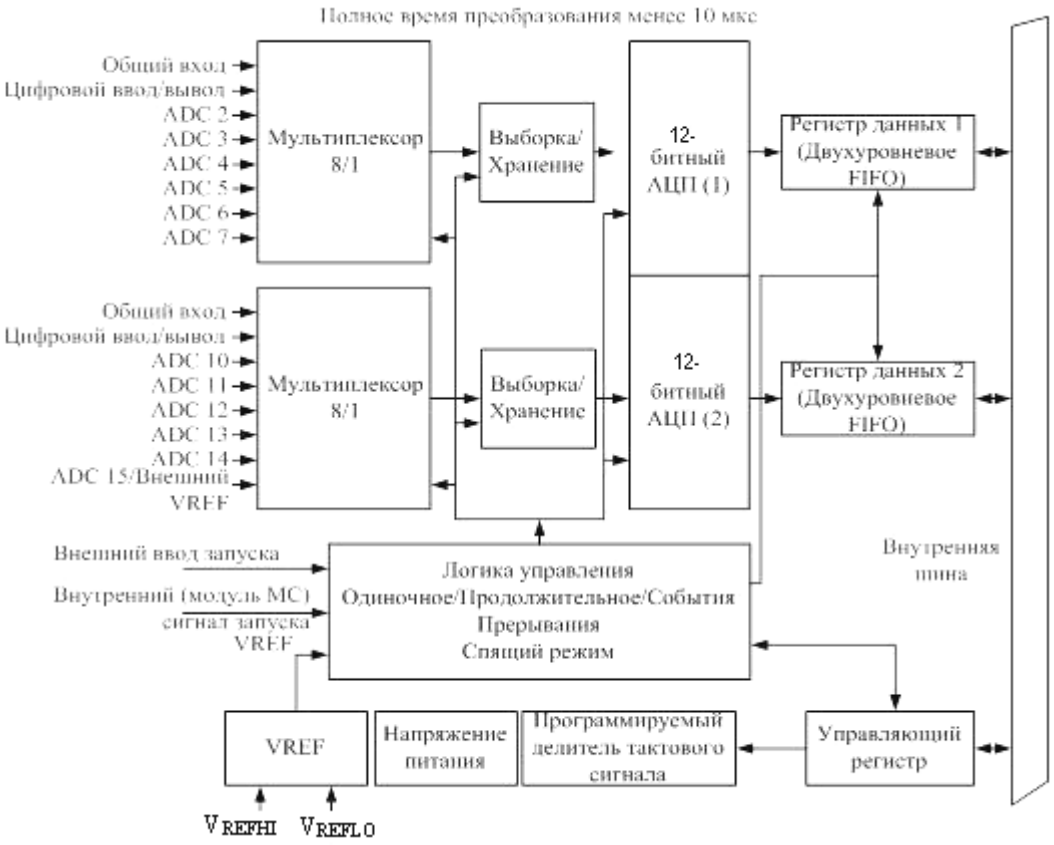


Рисунок 92 – Структурная схема модуля АЦП

13.3.2 Функционирование модуля АЦП

Цифровой результат процесса преобразования для 12-битного модуля АЦП описывается следующим уравнением:

$$\text{Цифровой результат} = 1023 \times \frac{\text{Входное напряжение}}{\text{Опорное напряжение}}$$

Описание вводов модуля АЦП

Модули АЦП предусматривают 20 выводов, которые могут взаимодействовать с внешними цепями. Шестнадцать этих выводов, ADC0–ADC15, предназначены для аналоговых входов. Два других вывода V_{REFHI} и V_{REFLO}, предназначены для входов аналогового опорного напряжения.

Вводы аналогового питания ∩VCC2 и ∩GND2 отделены от каких-либо вводов цифрового питания. Аналоговые шины питания, соединенные с ∩VCC2 и ∩GND2 должны быть как можно короче, с двумя правильно разъединенными шинами. Вся остальная стандартная техника уменьшения шума должна быть использована для обеспечения верного преобразования.

Вводы аналогового напряжения ADC0–ADC7 относятся к первому модулю АЦП и ADC8–ADC15 относятся ко второму модулю АЦП. Аналоговые входы ADC0 и ADC1 первого модуля и аналоговые входы ADC8 и ADC9 второго модуля мультиплексируются с цифровым вводом-выводом. Правильным программированием системного модуля эти четыре аналоговых ввода (ADC0, ADC1, ADC8 и ADC9) могут быть использованы как цифровые вводы-выводы. Точность этих четырёх вводов меньше чем, специальных аналоговых вводов (ADC2–ADC7 и ADC10–ADC15).

Режимы работы модуля АЦП

Функции модуля АЦП включают:

– Два входных канала (один на каждый блок модуля АЦП) могут быть дискретизированы и преобразованы одновременно.

– Каждый блок модуля АЦП может производить операции одиночной или продолжительной выборки/хранения и преобразования.

– Два двухуровневых FIFO регистров результата для модуля АЦП 1 и модуля АЦП 2.

– Модуль АЦП (оба А/Ц преобразователя) может запускаться программно, переключением внешнего сигнала на контактной площадке, или событием в модуля EV на каждом из выходов GP таймера сравнения (compare) и захвата 4.

– Определённые биты управляющего регистра модуля АЦП имеют двойную буферизацию с теневыми регистрами и могут быть записаны в любое время без влияния на действующий процесс преобразования. Новые значения битов сначала поступают в теневой регистр вместо активного регистра. Эта новая конфигурация бит далее автоматически загружается из теневого регистра в активный регистр

только после завершения текущего процесса преобразования. Следующий процесс преобразования будет описываться новой конфигурацией битов.

– В конце каждого преобразования, устанавливается флаг прерывания и формируется прерывание, если оно не маскировано/разрешено.

– Если третье преобразование завершилось без чтения FIFO, данные первого преобразования будут потеряны.

Дискретизация/преобразование аналогового сигнала

Одиночный модуль АЦП выполняет дискретизацию входа в одном делённом тактовом периоде модуля АЦП и преобразование в 5 делённых тактовых периодах модуля АЦП для полной дискретизации/преобразования за 6 тактовых периодов. Архитектура модуля АЦП требует время дискретизации/преобразования 6 мкс или более для обеспечения точного преобразования. Это отношение между требуемым числом тактовых периодов модуля АЦП. Минимум 6 мкс должно пройти при любых частотах системного тактового сигнала (SYSCLK) для обеспечения точного преобразования. Системный тактовый сигнал может работать на частотах, которые нарушают это отношение, в этом случае используется делитель модулей АЦП, который позволяет сохранять оптимальные характеристики при изменении тактового сигнала ЦПОС. Значение делителя модуля АЦП выбирается так, чтобы полное время дискретизации/преобразования было больше либо равно 6 мкс. Значение делителя должно удовлетворять следующей формуле:

$$(\text{Период SYSCLK}) \times (\text{Значение делителя}) \times 6 \geq 6 \text{ мкс}$$

В таблице 63 приведен пример частот тактового сигнала и соответствующих значений делителя.

Таблица 63 – Пример тактовых частот и соответствующих значений делителя

Биты ADCTRL2			Значение делителя
Бит 2	Бит 1	Бит 0	
0	0	0	4
0	0	1	6
0	1	0	8
0	1	1	10
1	0	0	12
1	0	1	16
1	1	0	20
1	1	1	32

13.3.3 Регистры модуля АЦП

В этой главе приведено побитное описание управляющих регистров, используемых для программирования модуля АЦП. В таблице 64 представлены адреса регистров модуля АЦП.

Таблица 64 – Адреса регистров модуля АЦП

Адрес	Регистр	Имя регистра
7032h	ADCTRL1	Управляющий регистр модуля АЦП 1
7034h	ADCTRL2	Управляющий регистр модуля АЦП 1
7036h	ADCFIFO1	Двухуровневый регистр данных FIFO для модуля АЦП 1
7038h	ADCFIFO2	Двухуровневый регистр данных FIFO для модуля АЦП 2

Управляющий регистр модуля АЦП 1 (ADCTRL1)

Управляющий регистр модуля АЦП 1 управляет запуском преобразования, функцией разрешения/запрещения модуля АЦП, разрешением прерываний и завершением преобразования. Описание бит ADCTRL1 показано на рисунке 93.

15	14	13	12	11	10	9	8
Suspend-soft	Suspend-free	ADCIMSTART	ADC1EN	ADC2EN	ADCCONRUN	ADCINTEN	ADCINTFLAG
RW-0	RW-0	RW-0	SRW-0	SRW-0	SRW-0	SRW-0	RW-0

7	6-4	3-1	0
ADCEOC	ADC2CHSEL	ADC1CHSEL	ADCSOC
R-0	SRW-0	SRW-0	SRW-0

R – доступ чтения, W – доступ записи, S = скрытый, -0 – значение после сброса

Рисунок 93 – Управляющий регистр модуля АЦП 1 (ADCTRL1) – адрес 7032h

Бит 15 Suspend–soft. Применяется только во время эмуляции.
0 = Немедленная остановка, когда suspend–free (бит 14) = 0.
1 = Завершение преобразования перед остановкой эмулятора.

Бит 14 Suspend–free. Применяется только во время эмуляции.
0 = Работа определяется suspend–soft (бит 15).
1= Продолжение работы при приостановке эмулятора.

Бит 13 ADCIMSTART. Немедленный запуск преобразования в модуле АЦП.
0 = нет действия.
1 = Немедленный запуск преобразования.

Бит 12 ADC1EN. Разрешение/запрещение бит для модуля АЦП 1. Этот бит может быть записан во время работы предыдущего преобразования. Однако, эффект от записи в этот бит будет только при следующем преобразовании.
0 = модуль АЦП 1 запрещён. (Не выполняются выборка/хранение /преобразование; данные в регистре FIFO1 не будут меняться).
1 = модуль АЦП 1 разрешен.

Бит 11 ADC2EN. Разрешение/запрещение бит для модуля АЦП 2. Этот бит может быть записан во время работы предыдущего преобразования. Одна-

ко, эффект от записи в этот бит будет только при следующем преобразовании.
0 = модуль АЦП 2 запрещён. (Не выполняются выборка/хранение/ преобразование; данные в регистре FIFO1 не будут меняться).
1 = модуль АЦП 2 разрешен.

Бит 10 ADCCONRUN. Этот бит устанавливает модуль АЦП в режим непрерывного преобразования. Этот бит может быть записан во время работы предыдущего преобразования. Однако, эффект от записи в этот бит будет только при следующем преобразовании.
0 = нет действия.
1 = Непрерывное преобразование.

Бит 9 ADCINTEN. Разрешение прерываний. Если бит ADCINTEN установлен, прерывания запрашиваются, когда ADCINTFLAG установлен. Этот бит очищается при сбросе. Этот бит может быть записан во время работы предыдущего преобразования. Однако, эффект от записи в этот бит будет только при следующем преобразовании.

Бит 8 ADCINTFLAG. Бит флага прерывания модуля АЦП. Этот бит определяет, произошло ли событие прерывания. Запись 1 в ADCINTFLAG очищает этот бит.
0 = Не возникает прерывания.
1 = Возникает прерывание.

Бит 7 ADCEOC. Этот бит определяет статус преобразования в модуле АЦП.
0 = Завершение преобразования.
1 = Преобразование выполняется.

Биты 6-4 ADC2CHSEL. Выбирает каналы для модуля АЦП 2. Этот бит может быть записан во время работы предыдущего преобразования. Однако, эффект от записи в этот бит будет только при следующем преобразовании.
000 = Канал 9.
001 = Канал 10.
010 = Канал 11.
011 = Канал 12.
100 = Канал 13.
101 = Канал 14.
110 = Канал 15.
111 = Канал 16.

Биты 3-1 ADC1CHSEL. Выбирает каналы для модуля АЦП 1. Этот бит может быть записан во время работы предыдущего преобразования. Однако, эффект от записи в этот бит будет только при следующем преобразовании.
000 = Канал 1.
001 = Канал 2.
010 = Канал 3.
011 = Канал 4.

100 = Канал 5.
101 = Канал 6.
110 = Канал 7.
111 = Канал 8.

Бит 0 ADCSOC. Запуск преобразования в модуле АЦП. Этот бит может быть записан во время работы предыдущего преобразования. Однако, эффект от записи в этот бит будет только при следующем преобразовании.
0 = нет действия.
1 = Запуск преобразования.

Примечание – Оба канала должны быть разрешены перед запуском преобразования.

Управляющий регистр модуля АЦП 2 (ADCTRL2)

Управляющий регистр модуля АЦП 2 выбирает делитель входного тактового сигнала модуля АЦП, режим преобразования, эмуляцию, а так же показывает статус FIFO модуля АЦП. Описание бит ADCTRL2 показано на рисунке 94.

15-11	10	9	8
Зарезервировано	ADCEVSOC	ADCEXTSOC	Зарезервировано
	SRW-0	SRW-0	

7-6	5	4-3	2-0
ADCFIFO1	Зарезервировано	ADCFIFO2	ADCPSCALE
R-0		R-0	SRW-0

R – доступ чтения, W – доступ записи, S = скрытый, -0 – значение после сброса

Рисунок 94 – Управляющий регистр модуля АЦП 2 (ADCTRL2) – адрес 7034h

Биты 15-11 Зарезервировано. Чтения неопределенны, запись не имеет эффекта.

Бит 10 ADC EVSOC. Бит маски запуска преобразования модуля EV. Когда установлен этот бит, преобразование в модуле АЦП может быть синхронизировано с сигналом модуля EV. модуля EV может запустить преобразование в зависимости от совпадения при сравнении.
0 = Маскирование ADC EVSOC (Запрещение запуска преобразования с помощью модуля EV).
0 = Разрешение запуска преобразования с помощью модуля EV.

Бит 9 ADCEXTSOC. Бит маски внешнего сигнала. Когда установлен этот бит, преобразование в модуле АЦП может быть синхронизировано с внешним сигналом.
Примечание – Если EXT_SOC выполняется более семи тактовых периодов, второе преобразование будет запущено после первого преобразования.

Бит 8 Зарезервировано. Чтения неопределенны, запись не имеет эффекта.

Биты 7-6 ADCFIFO1. Регистр даты статуса FIFO1. Эти два бита определяют статус FIFO модуля АЦП 1. Два результата преобразования могут храниться перед выполнением любой операции чтения. Однако, после двух преобразований, если произошло третье преобразование, старый результат будет утерян.
00 = FIFO1 пуст.
01 = FIFO1 имеет одну запись.
10 = FIFO1 имеет две записи.
11 = FIFO1 имел две записи, и другая запись была получена; первая запись была утеряна.

Бит 5 Зарезервировано. Чтения неопределенны, запись не имеет эффекта.

Биты 4-3 ADCFIFO2. Регистр даты статуса FIFO2. Эти два бита определяют статус FIFO модуля АЦП 2. Два результата преобразования могут храниться перед выполнением любой операции чтения. Однако, после двух преобразований, если произошло третье преобразование, старый результат будет утерян.
00 = FIFO2 пуст.
01 = FIFO2 имеет одну запись.
10 = FIFO2 имеет две записи.
11 = FIFO2 имел две записи, и другая запись была получена; первая запись была утеряна.

Биты 2-0 ADCPSCALE. Делитель внутреннего сигнала модуля АЦП. Эти биты описывают делитель сигнала модуля АЦП (см. таблицу 72).

Регистры цифрового результата модуля АЦП

Регистры цифрового результата содержат 12-битный цифровой результат текущего преобразования напряжения аналогового входа. Это регистры доступны только для чтения. При сбросе они очищаются. Результаты хранятся в двухуровневых FIFO. Это обеспечивает гибкость преобразования двух переменных перед их чтением из регистров данных. Однако если произошло третье преобразование, когда в FIFO содержатся два непрочитанных значения, первое значение преобразования будет потеряно. Описание бит этих регистров показано на рисунке 95.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0	0	0
MSB									LSB						

Рисунок 95 – Регистры цифрового результата модуля АЦП FIFO1 (ADCFIFO1) – адрес 7036h и FIFO2 (ADCFIFO2) – адрес 7038h

Биты 15-6 D9 – D0. Преобразованные 12-битные данные.

Биты 5-0 Зарезервировано. Чтения всегда равны 0.

13.4 Модуль последовательного коммуникационного интерфейса SCI

Модуль последовательного коммуникационного интерфейса SCI является частью библиотеки PRSIM. Архитектура, функции и программирование модуля SCI описаны в этой главе.

Примечание – Этот модуль соединён с 16-битной периферийной шиной как 8-битная периферия. Поэтому, чтения бит 15-8 не определены; записи не имеют эффекта.

13.4.1 Обзор модуля SCI

Программируемый модуль SCI поддерживает цифровую коммуникацию между процессором и другой асинхронной периферией, которая используют стандартный формат без возврата к нулю. Приёмник и передатчик модуля SCI имеют двойную буферизацию, и каждый имеет собственные отдельные биты разрешения и прерывания. Они могут работать независимо или совместно в режиме двусторонней передачи.

Для обеспечения целостности данных, модуль SCI проверяет принятые данные на разрывы, четность, переполнение и ошибки кадрирования. Скорость двоичной передачи программируема на более чем 65000 разных скоростей через 16-битный регистр выбора скорости двоичной передачи.

Примечание – Для удобства ссылки на биты регистра используются имя регистра и следующий за ним номер бита. Например, представление нулевого бита управляющего регистра 1 порта модуля SCI (SCIPC1) будет SCIPC1.0.

Физическое описание модуля SCI

Модуль SCI, представленный на рисунке 96, имеет следующие основные характеристики:

- Два ввода-вывода:
  - SCIRXD (вход принимаемых данных модуля SCI);
  - SCITXD (выход передаваемых данных модуля SCI).
- Программируемая двоичная передача на 65000 различных скоростей через 16-битный регистр выбора скорости двоичной передачи:
  - диапазон 10 МГц SYSCLK: от 19,07 бит/с до 625,0 Кбит/с;
  - число скоростей двоичной передачи: 64К.
- Программируемая длина слова данных от одного до восьми бит.
- Программируемые остановочные биты: или один или два бита.
- Внутренне генерируемый последовательный тактовый сигнал.
- Четыре флага определения ошибки:
  - ошибка четности;
  - ошибка переполнения;
  - ошибка кадрирования;
  - определение разрывов.
- Два мультипроцессорных режима запуска, которые могут использоваться с любым из форматов передачи данных:
  - запуск при простое линии;

– запуск по адресу бита.

– Работа в полном и в полудуплексном режиме.

– Двойная буферизация функций передачи и приёма.

– Работа передатчика и приёмника может осуществляться через прерывания или через операцию опроса статусных флагов:

– передатчик: флаг TXRDY (буферный регистр передатчика готов принять ещё один символ) и флаг TX EMPTY (сдвиговый регистр передачи пуст);

– приёмник: флаг RXRDY (буферный регистр приёмника готов принять ещё один символ), флаг BRKDT (произошел разрыв) и RX ERROR, следящий за условиями прерывания.

– Отдельные биты разрешения для прерываний приёмника и передатчика (исключая разрыв).

– Формат без возврата к нулю.

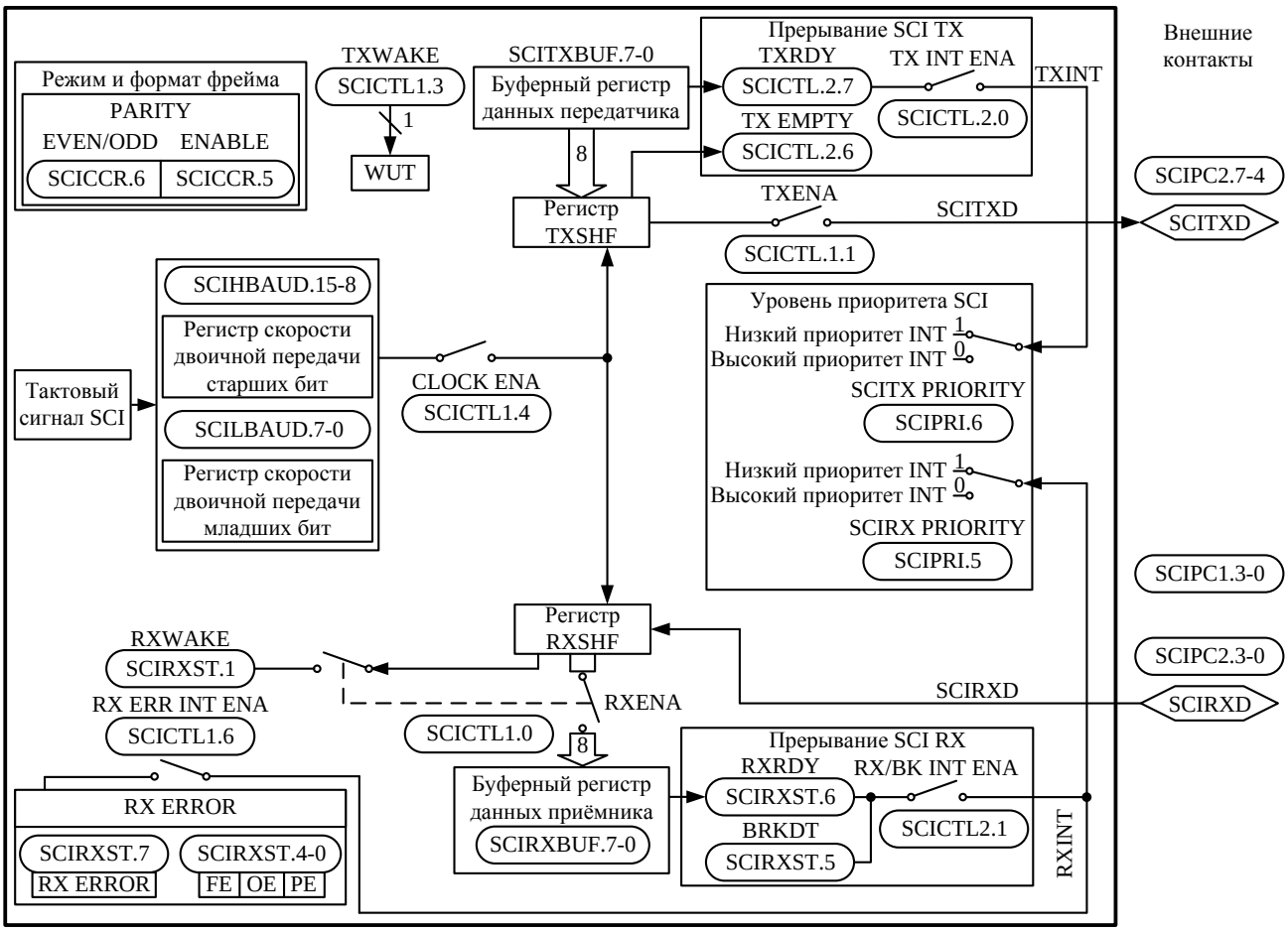


Рисунок 96 – Структурная схема модуля SCI

Архитектура

Основные элементы, используемые при двунаправленной передаче, показаны на рисунке 95 и включают:

– Передатчик (TX) и его основные регистры:

– SCITXBUF – буферный регистр данных передатчика. Содержит данные (загруженные процессором) для передачи;

– TXSHF – сдвиговый регистр передатчика. Загружает данные из SCITXBUF и сдвигает их на ввод SCITXD, один бит за раз.

– Приёмник (RX) и его основные регистры:

– RXSHF – сдвиговый регистр приёмника. Сдвигает данные от ввода SCIRXD, один бит за раз;

– SCIRXBUF – буферный регистр данных приёмника. Содержит данные, считываемые процессором. Данные от удаленного процессора загружаются в RXSHF, а затем в SCIRXBUF и SCIRXEMU.

– Контроллер скорости передачи.

– Картированные в память данных управляющие и статусные регистры.

Приёмник и передатчик модуля SCI могут работать независимо или совместно.

Управляющие регистры модуля SCI

Управляющие регистры модуля SCI и их функции показаны в таблице 65

Таблица 65 – Адреса регистров модуля SCI

Адрес	Регистр	Имя	Описание
7050h	SCICCR	Управляющий регистр связи модуля SCI	Определяет формат символа, протокол и режим коммуникации используемого модуля SCI
7051h	SCICTL1	Управляющий регистр модуля SCI 1	Управляет RX/TX, разрешением прерывания ошибки приемника, функциями TXWAKE и SLEEP, разрешением внутреннего тактового сигнала и программным сбросом модуля SCI
7052h	SCIHBAUD	Регистр скорости двоичной передачи старших бит модуля SCI	Содержит данные (старшие биты) требуемые для формирования скорости двоичной передачи
7053h	SCILBAUD	Регистр скорости двоичной передачи младших бит модуля SCI	Содержит данные (младшие биты) требуемые для формирования скорости двоичной передачи
7054h	SCICTL2	Управляющий регистр модуля SCI 2	Содержит разрешение прерывания передатчика, разрешение прерывания буфера/разрыва приёмника, флаг готовности передатчика и флаг опустошения передатчика

Окончание таблицы 65

Адрес	Регистр	Имя	Описание

7055h	SCIRXST	Регистр статуса приёмника модуля SCI	Содержит семь флагов статуса приёмника
7056h	SCIRXEMU	Буферный регистр данных эмуляции	Содержит данные, полученные для обновления изображения, в основном используются эмулятором
7057h	SCIRXBUF	Буферный регистр данных приёмника	Содержит текущие данные от сдвигового регистра приёмника
7058h	—	Зарезервировано	Зарезервировано
7059h	SCITXBUF	Буферный регистр данных передатчика	Хранит биты данных передаваемые SCITX
705Ah	—	Зарезервировано	Зарезервировано
705Bh	—	Зарезервировано	Зарезервировано
705Ch	—	Зарезервировано	Зарезервировано
705Dh	—	Зарезервировано	Зарезервировано
705Eh	SCIPC2	Управляющий регистр порта модуля SCI 2	Управляет функциями вводов SCIRXD и SCITXD
705Fh	SCIPRI	Управляющий регистр приоритета модуля SCI	Содержит биты выбора приоритета прерываний передатчика и приёмника и бит разрешения приостановки эмулятора

Мультипроцессорный и асинхронный коммуникационные режимы

Модуль SCI имеет два мультипроцессорных протокола, мультипроцессорный режим простоя линии (idle-line) и мультипроцессорный режим по адресному бита (address-bit). Эти протоколы позволяют эффективно переносить данные между составными процессорами.

Модуль SCI предоставляет универсальный асинхронный режим коммуникации приёмника/передатчика для связи с большим числом популярных периферийных устройств. Асинхронный режим требует две шины для связи со многими стандартными устройствами, такими как терминалы и принтеры, которые используют форматы RS-232-C. Передача данных характеризуется следующим:

- один бит запуска;
- от одного до восьми битов данных;
- четный/нечетный бит четности или нет бита четности;
- один или два стоповых бита.

13.4.2 Программируемый формат данных модуля SCI

Данные модуля SCI, полученные и переданные, имеют формат без возврата к нулю. Этот формат, представленный на рисунке 97, состоит из следующего:

- один бит запуска;
- от одного до восьми битов данных;
- четный/нечетный бит четности или нет бита (дополнительной) четности;

– один или два стоповых бита;

– дополнительный бит для различия адресов данных (только для режима по адресу бита).

Основная единица данных называется символом и имеет длину от одного до восьми бит. Каждый символ данных форматирован битом запуска, одним или двумя стоповыми битами и битами дополнительной четности и адреса. Символ данных с его форматированной информацией называется фреймом и показан на рисунке 97.

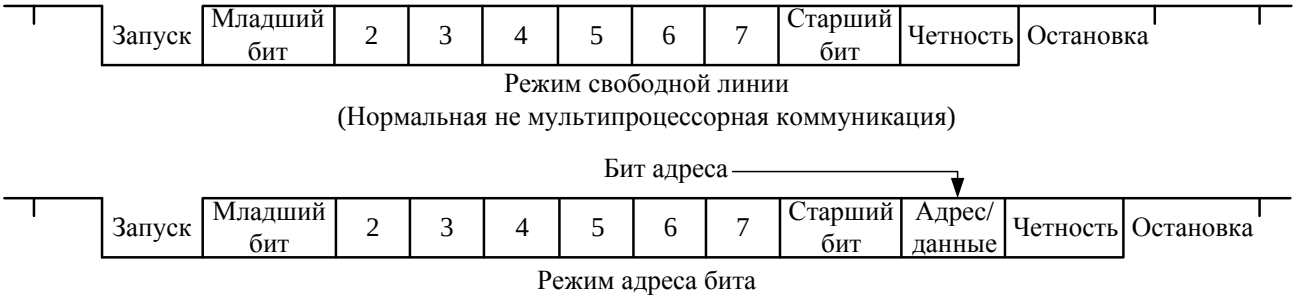


Рисунок 97 – Типичные форматы фреймов данных модуля SCI

Для программирования формата данных используется управляющий регистр связи (SCICCR). Биты этого регистра, используемые для программирования формата данных, показаны в таблице 66.

Таблица 66 – Программирование формата данных с использованием SCICCR

Имя бита	Обозначение	Функции
SCI CHAR2–0	SCICCR.2–0	Выбирает длину символа (от одного до восьми бит). Значения битов даны в таблице 68
PARITY ENABLE	SCICCR.5	Разрешает функцию четности, если установлен в 1 или запрещает функцию четности, если очищен до 0
EVEN/ODD PARITY	SCICCR.6	Если четность разрешена, выбирает нечет, если очищен до 0 или чёт, если установлен в 1
STOP BITS	SCICCR.7	Определяет количество передаваемых стоповых бит – один, очищен до 0 или два, если установлен в 1

13.4.3 Мультипроцессорная коммуникация модуля SCI

Формат мультипроцессорной коммуникации позволяет одному процессору эффективно посылать блоки данных другим процессорам по одной и той же последовательной шине. На одной последовательной линии должна быть только одна передача за единицу времени. Другими словами, только один источник сообщений может находиться на последовательной линии за единицу времени.

Первый бит блока информации, посылаемый источником сообщений, содержит адресный бит, который считывается всеми приёмниками. Только приёмники с правильным адресом могут быть прерваны битами данных, следующими за адресным битом. Приёмники с неправильным адресом остаются непрерванными до следующего адресного бита.

Все процессоры на последовательной шине устанавливают их модули SCI SLEEP бит (SCICTL1.2) в 1, таким образом, они прерываются только когда определён адресный бит. Когда процессор считывает адрес блока, который соответствует адресу процессорного устройства, как установлено программой, программа должна очистить SLEEP бит для разрешения модуля SCI формировать прерывание на получение каждого бита данных.

Несмотря на то, что приёмник продолжает работать при SLEEP бите установленном в 1, он не устанавливает RXRDY, RXINT или каких-либо статусных бит ошибки приёмника в 1, пока не определится адресный бит, и адресный бит в принятом фрейме не будет в 1. Модуль SCI не изменяет SLEEP бит; SLEEP бит должен быть изменён программно.

Процессор определяет адресный бит в соответствии с мультипроцессорным режимом. Например:

– Режим простоя линии оставляет пустую зону перед адресным байтом. Этот режим не имеет дополнительного адресного бита данных и более эффективен, чем режим адресного бита для обработки блоков, которые содержат более чем 10 байт данных. Режим простоя линии используется для обычной немультипроцессорной SCI коммуникации.

– Режим адресного бита добавляет дополнительный бит (адресный бит) в каждый байт для отделения адресов от данных. Этот режим более эффективен в обработке малых блоков данных, потому что в отличие от режима простоя линии ему не требуется ждать между блоками данных. Однако, при высокой скорости передачи, программа работает недостаточно быстро для предотвращения 10-битного простоя в передаваемом потоке.

Мультипроцессорный режим выбирается программно через ADDR/IDLE MODE бит (SCICCR.3). Оба режима используют TXWAKE бит флага (SCICTL1.3), RXWAKE бит флага (SCIRXST.1) и SLEEP биты флага (SCICTL1.2) для управления свойствами передатчика и приёмника модуля SCI в этих режимах.

В обоих мультипроцессорных режимах следующая последовательность получения:

1 При получении блока адреса, порт модуля SCI запускается и запрашивает прерывание (бит RX/BK INT ENA – SCICTL2.1 – должен быть разрешен для запроса прерывания). Он считывает первый фрейм блока, который содержит адрес назначения.

2 Подпрограмма заносится через прерывание и проверяет RXWAKE бит. Если RXWAKE бит в 1, поступающий бит является адресом (в обратном случае, битом данных), и этот адресный бит проверяется на соответствие его адресному биту устройства, содержащегося в памяти.

3 Если проверка показала что блок адресуется к ЦПОС контроллеру, процессор очищает SLEEP бит и считывает остаток блока; если нет, то подпрограмма выходит с установленным SLEEP битом и не получает прерываний до запуска следующего блока.

Мультипроцессорный режим простоя линии

В мультипроцессорном протоколе простоя линии (ADDR/IDLE MODE бит = 0) блоки разделены, имея большее время простоя между блоками, чем между фреймами в блоках. Время простоя десяти или более высокоуровневых бит вычисляется непосредственно от скорости двоичной передачи (бит/с). Мультипроцессорный формат коммуникации простоя линии показан на рисунке 98 (ADDR/IDLE MODE биты – SCICCR.3).

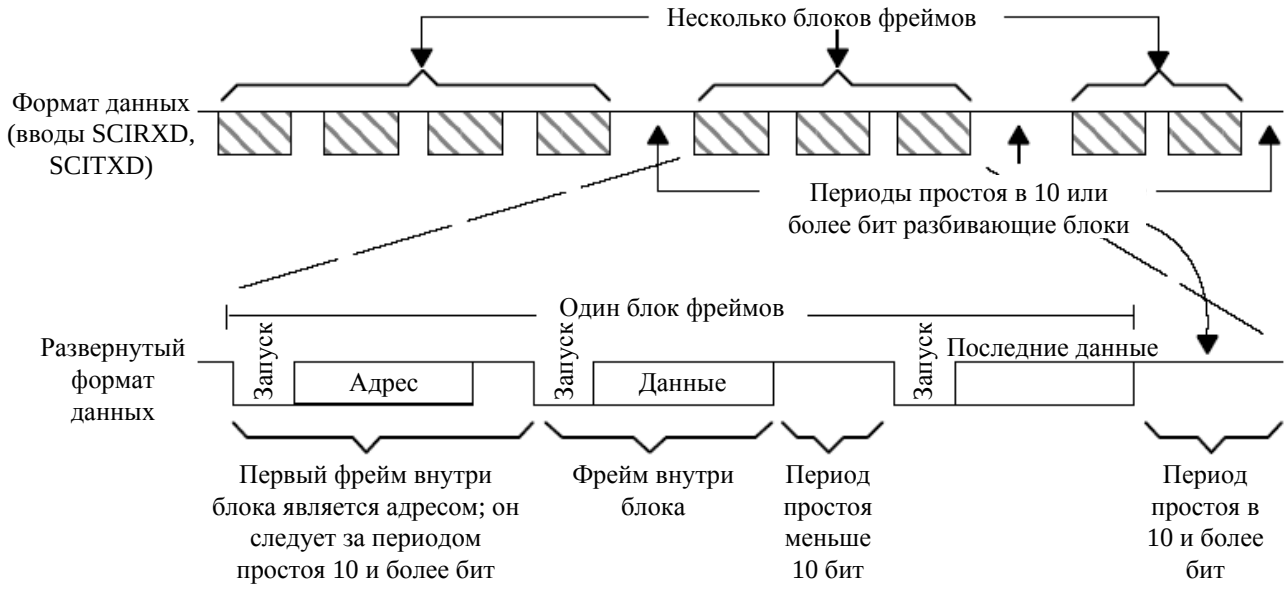


Рисунок 98 – Мультипроцессорный формат коммуникации простоя линии

Порядок запуска режима простоя линии:

1 Модуль SCI запускается после получения сигнала запуска блока.

2 Процессор определяет следующее прерывание модуля SCI.

3 Обслуживающая программа сравнивает полученный адрес (посланный удаленным передатчиком) со своим собственным.

4 Если процессор адресуется, обслуживающая программа очищает SLEEP бит и принимает остаток блока данных.

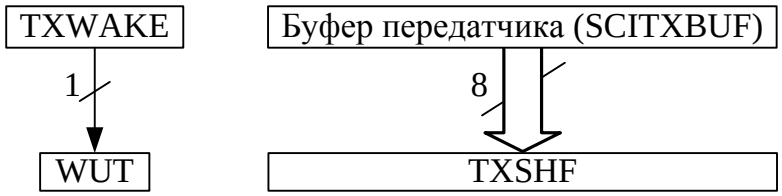
5 Если процессор не адресуется, SLEEP бит остается установленным. Это позволяет процессору продолжать выполнение основной программы, без каких либо прерываний от порта модуля SCI до следующего определения старта блока.

Существует два пути отправки сигнала старта блока:

– Метод 1: Осознанно оставить время простоя в 10 бит или более задержкой времени между передачей последнего фрейма данных в предыдущем блоке и передачей фрейма адреса нового блока.

– Метод 2: Порт модуля SCI сначала устанавливает TXWAKE бит (SCICTL1.3) в 1 перед записью в SCITXBUF. Это устанавливает время простоя точно в 11 бит. При этом методе линия последовательной коммуникации не простаивает больше чем необходимо.

С TXWAKE битом связан флаг временного запуска (WUT). WUT является внутренним флагом с двойной буферизацией, как и TXWAKE. Когда TXSHF загружается из SCITXBUF, WUT загружается из TXWAKE, и TXWAKE бит очищается до 0. Этот механизм показан на рисунке 99.



WUT - Временный запуск

Рисунок 99 – TXSHF и WUT с двойной буферизацией

Чтобы послать сигнал старта блока точно на время одного фрейма в течение последовательности передач блока, необходимо:

1 Записать 1 в TXWAKE бит.

2 Записать слово данных (содержимое неважно) в SCITXBUF (буфер данных передачи) для послания сигнала запуска блока. (Первое записанное слово сдерживается, пока сигнала запуска блока посылается, и игнорируется после этого). Когда TXSHF (регистр сдвига передачи) снова свободен, содержимое SCITXBUF сдвигается к TXSHF, значение TXWAKE сдвигается к WUT, и затем TXWAKE очищается.

Из-за того, что TXWAKE установлен в 1, биты запуска, данных и четности заменяются периодом простоя в 11 передаваемых бит, следуя за последним стоповым битом предыдущего фрейма.

3 Записать новое значение адреса в SCITXBUF.

Слово данных (содержимое неважно) должно быть сначала записано в SCITXBUF, таким образом, значение TXWAKE бита может быть сдвинуто к WUT. После того как это слово было сдвинуто к TXSHF, SCITXBUF (и TXWAKE, если необходимо) может быть снова записан, потому что TXSHF и WUT имеют двойную буферизацию.

Приёмник работает независимо от SLEEP бита. Тем не менее, приёмник не устанавливает ни RXRDY, ни статусного бита ошибки и не запрашивает прерывание приёма, пока не будет определен адресный фрейм.

Мультипроцессорный режим адресного бита

В протоколе адресного бита (ADDR/IDLE MODE бит = 1) фреймы имеют дополнительный бит, называемый адресным битом, который следует сразу за последним битом данных. Адресный бит установлен в 1 в первом фрейме блока и в 0 во всех других фреймах. Периоды простоя несущественны (смотри рисунок 100, ADDR/IDLE MODE бит – SCICCR.3).

Значение TXWAKE бита помещено в адресный бит. При передаче, когда SCITXBUF и TXWAKE загружены в TXSHF и WUT соответственно, TXWAKE сбрасывается до 0 и WUT становится значением адресного бита текущего фрейма. Таким образом, чтобы послать адрес нужно:

1 Установить TXWAKE бит в 1 и записать подходящее значение адреса в SCITXBUF.

2 Когда это значение адреса предано в TXSHF и сдвинуто без сохранения сдвигаемых разрядов, его адресный бит посылается как 1 и оповещает другие процессоры на последовательной линии для чтения адреса.

3 Как только TXSHF и WUT получают двойную буферизацию, SCITXBUF и TXWAKE могут быть сразу же записаны после загрузки TXSHF и WUT.

4 Для передачи безадресного фрейма в блоке необходимо оставить TXWAKE бит в 0.

Примечание – Как основное правило, формат адресного бита обычно используется для фреймов данных в 11 или меньше байт. Этот формат добавляет одно значение бита (1 для адресного фрейма, 0 для фрейма данных) ко всем передаваемым байтам данных. Формат простоя линии обычно используется для фреймов данных в 12 и более байт.

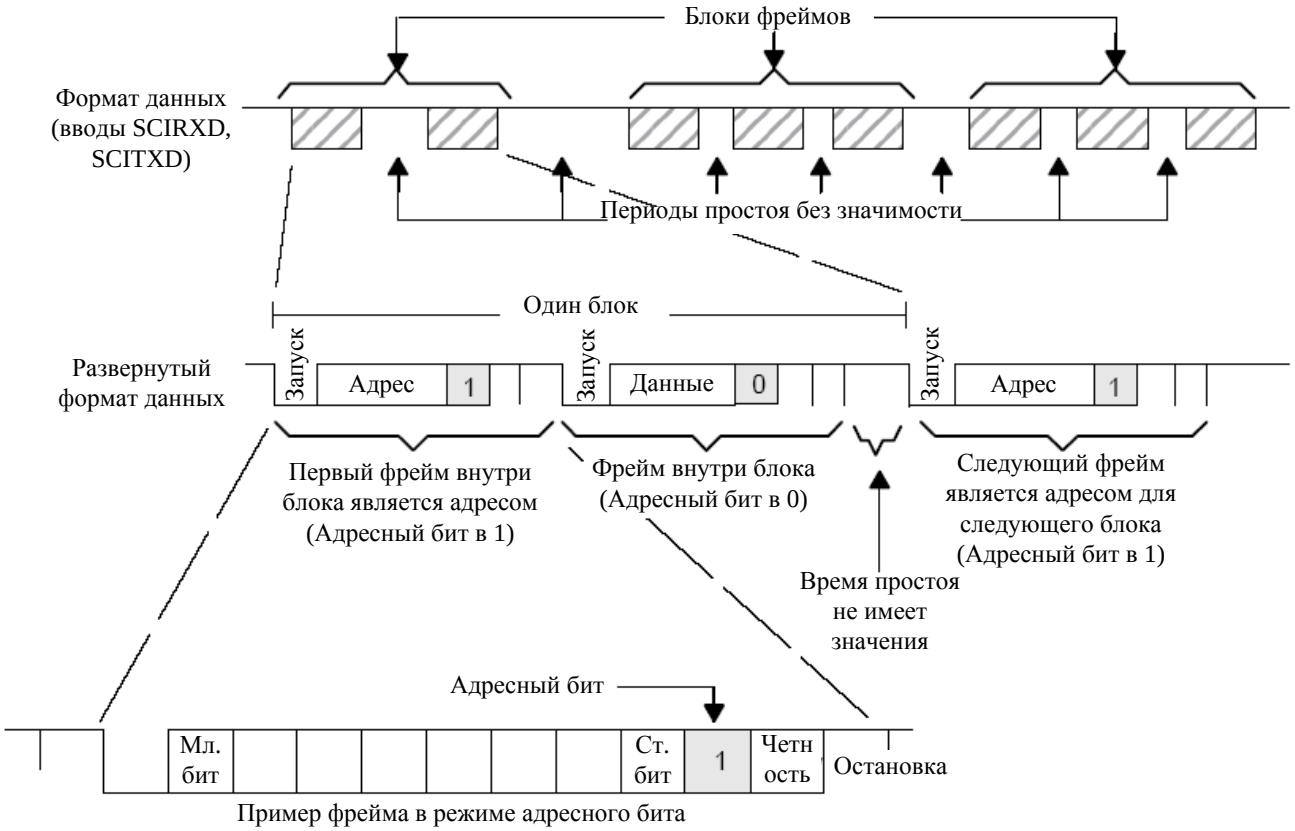


Рисунок 100 – Мультипроцессорный формат коммуникации адресного бита

13.4.4 Формат коммуникации модуля SCI

Асинхронный формат коммуникации модуля SCI использует или одиночную линию (односторонняя) или две линии (двухсторонняя) коммуникации. В этом режиме фрейм состоит из бита запуска, от одного до восьми битов данных, дополнительного чет/нечет бита четности и одного или двух стоповых битов (показанных на рисунке 101). На каждый бит данных приходится 8 периодов SCICLK.

Приемник начинает работать при получении правильного бита запуска. Правильный бит запуска идентифицируется четырьмя последовательными периодами внутреннего SCICLK с нулевыми битами, как показано на рисунке 101. Если ка-

кой-либо бит не нулевой, процессор снова запускается и начинает искать другой бит запуска.

Для бит, следующих за битом запуска, процессор определяет значение бит, производя три выборки в середине бит. Эти выборки происходят на четырёх, пяти и шести периодах SCICLK, и значение бита определяется по большинству (два из трёх). На рисунке 101 иллюстрирован асинхронный формат коммуникации с указанием, как находятся фронты бита запуска и где берётся выборка.

Как только приёмник синхронизируется с фреймами, внешние передающие и принимающие устройства могут не использовать синхронизированный последовательный тактовый сигнал; тактовый сигнал может быть сформирован локально.

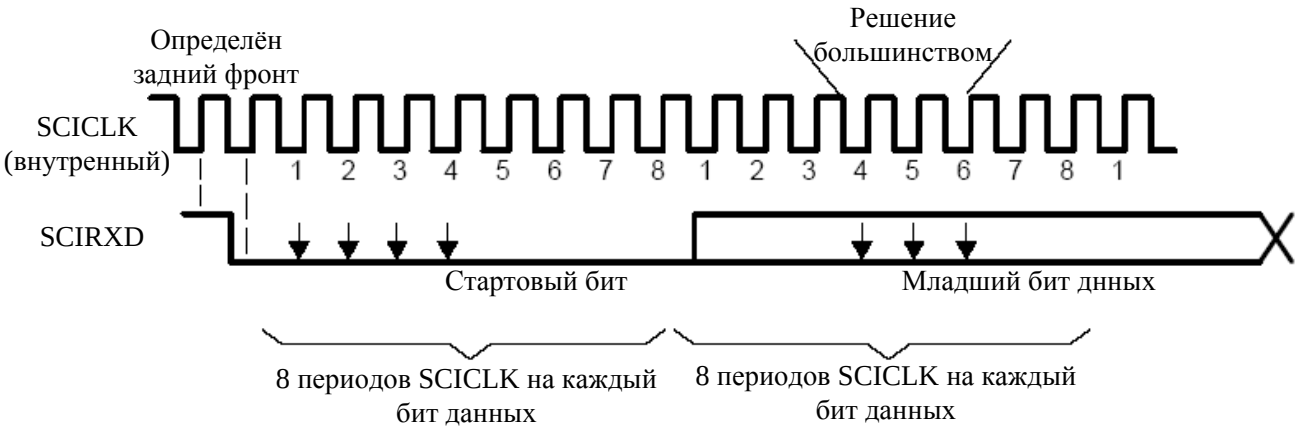


Рисунок 101 – Асинхронный формат коммуникации модуля SCI

Сигналы приёмника в коммуникационных режимах

На рисунке 102 показан пример изменения со временем сигнала приёмника, который предполагает следующие условия:

– Режим запуска адресным битом (адресный бит не появится в режиме простоя линии).

– 6 бит на символ.

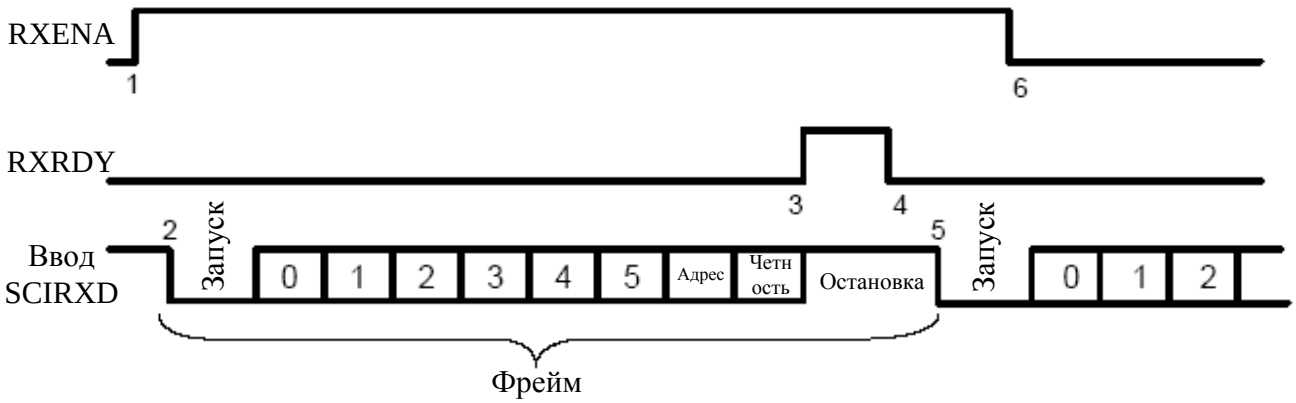


Рисунок 102 – Сигналы модуля SCI RX в коммуникационных режимах

Примечание – Обозначения 1 - 6 соответствуют цифрам на рисунке 102:

1 – Бит флага RXENA (SCICTL1.0) становится в высокое состояние для разрешения приёмника.

2 – Данные прибывают на ввод SCIRXD, определяется бит запуска.

3 – Данные сдвигаются из RXSHF для их принятия в буферном регистре (SCIRXBUF); прерывание запрошено. Бит флага RXRDY (SCIRXST.6) становится в высокое состояние, чтобы показать, что был принят новый символ.

4 – Программа считывает SCIRXBUF, флаг RXRDY автоматически очищается.

5 – Следующий байт прибывает на ввод SCIRXD; определяется бит запуска, затем очищается.

6 – Бит RXENA становится в низкое состояние для запрещения приёмника. Данные продолжают собираться в RXSHF, но не передаются буферному регистру приёмника.

Сигналы передатчика в коммуникационных режимах

На рисунке 103 иллюстрирован пример изменения со временем сигнала передатчика, который предполагает следующие условия:

– Режим запуска адресным битом (адресный бит не появится в режиме простоя линии).

– 3 бита на символ.

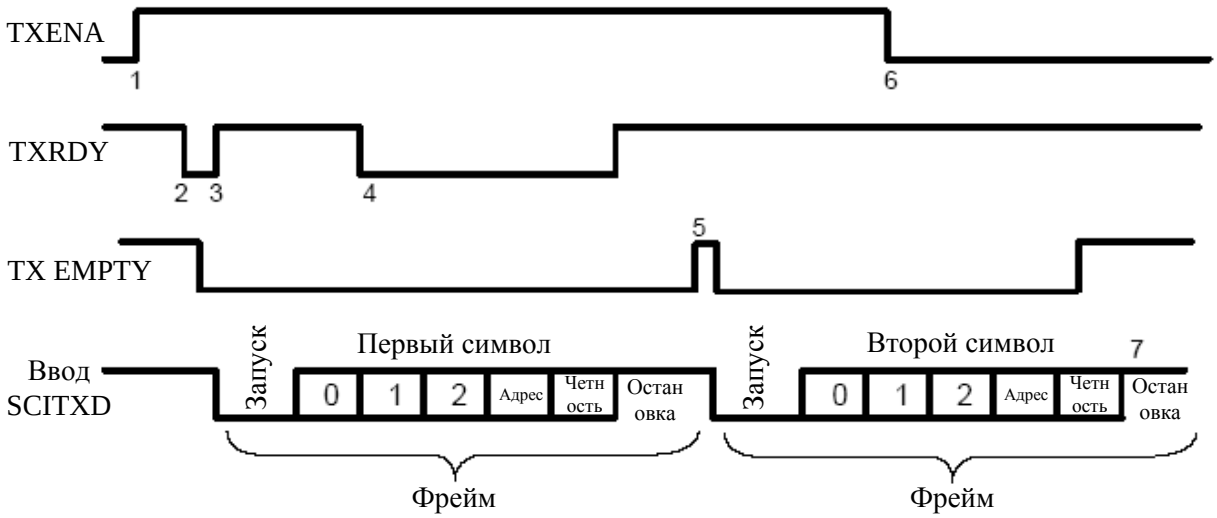


Рисунок 103 – Сигналы SCI TX в коммуникационных режимах

Примечание – Обозначения 1-7 соответствуют цифрам на рисунке 103:

1 – Бит TXENA (SCICTL1.1) становится в высокое состояние для разрешения передатчику посылать данные.

2 – SCITXD записывается; таким образом, (1) передатчик больше не пуст, и TXRDY становится в низкое состояние.

3 – Модуль SCI передает данные к сдвиговому регистру TXSHF. Передатчик готов для принятия следующего символа (TXRDY становится в высокое состояние), и он запрашивает прерывание (для разрешения прерывания бит TX INT ENA — SCICTL2.0 — должен быть установлен).

4 – Программа записывает второй символ в SCITXBUF после того, как TXRDY становится в высокое состояние (пункт 3).

5 – Передача первого символа завершена. TX EMPTY временно становится в высокое состояние. Начинается передача второго символа к сдвиговому регистру TXSHF.

6 – Бит TXENA становится в низкое состояние для запрещения передатчика; SCI завершает передачу текущего символа.

7 – Передача второго символа завершена; передатчик пуст и готов для нового символа.

13.4.5 Прерывания от порта модуля SCI

Внутренне сформированный последовательный тактовый сигнал определяется частотой SYSCLK и регистрами выбора блока. Модуль SCI использует 16-битное значение регистров выбора скорости двоичной передачи для выбора одного из 64К различных коэффициентов последовательного тактового сигнала.

Передатчик и приёмник модуля SCI могут быть управляемыми прерыванием. SCICTL2 имеет один бит флага (TXRDY), который указывает на действующие условия прерывания и SCIRXST имеет два бита флага прерывания (RXRDY и BRKDT). Передатчик и приёмник имеют различные биты разрешения прерывания. Когда они не разрешены, то прерывания блокируются; однако флаги условий приема/передачи продолжают устанавливаться, отражая передачу и статус приёма.

Модуль SCI предусматривает независимые запросы прерывания и векторы для передатчика и приёмника.

Если бит RX/BK INT ENA (SCICTL2.1) установлен, прерывание приёмника утверждается, когда происходит одно из следующих событий:

- Модуль SCI принимает полный фрейм и передает данные из RXSHF к SCIRXBUF. Это действие устанавливает флаг RXRDY (SCIRXST.6) и инициирует прерывание.

- Выполняется условие определения разрывов (SCIRXD в низком состоянии в течение 10 бит периодов, следующих за стоповым битом). Это действие устанавливает бит флага BRKDT (SCIRXST.5) и инициирует прерывание.

Если бит TX INT ENA (SCICTL2.0) установлен, прерывание приёмника утверждается всякий раз, когда данные в SCITXBUF передаются к TXSHF, указывая, что процессор может записывать в TXBUF. Это действие устанавливает бит флага TXRDY (SCICTL2.7) и инициирует прерывание.

Прерывания модуля SCI могут быть запрограммированы на различные уровни приоритета управляющими битами SCIRX PRIORITY (SCIPRI.5) и SCITX PRIORITY (SCIPRI.6). Когда оба запроса прерывания RX и TX совершены на одном уровне, приёмник всегда имеет высший приоритет, чем передатчик, уменьшая возможность перегрузки приёмника.

Вычисление скорости двоичной передачи модуля SCI

Внутренне сформированный последовательный тактовый сигнал определяется частотой SYSCLK и регистрами выбора блока. Модуль SCI использует 16-

битное значение регистров выбора скорости двоичной передачи для выбора одного из 64К различных коэффициентов последовательного тактового сигнала.

Скорость двоичной передачи модуля SCI для различных коммуникационных режимов определяется следующими путями:

– Асинхронный бод модуля SCI для BRR = 1 до 65535:

$$\text{Асинхронный бод SCI} = \frac{\text{SYSCLK}}{(\text{BRR}+1)\times 8},$$

$$\text{BRR} = \frac{\text{SYSCLK}}{(\text{Асинхронный бод SCI})\times 8} - 1.$$

– Асинхронный бод модуля SCI для BRR = 0:

$$\text{Асинхронный бод SCI} = \frac{\text{SYSCLK}}{16},$$

где BRR – 16-битное значение регистров выбора скорости двоичной передачи.

В таблице 67 представлены значения регистров скорости двоичной передачи для соответствующих скоростей передачи модуля SCI.

Таблица 67 – Значения асинхронных регистров скорости двоичной передачи для соответствующих скоростей передачи модуля SCI

Идеально выбранный бод	Частота SYSCLK					
	1 МГц			5 МГц		
	BRR	Фактически выбранный бод	% ошибки	BRR	Фактически выбранный бод	% ошибки
300	416	300	–0,08	2082	300	0,02
600	207	601	0,16	1041	600	–0,03
1200	103	1202	0,16	520	1200	–0,03
2400	51	2404	0,16	259	2404	0,16
4800	25	4808	0,16	129	4808	0,16
8192	14	8333	1,73	75	8224	0,39
9600	12	9615	0,16	64	9615	0,16
19200	6	17857	–6,99	32	18939	–1,36
300	3332	300	0,01	4166	300	0,01
600	1666	600	–0,02	2082	600	0,02
1200	832	1200	0,04	1041	1200	0,03
2400	416	2398	–0,08	520	2399	0,03
4800	207	4808	0,16	259	4808	0,16
8192	121	8197	0,06	152	8170	0,27
9600	103	9615	0,16	130	9542	0,60
19200	51	19231	0,16	64	19231	0,16

13.4.6 Управляющие регистры модуля SCI

Функции модуля SCI программно конфигурируемы. Установки управляющих бит, организованных в специализированных байтах, запрограммированы для инициализации требуемого формата коммуникации модуля SCI. Это включает режим и протокол работы, значение скорости двоичной передачи, длину символа, четность чет/нечет или нет четности, количество стоповых бит и приоритеты и разрешения прерывания. Модуль SCI управляется и утверждается регистрами, показанными на рисунке 104 и описанными в следующих подразделах.

Адрес	Регистр	Номер бита							
		7	6	5	4	3	2	1	0
7050h	SCICCR	STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	SCI ENA	ADDR/IDLE MODE	SCI CHAR2	SCI CHAR1	SCI CHAR0
7051h	SCICTL1	Зарезервировано	RX ERR INT ENA	SW RESET	CLOCK ENA	TXWAKE	SLEEP	TXENA	RXENA
7052h	SCIHBAUD	BAUD15 (ст. бит)	BAUD14	BAUD13	BAUD12	BAUD11	BAUD10	BAUD9	BAUD8
7053h	SCILBAUD	BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0 (мл. бит)
7054h	SCICTL2	TXRDY	TX EMPTY	Зарезервировано				RX/BK INT ENA	TX INT ENA
7055h	SCIRXST	RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	Зарезервировано
7056h	SCIRXEMU	ERXDT7	ERXDT6	ERXDT5	ERXDT4	ERXDT3	ERXDT2	ERXDT1	ERXDT0
7057h	SCIRXBUF	RXDT7	RXDT6	RXDT5	RXDT4	RXDT3	RXDT2	RXDT1	RXDT0
7058h	—	Зарезервировано							
7059h	SCITXBUF	TXDT7	TXDT6	TXDT5	TXDT4	TXDT3	TXDT2	TXDT1	TXDT0
705Ah	—	Зарезервировано							
705Bh	—	Зарезервировано							
705Ch	—	Зарезервировано							
705Dh	—	Зарезервировано							
705Eh	SCIPC2	SCITXD DATA IN	SCITXD DATA OUT	SCITXD FUNCTION	SCITXD DATA DIR	SCIRXD DATA IN	SCIRXD DATA OUT	SCIRXD FUNCTION	SCIRXD DATA DIR
705Fh	SCIPRI	Зарезервировано	SCITX PRIORITY	SCIRX PRIORITY	SCI ESPEN	Зарезервировано			

Рисунок 104 – Управляющие регистры SCI

Управляющий коммуникационный регистр модуля SCI (SCICCR)

Управляющий коммуникационный регистр модуля SCI (SCICCR) определяет формат символа, протокол и коммуникационный режим, используемый модулем SCI. Описание бит SCICCR приведено на рисунке 105.

7	6	5	4	3	2	1	0
STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	SCI ENA	ADDR/IDLE MODE	SCI CHAR2	SCI CHAR1	SCI CHAR0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 105 – Управляющий коммуникационный регистр SCI (SCICCR) – адрес 7050h

Бит 7 STOP BITS. Количество стоповых бит SCI. Этот бит определяет количество передаваемых стоповых бит. Приемник проверяет только один стоповый бит.
0 = Один стоповый бит.
1 = Два стоповых бита.

Бит 6 EVEN/ODD PARITY. Выбор четности чет/нечет SCI. Если бит PARITY ENABLE (SCICCR.5) установлен, PARITY (бит 6) обозначает четность или нечетность (четное или нечетное количество бит со значением 1 в переданных и принятых символах).
0 = Нечетность.
1 = Четность.

Бит 5 PARITY ENABLE. Разрешение четности модуля SCI. Этот бит разрешает или запрещает функцию четности. Если модуль SCI находится в мультипроцессорном режиме адресного бита (установка с использованием 3 бита этого регистра), адресный бит включается в вычисление четности (если четность разрешена). Для символов менее восьми бит оставшиеся неиспользуемые биты должны быть маскированы от вычисления четности.
0 = Четность запрещена. Бит четности не формируется при передаче или ожидается при приёме.
1 = Четность разрешена.

Бит 4 SCI ENA. Бит разрешения коммуникации модуля SCI. Этот бит должен быть установлен в 1 для корректной работы.

Бит 3 ADDR/IDLE MODE. Управляющий бит мультипроцессорного режима. Этот бит выбирает один из мультипроцессорных протоколов:
0 = Протокол режима простоя линии.
1 = Протокол режима адресного бита.
Мультипроцессорная коммуникация отличается от других режимов тем, что использует функции SLEEP и TXWAKE (биты SCICTL1.2 и SCICTL1.3, соответственно). Режим простоя линии обычно используется для нормальной коммуникации потому, что режим адресного бита добавляет дополнительный бит во фрейм. Режим простоя линии не добавляет этот дополнительный бит и совместим с RS-232 типом коммуникации.

Биты 2-0 SCI CHAR2–0. Управляющие биты длины символа. Эти биты выбирают длину символов модуля SCI от одного до восьми бит. Символы с длиной меньше восьми бит выровнены по правому краю в SCIRXBUF и SCIRXEMU, и старшие разряды заполнены нулями в SCIRXBUF. В SCITXBUF старшие разряды не заполняются нулями. В таблице 68 представлены значения бит и длина символов для бит SCI CHAR2–0

Таблица 68 – Значения бит SCI CHAR 2–0 и длина символов

Значения бит SCI CHAR2–0			Длина символов (биты)
SCI CHAR2	SCI CHAR1	SCI CHAR0	
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

Управляющий регистр модуля SCI 1 (SCICTL1)

Управляющий регистр модуля SCI 1 (SCICTL1) управляет разрешением приемника/передатчика, функциями TXWAKE и SLEEP, разрешением внутреннего тактового сигнала и программным сбросом модуля SCI. Описание бит SCICTL1 показано на рисунке 106.

7	6	5	4	3	2	1	0
Зарезервировано	RX ERR INT ENA	SW RESET	CLOCK ENA	TXWAKE	SLEEP	TXENA	RXENA
	RW-0	RW-0	RW-0	RS-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, S – только установка, -0 – значение после сброса

Рисунок 106 – Управляющий регистр модуля SCI 1 (SCICTL1) – адрес 7051h

Бит 7 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.

Бит 6 RX ERR INT ENA. Разрешение приёмника модуля SCI. Установка этого бита разрешает прерывание, если бит RX ERROR (SCIRXST.7) устанавливается из-за появления ошибок.
0 = Приём прерывания ошибки запрещен.
1 = Приём прерывания ошибки разрешен.

Бит 5 SW RESET. Программный сброс модуля SCI (активный низкий уровень). Запись 0 в этот бит инициализирует конечные автоматы модуля SCI и операционные флаги (SCICTL2 и SCIRXST) в условия сброса.
Бит SW RESET не затрагивает каких- либо бит конфигурации и не изменяет состояние бита CLOCK ENA.
Вся затрагиваемая логика удерживается в определенном состоянии сброса, пока 1 не будет записана в SW RESET. Таким образом, после системного сброса необходимо записать 1 в этот бит для разрешения модуля SCI.
Этот бит очищается после определения разрыва приёма (флаг BRKDT, бит SCIRXST.5).
SW RESET затрагивает операционные флаги модуля SCI, но не затрагивает биты конфигурации и не восстанавливает значения сброса. В таблице 69 представлены затрагиваемые флаги.

Как только SW RESET утвержден, флаги замораживаются, пока бит не будет снят.

Примечание – Конфигурация модуля SCI не должна устанавливаться или изменятся, пока бит SW RESET не будет очищен. Необходимо установить все регистры конфигурации перед установкой SW RESET; в противном случае возможно непредсказуемое поведение модуля.

Таблица 69 – Флаги, затрагиваемые SW RESET

Флаг SCI	Регистр.Бит	Значение после SW RESET
TXRDY	SCICTL2.7	1
TX EMPTY	SCICTL2.6	1
RXWAKE	SCIRXST.1	0
PE	SCIRXST.2	0
OE	SCIRXST.3	0
FE	SCIRXST.4	0
BRKDT	SCIRXST.5	0
RXRDY	SCIRXST.6	0
RX ERROR	SCIRXST.7	0

Бит 4 CLOCK ENA. Разрешение внутреннего тактового сигнала модуля SCI. Этот бит определяет источник тактового сигнала модуля на вводе SCICLK (рисунок 106).

0 = Запрещение внутреннего тактового сигнала.

1 = Разрешение внутреннего тактового сигнала.

Бит 3 TXWAKE. Выбор метода запуска передатчика. Бит TXWAKE управляет выбором признака передачи данных в зависимости от того, какой режим передачи (простоя линии или адресного бита) определен в бите ADDR/IDLE MODE (SCICCR.3):

0 = Признак передачи не выбран.

1 = Признак передачи выбран, он зависит от режима: простоя линии или адресного бита:

В режиме простоя линии: запись 1 в TXWAKE, затем запись данных в регистр SCITXBUF для формирования периода простоя в 11 бит.

В режиме адресного бита: запись 1 в TXWAKE, затем запись данных в регистр SCITXBUF для установки адресного бита для этого фрейма в 1.

TXWAKE не очищается битом SW RESET (SCICTL1.5).

Бит 2 SLEEP. Режим ожидания модуля SCI. В мультипроцессорной конфигурации этот бит управляет функцией ожидания приёма. Очистка этого бита переводит модуль SCI в режим ожидания.

0 = Режим ожидания запрещен.

1 = Режим ожидания разрешен.

Приемник продолжает работу, когда установлен SLEEP бит; однако это не обновляет бит готовности буфера приёма (SCIRXST.6, RXRDY) или статусные биты ошибки (SCIRXST.5–2: BRKDT, FE, OE и PE), пока не будет определён адресный байт. Этот бит не очищается при определении адресного байта.

Бит 1 TXENA. Разрешение передатчика модуля SCI. Данные передаются через ввод SCITXD (рисунок 106) только когда установлен TXENA. Если он сброшен, передача останавливается, но только после посылки данных записанных в SCITXBUF.
0 = Передатчик запрещен.
1 = Передатчик разрешен.

Бит 0 RXENA. Разрешение приёмника модуля SCI. Данные принимаются на ввод SCIRXD (рисунок 106) и передаются на сдвиговый регистр приёма и далее на приёмные буферы. Этот бит разрешает или запрещает приёмник (передачу на буферы).
0 = Предотвращение передачи полученных символов в SCIRXEMU и SCIRXBUF буферы приёма.
1 = Пересылка полученных символов в SCIRXEMU и SCIRXBUF.
Очистка RXENA останавливает передачу полученных символов в два буфера приёма, а также останавливает формирование прерываний приёмника. Однако сдвиговый регистр приёмника может продолжать собирать символы. Таким образом, если RXENA устанавливается во время принятия символа, полный символ будет передан в регистры буфера приёма, SCIRXEMU и SCIRXBUF.

Регистры выбора скорости двоичной передачи (SCIHBAUD и SCILBAUD)

Значения регистров выбора скорости двоичной передачи (SCIHBAUD и SCILBAUD) определяют скорость двоичной передачи для модуля SCI. Описание бит SCIHBAUD и SCILBAUD показано на рисунках 107 и 108, соответственно.

7	6	5	4	3	2	1	0
BAUD15 (ст. бит)	BAUD14	BAUD13	BAUD12	BAUD11	BAUD10	BAUD9	BAUD8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 107 – Регистр выбора скорости двоичной передачи старших разрядов (SCIHBAUD) – адрес 7052h

7	6	5	4	3	2	1	0
BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0 (мл. бит)
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 108 – Регистр выбора скорости двоичной передачи младших разрядов (SCILBAUD) – адрес 7053h

Биты BAUD15–BAUD0. Выбор 16-битной двоичной передачи. SCIHBAUD

15-0 (старший разряд) и SCILBAUD (младший разряд) объединяются для формирования 16-битного значения скорости двоичной передачи.
Внутренне сформированный последовательный тактовый сигнал определяется частотой SYSCLK и регистрами выбора скорости двоичной передачи. Модуль SCI использует 16-битное значение этих регистров для выбора одного из 64K различных коэффициентов последовательного тактового сигнала для коммуникационных режимов.
Скорость двоичной передачи модуля SCI для различных коммуникационных режимов определяется следующими путями:

– Для BRR=1 до 65535

$$\text{Асинхронный бод SCI} = \frac{\text{SYSCLK}}{(\text{BRR}+1)\times 8},$$

$$\text{BRR} = \frac{\text{SYSCLK}}{(\text{AАсинхроный бод SCI})\times 8} - 1.$$

– Для BRR=0

$$\text{Асинхронный бод SCI} = \frac{\text{SYSCLK}}{16},$$

где BRR – 16-битное значение регистров выбора скорости двоичной передачи.

Управляющий регистр модуля SCI 2 (SCICTL2)

Управляющий регистр модуля SCI 2 (SCICTL2) разрешает прерывания готовности приёма, обнаружения разрыва и готовности передачи, а так же флаги готовности и опустошения передатчика. Описание бит SCICTL2 показано на рисунке 109.

7	6	5-2	1	0
TXRDY	TX EMPTY	Зарезервировано	RX/BK INT ENA	TX INT ENA
R-1	R-1		RW-0	RW-0

R – доступ чтения, W – доступ записи, -n – значение после сброса

Рисунок 109 – Управляющий регистр модуля SCI 2 (SCICTL2) – адрес 7054h

Бит 7 TXRDY. Флаг готовности буферного регистра передатчика. Когда установлен, этот бит указывает, что буферный регистр передачи SCITXBUF готов к приёму другого символа. Запись данных в SCITXBUF автоматически очищает этот бит. Когда установлен, этот флаг утверждает запрос прерывания передатчика, если бит разрешения прерывания TX INT ENA (SCICTL2.0) так же установлен. TXRDY устанавливается в 1 разрешением бита SW RESET (SCICTL.2) или системным сбросом.
0 = SCITXBUF полон.

1 = SCITXBUF готов к приёму следующего символа.

Бит 6 TX EMPTY. Флаг опустошения передатчика. Значение этого флага показывает содержимое буферного регистра передатчика (SCITXBUF) и сдвигового регистра (TXSHF). Активный SW RESET (SCICTL1.2) или системный сброс устанавливают этот бит. Этот бит не запрашивает прерывания.
0 = Буфер передатчика или сдвиговый регистр, или они оба загружены данными.
1 = Буфер передатчика и сдвиговый регистр пусты.

Биты 5-2 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.

Бит 1 RX/BK INT ENA. Разрешение прерывания буфера/разрыва приёмника. Этот бит управляет запросом прерывания вызываемого установкой или флага RXRDY или флага BRKDT (биты SCIRXST.6 и SCIRXST. 5). Однако RX/BRK INT ENA не предотвращает установление этих флагов.
0 = Запрещение прерывания RXRDY/BRKDT.
1 = Разрешение прерывания RXRDY/BRKDT.

Бит 0 TX INT ENA. Разрешение прерывания SCITXBUF. Этот бит управляет совершением запроса прерывания установкой бита флага TXRDY (SCICTL2.7). Однако, он не предотвращает установку флага TXRDY (установлен означает, что SCITXBUF, готов к приёму другого символа).
0 = Запрещает прерывание TXRDY.
1 = Разрешает прерывание TXRDY.

Статусный регистр приёмника (SCIRXST)

Статусный регистр приёмника (SCIRXST) состоит из семи бит, которые являются флагами статуса приёмника (два из которых могут формировать запросы прерывания). Каждый раз, когда полный символ был передан в буферы приёма (SCIRXEMU и SCIRXBUF), статусные флаги обновляются. Каждый раз, когда буферы считываются, флаги очищаются. На рисунке 111 представлены связи между битами этого регистра. Описание бит SCIRXST показано на рисунке 110.

7	6	5	4	3	2	1	0
RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	Зарезервировано
R-0	R-0	R-0	R-0	R-0	R-0	R-0	

R – доступ чтения, -0 – значение после сброса

Рисунок 110 – Статусный регистр приёмника (SCIRXST) – адрес 7055h

Бит 7 RX ERROR. Флаг ошибки приёмника модуля SCI. Флаг RX ERROR указы-

вает на то, что один из флагов ошибки установлен в статусном регистре приёмника. RX ERROR является логическим ИЛИ для разрешения флагов определения разрыва, ошибки фрейма, перегрузки и четности (биты 5-2: BRKDT, FE, OE и PE).
0 = Не установлено никаких флагов ошибки.
1 = Флаг(и) ошибки установлен(ы).
Установка этого бита в 1 приводит к прерыванию, если установлен бит RX ERR INT ENA (SCICTL1.6). Этот бит может быть использован для быстрой проверки состоянии ошибки во время обслуживающей программы прерывания. Этот флаг ошибки не может быть очищен напрямую; он очищается активным SW RESET или системным сбросом.

Бит 6 RXRDY. Флаг готовности приёмника модуля SCI. Когда новый символ готов для чтения из SCIRXBUF, приёмник устанавливает этот бит, и прерывание приёмника формируется, если бит RX/BK INT ENA (SCICTL2.1) установлен в 1. RXRDY очищается чтением SCIRXBUF, активным SW RESET или системным сбросом.
0 = Нет нового символа в SCIRXBUF.
1 = Символ готов для чтения из SCIRXBUF.

Бит 5 BRKDT. Флаг определения разрыва модуля SCI. Модуль SCI устанавливает этот бит, когда происходит условие разрыва. Условие разрыва происходит, когда модуль SCI принимает строку данных (SCIRXD, рисунок 110), остающуюся в низком уровне в течение десяти бит, начинающихся после утерянного первого стопового бита. Возникновение разрыва приводит к формированию прерывания приёмника, если бит RX/BK INT ENA установлен в 1, но это не приводит к загрузке буфера приёмника. Прерывание BRKDT может возникнуть, даже если SLEEP бит приёмника установлен в 1.
BRKDT очищается активным SW RESET или системным сбросом. Он не очищается приёмом символа после определения разрыва. Для того чтобы принять больше символов, модуль SCI должен быть сброшен переключением бита SW RESET или системным сбросом.
0 = Нет условия разрыва.
1 = Произошло условие разрыва.

Бит 4 FE. Флаг ошибки фрейма модуля SCI. Модуль SCI устанавливает этот бит, когда ожидаемый стоповый бит не найден. Проверяется только первый стоповый бит. Утерянный стоповый бит указывает, что синхронизация с битом запуска была потеряна, и символ был некорректно создан. Флаг очищается активным SW RESET или системным сбросом.
0 = Не определено ошибки фрейма.
1 = Определена ошибка фрейма.

Бит 3 OE. Флаг ошибки перегрузки модуля SCI. Модуль SCI устанавливает этот бит, когда символ передается в SCIRXEMU и SCIRXBUF перед полным чтением процессором предыдущего символа. Предыдущий символ перезаписывается и теряется. Флаг OE сбрасывается активным SW RESET или

системным сбросом.
0 = Не определено ошибки перегрузки.
1 = Определена ошибка перегрузки.

Бит 2 PE. Флаг ошибки четности модуля SCI. Этот бит флага устанавливается, когда символ принят с несоответствием между количеством бит и его битом четности. Адресный бит включается в вычисление. Если формирование и определение не разрешены, флаг PE запрещен и считывается как 0. Бит PE сбрасывается активным SW RESET или системным сбросом.
0 = Нет ошибки четности или четность запрещена.
1 = Определена ошибка четности.

Бит 1 RXWAKE. Флаг определения запуска приёмника. Установка этого бита в 1 указывает на определение условия запуска приёмника. В мультипроцессорном режиме адресного бита (SCICCR.3 = 1) RXWAKE отображает значение адресного бита для символа, содержащегося в SCIRXBUF. В мультипроцессорном режиме простоя линии, RXWAKE устанавливается, если строка данных SCIRXD определена как простой (эта строка является входом RXSHF, как показано на рисунке 91). RXWAKE имеет доступ только для чтения, очищается только следующим:
- передачей первого байта после адресного байта в SCIRXBUF;
- чтением SCIRXBUF;
- активным SW RESET;
- системным сбросом.

Бит 0 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.

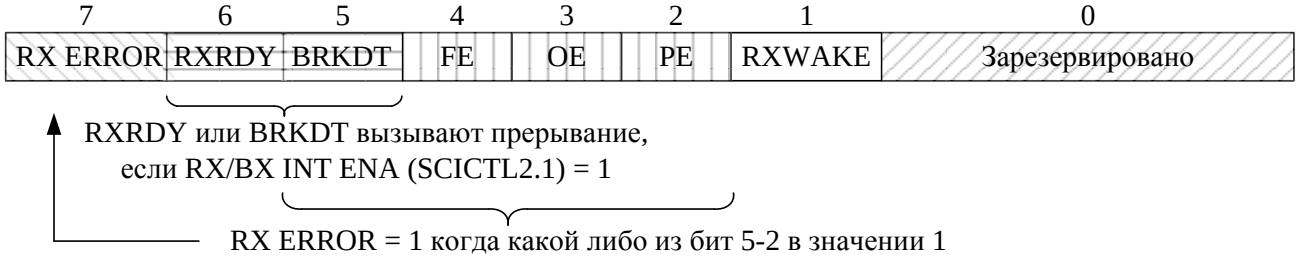


Рисунок 111 – Связи бит в SCIRXST

Регистры буфера данных приёмника

Принятые данные передаются от RXSHF в SCIRXEMU и SCIRXBUF. Когда передача завершена, устанавливается флаг RXRDY (бит SCIRXST.6), указывающий, что принятые данные готовы для чтения. Оба регистра содержат одинаковые данные; они имеют раздельные адреса, но не имеют физически разделённых буферов. Единственное отличие в том, что SCIRXEMU не очищает флаг RXRDY; однако, чтение SCIRXBUF очищает этот флаг.

Буфер данных эмуляции (SCIRXEMU)

Для нормальной операции приёма данных модуля SCI, считываются данные, полученные от SCIRXBUF. SCIRXEMU используется в основном эмулятором, потому что он может непрерывно считывать данные, полученные от обновлений изображения без очистки флага RXRDY. SCIRXEMU очищается системным сбросом. Описание бит SCIRXEMU показано на рисунке 112.

7	6	5	4	3	2	1	0
ERXDT7	ERXDT6	ERXDT5	ERXDT4	ERXDT3	ERXDT2	ERXDT1	ERXDT0
R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x

R – доступ чтения, -n – значение после сброса (x = неопределённый)

Рисунок 112 – Регистр буфера данных эмуляции модуля SCI (SCIRXEMU) – адрес 7056h

Буферный регистр данных приёмника (SCIRXBUF)

Когда текущие данные получены и сдвинуты из RXSHF в буфер приёма, устанавливается бит флага RXRDY и данные готовы для чтения. Если установлен бит RX/BK INT ENA (SCICTL2.1), этот сдвиг вызывает прерывание. Когда происходит чтение из SCIRXBUF, флага RXRDY сбрасывается. SCIRXEMU очищается системным сбросом. Описание бит SCIRXBUF показано на рисунке 113.

7	6	5	4	3	2	1	0
RXDT7	RXDT6	RXDT5	RXDT4	RXDT3	RXDT2	RXDT1	RXDT0
R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x

R – доступ чтения, -n – значение после сброса (x = неопределённый)

Рисунок 113 – Регистр буфера данных приёмника модуля SCI (SCIRXBUF) – адрес 7057h

Регистр буфера данных передатчика (SCITXBUF)

Передаваемые биты данных записываются в регистр буфера данных передачи (SCITXBUF). Передача данных из этого регистра в сдвиговый регистр передатчика TXSHF устанавливает флаг TXRDY (SCICTL2.7), указывающий, что SCITXBUF готов к приёму следующих данных. Если установлен бит TX INT ENA (SCICTL2.0), эта передача данных вызывает прерывание. Эти биты должны быть выровнены по правому краю, потому что выровненные слева биты, игнорируются для символов меньшей длины, чем восемь бит. Описание бит SCITXBUF показано на рисунке 114.

7	6	5	4	3	2	1	0
TXDT7	TXDT6	TXDT5	TXDT4	TXDT3	TXDT2	TXDT1	TXDT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 114 – Регистр буфера данных передатчика модуля SCI

(SCITXBUF) – адрес 7059h

Регистр управления порта модуля SCI 2 (SCIPC2)

Регистр управления порта модуля SCI 2 (SCIPC2) управляет функциями выводов SCITXD и SCIRXD. Описание бит SCIPC2 показано на рисунке 115.

7	6	5	4	3	2	1	0
SCITXD DATA IN	SCITXD DATA OUT	SCITXD FUNCTION	SCITXD DATA DIR	SCIRXD DATA IN	SCIRXD DATA OUT	SCIRXD FUNCTION	SCIRXD DATA DIR
RW-0	RW-0	RW-0	RW-0	R-x	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса (x = неопределённый)

Рисунок 115 – Регистр управления порта модуля SCI 2 (SCIPC2) – адрес 705Eh

Бит 7 SCITXD DATA IN. Содержит текущее значение вывода SCITXD.
0 = Значение вывода SCITXD считывается как 0.
1 = Значение вывода SCITXD считывается как 1.

Бит 6 SCITXD DATA OUT. Выходное значение на выводе SCITXD. Этот бит содержит выходные данные на SCITXD, если он является универсальным цифровым вводом - выводом. Для этой конфигурации нужно очистить бит SCIPC2.5 до 0 и установить бит SCIPC2.4 в 1.

Бит 5 SCITXD FUNCTION. Описывает функции вывода SCITXD.
0 = SCITXD – универсальный цифровой ввод - вывод.
1 = SCITXD – Вывод передачи модуля SCI.

Бит 4 SCITXD DATA DIR. Описывает направление поступления данных вывода SCITXD. Если SCITXD сконфигурирован как универсальный цифровой ввод - вывод (бит 5 = 0), этот бит определяет направление SCITXD:
0 = SCITXD – цифровой ввод.
1 = SCITXD – цифровой вывод.

Бит 3 SCIRXD DATA IN. Содержит текущее значение вывода SCIRXD.
0 = Значение вывода SCIRXD считывается как 0.
1 = Значение вывода SCIRXD считывается как 1.

Бит 2 SCIRXD DATA OUT. Выходное значение на выводе SCIRXD. Этот бит содержит выходные данные на SCIRXD, если он является универсальным цифровым вводом - выводом. Для этой конфигурации нужно очистить бит SCIPC2.1 до 0 (универсальный вывод) и установить бит SCIPC2.4 в 1 (цифровой выход).
0 = Выходное значение вывода SCIRXD равно 0.
1 = Выходное значение вывода SCIRXD равно 1.

Бит 1 SCIRXD FUNCTION. Описывает функции вывода SCIRXD.
0 = SCIRXD – универсальный цифровой ввод - вывод.
1 = SCIRXD – Вывод приёма модуля SCI.

Бит 0 SCIRXD DATA DIR. Описывает направление поступления данных вывода SCIRXD. Если SCIRXD сконфигурирован как универсальный цифровой ввод - вывод (бит SCIPC2.1 = 0), этот бит определяет направление SCIRXD:
0 = SCIRXD – цифровой ввод.
1 = SCIRXD – цифровой вывод.

Регистр управления приоритетом (SCIPRI)

Регистр управления приоритетом модуля SCI (SCIPRI) содержит биты выбора приоритета прерывания приёмника и передатчика. Описание бит SCIPRI показано на рисунке 116.

7	6	5	4	3-0
Зарезервировано	SCITX PRIORITY	SCIRX PRIORITY	SCI ESPEN	Зарезервировано
	RW-0	RW-0	RW-0	

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 116 – Регистр управления приоритетом модуля SCI (SCIPRI) – адрес 705Fh

Бит 7 Зарезервировано. Чтения неопределенны, запись не имеют эффекта.

Бит 6 SCITX PRIORITY. Выбор приоритета прерывания передатчика модуля SCI. Этот бит определяет уровень приоритета для прерываний передатчика модуля SCI.
0 = Прерывания запрашиваются с высоким приоритетом.
1 = Прерывания запрашиваются с низким приоритетом.

Бит 5 SCIRX PRIORITY. Выбор приоритета прерывания приёмника модуля SCI. Этот бит определяет уровень приоритета для прерываний приёмника модуля SCI.
0 = Прерывания запрашиваются с высоким приоритетом.
1 = Прерывания запрашиваются с низким приоритетом.

Бит 4 SCI ESPEN. Разрешение приостановки эмулятора модуля SCI. Этот бит действует только при отладке программы с помощью эмулятора. Этот бит определяет, как модулю SCI работать, когда программа приостановлена.
0 = Когда приостановленный сигнал становится в высокий уровень, модуль SCI продолжает операцию, пока не завершатся текущие функции передачи и приёма.
1 = Когда приостановленный сигнал становится в высокий уровень,

конечный автомат модуля SCI замораживается.

Биты 3-0 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.

13.4.7 Пример инициализации модуля SCI

Пример 10 показывает программу инициализации асинхронной коммуникации.

Пример 10 – Программа инициализации асинхронной коммуникации.

```
;*************************************************************
; Имя файла: 1867ВЦ10Т SCI Пример кода в режиме простоя линии
; Описание: Эта программа использует модуль SCI для реализации
простой программы асинхронной коммуникации. SCI инициализируется
для работы в режиме простоя линии, с 8 символьными битами 1,
стоповым битом и нечетностью. Регистры скорости двоичной переда-
чи SCI (BRR) Установлены на приём и передачу данных при 19200
бод. SCI формирует прерывание каждый раз, когда символ загружа-
ется в буфер приёмника (SCIRXBUF). Обслуживающая программа пре-
рывания (ISR) считывает буфер приёма и определяется, если была
нажата клавиша возврата каретки <CR>. Если эти действия произве-
дены, то передается строка символов ”Ready”. Если нет, то не пе-
редается никакой строки символов.
;*************************************************************
; Операторы SET для устройств 1867ВЦ10Т зависят от этих
устройств. Адреса SET используемые в этом примере справедливы
для устройств с одним модулем SCI. Для нахождения точных адресов
модулей в ячейках памяти нужно обратиться к спецификации
устройств.
;    .include ”1867wc10treg.h” ; содержит список операторов SET
;для всех регистров ИС 1867ВЦ10Т.
; Следующие операторы SET для SCI содержатся в 1867wc10treg.h.
; Регистры последовательного коммуникационного интерфейса (SCI)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SCICCR   .set 07050h    ;Управляющий регистр связи SCI
SCICTL1  .set 07051h    ;Управляющий регистр SCI 1
SCIHBAUD .set 07052h    ;Регистр скорости двоичной передачи SCI
SCILBAUD .set 07053h    ;Регистр скорости двоичной передачи SCI
SCICTL2  .set 07054h    ;Управляющий регистр SCI 2
SCIRXST  .set 07055h    ;Регистр статуса приёмника SCI
SCIRXEMU .set 07056h    ;Буфер данных эмуляции SCI
SCIRXBUF .set 07057h    ;Буфер данных приёмника SCI
SCITXBUF .set 07059h    ;Буфер данных передатчика SCI
SCIPC2   .set 0705Eh    ;Управляющий регистр порта SCI 2
SCIPRI   .set 0705Fh    ;Управляющий регистр приоритета SCI
;-------------------------------------------------------------
; Описание постоянных
;-------------------------------------------------------------
LENGTH   .set 00005h         ;Длина передаваемого потока данных
;-------------------------------------------------------------
; Описание переменных
;-------------------------------------------------------------
```

```
.bss DATA_OUT, LENGTH   ;Расположение байта LENGTH передаваемого
                        ;символьного потока
.bss GPR0, 1            ;Регистр общего назначения.
;---------------------------------------------------------------
; Определение макрокоманд
;---------------------------------------------------------------
KICK_DOG .macro         ;Макрокоманда сброса сторожевой схемы
         LDP  #00E0h
         SPLK #055h, WDKEY
         SPLK #0AAh, WDKEY
         LDP  #0h
         .endm
;---------------------------------------------------------------
; Коды битов для программы проверки кода (BIT)
;---------------------------------------------------------------
BIT15 .set 0000h        ;Код бита для 15
BIT14 .set 0001h        ;Код бита для 14
BIT13 .set 0002h        ;Код бита для 13
BIT12 .set 0003h        ;Код бита для 12
BIT11 .set 0004h        ;Код бита для 11
BIT10 .set 0005h        ;Код бита для 10
BIT9  .set 0006h        ;Код бита для 9
BIT8  .set 0007h        ;Код бита для 8
BIT7  .set 0008h        ;Код бита для 7
BIT6  .set 0009h        ;Код бита для 6
BIT5  .set 000Ah        ;Код бита для 5
BIT4  .set 000Bh        ;Код бита для 4
BIT3  .set 000Ch        ;Код бита для 3
BIT2  .set 000Dh        ;Код бита для 2
BIT1  .set 000Eh        ;Код бита для 1
BIT0  .set 000Fh        ;Код бита для 0
;---------------------------------------------------------------
; Инициализированные передаваемые данные для обслуживающей про-
граммы прерывания
;---------------------------------------------------------------
       .data
TXDATA .word 0052h      ;Hex эквивалент ASCII символа 'R'
       .word 0065h      ;Hex эквивалент ASCII символа 'e'
       .word 0061h      ;Hex эквивалент ASCII символа 'a'
       .word 0064h      ;Hex эквивалент ASCII символа 'd'
       .word 0079h      ;Hex эквивалент ASCII символа 'y'
;---------------------------------------------------------------
; Описания адресов векторов
;---------------------------------------------------------------
         .sect ".vectors"
RSVECT   B     START     ; PM 0  Вектор сброса                1
INT1     B     INT1_ISR  ; PM 2  Уровень прерывания 1         4
INT2     B     PHANTOM   ; PM 4  Уровень прерывания 2         5
INT3     B     PHANTOM   ; PM 6  Уровень прерывания 3         6
INT4     B     PHANTOM   ; PM 8  Уровень прерывания 4         7
INT5     B     PHANTOM   ; PM A  Уровень прерывания 5         8
INT6     B     PHANTOM   ; PM C  Уровень прерывания 6         9
```

```
RESERVED B     PHANTOM   ; PM E  (Анализ прерывания)         10
SW_INT8  B     PHANTOM   ; PM 10 Программное прерывание      -
SW_INT9  B     PHANTOM   ; PM 12 Программное прерывание      -
SW_INT10 B     PHANTOM   ; PM 14 Программное прерывание      -
SW_INT11 B     PHANTOM   ; PM 16 Программное прерывание      -
SW_INT12 B     PHANTOM   ; PM 18 Программное прерывание      -
SW_INT13 B     PHANTOM   ; PM 1A Программное прерывание      -
SW_INT14 B     PHANTOM   ; PM 1C Программное прерывание      -
SW_INT15 B     PHANTOM   ; PM 1E Программное прерывание      -
SW_INT16 B     PHANTOM   ; PM 20 Программное прерывание      -
TRAP     B     PHANTOM   ; PM 22 Вектор сист. прерывания     -
NMI      B     PHANTOM   ; PM 24 Не маскированное прерывание 3
EMU_TRAP       PHANTOM   ; PM 26 Сист. прерывания эмулятора  2
SW_INT20 B     PHANTOM   ; PM 28 Программное прерывание      -
SW_INT21 B     PHANTOM   ; PM 2A Программное прерывание      -
SW_INT22 B     PHANTOM   ; PM 2C Программное прерывание      -
SW_INT23 B     PHANTOM   ; PM 2E Программное прерывание      -
;=============================================================
; Основная программа
;=============================================================
        .text
START: SETC INTM        ;Запрещение прерываний
    CLRC SXM            ;Очистка режима дополнительного знакового
                        ;разряда
    CLRC OVM            ;Перезапуск режима переполнения
    CLRC CNF            ;Конфигурация блока B0 в память данных.
    LDP  #00E0h
    SPLK #006Fh, WDCR   ;Запрещение сторожевой схемы, если WDDIS
= 3,3 V
    KICK_DOG            ;Сброс сторожевой схемы
    LDP  #00E0h
    SPLK #00BBh, CKCR1 ;CLKIN(BQ)=10МГц, CPUCLK=20МГц
    SPLK #00C3h, CKCR0 ;SYSCLK=CPUCLK/2,
    SPLK #40C0h, SYSCR ;CLKOUT=CPUCLK
    LDP  #0000h
    SPLK #4h, GPR0
    OUT  GPR0, WSGR     ;Установка XMIF для запуска в режимах
    ;ожидания
;=============================================================
; Инициализация B2 RAM в нули.
;=============================================================
    LAR  AR2, #B2_SADDR ;AR2 -> B2 начальный адрес
    MAR  *, AR2          ;Установка ARP=AR2
    ZAC                  ;Установка ACC = 0
    RPT   #1fh              ;Установка повтора счетчика для 31+1
циклов
    SACL *+              ;Запись нулей в B2 RAM
;=============================================================
; Инициализация DATAOUT передаваемыми данными.
;=============================================================
    LAR  AR2, #B2_SADDR ;Сброс AR2 -> B2 начальный адрес
    LAR  AR1, #DATA_OUT ;AR1 -> DATAOUT начальный адрес
```

```
    RPT  #04h           ;Установка повтора счетчика для 4+1 цик-
лов
    BLPD #TXDATA, *+    ;Загрузка B2 с TXDATA
;=================================================================
; Инициализация подпрограммы управления прерываниями SCI
;=================================================================
SCI_INIT: LDP #00E0h
     SPLK #0037h, SCICCR   ;1 стоповый бит, нечетность, 8
                           ;символьных бит, асинхронный режим,
                           ;протокол простоя линии
     SPLK  #0013h,  SCICTL1    ;Разрешение  TX,  RX,  внутренний
SCICLK,
                           ;запрещение RX ERR, SLEEP, TXWAKE
     SPLK #0002h, SCICTL2  ;Разрешение RX INT, запрещение TX INT
     SPLK #0000h, SCIHBAUD
     SPLK  #0040h,  SCILBAUD  ;Скорость  двоичной  передачи=19200
бит/с
                           ;(10 МГц SYSCLK)
     SPLK #0022h, SCIPC2   ;Разрешение вводов TXD & RXD
     SPLK #0033h, SCICTL1  ;Освобождение SCI от сброса.
     LAR  AR0, #SCITXBUF   ;Загрузка AR0 адресом SCI_TX_BUF
     LAR  AR1, #SCIRXBUF   ;Загрузка AR1 адресом SCI_RX_BUF
     LAR  AR2, #B2_SADDR    ;Загрузка AR2 стартовым адресом данных
TX
     LDP  #0
     LACC IFR              ; Загрузка ACC флагами прерывания
     SACL IFR              ;Очистка всех незавершенных флагов
                           ;прерываний
     SPLK  #0001h,  IMR        ;Демаскирование уровня прерывания
INT1
     CLRC INTM             ;Разрешение прерываний
;=================================================================
; Основная подпрограмма
;=================================================================
MAIN:                           ;Основная часть кода начинается
здесь
                           ;ввести адресные коды
    NOP
    NOP
;   ....                   ;ввести адресные коды
   B MAIN                     ;Основная часть кода завершена в
один
;проход, возвращение обратно для ;продолжения.
;=================================================================
; I S R - INT1_ISR
; Описание: INT1_ISR сначала определяется, если SCI RXINT вызы-
вает ;прерывание. Если это произошло специальная ISR SCI считы-
вает ;символ в буфере RX. Если символ принят в соответствии с
возвратом ;каретки, <CR>, строка символов "Ready" передается.
Если символ ;принят не в соответствии с возвратом каретки <CR>
ISR возвращается ;к основной программе без передачи строки сим-
```

```
волов. Если SCI RXINT ;не вызвал прерывания, значение '0x0bad'
будет хранится в ;аккумуляторе и программа зациклится в BAD_INT.
;================================================================
INT1_ISR:  LDP  #00E0h
      LACL SYSIVR        ;Загрузка периферийного INT адреса
                         ;вектора
           SUB  #0006h   ;Вычитание начального номера RXINT
                         ;из вышеперечисленного
           BCND SCI_ISR, EQ    ;Проверка RXINT вызвавшего преры-
вание
       B   BAD_INT       ;Если нет, произошло не верное
                         ;прерывание
SCI_ISR:   MAR  *, AR1   ;Установка ARP=AR1
           LACC *, AR2     ;Загрузка ACC символом из буфера
RX
           SUB  #000Dh   ;Проверка если <CR> была нажата:
           BCND XMIT_CHAR, EQ ;YES? Передать данные.
       B   ISR_END       ;N0? Возврат от INT1_ISR.
XMIT_CHAR: LACC *+, AR0    ;Загрузка передаваемого символа в
ACC
           BCND ISR_END, EQ   ;Проверка на нулевой символ
                         ;YES? Возврат от INT1_ISR.
           SACL *, AR2   ;NO? Загрузка символа в буфер
                         ;передачи.
XMIT_RDY:  BIT  SCICTL2, BIT7 ;Проверка бита TXRDY
           BCND XMIT_RDY, NTC ;Если TXRDY=0, то повторить цикл
       B   XMIT_CHAR     ;если нет, то передать следующий
                         ;символ
ISR_END:    LAR  AR2, #B2_SADDR;перезагрузка AR2 стартовым адре-
сом
                         ;данных TX
           CLRC INTM     ;Очистка INT флага маски
           RET           ;Возврат от INT1_ISR
BAD_INT:   LACC #0BADh   ;Загрузка ACC значением "bad"
       B   BAD_INT       ;Повтор цикла
;================================================================
; I S R - PHANTOM
;
; Описание: Пустая ISR, используется для захвата ложных прерыва-
ний.
;================================================================
PHANTOM:   B   PHANTOM
```

13.5 Модуль последовательного периферийного интерфейса SPI

В этом подразделе рассматриваются архитектура, функции и программирование модуля последовательного периферийного интерфейса (SPI).

Примечание – Этот модуль соединён с 16-битной периферийной шиной как 8-битная периферия, поэтому чтения бит 15-8 не определены; записи не имеют эффекта.

13.5.1 Обзор модуля SPI

Модуль SPI – это высокоскоростной синхронный последовательный порт ввода-вывода, который позволяет потоку последовательных данных с запрограммированной длиной (от одного до восьми бит) приниматься и передаваться в/из устройства при программируемой скорости передачи двоичных данных. Модуль SPI обычно используется для коммуникации между контроллером ЦПОС и внешними периферийными устройствами или другим контроллером. Типичные применения модуля SPI включают внешний ввод - вывод от устройства с расширенными выводами, такими как сдвиговые регистры, драйверы дисплея и модуля АЦП.

Примечание – для удобства ссылки на биты регистра используется имя регистра и следующий за ним номер бита. Например, представление 7 бита буферного регистра эмуляции модуля SPI (SPIEMU) будет SPIEMU.7.

Физическое описание модуля SPI

Четырёхконтактный модуль SPI, показанный на рисунке 117, состоит из:

– Четырёх вводов - выводов:

– SPISIMO (вход ведомого (Slave)/выход ведущего (Master)) управляемый битами в SPIPC2,

– SPISOMI (выход ведомого (Slave)/вход ведущего (Master)) управляемый битами в SPIPC2,

– SPICLK (тактовый сигнал модуля SPI), управляемый битами в SPIPC1,

– SPISTE (строб модуля SPI), управляемый битами в SPIPC1.

– Режимов работы ведущего (Master) и ведомого (Slave).

– Последовательного входного буферного регистра модуля SPI (SPIBUF). Этот буферный регистр содержит данные, полученные из сети и готовые для считывания процессором.

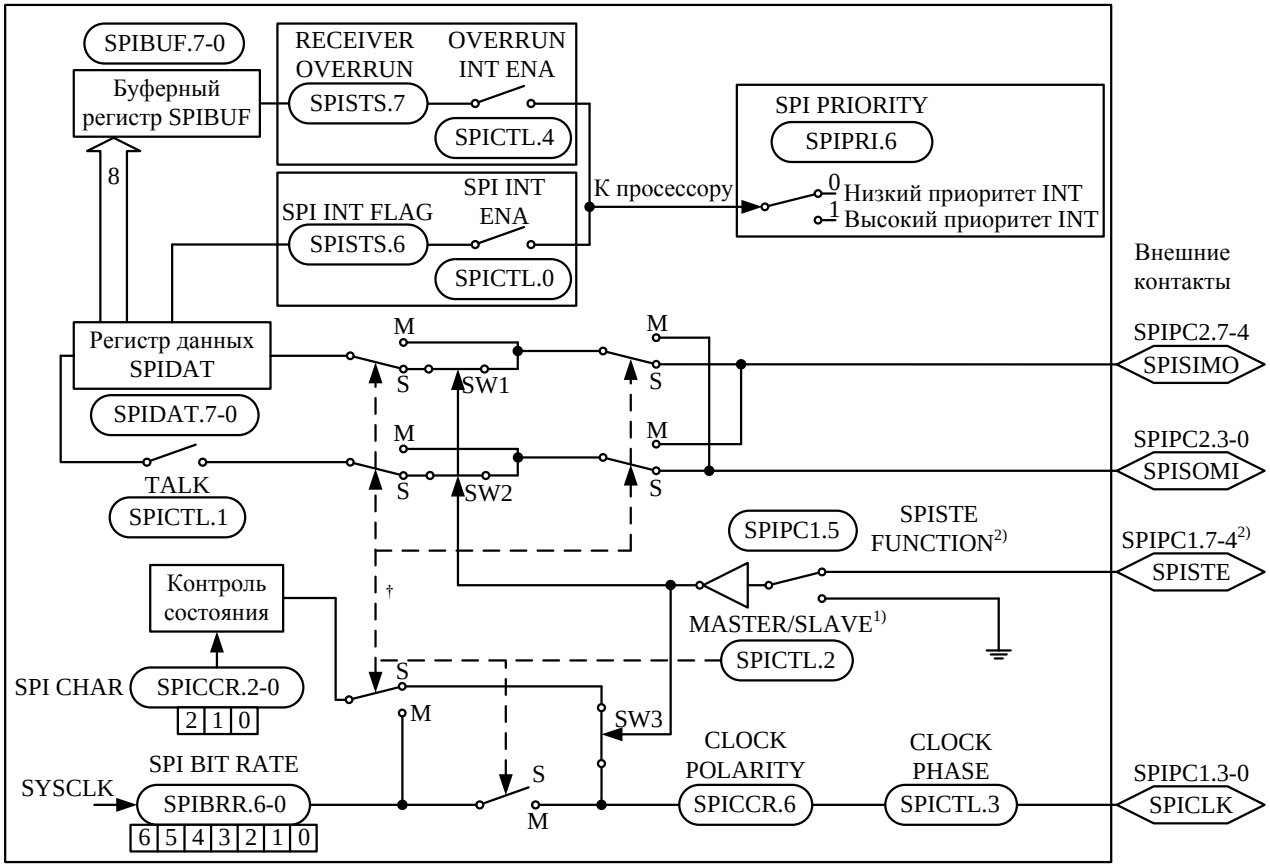
– Последовательного регистра данных модуля SPI (SPIDAT). Этот сдвиговый регистр данных используется как сдвиговый регистр приёма/передачи.

– SPICLK управления фазой и полярностью.

– Логики управления состоянием.

– Картированных в память управляющих и статусных регистров.

Основной функцией ввода строба (SPISTE) является разрешение передачи для модуля SPI в режиме ведомого (Slave). Он стробирует сдвиговый регистр, таким образом, он не может принимать данные. Он так же устанавливает ввод SOMI в третье состояние. Когда модуль SPI находится в режиме ведущего (Master), ввод SPISTE всегда функционирует как универсальный цифровой ввод - вывод и может работать как линия выбора модуля ведомого (Slave) SPI.



1) Структурная схема приведена в режиме ведомого (Slave)

2) Ввод SPISTE запрещен, это значит, что данные могут быть переданы или приняты в этом режиме. Следует отметить, что ключи SW1, SW2 и SW3 закрыты в этой конфигурации.

Рисунок 117 – Структурная схема четырёхконтактного модуля SPI

Управляющие регистры модуля SPI

Десять регистров в модуле SPI (показанные в таблице 70) управляют функционированием SPI:

– SPICCR (регистр управления конфигурацией модуля SPI). Содержит биты используемые для конфигурирования модуля SPI:

– программный сброс модуля SPI;

– SPICLK выбор полярности;

– три управляющих бита длины символа модуля SPI.

– SPICTL (регистр управления работы модулем SPI). Содержит биты управляющие передачей данных:

– два бита разрешения прерывания модуля SPI;

– SPICLK выбор фазы;

– режим работы (ведущий (Master)/ведомый (Slave));

– разрешение передачи данных.

– SPISTS (статусный регистр модуля SPI). Содержит два статусных бита буфера приёмника:

– RECEIVER OVERRUN;

– SPI INT FLAG.

– SPIBRR (регистр скорости двоичной передачи модуля SPI). Содержит семь бит, определяющих скорость двоичной передачи.

– SPIEMU (регистр буфера эмуляции модуля SPI). Содержит принятые данные. Поддерживает корректную эмуляцию. Для нормальной работы должен быть использован SPIBUF.

– SPIBUF (буфер приема модулем SPI – последовательный входной буферный регистр). Содержит принятые данные.

– SPIDAT (регистр данных модуля SPI). Содержит данные, переданные модулем SPI, действуя как сдвиговый регистр передачи/приёма. Данные, записанные в SPIDAT, сдвигаются наружу в последующие периоды SPICLK. Каждый бит, сдвинутый наружу модулем SPI, сдвигается внутрь другого конца сдвигового регистра.

– SPIPC1 (управляющий регистр порта модуля SPI 1). Содержит управляющие биты для выбора функций вводов SPISTE и SPICLK.

– SPIPC2 (управляющий регистр порта модуля SPI 2). Содержит управляющие биты для выбора функций вводов SPISIMO и SPISOMI.

– SPIPRI (регистр приоритета модуля SPI). Содержит два бита, которые определяют приоритет прерывания и работу модуля SPI при эмуляторе во время приостановок программы.

Таблица 70 – Адреса управляющих регистров SPI

Адрес	Регистр	Имя
7040h	SPICCR	Регистр управления конфигурацией модуля SPI
7041h	SPICTL	Регистр управления работы модулем SPI
7042h	SPISTS	Статусный регистр модуля SPI
7043h	—	Зарезервировано
7044h	SPIBRR	Регистр скорости двоичной передачи модуля SPI
7045h	—	Зарезервировано
7046h	SPIEMU	Регистр буфера эмуляции модуля SPI
7047h	SPIBUF	Последовательный входной буферный регистр модуля SPI
7048h	—	Зарезервировано
7049h	SPIDAT	Регистр данных модуля SPI
704Ah	—	Зарезервировано
704Bh	—	Зарезервировано
704Ch	—	Зарезервировано
704Dh	SPIPC1	Управляющий регистр порта модуля SPI 1
704Eh	SPIPC2	Управляющий регистр порта модуля SPI 2
704Fh	SPIPRI	Регистр приоритета модуля SPI

13.5.2 Функционирование модуля SPI

В это главе описывается функционирование модуля SPI, включая описание режимов работы, прерываний, формата данных, источников тактового сигнала и

инициализации. Представлены типичные временные диаграммы для передачи данных.

Примечание – Ссылка на бит в регистре сокращена, используя акроним регистра и следующий за ним период и номер бита. Например, представление бита MASTER/SLAVE управляющего регистра работы модулем SPI (SPICTL) будет SPICTL.2.

Введение в работу модуля SPI

На рисунке 118 показаны типичные соединения модуля SPI для коммуникации между двумя контроллерами: ведущим (Master) и ведомым (Slave).

Ведущий (Master) инициирует передаче данных посылкой сигнала SPICLK. Для ведомого (Slave) и ведущего (Master) данные сдвигаются наружу сдвиговых регистров при одном фронте SPICLK и защелкиваются в сдвиговом регистре на противоположном фронте SPICLK. Если бит CLOCK PHASE (SPICTL.3) установлен, данные передаются и принимаются в полупериоде перед переключением SPICLK. Как результат, оба контроллера посылают и принимают данные одновременно. Программное обеспечение определяет, будут ли данные значимыми или пустыми. Существуют три возможных метода передачи данных:

– Ведущий (Master) посылает данные, ведомый (Slave) посылает пустые данные.

– Ведущий (Master) посылает данные, ведомый (Slave) посылает данные.

– Ведущий (Master) посылает пустые данные, ведомый (Slave) посылает данные.

Ведущий (Master) может инициировать передачу данных в любое время, потому что он управляет сигналом SPICLK. Программа описывает, как ведущий (Master) определяет, когда ведомый (Slave) готов пересылать данные.

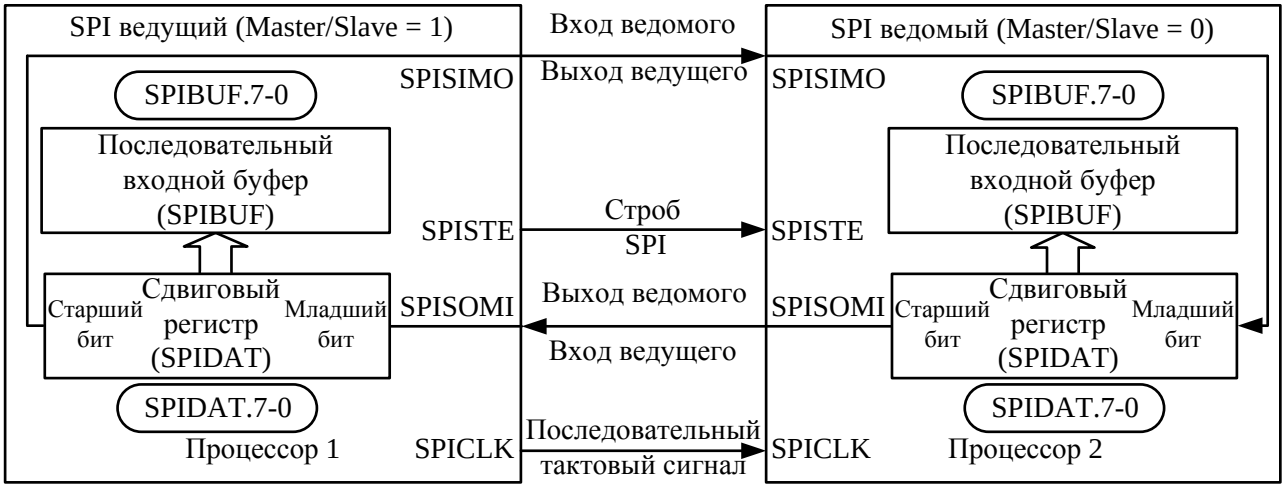


Рисунок 118 – Соединение ведущий (Master)/ведомый (Slave) модулей SPI (условие четырех контактов)

Режимы работы ведущего (Master) и ведомого (Slave) модуля SPI

Модуль SPI может функционировать в режиме ведущего (Master) или ведомого (Slave). Бит MASTER/SLAVE (SPICTL.2) выбирает режим работы и источник сигнала SPICLK.

Режим ведущего (Master)

В режиме ведущего (Master) (MASTER/SLAVE = 1) модуля SPI обеспечивается последовательным тактовым сигналом от ввода SPICLK для полной сети последовательной коммуникации. Данные выводятся на контакте SPISIMO и защелкиваются от ввода SPISOMI.

SPIBRR определяет скорость двоичной передачи приёма и передачи для сети. SPIBRR может выбирать 125 различных коэффициентов передачи данных.

Данные, записанные в SPIDAT, инициируют передачу данных на вводе SPISIMO с начала старшего разряда. Одновременно, полученные данные сдвигаются от ввода SPISOMI к младшим разрядам SPIDAT. Когда выбранное количество бит было передано, данные переносятся, начиная со старшего разряда, в SPIBUF (буферный приёмник) для считывания процессором. Данные в SPIBUF хранятся с выравниванием по правому краю.

Когда указанное количество бит данных было сдвинуто через SPIDAT, происходят следующие события:

– бит SPI INT FLAG (SPISTS.6) устанавливается в 1;

– содержимое SPIDAT переносится в SPIBUF;

– если бит SPI INT ENA (SPICTL.0) установлен в 1, то разрешается прерывание.

В режиме ведущего (Master) ввод SPISTE будет всегда функционировать как универсальный цифровой ввод - вывод, не учитывая значения бита SPISTE FUNCTION (SPIPC1.5). В обычном приложении ввод SPISTE может служить как ввод разрешения кристалла для ведомых модулем SPI устройств (нужно установить этот ввод в низкий уровень перед передачей данных ведущего к ведомому устройству, и установить этот ввод опять в высокий уровень после передачи данных ведущего).

Режим ведомого (Slave)

В режиме ведомого (Slave) (MASTER/SLAVE = 0) данные сдвигаются наружу на вводе SPISIMO и внутрь на вводе SPISOMI. Ввод SPICLK используется как вход для последовательного сдвигающего тактового сигнала, который доставляется от внешнего ведущего (Master) сети. Скорость передачи определяется этим тактовым сигналом. Входная частота SPICLK должна быть больше, чем деленная на восемь частота SYSCLK.

Данные, записанные в SPIDAT, передаются в сеть, когда сигнал SPICLK принят от ведущего (Master) сети. Для принятия данных модуль SPI ожидает, пока ведущий (Master) сети не пошлёт сигнал SPICLK, и затем сдвигает данные на вводе SPISIMO в SPIDAT.

Если данные передаются ведомым (Slave) одновременно, они должны быть записаны в SPIDAT перед началом сигнала SPICLK.

Когда бит TALK (SPICTL.1) очищается, передача данных запрещается и выходная линия (SPISOMI) вводится в высокоимпедансное состояние. Это позволяет большому количеству ведомых устройств быть связанными в сети, но за один раз может взаимодействовать только один ведомый (Slave).

В режиме ведомого (Slave) ввод SPISTE функционирует как универсальный цифровой ввод-вывод, если значение бита SPISTE FUNCTION (SPIPC1.5) очищается (0). Ввод SPISTE функционирует как ввод выбора ведомого (Slave), если SPIPC1.5 установлен в 1. Когда ввод SPISTE функционирует как ввод выбора ведомого (Slave), активный низкий уровень сигнала на вводе SPISTE позволяет подчиненному модулю SPI переносить данные по последовательной линии данных; пассивный высокий уровень сигнала приводит к тому, что сдвиговый регистр ведомого (Slave) модуля SPI останавливается и его последовательный вывод устанавливается в высокоимпедансное состояние. Это позволяет большому количеству ведомых устройств быть связанными в сети, но за один раз может взаимодействовать только один ведомый (Slave).

Прерывания модуля SPI

Для инициализации прерываний модуля SPI используются четыре управляющих бита:

- бит SPI INT ENA (SPICTL.0);
- бит SPI INT FLAG (SPISTS.6);
- бит OVERRUN INT ENA (SPICTL.4);
- бит флага RECEIVER OVERRUN (SPISTS.7).

Бит SPI INT ENA (SPICTL.0)

Когда бит разрешения прерывания модуля SPI установлен и возникло условие прерывания, соответствующее прерывание выполняется.

0 = Запрещение прерываний модуля SPI.

1 = Разрешение прерываний модуля SPI.

Бит SPI INT FLAG (SPISTS.6)

Этот статусный флаг указывает, что символ был помещен в буфер приёма модуля SPI и готов для чтения.

Когда полный символ был сдвинут внутрь или наружу SPIDAT, бит SPI INT FLAG (SPISTS.6) устанавливается, и формируется прерывание, если оно разрешено битом SPI INT ENA. Флаг прерывания остаётся установленным, пока он не очищается одним из следующих четырёх событий:

– Процессор считывает SPIBUF (чтение SPIEMU не очищает SPI INT FLAG).

– Процессор установил режимы OSC Power Down или PLL Power Down с инструкцией IDLE.

- Программа устанавливает бит SPI SW RESET (SPICCR.7).
- Происходит системный сброс.

Для предотвращения формирования другого прерывания, запрос прерывания должен быть однозначно очищен одним из четырёх вышеперечисленных методов. Запрос прерывания может быть временно запрещен очисткой бита SPI INT ENA, пока бит SPI INT FLAG не очищен, однако, запрос прерывания будет подтверждаться повторно, когда бит SPI INT ENA снова установится в 1.

Когда бит SPI INT FLAG установлен, символ помещен в SPIBUF и готов для чтения. Если процессор не прочитал символ и со временем был принят следующий полный символ, новый символ записывается в SPIBUF и устанавливается бит флага RECEIVER OVERRUN (SPISTS.7).

Бит OVERRUN INT ENA (SPICTL.4)

Установка бита разрешения прерывания перегрузки позволяет формировать прерывание всякий раз, когда бит флага RECEIVER OVERRUN (SPISTS.7) установлен аппаратно. Прерывания, формируемые SPISTS.7 и битом SPI INT FLAG (SPISTS.6), совместно используют одинаковый вектор прерывания.

0 = Запрещение прерываний бита флага RECEIVER OVERRUN.

1 = Разрешение прерываний бита флага RECEIVER OVERRUN.

Бит флага RECEIVER OVERRUN (SPISTS.7)

Бит

флага RECEIVER OVERRUN устанавливается всякий раз, когда новый символ был принят и загружен в SPIBUF перед чтением предыдущего принятого символа из SPIBUF. Бит флага RECEIVER OVERRUN должен быть очищен программно.

Формат данных

Три бита (SPICCR.2-0) определяют количество бит (от одного до восьми) в символе данных. Эти биты дают указание логике управления считать количество принятых или переданных бит, чтобы определить, когда полный символ был обработан. Следующее применение для символов с количеством бит менее восьми:

– Данные должны быть выровнены по левому краю, когда записываются в SPIDAT.

– Данные, считываемые из SPIBUF, выровнены по правому краю.

– SPIBUF содержит самый последний принятый символ, выровненный по правому краю, и какие-либо биты, оставшиеся от предыдущей передачи, которые были сдвинуты влево (показано на примере 11).

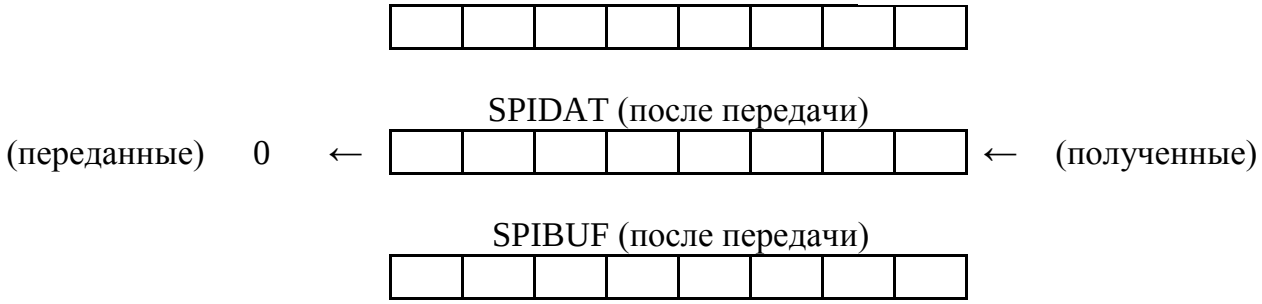
Пример 11 – Перенос бита из SPIBUF.

Условия:

1 Длина передаваемого символа = 1 бит (определяется битами SPICCR.2-0).

2 Текущее значение SPIDAT = 07Bh

SPIDAT (перед передачей)



Примечание – x = 1, если данные SPISOMI в высоком состоянии; x = 0, если данные SPISOMI в низком состоянии; предполагается режим ведущего (Master).

Скорость двоичной передачи и тактовые схемы

Модуль SPI поддерживает 125 различных скоростей двоичной передачи и четыре тактовые схемы. В зависимости от того находится тактовый сигнал модуля SPI в режиме ведомого (Slave) или ведущего (Master), ввод SPICLK может получать внешний тактовый сигнал SPI или обеспечивать тактовый сигнал модуля SPI, соответственно.

В режиме ведомого (Slave) тактовый сигнал модуля SPI принимается на ввод SPICLK от внешнего источника и может быть не более, чем деленная на 8 частота SYSCLK.

В режиме ведущего (Master) тактовый сигнал модуля SPI формируется модулем SPI и выводится на контакт SPICLK.

Определение скорости двоичной передачи

Ниже приведены равенства, позволяющие определить скорость двоичной передачи модуля SPI:

– Для SPIBRR = от 3 до 127:

$$\text{Скорость двоичной передачи модулем SPI} = \frac{\text{SYSCLK}}{(\text{SPIBRR} + 1)}.$$

– Для = 0, 1 или 2:

$$\text{Скорость двоичной передачи модулем SPI} = \frac{\text{SYSCLK}}{4},$$

где: SYSCLK = Частота системного тактового сигнала устройства,

SPIBRR = Содержимое SPIBRR в устройстве ведущего (Master) модуля SPI.

Для определения значения, загруженого в SPIBRR, нужно знать частоту системного тактового сигнала (SYSCLK) (которая определяется устройством или пользователем) и скорость двоичной передачи, на которой будет работать устройство.

На примере 12 показано, как определить максимальную скорость двоичной передачи, на которой ИС 1867ВЦ10Т может передавать информацию. Предполагается, что SYSCLK = 10 МГц.

Пример 12 – Вычисление максимальной скорости двоичной передачи.

$$\text{Скорость передачи} = \frac{\text{SYSCLK}}{(\text{SPIBRR}+1)} = \frac{10\times10^6}{(\text{SPIBRR}+1)} = \frac{10\times10^6}{(3+1)} = \frac{10\times10^6}{4} = 2{,}5\times10^6 = 2{,}5\,\text{Мбит/с}$$

Максимальная скорость двоичной передачи ведущего (Master) модуля SPI будет 2,5 Мбит/с. Однако, скорость двоичной передачи ведомого (Slave) модуля SPI будет SYSCLK/8.

Тактовые схемы модуля SPI

Биты CLOCK POLARITY (SPICCR.6) и CLOCK PHASE (SPICTL.3) управляют четырьмя различными тактовыми схемами на вводе SPICLK. Бит CLOCK POLARITY выбирает активный фронт тактового сигнала, или передний или задний. Бит CLOCK PHASE выбирает задержку тактового сигнала в половину периода. Четыре различные тактовые схемы описаны ниже:

– Задний фронт без задержки. Модуль SPI передает данные на заднем фронте SPICLK и принимает данные на переднем фронте SPICLK.

– Задний фронт с задержкой. Модуль SPI передает данные на полпериода раньше заднего фронта сигнала SPICLK и принимает данные на заднем фронте SPICLK.

– Передний фронт без задержки. Модуль SPI передает данные на переднем фронте SPICLK и принимает данные на заднем фронте SPICLK.

– Передний фронт с задержкой. Модуль SPI передает данные на полпериода раньше переднего фронта сигнала SPICLK и принимает данные на переднем фронте SPICLK.

Процедура выбора тактовой схемы модуля SPI показана в таблице 71. Примеры этих четырёх тактовых схем, относительно переданных и принятых данных, показаны на рисунке 119.

Таблица 71 – Выбор тактовой схемы модуля SPI

Схема SPICLK	CLOCK POLARITY (SPICCR.6)	CLOCK PHASE (SPICTL.3)
Передний фронт без задержки	0	0
Передний фронт с задержкой	0	1
Задний фронт без задержки	1	0
Задний фронт с задержкой	1	1

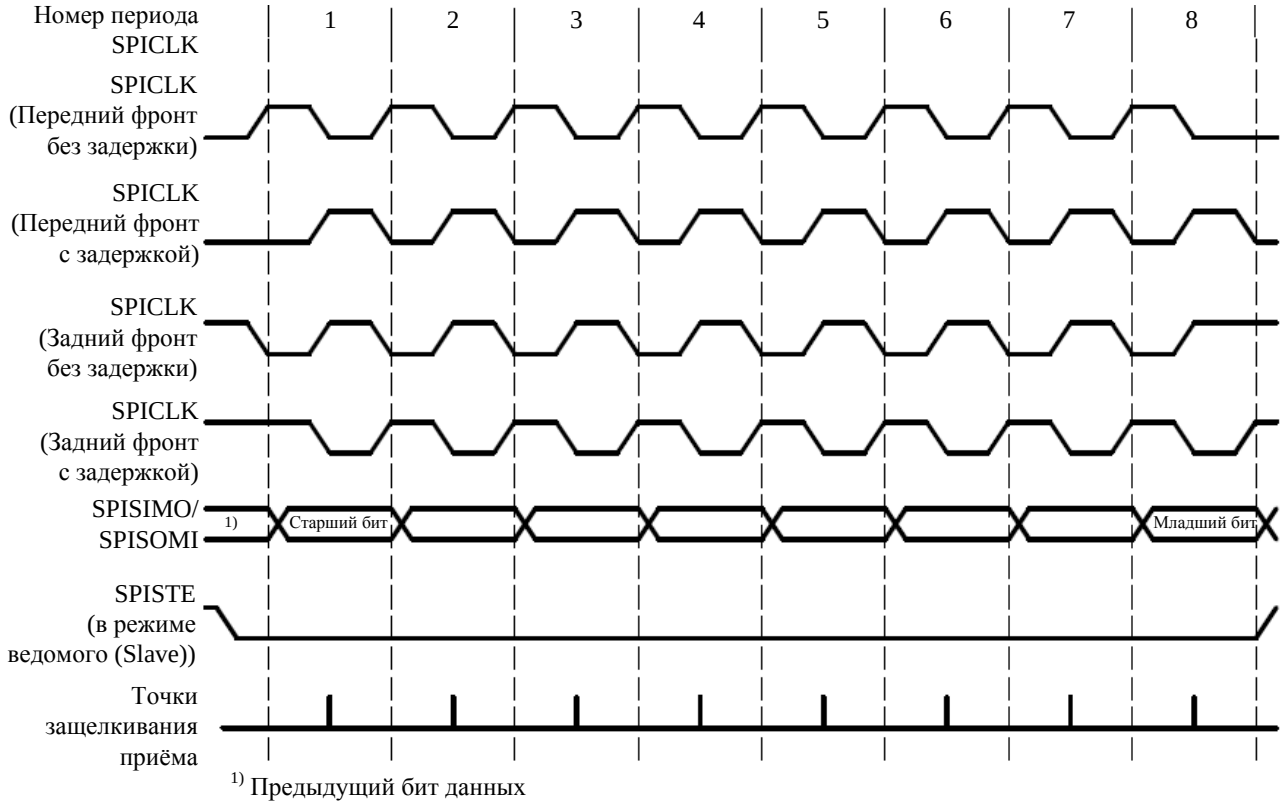


Рисунок 119 - Условия сигнала SPICLK

Для модуля SPI симметрия SPICLK выдерживается только когда результат (SPIBRR+1) является четным значением. Когда (SPIBRR + 1) является нечетным значением и SPIBRR больше чем 3, SPICLK становится асимметричным. Низкий уровень SPICLK на один SYSCLK длиннее, чем высокий уровень, когда бит CLOCK POLARITY очищен (0). Когда бит CLOCK POLARITY установлен в 1, высокий уровень SPICLK на один SYSCLK длиннее, чем низкий уровень, как показано на рисунке 120.

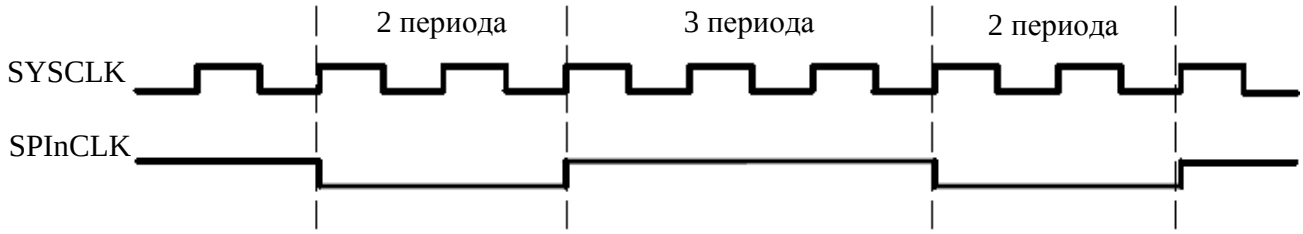


Рисунок 120 – SPI: SPICLK-SYSCLK характеристика, когда значение (BRR + 1) нечетное, BRR > 3 и CLOCK POLARITY = 1

Инициализация во время сброса

Системный сброс принуждает периферийный модуль SPI возвратиться к стандартной конфигурации:

– Блок сконфигурирован как модуль ведомого (Slave) (MASTER/SLAVE = 0).

– Запрещена возможность передачи (TALK = 0).
– Данные защелкиваются на входе при заднем фронте сигнала SPICLK.
– Длина символа принимается за один бит.
– Прерывания модуля SPI запрещены.
– Данные в SPIDAT сбрасываются до 00h.
– Функции вводов выбраны как универсальные входы.

Для изменения этой конфигурации модуля SPI требуется:

1 Установить бит SPI SW RESET (SPICCR.7) в 1.

2 Инициализировать конфигурацию, формат, скорость двоичной передачи и функции вводов модуля SPI как требуется.

3 Очистить бит SPI SW RESET (SPICCR.7).

Примечание – Для предотвращения нежелательных и непредсказуемых событий, возникающих во время или в результате изменений инициализации, нужно устанавливать бит SPI SW RESET (SPICCR.7) перед введением изменений инициализации, а затем очистить этот бит после завершения инициализации.

4 Записать данные в SPIDAT (это инициирует процесс коммуникации в ведущем (Master)).

5 Считать SPIBUF после окончания передачи данных (SPISTS.6 = 1) для определения: какие данные были приняты.

Пример передачи данных

Две временные диаграммы, рисунки 121 и 122, иллюстрируют передачу данных между двумя устройствами, используя длину символа в пять бит с симметричным SPICLK.

Временная диаграмма с несимметричным SPICLK (рисунок 121) имеет одинаковые характеристики, что и на рисунках 121 и 122, за исключением того, что передача данных на один SYSCLK длиннее на каждый бит во время низкого уровня (CLOCK POLARITY = 0) или высокого уровня (CLOCK POLARITY = 1) сигнала SPICLK.

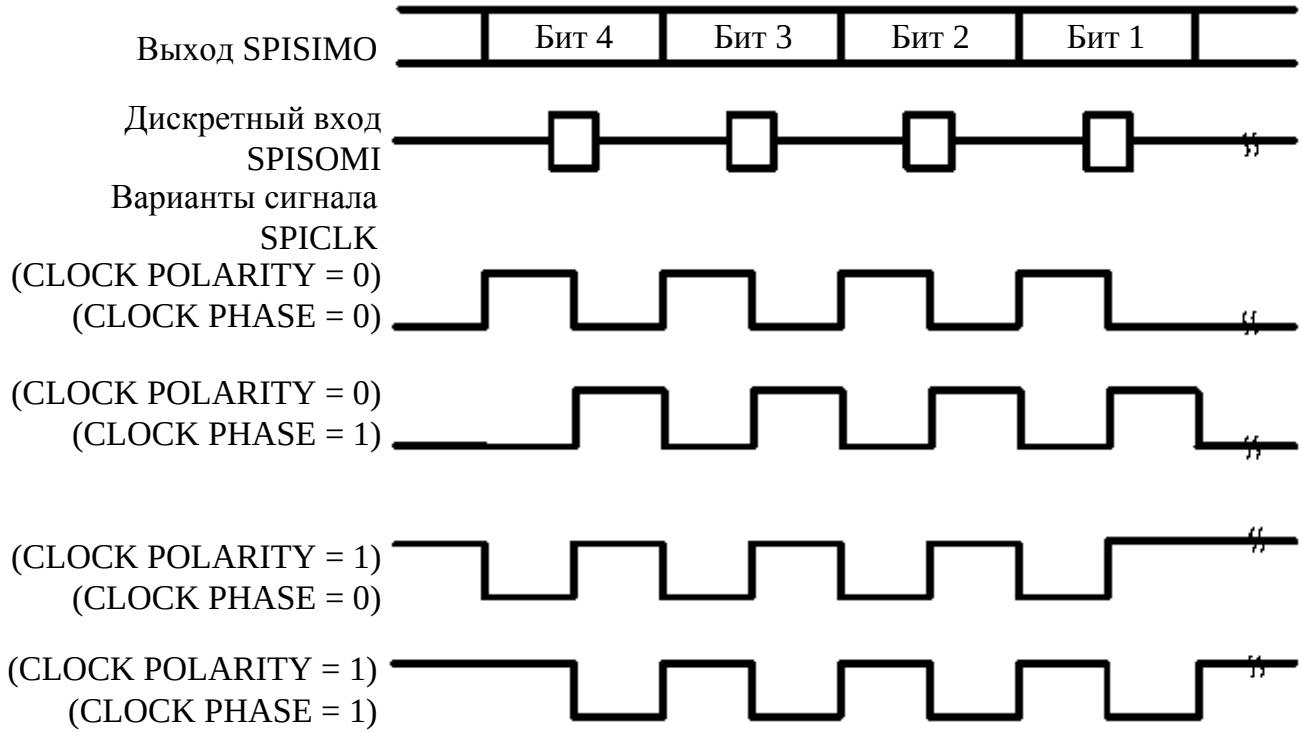


Рисунок 121 – Сигналы к процессору ведущего (Master)

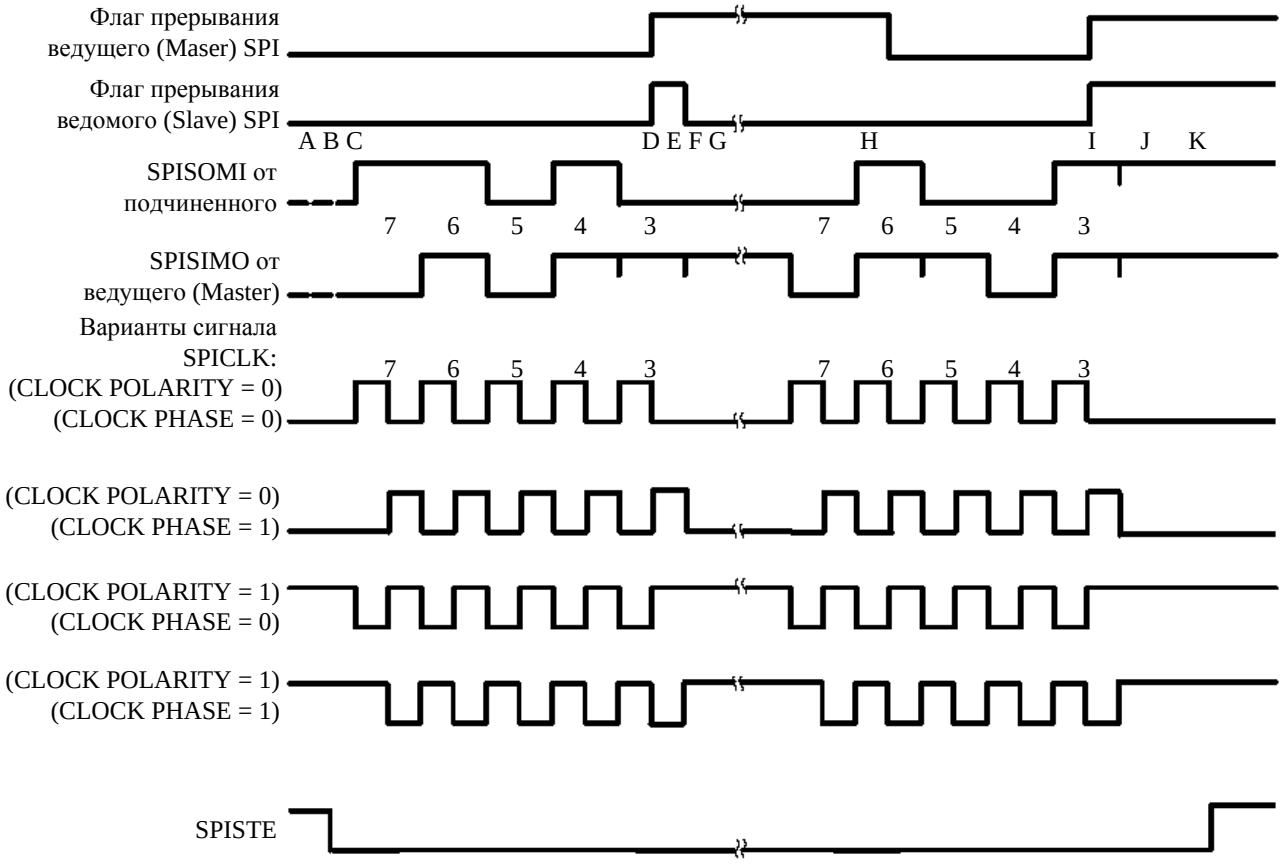


Рисунок 122 – Временная диаграмма передачи данных.

A. Ведомый (Slave) записывает 0D0h в SPIDAT и ждет ведущего (Master) для сдвига данных наружу.

B. Ведущий (Master) устанавливает сигнал ведомого (Slave) SPISTE в низкий (активный) уровень.

C. Ведущий (Master) записывает 058h в SPIDAT, который запускает процедуру передачи.

D. Первый байт завершен и устанавливаются флаги прерывания.

E. Ведомый (Slave) считывает 0Bh из его SPIBUF (выровненные по правому краю).

F. Ведомый (Slave) записывает 04Ch в SPIDAT и ждет ведущего (Master) для сдвига данных наружу.

G. Ведущий (Master) записывает 06Ch в SPIDAT, который запускает процедуру передачи.

H. Ведущий (Master) считывает 01Ah из SPIBUF (выровненные по правому краю).

I. Второй байт завершен и устанавливает флаги прерывания.

J. Ведущий (Master) считывает 89h и ведомый (Slave) считывает 8Dh из их соответствующего SPIBUF. После программного снятия масок с неиспользуемых бит ведущий (Master) принимает 09h, и ведомый (Slave) принимает 0Dh.

K. Ведущий (Master) очищает высокий (пассивный) сигнал ведомого (Slave) SPISTE.

13.5.3 Управляющие регистры модуля SPI

Модуль SPI управляет и получает доступ через регистры в файле управляющего регистра. Эти регистры показаны на рисунке 123 и описываются ниже.

Адрес	Регистр	Номер бита							
		7	6	5	4	3	2	1	0
7040h	SPICCR	SPI SW RESET	CLOCK POLARITY	Зарезервировано			SPI CHAR2	SPI CHAR1	SPI CHAR0
7041h	SPICTL	Зарезервировано			OVERRUN INT ENA	CLOCK PHASE	MASTER/ SLAVE	TALK	SPI INT ENA
7042h	SPISTS	RECEIVER OVERRUN	SPI INT FLAG	Зарезервировано					
7043h	—	Зарезервировано							
7044h	SPIBRR	Зарезервировано	SPI BIT RATE 6	SPI BIT RATE 5	SPI BIT RATE 4	SPI BIT RATE 3	SPI BIT RATE 2	SPI BIT RATE 1	SPI BIT RATE 0
7045h	—	Зарезервировано							
7046h	SPIEMU	ERCVD7	ERCVD6	ERCVD5	ERCVD4	ERCVD3	ERCVD2	ERCVD1	ERCVD0
7047h	SPIBUF	RCVD7	RCVD6	RCVD5	RCVD4	RCVD3	RCVD2	RCVD1	RCVD0
7048h	—	Зарезервировано							
7049h	SPIDAT	SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
704Ah	—	Зарезервировано							
704Bh	—	Зарезервировано							
704Ch	—	Зарезервировано							
704Dh	SPIPC1	SPISTE DATA IN	SPISTE DATA OUT	SPISTE FUNCTION	SPISTE DATA DIR	SPICLK DATA IN	SPICLK DATA OUT	SPICLK FUNCTION	SPICLK DATA DIR
704Eh	SPIPC2	SPISIMO DATA IN	SPISIMO DATA OUT	SPISIMO FUNCTION	SPISIMO DATA DIR	SPISOMI DATA IN	SPISOMI DATA OUT	SPISOMI FUNCTION	SPISOMI DATA DIR
704Fh	SPIPRI	Зарезервировано	SPI PRIORITY	SCI ESPEN	Зарезервировано				

Рисунок 123 – Управляющие регистры модуля SPI

Регистр управления конфигурацией модуля SPI (SPICCR)

SPICCR управляет установкой модуля SPI для работы. Описание бит SCICCR показано на рисунке 124.

7	6	5-3	2	1	0
SPI SW RESET	CLOCK POLARITY	Зарезервировано	SPI CHAR2	SPI CHAR1	SPI CHAR0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 124 – Регистр управления конфигурацией модуля SPI (SPICCR) – адрес 7040h

Бит 7 SPI SW RESET. Программный сброс модуля SPI. При изменении конфигурации нужно установить этот бит перед внесением изменений и очистить этот бит перед возвращением к работе.
1 = Инициализирует флаги работы модуля SPI в состояние сброса.
В частности, бит флага RECEIVER OVERRUN (SPISTS.7) и бит SPI INT FLAG (SPISTS.6) очищаются. Конфигурация модуля SPI остается неизменной. Если модуль работает как ведущий (Master), выход сигнала SPICLK возвращается к своему пассивному уровню.
0 = Модуль SPI готов передавать или принимать следующий символ.
Когда бит SPI SW RESET в 1, символ, записанный в передатчик, не будет сдвинут наружу, когда этот бит очищен. Новый символ должен быть записан в последовательный регистр данных.

Бит 6 CLOCK POLARITY. Изменение полярности тактового сигнала. Этот бит управляет полярностью сигнала SPICLK. CLOCK POLARITY и CLOCK PHASE (SPICTL.3), управляют четырьмя тактовыми схемами на вводе SPICLK.
0 = Пассивный уровень является низким.
Входные и выходные фронты данных зависят от значения бита CLOCK PHASE (SPICTL.3) как показано ниже:
– CLOCK PHASE = 0: Данные выводятся на переднем фронте SPICLK; входные данные защелкиваются на заднем фронте SPICLK.
– CLOCK PHASE = 1: Данные выводятся на полпериода раньше первого переднего фронта SPICLK и последующих задних фронтов сигнала SPICLK; входные данные защелкиваются на переднем фронте сигнала SPICLK.
1 = Пассивный уровень является высоким.
Входные и выходные фронты данных зависят от значения бита CLOCK PHASE (SPICTL.3) как показано ниже:
– CLOCK PHASE = 0: Данные выводятся на заднем фронте SPICLK;

входные данные защелкиваются на переднем фронте SPICLK.
– CLOCK PHASE = 1: Данные выводятся на полпериода раньше первого заднего фронта SPICLK и последующих передних фронтов сигнала SPICLK; входные данные защелкиваются на заднем фронте сигнала SPICLK.

Биты 5-3 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.

Биты 2-0 SPI CHAR2–SPI CHAR0. Управляющие биты длины символа 2-0. Эти три бита определяют количество бит, которые будут сдвинуты внутрь или наружу, как единичный символ во время одной последовательности сдвига. В таблице 72 представлена длина символа, выбираемая значениями бит.

Таблица 72 – Длина символа значения управляющих бит

SCI CHAR2	SCI CHAR1	SCI CHAR0	Длина символа
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

Регистр управления работой модулем SPI (SPICTL)

Регистр управления работой модулем SPI (SPICTL) управляет передачей данных, возможностью модулем SPI формировать прерывания, фазой SPICLK и режимом работы (ведомый (Slave) или ведущий (Master)). Описание бит SPICTL показано на рисунке 125.

7-5	4	3	2	1	0
Зарезервировано	OVERRUN INT ENA	CLOCK PHASE	MASTER/ SLAVE	TALK	SPI INT ENA
	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 125 – Регистр управления работой модулем SPI (SPICTL) – адрес 7041h

Биты 7-5 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.
Бит 4 OVERRUN INT ENA. Разрешение прерывания перегрузки. Установка этого бита (OVERRUN INTERRUPT ENABLE) вызывает прерывание, формируемое, когда бит флага RECEIVER OVERRUN (SPISTS.7) установлен аппаратно. Прерывания, формируемые битом флага

RECEIVER OVERRUN и битом SPI INT FLAG (SPISTS.6), имеют один и тот же вектор прерывания.
0 = Запрещение прерываний бита флага RECEIVER OVERRUN (SPISTS.7).
1 = Разрешение прерываний бита флага RECEIVER OVERRUN (SPISTS.7).

Бит 3 CLOCK PHASE. Выбор фазы тактового сигнала модуля SPI. Этот бит управляет фазой сигнала SPICLK.
0 = Нормальная тактовая схема модуля SPI, зависящая от бита CLOCK POLARITY (SPICCR.6)
1 = Сигнал SPICLK задерживается на полпериода, полярность определяется битом CLOCK POLARITY.
Биты CLOCK PHASE и CLOCK POLARITY делают возможными четыре различные тактовые схемы. Когда работа осуществляется при высоком уровне CLOCK PHASE, модуль SPI (ведущий (Master) или ведомый (Slave)) делает первый бит данных доступным, после записи SPIDAT и перед передним фронтом сигнала SPICLK, независимо от используемого режима модуля SPI.

Бит 2 MASTER/SLAVE. Управление режимом сети модуля SPI. Этот бит определяет, является ли сеть модуля SPI ведущим (Master) или ведомым (Slave). Во время инициализации сброса, модуль SPI автоматически конфигурируется как ведомый (Slave) сети.
0 = Модуль SPI сконфигурирован как ведомый (Slave).
1 = Модуль SPI сконфигурирован как ведущий (Master).

Бит 1 TALK. Разрешение передачи ведущего (Master)/ведомого (Slave). Бит TALK может запретить передачу данных (ведущий (Master) или ведомый (Slave)) помещением последовательного выхода данных в высокоимпедансное состояние. Если этот бит запрещен во время передачи, сдвиговый регистр передачи продолжает работать, пока предыдущий символ не будет сдвинут наружу. Когда бит TALK запрещен, модуль SPI всё еще способен принимать символы, обновлять статусные флаги. TALK очищается (запрещается) системным сбросом.
0 = Запрещает передачу.
Работа в режиме ведомого (Slave): если не был предварительно сконфигурирован как универсальный ввод - вывод, ввод SPISOMI будет установлен в высокоимпедансное состояние.
Работа в режиме ведущего (Master): если не был предварительно сконфигурирован как универсальный ввод - вывод, ввод SPISIMO будет установлен в высокоимпедансное состояние.
1 = Разрешает передачу.
Для условия четырёхконтактности, гарантирует разрешение ввода приёмника SPISTB.

Бит 0 SPI INT ENA. Разрешение прерывания модуля SPI. Этот бит управляет возможностью модулем SPI формировать прерывание. Бит SPI INT FLAG (SPISTS.6) не затрагивается этим битом.

0 = Запрещает прерывание.
1 = Разрешает прерывание.

Статусный регистр модуля SPI (SPISTS)

SPISTS содержит биты буфера приёма. Описание бит SPISTS показано на рисунке 126.

7	6	5-0
RECEIVER OVERRUN	SPI INT FLAG	Зарезервировано
RC-0	R-0	

R – доступ чтения, C – очистка, -0 – значение после сброса

Рисунок 126 – Статусный регистр модуля SPI (SPISTS) – адрес 7042h

Бит 7 RECEIVER OVERRUN. Флаг перегрузки приёмника модуля SPI. Этот бит считывает/очищает только флаг. Модуль SPI устанавливает этот бит аппаратно, когда операция передачи или приёма завершается перед чтением предыдущего принятого символа из буфера. Бит указывает, что последний принятый символ был перезаписан и, таким образом, утерян. Модуль SPI запрашивает одну последовательность прерываний каждый раз, когда этот бит устанавливается, если бит OVERRUN INT ENA (SPICTL.4) установлен в высокое состояние. Этот бит очищается одним из трех путей:

– записью 0 в этот бит;
– записью 1 в SPI SW RESET (SPICCR.7);
– сбросом системы.

Если бит OVERRUN INT ENA (SPICTL.4) установлен, SPI запрашивает одно прерывание каждый раз, когда происходит условие перегрузки. Другими словами, если бит флага RECEIVER OVERRUN остается установленным (не очищен) обслуживающей программой прерывания, другое прерывание перегрузки не будет немедленно введено заново, когда обслуживающая программа прерывания завершится. Прерывание запрашивается каждый раз, когда происходит условие перегрузки, если бит OVERRUN INT ENA разрешен, независимо от предыдущего состояния бита флага RECEIVER OVERRUN.

Однако бит флага RECEIVER OVERRUN должен быть очищен во время выполнения обслуживающей программы прерывания, потому что бит флага RECEIVER OVERRUN и SPI INT FLAG имеют один и тот же вектор прерывания. Это позволяет уменьшить возможную неопределенность при выборе источника прерывания, когда принят следующий байт.

Бит 6 SPI INT FLAG. Флаг прерывания модуля SPI. SPI INT FLAG имеет доступ только для чтения. Этот бит устанавливается аппаратно для указания, что модуль SPI завершил пересылку или приём последнего бита и готов к обслуживанию. Принятый символ помещается в буфер при-

ёмника во время установления этого бита. Этот флаг вызывает запрашивание прерывания, если бит SPI INT ENA (SPICTL.0) установлен. Этот бит очищается одним из трех путей:
– записью 0 в этот бит;
– записью 1 в SPI SW RESET (SPICCR.7);
– сбросом системы.

Биты 5-0 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.

Регистр скорости двоичной передачи модуля SPI (SPIBRR)

SPIBRR содержит биты, используемые для вычисления скорости двоичной передачи. Описание бит SPIBRR показано на рисунке 127.

7	6	5	4	3	2	1	0
Зарезервировано	SPI BIT RATE 6	SPI BIT RATE 5	SPI BIT RATE 4	SPI BIT RATE 3	SPI BIT RATE 2	SPI BIT RATE 1	SPI BIT RATE 0
	RW-0	RW-0	RW-0	RS-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, S – только установка, -0 – значение после сброса

Рисунок 127 – Регистр скорости двоичной передачи модуля SPI (SPIBRR) – адрес 7044h

Бит 7 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.

Бит 6-0 SPI BIT RATE 6 – SPI BIT RATE 0. Управление скоростью двоичной передачи модуля SPI. Эти биты определяют скорость двоичной передачи, если модуль SPI является ведущим (Master) сети. Существует 125 различных скоростей двоичной передачи (каждая является функцией системного тактового сигнала), которые могут быть выбраны. Один бит данных сдвигается за один период SPICLK.
Если модуль SPI является ведомым (Slave) сети, модуль принимает тактовый сигнал на вводе SPICLK от ведущего (Master) сети, а значит, эти биты не влияют на сигнал SPICLK. Частота входного тактового сигнала от ведущего (Master) не должна превышать деленный на восемь сигнал SPICLK ведомого (Slave) модуля SPI.
В режиме ведущего (Master) тактовый сигнал модуля SPI формируется модулем SPI и выводится на контакт SPICLK. Скорость двоичной передачи модуля SPI определяется равенствами, которые приведены ниже:

– Скорость двоичной передачи модуля SPI для SPIBRR = от 3 до 127:

$$\text{Скорость двоичной передачи модуля SPI} = \frac{\text{SYSCLK}}{(\text{SPIBRR} + 1)}.$$

– Скорость двоичной передачи SPI для = 0, 1 или 2:

$$\text{Скорость двоичной передачи модуля SPI} = \frac{\text{SYSCLK}}{4},$$

где: SYSCLK = Частота системного тактового сигнала устройства,
SPIBRR = Содержимое SPIBRR в устройстве ведущего (Master) модуля SPI.

Регистр буфера эмуляции модуля SPI (SPIEMU)

SPIEMU содержит принятые данные. Чтение SPIEMU не очищает бит SPI INT FLAG (SPISTS.6). Описание бит SPIEMU показано на рисунке 128.

7	6	5	4	3	2	1	0
ERCVD7	ERCVD6	ERCVD5	ERCVD4	ERCVD3	ERCVD2	ERCVD1	ERCVD0
R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x

R – доступ чтения, -n – значение после сброса (x = неопределённый)

Рисунок 128 – Регистр буфера эмуляции модуля SPI (SPIEMU) – адрес 7046h

Биты 7-0 ERCVD7–ERCVD0. Полученные данные буфера эмуляции модуля SPI. Функции SPIEMU идентичны функциям SPIBUF, за исключением того, что чтение из SPIEMU не очищает бит SPI INT FLAG (SPISTS.6). Как только SPIDAT получил полный символ, символ передается в SPIEMU и SPIBUF, где он может быть считан. В то же время устанавливается SPI INT FLAG.

Этот зеркальный регистр создан для поддержки эмуляции. Чтение SPIBUF очищает бит SPI INT FLAG (SPISTS.6). При нормальной работе эмулятора управляющие регистры считываются для постоянного обновления содержимого этих регистров на экране дисплея. Эмулятор-ВЦ10Т может считывать регистр SPIEMU и правильно обновлять его содержимое на экране дисплея. Чтение из SPIEMU не очищает бит SPI INT FLAG (SPISTS.6), но чтение SPIBUF очищает этот флаг. Другими словами, SPIEMU разрешает эмулятору имитировать работу модуля SPI более точно.

Рекомендуется считывать SPIBUF в нормальном режиме запуска эмулятора.

Последовательный входной буферный регистр модуля SPI (SPIBUF)

SPIBUF содержит принятые данные. Чтение SPIBUF очищает бит SPI INT FLAG (SPISTS.6). Описание бит SPIBUF показано на рисунке 129.

7	6	5	4	3	2	1	0
RCVD7	RCVD6	RCVD5	RCVD4	RCVD3	RCVD2	RCVD1	RCVD0
R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x

R – доступ чтения, -n – значение после сброса (x = неопределённый)

Рисунок 129 – Последовательный входной буферный регистр модуля SPI (SPIBUF) –адрес 7047h

Биты 7-0 RCVD7–RCVD0. Принятые данные. Как только SPIDAT получил полный символ, символ передается SPIBUF, где он может быть считан. В то же время устанавливается бит SPI INT FLAG (SPISTS.6). Как только данные сдвинуты сначала внутрь старшего разряда модуля SPI, они хранятся в этом регистре с выравниванием по правому краю.

Примечание – Чтение SPIBUF очищает бит SPI INT FLAG (SPISTS.6).

Последовательный регистр данных модуля SPI (SPIDAT)

SPIDAT - это сдвиговый регистр приёма передачи. Данные, записанные в SPIDAT, сдвигаются наружу (старший разряд) в последующие периоды SPICLK. Каждый бит, сдвинутый наружу (старший разряд) модуля SPI, сдвигается внутрь младшего разряда другого конца сдвигового регистра. Описание бит SPIDAT показано на рисунке 130.

7	6	5	4	3	2	1	0
SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
RW-x	RW-x	RW-x	RW-x	RW-x	RW-x	RW-x	RW-x

R – доступ чтения, W – доступ записи, -n – значение после сброса (x = неопределённый)

Рисунок 130 – Последовательный регистр данных модуля SPI (SPIDAT) – адрес 7049h

Биты 7-0 SDAT7–SDAT70. Последовательные данные. Запись в SPIDAT выполняет две функции:

– предоставляет возможность выхода данных на последовательном выводе данных, если установлен бит TALK (SPICTL.1);

– когда модуль SPI работает как ведущий (Master), запускается передача данных. При запуске передаче нужно использовать биты CLOCK POLARITY (SPICCR.6) и CLOCK PHASE (SPICTL.3) для необходимых условий.

Запись пустых данных в SPIDAT инициирует последовательность приёмника. Если данные не аппаратно выровнены для символов короче чем 8 бит, передаваемые данные должны быть записаны в выровненной по левому краю форме, а принимаемые данные в выровненной по правому краю форме.

Управляющий регистр порта модуля SPI 1 (SPIPC1)

SPIPC1 управляет функциями выводов SPISTE и SPICLK. Описание бит SPIPC1 показано на рисунке 131.

7	6	5	4	3	2	1	0
SPISTE DATA IN	SPISTE DATA OUT	SPISTE FUNCTION	SPISTE DATA DIR	SPICLK DATA IN	SPICLK DATA OUT	SPICLK FUNCTION	SPICLK DATA DIR
R-x	RW-0	RW-0	RW-0	R-x	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса (x = неопределённый)

Рисунок 131 – Управляющий регистр порта модуля SPI 1 (SPIPC1) – адрес 704Dh

Бит 7 SPISTE DATA IN. Флаг разрешения передачи входных данных ведомого (Slave). Этот бит содержит текущее значение на вводе SPISTE, независимо от режима. Запись в этот бит не имеет эффекта.

Бит 6 SPISTE DATA OUT. Флаг разрешения передачи выходных данных ведомого (Slave). Этот бит содержит выходные данные на вводе SPISTE (в зависимости от режима работы), если выполнены следующие условия:

– Режим ведущего (Master) (SPICTL.2 = 1):
  – Бит SPISTE DATA DIR (SPIPC1.4) = 1.
  – Бит SPISTE FUNCTION (SPIPC1.5) = x (не определён).
– Режим ведомого (Slave) (SPICTL.2 = 0):

Обычно, ввод SPISTE в режиме ведомого (Slave) выступает как выбор модуля SPI (SPIPC1.5 = 1); когда это происходит, не происходит попыток считывания значений с ввода SPISTE. Однако ввод SPISTE может быть использован как универсальный цифровой ввод - вывод в режиме ведомого (Slave), если выполнены следующие условия:

  – Бит SPISTE DATA DIR (SPIPC1.4) = x (1 = выход; 0 = вход).
  – Бит SPISTE FUNCTION (SPIPC1.5) = 1.

Бит 5 SPISTE FUNCTION. Флаг разрешения выбора функций ввода передачи ведомого (Slave). Этот бит выбирает функции ввода SPISTE (в зависимости от режима работы), если выполнены следующие условия:

– Режим ведущего (Master) (SPICTL.2 = 1):
  – Бит SPISTE FUNCTION не влияет на ввод SPISTE в режиме ведущего (Master).
  – Ввод SPISTE всегда функционирует как универсальный ввод - вывод.
– Режим ведомого (Slave) (SPICTL.2 = 0):
  – Ввод SPISTE функционирует как универсальный цифровой ввод-вывод.
  – Ввод SPISTE функционирует как ввод выбора модуля SPI.

Бит 4 SPISTE DATA DIR. Флаг разрешения направления данных ввода передачи ведомого (Slave). Этот бит определяет направление данных на вводе SPISTE, если бит SPISTE FUNCTION = 0 или если модуль SPI работает в режиме ведущего (Master) (SPICTL.2 = 1).

0 = SPISTE сконфигурирован как ввод.
1 = SPISTE сконфигурирован как вывод.

Бит 3 SPICLK DATA IN. Флаг ввода порта входных данных SPICLK. Этот бит содержит текущее значение на вводе SPICLK, независимо от режима. Запись в этот бит не имеет эффекта.

Бит 2 SPICLK DATA OUT. Флаг вывода порта входных данных SPICLK. Этот бит содержит выходные данные на вводе SPICLK, если выполнены следующие условия:
– Ввод SPICLK определён как универсальный цифровой ввод - вывод (SPICLK FUNCTION = 0).
– Ввод направления данных SPICLK определён как вывод (SPICLK DATA DIR = 1).

Бит 1 SPICLK FUNCTION. Выбор функций ввода SPICLK. Этот бит определяет функции ввода SPICLK.
0 = SPICLK является универсальным цифровым вводом - выводом.
1 = SPICLK содержит тактовый сигнал модуля SPI.

Бит 0 SPICLK DATA DIR. Направление данных SPICLK. Этот бит определяет направление данных на вводе SPICLK, если SPICLK был определен как универсальный цифровой ввод - вывод (SPICLK FUNCTION = 0).
0 = SPISTE является вводом.
1 = SPISTE является выводом.

Управляющий регистр порта модуля SPI 2 (SPIPC2)

SPIPC1 управляет функциями выводов SPISOMI и SPISIMO. Описание бит SPIPC2 показано на рисунке 132.

7	6	5	4	3	2	1	0
SPISIMO DATA IN	SPISIMO DATA OUT	SPISIMO FUNCTION	SPISIMO DATA DIR	SPISOMI DATA IN	SPISOMI DATA OUT	SPISOMI FUNCTION	SPISOMI DATA DIR
R-x	RW-0	RW-0	RW-0	R-x	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса (x = неопределённый)

Рисунок 132 – Управляющий регистр порта модуля SPI 2 (SPIPC2) – адрес 704Eh

Бит 7 SPISIMO DATA IN. Флаг ввода входных данных SPISIMO. Этот бит содержит текущее значение на вводе SPISIMO, независимо от режима. Запись в этот бит не имеет эффекта.

Бит 6 SPISIMO DATA OUT. Флаг вывода выходных данных SPISIMO. Этот бит содержит выходные данные на вводе SPISIMO, если выполнены оба следующих условия:
– Ввод SPISIMO определён как универсальный цифровой ввод - вывод (SPISIMO FUNCTION = 0).
– Ввод направления данных SPISIMO определён как вывод

(SPISIMO DATA DIR = 1).

Бит 5 SPISIMO FUNCTION. Выбор функций ввода SPISIMO. Этот бит определяет функции ввода SPISIMO.
0 = SPISIMO является универсальным цифровым вводом - выводом.
1 = SPISIMO содержит данные модуля SPI.

Бит 4 SPISIMO DATA DIR. Направление данных SPISIMO. Этот бит определяет направление данных на вводе SPISIMO, если этот ввод был определен как универсальный цифровой ввод - вывод (SPISIMO FUNCTION = 0).
0 = SPISIMO является вводом.
1 = SPISIMO является выводом.

Бит 3 SPISOMI DATA IN. Флаг ввода входных данных SPISOMI. Этот бит содержит текущее значение на вводе SPISOMI, независимо от режима. Запись в этот бит не имеет эффекта.

Бит 2 SPISOMI DATA OUT. Флаг вывода выходных данных SPISOMI. Этот бит содержит выходные данные на вводе SPISOMI, если выполнены оба следующих условия:
– Ввод SPISOMI определён как универсальный цифровой ввод - вывод (SPISOMI FUNCTION = 0).
– Ввод направления данных SPISOMI определён как вывод (SPISOMI DATA DIR = 1).

Бит 1 SPISOMI FUNCTION. Выбора функций ввода SPISOMI. Этот бит определяет функции ввода SPISOMI. Когда SPISOMI является входным сигналом и SPISOMI FUNCTION и SPISOMI DATA DIR запрещены, сигнал SPICLK тактирует внутренние цепи.
0 = SPISOMI является универсальным цифровым вводом - выводом.
1 = SPISOMI содержит данные модуля SPI.

Бит 0 SPISOMI DATA DIR. Направление данных SPISOMI. Этот бит определяет направление данных на вводе SPISOMI, если этот ввод был определен как универсальный цифровой ввод - вывод (SPISOMI FUNCTION = 0).
0 = SPISOMI является вводом.
1 = SPISOMI является выводом.

Регистр приоритета модуля SPI (SPIPRI)

SPIPRI выбирает уровень приоритета прерывания модуля SPI и управляет работой модуля SPI при эмуляторе ВЦ10Т во время приостановок программы. Описание бит SPIPRI показано на рисунке 133.

7	6	5	4-0
Зарезервировано	SPI PRIORITY	SCI ESPEN	Зарезервировано
	RW-0	RW-0	

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 133 – Регистр приоритета модуля SPI (SPIPRI) – адрес 704Fh

Бит 7 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.

Бит 6 SPI PRIORITY. Выбор приоритета прерывания. Этот бит определяет уровень приоритета прерывания модуля SPI.
0 = Прерывания запрашиваются с высоким приоритетом.
1 = Прерывания запрашиваются с низким приоритетом.

Бит 5 SPI ESPEN. Разрешение приостановки эмулятора.
0 = Когда эмулятор приостановлен, модуль SPI продолжает операцию пока, не завершится текущая последовательность передачи - приёма.
1 = Когда эмулятор приостановлен, состояние модуль SPI замораживается, таким образом он может быть исследован в точке приостановки эмуляции.

Биты 4-0 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.

13.5.4 Примеры инициализации в режимах работы

В примере 13 показывается программа инициализации модуля SPI в режиме ведомого (Slave), а в примере 14 – программа инициализации модуля SPI в режиме ведущего (Master).

Пример 13 – Программа инициализации модуля SPI в режиме ведомого (Slave).

```
;************************************************************
; Имя файла: SPI.asm пример кода для режима ведомого (Slave)
;
; пример кода #2: Условие четырёх выводов SPI
;                 – режим ведомого (Slave)
;                 – прерывания разрешены
;                 – # переданных байт данных – 7h
;                 – # принятых байт данных – 8h
************************************************************
;Операторы SET для устройств ИС 1867ВЦ10Т зависят от этих
устройств. Адреса SET используемые в этом примере справедливы
для устройств с одним модулем SPI. Для нахождения точных адресов
модулей в ячейках ;памяти нужно обратиться к спецификации
устройств.
;
```

```
.include ”1867wc10treg.h” ; содержит список операторов SET для
всех ;регистров 1867ВЦ10Т.
;Следующие операторы SET для SPI содержатся в 1867wc10treg.h.
; Регистры последовательного периферийного интерфейса (SPI)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SPICCR .set 07040h ; Регистр управления конфигурацией SPI
SPICTL .set 07041h ; Регистр управления работой SPI
SPISTS .set 07042h ; Статусный регистр SPI
SPIBRR .set 07044h ; Регистр скорости двоичной передачи SPI
SPIEMU .set 07046h ; Регистр буфера эмуляции SPI
SPIBUF .set 07047h ; Последовательный входной буферный регистр
SPI
SPIDAT .set 07049h ; Регистр данных SPI
SPIPC1 .set 0704Dh ; Управляющий регистр порта SPI #1
SPIPC2 .set 0704Eh ; Управляющий регистр порта SPI #2
SPIPRI .set 0704Fh ; Регистр приоритета SPI
;-------------------------------------------------------------
; Описание постоянных
;-------------------------------------------------------------
LENGTH .set 08h   ; Длина передаваемого потока данных
                  ;передаваемая/принимаемая SEND_ALL
DECODE .set 01h   ; Декодирование значение используемое для
                  ;разрешения передачи ведомого (Slave).
;-------------------------------------------------------------
; Описание переменных
;-------------------------------------------------------------
; Расположения буфера приёма и передачи указаны ниже.
; Фактическое расположение .bss будет требоваться для описания в
; управляющем файле редактора связей.
       .bss DATAOUT, LENGTH  ;Расположение байта LENGTH
                             ;передаваемого        подпрограммой
SEND_ALL.
       .bss DATAIN, LENGTH   ;Расположение байта LENGTH
                             ;принимаемого         подпрограммой
SEND_ALL.
;-------------------------------------------------------------
; Определение макрокоманд
;-------------------------------------------------------------
KICK_DOG   .macro               ; Макрокоманда сброса сторожевой
схемы
          LDP #00E0h
          SPLK #05555h, WD_KEY
          SPLK #0AAAAh, WD_KEY
          LDP #0h
         .endm
;=============================================================
; Инициализированные данные для подпрограммы SEND_ALL
;=============================================================
       .data
TXDATA    .word 0FDh, 0FBh, 0F7h, 0EFh, 0DFh, 0BFh, 07Fh
;=============================================================
; Векторы прерывания и сброса
```

```
;=========================================================
         .sect ".vectors"
RSVECT   B     START     ; PM 0  Вектор сброса                1
INT1     B     INT1_ISR  ; PM 2  Уровень прерывания 1         4
INT2     B     PHANTOM   ; PM 4  Уровень прерывания 2         5
INT3     B     PHANTOM   ; PM 6  Уровень прерывания 3         6
INT4     B     PHANTOM   ; PM 8  Уровень прерывания 4         7
INT5     B     PHANTOM   ; PM A  Уровень прерывания 5         8
INT6     B     PHANTOM   ; PM C  Уровень прерывания 6         9
RESERVED B     PHANTOM   ; PM E  (Анализ прерывания)          10
SW_INT8  B     PHANTOM   ; PM 10 Программное прерывание       -
SW_INT9  B     PHANTOM   ; PM 12 Программное прерывание       -
SW_INT10 B     PHANTOM   ; PM 14 Программное прерывание       -
SW_INT11 B     PHANTOM   ; PM 16 Программное прерывание       -
SW_INT12 B     PHANTOM   ; PM 18 Программное прерывание       -
SW_INT13 B     PHANTOM   ; PM 1A Программное прерывание       -
SW_INT14 B     PHANTOM   ; PM 1C Программное прерывание       -
SW_INT15 B     PHANTOM   ; PM 1E Программное прерывание       -
SW_INT16 B     PHANTOM   ; PM 20 Программное прерывание       -
TRAP     B     PHANTOM   ; PM 22 Вектор сист. прерывания      -
NMI      B     PHANTOM   ; PM 24 Не маскированное прерывание 3
EMU_TRAP       PHANTOM   ; PM 26 Сист. прерывание эмулятора  2
SW_INT20 B     PHANTOM   ; PM 28 Программное прерывание       -
SW_INT21 B     PHANTOM   ; PM 2A Программное прерывание       -
SW_INT22 B     PHANTOM   ; PM 2C Программное прерывание       -
SW_INT23 B     PHANTOM   ; PM 2E Программное прерывание       -
; Начало инициализации сброса ...
          .text
START:
       CLRC SXM          ;Очистка режима расширения знака
       CLRC OVM          ;Сброс режима переполнения
* Установка указателя страницы данных на страницу 1 периферийного
фрейма
       LDP   #DP_PF1       ;Страница DP_PF1 включает фреймы WET до
EINT
* Инициализация WDT регистров
  SPLK #06Fh, WDTCR ;Очистка WDFLAG, запрещение WDT, установка
                    ;WDT на 1 секунду переполнения (max)
       SPLK #07h, RTICR  ;Очистка флага RTI, установка RTI на 1
                         ;секунду переполнения (max)
* 10 МГц SYSCLK и 20 МГц CPUCLK
       SPLK #00E4h, CKCR1 ;CLKIN(BQ)=4 МГц, CPUCLK=20 МГц
       SPLK #0043h, CKCR0 ;SYSCLK=CPUCLK/2,
       SPLK #00C3h, CKCR0 ;SYSCLK=CPUCLK/2,
* Очистка битов флагов сброса в SYSSR (PORRST,ILLRST,SWRST,WDRST)
      LACL SYSSR         ;ACCL <= SYSSR
      AND  #00FFh        ;Очистка 8 старших бит SYSSR
      SACL SYSSR         ;Загрузка нового значения в SYSSR
* Инициализация ввода IOP20/CLKOUT для использования как выхода
тактового сигнала процессора
      SPLK #40C8h, SYSCR ;Нет сброса, CLKOUT=CPUCLK, ∩VCC2 вклю-
чен
```

```
* Инициализация B2 RAM в нули.
      LAR  AR1, #B2_SADDR;AR1 <= B2 стартовый адрес
      MAR  *, AR1         ;Использование стартового адреса B2 для
                          ;следующего
      ZAC                 ;ACC <= 0
      RPT   #1fh                ;Установка счетчика повтора для
1fh+1=20h
                          ;или 32 циклов
      SACL *+             ;запись нулей в B2 RAM
* Инициализация DATAOUT передаваемыми данными.
      LAR  AR1, #DATAOUT ;AR1 <= DATAOUT начальный адрес
      RPT  #06h           ;Установка счетчика повтора для 6h+1=7h
                          ;или 7 циклов
      BLPD #TXDATA, *+    ;Загрузка 60h – 67h с TXDATA
      CALL INIT_SPI
* Инициализация ЦПОС для прерываний
      LAR  AR6, #IMR      ;
      LAR  AR7, #IFR      ;
      MAR  *, AR6
      LACL #01h           ;
      SACL *, AR7         ;Разрешение только 1 прерываний
      LACL *              ;Очистка IFR чтением и записью содержи-
мого
                          ;обратно в себя
      SACL *, AR2
      CLRC INTM           ;Разрешение прерываний ЦПОС
; Основная программа.
MAIN:                     ;Основная часть кода начинается здесь
                          ;ввести адресные коды
      NOP
      NOP
      NOP
;       ....              ;ввести адресные коды
        B MAIN            ;Основная часть кода завершена в один
                          ;проход, возвращение обратно для
                          ;продолжения.
*************************************************************
* Подпрограммы *
*************************************************************
; Имя программы: INIT_SPI тип программы: SR
;
; Описание: Эта SR инициализирует SPI для передачи потока данных
к ;ведущему (Master) SPI. 1867ВЦ10Т SPI сконфигурирован для 8-
битных ;передач как ведомый (Slave).
;=============================================================
INIT_SPI:
* Инициализация SPI в режиме ведомого (Slave)
     SPLK #008Fh, SPICCR ;Сброс SPI записью 1 to SWRST
     SPLK #0008h, SPICTL ;Запрещение прерываний & TALK, нормаль-
ного
                         ;тактового сигнала, режима ведомого
(Slave)
```

```
     SPLK #000Eh, SPIBRR ;Установка скорости передачи
в ’наивысшую’
; Примечание - Скорость передачи должна быть насколько возможно
;быстрой для связи между двумя и более SPI. Расхождения в выборе
;скорости передачи указывают на различие максимальных скоростей
;ведущего (Master) и ведомого (Slave),  и на меньший порядок
;тактовой скорости каждого устройства. Для контроллеров ЦПОС и
;устройств PRISM это определяется SYSCLK.
;
; Значение ’0Eh’ в SPIBRR гарантирует быстрейшую возможную ско-
рость ;передами для устройств ведущего (Master) и ведомого
(Slave) ;(Полагая что два контроллера ЦПОС осуществляют связь с
одинаковым ;SYSCLK). Это случай, когда ведущий (Master) SPI ис-
пользует ;программу опроса для определения, когда передавать
следующий байт.
     SPLK #0022h, SPIPC1 ;Разрешение функций вводов SPISTE и
;SPICLK. SPISTE будет работать как вход разрешения передачи для
;модуля ведомого (Slave) SPI.
     SPLK #0022h, SPIPC2 ;Установка функций SIMO & SOMI в
;последовательный ввод/вывод
     SPLK #0000h, SPIPRI ;Установка высокого приоритета прерыва-
ния
;SPI. Для целей эмуляции, позволяет SPI продолжать после при-
остановки XDS. Не действует на фактическое устройство.
     SPLK #0000h, SPISTS ;Очистка статусных бит прерывания SPI
     SPLK #0007h, SPICCR ;Освобождение SWRST, clock polarity 0, 8
бит
     SPLK #0009h, SPICTL ;Запрещение TALK & RCV прерывания, фазы
1
;тактового сигнала CLK, режим ведомого (Slave)
* Инициализация вспомогательных регистров для приёма ISR SPI
     LAR  AR1, #LENGTH–1 ;Загрузка длины потока данных в AR1 и
;использование для счетчика цикла приёма/передачи.
     LAR  AR2, #DATAOUT  ;Загрузка расположения передаваемого
;потока данных в AR2.
     LAR  AR3, #DATAIN   ;Загрузка расположения принимаемого по-
тока
;данных в AR3.
RET ;Возврат к основной программе.
*****************************************************************
* ISR *
*****************************************************************
;================================================================
; I S R – INT1 обслуживающая программа прерывания
;
; Описание: Это осуществления метода 3: ISR для единичного собы-
тия ;на каждый уровень прерывания.
;
;  Эта ISR выполняет первичный приём байта DECODE от ведущего
;(Master) SPI. Этот байт проверяется для определения, если веду-
щий ;(Master) запрашивает данные от этого SPI, и если это так,
;устанавливает бит TALK для разрешения передачи и записывает
```

```
байт в ;SPIDAT из DATAOUT. Количество принятых байт управляется
константой ;LENGTH, которая определена ранее.
; Бит TALK очищается после LENGTH # байт было получено и указа-
тели ;вспомогательных регистров перезагружены для подготовки к
следующей ;передаче.
;===============================================================
INT1_ISR                ;Обслуживающая программа прерывания 1
      MAR  *, AR3       ;Использование расположения DATAIN для
;следующих косвенных адресаций
      LACL SPIBUF       ;ACC <= SPI буферный регистр
      SACL *+, AR2      ;хранить значение в B2 и DATAIN,
;использовать адрес DATAOUT для следующих косвенных адресаций
      XOR  #DECODE      ;Сравнение принятого байта для
                        ;определения,  если  выбран  ведомый
(Slave)SPI.
      BCND SKIP, NEQ
      SPLK #000Bh, SPICTL;Разрешение TALK & RCV прерывания, фазы
1 тактового сигнала CLK, режим ведомого
                        ;(Slave)
;SKIP
      LACL *+, AR1      ;ACC <= байт для передачи
                        ;Приращение AR2 на 1 для указания на
;следующий байт в потоке данных. Использование # байт оставлен-
ных
;для TX для следующего косвенного адреса
      SACL SPIDAT       ;Хранение переданного байта в SPIDAT и
;ожидание тактового сигнала ведущего (Master).
      BANZ SKIP2, AR2   ;Ссылка на SKIP2 если AR1 не в нуле,
;уменьшение AR1 на 1, использование адреса DATAOUT для следующе-
го ;адреса
* Повторная инициализация вспомогательных регистров для ISR приёма
SPI
      LAR  AR1, #LENGTH-1;Загрузка длины потока данных в AR1 и
;использование для счетчика цикла приёма/передачи.
      LAR  AR2, #DATAOUT ;Загрузка расположения передаваемого
;потока данных в AR2.
      LAR   AR3, #DATAIN    ;Загрузка расположения принимаемого
потока ;данных в AR3.
* Запрещение связи после LENGTH # передач.
      SPLK #0009h, SPICTL;Запрещение TALK & RCV прерывания, фазы
1
                       ;тактового сигнала CLK, режим ведомого
                       ;(Slave)
SKIP2
      CLRC INTM         ;Разрешение прерываний ЦПОС
      RET               ;Возврат от прерывания
;===============================================================
; I S R - PHANTOM
;
; Описание: Пустая ISR, используется для захвата ложных прерыва-
ний.
;
```

```
;=================================================================
PHANTOM
END   B   END    ;
 .end
```

Пример 14 – Программа инициализации модуля SPI в режиме ведущего (Master)

```
;*****************************************************************
; Имя файла: пример кода для режима ведущего (Master)
; пример кода #1: Условие четырёх выводов SPI
;                 – режим ведущего (Master)
;                 – прерывания опрошены
;                 – # переданных байт данных – 8h
;                 – # принятых байт данных – 8h
;
*****************************************************************
;Операторы SET для устройств 1867ВЦ10Т зависят от этих устройств.
Адреса ;SET используемые в этом примере справедливы для
устройств с одним ;модулем SPI. Для нахождения точных адресов
модулей в ячейках
;памяти нужно обратиться к спецификации устройств.
.include ”1867wc10treg.h” ; содержит список операторов SET для
всех ;регистров 1867ВЦ10Т.
;Следующие операторы SET для SPI содержатся в 1867wc10treg.h.
;Регистры последовательного периферийного интерфейса (SPI)
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SPICCR  .set 07040h ; Регистр управления конфигурацией SPI
SPICTL  .set 07041h ; Регистр управления работой SPI
SPISTS  .set 07042h ; Статусный регистр SPI
SPIBRR  .set 07044h ; Регистр скорости двоичной передачи SPI
SPIEMU  .set 07046h ; Регистр буфера эмуляции SPI
SPIBUF   .set 07047h ; Последовательный входной буферный регистр
SPI
SPIDAT  .set 07049h ; Регистр данных SPI
SPIPC1  .set 0704Dh ; Управляющий регистр порта SPI #1
SPIPC2  .set 0704Eh ; Управляющий регистр порта SPI #2
SPIPRI  .set 0704Fh ; Регистр приоритета SPI
;-----------------------------------------------------------
; Описание постоянных
;-----------------------------------------------------------
LENGTH  .set 08h    ; Длина передаваемого потока данных
                    ; передаваемая/принимаемая SEND_ALL
;-----------------------------------------------------------
; Описание переменных
;-----------------------------------------------------------
; Расположения буфера приёма и передачи указаны ниже.
; Фактическое расположение .bss будет требоваться для описания в
; управляющем файле редактора связей.
;
    .bss DATAOUT, LENGTH ;Расположение байта LENGTH
                         ;передаваемого подпрограммой
```

```
                        ;SEND_ALL.
    .bss DATAIN, LENGTH  ;Расположение байта LENGTH
                        ;принимаемого подпрограммой SEND_ALL.
;-------------------------------------------------------------
SPI_DONE .set 070h              ;Определяет B2 RAM расположение
'70h'
                              ;как расположение статуса передачи
                              ;SPI. 1=полный, 0=не полный
;-------------------------------------------------------------
; Определение макрокоманд
;-------------------------------------------------------------
KICK_DOG .macro                 ;Макрокоманда сброса сторожевой
схемы
         LDP #00E0h
         SPLK #05555h, WD_KEY
         SPLK #0AAAAh, WD_KEY
         LDP #0h
         .endm
;=============================================================
; Инициализированные данные для подпрограммы SEND_ALL
;=============================================================
.data
TXDATA   .word 01h, 02h, 04h, 08h, 10h, 20h, 40h, 80h
;=============================================================
; Векторы прерывания и сброса
;=============================================================
          .sect ".vectors"
RSVECT     B     START     ; PM 0  Вектор сброса              1
INT1       B     INT1_ISR  ; PM 2  Уровень прерывания 1       4
INT2       B     PHANTOM   ; PM 4  Уровень прерывания 2       5
INT3       B     PHANTOM   ; PM 6  Уровень прерывания 3       6
INT4       B     PHANTOM   ; PM 8  Уровень прерывания 4       7
INT5       B     PHANTOM   ; PM A  Уровень прерывания 5       8
INT6       B     PHANTOM   ; PM C  Уровень прерывания 6       9
RESERVED     B        PHANTOM     ;  PM  E   (Анализ  прерывания)
10
SW_INT8    B     PHANTOM   ; PM 10 Программное прерывание     -
SW_INT9    B     PHANTOM   ; PM 12 Программное прерывание     -
SW_INT10   B     PHANTOM   ; PM 14 Программное прерывание     -
SW_INT11   B     PHANTOM   ; PM 16 Программное прерывание     -
SW_INT12   B     PHANTOM   ; PM 18 Программное прерывание     -
SW_INT13   B     PHANTOM   ; PM 1A Программное прерывание     -
SW_INT14   B     PHANTOM   ; PM 1C Программное прерывание     -
SW_INT15   B     PHANTOM   ; PM 1E Программное прерывание     -
SW_INT16   B     PHANTOM   ; PM 20 Программное прерывание     -
TRAP       B     PHANTOM   ; PM 22 Вектор сист. прерывания    -
NMI        B     PHANTOM   ; PM 24 Не маскированное прерывание 3
EMU_TRAP         PHANTOM   ; PM 26 Сист. прерывание эмулятора  2
SW_INT20   B     PHANTOM   ; PM 28 Программное прерывание     -
SW_INT21   B     PHANTOM   ; PM 2A Программное прерывание     -
SW_INT22   B     PHANTOM   ; PM 2C Программное прерывание     -
SW_INT23   B     PHANTOM   ; PM 2E Программное прерывание     -
```

```
; Начало инициализации сброса ...
          .text
START:
      CLRC SXM          ;Очистка режима расширения знака
      CLRC OVM          ;Сброс режима переполнения
*Установка указателя страницы данных на страницу 1 периферийного
фрейма
      LDP   #DP_PF1       ;Страница DP_PF1 включает фреймы WET до
EINT
*Инициализация WDT регистров
      SPLK #06Fh, WDTCR ;Очистка WDFLAG, запрещение WDT, уста-
новка
;WDT на 1 секунду переполнения (max)
      SPLK #07h, RTICR  ;Очистка флага RTI, установка RTI на 1
;секунду переполнения (max)
* 10 МГц SYSCLK и 20 МГц CPUCLK
      SPLK #00E4h, CKCR1;CLKIN(BQ)=4 МГц, CPUCLK=20 МГц
      SPLK #0043h, CKCR0; SYSCLK=CPUCLK/2,
      SPLK #00C3h, CKCR0; SYSCLK=CPUCLK/2,
*Очистка битов флагов сброса в SYSSR (PORRST,ILLRST,SWRST,WDRST)
      LACL SYSSR        ;ACCL <= SYSSR
      AND  #00FFh       ;Очистка 8 старших бит SYSSR
      SACL SYSSR        ;Загрузка нового значения в SYSSR
*Инициализация ввода IOP20/CLKOUT для использования как выход
тактового сигнала процессора
      SPLK #40C8h, SYSCR;Нет сброса, CLKOUT=CPUCLK, ∩VCC2 вклю-
чен
*Инициализация B2 RAM в нули.
      LAR  AR1, #B2_SADDR; AR1 <= B2 стартовый адрес
      MAR  *, AR1       ;Использование стартового адреса B2 для
;следующего
      ZAC               ;ACC <= 0
      RPT   #1fh              ;Установка счетчика повтора для
1fh+1=20h
;или 32 циклов
      SACL *+           ;запись нулей в B2 RAM
* Инициализация DATAOUT передаваемыми данными.
      LAR  AR1, #DATAOUT ;AR1 <= DATAOUT начальный адрес
      RPT  #06h         ;Установка счетчика повтора для 7h+1=8h
;или 8 циклов
      BLPD #TXDATA, *+  ;Загрузка 60h – 68h с 01, 02, 04, ... ,
;40, 80h
      CALL INIT_SPI
;Основная программа. Когда бы поток данных, ранее загруженный в
;расположение DATAOUT, не требовался для передачи, вызывается
;подпрограмма SEND_ALL.
MAIN:                   ;Основная часть кода начинается здесь
                        ;ввести адресные коды
      CALL SEND_ALL     ;Вызов подпрограммы SEND_ALL и когда
;она завершится продолжить с основных циклом.
;     ....              ;ввести адресные коды
      B    MAIN         ;Основная часть кода завершена в один
```

```
;проход, возвращение обратно для продолжения.
*****************************************************************
* Подпрограммы *
*****************************************************************
;===============================================================
; Имя программы: INIT_SPI тип программы: SR
;
; Описание: Эта SR инициализирует SPI для передачи потока данных
к ;подчиненному SPI. 1867ВЦ10Т SPI сконфигурирован для 8-битных
;передач как ведущий (Master).
;===============================================================
INIT_SPI:
* Инициализация SPI в режиме ведущего (Master)
      SPLK #008Fh, SPICCR ;Сброс SPI записью 1 to SWRST
      SPLK #0008h, SPICTL ;Запрещение прерываний & TALK,
;нормального тактового сигнала, режима ведомого (Slave)
      SPLK  #000Eh,  SPIBRR  ;Установка  скорости  передачи  в
'наивысшую'
;Примечание – Скорости передачи должна быть насколько возможно
;быстрой для связи между двумя и более SPI. Расхождения в выборе
;скорости передачи указывают на различие максимальных скоростей
;ведущего (Master) и ведомого (Slave),  и на меньший порядок
;тактовой скорости каждого устройства. Для контроллеров ЦПОС и
;устройств PRISM это определяется SYSCLK.
;
;Значение '0Eh' в SPIBRR гарантирует быстрейшую возможную ско-
рость ;передами для устройств ведущего (Master) и ведомого
(Slave) ;(Полагая что два контроллера ЦПОС осуществляют связь с
одинаковым ;SYSCLK). Это случай, когда ведущий (Master) SPI ис-
пользует ;программу опроса для определения, когда передавать
следующий байт.
      SPLK  #0052h,  SPIPC1  ;Разрешение  функций  ввода  SPICLK.
SPISTE
;всегда будет универсальным вводом/выводом, когда SPI находится
в ;режиме ведущего (Master), независимо от состояния бита функ-
ции. ;Установка SPISTE как высокий уровень выхода – запрещает
выход ;приёмника SPI.
      SPLK #0022h, SPIPC2 ;Установка функций SIMO & SOMI в
;последовательный ввод/вывод
      SPLK #0000h, SPIPRI ;Установка высокого приоритета преры-
вания
;SPI. Для целей эмуляции, позволяет SPI продолжать после
;приостановки XDS. Не действует на фактическое устройство.
      SPLK #0000h, SPISTS ;Очистка статусных бит прерывания SPI
      SPLK  #0007h,  SPICCR  ;Освобождение  SWRST,  clock  polarity
0,8 бит
      SPLK #0009h, SPICTL ;Разрешение TALK & RCV прерывания, фа-
зы   ;фазы 1 тактового сигнала CLK, режим ведущего (Master)
      RET                 ;Возврат к основной программе.
;===============================================================
; Имя программы: SEND_ALL тип программы: SR
;
```

```
; Описание: Эта SR выполняет передачу потока данных. Передаваемые ;данные содержатся в DATAOUT.Полученные данные хранятся в DATAIN. ;Эта программа опрашивает бит SPI INT FLAG, SPISTS.6, для ;определения, когда завершилась передача каждого байта. Количество ;передаваемых байт управляется константой LENGTH, которая ;определена ранее.
;=================================================================
SEND_ALL:
      LAR  AR1, #LENGTH-1 ;Загрузка длины потока данных в AR1 и ;использование для счетчика цикла приёма/передачи.
      LAR  AR2, #DATAOUT  ;Загрузка расположения передаваемого ;потока данных в AR2.
      LAR   AR3, #DATAIN    ;Загрузка расположения принимаемого потока ;данных в AR3.
      MAR  *, AR2         ;Использование расположения DATOUT для ;следующего косвенного адреса
                          ;Выполнение чтения-изменения-записи на ;SPIPC1 для установки ввода SPISTE в активный низкий уровень и ;разрешения ведомого (Slave) SPI.
      LACL SPIPC1         ;Загрузка содержимого SPIPC1 в ACC.
      AND  #0BFh           ;Очистка SPIPC1.6 для установки ввода SPISTE ;в активный низкий уровень.
      SACL SPIPC1         ;Хранение ACC вне SPIPC1.
LOOP
                          ;Запуск передачи записью байта в SPIDAT.
      LACL *+, AR3        ;ACC <= байт для передачи
                          ;Приращение AR2 на 1 для указания на ;следующий байт в потоке данных. Использование DATAIN для следующего ;косвенного адреса
      SACL SPIDAT         ;Передача байта
POLL  LACL SPISTS         ;Опрос бита флага INT для определения, ;когда начинать следующую передачу
      AND  #040h          ;Очистка всех бит кроме SPI INT FLAG
      XOR  #040h          ;ACC=0 если бит установлен
      BCND POLL, NEQ      ;продолжить опрос если ACC != 0.
      LACL SPIBUF         ;Загрузит полученный байт в ACC.
      SACL *+, AR1        ;Сохранить полученный байт в DATAIN
                          ;Приращение AR3 на один указатель до ;следующего расположения DATAIN. Использование AR1, # байт, ;оставленных для передачи, для следующего косвенного адреса
      BANZ LOOP, AR2      ;Ссылка на LOOP если AR1 не в нуле, ;уменьшение AR1 на 1, использование адреса DATAOUT для следующего ;адреса
                          ;Секция дополнительного кода: Этот код ;загружает значение в B2 RAM для информирования основной программы, ;что передача потока данных завершена.
      LAR  AR4, #SPI_DONE;Загрузка адреса размещения статуса SPI вAR4
      MAR  *, AR4         ;Использование размещения статуса SPI для
;следующего косвенного адреса
```

```
      SPLK #01h, *       ;Запись '01h' в размещения статуса для
;определения, что передача потока данных завершена.
                         ;Выполнение чтения-изменения-записи на
SPIPC1 для установки ввода SPISTE в активный высокий уровень и
;запрещения ведомого (Slave)SPI.
      LACL SPIPC1        ;Загрузка содержимого SPIPC1 в ACC.
      OR   #040h         ;Установка SPIPC1.6 для установки ввода
;SPISTE в активный высокий уровень
      SACL SPIPC1        ;Хранение ACC вне SPIPC1.
      RET                ;Возврат к основной программе.
*****************************************************************
* ISR *
*****************************************************************
;================================================================
; I S R - PHANTOM
; Описание: Пустая ISR, используется для захвата ложных прерыва-
ний.
;
;================================================================
PHANTOM
END   B    END          ;
 .end
```

13.6 Модуль сторожевого таймера Watchdog (WD) и блока прерываний реального времени Real-Time Interrupt (RTI)

Модуль сторожевого таймера (WD) и блока прерываний реального времени (RTI) контролирует аппаратную и программную работу процессора, обеспечивает прерывания на программируемых интервалах и осуществляет функции системного сброса при процессорном сбое. Если программа осуществляет неразрешенный цикл, или процессор становится временно недоступным, сторожевой таймер переполняется для выполнения системного сброса.

Большинство условий, которые временно нарушают работу кристалла и блокируют правильную работу процессора, могут быть очищены и сброшены функцией сторожевого таймера. Сторожевой таймер повышает надежность системы, обеспечивая непрерывную работу процессора, тем самым гарантируя целостность системы.

Примечание – Этот модуль соединён с 16-битной периферийной шиной как 8-битная периферия, поэтому при чтении битов 15 – 8 их значение не определено; запись в эти биты не имеет эффекта.

13.6.1 Обзор модуля сторожевого таймера WD и блока прерываний реального времени RTI

Характеристики

Модуль WD/RTI имеет следующие характеристики:

Сторожевой таймер WD:

– 8-битный счетчик модуля WD, который формирует системный сброс при переполнении.

– 7-битный автономный счетчик, который подает на контакт счетчика модуля WD делитель счетчика модуля WD.

– Регистр ключа сброса модуля WD (WDKEY), который очищает счетчик модуля WD, когда записана корректная комбинация значений, и формирует сброс, если в регистр записано не корректное значение.

Бит флага модуля WD (WD FLAG), который указывает, будет ли сторожевой таймер инициировать системный сброс.

Биты проверки модуля WD, которые инициируют системный сброс, если сторожевой таймер повреждён.

Автоматическая активация сторожевого таймера при освобождении системного сброса.

Делитель модуля WD с выбором из шести значений из 7-битного автономного счетчика и два (одинаковых) входа сигнала WDCLK.

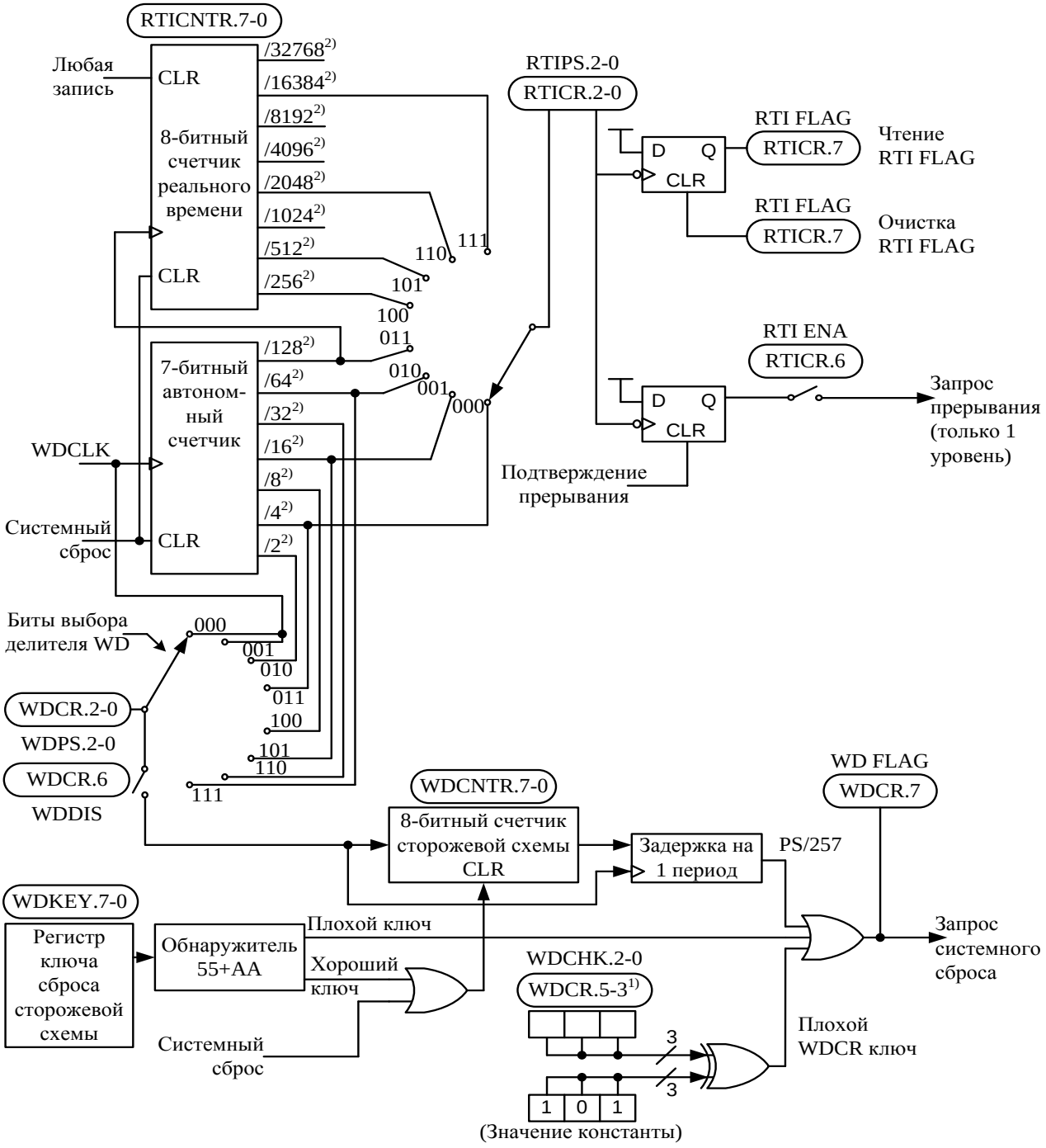
Таймер прерываний реального времени RTI:

– Делитель блока RTI, который выбирается из четырёх отводов 8-битного счетчика реального времени и четырёх отводов 7-битного автономного счетчика.

– Операция прерывания или опроса (программный бит разрешает/ запрещает прерывания блока RTI).

– Бит флага блока RTI (RTI FLAG), который указывает, будет ли счетчик блока RTI (RTICNTR) переполнен.

На рисунке 134 представлена структурная схема модуля WD/RTI.



[1] Запись в биты WDCR.5-3 в конфигурации, отличающейся от нормальной (101) формирует системный сброс.
[2] Эти значения делителя взяты относительно сигнала WDCLK.

Рисунок 134 – Структурная схема модуля сторожевого таймера (WD) и блока прерываний реального времени (RTI)

Управляющие регистры

Пять регистров управляют операциями модуля WD/RTI:

– Регистр счетчика блока RTI (RTICNTR) содержит значение счетчика блока RTI.

– Регистр счетчика модуля WD (WDCNTR) содержит значение счетчика модуля WD.

– Регистр ключа сброса модуля WD (WDKEY) очищает WDCNTR, когда значение 55h, следующее за значением AAh, записывается в WDKEY.

– Управляющий регистр блока RTI (RTICR) содержит следующие управляющие биты, используемые для конфигурирования модуля RTI:

– бит флага блока RTI;

– бит разрешения блока RTI;

– биты выбора делителя блока RTI.

– Управляющий регистр модуля WD (WDCR) содержит следующие управляющие биты, используемые для конфигурирования WD:

– бит флага модуля WD;

– биты проверки модуля WD;

– биты выбора делителя модуля WD.

В таблице 73 представлены адреса регистров модуля WD/RTI.

Таблица 73 – Адреса регистров модуля WD/RTI

Адрес	Регистр	Имя
7020h	—	Зарезервировано
7021h	RTICNTR	Регистр счетчика блока RTI
7022h	—	Зарезервировано
7023h	WDCNTR	Регистр счетчика модуля WD
7024h	—	Зарезервировано
7025h	WDKEY	Регистр ключа сброса модуля WD
7026h	—	Зарезервировано
7027h	RTICR	Управляющий регистр блока RTI
7028h	—	Зарезервировано
7029h	WDCR	Управляющий регистр модуля WD
702Ah	—	Зарезервировано
702Bh	—	Зарезервировано[1)]
702Ch	—	Зарезервировано
702Dh	—	Зарезервировано[1)]
702Eh	—	Зарезервировано
702Fh	—	Зарезервировано

[1)] Зарезервировано для управляющих регистров модуля PLL тактового сигнала.

13.6.2 Функционирование таймера сторожевой схемы WD и блока прерываний реального времени RTI

Таймер модуля WD

Таймер модуля WD является 8-битным возрастающим счетчиком со сбросом, который тактируется выходом делителя. Таймер защищает от системных программных сбоев и сбоев процессора обеспечением системного сброса, когда WDKEY не использован перед переполнением сторожевой схемы. Этот сброс возвращает систему к известному начальному значению. Затем программа очищает WDCNTR записью корректных данных в логику ключа модуля WD.

Разделённый внутренний тактирующий сигнал (WDCLK) формируется модулем тактовой частоты и является активным во всех режимах работы, за исключением режима пониженного потребления генератора. WDCLK разрешает функционирование таймера модуля WD, независимо от состояния каких-либо бит(а) регистра на кристалле, за исключением работы в режиме пониженного потребления генератора, которые запрещают сигнал WDCLK. Обычно частота WDCLK равна 16384 Гц. Текущее состояние WDCNTR может быть считано в любое время его работы.

Обычная частота WDCLK в 16384 Гц получается от двух тактовых частот (например 4,194 МГц, 8,389 МГц).

Выбор делителя модуля WD

8-битный WDCNTR может тактироваться непосредственно сигналом WDCLK или через один из шести отводов автономного счетчика. 7-битный автономный счётчик постоянно увеличивается через интервал, обеспечиваемый WDCLK (обычно 16384 Гц или 32768 Гц, в зависимости от специфики устройств). Функции модуля WD разрешены, пока WDCLK поступает в модуль. Любой из первых шести отводов или непосредственный вход от WDCLK могут быть выбраны делителем WD (биты WDPC2-0), как вход для оси времени WDCNTR. Этот делитель обеспечивает выбираемые коэффициенты переполнения сторожевой схемы от 15,63 мс до 1 с для WDCLK в 16384 Гц. Пока кристалл находится в нормальном режиме работы, автономный счетчик не может быть остановлен или сброшен, за исключением возникновения системного сброса. Очистка или RTICNTR или WDCNTR не очищает автономный счетчик.

Обслуживание таймера модуля WD

WDCNTR сбрасывается, когда правильная последовательность записана в WDKEY перед переполнением WDCNTR. WDCNTR разрешен для сброса, когда значение 55h записывается в WDKEY. Когда следующее AAh значение записывается в WDKEY, WDCNTR сбрасывается. Любое другое значение, записанное в WDKEYи отличающееся от 55h или AAh, вызывает системный сброс. Любая последовательность значений 55h и AAh может быть записана в WDKEY без вызова системного сброса; только запись 55h следующей за записью AAh в WDKEY сбрасывает WDCNTR.

В таблице 74 показана типичная последовательность, записанная в WDKEY после включения питания.

Таблица 74 – Типичная последовательность, записанная в регистр WDKEY после включения питания

Последовательный шаг	Значение, записанное в WDKEY	Результат
1	AAh	Нет действия

Последовательный шаг	Значение, записанное в WDKEY	Результат
2	AAh	Нет действия
3	55h	WDCNTR разрешен для сброса следующим AAh
4	55h	WDCNTR разрешен для сброса следующим AAh
5	55h	WDCNTR разрешен для сброса следующим AAh
6	AAh	WDCNTR сброшен
7	AAh	Нет действия
8	55h	WDCNTR разрешен для сброса следующим AAh
9	AAh	WDCNTR сброшен
10	55h	WDCNTR разрешен для сброса следующим AAh
11	23h	Система сброшена из-за неправильного значения, записанного в WDKEY

Шаг 3 в таблице 74 является первым действием для разрешения сброса WDCNTR. Фактически WDCNTR не сбрасывается до шага 6. Шаг 8 повторно разрешает сброс WDCNTR и шаг 9 сбрасывает WDCNTR. Шаг 10 опять повторно разрешает сброс WDCNTR. Запись неверного значения ключа в WDKEY в шаге 11 вызывает системный сброс.

Переполнение WDCNTR или неверное значения ключа, записанное в WDKEY устанавливают флаг модуля WD (WDFLAG). После сброса программа считывает этот флаг для определения источника сброса. После сброса WDFLAG должен быть программно очищен для разрешения определения источника последовательных сбросов модуля WD. Сбросы модуля WD не предотвращены, когда установлен флаг.

Сброс модуля WD

Когда WDCNTR переполняется, таймер модуля WD формирует системный сброс. Сброс возникает на один WDCNTR тактовый период (любой из WDCLK или WDCLK, разделенные значением делителя) позже. Сброс не может быть запрещен при нормальной работе, пока действует WDCLK. Таймер модуля WD, однако, запрещен в режиме пониженного потребления генератора, когда WDCLK не активен.

Запрещение модуля WD

Для целей разработки таймер модуля WD может быть запрещен подачей 3,3 В на ввод WDDIS во время последовательности сброса устройства и установкой бита WDDIS в управляющем регистре модуля WD (WDCR.6). Однако, если эти аппаратные и программные условия не выполняются, то таймер модуля WD не будет запрещен.

Логика проверки битов сброса модуля WD

Биты проверки модуля WD (WDCR.5-3) постоянно сравниваются со значением постоянной (101_2). Если биты проверки модуля WD не совпадают с этим значением, формируется системный сброс. Это функционирует как логическая проверка, в случае если программа сделала неверную запись в WDCR, или если внешние воздействия (такие как броски напряжения, электромагнитные помехи или другие разрушающие источники) повреждают содержимое WDCR. Запись в биты WDCR.5 – 3 любого значения, кроме правильной конфигурации (101_2), формирует системный сброс.

Примечание – Любые значения, записанные в WDCR, должны включать значение 101_2, записанное в биты 5 – 3 (WDCHK2 – WDCHK0).

Установка модуля WD

Таймер модуля WD работает независимо от процессора и всегда разрешен. Он не требует для работы какой-либо инициализации от процессора. Когда происходит системный сброс, таймер модуля WD возвращается к тактированию на самой большой частоте таймера модуля WD (15,63 мс для 16384 Гц сигнала WDCLK). Как только сброс прекращается, то процессор начинает выполнять программный код и таймер модуля WD начинает увеличиваться. Это означает, что для предотвращения преждевременного сброса, установку модуля WD/RTI необходимо произвести раньше, чем произойдет сброс или последовательность запускающая сброс.

Таймер блока RTI

Таймер блока RTI является 8-битным счетчиком, который может быть запрограммирован на формирование периодических прерываний на программно выбираемой частоте. Восемь отводов четыре от счетчика блока RTI (RTICNTR) и четыре от автономного счетчика) могут быть выбраны через мультиплексор 8 в 1 для формирования частоты периодических запросов прерываний. Прерывания разрешаются и запрещаются программно. RTICNTR может быть считан или очищен в любой момент времени. 7-битный автономный счетчик, однако, не может быть очищен, за исключением очищения системным сбросом.

Расположение вектора прерывания таймера блока RTI приведено в кратком описании ИС 1867ВЦ10Т.

Выбор делителя блока RTI

RTICNTR использует отвод автономного счетчика с делением на 128 как вход для формирования четырёх выходных отводов. Эти четыре отвода и четыре дополнительных отвода, получаемые от автономного счетчика, могут быть выбраны через мультиплексор 8 в 1, который управляется тремя битами выбора делителя управляющего регистра блока RTI (RTICR.2-0). Этот делитель обеспечивает выбираемые значения прерываний от 1 до 4096 прерываний в секунду (для WDCLK в 16384 Гц). RTICNTR тактируется сигналом WDCLK. RTICNTR может быть сбро-

шен до 00h в любой момент времени в течение нормальной работы. Работа таймера блока RTI разрешена во всех режимах, кроме режима останова.

Программная очистка RTICNTR не очищает автономный счетчик, который обеспечивает деление RTICNTR. Поэтому, после очистки RTICNTR, измерения синхронизации этого счетчика вступают в 1-битную неопределенность.

Разрешение блока RTI

Блок RTI может быть разрешен или запрещен битом разрешения блока RTI (RTI ENA), который является 6 битом управляющего регистра блока RTI (RTICR.6). Когда блок RTI разрешен, он позволяет формироваться прерыванию, когда происходит выбранное переполнение. Логика прерывания формирует только одно прерывание для каждого переключения на выбранном отводе.

Из-за того, что автономный счетчик не может быть очищен программно, программная логика полагает, что указанный период времени вычисляется между последовательными переполнениями RTICNTR. Поэтому синхронизация первого прерывания не может быть спрогнозирована. Ожидание, пока RTI FLAG (RTICR.7) подтвердит приём первого переполнения перед разрешением прерывания, обеспечивает правильную синхронизацию.

Флаг блока RTI

Бит флага блока RTI (RTIFLAG) указывает на то, что произошло переполнение. Это седьмой бит управляющего регистра блока RTI (RTICR.7). Этот бит может быть очищен обслуживающей программой таймера блока RTI. Очистка этого бита не требуется для формирования прерывания, а программная установка бита не формирует прерывание.

13.6.3 Управляющие регистры таймера сторожевой схемы WD и блока прерываний реального времени RTI

Управляющие регистры модуля WD/RTI показаны на рисунке 135 и описываются ниже.

Адрес	Регистр	Номер бита							
		7	6	5	4	3	2	1	0
7020h	—	Зарезервировано							
7021h	RTICNTR	D7	D6	D5	D4	D3	D2	D1	D0
7022h	—	Зарезервировано							
7023h	WDCNTR	D7	D6	D5	D4	D3	D2	D1	D0
7024h	—	Зарезервировано							
7025h	WDKEY	D7	D6	D5	D4	D3	D2	D1	D0
7026h	—	Зарезервировано							
7027h	RTICR	RTI FLAG	RTI ENA	Зарезервировано			RTIPS2	RTIPS1	RTIPS0
7028h	—	Зарезервировано							
7029h	WDCR	WD FLAG	Зарезерв.	WDCHK2	WDCHK1	WDCHK0	WDPS2	WDPS1	WDPS0
702Ah	—	Зарезервировано							
702Bh	—	Зарезервировано[1)]							
702Ch	—	Зарезервировано							
702Dh	—	Зарезервировано[1)]							
702Eh	—	Зарезервировано							
702Fh	—	Зарезервировано							

[1)] Зарезервированы для управляющих регистров модуля ФАПЧ тактового сигнала

Рисунок 135 – Управляющие регистры модуля WD/RTI

Регистр счетчика прерывания реального времени (RTICNTR)

8-битный регистр счетчика прерывания реального времени (RTICNTR) содержит значение счетчика реального времени. Он постоянно увеличивается, используя деленное на 128 переполнение от автономного счетчика. Несмотря на то, что это 8-битный счетчик, фактически только 7 бит требуется для работы блока RTI. Восьмой бит (бит 7) может быть считан и использован как бит расширения блока RTI. RTICNTR не останавливается, пока устройство находится в нормальном режиме работы, режиме idle 1, режиме idle 2 или режиме пониженного потребления PLL. Описание бит RTICNTR показано на рисунке 136.

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

R – доступ чтения, C – Очистка, -0 – значение после сброса

Рисунок 136 – Регистр счетчика прерывания реального времени (RTICNTR) –адрес 7021h

Биты 7-0 D7–D0. Значения данных. Эти, доступные только для чтения, биты данных содержат значение 8-битного счетчика блока RTI. Запись любого значения в этот регистр очищает их до 0.

Примечание – Автономный счетчик, который делит RTICNTR, не очищается во время очистки RTICNTR. Это приводит к накопительному максимуму неопределённости одного бита RTICNTR, когда RTICNTR используется для временных измерений.

Регистр счетчика модуля WD (WDCNTR)

8-битный регистр счетчика модуля WD (WDCNTR) содержит текущее значение счетчика модуля WD. Он постоянно увеличивается на значение, выбираемое из управляющего регистра модуля WD. Когда этот регистр переполняется, дополнительная однопериодная (либо WDCLK, либо WDCLK, деленный масштабирующим значением) задержка возникает перед утверждением системного сброса. Это позволяет таймеру блока RTI и таймеру модуля WD быть запрограммированными в одинаковом периоде, пока даётся программное время на запись правильной последовательности ключа модуля WD. Запись правильной последовательности в регистр ключа сброса модуля WD очищает WDCNTR и предотвращает системный сброс; однако, это не очищает автономный счетчик. Описание бит WDCNTR показано на рисунке 137.

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

R – доступ чтения, -0 – значение после сброса

Рисунок 137 – Регистр счетчика модуля WD (WDCNTR) – адрес 7023h

Биты 7-0 D7–D0. Значения данных. Эти, доступные только для чтения, биты данных содержат значение 8-битного счетчика модуля WD. Запись в этот регистр не имеет эффекта.

Регистр ключа сброса модуля WD (WDKEY)

Регистр ключа сброса модуля WD (WDKEY) очищает WDCNTR когда значение 55h, следующее за значением AAh, записывается в WDKEY. Разрешена любая комбинация 55h и AAh, но только запись 55h, следующего за AAh в WDKEY, сбрасывает счетчик. Любые другие значения вызывают системный сброс. Описание бит WDKEY показано на рисунке 138.

7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 138 – Регистр ключа сброса модуля WD (WDKEY) – адрес 7025h

Биты 7-0 D7–D0. Значения данных. Эти, доступные только для записи, биты содержат 8-битное значение ключа сброса модуля WD. Когда считывается, WDKEY не возвращает последнее значение ключа, но возвращает содержимое WDCR.

Управляющий регистр блока RTI (RTICR)

Описание бит RTICR показано на рисунке 139.

7	6	5-3	2	1	0
RTI FLAG	RTI ENA	Зарезервировано	RTIPS2	RTIPS1	RTIPS0
RW-0	RW-0		RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 139– Управляющий регистр блока RTI (RTICR) – адрес 7027h

Бит 7 RTI FLAG. Бит флага прерывания реального времени. Этот статусный бит устанавливается при возникновении переполнения. Бит может быть очищен обслуживающей программой блока RTI (запись 0 очищает бит). Очистка этого бита не требуется для формирования прерывания. Установка бита не формирует прерывание.
0 = Переполнения не произошло.
1 = Произошло переполнение.

Бит 6 RTI ENA. Бит разрешения прерывания реального времени. Этот бит позволяет прерываниям формироваться, когда происходит выбранное переполнение. Очистка этого бита очищает любые незавершенные и еще не утвержденные запросы прерывания. Прерывание выпускается, основываясь на значении автономного счетчика (и RTICNTR при частотах 64 Гц и ниже). Только единичное прерывание запрашивается на лидирующем фронте переполнения.
0 = Очищает любые незавершенные и еще не утвержденные запросы прерывания и запрещает будущие прерывания блока RTI.
1 = Разрешает формирование прерываний, когда RTI FLAG определяет переполнение.
Примечание – Значение автономного счетчика программно независимо. Программа должна полагать, что временной период определен только при измерении между последовательными прерываниями. Программа не может предсказать, когда произошло первое прерывание, исключая вычисленное от системного сброса.

Биты 5-3 Зарезервировано. Запись в эти биты не имеет эффекта. Эти биты всегда считываются как 0.

Биты 2-0 RTIPS2–RTIPS0. Биты выбора делителя прерывания реального времени. Эти биты выбирают разделяющий отвод, используемый для формирования прерывания. В таблице 75 показаны временные данные с предположением, что WDCLK работает на нормальных частотах 16384 или 15625 Гц.

Таблица 75 – Выбор прерываний реального времени

Биты выбора делителя RTI			Делитель WDCLK	16384 Гц WDCLK[1)]		15625 Гц WDCLK[2)]	
RTIPS2	RTIPS1	RTIPS0		Частота, Гц	Время переполнения	Частота, Гц	Время переполнения
0	0	0	4	4096	244,14 мкс	3906,25	256 мкс
0	0	1	16	1024	976,56 мкс	976,56	1,024 мс
0	1	0	64	256	3,91 мс	244,14	4,096 мс
0	1	1	128	128	7,81 мс	122,07	8,192 мс
1	0	0	256	64	15,63 мс	61,04	16,384 мс
1	0	1	512	32	31,25 мс	30,52	32,768 мс
1	1	0	2048	8	125,00 мс	7,63	131,072 мс
1	1	1	16 384	1	1,0 с	0,95	1,049 с

[1)] Может быть сформировано 4,194 МГц кварцем.
[2)] Может быть сформировано 4,0 МГц кварцем.

Управляющий регистр модуля WD (WDCR)

Описание бит WDCR показано на рисунке 140.

7	6	5	4	3	2	1	0
WD FLAG	WDDIS	WDCHK2	WDCHK1	WDCHK0	WDPS2	WDPS1	WDPS0
RW-x		RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -n – значение после сброса (x = неопределённый)

Рисунок 140 – Управляющий регистр модуля WD (WDCR) – адрес 7029h

Примечание – Любые значения, записанные в WDCR, должны включать значение 101_2, записанное в биты 5-3 (WDCHK2 – WDCHK0).

Бит 7 WD FLAG. Бит флага сторожевой схемы. Этот бит указывает, был ли системный сброс утвержден таймером модуля WD. Бит устанавливается в 1 WD-формируемым сбросом. Он не затрагивается любым другим типом системного сброса.
0 = Указывает, что таймер модуля WD не утвердил сброс со времени последней очистки бита.
1 = Указывает, что таймер модуля WD утвердил сброс со времени последней очистки бита.

Бит 6 WDDIS. Запрещение сторожевой схемы. Этот бит действителен (доступен для пользователя) только когда бит HP0 в

SYSCR(управляющий регистр системы) установлен. Это происходит только если ввод WDDIS находится в 3,3 В во время последовательности сброса устройства.
0 = Сторожевая схема разрешена.
1 = Сторожевая схема запрещена.

Бит 5 WDCHK2. Бит проверки сторожевой схемы 2. Этот бит должен быть записан как 1, когда данные записывается в WDCR, иначе будет утверждён системный сброс. Этот бит всегда считывается как 0.
0 = Утверждён системный сброс.
1 = Нормальная работа продолжается, если биты проверки записаны корректно.

Бит 4 WDCHK1. Бит проверки сторожевой схемы 1. Этот бит должен быть записан как 0, когда данные записывается в WDCR, иначе будет утверждён системный сброс. Этот бит всегда считывается как 0.
0 = Нормальная работа продолжается, если биты проверки записаны корректно.
1 = Утверждён системный сброс.

Бит 3 WDCHK0. Бит проверки сторожевой схемы 0. Этот бит должен быть записан как 1, когда данные записывается в WDCR, иначе будет утверждён системный сброс. Этот бит всегда считывается как 0.
0 = Утверждён системный сброс.
1 = Нормальная работа продолжается, если биты проверки записаны корректно.

Биты 2-0 WDPS2–WDPS0. Биты выбора делителя сторожевой схемы. Эти биты выбирают отвод переполнения счетчика, который формирует сброс. Каждая выборка устанавливает максимальное время, проходящее перед обслуживанием логики ключа модуля WD. Все временные выборки предполагают, что WDCLK работает на 16384 Гц. Из-за того, что таймер модуля WD считает на 257 тактов перед переполнением, время дано как минимальное для переполнения (сброса). Максимальный простой может быть вплоть до 1/256 длиннее, чем времена, указанные в таблице 76, из-за добавленной неопределённости, возникающей при отсутствии очистки делителя.

Таблица 76 – Выбор переполнения (простоя) модуля WD

Биты выбора делителя RTI			Делитель WDCLK	16384 Гц WDCLK[2)]		15625 Гц WDCLK[3)]	
WDPS2	WDPS1	WDPS0		Частота, Гц	Время переполнения	Частота, Гц	Время переполнения
0	0	X[1)]	1	64	15,63 мс	61,04	16,38 мс
0	1	0	2	32	31,25 мс	30,52	32,77 мс

Окончание таблицы 76

Биты выбора делителя RTI	Дели-	16384 Гц WDCLK[2)]	15625 Гц WDCLK[3)]

WDPS2	WDPS1	WDPS0	тель WDCLK	Частота, Гц	Время перепол-нения	Частота, Гц	Время пе-репол-нения
0	1	1	4	16	62,50 мс	15,26	65,54 мс
1	0	0	8	8	125,00 мс	7,63	131,07 мс
1	0	1	16	4	250,00 мс	3,81	262,14 мс
1	1	0	32	2	500,00 мс	1,91	524,29 мс
1	1	1	64	1	1,0 с	0,95	1,05 с

[1] X = не важно.
[2] Может быть сформировано 4,194 МГц кварцем.
[3] Может быть сформировано 4,00 МГц кварцем.

13.6.4 Программы таймера сторожевой схемы WD и блока прерываний реального времени RTI

В примере 15 показывается программа разрешения модуля WD/RTI, а в примере 16 – программа запрещения модуля WD/RTI SPI для контроллера ЦПОС.

Пример 15 – Программа разрешения модуля сторожевой схемы (WD) и блока прерываний реального времени (RTI).

```
;***************************************************************
; Имя файла: Пример кода разрешения сторожевой схемы
;
; Описание: Этот пример кода демонстрирует программу требуемую
для ;разрешения сторожевой схемы с временем переполнения в 1,05
с. ;Счетчик сторожевой схемы должен быть сброшен перед перепол-
нением, ;для предотвращения сброса сторожевой схемы. Макрокоман-
да KICK_DOG ;используется для сброса счетчика сторожевой схемы.
Эта ;макрокоманда должна быть помещена в тело основной программы
для ;гарантии, что счетчик, в самом деле, сбрасывается.
;
; Примечание - Код, указанный ниже не является законченной
;программой. Он только указывает шаги необходимые для разрешения
;сторожевой схемы.
;---------------------------------------------------------------
; Описание макрокоманд
;---------------------------------------------------------------
KICK_DOG  .macro          ;Макрокоманда сброса сторожевой схемы
          LDP  #00E0h
          SPLK #05555h, WDKEY
          SPLK #0AAAAh, WDKEY
          LDP  #0h
          .endm
;===============================================================
; Основной код
;===============================================================
         .text
```

```
START:   LDP  #00E0h
         SPLK #002Fh, WDCR ;Разрешение сторожевой схемы с
;максимальным переполнением
         KICK_DOG          ;Сброс счетчика сторожевой схемы
;=================================================================
; Основная программа
;=================================================================
MAIN:                      ;Основная часть кода начинается здесь
;        ...               ;ввести адресные коды
         KICK_DOG          ;Сброс счетчика сторожевой схемы
;        ....              ;ввести адресные коды
         KICK_DOG          ;Сброс счетчика сторожевой схемы
       B  MAIN             ;Основная часть кода завершена в один
;проход, возвращение обратно для продолжения.
```

Пример 16 – Программа запрещения модуля сторожевой схемы (WD) и блока прерываний реального времени (RTI)

```
;*****************************************************************
; Имя файла: Пример кода запрещения сторожевой схемы
;
; Описание: Этот пример кода демонстрирует программу, требуемую
для ;запрещения сторожевой схемы, когда на ввод WDDIS устройства
;приложено 3,3 В. После запрещения сторожевой схемы, Счетчик
;сторожевой схемы должен быть сброшен для очистки выполняемого
;прерывания. Макрокоманда KICK_DOG используется для сброса
;счетчика сторожевой схемы.
;
; Примечание – Код, указанный ниже не является законченной
;программой. Он только указывает шаги необходимые для запрещения
;сторожевой схемы.
;-----------------------------------------------------------------
; Описание макрокоманд
;-----------------------------------------------------------------
KICK_DOG    .macro               ;Макрокоманда сброса сторожевой
схемы
            LDP  #00E0h
            SPLK #055h, WDKEY
            SPLK #0AAh, WDKEY
            LDP  #0h
            .endm
;=================================================================
; Основной код
;=================================================================
            .text
START:      LDP #00E0h
            SPLK #006Fh, WDCR ;Запрещение сторожевой схемы, если
                              ; WDDIS =3,3 В
            KICK_DOG          ;Сброс счетчика сторожевой схемы
;=================================================================
; Основная программа
;=================================================================
```

```
MAIN:                                ;Основная часть кода начинается
здесь
                                   ;ввести адресные коды
           NOP
           NOP
           NOP
;          ....                    ;ввести адресные коды
       B    MAIN                      ;Основная часть кода завершена в
один
;проход, возвращение обратно для продолжения.
```

13.7 Внешний интерфейс памяти

13.7.1 Внешний интерфейс к памяти программ

ИС 1867ВЦ10Т может адресовать до 64К слов памяти программ. Пока микросхема организует доступ к внутрикристальным блокам памяти программ, сигналы внешней памяти PS# и STRB# находятся в высокоимпедансном состоянии. Внешние данные и адресная шина активны, только когда ИС 1867ВЦ10Т выполняет доступ к ячейкам внутри диапазона адресов, картированных в области внешней памяти. В таблице 77 показаны ключевые сигналы для связи с внешней памятью.

Таблица 77 – Ключевые сигналы для внешней связи с памятью программ

Сигнал	Описание
A15–A0	16-битная двунаправленная адресная шина
D15–D0	16-битная двунаправленная шина данных
PS#	Выбор памяти программ
READY	Память готова для завершения периода
RD/WR#	Сигнал чтение/нет запись
STRB#	Активный строб доступа внешней памяти
WREN#	Сигнал разрешения записи
WR/RD#	Сигнал запись/нет чтение

Рисунок 141 показывает пример минимального интерфейса внешней памяти программ. На этом рисунке устройство 1867ВЦ10Т соединяется с двумя 16К×8-битными одиночными ОЗУ. Две памяти шириной в 8 бит используются для обеспечения 16-битной ширины слова, требуемой для ИС 1867ВЦ10Т.

Интерфейс, показанный на рисунке 141, полагает, что период чтения/записи режима ожидания равен нулю, то есть время доступа к памяти было выбрано правильно.

Если используется более медленная память, внутрикристальный генератор периодов ожидания может быть использован для включения одного периода ожидания в цикл обращения. Если требуется больше одного периода ожидания, требуется внешняя логика периода ожидания, которая использует сигнал READY для расширения периода шины на требуемое количество периодов ожидания.

Сигнал выбора программы (PS#) напрямую соединен с выбором кристалла (CS#) для выбора памяти при любом доступе внешней программы. Память адресована в любой из 16К блоков адресов в пространстве программ. Если в пространстве

программ соединено несколько блоков памяти, дешифратор, который содержит PS# и соответствующие биты адреса, может быть использован для управления выбором блока памяти кристалла.

Сигнал WR/RD# связан напрямую с выводом разрешения памяти OE#. Сигнал OE# разрешает выходные драйверы памяти. Драйверы отключаются на время, для гарантии, что не возникнет конфликта шины данных при внешней записи устройством 1867ВЦ10Т.

ИС 1867ВЦ10Т требуется два периода на все внешние записи, включая половину периода перед тем, как WREN# перейдет в низкий уровень и полпериода после того, как WREN# перейдет в высокий уровень. Это предотвращает конфликты буфера на внешних шинах.

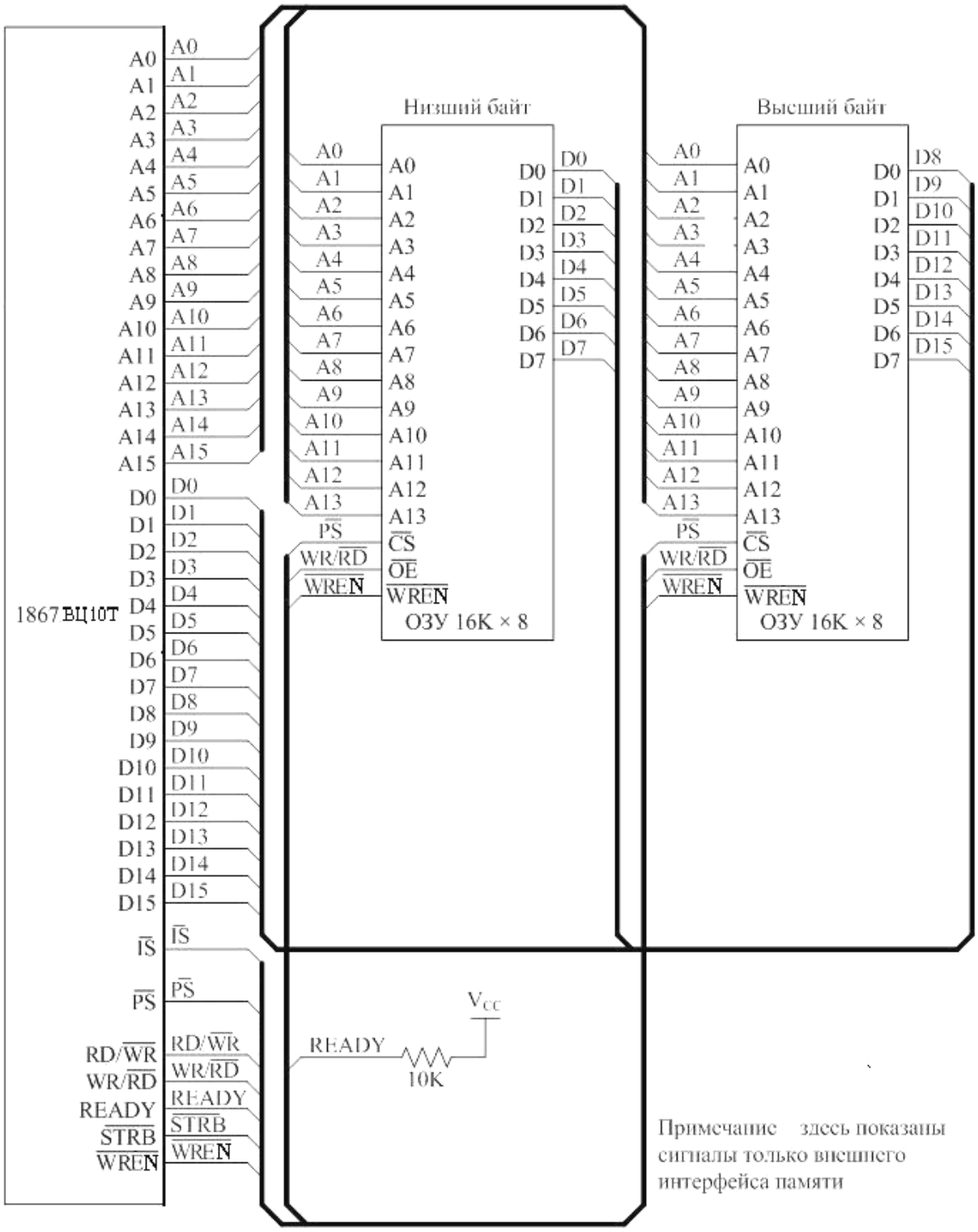


Рисунок 141 – Интерфейс к внешней памяти программ

13.7.2 Интерфейс к пространству ввода-вывода

Доступ к пространству ввода-вывода отличается от доступа памяти данных и программ установлением IS# в низкий уровень. Все 64К слов (внешние порты ввода-вывода и внутрикристальные регистры ввода-вывода) имеют доступ через выполнение команд IN и OUT. Это показано в примерах 17 и 18 соответственно.

Пример 17 – Доступ к внешнему порту ввода-вывода.

```
IN DAT7,   0AFEEh     ; Чтение данных в память данных от
                      ;внешнего устройства на порте 45038.
OUTDAT7,   0CFEFh     ; Запись данных в память данных от
                      ;внешнего устройства на порте 53231.
```

Пример 18 – Доступ к регистру ввода–вывода

```
IN DAT7,   FFFFh      ; Чтение данных в память данных от
                      ;1867ВЦ10Т в управляющий регистр
                      ;генератора периодов ожидания
OUTDAT8,   FFFFh      ; Запись данных в память данных от
                      ;1867ВЦ10Т в управляющий регистр
                      ;генератора периодов ожидания
```

Доступ к внешним параллельным портам ввода-вывода осуществляется по тем же адресным шинам и шинам данных, что и для доступа к памяти данных и программ. Шина данных имеет ширину в 16 бит; однако, если используется 8-битная периферия, можно использовать либо младший, либо старший байты шины данных для соответствия конкретному приложению.

WR/RD# может быть использован с логикой выбора кристалла для формирования сигнала разрешения выхода для внешней периферии. Сигнал WREN# может быть использован с логикой выбора кристалла для формирования сигнала разрешения выхода для внешней периферии. На рисунке 142 показан пример схемы интерфейса для 16 портов ввода-вывода. Следует отметить, что секция декодирования может быть упрощена, если используется меньшее количество портов ввода-вывода.

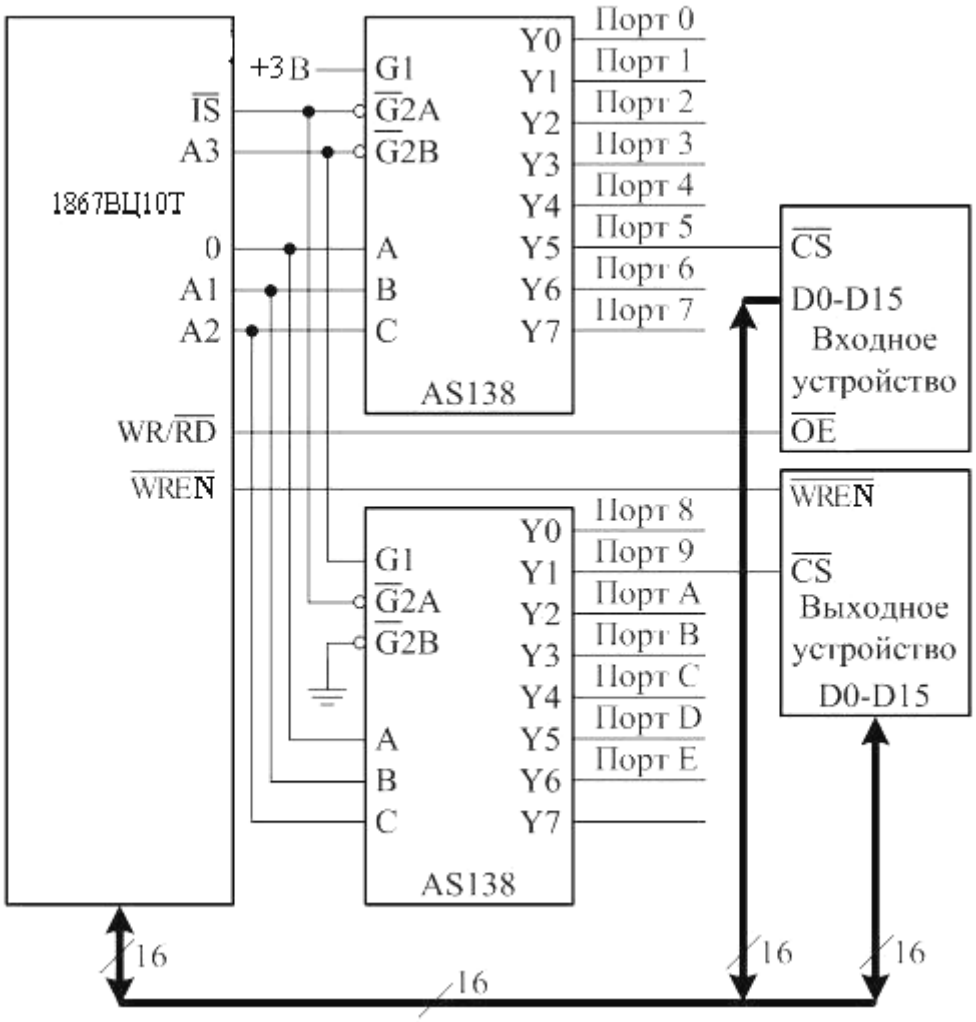


Рисунок 142 – Интерфейс порта ввода-вывода

13.7.3 Временные диаграммы интерфейса памяти

На рисунке 143 показаны сигналы чтения интерфейса памяти, а на рисунке 144 показаны сигналы записи интерфейса памяти. Оба рисунка используются только для демонстрации. Для точных временных параметров следует обратиться к спецификации 1867ВЦ10Т.

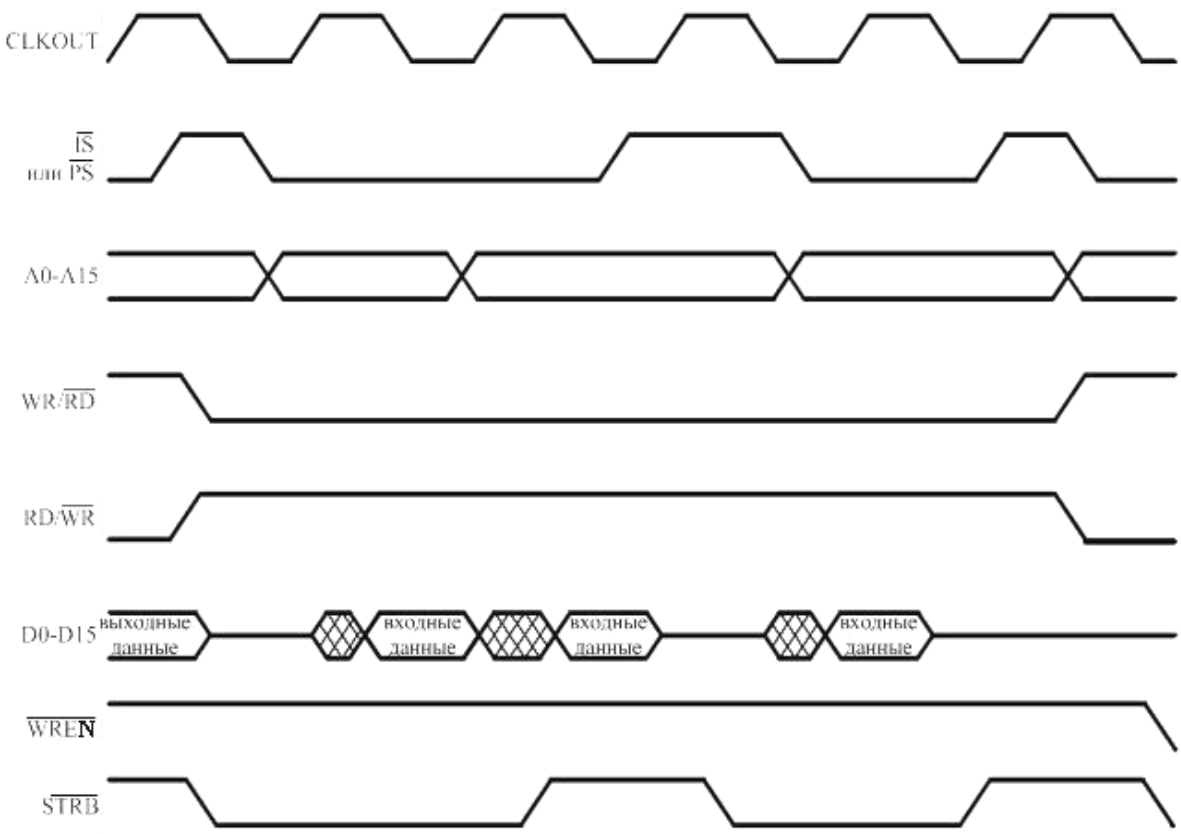


Рисунок 143 – Сигналы чтения интерфейса памяти

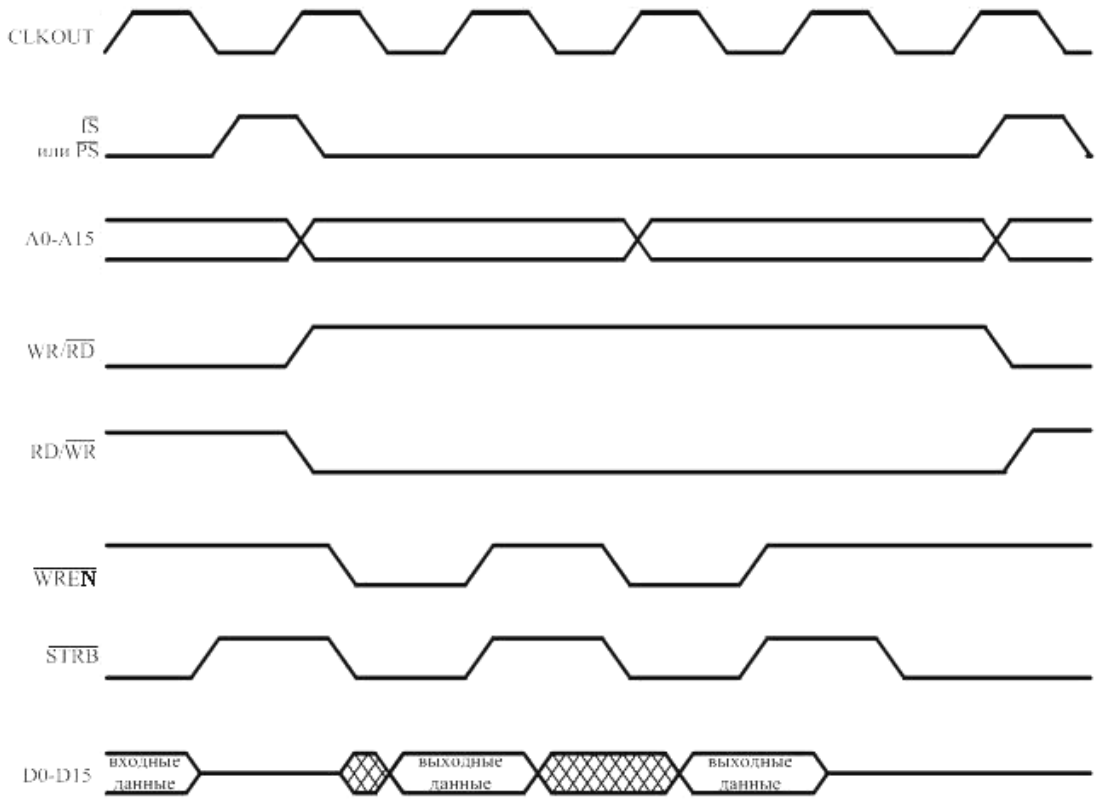


Рисунок 144 – Сигналы записи интерфейса памяти

13.7.4 Генератор периодов ожидания

Периоды ожидания могут быть сформированы при организации доступа к медленным внешним устройствам. Периоды ожидания работают на границах машинного цикла и запускаются либо использованием сигнала готовности, либо использованием программного генератора периодов ожидания. READY может быть использован для формирования любого числа периодов ожидания.

Формирование периодов ожидания с помощью сигнала READY

Установкой сигнала READY в высокий уровень внешнее устройство показывает, что оно готово для завершения обращения к памяти. Если внешнее устройство не готово, оно может удерживать READY в низком уровне так долго, как это требуется. Когда READY в низком уровне, 1867ВЦ10Т ждёт один CLKOUT1 и снова проверяет READY. ИС 1867ВЦ10Т не будет продолжать выполнение, пока READY установлен в высокий уровень, поэтому, если сигнал READY не используется, он должен быть установлен в высокий уровень во время организации внешних и внутренних обращений.

Ввод READY может быть использован для формирования любого числа периодов ожидания. Однако когда ИС 1867ВЦ10Т работает на полной скорости, то может не успеть обеспечить периоды ожидания, основанные на READY для первого периода. Для абсолютной уверенности, что периоды ожидания будут сформированы немедленно, необходимо использовать сначала внутрикристальный генератор периодов ожидания, а затем дополнительные периоды ожидания, которые могут быть сформированы с помощью сигнала READY.

Формирование периодов ожидания с помощью генератора периодов ожидания

Генератор периодов ожидания может быть запрограммирован для формирования первого периода ожидания для данного внекристального пространства памяти (данные программы или ввод-вывод), независимо от состояния сигнала READY. Для управления генератором периодов ожидания, требуемых для записи или чтения, нужно программировать управляющий регистр генератора периодов ожидания (WSGR), картированный в области памяти ввода-вывода по адресу FFFFh. На рисунке 145 показано распределение бит этого регистра.

15-4	3	2	1	0
Зарезервировано	AVIS	ISWS	DSWS	PSWS
0	W-1	W-1	W-1	W-1

W – запись, 0 – чтение устройством как нули, -1 – значение после сброса

Рисунок 145 – Управляющий регистр генератора периодов ожидания (WSGR)

Биты 15-4 Зарезервировано. Всегда считывается как 0.

Бит 3 AVIS. Режим видимости адреса. При сбросе этот бит установлен в 1. AVIS не формирует каких - либо периодов ожидания.
0 = Очищен; уменьшает питание и шумы (для производственных систем).
1 = Разрешает режим видимости адреса устройства. В этом режиме устройство обеспечивает метод трассировки операции внутреннего кода: он передает внутренний адрес программы в шину адреса, когда эта шина не используется для внешнего доступа.

Бит 2 ISWS. Бит периода ожидания пространства ввода - вывода. При сбросе этот бит, установлен в 1.
0 = Не формируется никаких периодов ожидания для внекристального пространства ввода-вывода.
1 = Один период ожидания будет применён ко всем чтениям из внекристального пространства памяти ввода-вывода. (Записи всегда занимают два периода, независимо от PSWS или READY.)

Бит 1 DSWS. Бит периода ожидания пространства данных. При сбросе этот бит установлен в 1.
0 = Не формируется никаких периодов ожидания для внекристального пространства данных.
1 = Один период ожидания будет применён ко всем чтениям из внекристального пространства памяти данных. (Записи всегда занимают два периода, независимо от DSWS или READY.)

Бит 0 PSWS. Бит периода ожидания пространства программ. Этот бит устанавливает конец аналогово-цифрового преобразования (одиночный или сдвоенный канал). При сбросе, этот бит установлен в 1.
0 = Не формируется никаких периодов ожидания для внекристального пространства программ.
1 = Один период ожидания будет применён ко всем чтениям из внекристального пространства памяти программ. (Для предотвращения конфликтов шины, запись всегда занимает два периода, независимо от PSWS или READY.)

Генератор периодов ожидания вносит период ожидания в данное пространство памяти (данные программы или ввод-вывод), если соответствующий бит в WSGR установлен в 1, независимо от состояния сигнала READY. Сигнал READY может быть использован для продолжения периодов ожидания. Все биты WSGR при сбросе устанавливаются в 1, таким образом, устройство может работать от медленной памяти после сброса. Для предотвращения конфликтов шины, записи от генератора периодов ожидания всегда занимают два CLKOUT1 (внутренний тактовый сигнал) периода каждый.

13.8 Цифровые порты ввода-вывода

13.8.1 Обзор цифровых портов ввода-вывода

Модуль цифровых портов ввода-вывода обеспечивает гибкую схему для управления специальными вводами-выводами (внутренние и внешние к ИС 1867ВЦ10Т) и общими функциями выводов. Все функции ввода-вывода и общая функция вывода управляются использованием 16-битных регистров, картированных внутри периферийной области. Эти регистры разделены на три типа:

– Управляющие регистры вывода – используются для прямого управления ввода-вывода или для внутреннего осуществления функции управления кристаллом.

– Статусные регистры ввода – используются для прямого слежения за состоянием ввода-вывода или для внутреннего слежения за статусом событий и состоянием кристалла.

– Управляющие регистры данных и направления – используются для управления данными и направлением данных для двунаправленных вводов-выводов. Эти регистры напрямую соединены с двунаправленными вводами-выводами.

Модуль цифровых портов ввода-вывода максимально разрешает 32 выходные/внутренние управляющие функции, 32 входные/внутренние статусные функции и 32 функции двунаправленных вводов-выводов. Общее число доступных цифровых вводов-выводов, функций управления/статуса и соответствующих регистров зависит от устройства.

13.8.2 Регистры цифровых портов ввода-вывода

В таблице 78 представлены регистры, доступные для модуля цифровых портов ввода-вывода. Как и другие периферийные устройства, регистры картированы в память пространства данных. В частности, эти регистры занимают периферийный блок от 7090h до 709Fh.

Таблица 78 – Адреса регистров цифровых портов ввода-вывода

Адрес	Регистр	Имя регистра
7090h	OCRA	Управляющий регистр вывода A
7092h	OCRB	Управляющий регистр вывода B
7094h	ISRA	Статусный регистр ввода A
7096h	ISRB	Статусный регистр ввода B
7098h	PADATDIR	Управляющий регистр данных и направления порта A ввода-вывода
709Ah	PBDATDIR	Управляющий регистр данных и направления порта B ввода-вывода
709Ch	PCDATDIR	Управляющий регистр данных и направления порта C ввода-вывода
709Eh	PDDATDIR	Управляющий регистр данных и направления порта D ввода-вывода

Управляющий регистр вывода A (OCRA)

Описание бит OCRA показано на рисунке 146.

15	14	13	12	11	10	9	8
OPA15	OPA14	OPA13	OPA12	OPA11	OPA10	OPA9	OPA8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
OPA7	OPA6	OPA5	OPA4	OPA3	OPA2	OPA1	OPA0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 146 – Управляющий регистр вывода A (OCRA) – адрес 7090h

Биты 15-0 OPA15–OPA0
0 = Установка соответствующего вывода или внутренней шины управления в низкий уровень.
1 = Установка соответствующего вывода или внутренней шины управления в высокий уровень.

Управляющий регистр вывода B (OCRB)

Описание бит OCRB показано на рисунке 147.

15	14	13	12	11	10	9	8
OPB15	OPB14	OPB13	OPB12	OPB11	OPB10	OPB9	OPB8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
OPB7	OPB6	OPB5	OPB4	OPB3	OPB2	OPB1	OPB0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 147 – Управляющий регистр вывода B (OCRB) – адрес 7092h

Биты 15-0 OPB15–OPB0
0 = Установка соответствующего вывода или внутренней шины управления в низкий уровень.
1 = Установка соответствующего вывода или внутренней шины управления в высокий уровень.

Статусный регистр ввода A (ISRA)

Описание бит ISRA показано на рисунке 148.

15	14	13	12	11	10	9	8
IPA15	IPA14	IPA13	IPA12	IPA11	IPA10	IPA9	IPA8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
IPA7	IPA6	IPA5	IPA4	IPA3	IPA2	IPA1	IPA0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 148 – Статусный регистр ввода A (ISRA) – адрес 7094h

Биты 15-0 IPA15–IPA0
0 = Соответствующий ввод или внутренний сигнал в низком уровне.
1 = Соответствующий ввод или внутренний сигнал в высоком уровне.

Статусный регистр ввода B (ISRB)

Описание бит ISRB показано на рисунке 149.

15	14	13	12	11	10	9	8
IPB15	IPB14	IPB13	IPB12	IPB11	IPB10	IPB9	IPB8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
IPB7	IPB6	IPB5	IPB4	IPB3	IPB2	IPB1	IPB0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 149 – Статусный регистр ввода B (ISRB) – адрес 7096h

Биты 15-0 IPB15–IPB0
0 = Соответствующий ввод или внутренний сигнал в низком уровне.
1 = Соответствующий ввод или внутренний сигнал в высоком уровне.

Управляющие регистры данных и направления

Описание бит PxDATDIR; x = A, B, C или D показано на рисунке 150.

15	14	13	12	11	10	9	8
x7DIR	x6DIR	x5DIR	x4DIR	x3DIR	x2DIR	x1DIR	x0DIR
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
IOPx7	IOPx6	IOPx5	IOPx4	IOPx3	IOPx2	IOPx1	IOPx0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения; W – доступ записи; -0 – значение после сброса; x – порты A, B, C и D; адреса каждого регистра даны в таблице 79

Рисунок 150 – Управляющие регистры данных и направления (PxDATDIR; x = A, B, C или D)

Биты 15-8 x7DIR–x0DIR
0 = Формирование соответствующего ввода как вход.
1 = Формирование соответствующего ввода как выход.

Биты 7-0 IOPx7–IOPx0
Если xnDIR = 0, то:
0 = Соответствующий ввод-вывод считывается как низкий уровень.
1 = Соответствующий ввод-вывод считывается как высокий уровень.
Если xnDIR = 1, то:
0 = Установка соответствующего ввода-вывода в низкий уровень.
1 = Установка соответствующего ввода-вывода в высокий уровень.

13.9 Модуль фазовой автоподстройки частоты (Phase Locked Loop (PLL))

В этой главе описаны архитектура, функции и программирование модуля PLL тактового сигнала.

Примечание – Этот модуль соединён с 16-битной периферийной шиной как 8-битная периферия. Поэтому, при чтении бит 15 - 8 их значения не определены, запись в эти биты не имеет эффекта.

13.9.1 Обзор модуля PLL тактового сигнала

Модуль PLL тактового сигнала обеспечивает устройство ИС 1867ВЦ10Т необходимыми тактовыми сигналами.

Тактовый модуль связан с периферийной шиной и предоставляет все тактовые сигналы, требуемые для всего устройства (смотри рисунок 151). Существует четыре установки тактовых сигналов, работающих на разных частотах:

– CPUCLK – это тактовый сигнал с наивысшей частотой, предоставляемый модулем и используемый процессором, всеми блоками памяти и любыми периферийными устройствами, соединенными напрямую с процессорными шинами, включая, если используется, интерфейс внешней памяти. Все остальные тактовые сигналы получены делением этого тактового сигнала до меньших частот.

– SYSCLK – этот тактовый сигнал является деленным на 2 или на 4 сигналом CPUCLK. Он используется для тактирования всех периферийных устройств на шине периферии.

– ACLK – этот тактовый сигнал используется для тактирования аналоговых модулей и имеет номинальную частоту в 1,0 МГц ± 10 %, если используется одна из рекомендуемых входных частот и биты CKINF(3:0) запрограммированы правильно, и f_{CPUCLK} имеет четное количество МГц.

– WDCLK – этот низкочастотный тактовый сигнал используется модулем сторожевой схемы/прерываний реального времени. Он имеет номинальную частоту в 16 кГц с 25 % рабочим циклом.

Тактовый модуль работает с 4, 6 или 8 МГц опорным кварцем в соединении с собственной внутрикристальной цепью генератора, или внешним тактовым сигналом шунта генератора в диапазоне от 2 до 32 МГц. PLL может умножать входную частоту на коэффициенты 1, 2, 3, 4, 5 и 9. Входной тактовый сигнал так же может быть делен на 2 перед умножением для получения дополнительных коэффициен-

тов 1, 5, 2, 5 и 4, 5. Действующая частота тактового сигнала процессора выбирается программно от 2 МГц до максимальной рабочей частоты устройства. PLL так же может быть шунтирован с 1 × или 2 × (частота CPUCLK) тактовым входом.

Примечание – Если тактовый вход имеет частоту большую, нежели 32 МГц, частоты ACLK и WDCLK будут неправильными.

Тактовый модуль содержит все требуемые управляющие регистры. Он так же содержит управляющие биты режима пониженного потребления, которые определяют какой тактовый сигнал отключается, когда процессор переходит в режим простоя.

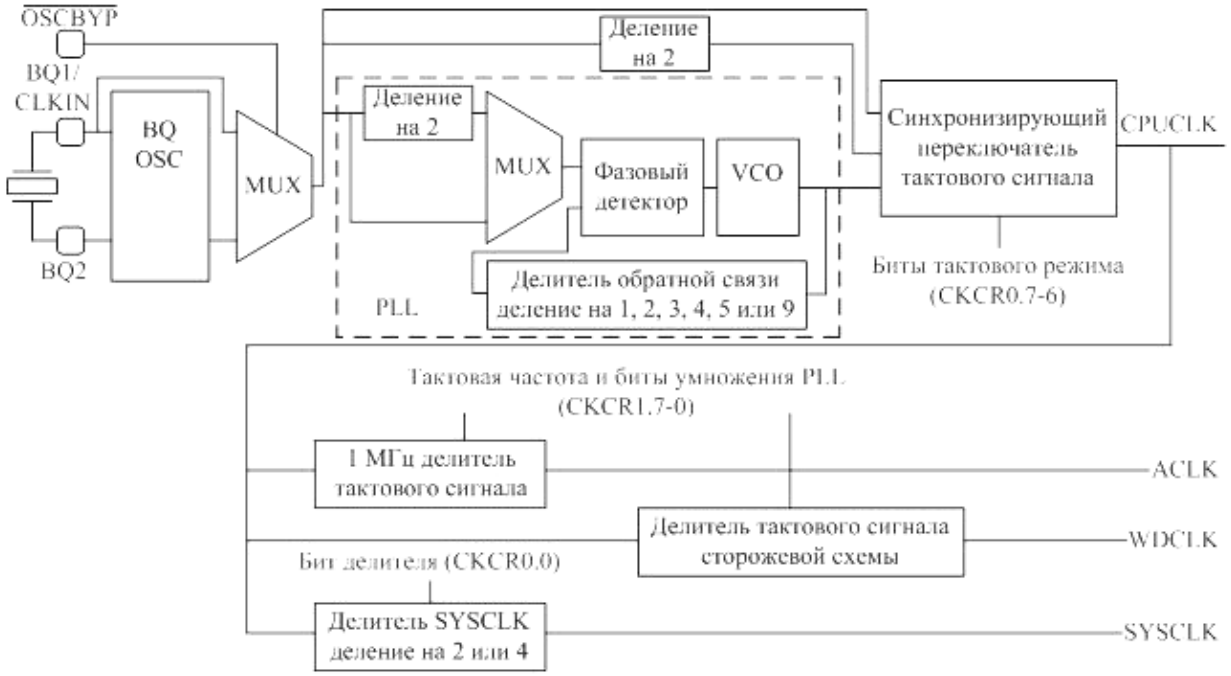


Рисунок 151 – Структурная схема модуля PLL тактового сигнала

Два регистра (показанные в таблице 79) управляют операциями модуля PLL тактового сигнала:

– CKCR0 (Управляющий регистр тактового сигнала 0). Этот регистр содержит биты, используемые для общего управления тактовым модулем, включая тактовый режим, выбор режима меньшего потребления, выбор делителя SYSCLK и статусные флаги.

– CKCR1 (Управляющий регистр тактового сигнала 1). Этот регистр опредляет частоту входного тактового сигнала.

Таблица 79 – Адреса управляющих регистров модуля PLL тактового сигнала:

Адрес	Регистр	Имя
7020h	—	Зарезервировано [1)]
7022h	—	Зарезервировано [1)]
7024h	—	Зарезервировано [1)]
7026h	—	Зарезервировано [1)]
7028h	—	Зарезервировано [1)]

Продолжение таблицы 79

Адрес	Регистр	Имя
702Ah [1)]	CKCR0	Управляющий регистр тактового сигнала 0
702Ch [2)]	CKCR1	Управляющий регистр тактового сигнала 1
702Eh	—	Зарезервировано

[1)] Зарезервировано для управляющих регистров модуля сторожевой схемы и прерываний реального времени.

[2)] Каждый регистр так же появляется в следующем нечетном адресе, то есть CKCR0 появляется в 702Bh и CKCR1 появляется в 702Dh.

13.9.2 Функционирование PLL

В этом подразделе описывается работа и выполняемые функции модуля PLL, включая следующие пункты:

- Описание вводов.
- Режимы работы генератора.
- Режимы работы PLL.
- Выбор частоты тактового сигнала процессора (CPUCLK).
- Выбор делителя системного тактового сигнала (SYSCLK).
- 1 МГц тактовый сигнал аналогового модуля (ACLK).
- Тактовый сигнал счетчика сторожевой схемы (WDCLK).
- Запуск PLL.
- Режимы пониженного потребления.

Описание выводов

Модуль PLL имеет три вывода:

- OSCBYP#. Ввод шунта генератора (OSCBYP#) используется для выбора: шунтируется генерация от кварцевого резонатора или нет. Если устройство используется с внешним таковым входом (то есть, не используется с опорным кварцем), этот сигнал должен быть соединен с 0 В для шунтирования цепи опорного кварцевого резонатора.

- BQ1/CLKIN. Входной контакт кварцевого резонатора (BQ1/CLKIN) обычно соединен с одним из контактов 4, 6 или 8 МГц опорного кварца. Этот вывод также может быть использован как тактовый ввод для генератора внешнего сигнала.

- BQ2. Вывод кварцевого резонатора (BQ2) является соединенным с одним из контактов 4, 6 или 8 МГц опорного кварца, или остается открытым, когда внешний генератор формирует тактовый сигнал через BQ1/CLKIN .

Режимы работы генератора

Генератор имеет два рабочих режима, как показано в таблице 80:

- Режим с кварцевым резонатором.

Это режим работы, когда используется внешний опорный кварц. Этот режим включается, когда ввод OSCBYP# имеет высокий уровень (V_{IH}) и 4, 6 или 8 МГц опорный кварц соединен между BQ1 и BQ2 для обеспечения частоты опорного

кварца. Запуск такого устройства занимает около 1 мс для запуска цепи опорного кварца и начала формирования правильного тактового сигнала.

– Режим внешнего тактового входа от генератора.

Можно шунтировать цепь кварцевого резонатора переводом ввода OSCBYP# в низкий уровень (V_{IL}). Это позволяет устройству тактироваться от внешнего сигнала на вводе BQ1/CLKIN. Цепь кварцевого резонатора выключена, когда она шунтирована.

Таблица 80 – Выбор режима работы гетеродина

Уровни ввода OSCBYP#	Режим работы модуля синхронизации
V_{IH}	Режим с кварцевым резонатором
V_{IL}	Режим внешнего тактового входа от генератора

Режимы работы PLL

Тактовый модуль может работать с PLL, как с источником тактового сигнала или с деленным на 1 или с деленным на 2 параллельным тактовым сигналом.

Биты CLKMD(1:0) (CKCR0.7–6) устанавливают источник тактового сигнала, как показано далее:

CLKMD(1:0)	Режим
00	CLKIN / 2
01	CLKIN
10	PLL
11	PLL

PLL запущен, только когда разрешен, CLKMD(1) = 1.

Этот PLL имеет счетчик для гарантии того, что прошло достаточно времени, чтобы PLL зафиксировался на всех частотах, перед тем как устройство переключится для запуска от тактов PLL. Этот фиксирующий счетчик очищается начальным сбросом. Бит PLLOCK(1) в CKCR0 указывает, что счетчик PLL переполнился, PLL зафиксирован и устройство запускается от тактов PLL.

Коэффициент умножения PLL может быть установлен для умножения на 1, 2, 3, 4, 5 и 9. Он управляется битами PLLFB(2:0) в CKCR1. Кроме того, тактовый вход к PLL может быть делен на 2 перед использованием для получения дополнительных коэффициентов умножения в 1, 5, 2, 5 и 4, 5. Это управляется битом PLLDIV2 в CKCR1.

Выбор частоты тактового сигнала процессора (CPUCLK)

Модуль PLL тактового сигнала дает возможность для формирования одного из многих возможных программно выбираемых частот CPUCLK тактовых сигналов процессора для данного кварца или частоты тактового входа. Выбор фактической частоты CPUCLK управляется четырьмя битами в управляющем регистре CKCR1:

– Биты выбора коэффициента умножения PLL, PLLFB(2:0) (CKCR1.2–0). Эти биты управляют коэффициентом умножения PLL.

– Управляющий бит деленного на 2 входа PLL, PLLDIV2 (CKCR1.3). Этот бит управляет тем, что будет ли тактовый вход PLL делен на 2 или нет.

Следующая формула может быть использована для расчета частоты тактового сигнала процессора для данной частоты кварца и значений регистров:

$$f_{CPUCLK} = f_{CKIN} \times (\text{Коэффициент умножения PLL}) / 2^{PLLDIV2},$$

где 2 МГц < f_{CKIN} < 32 МГц.

В таблице 81 показаны все возможные фиксированные частоты тактового сигнала процессора с 16 различными частотами кварца или тактового входа (которая будет выдавать правильный 1 МГц аналоговый тактовый сигнал ACLK), использованием различных установок бита обратной связи. В зависимости от быстродействия конкретного устройства, некоторые значения таблицы будут не пригодны. То есть, устройство, описанное для запуска на 20 МГц или ниже, не должно использоваться с установками регистров и значениями кварца и тактового входа, которые вырабатывают частоты CPUCLK выше этого значения (затененные значения в таблице 81).

Таблица 81 – Выбираемые частоты тактового сигнала процессора в МГц

Частота кварца или тактового входа, МГц	Коэффициент умножения PLL × $2^{PLLDIV2}$					
	1	2	3	4	5	9
2	2	4	6	8	10	18
4	4	8	12	16	20	36
6	6	12	18	24	30	54
8	8	16	24	32	40	72
10	10	20	30	40	50	90
12	12	24	36	48	60	108
14	14	28	42	56	70	126
16	16	32	48	64	80	144
18	18	36	54	72	90	162
20	20	40	60	80	100	180
22	22	44	66	88	110	198
24	24	48	72	96	120	216
26	26	52	78	104	130	234
28	28	56	84	112	140	252
30	30	60	90	120	150	270
32	32	64	96	128	160	288

Выбор частоты системного тактового сигнала SYSCLK

Частота системного тактового сигнала SYSCLK формируется делением CPUCLK на 2 или 4. Делитель SYSCLK управляется битом выбора деления, PLLPS (CKCR0.0). Этот бит управляет двумя возможными условиями деления, как показано в таблице 82.

Таблица 82 – Условия выбора деления PLL

PLLPS (CKCR0.0)	Условия выбора деления SYSCLK
0	CPUCLK деленный на 4
1	CPUCLK деленный на 2

1 МГц тактовый сигнал аналогового модуля (ACLK)

Тактовый модуль формирует тактовый сигнал аналогового модуля (ACLK) для некоторых аналоговых периферийных модулей. Этот тактовый сигнал имеет номинальную частоту в 1 МГц. ACLK формируется цепью делителя и управляется содержимым бит PLLFB(2:0), PLLDIV2, CKINF(3:0), CLKMD(1:0) и PLLLOCK(1). Эта цепь корректирует деление частоты CPUCLK для создания ACLK как можно ближе к 1,0 МГц ± 10 %, независимо от выбранной частоты CPUCLK, но только пока используется одна из рекомендованных частот тактового входа.

ACLK синхронно запускается/останавливается в зависимости от CPUCLK, при входе - выходе из режимов низкого потребления. Во время приостановки эмулятором, ACLK продолжает работать, для предотвращения повреждений аналоговых блоков при нагрузке.

Существует бит управляющего регистра ACLKENA (CKCR0.1), который используется для включения и выключения ACLK. Это сделано для устройств, которые не требуют 1 МГц ACLK для уменьшения потребляемой мощности и электромагнитных излучений. Бит ACLKENA очищается до 0 после стартового сброса, который вызывает запуск устройства с выключенным 1 МГц тактовым сигналом.

Следует отметить, что если частота CPUCLK является нечетным числом, ACLK будет фактически иметь частоту в 0,5 МГц.

Тактовый сигнал счетчика сторожевой схемы (WDCLK)

Модуль PLL тактового сигнала предоставляет тактовый сигнал счетчика сторожевой схемы (WDCLK) для модуля WD/RTI (если он разрешен в устройстве). WDCLK формируется делением CPUCLK для выработки сигнала WDCLK примерно в 16384 Гц. WDCLK производится цепью делителя и управляется содержимым бит PLLFB(2:0), PLLDIV2, CKINF(3:0), CLKMD(1:0) и PLLLOCK(1). Если биты CKINF(3:0) запрограммированы не корректно, частота WDCLK будет неверной, также как и ACLK.

Следует отметить, что WDCLK равна 16384 Гц (2^{14} Гц) только когда CLKIN имеют степень 2 Гц, смотри таблицу 83.

Таблица 83 – Частоты тактового сигнала счетчика сторожевой схемы

CLKIN, Гц	WDCLK, Гц
4 000 000	15 625
4 194 304 (2^{22})	16 384
8 000 000	15 625
8 388 608 (2^{23})	16 384

Для получения больших или меньших частот WDCLK можно ввести неправильное значение в биты CKINF. Например, если CLKIN = 4,194304 МГц, но CKINF установлен в 1111 (2 МГц), WDCLK будет 32,768 кГц. Следует отметить, что будет затронут ACLK, а значит, такая техника не должна использоваться в устройствах, использующих ACLK.

Запуск PLL

Когда устройство в первый раз запускается, PLL выбирается незапущенным, устройство работает без деленных на два тактов кварцевого резонатора (или с шунтом кварцевого резонатора) (CLKMD = 00). Биты CLKMD(1:0) очищаются до нуля при начальном сбросе, так же как биты PLLFB(2:0) и PLLDIV2.

Если требуются такты PLL, биты PLLFB(2:0) и PLLDIV2 должны быть установлены в требуемые значения и бит CLKMD(1) установлен в 1. PLL запустится и начнет подстраивать частоту. Это занимает около 100 мкс. Устройство будет продолжать работать без деленных на 2 (или 1) тактов, пока фиксирующий счетчик PLL не переполнится, указывая, что PLL достиг фиксации с его новыми установками. В это время, тактовый модуль будет автоматически давать сбои при переключении на такты PLL. Если требуется предотвратить выполнение некоторого кода, перед тем как произошло переключение на такты PLL (более высокую частоту), бит PLLOCK(1) в CKCR0 может быть опрошен. Этот бит указывает, что PLL зафиксирован и устройство работает на тактах PLL.

Последующие изменения в битах PLLFB и DIV2 не будут сразу изменять работу PLL, если он выбран как источник тактового сигнала. Если эти биты изменены, то изменения не произойдут до тех пор, пока PLL не будет выбран очисткой бита CLKMD(1). Бит CLKMD(1) должен быть сразу же установлен обратно в 1, PLL снова запустится и начнет фиксироваться с новыми настройками. Устройство продолжит работать на тактах кварцевого резонатора. Когда PLL снова зафиксируется, устройство снова переключится на такты PLL.

Режимы пониженного потребления

Когда выполняется команда IDLE, энергопотребление уменьшается выключением некоторых или всех внутрикристальных источников тактовых сигналов. Для целей режимов пониженного энергопотребления существуют три различных тактовых области, каждая из которых может выключаться независимо:

– Область тактового сигнала процессора. Все тактовые сигналы процессора, за исключением регистров прерывания.

– Область системного тактового сигнала. Все периферийные тактовые сигналы (CPUCLK или SYSCLK), тактовые сигналы для регистров прерывания процессора и ACLK.

– Тактовый сигнал сторожевой схемы. Тактовый сигнал номиналом в 16 кГц, используемый для приращения таймера сторожевой схемы (WDCLK).

Примечание – Термины CPUCLK и область тактового сигнала процессора, SYSCLK и область системного тактового сигнала не взаимозаменяемы.

Выполнение команды IDLE вызывает включение устройством одного из пяти режимов пониженного потребления, каждый из которых зависит от бит PLLPDM (1:0) (CKCR0.3–2). Выбор бит приведён в таблицах 84, 85.

Таблица 84 – Режимы пониженного потребления

Режим пониженного потребления	Биты PLLPDM (1:0)	Область тактового сигнала процессора	Область системного тактового сигнала/ ACLK	WDCLK
X + not IDLE	XX	Вкл.	Вкл.	Вкл.
0 + IDLE LPM0 (IDLE1)	00	Выкл.	Вкл.	Вкл.
1 + IDLE LPM1 (IDLE2)	01	Выкл.	Выкл.	Вкл.
2 + IDLE LPM2 (Отключение PLL)	10	Выкл.	Выкл.	Вкл.
3 + IDLE LPM3 (Отключение гетеродина)	11	Выкл.	Вкл.	Выкл.

Таблица 85 – Режимы пониженного энергопотребления

Режим пониженного потребления	Состояние PLL[1)]	Состояние гетеродина[1)]	Условие выхода	Описание
X + not IDLE	Вкл.	Вкл.	—	Обычный режим работы
0 + IDLE LPM0 (IDLE1)	Вкл.	Вкл.	Прерывание, сброс	Idle1
1 + IDLE LPM1 (IDLE2)	Вкл.	Вкл.	Прерывание запуска, сброс	Idle2
2 + IDLE LPM2 (Отключение PLL)	Выкл.	Вкл.	Прерывание запуска, сброс	Отключение PLL
3 + IDLE LPM3 (Отключение гетеродина)	Выкл.	Выкл.	Прерывание запуска, сброс	Отключение гетеродина

[1)] Если разрешено.

Режим пониженного потребления может быть возбужден сбросом или любым индивидуальным или общим прерыванием, которое выводит микросхему из состояния пониженного энергопотребления. Доступные действующие прерывания определяются ИС 1867ВЦ10Т, но обычно включают прерывание реального времени (RTI) и внешние прерывания (XINTn). Для определения доступных прерываний запуска микросхемы после Idle нужно обратиться к описанию на устройство.

При выходе из режимов LPM2 и LPM3 получается задержка до 100 мкс перед началом фиксации PLL, если PLL выбран. Кроме того, при выходе из режима LPM3 с генератором от кварца, будет задержка около 1 мс, пока генератор не запустится. Если генератор от кварца шунтирован, то дополнительной задержки не будет.

WDCLK синхронно отключается при входе в режим LPM3. Здесь возможна задержка, пока устройство ожидает WDCLK для ввода своей основной фазы, перед тем как отключатся область тактового сигнала процессора и область системного тактового сигнала.

Режим LPM2 останавливает тактовые сигналы на всех модулях, за исключением того, когда WDCLK всё еще активен. Это означает, что счетчик модуля WD будет активен в режиме LPM2. Как только процессор становится пассивным, модуль WD перестает обслуживаться и эффективно выведет устройство из режима LPM2 сбросом от модуля WD, когда произойдет переполнение модуля WD. Если разрешен блок RTI, то он будет прерывать процессор прерыванием для старта ИС, вызывая выход ИС из режима ожидания. В это время модуль WD может обслуживаться для предотвращения сброса устройства.

Режим LPM3 останавливает все внутренние тактовые сигналы и выключает генератор от кварца. Это останавливает все модули, включая WD/RTI, что приводит к наименьшей возможной потребляемой мощности.

Примечание – Вход в режим LPM2 не выполняется, если PLL не разрешен.

LPMODE 0 последовательность входа-выхода

При входе в этот режим:

1 Область тактового сигнала процессора выключается незамедлительно.

2 Все другие тактовые сигналы продолжают работать.

При выходе из этого режима:

1 Область тактового сигнала процессора снова запускается незамедлительно.

LPMODE 1 последовательность входа-выхода

При входе в этот режим:

1 Область тактового сигнала процессора выключается незамедлительно.

2 Ожидание пока область системного тактового сигнала и ACLK оба не будут в высоком состоянии, а затем остановка в этом состоянии.

3 WDCLK продолжает работать.

При выходе из этого режима при прерывании запуска или сбросе:

1 Внутренние тактовые сигналы тактового модуля запускаются незамедлительно.

2 Позже, на несколько периодов, снова запускаются область тактового сигнала процессора и область системного тактового сигнала и ACLK.

LPMODE 2 последовательность входа-выхода

При входе в этот режим:

1 Область тактового сигнала процессора выключается незамедлительно.

2 Ожидание, пока домен системного тактового сигнала и ACLK не будут в высоком состоянии, а затем остановка в этом состоянии.

3 WDCLK продолжает работать.

4 Тактовые сигналы переключаются от PLL на 1 или на 2 режим.

5 PLL отключается.

При выходе из этого режима при прерывании запуска или сбросе:

1 PLL запускается, и фиксирующий счетчик начинает считать внутренние тактовые сигналы тактового модуля, запускаемые в 1 или во 2 режиме.

2 Снова запускается домен тактового сигнала процессора, область системного тактового сигнала и ACLK.

3 Когда PLL зафиксирован (фиксирующий счетчик переполнен), тактовые сигналы автоматически переключаются обратно на PLL.

LPMODE 3 последовательность входа-выхода

При входе в этот режим:

1 Домен тактового сигнала процессора выключается незамедлительно.

2 Ожидание, пока домен системного тактового сигнала и ACLK не будут в высоком состоянии, а затем остановка в этом состоянии.

3 WDCLK продолжает работать.

4 Тактовые сигналы переключаются от PLL на 1 или на 2 режим.

5 PLL отключается.

6 WDCLK продолжает работать до тех пор, пока основная фаза внутреннего WDCLK в низком уровне.

7 Логика переключения тактового сигнала становится в состояние «всё выключено», то есть все внутренние тактовые сигналы останавливаются.

8 Кварцевый генератор (если используется) выключается.

При выходе из этого режима при прерывании для старта или сбросе:

1 Кварцевый генератор снова разрешается, но выход Кварцевый генератор непригоден для тактирования в течение 1 мс.

2 Логика переключения тактового сигнала переключается в состояние на 1 или на 2, в зависимости от значения бита CKMD(0).

3 PLL запускается и начинает фиксировать частоту.

4 Снова запускаются область тактового сигнала процессора, область системного тактового сигнала и ACLK.

5 Когда PLL зафиксирован (фиксирующий счетчик переполнен), тактовые сигналы автоматически переключаются обратно на PLL.

13.9.3 Управляющие регистры модуля синхронизации

Модуль PLL тактового сигнала управляет и организует доступ через управляющие регистры. Эти регистры показаны и описаны в этом подразделе.

Адрес, показанный для каждого регистра, – это типичный адрес, используемый для таких модулей с начальным адресом 7020h. Другие начальные адреса могут быть использованы в некоторых устройствах, для детальной информации следует обратиться к спецификации устройства.

Управляющий регистр тактового сигнала 0 (CKCR0)

Описание бит CKCR0 показано на рисунке 152.

7	6	5	4	3	2	1	0
CLKMD(1)	CLKMD(0)	LOCK(1)	LOCK(0)	PDM(1)	PDM(0)	ACLKENA	PS
RW-x	RW-x	R-x	R-x	RW-0	RW-0	RW-x	RW-0

R – доступ чтения, W – доступ записи, -x – не реагирует на системный сброс, очищается до нуля при начальном сбросе

Рисунок 152 – Управляющий регистр тактового сигнала 0 (CKCR0) – адрес 702Bh

Биты 7-6 CLKMD(1), CLKMD(0). Биты чтения/записи. Эти биты выбирают режим работы тактового модуля (таблица 86).

Таблица 86 – Тактовый режим в зависимости от бит CLKMD(1:0)

CLKMD(1:0)	Режим
00	CLKIN / 2
01	CLKIN
10	PLL разрешен
11	PLL разрешен

Если устройство входит в режим пониженного потребления, который выключает PLL, то при выходе из этого режима устройство будет работать на CLKIN/2, пока PLL не зафиксируется, если CLKMD = 00, или будет работать на CLKIN, если CLKMD = 01.

Биты 5-4 PLLOCK(1), PLLOCK(0). Биты только чтения. Эти биты определяют, когда PLL вошел в режим, выбранный битами CLKMD(1:0). Бит 0, в действительности, нужен только для теста устройства. Бит 1 может быть использован для определения фиксации PLL. Этот бит может быть программно опрошен, после того как PLL был разрешен, для предотвращения выполнения критичного кода перед переключением модуля на тактовые сигналы PLL. Этот бит не затрагивается системным сбросом и очищается сбросом при включении.
0 = PLL не зафиксирован – работает вне тактового входа.
1 = PLL зафиксирован и работает вне тактов PLL.

Биты 3-2 PLLPDM(1), PLLPDM(0). Биты чтения/записи. Эти биты определяют, какой режим пониженного потребления будет введен во время выполнения команды IDLE. Эти биты очищаются (00b) во время системного сброса и сброса при включении.

Бит 1 ACLKENA. Бит чтения/записи. Разрешает 1 МГц ACLK, если установлен в 1, останавливает ACLK, если очищен до 0. Этот бит не затрагивается системным сбросом и очищается сбросом при включении.
0 = ACLK запрещен (остановлен).
1 = ACLK разрешен.

Бит 0 PLLPS. Бит чтения/записи. Этот бит определяет, какое из двух значений делителя будет выбрано для системных тактовых сигналов. Этот бит очищается (0b) во время системного сброса и сброса при включении, делая CPUCLK/4 частотой по умолчанию системного тактового сигнала (SYSCLK)
0 = $f_{(SYSCLK)} = f_{(CPUCLK)} / 4$
1 = $f_{(SYSCLK)} = f_{(CPUCLK)} / 2$

Управляющий регистр тактового сигнала 1 (CKCR1)

Описание бит CKCR1 показано на рисунке 153.

7	6	5	4	3	2	1	0
CKINF(3)	CKINF(2)	CKINF(1)	CKINF(0)	DIV2	FB(2)	FB(1)	FB(0)
RW-x	RW-x	RW-x	RW-x	RW-x	RW-x	RW-x	RW-x

R – доступ чтения, W – доступ записи, -x – не реагирует на системный сброс, очищается до нуля при начальном сбросе

Рисунок 153 – Управляющий регистр тактового сигнала 1 (CKCR1) – адрес 702Dh

Биты 7-4 CKINF(3)–CKINF(0). Биты чтения/записи. Эти биты определяют используемую частоту кварца или частоту тактового входа (таблица 87). Это используется делителем ACLK для обеспечения формирования 1,0 ± 10 % МГц тактов. Если ACLK не используется никаким модулем в устройстве, любые частоты могут быть использованы (внутри диапазона гетеродина), но если требуются 1 МГц такты, то одна из следующих частот кварца должна быть использована. Эти биты не затрагиваются системным сбросом и очищаются сбросом при включении.

Таблица 87 – Частота тактового входа в зависимости от бит CKINF(3:0)

CKINF(3:0)	Частота, МГц	CKINF(3:0)	Частота, МГц
0000	32	1000	16
0001	30	1001	14
0010	28	1010	12
0011	26	1011	10
0100	24	1100	8
0101	22	1101	6
0110	20	1110	4
0111	18	1111	2

Бит 3 PLLDIV2. Бит чтения/записи. Этот бит определяет, будет ли вход PLL делен на 2. Запись в этот бит не дает эффекта на PLL пока биты CLKMD(1:0) не изменятся от 1 ×. Этот бит не затрагивается системным сбросом и очищается сбросом при включении.
0 = Не делить вход PLL.
1 = Делить вход PLL на 2.

Биты 2-0 PLLFB(2)– PLLFB(0). Биты чтения/записи. Эти биты определяют один из 6 возможных коэффициентов умножения (таблица 88). Запись в эти биты не дает эффекта на PLL, пока биты CLKMD(1:0) не изменятся от 1×. Эти биты не затрагиваются системным сбросом и очищаются сбросом при включении.

Таблица 88 – Коэффициент умножения в зависимости от бит PLLFB(2:0)

PLLFB(2:0)	Коэффициент умножения PLL
000	1
001	2
010	3
011	4
100	5
101	9
110	1
111	1

13.10 Контроллер ЦПОС 1867ВЦ10Т

13.10.1 Обзор контроллера ЦПОС

Устройство 1867ВЦ10Т является первым представителем нового семейства контроллеров ЦПОС, основанном на 1867ВЦ2Т, 1867ВЦ5Т 16-битном цифровом сигнальном процессоре с фиксированной запятой (в таблице 89 приведены некоторые характеристики микросхем). Это новое семейство оптимизировано для приложений цифрового управления двигателем и перемещением. Контроллеры ЦПОС объединяют улучшенную архитектуру ядра процессора 1867ВЦ2Т для недорогих высокоэффективных возможностей обработки и некоторых периферийных устройств, оптимизированных для приложений управления двигателем и перемещением. Эти периферийные устройства включают модуль менеджера событий, который предусматривает таймеры общего назначения и регистры сравнения для формирования до 12 PWM выходов, и сдвоенный 12-битный аналогово-цифровой преобразователь (модуля АЦП), который осуществляет два одновременных преобразования за 1,25 мкс.

На рисунке 154 показаны сигналы ИС 1867ВЦ10Т, а на рисунке 155 – расположение контактов микросхемы.

Таблица 89 – Характеристики контроллеров 1867ВЦ10Т, 1867ВЦ2Т, 1867ВЦ5Т

Устройства	Внутрикристальная память (16-разрядных слов)			Источник питания, В	Период, нс	Тип корпуса
	ОЗУ		ПЗУ			
	Данные	Данные/ программы.				
1867ВЦ2Т	288	256	16К	5,0	50	4229.132-3
1867ВЦ5Т	288	256	16К	5,0	50	4229.132-3
1867ВЦ10Т	**47376**	**512**	**64К**	**3,3/1,8**	**8,33**	4229.132-3

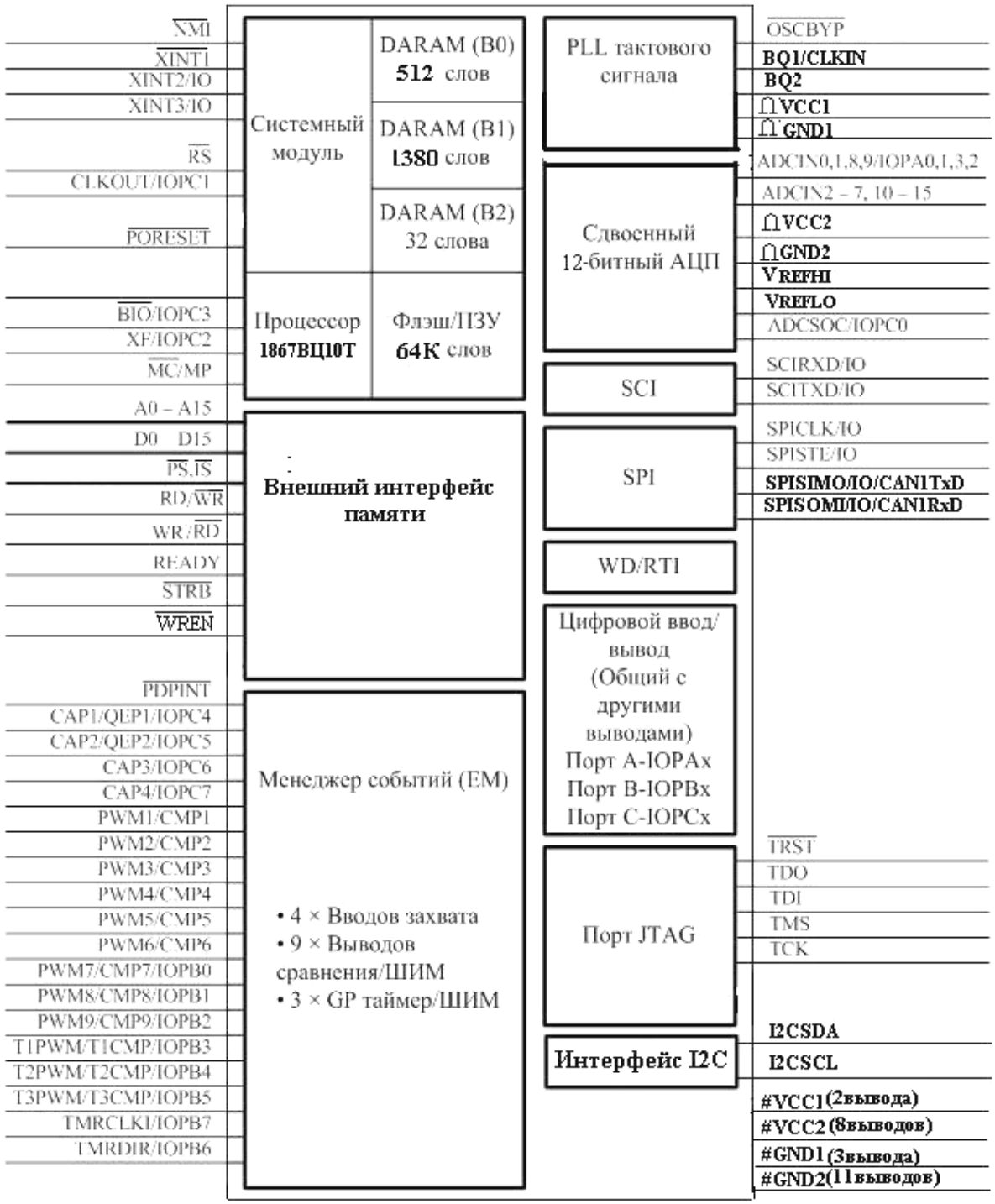


Рисунок 154 – Обзор сигналов ИС 1867ВЦ10Т

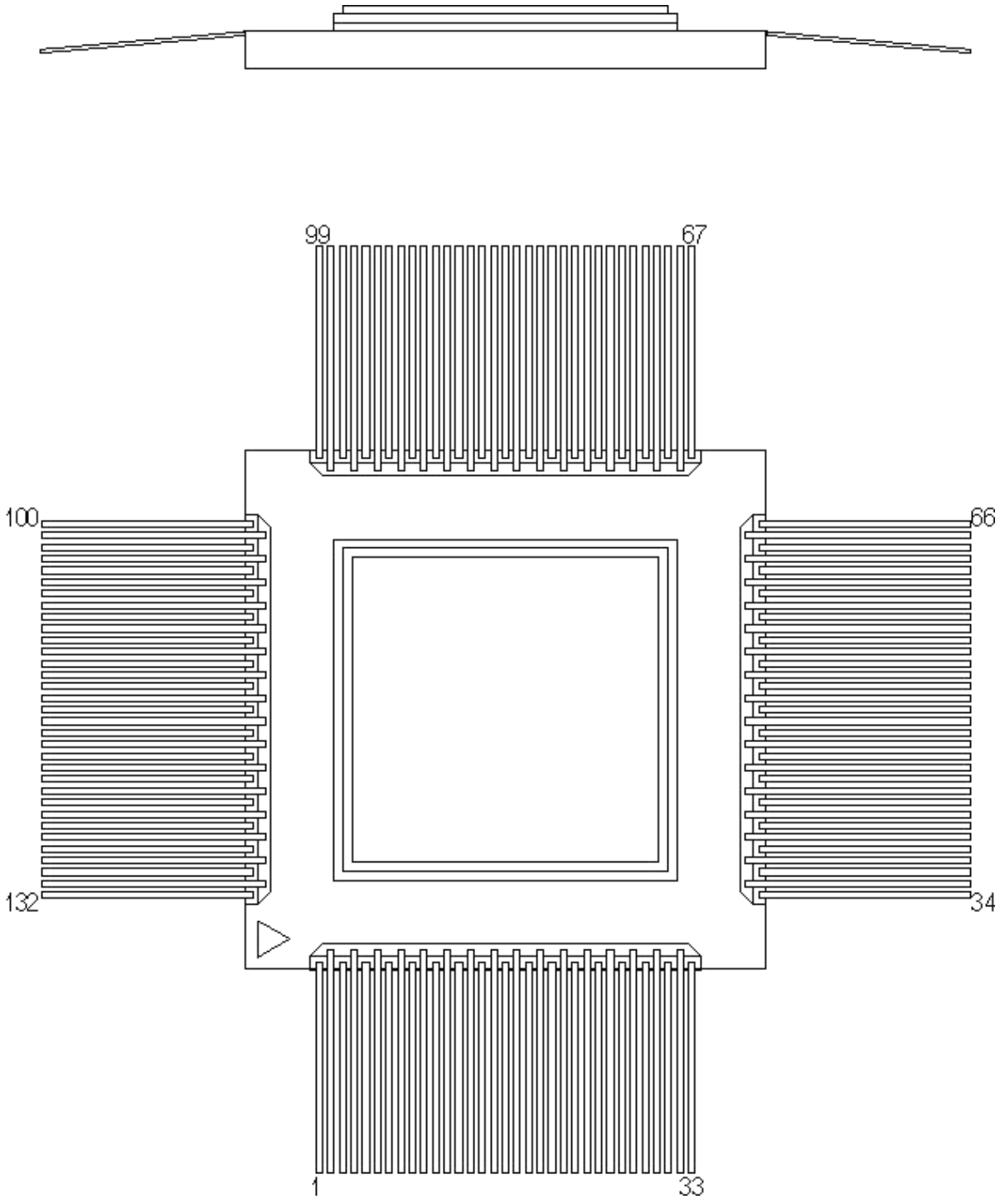


Рисунок 155 – Расположение контактов ИС 1867ВЦ10Т

Особенности 1867ВЦ10Т

ИС 1867ВЦ10Т, показанная на рисунке 154, разработана на базе высокоэффективной статической КМОП технологии.

Имеет процессорное ядро, которое совместимо по исходному коду с М1867ВМ1, 1867ВМ2, 1867ВЦ2Т, 1867ВЦ5Т.

Реализовано в 132-выводной корпус (4229.132-3).

Имеет время выполнения команд 8,33 нс.

Распределение памяти представлено в 13.10.2.

Содержит модуль управления событиями, включающий:

- 12 широтно-импульсно модулированных (PWM) каналов (9 независимых);
- три 16-битных универсальных таймера с шестью режимами, включающие непрерывное суммирование и непрерывное суммирование/вычитание;
- три 16-битных блока полного сравнения с возможностью задания «мертвого» времени,
- три 16-битных блока простого сравнения;
- четыре блока захвата, два из которых имеют интерфейс «квадратурной» обработки сигналов импульсного датчика положения.

ИС 1867ВЦ10Т содержит:

- два 12-битных аналогово-цифровых преобразователя;
- 28 индивидуально-программируемых, мультиплексированных выводов входа / выхода;
- модуль фазовой автоподстройки частоты тактового сигнала (PLL);
- модуль сторожевого таймера (WD) и блока прерываний реального времени (RTI);
- модуль последовательного коммуникационного интерфейса (SCI);
- модуль последовательного периферийного интерфейса (SPI);
- модуль интерфейса CAN;
- модуль интерфейса I2C.

Микросхема обеспечивает:

- шесть внешних прерываний: прерывание защиты электропитания, сброс, NMI и три маскируемых прерывания;
- четыре режима пониженного потребления энергии для снижения потребляемой мощности.

Микросхема включает:

- канирующий эмулятор;
- Доступные средства для разработки:
  - TI ANSI C компилятор, ассемблер/редактор и отладчик С-кода;
  - внутрикристальный эмулятор ВЦ10Т;
  - библиотеки программ для управления двигателем и поддержка разработки приложений нечёткой логики.

Обзор архитектуры

Структурная схема функциональных блоков (рисунок 156) обеспечивает высокоуровневое описание каждого компонента контроллера ЦПОС 1867ВЦ10Т. Устройства 1867ВЦ10Т составлены из трёх основных функциональных блоков: ядро ЦПОС, внутренней памяти и периферийных устройств. В дополнение к этим трём функциональным блокам существует несколько распределённых особенностей системного уровня 1867ВЦ10Т. Эти системные особенности включают карту памяти, сброс устройства, прерывания, цифровые вводы-выводы, формирование тактового сигнала и работа с пониженным потреблением энергии.

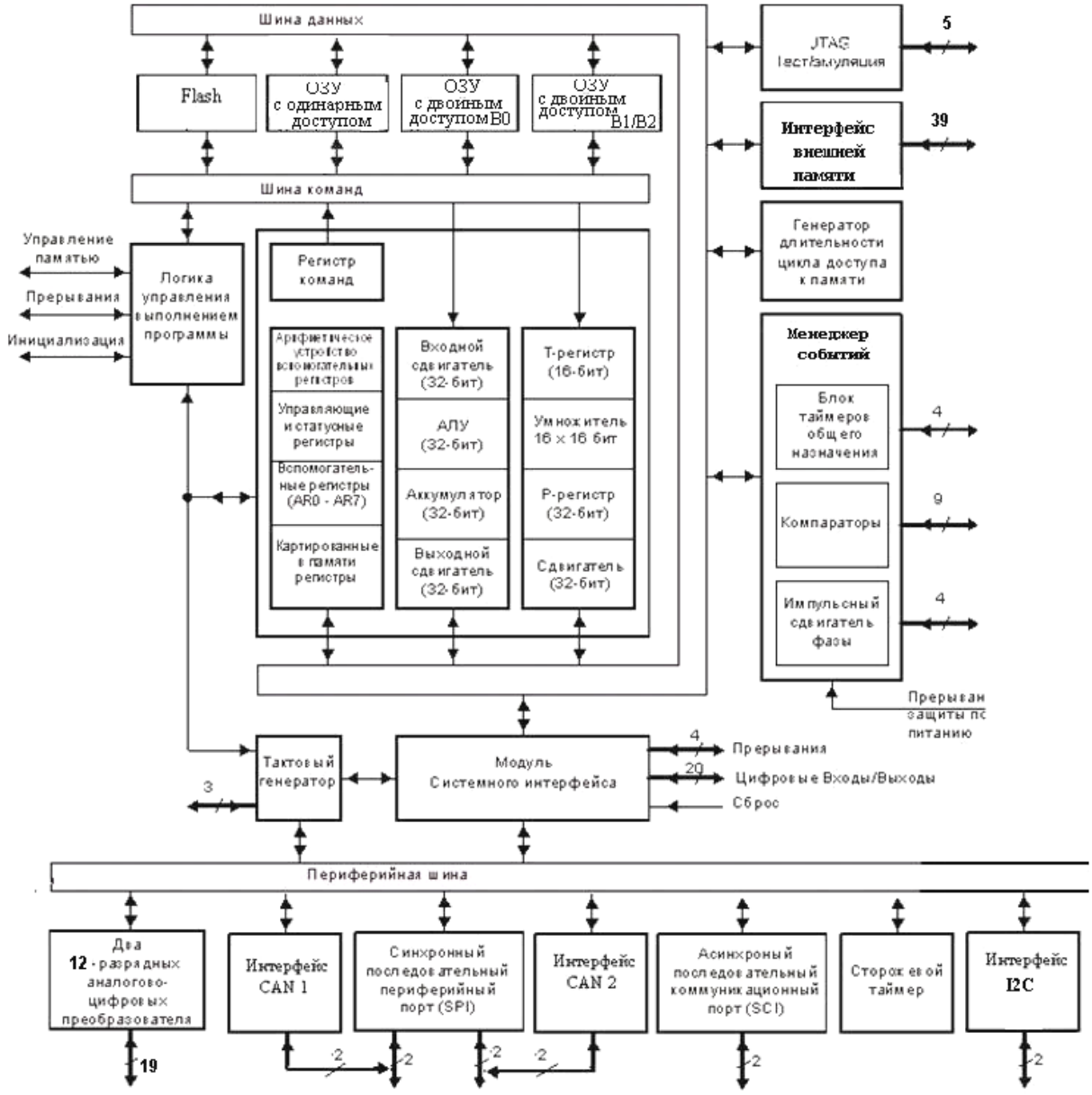


Рисунок 156 – Структурная схема функциональных блоков 1867ВЦ10Т

13.10.2 Карта памяти

ИС 1867ВЦ10Т реализует три отдельных адресных пространства: адресное пространство программной памяти, памяти данных и I/O (ввода-вывода).

ИС 1867ВЦ10Т может адресовать память программ

- 64К слов внутрикристальной Flash памяти при MP/MC=0;
- 64К слов внешней памяти при MP/MC=1;
- 0.5К слов внутрикристальной DARAM при CNF=1.

ИС 1867ВЦ10Т может адресовать память данных

- 45К слов внутрикристальной SRAM;
- 1824 слов ×16 бит внутрикристальной DARAM при CNF=0;
- 1312 слов ×16 бит внутрикристальной DARAM при CNF=1.

ИС 1867ВЦ10Т может адресовать память пространства ввода-вывода

- 64К слов внешней памяти.

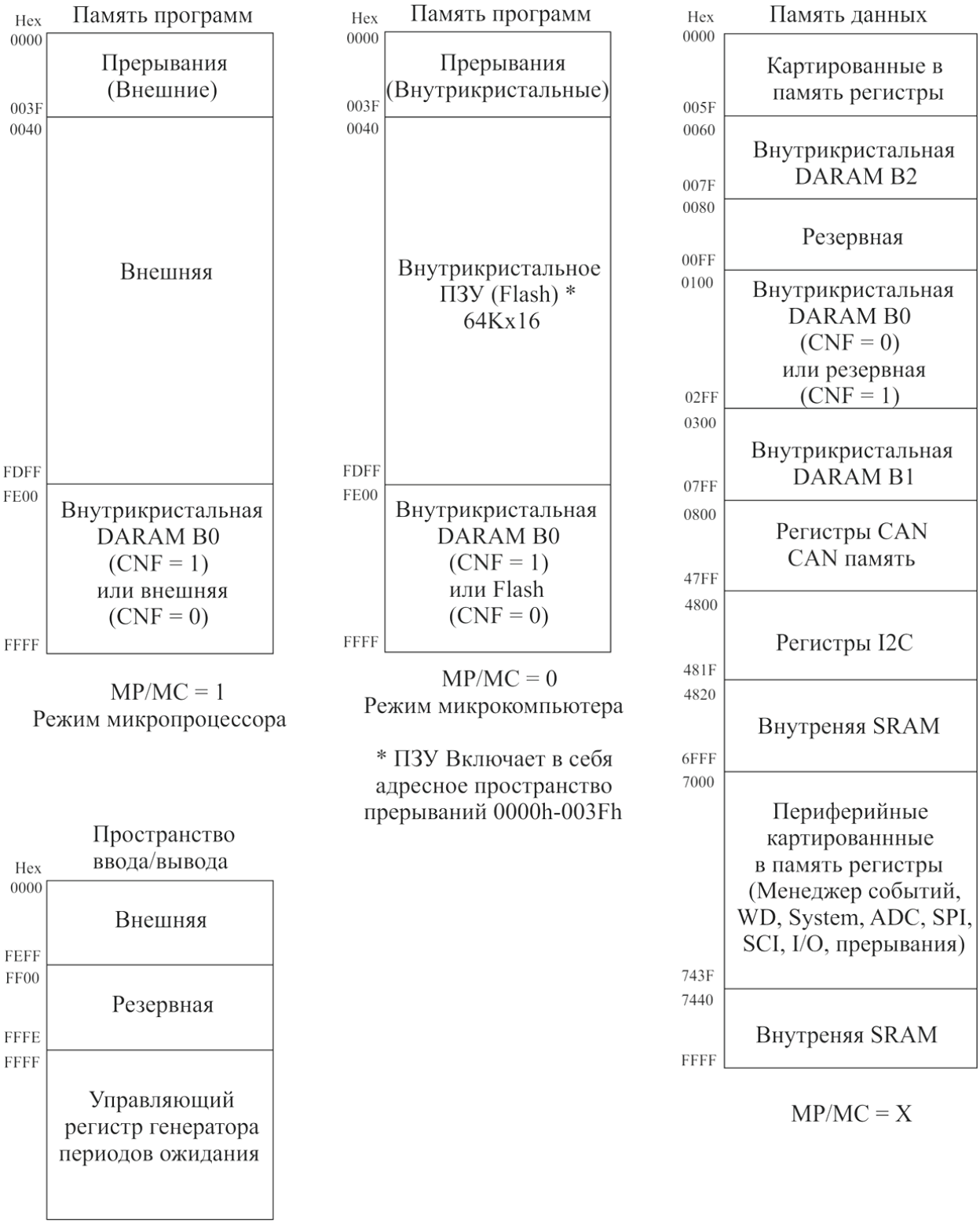


Рисунок 157 – Карта памяти 1867ВЦ10Т

13.10.3 Периферийная карта памяти

Управляющие регистры системы и периферийных устройств содержат все биты данных, статуса и управления для работы с системными и периферийными модулями в устройстве (исключая модуль менеджера событий). На рисунке 158 представлена периферийная карта памяти.

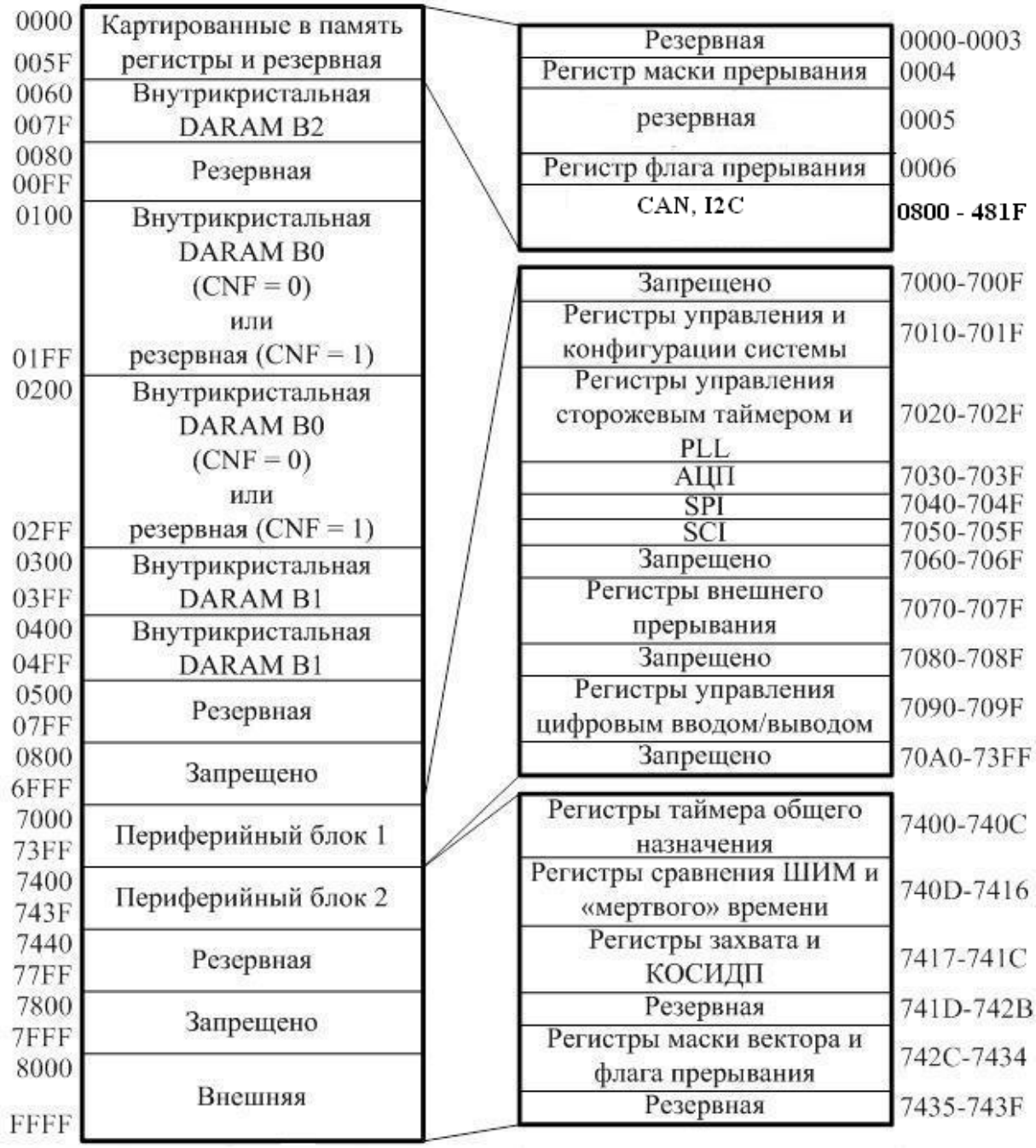


Рисунок 158 – Периферийная карта памяти 1867ВЦ10Т

13.10.4 Функции цифровых вводов-выводов и общих выводов

ИС 1867ВЦ10Т имеет 28 выводов общих для первичных функций и вводов-выводов. Эти выводы разделены на две группы:

– Группа 1 – первичные функции, объединенные с вводами-выводами, относящиеся к специализированным портам ввода-вывода, порт A, порт B и порт C.

– Группа 2 – первичные функции, относящиеся к периферийным модулям, которые так же имеют встроенную способность ввода-вывода как второстепенную функцию, например, внешние прерывания модулей SCI, SPI и модуль PLL.

Описание группы 1 общих вводов-выводов

Управляющая структура для общих выводов группы 1 показана на рисунке 159. Единственным исключением в этой конфигурации является ввод CLKOUT/IOPC1, который описывается далее в этой главе. Каждый вывод имеет три бита, которые определяют его работу:

– бит управления мультиплексированием – этот бит выбирает между первичной функцией (1) и функцией ввода-вывода (0) на выводе;

– бит направления ввода-вывода – если выбрана функция ввода-вывода (бит управления мультиплексированием установлен в 0), этот бит определяет, будет на этом выводе вход (0) или выход (1);

– бит данных ввода-вывода – если выбрана функция ввода-вывода (бит управления мультиплексированием установлен в 0) и направление установлено как вход, данные считываются из этого бита; если направление выбрано как выход, данные записываются в этот бит.

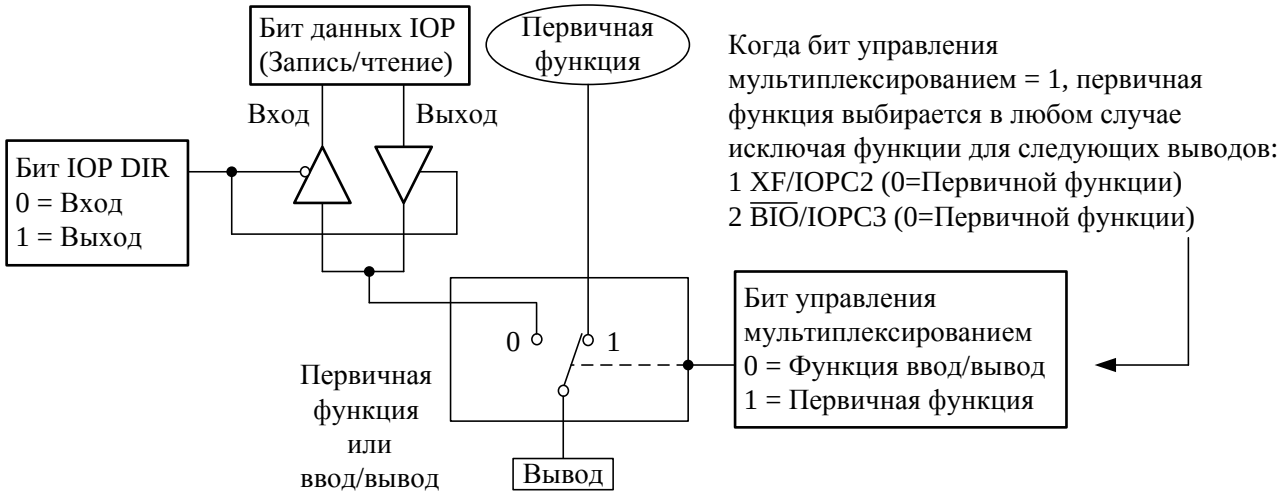


Рисунок 159 – Конфигурация общих выводов

Конфигурация вводов группы 1 и соответствующие биты показаны в таблице 90.

Таблица 90 – Конфигурация общих выводов 1867ВЦ10Т

Вывод	Регистр управления мультиплек-сированием (имя бит)	Выбранная функция вывода		Данные и направление порта ввода-вывода[1)]		
		CRx.n = 1	CRx.n = 0	Регистр	Бит данных	Бит направления
55	CRA.0	ADCIN0	IOPA0	PADATDIR	0	8
56	CRA.1	ADCIN1	IOPA1	PADATDIR	1	9
73	CRA.2	ADCIN9	IOPA2	PADATDIR	2	10
74	CRA.3	ADCIN8	IOPA3	PADATDIR	3	11
83	CRA.8	PWM7/CMP7	IOPB0	PBDATDIR	0	8

Окончание таблицы 90

Вывод	Регистр управления мультиплек-сированием (имя бит)	Выбранная функция вывода		Данные и направление порта ввода-вывода[1)]		
		CRx.n = 1	CRx.n = 0	Регистр	Бит дан-ных	Бит направ ления
84	CRA.9	PWM8/CMP8	IOPB1	PBDATDIR	1	9
85	CRA.10	PWM9/CMP9	IOPB2	PBDATDIR	2	10
88	CRA.11	T1PWM/T1CMP	IOPB3	PBDATDIR	3	11
89	CRA.12	T2PWM/T2CMP	IOPB4	PBDATDIR	4	12
90	CRA.13	T3PWM/T3CMP	IOPB5	PBDATDIR	5	13
91	CRA.14	TMRDIR	IOPB6	PBDATDIR	6	14
92	CRA.15	TMRCLK	IOPB7	PBDATDIR	7	15
46	CRB.0	ADCSOC	IOPC0	PCDATDIR	0	8
47	SCR.7–6[2)]					
	00	IOPC1		PCDATDIR	1	9
	01	CLKOUT (такт WD)		—	—	—
	10	CLKOUT (SYSCLK)		—	—	—
	11	CLKOUT (CPUCLK)		—	—	—
48	CRB.2	IOPC2	XF	PCDATDIR	2	10
49	CRB.3	IOPC3	$\overline{BIO}$	PCDATDIR	3	11
50	CRB.4	CAP1/QEP1	IOPC4	PCDATDIR	4	12
51	CRB.5	CAP2/QEP2	IOPC5	PCDATDIR	5	13
52	CRB.6	CAP3	IOPC6	PCDATDIR	6	14
53	CRB.7	CAP4	IOPC7	PCDATDIR	7	15

[1)] Действительны только когда выбрана функция ввода-вывода.
[2)] Биты 7 и 6 с регистре управления системой.

Описание группы 2 общих вводов-выводов

Группа 2 общих выводов относится к периферийным устройствам, которые имеют встроенную способность ввода-вывода общего назначения. Управление и конфигурация этих выводов осуществляется установкой соответствующих бит в регистрах управления и конфигурации периферийных устройств. В таблице 91 показаны общие выводы группы 2.

Таблица 91 – Группа 2 общих выводов

Вывод	Первичная функция	Периферийный модуль
40	SCIRXD	SCI
41	SCITXD	SCI
27	SPISIMO	SPI
31	SPISOMI	SPI
32	SPICLK	SPI
34	SPISTE	SPI

Окончание таблицы 91

Вывод	Первичная функция	Периферийный модуль
37	XINT2	Внешние прерывания
38	XINT3	Внешние прерывания

Управляющие регистры цифрового ввода-вывода

В таблице 92 представлены регистры, доступные для модуля цифрового ввода-вывода. Как и другие периферийные устройства, эти регистры картированы в памяти в пространстве данных.

Таблица 92 – Адреса управляющих регистров цифрового ввода-вывода

Адрес	Регистр	Имя
7090h	OCRA	Управляющий регистр мультиплексирования ввода-вывода A
7092h	OCRB	Управляющий регистр мультиплексирования ввода-вывода B
7098h	PADATDIR	Регистр данных и направления порта A ввода-вывода
709Ah	PBDATDIR	Регистр данных и направления порта B ввода-вывода
709Ch	PCDATDIR	Регистр данных и направления порта C ввода-вывода

Управляющие регистры мультиплексирования ввода-вывода

Описание бит OCRA показано на рисунке 160.

15	14	13	12	11	10	9	8
CRA.15	CRA.14	CRA.13	CRA.12	CRA.11	CRA.10	CRA.9	CRA.8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
Зарезервировано				CRA.3	CRA.2	CRA.1	CRA.0
RW-0				RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 160 – Управляющий регистр мультиплексирования ввода-вывода (OCRA) – адрес 7090h

Конфигурация управляющего регистра мультиплексирования ввода-вывода (OCRA) представлена в таблице 93.

Таблица 93 – Конфигурация управляющего регистра мультиплексирования ввода-вывода (OCRA)

Бит	Имя.бит	Выбранные функции вывода	
		(CRA.n = 1)	(CRA.n = 0)
0	CRA.0	ADCIN0	IOPA0
1	CRA.1	ADCIN1	IOPA1
2	CRA.2	ADCIN9	IOPA2
3	CRA.3	ADCIN8	IOPA3

Окончание таблицы 93

Бит	Имя.бит	Выбранные функции вывода	
		(CRA.n = 1)	(CRA.n = 0)
8	CRA.8	PWM7/CMP7	IOPB0
9	CRA.9	PWM8/CMP8	IOPB1
10	CRA.10	PWM9/CMP9	IOPB2
11	CRA.11	T1PWM/T1CMP	IOPB3
12	CRA.12	T2PWM/T2CMP	IOPB4
13	CRA.13	T3PWM/T3CMP	IOPB5
14	CRA.14	TMRDIR	IOPB6
15	CRA.15	TMRCLK	IOPB7

Описание бит OCRB показано на рисунке 161.

15	14	13	12	11	10	9	8
Зарезервировано							
RW-0							
7	6	5	4	3	2	1	0
CRB.7	CRB.6	CRB.5	CRB.4	CRB.3	CRB.2	CRB.1	CRB.0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 161 – Управляющий регистр мультиплексирования ввода-вывода (OCRB) – адрес 7092h

Конфигурация управляющего регистра мультиплексирования ввода-вывода (OCRB) представлена в таблице 94.

Таблица 94 – Конфигурация управляющего регистра мультиплексирования ввода-вывода (OCRB)

Бит	Имя.бит	Выбранные функции вывода	
		(CRB.n = 1)	(CRB.n = 0)
0	CRB.0	ADCSOC	IOPC0
2	CRB.2	IOPC2	XF
3	CRB.3	IOPC3	BIO#
4	CRB.4	CAP1/QEP1	IOPC4
5	CRB.5	CAP2/QEP2	IOPC5
6	CRB.6	CAP3	IOPC6
7	CRB.7	CAP4	IOPC7

Регистры данных и направления порта ввода-вывода

Описание бит PADATDIR показано на рисунке 162.

15	14	13	12	11	10	9	8
Зарезервировано				A3DIR	A2DIR	A1DIR	A0DIR
				RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
Зарезервировано				IOPA3	IOPA2	IOPA1	IOPA0
				RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса, n= 0-3

Рисунок 162 – Регистр данных и направления порта A ввода - вывода (PADATDIR) – адрес 7098h

Биты 15-12 Зарезервировано.

Биты 11-8 A3DIR–A0DIR. Управляющие биты направления порта A
0 = Формирование соответствующего вывода как вход.
1 = Формирование соответствующего вывода как выход.

Биты 7-4 Зарезервировано.

Биты 3-0 IOPA3–IOPA0. Биты данных порта A
Если AnDIR = 0, то:
0 = Соответствующий ввод-вывод считывается как низкий уровень.
1 = Соответствующий ввод-вывод считывается как высокий уровень.
Если AnDIR = 1, то:
0 = Установка соответствующего ввода-вывода в низкий уровень.
1 = Установка соответствующего ввода-вывода в высокий уровень.

Описание бит PBDATDIR показано на рисунке 163.

15	14	13	12	11	10	9	8
B7DIR	B6DIR	B5DIR	B4DIR	B3DIR	B2DIR	B1DIR	B0DIR
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
IOPB7	IOPB6	IOPB5	IOPB4	IOPB3	IOPB2	IOPB1	IOPB0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса, n= 0-7

Рисунок 163 – Регистр данных и направления порта B ввода-вывода (PBDATDIR) – адрес 709Ah

Биты 15-8 B7DIR–B0DIR. Управляющие биты направления порта B
0 = Формирование соответствующего вывода как вход.
1 = Формирование соответствующего вывода как выход.

Биты 7-0 IOPB7–IOPB0. Биты данных порта B
Если BnDIR = 0, то:
0 = Соответствующий ввод-вывод считывается как низкий уровень.

1 = Соответствующий ввод-вывод считывается как высокий уровень.
Если BnDIR = 1, то:
0 = Установка соответствующего ввода-вывода в низкий уровень.
1 = Установка соответствующего ввода-вывода в высокий уровень.

Описание бит PCDATDIR показано на рисунке 164.

15	14	13	12	11	10	9	8
C7DIR	C6DIR	C5DIR	C4DIR	C3DIR	C2DIR	C1DIR	C0DIR
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
IOPC7	IOPC6	IOPC5	IOPC4	IOPC3	IOPC2	IOPC1	IOPC0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса, n= 0-7

Рисунок 164– Регистр данных и направления порта C ввода-вывода (PCDATDIR) – адрес 709Ch

Биты 15-8 C7DIR–C0DIR. Управляющие биты направления порта B
0 = Формирование соответствующего вывода как вход.
1 = Формирование соответствующего вывода как выход.

Биты 7-0 IOPC7–IOPC0. Биты данных порта B
Если CnDIR = 0, то:
0 = Соответствующий ввод-вывод считывается как низкий уровень.
1 = Соответствующий ввод-вывод считывается как высокий уровень.
Если CnDIR = 1, то:
0 = Установка соответствующего ввода-вывода в низкий уровень.
1 = Установка соответствующего ввода-вывода в высокий уровень.

13.10.5 Сброс и прерывания

Структура программируемых прерываний ИС 1867ВЦ10Т поддерживает гибкую конфигурацию внутрикристальных и внешних прерываний для требований приложений реального времени, управляемых прерыванием. Микросхема распознает четыре вида источников прерывания:

– Сброс (вызываемый аппаратно или программно) не управляется процессором и занимает высший приоритет среди всех других выполняемых функций. Все маскируемые прерывания запрещены, пока обслуживающая программа сброса не разрешит их.

– Аппаратно формируемые прерывания запрашиваются внешними выводами или внутрикристальной периферией. Их существует два типа:

– Внешние прерывания формируются одним из пяти внешних выводов, соответствующих прерываниям XINT1, XINT2, XINT3, PDPINT и NMI. Первые четыре могут быть маскированы определенными битами разрешения и регистром маски прерывания процессора (IMR), который может маскировать каждую маскируемую шину прерывания в ядре ЦПОС. Немаскируемое NMI имеет высший прио-

ритет среди прерываний периферии и программно формируемых прерываний. Оно может быть заблокировано только выполняемым NMI или сбросом.

– Периферийные прерывания вызываются внутренне следующими внутрикристальными периферийными модулями: менеджером событий, SPI, SCI, WD/RTI и модуля АЦП. Они могут быть маскированы битами разрешения для каждого события в каждом периферийном устройстве и регистром маски прерывания процессора (IMR), который может маскировать каждую маскируемую шину прерывания в ядре ЦПОС.

- Программно формируемые прерывания для устройства включают:

– Команда INTR. Эта команда позволяет инициализировать любое прерывание 1867ВЦ10Т программно. Её операнд указывает на какую позицию вектора прерывания переходит процессор. Эта команда глобально запрещает маскируемые прерывания (устанавливает бит INTM в 1).

– Команда NMI. Эта команда вызывает переход на позицию вектора прерывания 24h, такая же позиция используется для немаскируемого аппаратного прерывания NMI. NMI может быть запущена установкой вывода NMI в низкий уровень или выполнением команды NMI. Эта команда глобально запрещает маскируемые прерывания.

– Команда TRAP. Эта команда вызывает переход процессора на позицию вектора прерывания 22h. Команда TRAP не запрещает маскируемые прерывания (бит INTM не установлен в 1); таким образом, когда процессор переходит к обслуживающей программе прерывания, эта программа может быть прервана маскируемыми аппаратными прерываниями.

- Системное прерывание эмулятора. Это прерывание может быть сформировано или командой INTR или командой TRAP.

Сброс

Операция сброса гарантирует упорядоченную последовательность запуска ИС. Существует четыре возможных причины сброса, как показано на рисунке 165. Три из этих причин формируются внутренне; четвертая, вывод RESET#, – управляемый внешне.

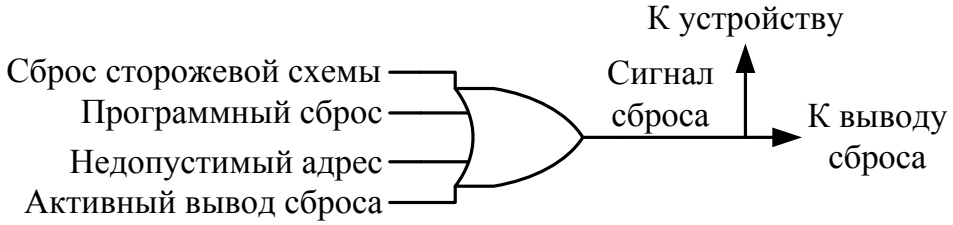


Рисунок 165 – Сигналы сброса

Четыре возможных сигнала сброса формируются следующим образом:

– Сброс сторожевого таймера. Сформированный сторожевым таймером сброс происходит, если сторожевой таймер переполняется и неверное значение записано либо в регистр ключа сторожевого таймера, либо в управляющий регистр сторожевого таймера. Следует отметить, что при включенном устройстве сторожевой таймер автоматически становится активным.

– Программно формируемый сброс. Он осуществляется системным управляющим регистром (SCR). Очистка бита RESET0 (бит 14) или установка бита RESET1 (бит 15) вызывает системный сброс.

– Недопустимый адрес. Карта адресов управляющих регистров системных и периферийных модулей содержит невыполнимые позиции адресов в диапазонах, отмеченных как зарезервировано. Любой доступ к адресам, расположенным в диапазонах «зарезервировано», будет формировать сброс недопустимого адреса.

– Активный вывод сброса. Для формирования внешнего импульса сброса на выводе RESET# обычно используются импульсы малой длительности (до нескольких наносекунд); однако импульсы одного периода SYSCLK требуются для обеспечения того, что устройство распознает сигнал сброса. Обычная цепь сброса, требуемая для устройства 1867ВЦ10Т, состоит из 10 кОм повышающего резистора от вывода RESET# до напряжения питания.

Как только активирован источник сброса, внешний вывод RESET# переходит в низкий (активный) уровень, как минимум на восемь периодов SYSCLK. Это позволяет ИС 1867ВЦ10Т сбрасывать внешние устройства, соединенные с выводом RESET#. (Вывод RESET# является вводом - выводом с открытым стоком и должен иметь присоединенный повышающий резистор.) К тому же, если вывод RESET# удерживается в низком уровне, логика сброса удерживает устройство в состоянии сброса на все время, пока вывод RESET# удерживается в низком уровне.

Когда получен сигнал сброса, программа определяет источник сброса чтением содержимого системного статусного регистра (SSR). SSR содержит один статусный бит для каждого из четырёх внутренних источников, вызывающих сброс. Во время сброса содержимое ОЗУ остается неизменным и все управляющие биты затрагиваемые сбросом, инициализируются.

Аппаратно-формируемые прерывания

Все шины аппаратных прерываний ядра ЦПОС имеют ранг приоритета от 1 до 10 (1 – наивысший). Когда более чем одно из этих аппаратных прерываний продолжает подтверждаться, то сперва подтверждается прерывание с более высоким рангом. Все остальные подтверждаются после него по порядку. Из этих десяти шин, шесть для маскируемых шин прерываний (INT1–INT6) и одна для немаскируемой шины прерывания (NMI). INT1–INT6 и NMI имеют приоритеты, показанные в таблице 95.

Таблица 95 – Приоритеты маскируемых прерываний на уровне ядра ЦПОС

Приоритет на ядре ЦПОС	Маскируемое прерывание
3	NMI
4	INT1
5	INT2
6	INT3
7	INT4
8	INT5
9	INT6

Входы к этим шинам управляются системным модулем и менеджером событий, как приведено в таблице 96 и показано на рисунке 166.

Таблица 96 – Шины прерываний, управляемые системным модулем и менеджером событий

Шина прерывания	Управляется
INT1	Системным модулем
INT5	
INT6	
NMI	
INT2	Менеджером событий
INT3	
INT4	

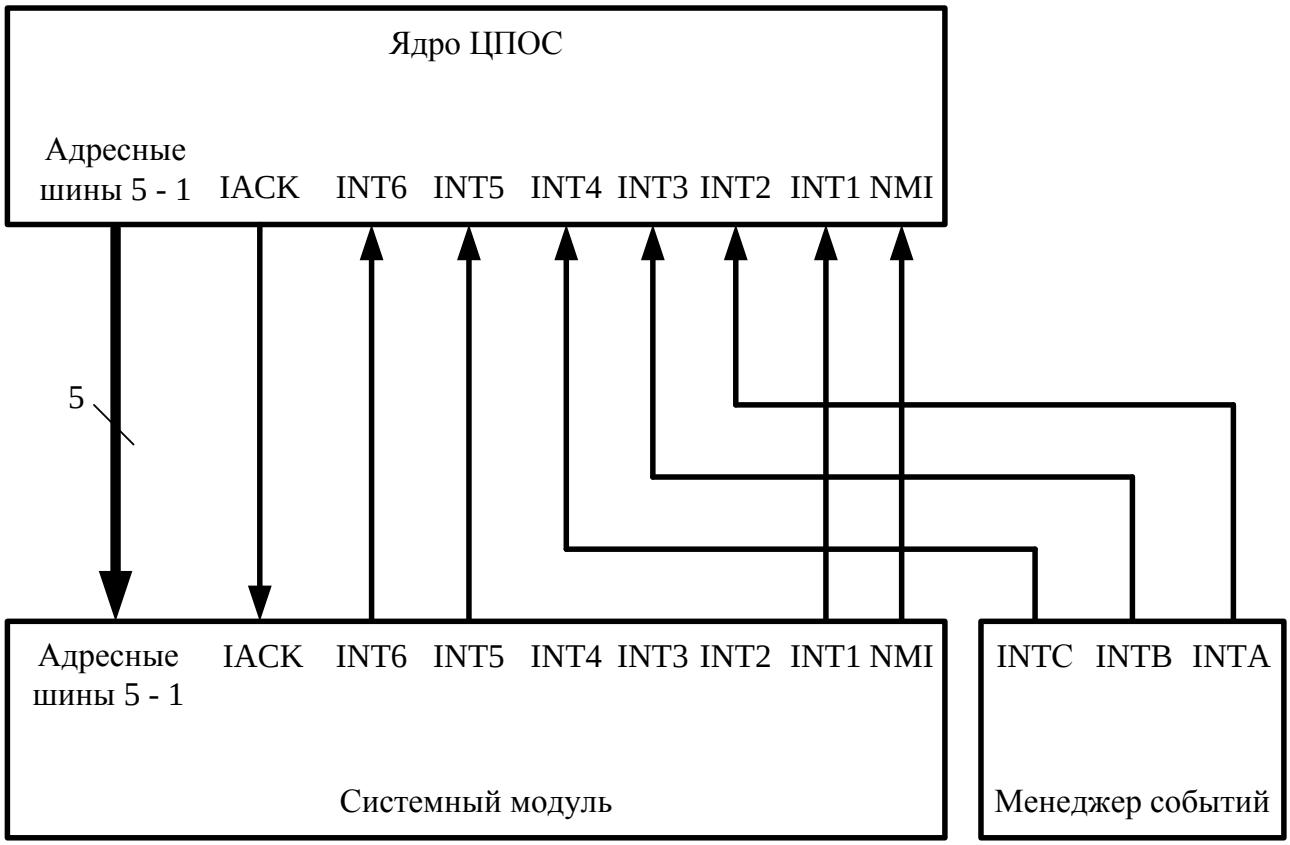


Рисунок 166 – Структура прерываний ЦПОС

На уровне системного модуля и менеджера событий, каждая шина маскируемых прерываний (INT1–INT6) соединена с несколькими источниками маскируемых прерываний. Источники, соединенные в шину прерывания INT1, называются прерываниями уровня 1; источники, соединенные в шину прерывания INT2, называются прерываниями уровня 2; и так далее. Для каждой шины прерываний, несколько источников так же имеют установку ранга приоритета. Ядро ЦПОС отвечает сначала на запрос прерывания от источника с более высоким приоритетом.

На рисунке 167 показаны источники и ранги приоритета для прерываний, управляемых системным модулем. Следует отметить, что для каждой цепочки пре-

рываний, источник прерывания с высшим приоритетом находится наверху. Приоритет уменьшается от верха до низа цепочки. На рисунке 168 показаны источники и ранги приоритета для прерываний управляемых менеджером событий.

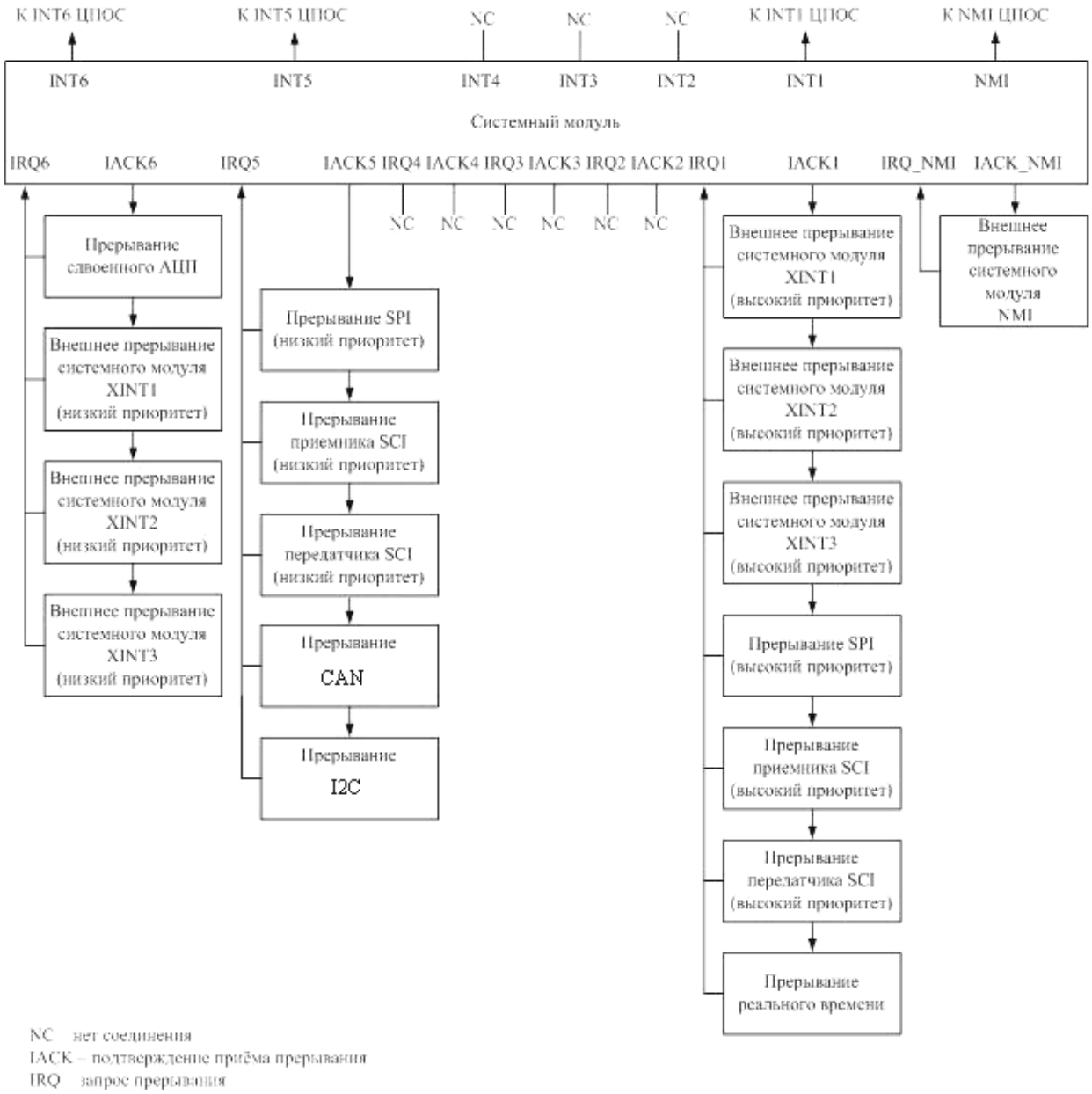


Рисунок 167 – Структура прерываний системного модуля

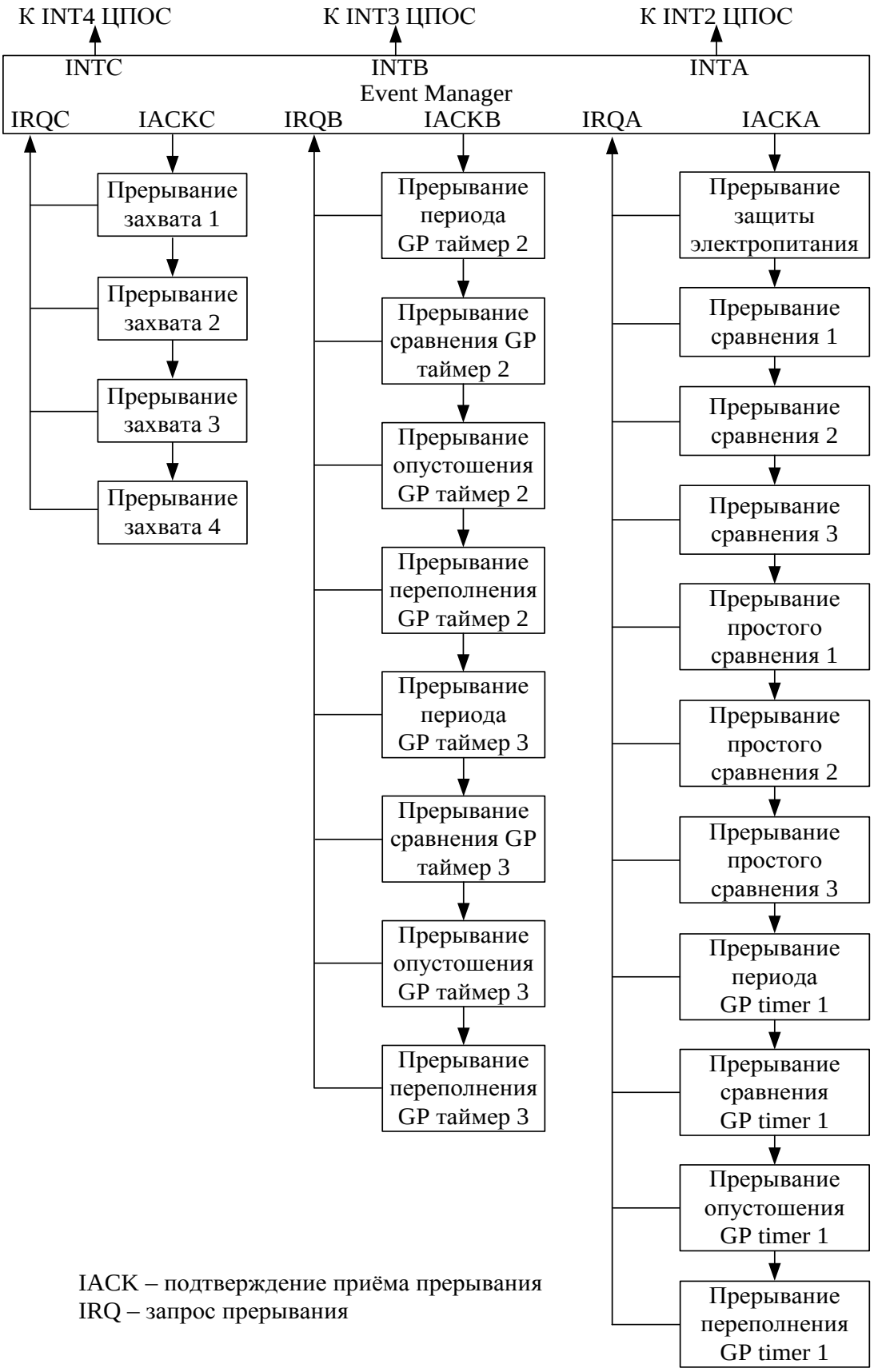


Рисунок 168 – Структура прерываний менеджера событий

Каждый источник прерывания имеет собственный управляющий регистр с битом флага и разрешения. Когда принят запрос прерывания, в соответствующем управляющем регистре устанавливается бит флага. Если так же установлен бит разрешения, сигнал посылается на арбитражную логику, которая может одновременно принимать одинаковые сигналы от ещё одного или более управляющего регистра. Арбитражная логика сравнивает уровень приоритета параллельных запросов прерывания и пропускает прерывание с более высоким приоритетом к процессору. Соответствующий флаг устанавливается в регистре флага прерывания (IFR), указывая, что прерывание выполняется. Затем процессор должен решить нужно ли подтверждать прерывание. Маскируемые аппаратные прерывания подтверждаются только после точного выполнения следующих условий:

– Наивысший приоритет. Когда больше, чем одно аппаратное прерывание запрашивается в одно и то же время, 1867ВЦ10Т обслуживает их в соответствии с установленным рангом приоритета.

– Бит INTM в 0. Бит режима прерывания (INTM), бит 9 в статусном регистре ST0, разрешает или запрещает все маскируемые прерывания:

– Когда INTM = 0, все маскируемые прерывания разрешены.

– Когда INTM = 1, все маскируемые прерывания запрещены.

– INTM устанавливается в 1 автоматически, когда процессор подтверждает прерывание (за исключением запуска командой TRAP) и во время сброса. Он может быть установлен и очищен программно.

– Бит маски IMR в 1. Каждая шина маскируемых прерываний имеет бит маски в регистре маски прерывания (IMR). Для демаскирования шины прерываний нужно установить её бит IMR в 1.

Когда процессор подтверждает маскируемое аппаратное прерывание, он заполняет шину команд командой INTR. Эта команда переводит ПК на соответствующий адрес, с которого процессор выбирает программный вектор. Этот вектор приводит к выполнению обслуживающей программы прерывания.

Обычно, обслуживающая программа прерывания считывает начальный номер вектора адреса периферийного устройства из регистра вектора адреса периферийного устройства (см. таблицу 96) для перехода на код, предназначенный для специального источника прерывания, который вызывает запрос прерывания. ИС 1867ВЦ10Т включает начальный номер вектора фантома прерывания (0000h), который является функцией целостности системного прерывания, которая позволяет выполнять управляемый выход из некорректной последовательности прерываний. Если процессор подтверждает запрос от периферийного устройства, когда от периферийного устройства не поступало запроса прерывания, из регистра вектора прерывания считывается вектор фантома прерывания.

В таблице 97 приведены источники прерываний, общий приоритет, адрес/ начальный номер вектора, источник и функции каждого прерывания, доступного для 1867ВЦ10Т.

Таблица 97 – Размещение и приоритеты прерываний 1867ВЦ10Т

Общий приоритет	Имя прерывания	Прерывание ядра ЦПОС и адрес	Адрес периферийного вектора	Начальный адрес периферийного вектора	Маскируемое	Модуль 1867ВЦ10Т	Функция прерывания
1 (Высший)	RESET	RESET 0000h			нет	Ядро, SD	Внешний системный сброс (RESET)
2	Зарезервир.	INT7 0026h			нет	Ядро ЦПОС	Сист. прер. эмулятора
3	NMI	NMI 0024h		0002h	нет	Ядро, SD	Внешнее прерывание пользователя
4	XINT1	INT1 0002h	SYSIVR	0001h	да	SD	Внешние прерывания пользователя с высоким приоритетом
5	XINT2			0011h	да		
6	XINT3			001Fh	да		
7	SPINT			0005h	да	SPI	Прерывание SPI с высоким приоритетом
8	RXINT			0006h	да	SCI	Прерывание приёмника SCI
9	TXINT			0007h	да	SCI	Прерывание передатчика SCI
10	RTINT	(сист.)	701Eh	0010h	да	WDT	Прерывание реального времени
11	PDPINT		7432h	0020h	да	Внешн.	Прерывание защиты питания
12	CMP1INT	$\overline{\text{INT2}}$ 0004h		0021h	да	модуля EV.CMP1	Прерывание полного сравнения 1
13	CMP2INT			0022h	да	модуля EV.CMP2	Прерывание полного сравнения 2
14	CMP3INT			0023h	да	модуля EV.CMP3	Прерывание полного сравнения 3
15	SCMP1INT			0024h	да	модуля EV.CMP4	Прерывание простого сравнения 1

Продолжение таблицы 97

Общий приоритет	Имя прерывания	Прерывание ядра ЦПОС и адрес	Адрес периферийного вектора	Начальный адрес периферийного вектора	Маскируемое	Модуль 1867ВЦ10Т	Функция прерывания
16	SCMP2INT	____ INT2 0004h		0025h	да	модуля EV.CMP5	Прерывание простого сравнения 2
17	SCMP3INT			0026h	да	модуля EV.CMP6	Прерывание простого сравнения 3
18	TPINT1	Группа А модуля EV		0027h	да	модуля EV.GPT1	Прерывание периода таймера 1
19	TCINT1			0028h	да	модуля EV.GPT1	Прерывание сравнения таймера 1
20	TUFINT1			0029h	да	модуля EV.GPT1	Прерывание опустошения таймера 1
21	TOFINT1			002Ah	да	модуля EV.GPT1	Прерывание переполнения таймера 1
22	TPINT2	INT3 0006h (модуля EV INTB)	7433h	002Bh	да	модуля EV.GPT2	Прерывание периода таймера 2
23	TCINT2			002Ch	да	модуля EV.GPT2	Прерывание сравнения таймера 2
24	TUFINT2			002Dh	да	модуля EV.GPT2	Прерывание опустошения таймера 2
25	TOFINT2			002Eh	да	модуля EV.GPT2	Прерывание переполнения таймера 2
26	TPINT3	Группа В модуля EV		002Fh	да	модуля EV.GPT3	Прерывание периода таймера 3
27	TCINT3			0030h	да	модуля EV.GPT3	Прерывание сравнения таймера 3
28	TUFINT3			0031h	да	модуля EV.GPT3	Прерывание опустошения таймера 3
29	TOFINT3			0032h	да	модуля EV.GPT3	Прерывание переполнения таймера 3

Окончание таблицы 97

Общий приоритет	Имя прерывания	Прерывание ядра ЦПОС и адрес	Адрес периферийного вектора	Начальный адрес периферийного вектора	Маскируемое	Модуль 1867ВЦ10Т	Функция прерывания
30	CAPINT1	INT4 0008h	7434h	0033h	да	модуля EV.CAP1	Прерывание захвата 1
31	CAPINT2			0034h	да	модуля EV.CAP2	Прерывание захвата 2
32	CAPINT3	Группа С модуля EV		0035h	да	модуля EV.CAP3	Прерывание захвата 3
33	CAPINT4			0036h	да	модуля EV.CAP4	Прерывание захвата 4
34	SPINT	INT5 000Ah (сист.)	SYSIVR/ 701Eh	0005h	да	SPI	Прерывание SPI с низким приоритетом
35	RXINT			0006h	да	SCI	Прерывание приёмника SCI
36	TXINT			0007h	да	SCI	Прерывание передатчика SCI
37	CANINT			0008h	да	CAN	Прерывание от CAN
38	I2CINT			0009h	да	I2C	Прерывание От I2C
39	ADCINT	INT6 00Ch (сист.)		0004h	да	модуля АЦП	Прерывание модуля АЦП
40	XINT1			0001h	да	Внешние выводы	Внешние прерывания пользователя с высоким приоритетом
41	XINT2			0011h	да		
42	XINT3			001Fh	да		
43	Зарезервир.	000Eh			да	Ядро ЦПОС	Используется для анализа
нет	TRAP	0022h					Вектор команды TRAP

Внешние прерывания

ИС 1867ВЦ10Т имеет пять внешних прерываний. Эти прерывания включают:

– XINT1. Прерывание типа А. Управляющий регистр XINT1 (7070h) обеспечивает управление и статус для этого прерывания. XINT1 может быть использован как маскируемое прерывание высокого приоритета (уровень 1) или низкого приоритета (уровень 6), или как ввод общего назначения.

– NMI. Прерывание типа A. Управляющий регистр NMI (7072h) обеспечивает управление и статус для этого прерывания. NMI является немаскируемым внешним прерыванием, или вводом общего назначения.

– XINT2. Прерывание типа C. Управляющий регистр XINT2 (7078h) обеспечивает управление и статус для этого прерывания. XINT2 может быть использован как маскируемое прерывание высокого приоритета (уровень 1) или низкого приоритета (уровень 6), или как универсальный ввод-вывод.

– XINT3. Прерывание типа C. Управляющий регистр XINT3 (707Ah) обеспечивает управление и статус для этого прерывания. XINT3 может быть использован как маскируемое прерывание высокого приоритета (уровень 1) или низкого приоритета (уровень 6), или как универсальный ввод-вывод.

– PDPINT. Это прерывание предназначено для безопасной работы силового преобразователя и электропривода. Это маскируемое прерывание может устанавливать таймеры и вывода PWM в высокоипмедансные состояния и информировать процессор о нарушениях в работе электропривода, таких как перенапряжение, перегрузка по току и чрезмерный рост температуры. PDPINT является прерыванием уровня 1. На рисунке 169 показана последовательность запуска после выключения питания.

В таблице 98 показана возможность внешних прерываний 1867ВЦ10Т.

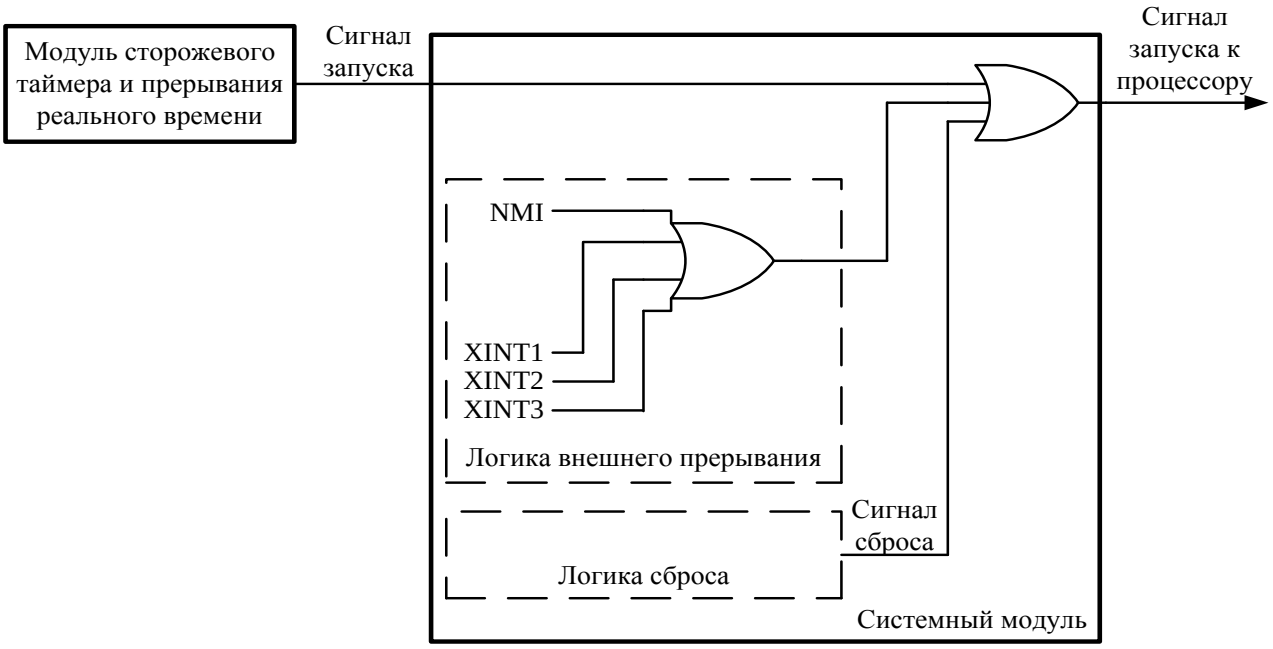


Рисунок 169 – Запуск устройства после выключения питания

Таблица 98 – Типы и функции внешних прерываний

Внешнее прерывание	Адрес управляющего регистра	Тип прерывания	Выполняет NMI	Цифровой ввод-вывод	Маскируемое
XINT1	7070h	A	Нет	Только ввод	Да (уровень 1 или 6)
NMI	7072h	A		Только ввод	нет
N/C	7074h	B	Зарезервировано		

Окончание таблицы 98

Внешнее прерывание	Адрес управляющего регистра	Тип прерывания	Выполняет NMI	Цифровой ввод-вывод	Маскируемое
N/C	7076h	B	Зарезервировано		
XINT2	7078h	C	Нет	Ввод-вывод	Да (уровень 1 или 6)
XINT3	707Ah	C	Нет	Ввод-вывод	Да (уровень 1 или 6)
N/C	707Ch	PM	Зарезервировано		
N/C	707Eh	PM	Зарезервировано		
PDPINT	742Ch				Да (уровень 2)

13.10.6 Формирование тактового сигнала

ИС 1867ВЦ10Т имеет внутрикристальный модуль PLL тактового сигнала. Этот модуль обеспечивает все необходимые тактирующие сигналы для устройства, а также управление элементами режимов пониженного потребления. Единственный внешний компонент, требуемый для этого модуля, это внешний основной опорный кварц.

ИС 1867ВЦ10Т имеет две основных тактовых области: область тактового сигнала процессора (CPUCLK) и область системного тактового сигнала (SYSCLK). Процессор памяти, интерфейс внешней памяти и менеджер событий расположены в области тактового сигнала процессора. Вся остальная периферия находится в области системного тактового сигнала. CPUCLK работает на 2× или 4× частоте SYSCLK; то есть CPUCLK = 20 МГц, SYSCLK = 10 МГц или CPUCLK = 20 МГц, SYSCLK = 5 МГц.

Тактовый модуль включает два внешних вывода:

- BQ1/CLKIN – вход кварца/источник тактового сигнала;
- BQ2 – выход к кварцу.
- OSCBYP#.

Для внешних выводов, если OSCBYP# = 3,3 В, то гетеродин разрешен, и если OSCBYP# = 0 В, то гетеродин шунтирован.

13.10.7 Режим пониженного потребления

ИС 1867ВЦ10Т имеет четыре режима пониженного потребления (IDLE1, IDLE2, генератор со снижением мощности потребления и осциллятор со снижением мощности потребления). Режимы пониженного потребления уменьшают потребляемую мощность за счет уменьшения потребления или остановки работы различных модулей (остановкой их тактовых сигналов). Два бита PLLPM регистра

управления тактовым модулем (CKCR0) выбирают в какой из режимов пониженного потребления входит устройство при выполнении команды IDLE. Сброс или немаскируемое прерывание от любого источника вызывает выход устройства из режима пониженного потребления IDLE 1. Прерывание реального времени (от WD/RTI) вызывает выход устройства из всех (за исключением режима осциллятора со снижением мощности потребления) режимов пониженного потребления (IDLE1, IDLE2, генератор со снижением мощности потребления). Оно является прерыванием запуска.

Сброс или любое из внешних прерываний (NMI, XINT1, XINT2 или XINT3, если они разрешены) вызывают выход устройства из любого режима пониженного потребления (IDLE1, IDLE2, генератор со снижением мощности потребления и осциллятор со снижением мощности потребления). Внешние прерывания являются прерываниями для запуска из этого режима. Любое прерывание, предназначенное для разрешения выхода из режима пониженного потребления, должно быть разрешено индивидуально и глобально для правильного вывода устройства из режима пониженного потребления. Это важно для гарантии того, что выход из режима пониженного потребления разрешен перед входом в этот режим пониженного потребления.

В таблицах 99, 100 представлены режимы пониженного потребления.

Таблица 99 – Режимы пониженного потребления

Режим пониженного потребления	Биты PLLPM (1:0) в CKCR0	Статус тактового сигнала процессора	Статус системного тактового сигнала	Статус WDCLK
X + not IDLE	XX	Вкл.	Вкл.	Вкл.
0 + IDLE	00	Выкл.	Вкл.	Вкл.
1 + IDLE	01	Выкл.	Выкл.	Вкл.
2 + IDLE	10	Выкл	Выкл	Вкл
3 + IDLE	11	Выкл.	Выкл.	Выкл.

Таблица 100 – Режимы пониженного потребления

Режим пониженного потребления	Состояние PLL	Состояние гетеродина	Условие выхода	Питание	Описание
X + not IDLE	Вкл	Вкл	—	240 мА	Обычный режим работы
0 + IDLE	Вкл	Вкл	Любое прерывание, сброс	120мА	Idle1
1 + IDLE	Вкл	Вкл	Прерывание запуска, сброс	20 мА	Idle2
2 + IDLE	Выкл	Вкл	Прерывание запуска, сброс	10 мА	Выключен генератор
3 + IDLE	Выкл	Выкл	Прерывание запуска, сброс	3 мкА	Выключен осциллятор

13.10.8 Программируемые регистры

В таблицу 101 сведены все программируемые регистры ИС 1867ВЦ10Т.

Таблица 101 – Адреса регистров ИС 1867ВЦ10Т

Адрес	Регистр	Имя регистра	Рисунок
Внутр.	ST0	Статусный регистр 0	170
Внутр.	ST1	Статусный регистр 1	171
0004h	IMR	Регистр маски прерывания	172
0006h	IFR	Регистр флага прерывания	173
7018h	SYSCR	Системный управляющий регистр	174
701Ah	SYSSR	Системный статусный регистр	175
702Bh	CKCR0	Управляющий регистр тактового сигнала 0	176
702Dh	CKCR1	Управляющий регистр тактового сигнала 1	177
7032h	ADCTRL1	Управляющий регистр модуля АЦП 1	178
7034h	ADCTRL2	Управляющий регистр модуля АЦП 1	179
7040h	SPICCR	Регистр управления конфигурацией SPI	180
7041h	SPICTL	Регистр управления работы SPI	181
7042h	SPISTS	Статусный регистр SPI	182
7044h	SPIBRR	Регистр скорости двоичной передачи SPI	183
7046h	SPIEMU	Регистр буфера эмуляции SPI	184
7047h	SPIBUF	Последовательный входной буферный регистр SPI	185
7049h	SPIDAT	Регистр данных SPI	186
704Dh	SPIPC1	Управляющий регистр порта SPI 1	187
704Eh	SPIPC2	Управляющий регистр порта SPI 2	188
704Fh	SPIPRI	Регистр приоритета SPI	189
7050h	SCICCR	Управляющий регистр связи SCI	190
7051h	SCICTL1	Управляющий регистр SCI 1	191
7052h	SCIHBAUD	Регистр скорости двоичной передачи старших бит SCI	192
7053h	SCILBAUD	Регистр скорости двоичной передачи младших бит SCI	193
7054h	SCICTL2	Управляющий регистр SCI 2	194
7055h	SCIRXST	Регистр статуса приёмника SCI	195
705Eh	SCIPC2	Управляющий регистр порта SCI 2	196
705Fh	SCIPRI	Управляющий регистр приоритета SCI	197
7070h	XINT1	Управляющий регистр внешнего прерывания	198
7072h	NMI	Управляющий регистр внешнего прерывания	199
7078h	XINT2	Управляющий регистр внешнего прерывания	200
707Ah	XINT3	Управляющий регистр внешнего прерывания	201
7090h	OCRA	Управляющий регистр мультиплексирования ввода-вывода A	202
7092h	OCRB	Управляющий регистр мультиплексирования ввода-вывода B	203
7098h	PADATDIR	Регистр данных и направления порта A ввода-вывода	204

Окончание таблицы 101

Адрес	Регистр	Имя регистра	Рисунок
709Ah	PBDATDIR	Регистр данных и направления порта ввода-вывода	205
709Ch	PCDATDIR	Регистр данных и направления порта С ввода-вывода	206
7400h	GPTCON	Управляющий регистр таймера общего назначения	207
7404h	T1CON	Управляющий регистр GP таймер1	208
7408h	T2CON	Управляющий регистр GP таймер2	209
740Ch	T3CON	Управляющий регистр GP таймер3	210
7411h	COMCON	Управляющий регистр сравнения	211
7413h	ACTR	Операционный управляющий регистр полного сравнения	212
7414h	SACTR	Операционный управляющий регистр простого сравнения	213
7415h	DBTCON	Управляющий регистр таймера «мертвого» времени	214
7420h	CAPCON	Управляющий регистр захвата	215
7422h	CAPFIFO	Статусный регистр захвата FIFO	216
742Ch	EVIMRA	Регистр маски прерывания А	217
742Dh	EVIMRB	Регистр маски прерывания В	218
742Eh	EVIMRC	Регистр маски прерывания С	219
742Fh	EVIFRA	Регистр флага прерывания А	220
7430h	EVIFRB	Регистр флага прерывания В	221
7431h	EVIFRC	Регистр флага прерывания С	222
7432h	EVIVRA	Регистр вектора прерывания А	223
7433h	EVIVRB	Регистр вектора прерывания В	224
7434h	EVIVRC	Регистр вектора прерывания С	225
FFFFh[2)]	WSGR	Управляющий регистр генератора периодов ожидания	226

[1)] Адрес в пространстве памяти программ.
[2)] Адрес в пространстве ввод-вывод.

Регистры процессора

Статусные регистры процессора (ST0 и ST1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARP			OV	OVM	1	INTM	DP								
R/W-x			R/W-0	R/W-x		R/W-1	R/W-x								

Рисунок 170 – Статусный регистр процессора ST0 – внутренний регистр процессора

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARB			CNF	TC	SXM	C	1	1	1	1	XF	1	1	PM	
R/W-x			R/W-0	R/W-x	R/W-1	R/W-1					R/W-1			R/W-00	

Рисунок 171 – Статусный регистр процессора ST1 – внутренний

регистр процессора

Регистры прерывания процессора (IMR и IFR)

15-6	5	4	3	2	1	0
Зарезервировано	INT6	INT5	INT4	INT3	INT2	INT1
0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Рисунок 172 – Регистр маски прерывания (IMR) – адрес 0004h

15-6	5	4	3	2	1	0
Зарезервировано	INT6	INT5	INT4	INT3	INT2	INT1
0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

Рисунок 173 – Регистр флага прерывания (IFR) – адрес 0006h

Системный управляющий регистр (SYSCR)

15	14	13-8	7	6	5-0
RESET1	RESET1	Зарезервировано	CLKSRC1	CLKSRC0	Зарезервировано
R/W-0	R/W-1		R/W-0	R/W-0	

Рисунок 174 – Системный управляющий регистр (SYSCR) – адрес 7018h

Системный статусный регистр (SYSSR)

15	14-13	12	11	10	9	8-6	5	4	3	2-1	0
PORST	Зар.	ILLADR	Зар.	SWRST	WDRST	Зар.	HPO	Зар.	VCCAOR	Зар.	VECRD
R/C-x		R/C-x		R/C-x	R/C-x		R/C-i		R-1		R-0

Рисунок 175 – Системный статусный регистр (SYSSR) – адрес 701Ah

Регистры PLL тактового сигнала

Управляющий регистр тактового сигнала 0 (CKCR0)

7	6	5	4	3	2	1	0
CLKMD(1)	CLKMD(0)	PLLOCK(1)	PLLOCK(0)	PLLPM(1)	PLLPM(0)	ACLKENA	PLLPS
RW-x	RW-x	R-x	R-x	RW-0	RW-0	RW-x	RW-0

Рисунок 176 – Управляющий регистр тактового сигнала 0 (CKCR0) – адрес 702Bh

Управляющий регистр тактового сигнала 1 (CKCR1)

7	6	5	4	3	2	1	0
CKINF(3)	CKINF(2)	CKINF(1)	CKINF(0)	PLLDIV(1)	PLLFB(2)	PLLFB(1)	PLLFB(0)
RW-x	RW-x	RW-x	RW-x	RW-x	RW-x	RW-x	RW-x

Рисунок 177 – Управляющий регистр тактового сигнала 1 (CKCR1) – адрес 702Dh

Регистры сдвоенного 12-битного модуля АЦП

Управляющий регистр модуля АЦП 1 (ADCTRL1)

15	14	13	12	11	10	9	8
Suspend-soft	Suspend-free	ADCIMSTART	ADC1EN	ADC2EN	ADCCONRUN	ADCINTEN	ADCINTFLAG
RW-0	RW-0	RW-0	SRW-0	SRW-0	SRW-0	SRW-0	RW-0

7	6-4	3-1	0
ADCEOC	ADC2CHSEL	ADC1CHSEL	ADCSOC
R-0	SRW-0	SRW-0	SRW-0

Рисунок 178 – Управляющий регистр модуля АЦП 1 (ADCTRL1) – адрес 7032h

Управляющий регистр модуля АЦП 2 (ADCTRL2)

15-11	10	9	8
Зарезервировано	ADCEVSOC	ADCEXTSOC	Зарезервировано
	SRW-0	SRW-0	

7-6	5	4-3	2-0
ADCFIFO1	Зарезервировано	ADCFIFO2	ADCPSCALE
R-0		R-0	SRW-0

Рисунок 179 – Управляющий регистр модуля АЦП 2 (ADCTRL2) – адрес 7034h

Регистры модуля последовательного периферийного интерфейса (SPI)

Регистр управления конфигурацией модуля SPI (SPICCR)

7	6	5-3	2	1	0
SPI SW RESET	CLOCK POLARITY	Зарезервировано	SPI CHAR2	SPI CHAR1	SPI CHAR0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 180 – Регистр управления конфигурацией SPI (SPICCR) – адрес 7040h

Регистр управления работы модуля SPI (SPICTL)

7-5	4	3	2	1	0
Зарезервировано	OVERRUN INT ENA	CLOCK PHASE	MASTER/ SLAVE	TALK	SPI INT ENA
	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 181 – Регистр управления работы SPI (SPICTL) – адрес 7041h

Статусный регистр модуля SPI (SPISTS)

7	6	5-0
RECEIVER OVERRUN	SPI INT FLAG	Зарезервировано
RC-0	R-0	

Рисунок 182 – Статусный регистр SPI (SPISTS) – адрес 7042h

Регистр скорости двоичной передачи модуля SPI (SPIBRR)

7	6	5	4	3	2	1	0
Зарезервировано	SPI BIT RATE 6	SPI BIT RATE 5	SPI BIT RATE 4	SPI BIT RATE 3	SPI BIT RATE 2	SPI BIT RATE 1	SPI BIT RATE 0
	RW-0	RW-0	RW-0	RS-0	RW-0	RW-0	RW-0

Рисунок 183 – Регистр скорости двоичной передачи SPI (SPIBRR) – адрес 7044h

Регистр буфера эмуляции модуля SPI (SPIEMU)

7	6	5	4	3	2	1	0
ERCVD7	ERCVD6	ERCVD5	ERCVD4	ERCVD3	ERCVD2	ERCVD1	ERCVD0
R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x

Рисунок 184 – Регистр буфера эмуляции SPI (SPIEMU) – адрес 7046h

Последовательный входной буферный регистр модуля SPI (SPIBUF)

7	6	5	4	3	2	1	0
RCVD7	RCVD6	RCVD5	RCVD4	RCVD3	RCVD2	RCVD1	RCVD0
R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x

Рисунок 185 – Последовательный входной буферный регистр SPI (SPIBUF) –адрес 7047h

Регистр данных модуля SPI (SPIDAT)

7	6	5	4	3	2	1	0
SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
RW-x	RW-x	RW-x	RW-x	RW-x	RW-x	RW-x	RW-x

Рисунок 186– Регистр данных SPI (SPIDAT) – адрес 7049h

Управляющий регистр порта модуля SPI 1 (SPIPC1)

7	6	5	4	3	2	1	0
SPISTE DATA IN	SPISTE DATA OUT	SPISTE FUNCTION	SPISTE DATA DIR	SPICLK DATA IN	SPICLK DATA OUT	SPICLK FUNCTION	SPICLK DATA DIR
R-x	RW-0	RW-0	RW-0	R-x	RW-0	RW-0	RW-0

Рисунок 187 – Управляющий регистр порта SPI 1 (SPIPC1) – адрес 704Dh

Управляющий регистр порта модуля SPI 2 (SPIPC2)

7	6	5	4	3	2	1	0
SPISIMO DATA IN	SPISIMO DATA OUT	SPISIMO FUNCTION	SPISIMO DATA DIR	SPISOMI DATA IN	SPISOMI DATA OUT	SPISOMI FUNCTION	SPISOMI DATA DIR
R-x	RW-0	RW-0	RW-0	R-x	RW-0	RW-0	RW-0

Рисунок 188– Управляющий регистр порта SPI 2 (SPIPC2) – адрес 704Eh

Регистр приоритета модуля SPI (SPIPRI)

7	6	5	4-0
Зарезервировано	SPI PRIORITY	SCI ESPEN	Зарезервировано
	RW-0	RW-0	

Рисунок 189 – Регистр приоритета SPI (SPIPRI) – адрес 704Fh

Регистры модуля последовательного коммуникационного интерфейса (SCI)

Управляющий регистр связи модуля SCI (SCICCR)

7	6	5	4	3	2	1	0
STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	SCI ENA	ADDR/IDLE MODE	SCI CHAR2	SCI CHAR1	SCI CHAR0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 190 – Управляющий регистр связи SCI (SCICCR) – адрес 7050h

Управляющий регистр модуля SCI 1 (SCICTL1)

7	6	5	4	3	2	1	0
Зарезервировано	RX ERR INT ENA	SW RESET	CLOCK ENA	TXWAKE	SLEEP	TXENA	RXENA
	RW-0	RW-0	RW-0	RS-0	RW-0	RW-0	RW-0

Рисунок 191 – Управляющий регистр SCI 1 (SCICTL1) – адрес 7051h

Регистры скорости двоичной передачи SCI (SCIHBAUD и SCILBAUD)

7	6	5	4	3	2	1	0
BAUD15 (ст. бит)	BAUD14	BAUD13	BAUD12	BAUD11	BAUD10	BAUD9	BAUD8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 192 – Регистр скорости двоичной передачи старших бит SCI (SCIHBAUD) – адрес 7052h

7	6	5	4	3	2	1	0
BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0 (мл. бит)
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 193 – Регистр скорости двоичной передачи младших бит SCI (SCILBAUD) – адрес 7053h

Управляющий регистр модуля SCI 2 (SCICTL2)

7	6	5-2	1	0
TXRDY	TX EMPTY	Зарезервировано	RX/BK INT ENA	TX INT ENA
R-1	R-1		RW-0	RW-0

Рисунок 194 – Управляющий регистр SCI 2 (SCICTL2) – адрес 7054h

Регистр статуса приёмника модуля SCI (SCIRXST)

7	6	5	4	3	2	1	0
RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	Зарезервировано
R-0	R-0	R-0	R-0	R-0	R-0	R-0	

Рисунок 195 – Регистр статуса приёмника SCI (SCIRXST) – адрес 7055h

Управляющий регистр порта модуля SCI 2 (SCIPC2)

7	6	5	4	3	2	1	0
SCITXD DATA IN	SCITXD DATA OUT	SCITXD FUNCTION	SCITXD DATA DIR	SCIRXD DATA IN	SCIRXD DATA OUT	SCIRXD FUNCTION	SCIRXD DATA DIR
RW-0	RW-0	RW-0	RW-0	R-x	RW-0	RW-0	RW-0

Рисунок 196 – Управляющий регистр порта SCI 2 (SCIPC2) – адрес 705Eh

Управляющий регистр приоритета модуля SCI (SCIPRI)

7	6	5	4	3-0
Зарезервировано	SCITX PRIORITY	SCIRX PRIORITY	SCI ESPEN	Зарезервировано
	RW-0	RW-0	RW-0	

Рисунок 197 – Управляющий регистр приоритета SCI (SCIPRI) – адрес 705Fh

Управляющие регистры внешнего прерывания (XINT1, NMI, XINT2 и XINT3)

15	14-7	6	5	4-3	2	1	0
XINTA1 Flag	Зарезервир.	XINTA1 Pin Data	XINTA1 NMI	Зарезервировано	XINTA1 Polarity	XINTA1 Priority	XINTA1 Enable

Рисунок 198 – Управляющий регистр внешнего прерывания (XINT1) – адрес 7070h

15	14-7	6	5	4-3	2	1-0
XINTA-NMI Flag	Зарезервир.	XINTA-NMI Pin Data	XINTA-NMI NMI	Зарезервир.	XINTA-NMI Polarity	Зарезервир.

Рисунок 199 – Управляющий регистр внешнего прерывания (NMI) – адрес 7072h

15	14-7	6	5	4	3	2	1	0
XINTC1 Flag	Зарезервир.	XINTC1 Pin Data	Зарезервир.	XINTC1 Data dir	XINTC1 Data out	XINTC1 Polarity	XINTC1 Priority	XINTC1 Enable

Рисунок 200 – Управляющий регистр внешнего прерывания (XINT2) – адрес 7078h

15	14-7	6	5	4	3	2	1	0
XINTC2 Flag	Зарезервир.	XINTC2 Pin Data	Зарезервир.	XINTC2 Data dir	XINTC2 Data out	XINTC2 Polarity	XINTC2 Priority	XINTC2 Enable

Рисунок 201 – Управляющий регистр внешнего прерывания (XINT3) – адрес 707Ah

Управляющие регистры цифрового ввода - вывода

Управляющие регистры мультиплексирования ввода - вывода (OCRA и OCRB)

15	14	13	12	11	10	9	8
CRA.15	CRA.14	CRA.13	CRA.12	CRA.11	CRA.10	CRA.9	CRA.8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
Зарезервировано				CRA.3	CRA.2	CRA.1	CRA.0
RW-0				RW-0	RW-0	RW-0	RW-0

Рисунок 202 – Управляющий регистр мультиплексирования ввода-вывода A (OCRA) – адрес 7090h

15	14	13	12	11	10	9	8
Зарезервировано							
RW-0							
7	6	5	4	3	2	1	0
CRB.7	CRB.6	CRB.5	CRB.4	CRB.3	CRB.2	CRB.1	CRB.0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 203– Управляющий регистр мультиплексирования ввода-вывода B (OCRB) – адрес 7092h

Регистры данных и направления порта ввода-вывода (PADATDIR, PBDATDIR и PCDATDIR)

15	14	13	12	11	10	9	8
Зарезервировано				A3DIR	A2DIR	A1DIR	A0DIR
				RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
Зарезервировано				IOPA3	IOPA2	IOPA1	IOPA0
				RW-0	RW-0	RW-0	RW-0

Рисунок 204 – Регистр данных и направления порта A ввода-вывода (PADATDIR) – адрес 7098h

15	14	13	12	11	10	9	8
B7DIR	B6DIR	B5DIR	B4DIR	B3DIR	B2DIR	B1DIR	B0DIR
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
IOPB7	IOPB6	IOPB5	IOPB4	IOPB3	IOPB2	IOPB1	IOPB0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 205 – Регистр данных и направления порта B ввода-вывода (PBDATDIR) – адрес 709Ah

15	14	13	12	11	10	9	8
C7DIR	C6DIR	C5DIR	C4DIR	C3DIR	C2DIR	C1DIR	C0DIR
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
IOPC7	IOPC6	IOPC5	IOPC4	IOPC3	IOPC2	IOPC1	IOPC0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 206 – Регистр данных и направления порта C
ввода-вывода (PCDATDIR) – адрес 709Ch

Регистры таймера общего назначения (GP таймера)

Управляющий регистр таймера общего назначения (GPTCON)

15	14	13	12-11	10-9	8-7
T3STAT	T2STAT	T1STAT	T3TOADC	T2TOADC	T1TOADC
R-1	R-1	R-1	RW-0	RW-0	RW-0

6	5-4	3-2	1-0
TCOMPOE	T3PIN	T2PIN	T1PIN
RW-0	RW-0	RW-0	RW-0

Рисунок 207 – Управляющий регистр таймера общего назначения
(GPTCON) – адрес 7400h

Управляющие регистры GP таймера (T1CON, T2CON и T3CON)

15	14	13	12	11	10	9	8
Free	Soft	TMODE2	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
TSWT1	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 208 – Управляющий регистр GP таймера 1 (T1CON) – адрес 7404h

15	14	13	12	11	10	9	8
Free	Soft	TMODE2	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
TSWT1	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 209 – Управляющий регистр GP таймера 2 (T2CON) – адрес 7408h

15	14	13	12	11	10	9	8
Free	Soft	TMODE2	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
TSWT1	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 210 – Управляющий регистр GP таймера 3 (T3CON) – адрес 740Ch

Регистры блоков сравнения

Управляющий регистр сравнения (COMCON)

15	14	13	12	11	10	9	8
CENABLE	CLD1	CLD0	SVENABLE	ACTRDL1	ACTRDL0	FCOMPOE	SCOMPOE
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
SELTMR	SCLD1	SCLD0	SACTRDL1	SACTRDL0	SELCMP3	SELCMP2	SELCMP1
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 211 – Управляющий регистр сравнения (COMCON) – адрес 7411h

Операционный управляющий регистр полного сравнения (ACTR)

15	14	13	12	11	10	9	8
SVRDIR	D2	D1	D0	CMP6ACT1	CMP6ACT0	CMP5ACT1	CMP5ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
CMP4ACT1	CMP4ACT0	CMP3ACT1	CMP3ACT0	CMP2ACT1	CMP2ACT0	CMP1ACT1	CMP1ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 212 – Операционный управляющий регистр
полного сравнения (ACTR) – адрес 7413h

Операционный управляющий регистр простого сравнения (SACTR)

15-8						
Зарезервировано						
7-6	5	4	3	2	1	0
	SCMP3ACT1	SCMP3ACT0	SCMP2ACT1	SCMP2ACT0	SCMP1ACT1	SCMP1ACT0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 213 – Операционный управляющий регистр
простого сравнения (SACTR) – адрес 7414h

Управляющий регистр таймера «мертвого» времени (DBTCON)

15	14	13	12	11	10	9	8
DBT7	DBT6	DBT5	DBT4	DBT3	DBT2	DBT1	DBT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2-0
EDBT3	EDBT2	EDBT1	DBTPS1	DBTPS0	
RW-0	RW-0	RW-0	RW-0	RW-0	

Рисунок 214 – Управляющий регистр таймера «мертвого» времени (DBTCON) – адрес 7415h

Регистры блоков захвата

Управляющий регистр захвата (CAPCON)

15	14-13	12	11	10	9	8
CAPRES	CAPQEPN	CAP3EN	CAP4EN	CAP34TSEL	CAP12TSEL	CAP4TOADC
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7-6	5-4	3-2	1-0
CAP1EDGE	CAP2EDGE	CAP3EDGE	CAP4EDGE
RW-0	RW-0	RW-0	RW-0

Рисунок 215 – Управляющий регистр захвата (CAPCON) – адрес 7420h

Статусный регистр захвата FIFO (CAPFIFO)

15-14	13-12	11-10	9-8
CAP4FIFO	CAP3FIFO	CAP2FIFO	CAP1FIFO
R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
CAPFIFO15	CAPFIFO14	CAPFIFO13	CAPFIFO12	CAPFIFO11	CAPFIFO10	CAPFIFO9	CAPFIFO8
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

Рисунок 216 – Статусный регистр захвата FIFO (CAPFIFO) – адрес 7422h

Регистры прерывания модуля менеджера событий (модуля EV)

Регистры маски прерывания модуля EV (EVIMRA, EVIMRB и EVIMRC)

15-11	10	9	8
Зарезервировано	T1OFINT ENABLE	T1UFINT ENABLE	T1CINT ENABLE
	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
T1PINT ENABLE	SCMP3INT ENABLE	SCMP2INT ENABLE	SCMP1INT ENABLE	CMP3INT ENABLE	CMP2INT ENABLE	CMP1INT ENABLE	PDPINT ENABLE
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 217 – Регистры маски прерывания A (EVIMRA) – адрес 742Ch

15-8	7	6	5	4	3	2	1	0
Зарезервировано	T3OFINT ENABLE	T3UFINT ENABLE	T3CINT ENABLE	T3PINT ENABLE	T2OFINT ENABLE	T2UFINT ENABLE	T2CINT ENABLE	T2PINT ENABLE
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 218 – Регистры маски прерывания B (EVIMRB) – адрес 742Dh

15-4	3	2	1	0
Зарезервировано	CAP4INT ENABLE	CAP3INT ENABLE	CAP2INT ENABLE	CAP1INT ENABLE
	RW-0	RW-0	RW-0	RW-0

Рисунок 219 – Регистры маски прерывания C (EVIMRC) – адрес 742Eh

Регистры флага прерывания модуля EV (EVIFRA, EVIFRB и EVIFRC)

15-11	10	9	8
Зарезервировано	T1OFINT	T1UFINT	T1CINT
	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
T1PINT	SCMP3INT	SCMP2INT	SCMP1INT	CMP3INT	CMP2INT	CMP1INT	PDPINT
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 220 – Регистры флага прерывания A (EVIFRA) – адрес 742Fh

15-8	7	6	5	4	3	2	1	0
Зарезервировано	T3OFINT	T3UFINT	T3CINT	T3PINT	T2OFINT	T2UFINT	T2CINT	T2PINT
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Рисунок 221 – Регистры флага прерывания B (EVIFRB) – адрес 7430h

15-4	3	2	1	0
Зарезервировано	CAP4INT	CAP3INT	CAP2INT	CAP1INT
	RW-0	RW-0	RW-0	RW-0

Рисунок 222 – Регистры флага прерывания C (EVIFRC) – адрес 7431h

Регистры вектора прерывания модуля EV (EVIVRA, EVIVRB и EVIVRC)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Рисунок 223 – Регистры вектора прерывания A (EVIVRA) – адрес 7432h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Рисунок 224 – Регистры вектора прерывания B EVIVRB) – адрес 7433h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Рисунок 225 – Регистры вектора прерывания C (EVIVRC) – адрес 7434h

Управляющий регистр генератора периодов ожидания (WSGR)

15-4	3	2	1	0
Зарезервировано	AVIS	ISWS	DSWS	PSWS
0	W-1	W-1	W-1	W-1

Рисунок 226 – Управляющий регистр генератора периодов ожидания (WSGR) – адрес в пространстве ввод - вывод FFFFh

14 Описание центрального процессора и системы команд

14.1 Обзор раздела

В данном разделе «Описание центрального процессора и системы команд» содержится описание:

- архитектуры контроллера с процессором цифровой обработки сигналов;
- центрального процессора (ЦП);
- системного оборудования;
- инструкций ассемблера и основных операций ИС 1867ВЦ10Т.

14.1.1 Параметры микросхемы

Основные характеристики ИС:

- Ядро процессора:

– 32-битное центральное арифметическое устройство (CALU);

– 32-битный аккумулятор;

– 16-битный параллельный умножитель с 32-битным произведением;

– 3 масштабирущих сдвигателя;

– 8 вспомогательных регистров с арифметическим устройством (ARAU) для осуществления косвенной адресации памяти данных.

– Информация о памяти представлена в разделе 13.10.2.

– Программное управление:

– четырехуровневый конвейер;

– восьмиуровневый стек;

– шесть внешних прерываний (прерывание при нарушении питания, прерывание по сбросу, немаскируемое прерывание NMI и три маскируемых прерывания).

– Система команд:

– исходный код совместим с ИС М1867ВМ1, 1867ВМ2, 1867ВЦ2Т;

– операция повтора однократной инструкции;

– одноцикловые инструкции умножения с накоплением;

– инструкции перемещения блоков для управления памятью программ/данных;

– индексная адресация;

– бит-реверсивная адресация для БПФ с основанием 2.

– Питание:

– статическая КМОП технология;

– четыре режима пониженного энергопотребления.

– Внутрикристальная логика эмуляции с тестовым интерфейсом, соответствующим стандарту IEEE Standard 1149.1.

– 8,33 нс цикл инструкций (120 MIPS) для большинства одноцикловых инструкций.

– Модуль менеджера событий:

– 12 канальная (9 независимых каналов) широтно-импульсная модуляция (PWM) с компарацией;

– три 16-битных таймера общего назначения с шестью режимами работы;

– три полных 16-битных устройства сравнения с возможностью задания «мертвого времени»;

– четыре устройства захвата, два из которых имеют интерфейс «квадратурной» обработки сигналов импульсного датчика положения (quadrature encoder-pulse interface).

– Два 12-битных аналого-цифровых преобразователя.

– Двадцать восемь индивидуально программируемых мультиплексированных входных/выходных выводов.

– Модуль ФАПЧ (PLL).

– Модуль сторожевого таймера с прерыванием в реальном времени (WDTI).

– Модуль последовательного коммуникационного интерфейса (SCI).

– Модуль последовательного периферийного интерфейса (SPI).

– CAN-модуль.

– I2C-модуль.

14.2 Обзор архитектуры

В этом разделе описываются структура и компоненты ИС 1867ВЦ10Т. ИС 1867ВЦ10Т использует улучшенную, модифицированную Гарвардскую архитектуру, которая увеличивает производительность путем разделения шинных структур для программной памяти и памяти данных.

14.2.1 Архитектура

Высокоуровневая блок-схема ИС 1867ВЦ10Т показана на рисунке 227. ИС 1867ВЦ10Т основана на модифицированной Гарвардской архитектуре, которая поддерживает шинную структуру с отдельным пространством для программ и данных. Третья область – это область ввода-вывода, доступ к которой возможен через внешний интерфейс доступа. Для поддержки внутрикристальной периферии используется периферийная шина. Так, например, все инструкции, оперирующие с пространством данных, также оперируют и со всеми периферийными регистрами.

Разделение областей для программ и данных разрешает одновременный доступ к инструкции и данным в одном цикле. Для примера, пока данные умножаются, предыдущее произведение может быть добавлено к содержимому аккумулятора, и, в то же самое время, может быть сгенерирован новый адрес. Такой параллелизм обеспечивает выполнение в одном машинном цикле набора арифметических, логических инструкций и операций с битами. ИС 1867ВЦ10Т также включает управляющие механизмы для обслуживания прерываний, операций повтора и вызовов функций и/или подпрограмм.

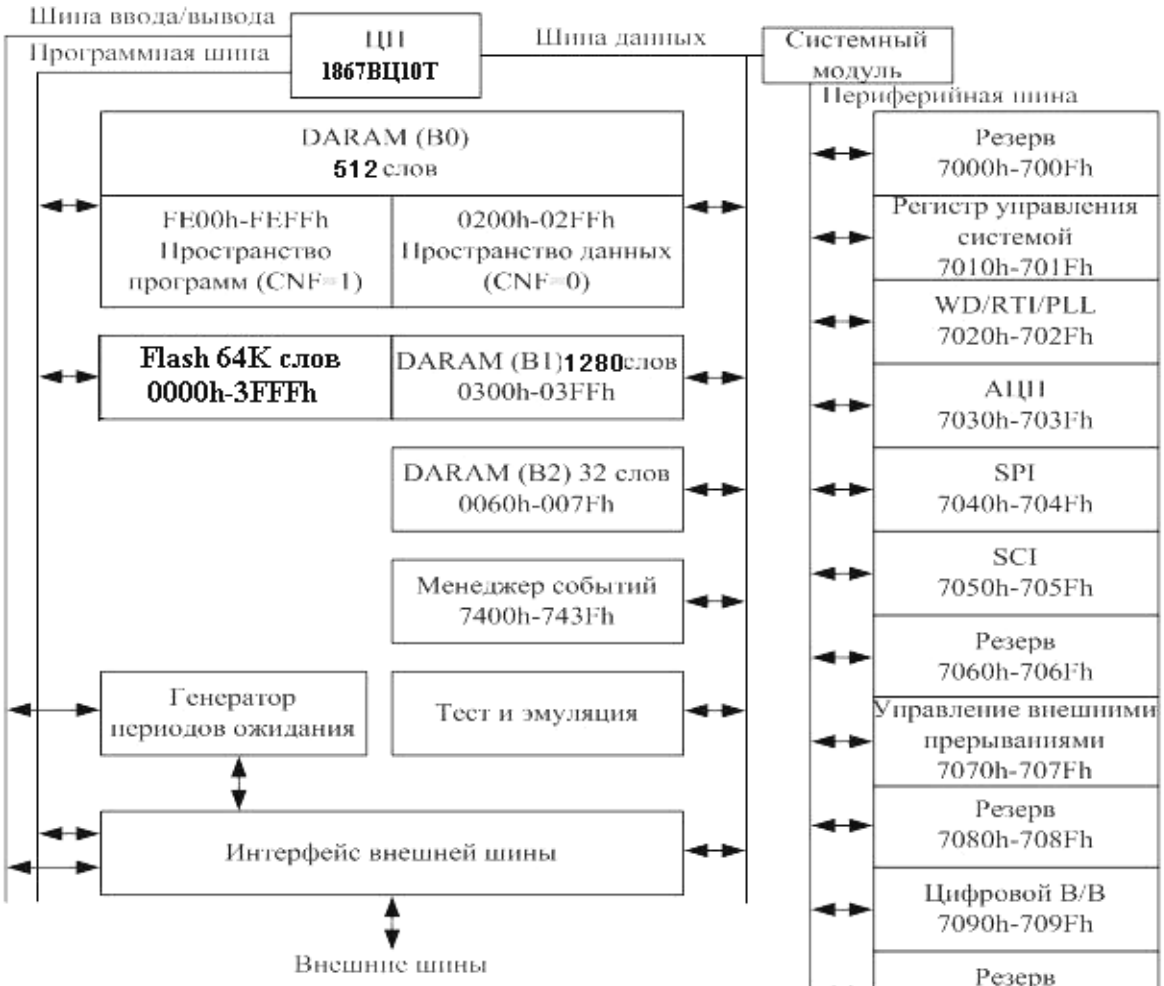


Рисунок 227 – Высокоуровневая блок-схема ИС 1867ВЦ10Т

Структура внутренних шин программ и данных состоит из шести 16-битных шин (см. рисунок 228).

PAB – адресная шина программ, которая обеспечивает адресацию при чтении и записи в программную память.

DRAB – адресная шина чтения данных. Обеспечивает адресацию чтения памяти данных.

DWAB – адресная шина записи данных. Обеспечивает адресацию записи в память данных.

PRDB – шина чтения из программной памяти кодов инструкций и непосредственных операндов.

DRDB – шина чтения памяти данных. Обеспечивает подачу операндов из памяти данных в центральное арифметическое устройство (CALU) и в арифметическое устройство вспомогательных регистров (ARAU).

DWEB – шина записи данных. Предназначена для записи данных в память программ и память данных.

Наличие разделенных адресных шин для чтения данных (DRAB) и записи данных (DWAB) позволяет центральному процессору (ЦП) читать и писать в одном машинном цикле.

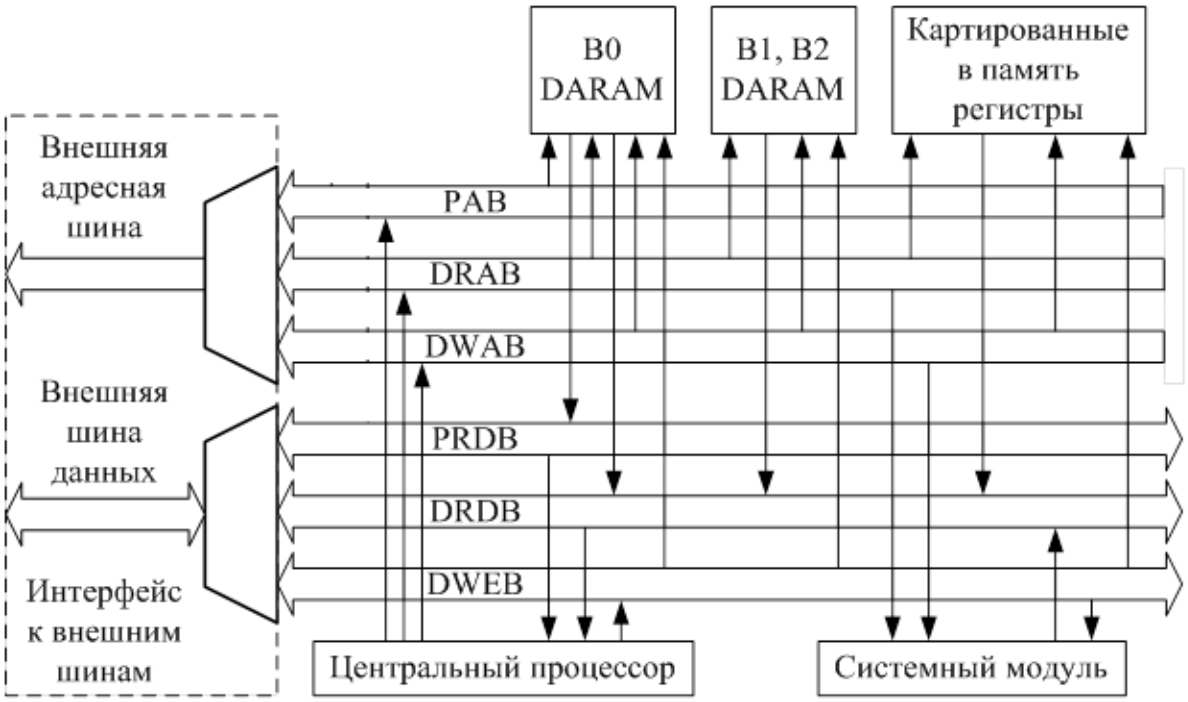


Рисунок 228 – Шинная структура ИС 1867ВЦ10Т

14.2.2 Память

ИС 1867ВЦ10Т реализует три отдельных адресных пространства: адресное пространство программной памяти, памяти данных и I/O (ввода-вывода).

ИС 1867ВЦ10Т может адресовать память программ:

- 64К слов внутрикристальной Flash памяти при MP/MC=0;
- 64К слов внешней памяти при MP/MC=1;
- 0.5К слов внутрикристальной DARAM при CNF=1.

ИС 1867ВЦ10Т может адресовать память данных:

- 45К слов внутрикристальной SRAM;
- 1824 с лов ×16 бит внутрикристальной DARAM при CNF=0;
- 1312 слов ×16 бит внутрикристальной DARAM при CNF=1.

ИС 1867ВЦ10Т может адресовать память пространства ввода-вывода:

- 64К слов внешней памяти.

14.2.3 Центральный процессор

ЦП микросхемы 1867ВЦ10Т содержит:

– 32-битное центральное арифметико-логическое устройство (CALU);

– 32-битный аккумулятор;

– сдвигатель для масштабирования на входе/выходе CALU;

– 16 бит × 16 бит аппаратный умножитель;

– сдвигатель для масштабирования произведения;

– логику генерации адреса данных, которая включает восемь вспомогательных регистров и арифметическое устройство вспомогательных регистров (ARAU);

– логику генерации программного адреса.

Центральное арифметическое устройство (CALU) и аккумулятор

32-битное CALU микросхемы 1867ВЦ10Т выполняет арифметические действия над двоичными операндами, представленными в коде с дополнением до двух. CALU оперирует с 16-битными словами, взятыми из памяти данных или извлеченными из кода инструкций с непосредственной адресацией, а также с 32-битным результатом с выхода умножителя. В дополнение к арифметическим операциям, CALU может выполнять и логические операции.

Выход CALU подключен к входу аккумулятора для запоминания результата; в свою очередь, данные из аккумулятора могут подаваться на один из входов CALU. Аккумулятор имеет ширину 32 бита и разделен на старшее слово (биты 31-16) и младшее слово (биты 15-0). В системе команд предусмотрены инструкции, позволяющие сохранять в памяти как старшее, так и младшее слова аккумулятора.

Масштабирующий сдвигатель

ИС 1867ВЦ10Т содержит три 32-битных сдвигателя, обеспечивающих операции масштабирования, выделения (извлечения) отдельных разрядов, арифметики с повышенной точностью и предохранения от переполнения:

– Входной масштабирующий сдвигатель данных. Имеет 16-разрядный вход и 32-разрядный выход; обеспечивает сдвиг входных данных влево на величину от 0 до 16 бит, чтобы выровнять 16-битные данные для 32-битного входа CALU.

– Выходной масштабирующий сдвигатель данных. Используется при сохранении данных с выхода аккумулятора в памяти, обеспечивая сдвиг влево на величину от 0 до 7 бит. Содержимое аккумулятора при этом не изменяется.

– Сдвигатель для масштабирования произведения. Регистр произведения (PREG) хранит данные с выхода умножителя. Сдвигатель, масштабирующий произведение, сдвигает данные с выхода PREG перед тем, как они будут поданы на вход CALU. Сдвигатель произведения имеет четыре сдвиговых режима (нет сдвига, левый сдвиг на один бит, левый сдвиг на четыре бита и правый сдвиг на шесть бит), которые используются для выполнения операций умножения с накоплением, выполнения дробной арифметики или для выравнивания дробных произведений.

Умножитель

Внутрикристальный умножитель выполняет умножение 16-битных данных, представленных в коде с дополнением до двух, для получения 32-битного результата. ИС 1867ВЦ10Т содержит 16-битный временный регистр (TREG) и 32-битный регистр произведения (PREG), обеспечивает подачу одного из сомножителей на вход умножителя, сохраняет результат каждого умножения.

Благодаря наличию аппаратного умножителя TREG и PREG, ИС 1867ВЦ10Т эффективно выполняет фундаментальные операции цифровой обработки сигналов, такие как свертка (convolution), корреляция и фильтрация. Эффективное время выполнения каждой инструкции умножения занимает один машинный цикл.

Арифметическое устройство вспомогательных регистров (ARAU) и вспомогательные регистры

ARAU генерирует адрес памяти данных в инструкциях доступа к памяти, использующих косвенную адресацию. ARAU оперирует с восемью вспомогательными регистрами (AR0 до AR7), каждый из которых может быть загружен 16-битным значением из памяти данных или непосредственно из инструкции. Значение каждого регистра может быть сохранено в памяти данных. Вспомогательные регистры адресуются 3-битным указателем (ARP – auxiliary register pointer), включенным в статусный регистр ST0.

14.2.4 Программное управление

Несколько аппаратных и программных механизмов обеспечивают программное управление:

– Логика программного управления декодирует инструкцию, управляет четырехуровневым конвейером, сохраняет статус операций и декодирует условные операции. Аппаратные элементы программного управления включают: программный счетчик (PC), статусные регистры, стек и логику генерации адреса следующей инструкции.

– Программные механизмы, используемые для управления последовательностью выполнения программы, включают переходы, вызовы функции и/или подпрограммы, инструкции повторения, прерывания и режимы пониженного энергопотребления.

14.2.5 Внутрикристальная периферия

Как показано на рисунке 228, структура шин ИС 1867ВЦ10Т обеспечивает доступ к многочисленным периферийным устройствам.

Два типа шинных интерфейсов используются для внутрикристальной периферии. Доступ к большинству из периферийных устройств осуществляется через периферийную шину. Эта шина картрируется в пространстве данных через модуль системного управления. Каждый доступ к одному из этих периферийных устройств требует более, чем один машинный цикл. Однако, менеджер событий, установленный прямо на шине данных, дает возможность использовать высокую скорость ЦП. Доступ к менеджеру событий осуществляется с нулевым циклом ожидания. Чтение осуществляется за один цикл и запись за два цикла. Адресация

регистров периферийных устройств фиксирована. Более детально описание периферийных устройств приведено далее в этом разделе.

14.2.6 Эмулятор ВЦ10Т

ИС 1867ВЦ10Т имеет семь выводов для последовательного сканирующего эмуляционного порта (JTAG порт). Этот порт позволяет выполнять эмуляцию без нарушения работы ИС 1867ВЦ10Т и поддерживается средствами эмуляции и отладки, такими как Эмулятор-ВЦ10Т. В приложении Б приведены правила использования эмулятора ВЦ10Т.

14.3 Центральное процессорное устройство

В подразделе описываются операции центрального процессорного устройства ИС 1867ВЦ10Т (ЦП). ЦП может выполнять высокоскоростные арифметические операции внутри одного цикла инструкции за счет его параллельной архитектуры. На рисунке 229 представлена основная секция ЦП.

Также описано арифметическое устройство вспомогательных регистров (ARAU), которое выполняет арифметические операции независимо от центрального арифметического устройства.

Подраздел также включает описание статусных регистров ST0 и ST1, которые содержат биты для задания определенных режимов работы процессора, значения адресного указателя и индицируют различные процессорные состояния и результаты выполнения арифметических и логических операций.

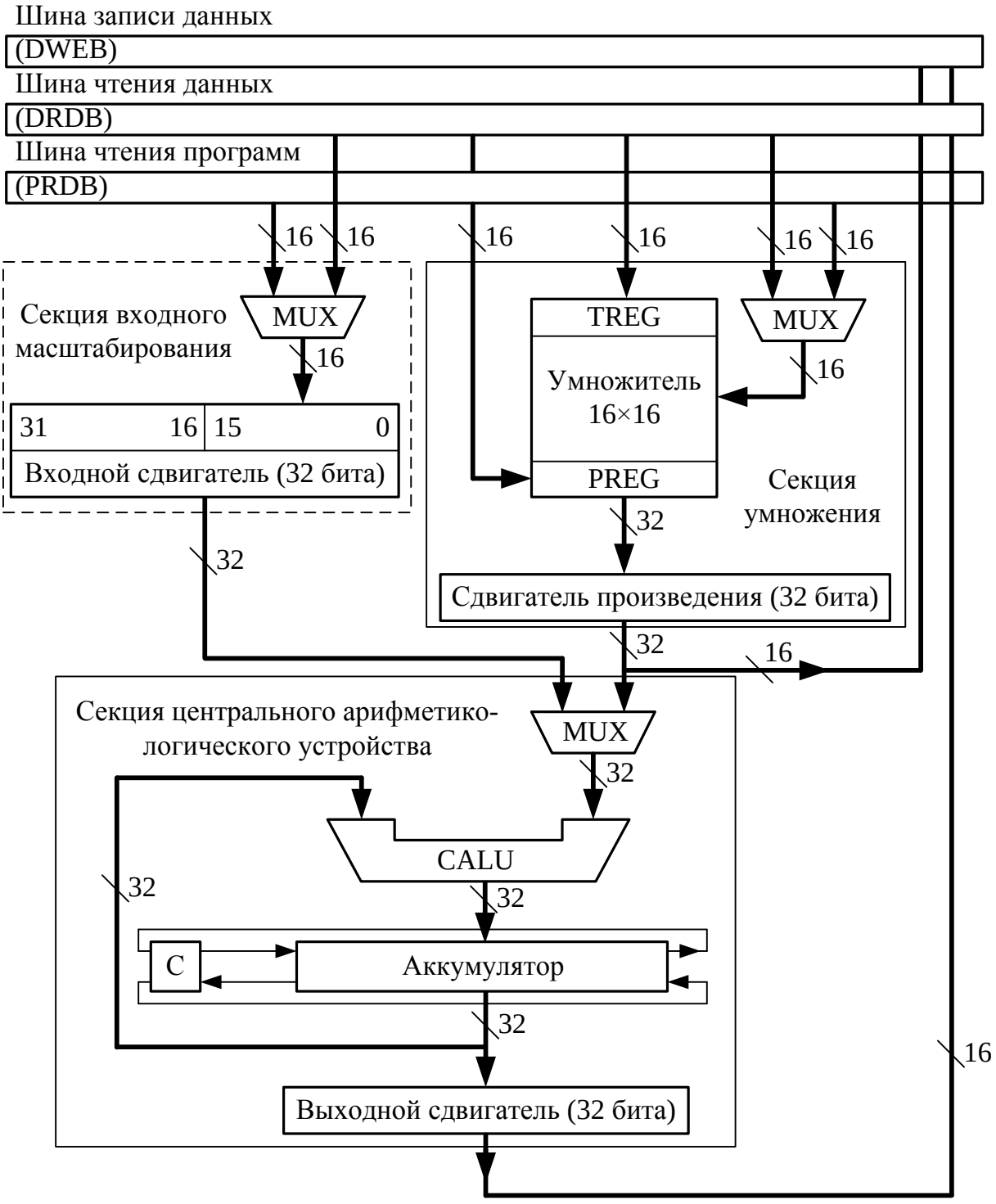


Рисунок 229 – Основная секция ЦП

14.3.1 Вход секции масштабирования

32-битное масштабирующее устройство сдвига (сдвигатель) данных (входной сдвигатель) выравнивает 16-битное значение, которое приходит из памяти в центральное арифметическое устройство (CALU). Это выравнивание данных необходимо как для арифметики масштабирования данных, так и для выравнива-

ния маски для логических операций. Входной сдвигатель является составной частью тракта переданных данных между областями программ или данных и CALU и не требует дополнительного цикла. На рисунке 230 показаны вход, выход и счетчик сдвигов входного сдвигателя.

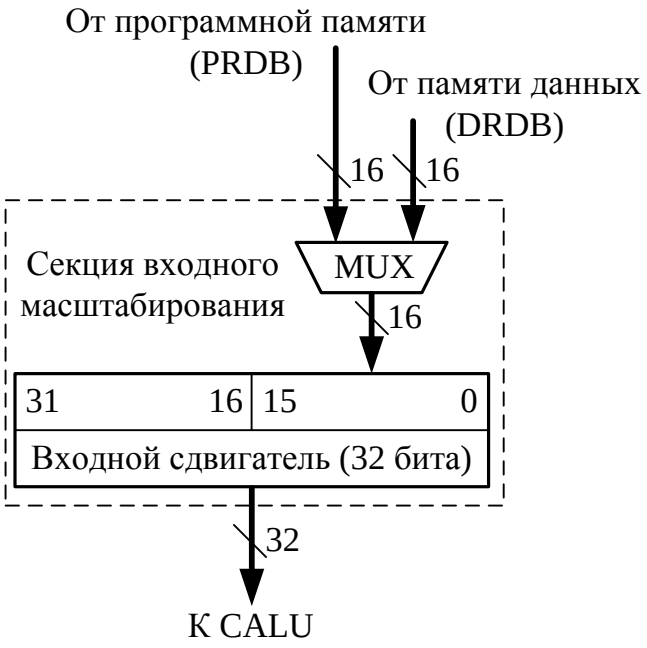


Рисунок 230 – Блок-схема секции масштабирования входа

Вход

Биты с 15 по 0 входного сдвигателя принимают 16-битные данные как от одного, так и от другого источника (см. рисунок 230):

– Шина чтения данных (DRDB). По этому входу поступает значение из ячейки памяти данных, адресуемой операндом инструкции.

– Шина чтения данных из области программы (PRDB). На этот вход поступает значение, заданное в операнде инструкции.

Выход

После того как значение бит 15 – 0 получено, входной сдвигатель выравнивает 16-битное значение до 32-битной шины CALU, как показано на рисунке 230. Сдвигатель сдвигает значение влево от 0 до 16 бит и посылает 32-битный результат в CALU.

Во время левого сдвига неиспользуемые младшие биты 16-битного слова (LSB) заполняются 0, а неиспользованные старшие биты 16-битного слова в сдвигателе или заполняются 0 или расширением знака, в зависимости от бита расширения знака (SXM – the sign-extension mode) статусного регистра ST1.

Счетчик сдвига

Сдвигатель может сдвигать 16-битное значение на величину от 0 до 16 бит. Коэффициент сдвига может быть получен от одного из двух источников:

– из константы, содержащейся в поле слова инструкции. Это позволяет использовать специфические операции масштабирования данных или выравнивания, настроенные для конкретного программного кода;

– из четырех младших бит временного регистра TREG. Сдвиг, основанный на TREG, позволяет задавать коэффициент сдвига динамически, давая возможность адаптации к характеристикам системы.

Для многих, но не всех инструкций, значение SXM бита (бит 10 статусного регистра ST1) определяет, используется ли в CALU расширение знака (знаковое расширение) во время вычислений. Если SXM = 0, тогда знаковое расширение подавляется. Если SXM = 1, тогда выходные данные входного сдвигателя расширяются на знак. На рисунке 231 показан пример сдвига входного значения влево на 8 бит для SXM = 0. В этом примере старшие биты значения, переданные CALU, заполняются 0. На рисунке 232 показан тот же пример, но с SXM = 1. На вход CALU подается значение с расширенным во время сдвига знаком.

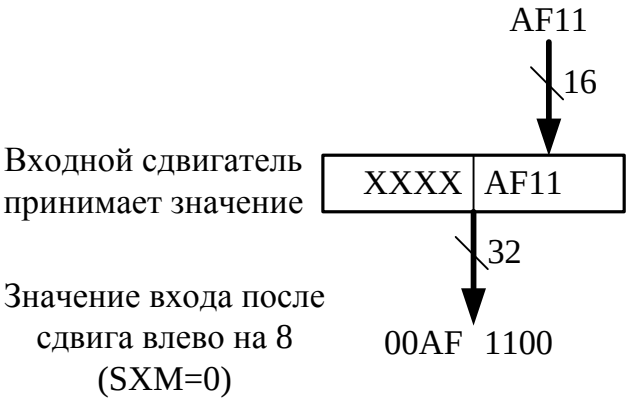


Рисунок 231 – Работа входного сдвигателя для SXM = 0

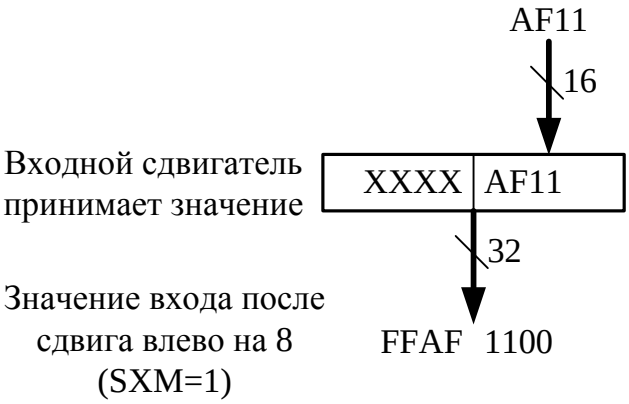


Рисунок 232 – Работа входного сдвигателя для SXM = 1

14.3.2 Секция умножителя

ИС 1867ВЦ10Т использует 16-битный аппаратный умножитель, который может получать как знаковое, так и беззнаковое 32-разрядное произведение в одном машинном цикле. Как показано на рисунке 233, секция умножителя содержит:

– 16-битный временный регистр TREG, который хранит один из сомножителей;

– 32-битный регистр произведения (PREG), в который записывается результат умножения;

– устройство сдвига произведения, масштабирующее значение PREG перед передачей его в CALU.

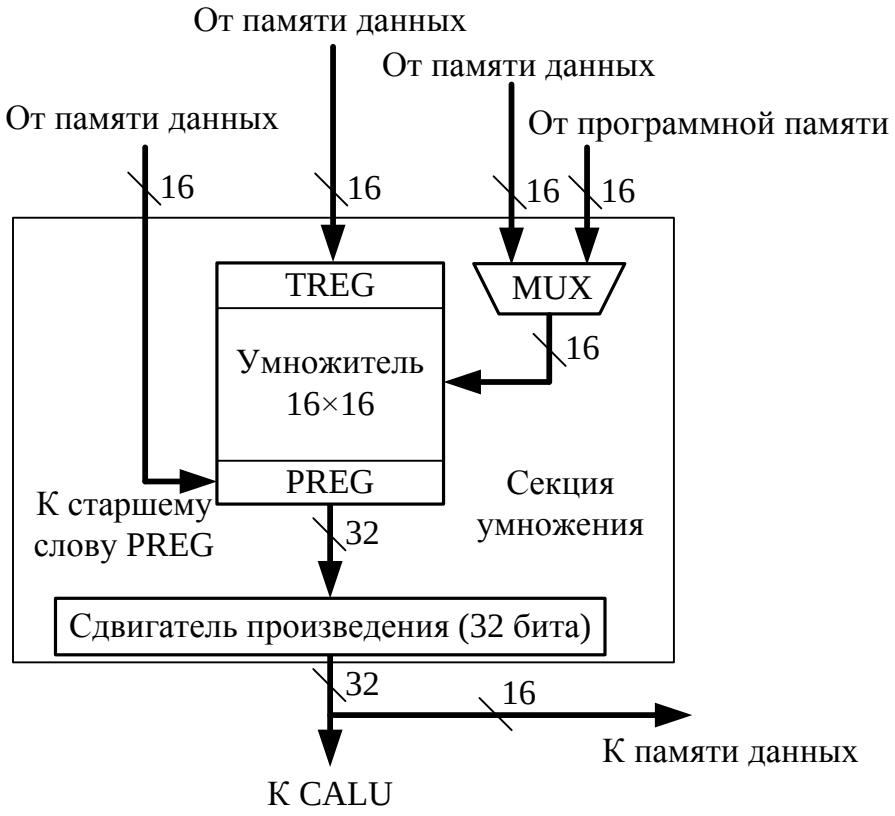


Рисунок 233 – Блок-схема секции умножителя

Умножитель

16-битный аппаратный умножитель может осуществлять как знаковое, так и беззнаковое произведение в одном машинном цикле. Два перемножаемых числа обрабатываются как двоичные числа, представленные в коде с дополнением до 2, исключая беззнаковое умножение (инструкция MPYU). Описание входов и выходов представлено ниже.

Входы

Умножитель имеет два 16-разрядных входа.

– Первый вход всегда соединен с временным регистром TREG. Значение в регистр TREG загружается с шины чтения данных (DRDB) перед умножением.

– На второй вход может быть подано:

– значение из памяти данных с шины чтения данных (DRDB);

– значение из памяти программ с шины чтения программ (DRDB).

Выход

После умножения двух 16-разрядных входных данных 32-разрядный результат сохраняется в регистре произведения (PREG). Выход PREG соединен с 32-битным сдвигателем произведения. Через этот сдвигатель произведение может быть передано из PREG в CALU или в память данных (инструкциями SPH и SPL).

Масштабирующий сдвигатель произведения

Масштабирующий сдвигатель произведения выполняет масштабирование значения регистра PREG. Сдвигатель имеет 32-битный вход, подключенный к выходу PREG, и 32-битный выход, подключенный к входу CALU.

Вход

Сдвигатель данных имеет 32-битный вход, подключенный к выходу PREG.

Выход

После выполнения сдвига 32-битный результат с выхода сдвигателя может быть подан на вход CALU или сохранен в память данных (16-битным словом – старшим или младшим).

Режимы сдвига

Сдвигатель использует один из четырех вариантов сдвига произведения. В таблице 102 представлены различные варианты работы сдвигателя произведения в зависимости от значения разрядов PM в статусном регистре ST1. В первом варианте (PM = 00) – сдвигатель не сдвигает произведение перед передачей его в CALU или память данных. В двух следующих вариантах выполняется левый сдвиг (на один или четыре бита). Эти варианты используются при реализации дробной арифметики или для выравнивания произведения. В варианте с правым сдвигом произведение сдвигается на 6 бит, позволяя, тем самым, выполнять до 128 последовательных умножений с накоплением без переполнения аккумулятора. Необходимо отметить, что сдвинутое значение присутствует на выходе сдвигателя произведения, а содержимое PREG (до выполнения следующего умножения) при этом остается неизмененным.

Примечание – Правый сдвиг в сдвигателе произведения всегда выполняется со знаковым расширением, независимо от значения бита режима распространения знака (SXM) статусного регистра ST1.

Таблица 102 – Четыре варианта сдвига произведения

PM	Сдвиг	Описание
00	Нет сдвига	Произведение передается в CALU или на шину данных DWEB без сдвига
01	Левый сдвиг на 1	Удаляет знаковый бит, сгенерированный при умножении с дополнением до 2, для получения произведения Q_{31}*
10	Левый сдвиг на 4	Удаляет четыре знаковых бита, сгенерированные при умножении 16-бит на 13-бит с дополнением до 2, для получения произведения в формате Q_{31}*. (Для выполнения инструкции MPY #k — умножение на 13-битную константу)
11	Правый сдвиг на 6	Масштабирует произведение, позволяя выполнить до 128 последовательных умножений с накоплением без переполнения аккумулятора. При правом сдиге всегда выполняется знаковое расширение, независимо от состояния бита SXM статусного регистра ST1

*Формат Q_{31} – это двоичная дробь, в которой имеется 31 цифра вправо от двоичной точки (основание 2 – это эквивалент десятичной точки с основанием 10).

14.3.3 Секция центрального арифметического устройства

На рисунке 234 показаны основные компоненты секции центрального арифметического устройства, которая состоит из:

– центрального арифметическо-логического устройства (CALU), которое реализует широкий спектр арифметических и логических функций;

– 32-битного аккумулятора (ACC), в который поступают данные с выхода CALU. Аккумулятор способен выполнять побитный сдвиг своего содержимого с использованием бита переноса C (Carry). На рисунке 234 показаны старшее (ACCH) и младшее (ACCL) слова аккумулятора;

– выходного сдвигателя, который может сдвигать копию старшего или младшего слов аккумулятора перед их пересылкой для сохранения в памяти данных.

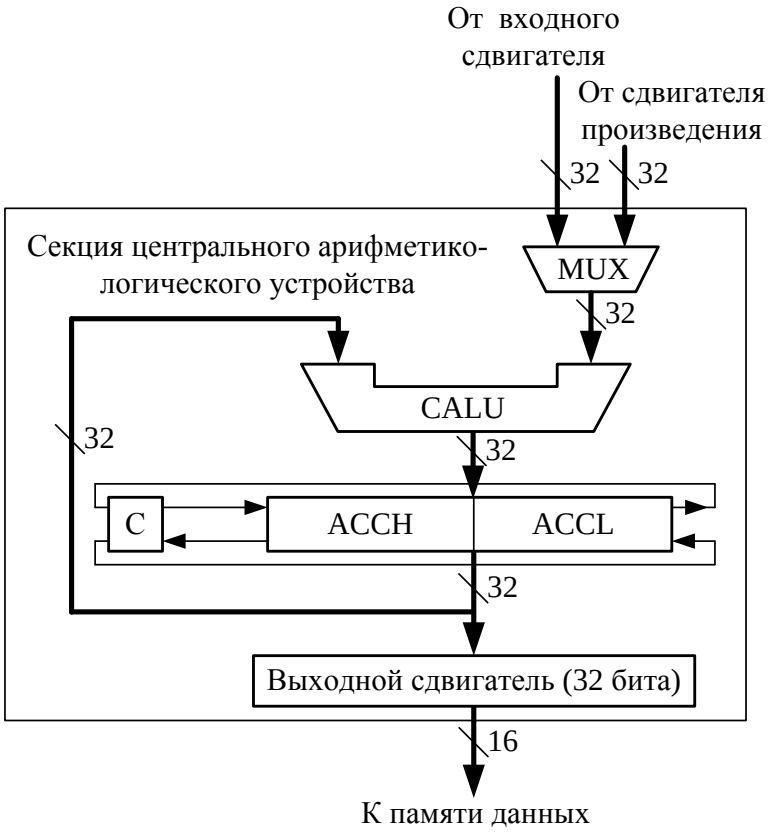


Рисунок 234 – Блок-схема центрального арифметико-логического устройства

Центральное арифметическое устройство (CALU)

CALU реализует широкий набор арифметических и логических функций, большинство из которых выполняются за один цикл. Эти функции могут быть сгруппированы в четыре категории:

- 16-битное сложение;
- 16-битное вычитание;
- логические операции;
- операции проверки бит, сдвиг и кольцевой (циклический) сдвиг.

Поскольку CALU способно выполнять булевские операции, то можно осуществлять манипуляции с битами. Для сдвига и циклического сдвига (вращения) CALU использует аккумулятор. CALU позиционируется как центральное устройство, потому что кроме него имеется еще одно независимое арифметическое устройство, обслуживающее вспомогательные регистры (ARAU), которое описано в подразделе 14.3.4.

Описание входов и выходов CALU и связанных с ним разрядов состояния приведено ниже.

Входы

CALU имеет два входа (см. рисунок 234):

- один вход всегда соединен с выходом 32 битного аккумулятора;

– на другой вход данные могут поступать из следующих источников:
– с масштабирующего сдвигателя произведения (см. 14.3.2);
– с входного сдвигателя данных (см. 14.3.1).

Выход

После выполнения операции CALU, результат поступает в 32-битный аккумулятор, который способен выполнять побитные сдвиги своего содержимого. Выход аккумулятора соединен с 32-битным выходным масштабирующим сдвигателем. Через выходной сдвигатель старшие или младшие 16-битные слова аккумулятора могут быть индивидуально сохранены в памяти данных.

Режим расширения знака

Для большинства (но не для всех) инструкций уровень бита SXM (бит 10 статусного регистра ST1) определяет, происходит ли при вычислениях знаковое расширение результата CALU. При SXM = 1 – режим расширения знака разрешен, SXM = 0 – запрет знакового расширения.

Аккумулятор

Основное назначение аккумулятора – хранение 32-битного результата вычислений CALU. Кроме того, аккумулятор способен выполнять побитный сдвиг (на один бит за один такт) или побитный циклический сдвиг (вращение) своего содержимого. Выход аккумулятора соединен с выходным масштабирующим сдвигателем, который позволяет индивидуально сохранять в памяти данных старшее или младшее 16-битные слова (со сдвигом или без). Далее описаны связанные с аккумулятором статусные биты и инструкции перехода.

Статусные биты

С аккумулятором связаны четыре статусных бита:

– Бит переноса C (Carry bit) – 9 бит статусного регистра ST1, который может измениться при выполнении:

– Операций сложения и вычитания.

C=0 если в результате операции вычитания генерируется заем или если в результате операции сложения не генерируется перенос. (Исключение: когда инструкция ADD используется со сдвигом 16 и перенос не генерируется, бит C сохраняет прежнее значение).

C = 1 если в результате операции сложения генерируется перенос или если в результате операции вычитания не генерирует заем. (Исключение: когда инструкция SUB используется со сдвигом 16 и заем не генерируется, бит C сохраняет прежнее значение).

– Побитного сдвига или вращения содержимого аккумулятора. При левом сдвиге или вращении в разряд переноса C попадает старший значащий бит (MSB) аккумулятора. При правом сдвиге (вращении) в разряд переноса C попадает младший значащий бит (LSB) аккумулятора.

– Бит режима переполнения (OVM бит, бит 11 статусного регистра ST0) определяет, как аккумулятор обрабатывает арифметическое переполнение. Когда

процессор находится в режиме OVM=1 и происходит переполнение, то аккумулятор заполняется одним из следующих значений:

– в случае положительного переполнения аккумулятор заполняется максимальным положительным числом (7FFF FFFFh);

– в случае отрицательного переполнения, аккумулятор заполняется максимальным отрицательным числом (8000 0000h).

– Флаг переполнения (OV). OV – 12 бит статусного регистра ST1. Когда в аккумуляторе не происходит переполнения, то OV устанавливается в 0, когда в аккумуляторе происходит переполнение (положительное или отрицательное), то OV устанавливается в 1.

– Бит тестирования/управления (TC). TC – 11 бит статусного регистра ST1. Он устанавливается в 0 или 1 в зависимости от значения проверяемого (тестируемого) бита. В случае инструкции NORM, если «исключающее – ИЛИ» двух самых старших разрядов аккумулятора равно 1, TC бит устанавливается в 1.

В ИС 1867ВЦ10Т предусмотрены инструкции ветвления (условные переходы, вызовы и возвраты), основанные на значениях статусных разрядов C, OV и TC и на результате сравнения содержимого аккумулятора с нулем. Более полная информация об этих инструкциях приведена в 14.8.3.

Выходной масштабирующий сдвигатель данных

Выходной масштабирующий сдвигатель данных (выходной сдвигатель) имеет 32-битный вход, соединенный с 32-битным выходом аккумулятора, и 16-битный выход на шину данных. Сдвигатель копирует 32 бита аккумулятора и, после этого, выполняет левый сдвиг на величину от 0 до 7 бит, в соответствии с заданным в инструкции коэффициентом. Старшее (инструкция SACH) или младшее (инструкция SACL) слово сдвигается и запоминается в памяти данных. Содержимое аккумулятора при этом остается неизменным.

Когда выходной сдвигатель выполняет сдвиг, самые старшие биты (MSB) теряются, а самые младшие биты (LSB) заполняются 0. На рисунке 235 показан пример, в котором значение аккумулятора сдвигается на четыре бита влево и сдвинутое старшее слово запоминается в памяти данных. На рисунке 236 показан сдвиг того же значения аккумулятора влево на 6 бит с последующим сохранением в памяти данных сдвинутого младшего слова.

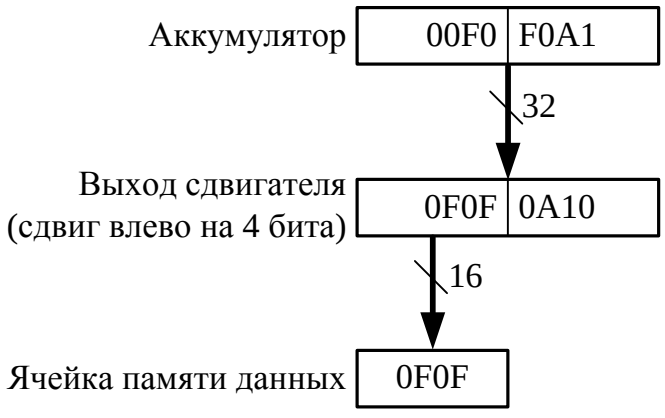


Рисунок 235 – Сдвиг и сохранение старшего слова аккумулятора

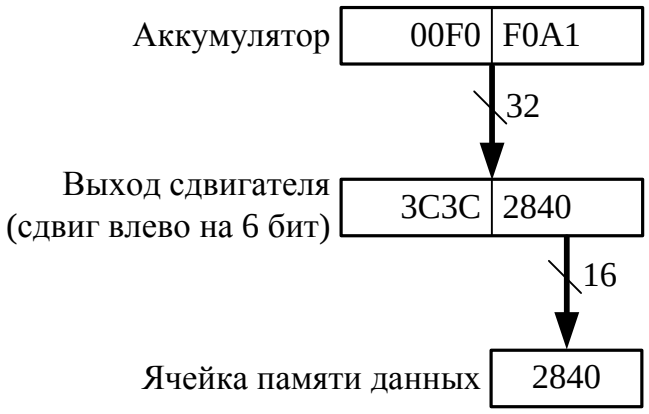


Рисунок 236 – Сдвиг и сохранение младшего слова аккумулятора

14.3.4 Арифметическое устройство вспомогательных регистров (ARAU)

Центральный процессор содержит блок независимого от CALU арифметического устройства ARAU. Основной функцией ARAU является выполнение арифметических операций над содержимым восьми вспомогательных регистров (AR7 - AR0) параллельно с операциями CALU. На рисунке 237 показан блок ARAU и относящаяся к нему логика.

Восемь вспомогательных регистров (AR7 – AR0) обеспечивают гибкую и мощную систему косвенной адресации. Любая ячейка из 64K-словного пространства памяти данных может быть доступна путем использования 16-битного адреса, содержащегося во вспомогательном регистре. Детальное описание косвенной адресации приведено в подразделе 14.7.3 «Косвенный режим адресации».

Для выбора конкретного вспомогательного регистра (AR7 – AR0) необходимо загрузить 3-битный указатель вспомогательных регистров (ARP) в статусном регистре ST0 значением от 0 до 7. ARP может быть загружен инструкцией MAR, выполняющей модификацию только вспомогательных регистров и ARP, или инструкцией LST, которая может загружать нужное значение в ST0 из памяти данных через шину чтения данных (DRDB). В любой инструкции, поддерживающей косвенную адресацию, в качестве дополнительной операции также предусмотрено изменение содержимого ARP.

Регистр, на который ссылается указатель ARP, в дальнейшем будем называть текущим вспомогательным регистром или текущим AR. Во время выполнения инструкции содержимое текущего вспомогательного регистра используется как адрес ячейки памяти данных, к которой осуществляется доступ. Если инструкция доступа к памяти осуществляет чтение операнда, ARAU выставляет значение, содержащееся в текущем AR, на адресную шину чтения данных (DRAB). В случае, если инструкция осуществляет запись в память данных, содержимое текущего AR выставляется на адресную шину записи данных (DWAB). После того как инструкция считает (запишет) данные по указанному адресу, ARAU может инкрементировать или декрементировать содержимое текущего вспомогательного регистра. ARAU реализует беззнаковую 16-разрядную арифметику.

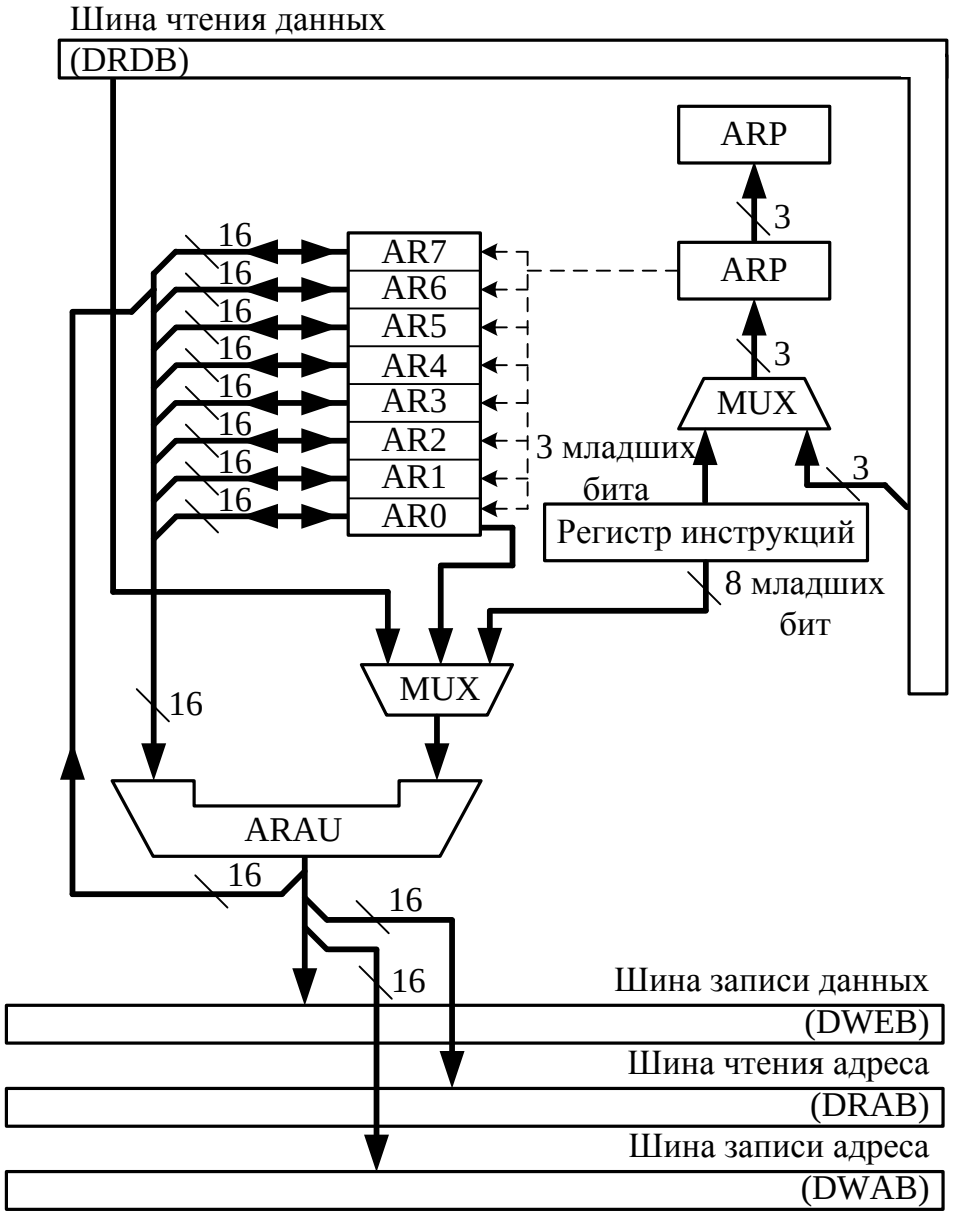


Рисунок 237 – ARAU и относящаяся к нему логика

Функции ARAU

ARAU выполняет следующие операции:

– инкрементирует или декрементирует текущий вспомогательный регистр на 1 или индексом (любая инструкция, поддерживающая косвенную адресацию);

– прибавляет значение константы к содержимому вспомогательного регистра (инструкция ADRK) или вычитает значение константы из содержимого вспомогательного регистра (инструкция SBRK). Константа – это 8-битное значение, взятое из восьми младших значащих бит слова инструкции;

– сравнивает содержимое AR0 с содержимым текущего вспомогательного регистра и по результату сравнения модифицирует бит TC в статусном регистре ST1 (инструкция CMPR). Результат сравнения передается в TC через шину записи данных (DWEB).

Обычно ARAU выполняет операции в фазе декодирования конвейера (когда инструкция задает операции декодирования). Это позволяет сгенерировать адрес перед фазой декодирования следующей инструкции. Имеется исключение из этого правила: во время выполнения инструкции NORM, модификация ARP выполняется в течение фазы исполнения (execute phase) конвейера. Более подробная информация о конвейере представлена в подразделе 14.5.2 «Операции конвейера».

Функции вспомогательных регистров

В дополнение к использованию вспомогательных регистров для адресации ячеек в памяти данных можно их использовать для других целей:

– для поддержки условных переходов, вызовов и возвратов посредством инструкции CMPR. Эта инструкция сравнивает содержимое AR0 с содержимым текущего вспомогательного регистра и, в зависимости от результата сравнения, устанавливает в соответствующий уровень бит TC статусного регистра ST1;

– для временного хранения данных, путем использования инструкций загрузки значения в регистр (инструкция LAR) и сохранения значения регистра в памяти данных (инструкция SAR);

– для использования вспомогательных регистров в качестве программных счетчиков, инкрементируя или декрементируя их как необходимо.

14.3.5 Статусные регистры ST0 и ST1

В ИС 1867ВЦ10Т предусмотрены два статусных регистра ST0 и ST1, содержащие биты состояния и управляющие биты. Эти регистры могут быть загружены из памяти данных и сохранены в памяти данных, позволяя запоминать состояние процессора при вызове процедур. Инструкция LST записывает в ST0 и ST1 (за исключением бита INTM, который не затрагивается этой инструкцией), а инструкция SST сохраняет ST0 и ST1. Некоторые индивидуальные биты могут устанавливаться и сбрасываться посредством использования инструкций SETC и CLRC. Для примера, SXM режим устанавливается SETC SXM и сбрасывается CLRC SXM. На рисунках 238 и 239 показана организация статусных регистров ST0 и ST1 соответственно. Некоторые биты зарезервированы и читаются как логические единицы. Остальные биты, описаны в таблице 103.

Таблица 103 – Формат статусных регистров ST0 и ST1

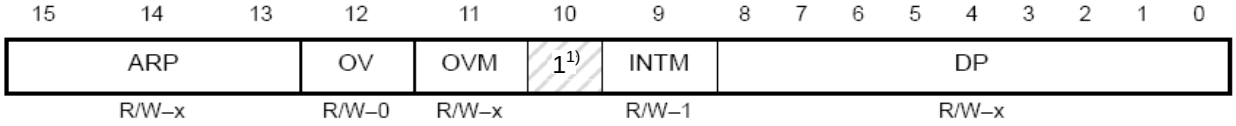
Имя бита	Описание
ARB	Буфер указателя вспомогательного регистра. Каждый раз, когда происходит загрузка ARP новым значением, предыдущее значение ARP копируется в ARB, за исключением выполнения инструкции LST. Когда ARB загружается инструкцией LST, это же значение копируется в ARP
ARP	Указатель вспомогательного регистра. 3-битное поле выбирает вспомогательный регистр, который будет использоваться для косвенной адресации. Когда ARP загружается, то предыдущее значение ARP копируется в ARB, за исключением выполнения инструкции LST. ARP может быть модифицирован инструкциями обращения к памяти, использующими косвенную адресацию, а также инструкциями MAR и LST. Когда ARB загружается инструкцией LST, это же значение копируется в ARP
C	Бит переноса. Этот бит устанавливается в 1, если в результате операции сложения генерируется перенос, или сбрасывается в 0, если в результате вычитания генерируется заем. В противном случае, он сбрасывается после сложения или устанавливается после вычитания, за исключением варианта выполнения инструкций ADD и SUB с 16-битным сдвигом. В этом случае инструкция ADD может только устанавливать, а инструкция SUB только очищать этот бит. На бит C также воздействуют инструкции побитного сдвига или вращения аккумулятора и инструкции SETC, CLRC и LST. Состояние этого бита может быть использовано для организации ветвления программы путем его проверки в условных переходах, вызовах и возвратах. После сброса бит C устанавливается в 1
CNF	Бит конфигурации DARAM. В зависимости от значения этого бита блок B0 внутрикристального ОЗУ конфигурируется или в пространство памяти данных (CNF = 0) или в пространство памяти программ (CNF = 1). CNF может модифицироваться инструкциями SETC CNF, CLRC CNF и LST. После сброса бит CNF устанавливается в 0
DP	Указатель страницы памяти данных. Используется в инструкциях с прямой адресацией, указывая на одну из 512 страниц памяти данных. При прямой адресации полный 16-битный адрес памяти данных формируется конкатенацией 9-битного поля DP (старшие разряды адреса) и 7 младших значащих бит (LSB) слова инструкции (младшие разряды адреса). Модифицируется инструкциими LST и LDP (загрузка DP)
INTM	Бит разрешения прерывания. Разрешает (INTM = 0) или запрещает (INTM = 1) все маскируемые прерывания. Этот бит устанавливается или сбрасывается командами SETC INTM и CLRC INTM соответственно. Бит INTM не влияет на немаскируемые прерывания RESET и NMI или программно инициированные прерывания. Бит INTM не меняется инструкцией LST. Бит INTM устанавливается в 1, после подтверждения прерывания (за исключением инструкции TRAP) и сброса

Продолжение таблицы 103

Имя бита	Описание
OV	Бит флага переполнения. Этот бит хранит значение, которое отражает наличие или отсутствие переполнения в CALU. После возникновения переполнения бит OV устанавливается в 1 и остается в этом уровне до тех пор, пока не будет очищен сбросом, условным переходом по переполнению (OV) или по его отсутствию (NOV) или инструкцией LST
OVM	Бит режима переполнения. Определяет как будет обрабатываться переполнение CALU. Инструкции SETC и CLRC устанавливают и сбрасывают этот бит соответственно. Инструкция LST также может быть использована для модификации бита OVM. OVM = 0 результат операции сохраняется в аккумуляторе обычным образом, без учета переполнения. OVM = 1 при переполнении в аккумулятор загружается максимальным положительным или отрицательным значением
PM	Режим сдвига произведения. PM определяет количество бит, на которое сдвигается выход PREG при передаче произведения в CALU. При этом содержимое PREG не меняется – произведение копируется в сдвигатель и в нем сдвигается. PM загружается инструкциями SPM и LST. При сбросе биты PM очищаются. PM = 00 32-битное произведение передается в CALU или память данных без сдвига. PM = 01 32-битное произведение передается в CALU или память данных с левым сдвигом на 1 бит (младший бит заполняется 0). PM = 10 32-битное произведение передается в CALU или память данных с левым сдвигом на 4 бита (младшие биты заполняются 0). PM = 11 32-битное произведение передается в CALU или память данных с правым сдвигом на 6 бит (знаковый разряд расширяется)
SXM	Режим расширения знака. Уровень SXM определяет запрет/разрешение знакового расширения в арифметических операциях. Бит SXM не влияет на работу некоторых арифметических инструкций. Например, инструкция ADDS подавляет знаковое расширение независимо от бита SXM. Этот бит устанавливается и сбрасывается командами SETC SXM и CLRC SXM, соответственно, и может быть загружен инструкцией LST. SXM = 0 режим подавления знакового расширения. SXM = 1 режим знакового расширения данных, передаваемых из входного сдвигателя в аккумулятор
TC	Бит флага тестирования/управления. Бит TC устанавливается в 1 если: – при выполнении инструкций BIT или BITT тестируемый ими разряд равен 1; – выполнено условие сравнения между текущими AR и AR0, проверяемое в инструкции CMPR; – при выполнении инструкции NORM не совпадают 2 старших значащих разряда аккумулятора (ACC(31) xor ACC(30)) = 1). Состояние этого бита может быть использовано для организации ветвления программы путем его проверки в условных переходах, вызовах и возвратах. TC зависит от инструкций BIT, BITT, CMPR, LST и NORM

Окончание таблицы 103

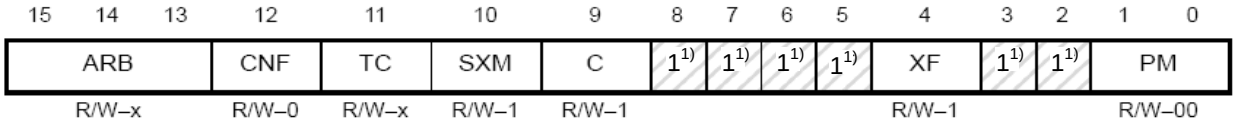
Имя бита	Описание
XF	Состояние вывода XF. Этот бит определяет состояние вывода XF, который используется как выход общего назначения. XF устанавливается в 1 инструкцией SETC XF и сбрасывается в 0 инструкцией CLRC XF. Бит XF модифицируется инструкцией LST. При сбросе устанавливается в 1



Примечание – R – доступ на чтение, W – доступ на запись,
через тире (–) значение бита после сброса,
x означает, что сброс не оказывает влияния.

1) – Зарезервированные биты. Всегда читаются как 1, запись в них не выполняется.

Рисунок 238 – Статусный регистр ST0



Примечание – R – доступ на чтение, W – доступ на запись,
через тире (–) значение бита после сброса,
x означает, что сброс не оказывает влияния.

1) – Зарезервированные биты. Всегда читаются как 1, запись в них не выполняется.

Рисунок 239 – Статусный регистр ST1

14.4 Память и пространство ввода-вывода

ИС 1867ВЦ10Т имеет 16 разрядную адресную шину, через которую может быть осуществлен доступ к трем индивидуальным областям:

– 64 К слов программной памяти;
– 64 К К слов памяти данных;
– 64 К слов пространства ввода-вывода.

В настоящем подразделе приведено описание этих трех областей и карт памяти для пространств программы, данных и ввода-вывода.

Также рассматриваются различные варианты конфигурации памяти.

14.4.1 Обзор памяти и пространства ввода-вывода

ИС 1867ВЦ10Т спроектирована на основе модифицированной Гарвардской архитектуры. В сотответствии с этой архитектурой, процессор способен осуществлять доступ к различным областям памяти, используя для этого три параллельные шины: шину адреса программ (PAB) и шины адреса данных для чтения (DRAB) и записи (DWAB). Каждая из этих трех шин может осуществлять доступ к различным областям памяти в зависимости от выполняемых микросхемой операций. Поскольку шины функционируют независимо, ИС 1867ВЦ10Т поддерживает одновременный доступ к пространствам программ и данных. Внутри заданного машинного цикла CALU может выполнять три параллельные операции с памятью.

Карта памяти ИС 1867ВЦ10Т организована в виде трех индивидуально адресуемых областей:

– программная память (64К слов), которая содержит как инструкции для выполнения, так и данные, используемые при выполнении программы;

– память данных (64К слов), которая хранит данные, используемые инструкциями;

– область ввода - вывода (64К слов), которая является интерфейсом к внешней периферии и содержит внутрикристальные регистры.

14.4.2 Программная память

Область программной памяти – это место, где находится программный код приложения. В этой области может также храниться табличная информация и непосредственные операнды. В пространстве программной памяти может быть адресовано до 64К 16-битных слов. Когда генерируется адрес, выходящий за пределы пространства, сконфигурированного как внутрикристальная память, ИС 1867ВЦ10Т автоматически осуществляет доступ к внешней памяти, формируя при этом необходимые интерфейсные сигналы. На рисунке 240 показана карта программной памяти.

Рисунок 240 – Карта программной памяти ИС 1867ВЦ10Т

Конфигурирование программной памяти

Конкретная конфигурация программной памяти задается двумя факторами: программно управляемым битом CNF и аппаратно управляемым входом MP/MC.

– Бит CNF. Этот бит (бит 12 статусного регистра ST1) определяет: разрешен ли доступ к DARAM B0 как программной памяти. При CNF = 0 блок B0 внутрикристального ОЗУ не конфигурирован в программную память. При CNF = 1, 256 слов DARAM B0 используются в качестве памяти программ. После сброса CNF = 0, то есть DARAM B0 картируется во внутреннюю область данных.

– Вход MP/MC. Уровень актуален только во время сброса и определяет: резерв или читаются инструкции из внешней памяти. Когда MP/MC = 0, ИС 1867ВЦ10Т конфигурируется как микрокомпьютер и при этом память зарезервирована. При MP/MC = 1 ИС 1867ВЦ10Т конфигурируется как микропроцессор, и программа будет выполняться из внешней программной памяти. В обоих случаях (независимо от MP/MC), программа начнет выполняться с вектора сброса по адресу 0000h.

14.4.3 Память данных

В пространстве внутренней памяти данных может быть адресовано до 64К 16-битных слов. На рисунке 241 показана карта памяти данных ИС 1867ВЦ10Т. Микросхема содержит три внутрикристальных блока DARAM – B0, B1 и B2, адресуемых как память данных.

При доступе к памяти данных может использоваться либо прямая, либо косвенная адресация. Режимы адресации детально описаны в подразделе 14.7.

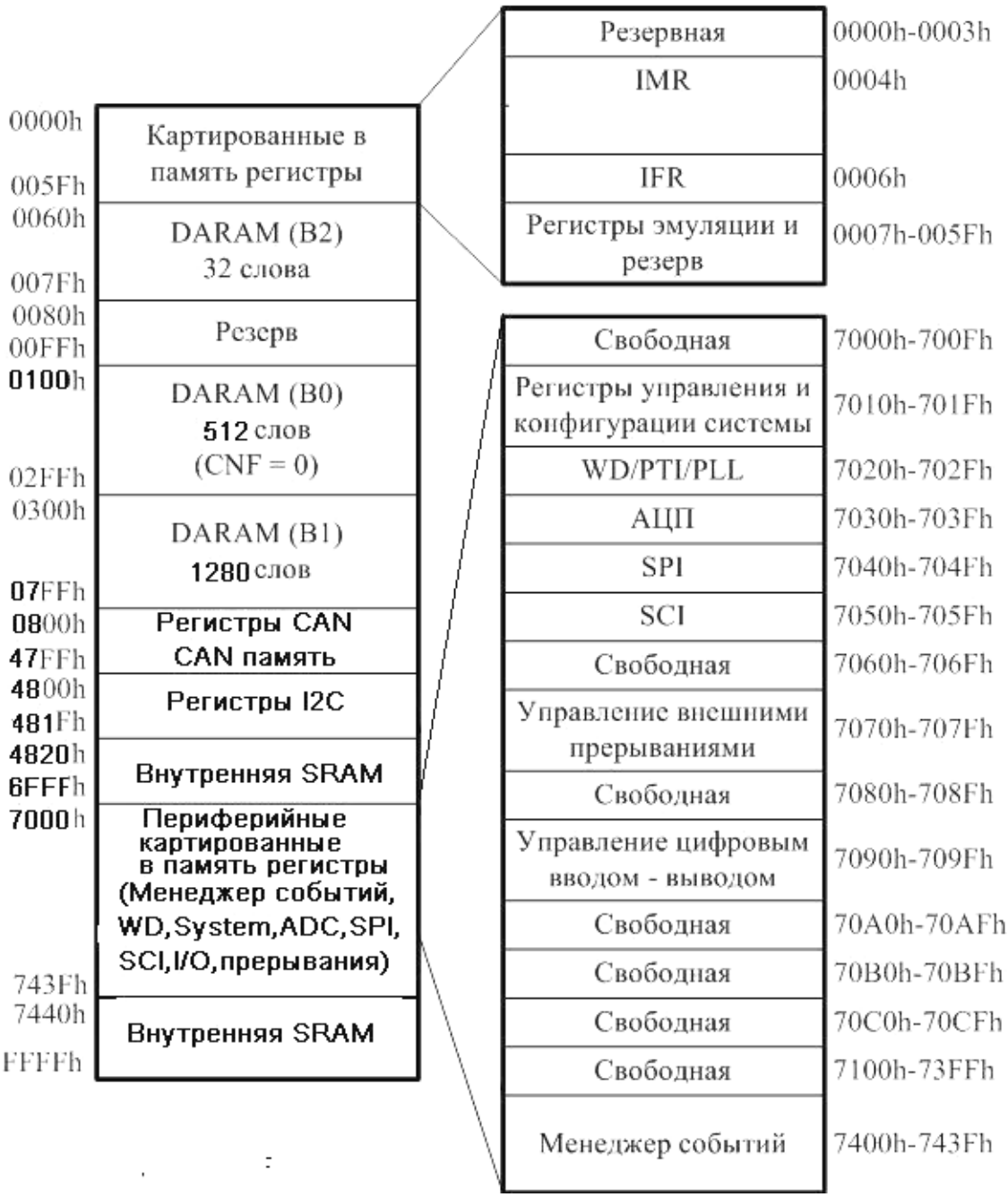


Рисунок 241 – Карта памяти данных ИС 1867ВЦ10Т

При использовании прямой адресации, память данных адресуется блоками по 128 слов. Эти блоки называются страницами. На рисунке 242 показано, как эти блоки адресуются в 64K словной памяти данных, состоящей из 512 страниц, пронумерованных от 0 до 511. Текущая страница определяется содержимым 9-битного указателя страниц (DP) статусного регистра ST1. Каждые из 128 слов на текущей странице адресуются 7 битами смещения, которое берется из инструкции, использующей прямую адресацию. Следовательно, когда инструкция использует прямую адресацию, то необходимо задать номер страницы (в предшествующей инструкции) и смещение в текущей инструкции (осуществляющей доступ к памяти).

Значение DP	Смещение	Память данных
0000 0000 0 … 0000 0000 0	000 0000 … 111 1111	Страница 0: 0000h–007Fh
0000 0000 1 … 0000 0000 1	000 0000 … 111 1111	Страница 1: 0080h–00FFh
0000 0001 0 … 0000 0001 0	000 0000 … 111 1111	Страница 2: 0100h–017Fh
… …	… …	… …
1111 1111 1 … 1111 1111 1	000 0000 … 111 1111	Страница 511: FF80h–FFFFh

Рисунок 242 – Страницы локальной памяти данных

Карта страницы 0 памяти данных (Page 0)

В 64K словную память данных включены картированные регистры, расположенные в младших адресах нулевой страницы. Речь идет о регистрах, имеющих доступ с нулевым ожиданием: регистре маски прерываний (IMR и регистре флагов прерываний (IFR). Кроме этого, страница 0 памяти данных содержит резервную область, предназначенную для тестирования, функционирования системы эмуляции и для передачи специальной информации.

ПРЕДУПРЕЖДЕНИЕ! Не используйте память для тестирования и эмуляции в своих приложениях. Это может вызвать изменение режима работы схемы и, следовательно, непредсказуемо влиять на выполнение приложения.

В старших адресах нулевой страницы расположен 32-словный блок B2 (сверхоперативная память). Этот блок, как правило, используется для хранения переменных и контекста (состояния процессора до прерывания, вызова подпрограммы и т.п.), позволяя тем самым избежать фрагментирования больших блоков внешнего или внутреннего ОЗУ. Блок B2 поддерживает операции с двойного доступа и все режимы адресации. Карта страницы 0 приведена в таблице 104.

Таблица 104 – Карта страницы 0 памяти данных

Адрес	Описание
0000h–0003h	Резерв
0004h	Регистр маски прерывания IMR
0005h	Резерв
0006h	Регистр флагов прерываний IFR
0007h–005fh	Резерв
0060h–007Fh	Блок сверхоперативной памяти данных (DARAM B2)

Конфигурация памяти данных

Бит CNF (бит 12 статусного регистра ST1) позволяет конфигурировать память данных.

– CNF = 0. DARAM B0 конфигурируется в пространство данных.

– CNF = 1. DARAM B0 конфигурируется в пространство программ.

После сброса B0 конфигурируется в локальную пространство данных (CNF =0).

14.4.4 Пространство ввода-вывода

Область памяти ввода вывода адресует до 64К 16-битных слов. Рисунок 243 иллюстрирует карту области ввода вывода для ИС 1867ВЦ10Т.

Рисунок 243 – Карта пространства ввода-вывода

14.5 Управление последовательностью выполнения программы

Программное управление включает управление порядком, в котором выполняются один или более блоков инструкций. Обычно программный поток – последовательный (ИС 1867ВЦ10Т, в основном, выполняет инструкции в последовательных программных адресах, за исключением команд, которые меняют эту последовательность). Иногда возникает необходимость перехода в новую область адресов, чтобы затем вновь продолжить последовательное выполнение инструк-

ций. Для этих целей в ИС 1867ВЦ10Т предусмотрены инструкции переходов, вызовов подпрограмм, возвратов, повторов и набор аппаратных и программных прерываний. Прерывания описаны в подразделе 14.6 «Системные функции».

14.5.1 Генератор программных адресов

Для обеспечения непрерывности программного потока процессор должен выполнить генерацию следующего программного адреса (последовательного или нет), пока выполняется текущая инструкция. Блок-схема генератора программного адреса приведена на рисунке 244.

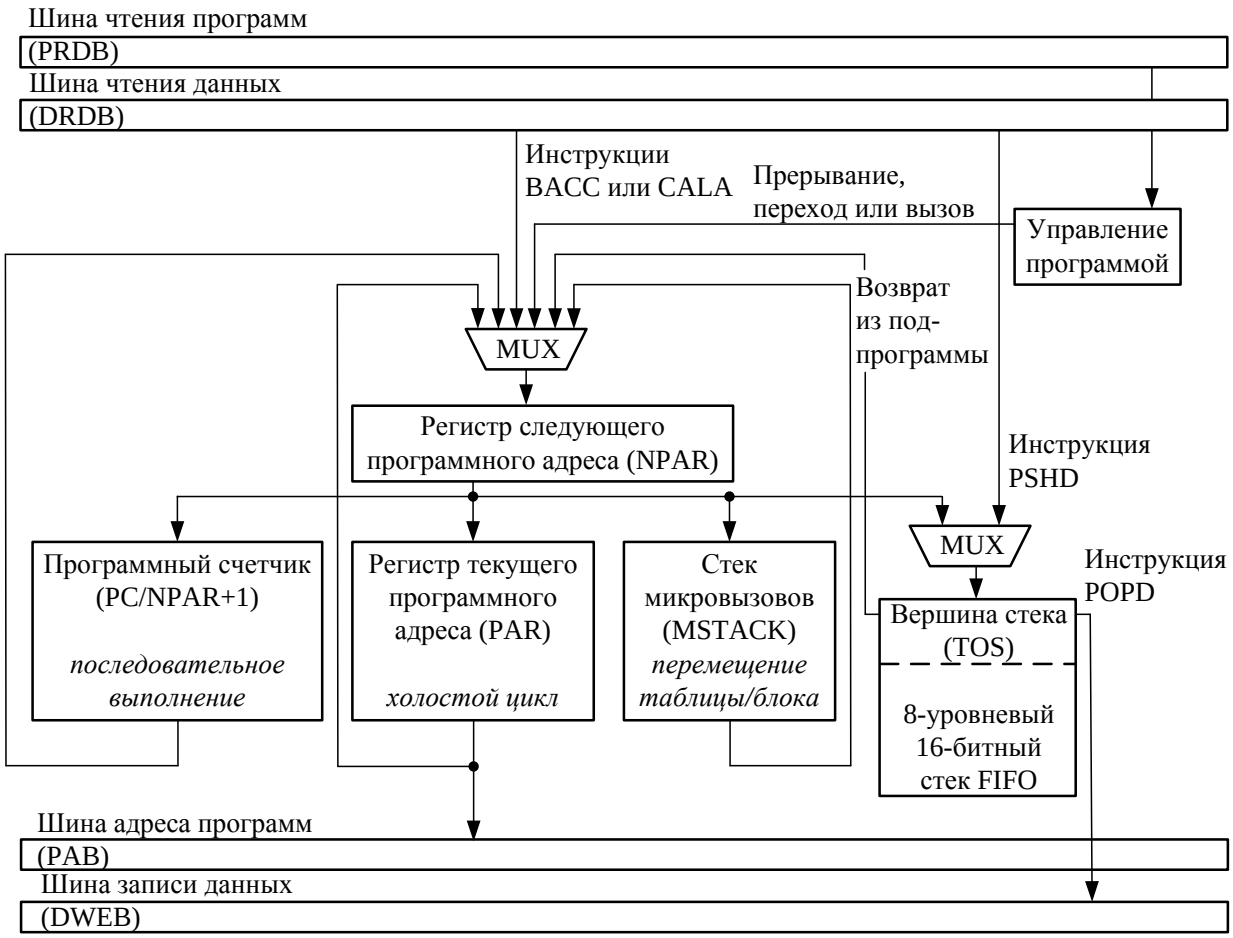


Рисунок 244 – Блок-схема логики генерации программного адреса

В таблице 105 приведены основные операции, используемые при генерации программного адреса.

Таблица 105 – Генерация программного адреса

Операция	Источник программного адреса
Последовательная операция	PC (содержит программный адрес + 1)
Холостой цикл	PAR (содержит программный адрес)
Возврат из подпрограммы	TOS (вершина стека содержит адрес возврата)

Продолжение таблицы 105

Операция	Источник программного адреса
Возврат из перемещения таблицы/блока	MSTACK (стек микровызовов или микростек содержит адрес возврата)
Переход или вызов по адресу, указанному в инструкции	Инструкция перехода или вызова; адрес поступает с шины чтения программы (PRDB)
Переход или вызов по адресу из младшей части аккумулятора	Выставленное на шину чтения данных (DRDB) значение младшего слова аккумулятора
Переход к подпрограмме обработки прерывания	Вектор прерывания поступает с шины чтения программы (PRDB)

Генератор программного адреса ИС 1867ВЦ10Т содержит следующие устройства:

– Программный счетчик (PC). ИС 1867ВЦ10Т имеет 16-битный программный счетчик, который адресует внутреннюю и внешнюю программную память при выборке инструкций.

– Регистр адреса программы (PAR). Выходом этого регистра является шина программного адреса (PAB). PAB – это 16-битная шина, которая обеспечивает программный адрес для чтения и записи.

– Стек. Логика генерации программного адреса включает 16-битный аппаратный стек глубиной 8 слов для запоминания до 8 адресов возврата. Кроме того, стек можно использовать для временного хранения данных.

– Микростек. (MSTACK). Этот одноуровневый 16-битный стек используется в инструкциях перемещений таблиц и блоков и в инструкциях умножения с накоплением. В этих инструкциях производится одновременная адресация двух массивов и, в качестве одного из источников адреса, используется счетчик команд (PC). Для корректного возобновления выполнения программы в MSTACK сохраняется адрес инструкции, расположенной сразу за выполняемой. По завершении ее выполнения этот адрес переписывается в счетчик команд. Этот механизм подобен вызову подпрограммы, состоящей из одной команды, с последующим возвратом. Поэтому микростек корректнее было бы назвать стеком микровызовов.

– Счетчик повтора (RPTC). 16-битный RPTC используется инструкцией повтора RPT для определения числа повторений инструкции, следующей за RPT.

Программный счетчик (PC)

Логика генерации программного адреса использует 16-битный программный счетчик (PC) для адресации внутренней или внешней программной памяти. PC содержит адрес следующей выполняемой инструкции. Через шину адреса программы PAB происходит выборка инструкции по этому адресу программной памяти и загрузка ее в регистр инструкций. Когда регистр инструкций загружается, то PC уже содержит адрес следующей инструкции.

Чтобы обеспечить последовательный и непоследовательный программный поток, предусмотрены различные способы загрузки PC. Что именно загружается в PC в соответствии с исполняемой операцией, показано в таблице 106.

Таблица 106 – Адрес, загружаемый в программный счетчик

Операция	Адрес, загружаемый в программный счетчик
Последовательное выполнение	В PC загружается PC + 1 для однословных и PC + 2 для двухсловных инструкций
Переход	PC загружается длинным непосредственным значением, следующим сразу за инструкцией перехода
Вызов подпрограммы и возврат	При вызове адрес следующей инструкции вталкивается в стек, а PC загружается длинным непосредственным значением, следующим сразу за инструкцией вызова. Инструкция возврата выталкивает из стека адрес возврата обратно в PC для возврата к вызывающей последовательности
Программное или аппаратное прерывание	PC загружается адресом из ячейки соответствующего вектора прерывания. В этой ячейке находится инструкция перехода, которая загружает в PC адрес соответствующей подпрограммы обработки прерывания
Вычисляемый переход (calculated GOTO)	В PC загружается содержимое 16 младших бит аккумулятора. В операциях вычисляемых переходов используются инструкции BACC (переход по адресу в аккумуляторе) или CALA (вызов подпрограммы из ячейки, указанной в аккумуляторе)

Стек

ИС 1867ВЦ10Т содержит 16-битный восьмиуровневый аппаратный стек. Логика формирования адреса программы использует стек для хранения адресов возврата, когда вызывается подпрограмма или происходит прерывание. При переходе ЦП к подпрограмме или к обработке прерывания адрес возврата автоматически загружается в верхушку стека (это не требует дополнительного цикла). После завершения подпрограммы или обработки прерывания, инструкция возврата пересылает адрес возврата из верхушки стека в программный счетчик.

Когда не все восемь уровней используются для адресов возврата, стек может использоваться для сохранения контекстных данных на время выполнения подпрограммы или обработки прерывания или для других целей.

Доступ к стеку может быть осуществлен двумя группами инструкций:

– Инструкции PUSH и POP. Инструкция PUSH копирует 16 младших бит аккумулятора в вершину стека. Инструкция POP копирует значение из вершины стека в 16 младших бит аккумулятора.

– Инструкции PSHD и POPD. Эти инструкции позволяют организовать стек в памяти данных, если для вложенных подпрограмм или прерываний не хватает 8 уровней аппаратного стека. Инструкция PSHD копирует значение ячейки памяти данных в верхушку стека. Инструкция POPD копирует значение из верхушки стека в ячейку памяти данных.

Всякий раз, когда значение вталкивается в вершину стека (инструкцией или логикой формирования адреса программы), содержимое каждого уровня проталкивается на один уровень вниз и значение нижней ячейки стека теряется. Следовательно, данные теряются, если более чем восемь последовательных записей в стек происходит без чтения из стека. На рисунке 245 показана операция записи в стек.

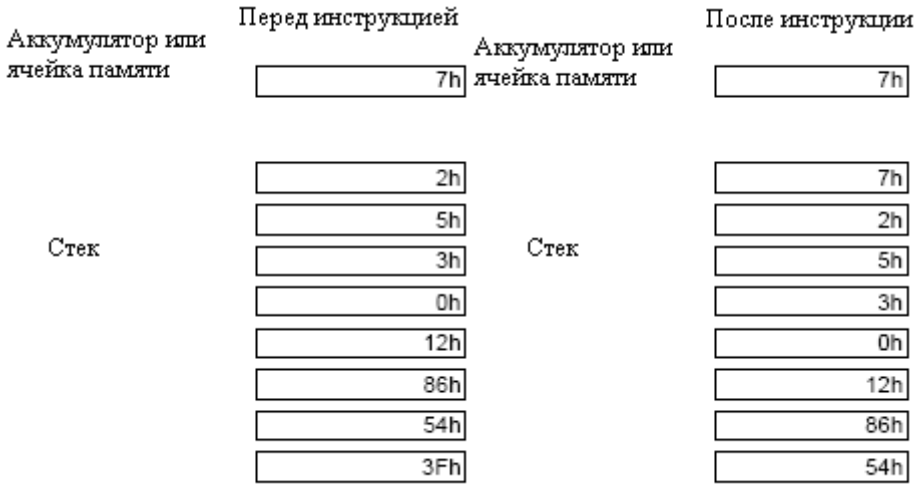


Рисунок 245 – Операция записи в стек (PUSH)

Операция чтения из стека является операцией, инверсной операции записи в стек. Эта операция копирует значение из каждого уровня стека в следующий более высокий уровень. Любое чтение из стека после семи последовательных чтений копирует значение из самой нижней ячейки стека вверх. На рисунке 246 показана операция чтения из стека.

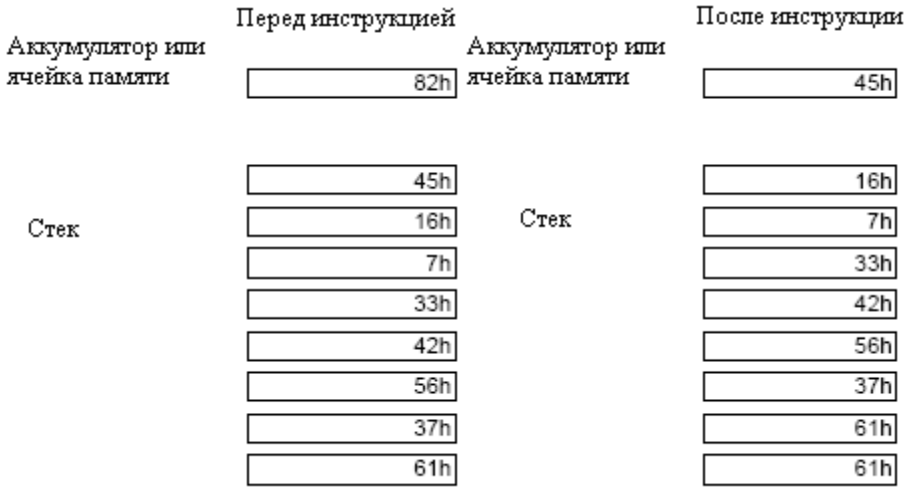


Рисунок 246 – Операция чтения из стека (POP)

Микростек (MSTACK)

Логика формирования программного адреса использует 16-битный одноуровневый стек MSTACK для сохранения адреса возврата перед выполнением определенных инструкций. Речь идет о двухоперандных инструкциях, в которых производится одновременная адресация двух массивов, и в качестве одного из источников адреса задействована логика генерации программного адреса. Это инструкции BLDD, BLPD, MAC, MACD, TBLR и TBLW. В режиме повтора они ис-

пользуют инкремент PC для получения адреса первого операнда и могут использовать ARAU для формирования адреса второго операнда. До выполнения основной операции эти инструкции записывают адрес возврата (адрес следующей инструкции для выборки) в MSTACK. После завершения повторяемой инструкции значение из MSTACK пересылается назад в логику генерации программного адреса. Операции с MSTACK невидимы для пользователя. В отличие от стека, MSTACK может быть задействован только логикой формирования программного адреса. Не предусмотрено инструкций, позволяющих использовать MSTACK для хранения данных.

14.5.2 Операции конвейера

Конвейер инструкций состоит из последовательности шинных операций, происходящих во время выполнения инструкции.

ИС 1867ВЦ10Т имеет четыре независимых стадии конвейера:

- выборка инструкции;
- декодирование инструкции;
- выборка операнда;
- выполнение инструкции.

Поскольку стадии конвейера независимы, эти операции могут перекрываться во времени. В течение какого-либо конкретного цикла одна из четырех различных инструкций может быть активна, каждая на разной стадии завершения. На рисунке 247 показана работа четырехуровневого конвейера для однословной одноцикловой инструкции, выполняющейся без состояний ожидания.

Конвейер, по существу, невидим для программиста, за исключением следующих случаев:

– Инструкция NORM модифицирует ARP и использует текущий вспомогательный регистр (он указывается ARP) во время фазы исполнения конвейера. Если следующие два слова инструкции связаны с изменением значения текущего вспомогательного регистра или ARP, то это изменение будет происходить во время фазы декодирования конвейера (до выполнения инструкции NORM). Соответственно, инструкция NORM будет использовать неверное значение вспомогательного регистра, и следующие инструкции будут использовать неверное значение ARP.

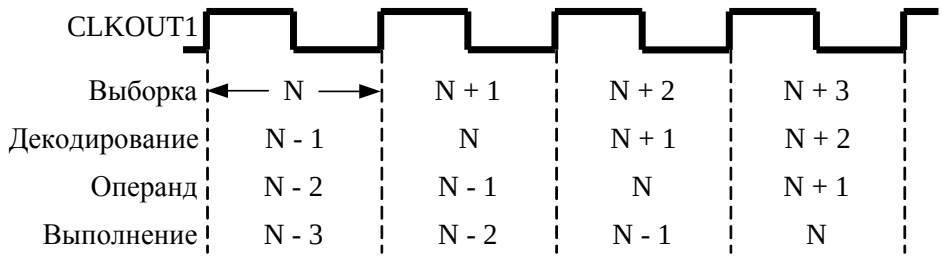


Рисунок 247 – Четырехуровневые конвейерные операции

14.5.3 Переходы, вызовы подпрограмм и возвраты

Инструкции перехода, вызова подпрограмм и возврата прекращают последовательное выполнение потока инструкций и передают управление на другую ячейку программной памяти. Отличие инструкций переходов от инструкций вызовов подпрограмм в том, что в первом случае происходит только передача управления на другой адрес, а во втором, кроме этого, сохраняется адрес возврата (адрес инструкции, следующей за инструкцией вызова) в верхушке аппаратного стека. Каждая вызываемая процедура или программа обработки прерывания должны завершаться инструкцией возврата, которая пересылает адрес возврата из стека обратно в программный счетчик.

ИС 1867ВЦ10Т имеет два типа инструкций переходов, вызовов подпрограмм и возвратов:

– Безусловные инструкции. Безусловные инструкции переходов, вызовов подпрограмм и возвратов выполняются всегда.

– Условные инструкции. Условные инструкции переходов, вызовов подпрограмм и возвратов выполняются только при удовлетворении определенных, указанных в инструкции условий.

Безусловные переходы

Когда встречается безусловный переход, то он всегда выполняется. Во время выполнения этой инструкции программный счетчик загружается указанным программным адресом (адресом перехода), и программа начинает выполняться с этого адреса. Источником адреса, загружаемого в программный счетчик, может быть следующее слово инструкции перехода или 16 младших бит аккумулятора.

По достижении инструкцией перехода фазы выполнения конвейера два следующих командных слова, пройдя фазу выборки, уже попадут в конвейер. Эти два командных слова будут удалены из конвейера (не будут выполняться), и после этого выполнение потока инструкций продолжится с адреса перехода.

К безусловным инструкциям переходов относятся инструкции B (переход) и BACC (переход по содержимому аккумулятора).

Безусловные вызовы подпрограмм

Когда встречается безусловный вызов подпрограммы, то он всегда выполняется. Во время выполнения вызова программный счетчик загружается указанным программным адресом (адресом подпрограммы), и выполнение начинается с этого адреса. Источником адреса, загружаемого в программный счетчик, может быть следующее слово инструкции перехода или 16 младших бит аккумулятора. Перед загрузкой PC адрес возврата сохраняется в стеке. После выполнения вызванной подпрограммы инструкция возврата загрузит в PC адрес возврата из стека, и программа продолжит выполнение с инструкции, следующей за инструкцией вызова.

Когда инструкция вызова подпрограммы достигнет в конвейере фазы выполнения, два следующих командных слова, пройдя фазу выборки, уже попадут в конвейер. Эти два командных слова будут удалены из конвейера (не будут выполняться), адрес возврата из подпрограммы будет сохранен в стеке, и после этого начнется выполнение потока инструкций вызванной подпрограммы.

К безусловным инструкциям вызова подпрограмм относятся инструкции CALL (вызовы) и CALA (вызов подпрограммы по адресу, определяемому содержимым аккумулятора).

Безусловные возвраты

Когда встречается безусловный возврат, то он всегда выполняется. Когда выполняется инструкция возврата, PC загружается значением из верхушки стека, и выполнение программы продолжается с этого адреса.

Когда инструкция возврата достигнет в конвейере фазы выполнения, два следующих командных слова, пройдя фазу выборки, уже попадут в конвейер. Эти два командных слова будут удалены из конвейера (не будут выполняться), из стека будет взят адрес возврата, и продолжится выполнение потока инструкций основной программы.

14.5.4 Условные переходы, вызовы подпрограммы и возвраты

ИС 1867ВЦ10Т реализует инструкции переходов, вызовов подпрограмм и возвратов, которые исполняются только в том случае, если удовлетворяется одно или несколько условий. Для этого необходимо в условных инструкциях задать условия в виде операндов. В таблице 107 приведены условия, которые могут использоваться в этих инструкциях, и соответствующие им символы операндов.

Таблица 107 – Условия для условных вызовов подпрограммы и возвратов

Символ операнда	Условие	Описание
EQ	ACC = 0	Аккумулятор равен 0
NEQ	ACC ≠ 0	Аккумулятор не равен 0
LT	ACC < 0	Аккумулятор меньше 0
LEQ	ACC ≤ 0	Аккумулятор меньше или равен 0
GT	ACC > 0	Аккумулятор больше 0
GEQ	ACC ≥ 0	Аккумулятор больше или равен 0
C	C = 1	Бит переноса установлен в 1
NC	C = 0	Бит переноса установлен в 0
OV	OV = 1	Бит переполнения установлен в 1
NOV	OV = 0	Бит переполнения установлен в 0
BIO	BIO# = 0	На выводе BIO# активный (низкий) уровень
TC	TC = 1	Тестовый флаг установлен в 1
NTC	TC = 0	Тестовый флаг установлен в 0

Использование набора условий

В условных инструкциях можно проверять одно или несколько условий, указав один или несколько операндов соответственно. Если используется комбинация условий, то для выполнения этой инструкции все указанные условия должны быть удовлетворены. Необходимо отметить, что допустимы и имеют смысл только определенные комбинации условий (см. таблицу 108).

Для каждой комбинации условия должны выбираться из группы 1 и группы 2 следующим образом:

– Группа 1. Можно выбрать до двух условий. Каждое из этих условий должно быть из различных категорий (A или B). Нельзя указывать условия из одной и той же категории. Например, можно одновременно (в одной инструкции) проверять условия EQ и OV, но нельзя одновременно проверять условия GT и NEQ.

– Группа 2. Можно выбрать до трех условий. Все выбранные условия должны принадлежать к различным категориям (A, B или C). Нельзя выбрать два условия из одной и той же категории. Например, можно проверять условия TC, C и BIO одновременно, но нельзя проверять одновременно C и NC.

Таблица 108 – Комбинации условий

Группа 1		Группа 2		
Категория A	Категория B	Категория A	Категория B	Категория C
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	-
LT	-	-	-	-
LEQ	-	-	-	-
GT	-	-	-	-
GEQ	-	-	-	-

Установление истинных значений (стабилизация) условий

Условная инструкция должна проверять последние (новые) значения разрядов статусного регистра. До тех пор пока, не завершена четвертая фаза (фаза выполнения) конвейера, условия не могут считаться установившимися (истинными), поэтому должен быть выполнен еще один цикл после предыдущей инструкции. Контроллер конвейера останавливает декодирование любых инструкций, следующих за условной инструкцией до тех пор, пока значения условий не установятся (будут истинными).

Условные переходы

Инструкция перехода может передать управление программой в любую ячейку программной памяти. Инструкция условного перехода выполняется только в том случае, когда удовлетворяются одно или больше заданных пользователем условий (см. таблицу 107). Если все условия выполнены, PC загружается вторым словом инструкции перехода, которое содержит адрес перехода, и выполнение программы продолжается с этого адреса.

Во время тестирования условий два командных слова, следующих за инструкцией перехода, уже пройдут фазу выборки и попадут в конвейер. Если все заданные условия удовлетворены, эти два командных слова будут удалены из конвейера (не будут выполняться) и, после этого, выполнение потока инструкций продолжится с адреса перехода. Если условия (или хотя бы одно из них) не удовлетворены, то вместо перехода будут выполняться два командных слова, уже находящиеся в конвейере. Поскольку условные переходы используют условия,

определенные выполнением предыдущих инструкций, они выполняются на один цикл дольше по сравнению с безусловными.

К инструкциям условного перехода относятся инструкции BCND (условный переход) и BANZ (переход, если текущий AR не равен 0). Инструкция BANZ полезна для реализации циклов.

Условные вызовы подпрограмм

Инструкция условного вызова подпрограммы CC (условный вызов) выполняется только тогда, когда удовлетворяется заданное условие или условия (см. таблицу 107). Это позволяет программе выбрать среди множества подпрограмм нужную, основываясь на данных результатов вычислений. Если все условия удовлетворены, в PC загружается второе слово инструкции вызова, содержащее стартовый адрес подпрограммы. Перед переходом к подпрограмме, процессор сохраняет в стеке адрес инструкции, следующей за инструкцией CC (адрес возврата из подпрограммы). Подпрограмма должна заканчиваться инструкцией возврата. Эта инструкция загрузит в PC адрес возврата из стека, и процессор возобновит выполнение основной программы с инструкции, следующей за инструкцией вызова.

Во время тестирования условий два командных слова, следующих за инструкцией вызова подпрограммы, уже пройдут фазу выборки и попадут в конвейер. Если все заданные условия удовлетворены, эти два командных слова будут удалены из конвейера (не будут выполняться) и, после этого, выполнение потока инструкций продолжится с начального адреса вызванной подпрограммы. Если условия (или хотя бы одно из них) не удовлетворены, то вместо вызова будут выполняться два командных слова, уже находящиеся в конвейере. Поскольку имеется цикл ожидания для стабилизации условий, то условный вызов подпрограммы выполняется на один цикл дольше по сравнению с безусловным.

Условные возвраты

Возвраты используются в сочетании с вызовами подпрограмм и с прерываниями. При вызове подпрограммы или при прерывании адрес возврата сохраняется в стеке, и после этого происходит передача управления программой в другое место программной памяти. Вызванная подпрограмма или прерывание заканчиваются инструкцией возврата, которая выталкивает адрес возврата из верхушки стека в PC.

Условная инструкция возврата RETC (условный возврат) выполняется только в том случае, если условие или условия удовлетворены (см. таблицу 107). Использование инструкции RETC позволяет избежать в вызванной подпрограмме или в программе обработки прерывания применения условных переходов, если необходимо указать несколько возможных путей возврата, выбор одного из которых зависит от результата обработки данных.

Если все условия удовлетворены, то инструкция RETC выполняется, процессор выталкивает адрес возврата из стека в PC и продолжает выполнение основной (прерванной) программы. RETC, подобно RET, является однословной инструкцией. Однако, из-за потенциального непоследовательного изменения счетчика команд, эта инструкция отрабатывается с тем же эффективным временем вы-

полнения, что и инструкции условного перехода BCND и условного вызова подпрограммы CC.

Во время проверки условий возврата два командных слова, следующих за инструкцией условного возврата, уже пройдут фазу выборки и попадут в конвейер. Если все заданные условия удовлетворены, эти два командных слова будут удалены из конвейера (не будут выполняться), произойдет возврат и возобновится выполнение потока инструкций основной программы. Если условия (или хотя бы одно из них) не удовлетворены, то вместо инструкции условного возврата будут выполняться два командных слова, уже находящиеся в конвейере. Поскольку имеется цикл ожидания для стабилизации условий, то условный возврат из подпрограммы (из прерывания) выполняется на один цикл дольше по сравнению с безусловным.

14.5.5 Повтор однократных инструкций

Инструкция повтора RPT позволяет организовать выполнение однократной инструкции N + 1 раз, где N – операнд, задаваемый в инструкции RPT. Когда инструкция RPT выполняется, в счетчик повтора RPTC загружается значение N. Затем, после каждого повтора выполняемой инструкции, RPTC уменьшается на 1 до тех пор, пока не станет равным 0. RPTC может использоваться как 16-битный счетчик, если число повторов считывается из памяти данных, и как 8-битный счетчик, если число повторов определено непосредственным операндом в инструкции RPT. Инструкция повтора полезна с инструкциями NORM (нормализация содержимого аккумулятора), MACD (умножение с накоплением и сдвигом данных) и SUBC (условное вычитание). При выполнении этих инструкций в режиме повтора, шины программного адреса и инструкций задействуются для выборки второго операнда параллельно с выборкой первого через шины адреса и данных. Это позволяет инструкциям MACD и BLPD в режиме повтора эффективно выполняться за один цикл.

14.6 Системные функции

В этом подразделе описываются системные функции контроллера ПЦОС:

– периферийный интерфейс, обеспечивающий обмен данными между шиной данных ЦП и назависимой от ЦП периферийной шиной;

– системные конфигурационные регистры, позволяющие программе управлять и получать статусную информацию для функций, которые влияют как на ЦП, так и на соответствующую периферию;

– аппаратные прерывания (включая сброс), управляемые как со стороны регистров ЦП, так и со стороны регистров периферии;

– режимы с уменьшенным энергопотреблением, влияющие как на ЦП, так и на периферию.

14.6.1 Интерфейс периферийных устройств (периферийный интерфейс)

Для того, чтобы обеспечить функционирование большого числа периферийных устройств без дополнительной электрической нагрузки на шины данных ЦП, ИС 1867ВЦ10Т имеет отдельную периферийную шину, которая работает с более

низкой частотой, чем шины ЦП. Большинство периферийных устройств подключено к этой шине, хотя некоторые из них (к примеру, менеджер событий) имеют интерфейс непосредственно с шиной данных ЦП.

Одна из функций периферийного интерфейса – это взаимодействие между ЦП и периферийной шиной. Ядро процессора тактируется или в два раза (режим “2×”) или в четыре раза (режим “4×”) быстрее, чем периферийная шина. Поскольку периферийная шина работает медленнее, чем шина ЦП, то чтение или запись на периферийной шине осуществляются за нескольких циклов ЦП. Точное число циклов ЦП для доступа к периферии зависит от:

– частоты тактирования периферийного устройства;

– фазы периферийного тактового сигнала, в которой ЦП инициирует доступ к периферии;

– типа доступа: чтение или запись.

В таблице 109 приведено требуемое количество циклов ЦП, необходимых для завершения чтения и записи через периферийную шину.

Если, для примера, тактирование происходит в четырехкратном режиме, то однократное периферийное чтение может потребовать 5, 6, 7 или 8 циклов ЦП, в зависимости от фазы периферийного тактового сигнала, в которой ЦП инициирует доступ к периферии. Если выполняется доступ туда и обратно, все доступы после первого будут требовать восемь циклов в четырехкратном кратном режиме. Необходимо отметить, что при записи всегда требуется на один цикл больше, чем при чтении. Это в сочетании с нулевым ожиданием доступа к внешней памяти и доступом менеджера событий через шины данных ЦП. Эти обращения требуют один цикл для чтения и два цикла для записи.

Таблица 109 – Циклы ЦП для полного чтения и записи в периферийную шину

Тип доступа	Двукратный режим		Четырехкратный режим	
	Одиночный доступ (циклов)	Обратный/повторный доступ (циклов)	Одиночный доступ (циклов)	Обратный/повторный доступ (циклов)
Чтение	3 или 4	4	5, 6, 7 или 8	8
Запись	4 или 5	4	6, 7, 8 или 9	8

Вся доступная память ЦП является 16-битной. Чтение из 8-битной периферии выравнивается по младшим значащим разрядам. Старшие восемь бит при записи в 8-битную периферию игнорируются. Вся периферия расположена в области данных ЦП, что позволяет всему множеству инструкций работать с периферийными регистрами. Область ввода-вывода не используется внутрикристальной периферией.

14.6.2 Системные конфигурационные регистры

Системные конфигурационные регистры показаны на рисунке 248.

Необходимо отметить что:

– все резервные биты читаются как неопределенные значения (если не оговорено особо);

– бит 0 периферийного адреса не декодируется, следовательно, каждый из 16-битных регистров доступен как по соответствующему четному адресу, так и по следующему за ним нечетному адресу. Например, регистр SYSIVR стандартно адресуется значением 701Eh, но может также быть доступен по адресу 701Fh.

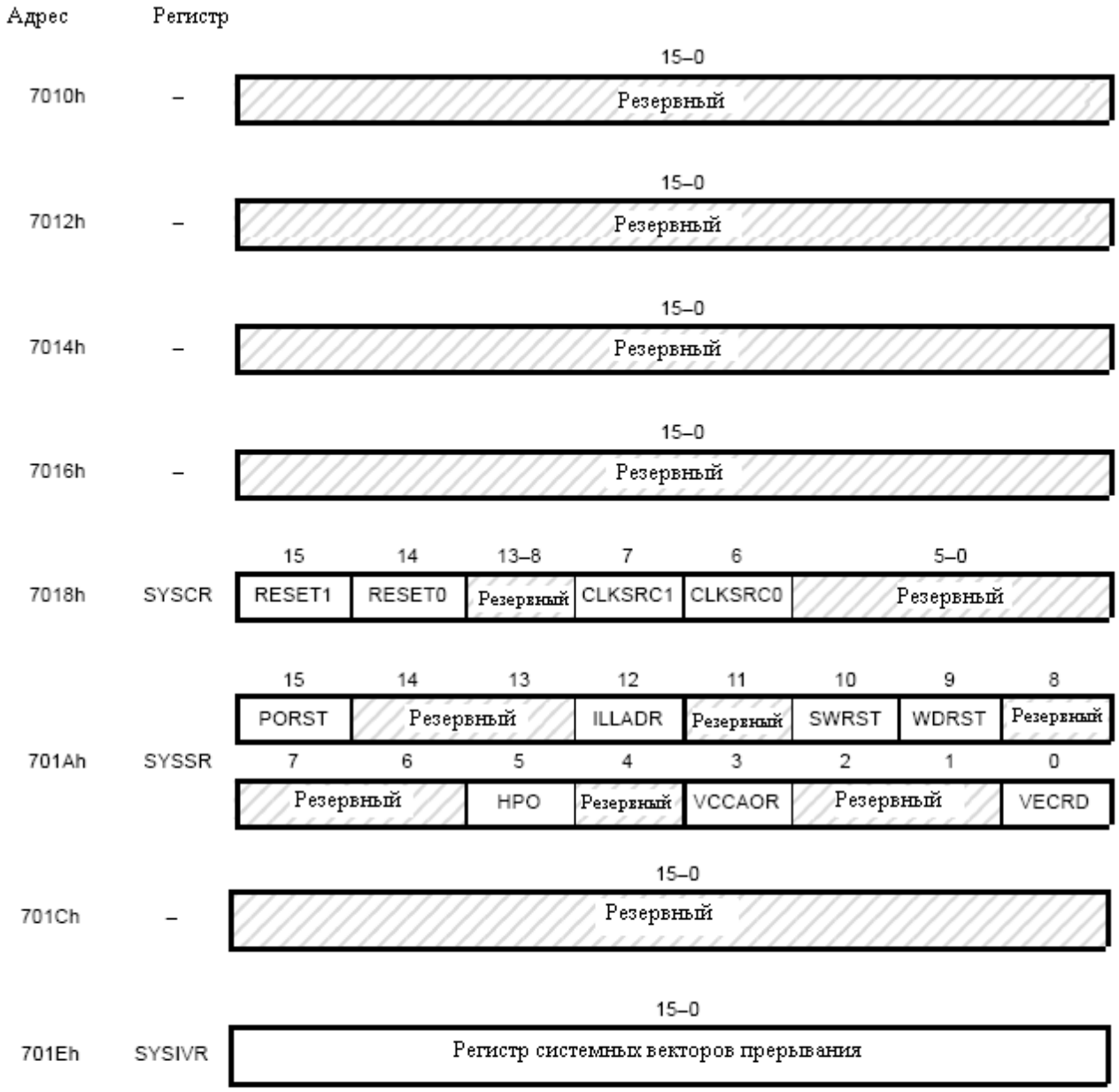
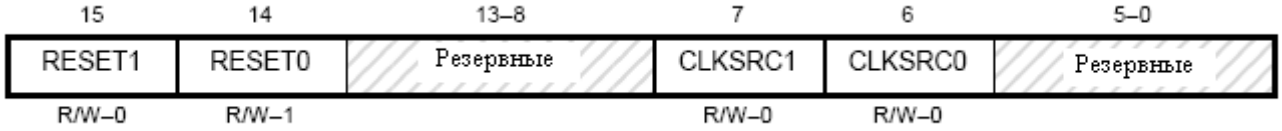


Рисунок 248 – Системные конфигурационные регистры

Системный управляющий регистр (SYSCR)

На рисунке 249 представлен формат системного управляющего регистра (SYSCR), описание разрядов этого регистра приведено в таблице 110.



Примечание – R – доступ на чтение, W – доступ на запись, через тире (–) значение бита после сброса.

Рисунок 249 – Системный управляющий регистр (SYSCR) - адрес 7018h

Таблица 110 – Системный управляющий регистр

Биты	Название		Описание
15, 14	RESET1	RESET0	Биты программного сброса. Управляют функцией программного сброса ИС 1867ВЦ10Т, должны быть записаны в одно и то же время. Запись 1 в RESET1 или 0 в RESET0 вызывает глобальный сброс, как показано ниже
	0	0	Глобальный сброс
	0	1	–
	1	0	Глобальный сброс
	1	1	Глобальный сброс
13-8	Резерв		Читаются как неопределенное значение. Запись в них эффекта не имеет
7-6	CLKSRC1	CLKSRC0	Управление/Выбор источника сигнала для вывода CLKOUT
	0	0	Режим цифрового ввода - вывода (направление определяется регистром управления ввода - вывода, IOPC1)
	0	1	Выход сторожевого таймера, стандартно 16 КГц. (WDCLK)
	1	0	Системный тактовый сигнал (SYSCLK)
	1	1	Тактовый сигнал ЦП (CPUCLK)
5-0	Резерв		Читаются как неопределенное значение. Запись в них эффекта не имеет

Системный статусный регистр (SYSSR)

Биты 15, 12 и 9 системного статусного регистра служат для индикации причины сброса. Чтение этих разрядов в программе обработки прерывания позволяет выполнить подходящее действие, которое соответствует причине сброса. Например, если произошел сброс по включению питания, то регистры модуля управления генератором могут быть реконфигурированы. На рисунке 250 представлен формат системного статусного регистра (SYSSR), описание разрядов этого регистра приведено в таблице 111.

15	14-13	12	11	10	9	8-6	5	4-1	0
PORST	Резерв	ILLADR	Резерв	SWRST	WDRST	Резерв	HPO	Резерв	VECRD
R/C-x		R/C-x		R/C-x	R/C-x		R-i		R-0

Примечание – R – доступ на чтение; W – доступ на запись; через тире (–) значение бита после сброса; x означает, что сброс не оказывает влияния; I – значение V_{CCP}, защелкнутое по положительному фронту сиг нала RESET.

Рисунок 250 – Статусный регистр системы (SYSSR) – адрес 701Ah

Таблица 111 – Описание бит регистра

Бит	Мнемоника	Описание
15	PORST	Статусный бит сброса по включению питания. Устанавливается в 1 при сбросе по включению питания. PORST=0, нет сброса по питанию. PORST=1, сброс по питанию
14-13	Res	Резерв. Читаются как неопределенное значение. Запись в них эффекта не имеет
12	ILLADR	ILLADR=0, нет обращения к несуществующему адресу. ILLADR=1, есть обращение к несуществующему адресу
11	Res	Резерв. Читается как неопределенное значение. Запись в него эффекта не имеет
10	SWRST	Статусный бит программного сброса. SWRST=0, нет программного сброса. SWRST=1, произошел программный сброс. (В 15 бит SYSCR была записана 1 или 0 был записан в 14 бит SYSCR)
09	WDRST	Статусный бит сброса от сторожевого таймера. WDRST=0, нет сброса WDRST=1, сброс по переполнению сторожевого таймера
08-06	Res	Резерв. Читаются как неопределенное значение. Запись в них эффекта не имеет
05	HPO	Отмена аппаратной защиты. Устанавливается в 1, если на вывод запрещение сторожевого таймера (WDDIS) подано +3,3 В на заднем фронте вывода сброс (RESET#) и это значение удерживается. Этот бит сбрасывается в 0 программно или если вывод WDDIS изменяется на 0 В. HPO=0, нормальный режим работы. HPO=1, разрешено программирование флэш-памяти и сторожевой таймер запрещается установкой бита WDDIS в управляющем регистре сторожевого таймера (WD)

Окончание таблицы 111

Бит	Мнемоника	Описание
01-04	Res	Резерв. Читается как неопределенное значение. Запись в него эффекта не имеет
00	VECRD	Бит ожидания чтения вектора прерывания. Этот бит устанавливается, когда вектор прерывания загружается в SYSIVR (когда прерывание подтверждается). Сбрасывается при чтении SYSIVR. Этот бит используется при обработке немаскируемого прерывания NMI (см. Обработка прерывания NMI). VECRD=0, чтение вектора из SYSIVR выполнено. VECRD=1, вектор, который загружен в SYSIVR, еще не прочитан

Регистр системного вектора прерывания

На рисунке 251 представлен регистр системного вектора прерывания (SYSIVR), описание бит этого регистра дано в таблице 112.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSBSIV								LSBSIV							
R-0															

Примечание – R – доступ на чтение; 0 – значение после сброса.

Рисунок 251 – Регистр системного вектора прерывания (SYSIVR) – адрес 701Eh

Таблица 112 – Описание бит регистра

Бит	Мнемоника	Описание
15-08	MSBSIV	Восемь старших значащих бит вектора системного прерывания. Эти биты всегда читаются как 0
07-00	LSBSIV	Восемь младших значащих бит вектора системного прерывания. Эти биты загружаются значением смещения адреса вектора прерывания. Это значение генерируется периферией, подключенной к периферийной шине, в ответ на подтверждение соответствующего маскируемого прерывания

14.6.3 Прерывания

Аппаратные или программные прерывания вызывают в ИС 1867ВЦ10Т приостановку его основной программы и выполнение подпрограммы. Обычно, прерывания генерируются аппаратными устройствами для осуществления обмена данными (к примеру – АЦП). Прерывания могут также использоваться для сигнализации об имевшем место конкретном событии (например, окончание счета таймера).

ИС 1867ВЦ10Т поддерживает и аппаратные, и программные прерывания:

– программное прерывание запрашивается инструкцией (INTR, NMI или TRAP);

– аппаратное прерывание запрашивается сигналом от физического устройства.

Существует два типа аппаратных прерываний:

– внешние аппаратные прерывания, инициируются сигналами с внешних входов (с программируемой полярностью). Полярность и приритетность определяются регистрами управления внешними прерываниями;

– внутренние аппаратные прерывания инициируются сигналами от внутрикристальной периферии.

Если аппаратные прерывания инициируются одновременно, то ИС 1867ВЦ10Т обслуживает их в соответствии с установленным приоритетом. Каждое из прерываний ИС 1867ВЦ10Т, аппаратное или программное, может быть помещено в одну из категорий:

– маскируемые прерывания. Эти аппаратные прерывания могут программно блокироваться (маскироваться) или разрешаться (размаскироваться);

– немаскируемые прерывания. Эти прерывания не могут блокироваться. ИС 1867ВЦ10Т всегда реагирует на эти прерывания и передает управление от главной программы к подпрограмме обработки прерывания. К немаскируемым относятся все программные прерывания и два внешних прерывания – это (RESET) и (NMI).

Примечание – Прерывание по сбросу (RESET) всегда имеет активный низкий уровень, в отличие от NMI, полярностью которого можно управлять программно.

В таблице 113 приведены все прерывания ядра ИС 1867ВЦ10Т. Остальные маскируемые прерывания доступны через внутрикристальную периферию. Взаимосвязь между маскируемыми прерываниями ЦП (INT1 – INT6) и маскируемыми периферийными прерываниями описана в этом разделе.

Таблица 113 – Размещение адресов векторов и приоритет прерываний

К	Вектор	Мнемоника	Приоритет	Описание
0	0h	RESET#	1 (высший)	Аппаратный сброс (немаскируемое)
1	2h	INT1	4	Маскируемое прерывание уровня #1
2	4h	INT2	5	Маскируемое прерывание уровня #2
3	6h	INT3	6	Маскируемое прерывание уровня #3
4	8h	INT4	7	Маскируемое прерывание уровня #4
5	Ah	INT5	8	Маскируемое прерывание уровня #5
6	Ch	INT6	9	Маскируемое прерывание уровня #6
7	Eh	-	10	Резерв
8	10h	INT8	-	Определенное пользователем программное прерывание
9	12h	INT9	-	Определенное пользователем программное прерывание
10	14h	INT10	-	Определенное пользователем программное прерывание

Продолжение таблицы 113

K	Вектор	Мнемоника	Приоритет	Описание
11	16h	INT11	-	Определенное пользователем программное прерывание
12	18h	INT12	-	Определенное пользователем программное прерывание
13	1Ah	INT13	-	Определенное пользователем программное прерывание
14	1Ch	INT14	-	Определенное пользователем программное прерывание
15	1Eh	INT15	-	Определенное пользователем программное прерывание
16	20h	INT16	-	Определенное пользователем программное прерывание
17	22h	TRAP	-	Прерывание по инструкции TRAP
18	24h	NMI	3	Резерв
19	26h	-	2	Определенное пользователем программное прерывание
20	28h	INT20	-	Определенное пользователем программное прерывание
21	2Ah	INT21	-	Определенное пользователем программное прерывание
22	2Ch	INT22	-	Определенное пользователем программное прерывание
23	2Eh	INT23	-	Определенное пользователем программное прерывание
24	30h	INT24	-	Определенное пользователем программное прерывание
25	32h	INT25	-	Определенное пользователем программное прерывание
26	34h	INT26	-	Определенное пользователем программное прерывание
27	36h	INT27	-	Определенное пользователем программное прерывание
28	38h	INT28	-	Определенное пользователем программное прерывание
29	3Ah	INT29	-	Определенное пользователем программное прерывание
30	3Ch	INT30	-	Определенное пользователем программное прерывание
31	3Eh	INT31	-	Определенное пользователем программное прерывание
Примечание – K – это операнд в инструкции INTR, который определяет соответствующую ячейку вектора прерывания.				

Три фазы операции прерывания

Процесс обработки прерывания можно разбить на три основные фазы:

1 Прием запрос прерывания.

2 Подтверждение прерывания. ИС 1867ВЦ10Т должна подтвердить запрос на прерывание. Для подтверждения маскируемых прерываний должны быть удовлетворены определенные условия. Немаскируемые аппаратные прерывания и программные прерывания подтверждаются немедленно.

3 Выполнение подпрограммы обработки прерывания. После подтверждения прерывания, ЦП переходит к выполнению подпрограммы обработки прерывания, называемой ISR (Interrupt service routine). Процессор аппаратно обращается к ячейке с вектором соответствующего прерывания, в которую пользователь заранее помещает инструкцию перехода на стартовый адрес, написанной им ISR.

Немаскируемые прерывания

Немаскируемые аппаратные прерывания могут запрашиваться через два вывода:

– RESET (сброс). Это прерывание немедленно, не дожидаясь завершения работы текущей инструкции, останавливает программный поток, возвращает процессор в предопределенное состояние и, после этого, передает управление на ячейку с адресом 0000h программной памяти.

– NMI. Это прерывание используется как «мягкий» сброс. В отличие от аппаратного сброса, NMI не изменяет режимы работы ЦП или периферии, ожидает окончания исполнения текущей инструкции или операции с памятью. Хотя NMI использует ту же логику, что и маскируемое прерывание, но это прерывание не маскируется. NMI прерывание происходит независимо от значения бита INTM. Маска для этого прерывания отсутствует. NMI блокируется только уже выполняющимся NMI или сбросом. Когда NMI прерывание активизируется (или выводом NMI# или инструкцией NMI), процессор передает управление вектору этого прерывания, размещенному в ячейке программной памяти с адресом 24h. При активизации NMI аппаратно устанавливается в 1 бит INTM в статусном регистре ST0, запрещая маскируемые прерывания.

Программные прерывания являются по существу немаскируемыми и вызываются следующими инструкциями:

– INTR. Эта инструкция позволяет инициировать любое прерывание, включая прерывания INT8–INT16 и INT20–INT31, определенные пользователем (user-defined interrupts). Операнд K в инструкции определяет, к какому вектору выполнит переход процессор. В таблице 113 показывается соответствие между значением K и расположением ячейки вектора прерывания.

– NMI. Эта инструкция вызывает переход к вектору прерывания в ячейке 24h (та же ячейка используется для немаскируемого аппаратного прерывания NMI). Таким образом, немаскируемое прерывание может быть инициировано как посредством входа (NMI#), так и выполнением одноименной инструкции. Как и при

аппаратном немаскируемом прерывании, инструкция NMI устанавливает в 1 бит INTM в статусном регистре ST0, запрещая маскируемые прерывания.

– TRAP. Эта инструкция вызывает переход к вектору прерывания в ячейке с адресом 22h. Инструкция TRAP не запрещает маскируемые прерывания (INTM не устанавливается в 1); таким образом, исполнение подпрограммы обработки TRAP может быть прервано аппаратным маскируемым прерыванием.

После подтверждения немаскируемого прерывания процессор:

1 Сохраняет значение PC (адрес возврата) в верхушке аппаратного стека.

2 Загружает PC адресом вектора прерывания.

3 Производит выборку инструкции перехода, хранящейся в векторе прерывания. Если прервание было аппаратным или было инициировано программно (инструкцией INTR или NMI), устанавливает бит INTM в 1, запрещая маскируемые прерывания.

4 Выполняет переход к адресу подпрограммы обработки прерывания.

5 Выполняет ISR до тех пор, пока не встретится инструкция возврата. Если INTM = 1, все маскируемые прерывания запрещены во время выполнения этой подпрограммы.

6 Читает адрес возврата из верхушки стека в PC.

7 Продолжает выполнение основной программы.

Карта векторов прерываний приведена в таблице 114. Каждый вектор занимает две ячейки, позволяя разместить в них двухсловную инструкцию перехода к ISR.

Структура маскируемых прерываний

Ядро процессора обеспечивает шесть уровней маскируемых прерываний. ИС 1867ВЦ10Т имеет большее количество источников маскируемых прерываний, поэтому каждый из шести уровней используется совместно несколькими источниками прерываний. Рисунок 252 иллюстрирует структуру, используемую для получения и подтверждения маскируемых прерываний. На этом рисунке показаны четыре источника прерываний (XINT1, XINT2, XINT3 и RTI), совместно использующие уровень прерывания INT1. Подобно этому реализованы и остальные уровни прерываний (INT2 - INT6).

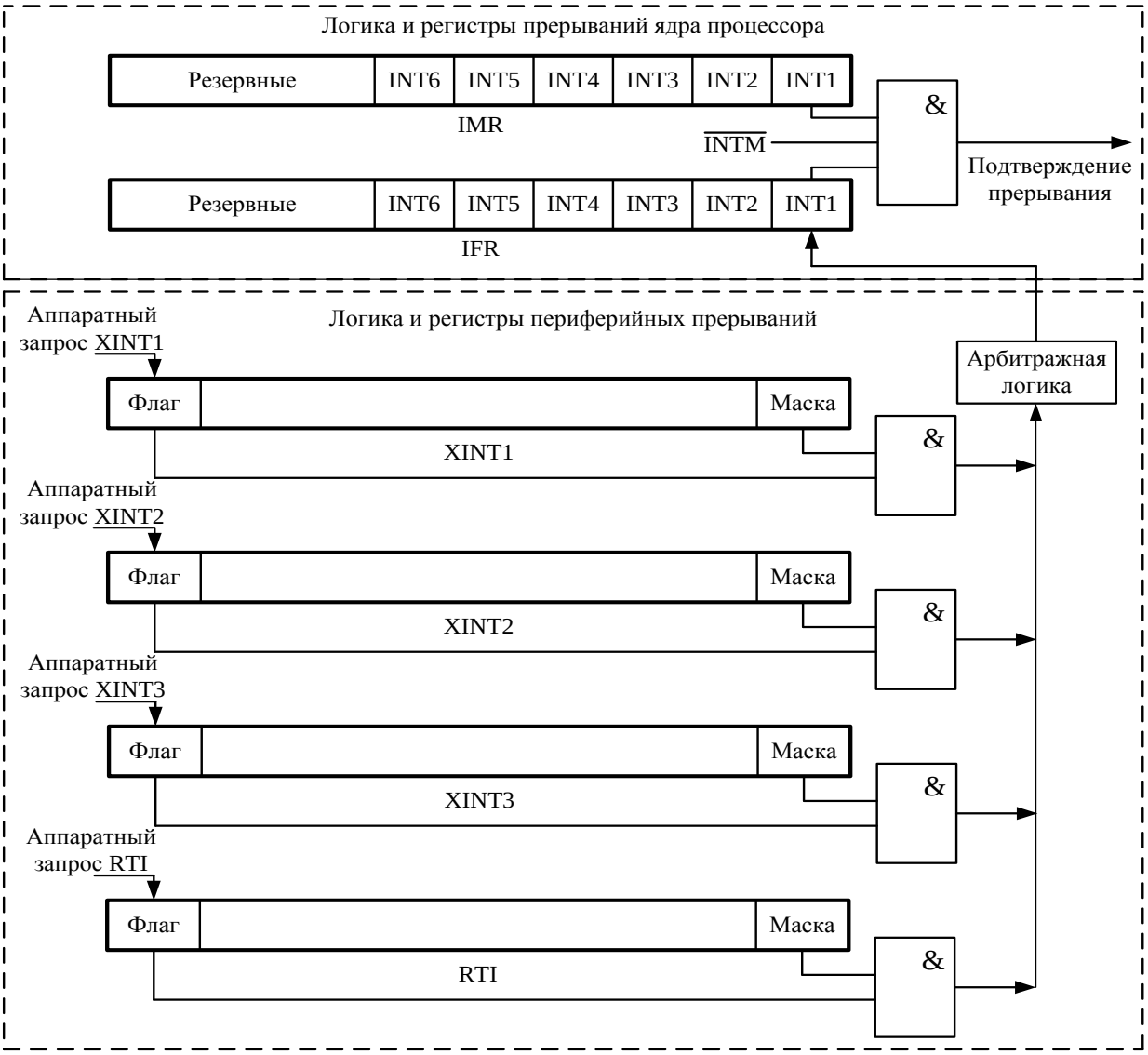


Рисунок 252 – Пример структуры маскируемых прерываний

Передача запроса маскируемого прерывания

Каждый источник прерывания имеет свой собственный управляющий регистр с битом флага и маскирующим битом (см. 14.6.3 «Прерывания»). Когда сигнал прерывания получен, устанавливается в 1 флаг в соответствующем регистре управления, индицируя запрос прерывания. Если бит маски также установлен, то сигнал поступает в блок арбитражной логики. В этот блок могут одновременно поступить аналогичные сигналы от одного или нескольких других регистров управления. Арбитражная логика сравнивает уровни приоритета конкурирующих запросов, выбирает прерывание с наивысшим приоритетом и передает его сигнал в ядро процессора. Этот сигнал, в свою очередь, устанавливает в 1 флаг прерывания соответствующего уровня в регистре флагов прерывания ядра процессора (IFR), показывая наличие ожидающего (необработанного) прерывания. Если в регистре маски прерывания ядра процессора (IMR) соответствующий бит установлен в 1, и прерывания глобально незамаскированы (INTM = 0), то процессор подтверждает прерывание и передает управление ISR.

Приоритеты маскируемых прерываний

Все аппаратные прерывания имеют ранг приоритета от 1 до 10 (наивысший приоритет – 1). Приоритеты приведены в таблице 113. Если в ожидании обслуживания находится несколько необработанных прерываний, первым будет обрабатываться прерывание с более высоким приоритетом. Затем, в порядке их приоритета, будут обслуживаться другие прерывания. Приоритеты уровней маскируемых прерываний ядра процессора приведены в таблице 114.

Таблица 114 – Приоритеты уровней маскируемых прерываний в ЦП

Уровень маскируемого прерывания	Приоритет в ядре процессора
INT1	4
INT2	5
INT3	6
INT4	7
INT5	8
INT6	9

В качестве примера, если в ожидании обслуживания находятся прерывания INT1 и INT2 и оба они не замаскированы, тогда прерывание уровня INT1 будет обслуживаться первым, а INT2 вторым.

К каждому уровню маскируемых прерываний (INT1 – INT6) подключено несколько источников, которые также имеют набор рангов приоритета. Источник с высшим приоритетом посылает запрос на свой уровень прерывания первым. В примере на рисунке 253 рассмотрены источники прерываний XINT1, XINT2, XINT3 и RTI. Эти прерывания имеют ранги приоритета по отношению к INT1, как показано в таблице 115.

Таблица 115 – Приоритетные ранги под INT1

Маскируемый источник прерывания	Приоритет под INT1
XINT1	1
XINT2	2
XINT3	3
RTI	4

Если все источники сгенерировали прерывание в одно и то же время, то XINT1 получит обслуживание первым, затем XINT2, XINT3 и, наконец, RTI.

Регистры прерывания процессорного ядра

В состав процессорного ядра ИС 1867ВЦ10Т входят два регистра, предназначенные для управления прерываниями:

– Регистр флагов прерываний (IFR), содержащий флаговые разряды запросов прерываний (INT1 - INT6), поступивших в процессор.

– Регистр маски прерывания (IMR), позволяющий индивидуально для каждого из уровней (INT1 - INT6) запрещать или разрешать прерывания.

Регистр флагов прерываний (IFR)

IFR - это 16-битный регистр, картированный в памяти данных по адресу 0006h, который используется для идентификации ожидающих обработки прерываний. Он содержит флаговые разряды для всех маскируемых прерываний.

Механизм передачи сигналов запроса периферийных маскируемых прерываний от какого-либо устройства до соответствующего уровня регистра IFR флагов прерываний ядра процессора (через имеющиеся в этом устройстве собственные регистры флагов и масок; через арбитражную логику) проиллюстрирован на примере уровня INT1 на рисунке 254 и подробно описан в 14.6.3. Когда сигналы запросов маскируемых прерываний доходят до IFR, в нем устанавливаются в 1 биты флагов соответствующих уровней, индицируя наличие ожидающих подтверждения и обработки прерываний.

IFR доступен на чтение для идентификации этих прерываний и на запись для их очистки (т.е. для сброса их флагов). Необходимо отметить, что запись в IFR, как в ячейку памяти с адресом 0006h, осуществляется в инверсном виде. Поэтому, чтобы очистить конкретный бит флага прерывания, необходимо записать в соответствующий разряд единицу; очистку всех ожидающих подтверждения и обработки прерываний можно осуществить путем записи в IFR его же содержимого. Сброс процессора очищает все разряды IFR. Речь в данном случае идет о неподтвержденных прерываниях. Если же прерывание подтверждено, то одновременно с началом обработки, его флаг сбрасывается автоматически.

Примечания

1 Для сброса бита IFR нужно записать его 1, а не 0.

2 Когда маскируемое прерывание подтверждено, то только бит в IFR сбрасывается автоматически. Разряд флага в соответствующем периферийном регистре не очищается. Если приложение требует сброса флага в устройстве, это необходимо сделать програмно.

3 Если запрос прерывания сформирован инструкцией INTR, и в IFR установлен соответствующий флаг, то процессор не очищает его автоматически. Если приложение требует, чтобы разряд IFR был очищен, это необходимо сделать програмно.

Регистр IFR показан на рисунке 253, описание его разрядов приведено в таблице 116.

15–6	5	4	3	2	1	0
Резерв	INT6	INT5	INT4	INT3	INT2	INT1
0	R/W1C–0	R/W1C–0	R/W1C–0	R/W1C–0	R/W1C–0	R/W1C–0

Примечание – 0 – всегда читается как 0,
R – доступ на чтение,
W1C – запись 1 очищает этот бит,
через тире (–) значение бита после сброса.

Рисунок 253 – Регистр флагов прерывания (IFR) – адрес 0006h

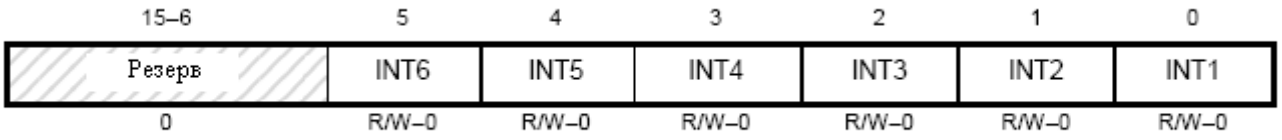
Таблица 116 – Описание регистра IFR

Бит	Мнемоника	Описание
15-06	Res	Резерв
05	INT6	Флаг прерывания 6. Бит флага подключен к уровню прерывания 6. INT6=0, отсутствие запроса прерывания. INT6=1, по меньшей мере одно прерывание этого уровня выставило запрос. Запись 1 в этот бит очищает этот бит и сбрасывает запрос прерывания.
04	INT5	Флаг прерывания 5. Бит флага подключен к уровню прерывания 5. INT5=0, отсутствие запроса прерывания. INT5=1, по меньшей мере одно прерывание этого уровня выставило запрос. Запись 1 в этот бит очищает этот бит и сбрасывает запрос прерывания.
03	INT4	Флаг прерывания 4. Бит флага подключен к уровню прерывания 4. INT4=0, отсутствие запроса прерывания. INT4=1, по меньшей мере одно прерывание этого уровня выставило запрос. Запись 1 в этот бит очищает этот бит и сбрасывает запрос прерывания.
02	INT3	Флаг прерывания 3. Бит флага подключен к уровню прерывания 3. INT3=0, отсутствие запроса прерывания. INT3=1, по меньшей мере одно прерывание этого уровня выставило запрос. Запись 1 в этот бит очищает этот бит и сбрасывает запрос прерывания.
01	INT2	Флаг прерывания 2. Бит флага подключен к уровню прерывания 2. INT2=0, отсутствие запроса прерывания. INT2=1, по меньшей мере одно прерывание этого уровня выставило запрос. Запись 1 в этот бит очищает этот бит и сбрасывает запрос прерывания.
00	INT1	Флаг прерывания 1. Бит флага подключен к уровню прерывания 1. INT1=0, отсутствие запроса прерывания. INT1=1, по меньшей мере одно прерывание этого уровня выставило запрос. Запись 1 в этот бит очищает этот бит и сбрасывает запрос прерывания.

Регистр маски прерываний (IMR)

IMR – это 16-битный регистр, картированный в памяти данных по адресу 0004h. IMR содержит биты масок для всех уровней маскируемых прерываний (INT1 - INT6). Ни NMI, ни RESET не включены в IMR, и поэтому IMR не влияет на эти прерывания.

IMR доступен на чтение для идентификации текущей маски и на запись для разрешения или запрета прерываний тех или иных уровней. Чтобы разрешить прерывание, соответствующий бит этого регистра надо установить в 1; 0 означает, что прерывание данного уровня запрещено. Запрос запрещенного (замаскированного) прерывания не подтверждается и не обрабатывается, независимо от состояния бита INTM. Когда прерывание разрешено (немаскировано), то, при установке соответствующего бита IFR в 1, и при условии, что INTM = 0, его запрос подтверждается. При сбросе все разряды регистра IMR устанавливаются в 0, запрещая все маскируемые прерывания. Формат IMR приведен на рисунке 254, таблица 117 содержит описание его разрядов.



Примечание – 0 – всегда читается как 0; R – доступ на чтение, W – доступ на запись; через тире (–) значение бита после сброса.

Рисунок 254 – Регистр масок прерываний (IMR) – адрес 0004h

Таблица 117 – Описание разрядов регистра IMR

Бит	Мнемоника	Описание
15-06	Res	Резерв. Эти биты читаются как 0.
05	INT6	Маска 6-го прерывания. Разрешает или запрещает прерывания уровня INT6. INT6=0, уровень INT6 маскирован. INT6=1, уровень INT6 не маскирован.
04	INT5	Маска 5-го прерывания. Разрешает или запрещает прерывания уровня INT5. INT5=0, уровень INT5 маскирован. INT5=1, уровень INT5 не маскирован.
03	INT4	Маска 4-го прерывания. Разрешает или запрещает прерывания уровня INT4. INT4=0, уровень INT4 маскирован. INT4=1, уровень INT4 не маскирован.
02	INT3	Маска 3-го прерывания. Разрешает или запрещает прерывания уровня INT3. INT3=0, уровень INT3 маскирован. INT3=1, уровень INT3 не маскирован.
01	INT2	Маска 2-го прерывания. Разрешает или запрещает прерывания уровня INT2. INT2=0, уровень INT2 маскирован. INT2=1, уровень INT2 не маскирован.
00	INT1	Маска 1-го прерывания. Разрешает или запрещает прерывания уровня INT1. INT1=0, уровень INT1 маскирован. INT1=1, уровень INT1 не маскирован.

Подтверждение и обслуживание маскируемых прерываний

После запроса прерывания, выставленного аппаратурой или инициированного программно, логика обработки прерываний принимает решение о подтверждении или отклонении этого запроса. Если запрос подтвержден, процессор приступает к выполнению подпрограммы обработки прерывания, называемой ISR (Interrupt service routine).

Подтверждение маскируемых прерываний

Подтверждение программных и немаскируемых аппаратных прерываний происходит немедленно после поступления запроса на их обработку. Маскируемые прерывания подтверждаются только при соблюдении следующих условий:

– Приоритет высший. Когда запросы от нескольких прерываний поступают одновременно, ИС 1867ВЦ10Т обслуживает каждый запрос в соответствии с установленными рангами приоритетов (см. таблицу 113).

– Бит NTM = 0. Бит режима прерывания (бит 9 статусного регистра ST0) глобально разрешает или запрещает все маскируемые прерывания. При INTM = 0, все маскируемые прерывания разрешены. При INTM = 1, все маскируемые прерывания запрещены.

При подтверждении прерывания бит INTM устанавливается в 1 автоматически, (за исключением прерывания, инициированого инструкцией TRAP). INTM также устанавливается в 1 при аппаратном сбросе или инструкцией запрета прерываний (SETC INTM). INTM сбрасывается в 0 выполнением инструкции разрешения прерываний (CLRC INTM). Бит INTM не влияет на сброс, NMI или программные прерывания. Инструкция LST (загрузка статусного регистра) не изменяет этот бит.

Сброс или установка бита INTM не модифицирует содержимое ни регистра IMR, ни регистра IFR.

– Бит маски IMR равен 1. Каждый из уровней маскируемых прерываний имеет бит маски в IMR. Чтобы разрешить прерывание того или иного уровня необходимо записать 1 в соответствующий бит IMR.

Когда процессор подтверждает незамаскированное аппаратное прерывание, на шине инструкций аппаратно выставлется код команды INTR. Выполнение этого кода прводит к загрузке в PC соответствующего адреса, из которого производится выборка вектора прерывания.

Две части подпрограммы обработки прерывания

Процессорное ядро 1867ВЦ10Т имеет шесть уровней прерываний, но поскольку количество имеющихся в микросхеме прерываний больше шести, предусмотрен способ создания соответствующего количества ISR. Рисунок 255 иллюстрирует процесс обслуживания запроса прерывания. Для каждого из 6 уровней прерывания процессорное ядро переходит к соответствующей главной подпрограмме обслуживания прерывания (GISR – general interrupt service routine). Например, когда запрашивается прерывание уровня INT1, процессор осуществляет пере-

ход к GISR1 и ее исполнению. Главная подпрограмма после сохранения необходимого контекста осуществляет точное распознавание прерывания и после этого переходит к специальной подпрограмме обработки прерывания (SISR – specific interrupt service routine). В специальной подпрограмме выполняются действия, индивидуальные для этого прерывания, и после их выполнения осуществляется возврат управления к прерванной программной последовательности.

Переход к GISR. В таблице 118 показываются адреса и содержимое ячеек векторов прерываний для уровней маскируемых прерываний (все ячейки векторов прерываний приведены в таблице 113). При подтверждении прерывания для одного из уровней, процессор выполняет переход к соответствующему адресу вектора и затем переход к адресу GISR. Например, при подтверждении прерывания уровня INT3, PC сохраняется в стеке и после этого загружается содержимым программной памяти по адресу 0006h. В ячейках 0006h и 0007h содержится инструкция перехода к GISR.

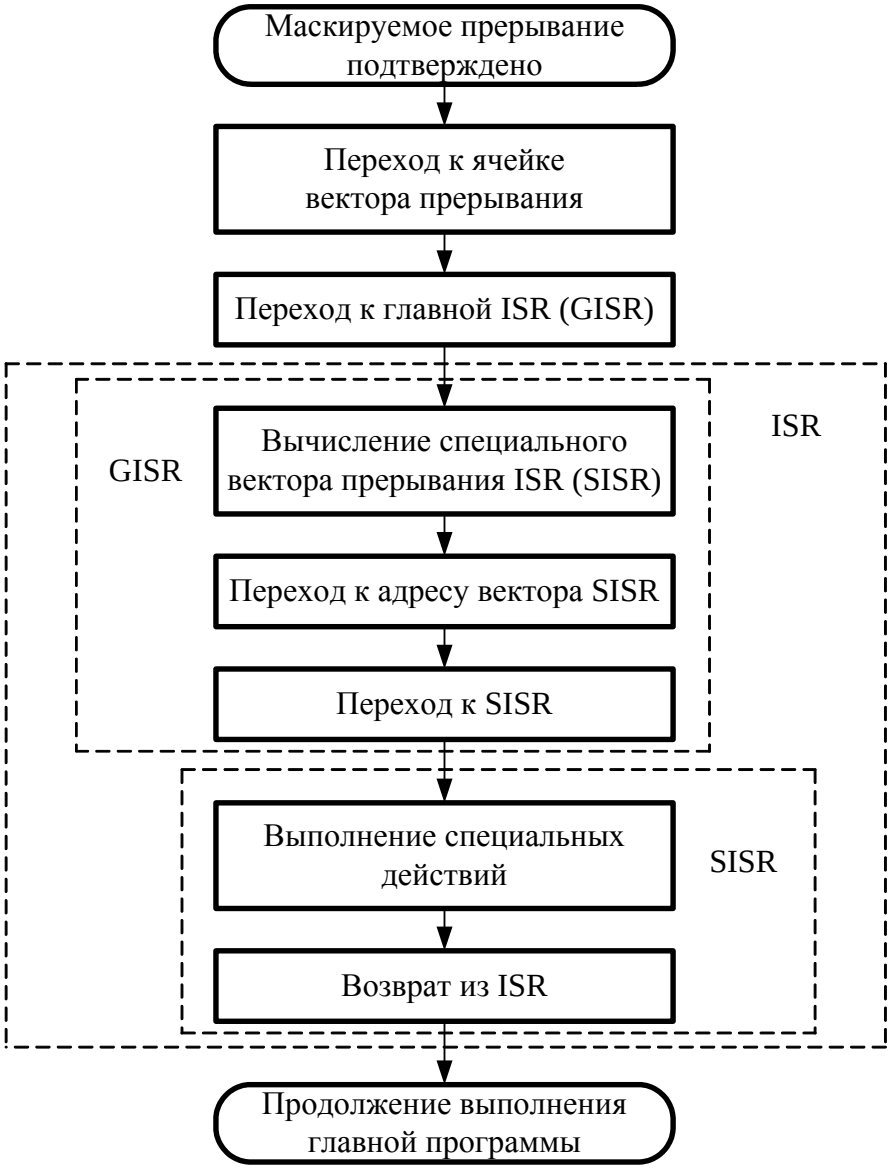


Рисунок 255 – Диаграмма обслуживания прерывания

Таблица 118 – Векторы маскируемых прерываний

Уровень прерывания	Ячейка вектора прерывания	Содержимое ячейки вектора прерывания
INT1	0002h	Переход к GISR1
INT2	0004h	Переход к GISR2
INT3	0006h	Переход к GISR3
INT4	0008h	Переход к GISR4
INT5	000Ah	Переход к GISR5
INT6	000Ch	Переход к GISR6

Переход к SISR. При подтверждении запроса периферийного прерывания (включая управляющие регистры внешних прерываний), периферия генерирует смещение адреса вектора, которое соответствует этому событию прерывания. Это смещение вектора прерывания обычно записывается в (SYSIVR – system interrupt vector register), хотя некоторая периферия, в том числе менеджер событий, может хранить это смещение в периферийном регистре. Более детельно размещение регистров векторов прерываний (IVR – interrupt vector register) для каждого уровня прерывания, а также значение вектора для каждого события приведено в разделе GISR считывает значение, сохраненное в IVR каждого уровня прерывания, и использует его для генерации адреса перехода к SISR.

Векторы прерываний

В таблице 119 приведен пример адресов векторов прерывания ИС 1867ВЦ10Т. Заметьте, что большинство периферийных устройств использует регистр SYSIVR для выборки смещения адреса вектора прерывания, но менеджер событий имеет свой собственный регистр для каждого из его трех уровней приоритета запроса прерывания:

- EVIVRA для прерываний группы A;
- EVIVRB для прерываний группы B;
- EVIVRC для прерываний группы C.

Таблица 119 – Расположение адресов векторов прерывания и приоритеты

Имя прерывания	Приоритет	Прерывание ЦП. Адрес прерывания	Периферийный адрес вектора	Смещение периферийного адреса вектора	Маскируемое прерывание	Источник прерывания	Функция
RESET#	1	RESET# 0000h	Недоступен	Недоступен	Нет	Вывод, S/W, модуль сторожевого таймера	Внешний системный сброс (RESET)

Имя прерывания	Приоритет	Прерывание ЦП. Адрес прерывания	Периферийный адрес вектора	Смещение периферийного адреса вектора	Маскируемое прерывание	Источник прерывания	Функция
Reserved	2	0026h	Недоступен		Нет	ЦП	TRAP

Продолжение таблицы 119

Имя прерывания	Приоритет	Прерывание ЦП. Адрес прерывания	Периферийный адрес вектора	Смещение периферийного адреса	Маскируемое	Источник прерывания	Функция
NMI	3	NMI 0024h	Недоступен	Недоступен	Нет	Внешний вывод	Внешнее прерывание
XINT1	4	INT1 (Система)	SYSIVR 701Eh	0002h	Да	Внешний вывод	Высокоприоритетное внешнее прерывание
XINT2	5			0011h			
XINT3	6			001Fh			
RTI	10			0010h		Модуль сторожевого таймера	Прерывание модуля сторожевого таймера
PDPINT#	11	INT2 0004h (EVINTA)	EVIVRA 7432h	0020h	Да	Внешний вывод	Прерывание устройства от защиты по напряжению
CMP1INT	12	Группа A		0021p	Да	Модуль менеджера событий	Сравнение 1 Прерывание
TPINT2	22	INT3 0006h	EVIVRB 7434h	002Bh	Да	Модуль менеджера событий	Таймер 2. Прерывание по периоду
TCINT2	23	Группа B		002Ch	Да	Модуль менеджера событий	Таймер 2. Прерывание по сравнению
CAPINT1	30	INT4 0008h (EVINTC)	EVIVRC	0033h	Да	Модуль менеджера событий	Захват 1 Прерывание

CAPINT2	31	Группа C		0034h	Да	Модуль менеджера событий	Захват 2 Прерывание
CAPINT3	32			0035h	Да	Модуль менеджера событий	Захват 3 Прерывание

Окончание таблицы 119

Имя прерывания	Приоритет	Прерывание ЦП. Адрес прерывания	Периферийный адрес вектора	Смещение периферийного адреса	Маскируемое прерывание	Источник прерывания	Функция
CAPINT4	33			0038h	Да	Модуль менеджера событий	Захват 4 Прерывание
и т.д.							
ADCINT	37	INT6 000Ch	SYSIVR 701Eh	0004h	Да	Модуль АЦП	АЦП Прерывание
XINT1	38			0002h	Да	Внешний вывод	Низкоприоритетное внешнее прерывание
XINT2	39			0011h	Да		
XINT3	40			001Fh	Да		
Reserved	41	- 000Eh	Недоступен		Да	ЦП	Используется для анализа
TRAP	n/a	0022h			Недоступен		

Искусственный вектор прерывания

Искусственный вектор прерывания (PIV - Phantom Interrupt Vector) – это особенность системной интеграции прерывания. В событии, в котором прерывание подтверждается, но периферия не отвечает загрузкой значения смещения адреса вектора прерывания в регистр вектора прерывания (IVR), искусственный вектор (0000h) загружается в IVR.

Это вызывает искусственное прерывание:

– выполнение инструкции INTR с аргументом от 1 до 6. Это есть программный запрос на обслуживание одного из 6 маскируемых уровней прерывания (INT1, INT2, INT3, INT4, INT5, or INT6);

– сбой на линии запроса прерывания. В другом случае, когда прерывание подтверждается и периферия загружает вектор в IVR.

Загрузка IVR искусственным вектором прерывания гарантирует, что ЦП перейдет к известной ячейке.

Программирование ISR (программ обработки прерывания) для маскируемых прерываний

В этом разделе приведены три метода для создания программы обслуживания прерывания (ISR) для маскируемого прерывания.

– Метод 1. Типичный ISR.

– Метод 2. ISR, который проектируется для уменьшения времени ожидания временем, когда прерывание запрашивается и временем выполнения специальной секции ISR.

– Метод 3. ISR, которая имеет очень маленькое время задержки. Он используется в случае, если единственный периферийный источник прерывания подсоединяется к запросу прерывания приоритетного уровня (interrupt request priority level) или если только один из множества источников прерывания данного приоритетного уровня разрешен.

Типичный ISR

В методе 1 программа обработки прерывания разбивается на два сегмента:

1 Основной ISR (GISR). Когда прерывание на одном из 6 уровней прерывания (INT1–INT6) подтверждается, то GISR читает соответствующий регистр IVR, сдвигает влево на один бит и добавляет смещение к этому значению, и после этого переходит к таблице векторов прерывания периферии. Из этой таблицы программа выбирает и выполняет подходящий адрес перехода к специальной ISR (SISR).

2 Специальная ISR (SISR). Выполняет специальные действия для события, которое вызвало прерывание и после этого возвращает управление к прерванной последовательности кода.

В таблице 120 приведен пример метода 1.

Таблица 120 – Метод 1

Число циклов	Адрес	Ассемблерный код	Комментарий
Таблица векторов прерывания ЦП			
0	0006h 0007h …	INT3: B GISR3 ; …	;Переход к адресу главной ISR для INT3 ; ;
4	GISR3 Addr	GISR3: LACC xIVR, 1	;Загрузка аккумулятора содержимым регистра xIVR

Число циклов	Адрес	Ассемблерный код	Комментарий
4+n*	GISR3 Addr+1	ADD offset	;Добавление смещения к аккумулятору
5+n*	GISR3 Addr+1	BACC	;Переход к адресу в аккумуляторе (2*IVR+offset)
…			
9+n*	2*IVR+offset (2*IVR+offset)+1	B SISRx	;Переход к специальной ISR для события, которое запросило прерывание

Продолжение таблицы 120

Число циклов	Адрес	Ассемблерный код	Комментарий
…			
13+n*	SISRx Addr SISRx Addr+1 …. SISRx Addr + n	SISRx: RET:	;Выполнение специальных действий ;Возврат из ISR

* Для периферийного интерфейса n = 2, для менеджера событий n = 1.

В данном примере происходят следующие события:

1 Некоторое периферийное устройство устанавливает INT3 и заставляет ЦП ввести его таблицу векторов прерывания в адрес 0006h.

2 От адреса 0006h по 0007h ЦП читает переход, который следует к GISR3.

3 GISR3 загружает аккумулятор содержимым соответствующего регистра IVR, сдвинутым влево на 1 разряд (что эквивалентно умножению на 2). Следует отметить:

– инструкция LACC использует прямую адресацию для доступа к IVR. Для этого указатель страниц DP в статусном регистре ST0 должен быть установлен на страницу, в которой находится IVR;

– можно сохранить значение аккумулятора перед загрузкой аккумулятора значением IVR;

– входной вектор умножается на 2, потому, что таблица векторов прерывания периферии должна поддерживать 2 слова для инструкции перехода по каждому периферийному прерыванию.

4 GISR3 добавляет к аккумулятору смещение, которое соответствует началу таблицы векторов прерывания периферии. Теперь аккумулятор содержит адреса, которые нужны для доступа к таблице адресов прерываний.

5 GISR3 переходит к адресу в аккумуляторе.

6 Из ячейки вектора прерывания периферии ЦП читает инструкцию перехода, которая ведет к программе SISR для события, вызвавшего прерывание.

7 SISR выполняется. Так как SISR идет с инструкцией возврата, которая возвращает управление прерванной программной последовательности кода.

Минимальное время ожидания ISR для множества событий на уровне прерывания

Этот метод аналогичен методу 1, но он уменьшает время ожидания, потому что один переход (переход к таблице векторов прерывания периферии) пропускается. Вместо этого GISR переходит прямо к специальной SISR. Чтобы это сделать, GISR сдвигает значение от IBR более, чем на два разряда, и использует этот результат как адрес перехода. Это может затруднить расположение всех SISR в пространстве программной памяти без существования некоторых дыр в этой памяти. Лучше использовать метод 1 и, если для некоторого прерывания время задержки не подходит, то нужно использовать метод 2.

Метод 2 реализуется как следующее:

1 Главная ISR (GISR). Когда прерывание на одном из шести маскируемых уровней прерывания (INT1–INT6) подтверждается, то GISR читает соответствующий регистр вектора прерывания IVR, сдвигает это значение влево на предопределенное количество разрядов и переходит к специальной программе обработки прерывания ISR (SISR). Количество сдвигов выбирается таким образом, чтобы аккумулятор содержал адрес SISR. Для примера, если три источника прерывания привязаны к INT2 и SISR для каждого события не больше 16 слов, тогда сдвиг на четыре разряда будет подходящим.

2 Специальная ISR (SISR). SISR выполняет действия специально для события, которое вызвало прерывание, и после этого возвращает управление прерванной последовательности кода.

В таблице 121 иллюстрируется пример Метода 2.

Таблица 121 – Метод 2

Число циклов	Адрес	Ассемблерный код	Комментарий
Таблица векторов прерывания ЦП			
0	0004h 0005h …	INT2: B ; GISR2 ; …	;Переход к адресу главной ISR для INT2
4	GISR2 Addr	GISR2: LACC xIVR, shift	; ;Загрузка аккумулятора содержимым регистра xIVR
4+n*	GISR2 Addr+1	BACC	;Переход к адресу в аккумуляторе
….			
8+n*	SISRx Addr SISRx Addr+1 …. SISRx Addr + n	SISRx: RET	;Выполнение специальных действий ;Возврат из ISR

_____

* Для периферийного интерфейса n =2, для менеджера событий n = 1.

В этом примере имеют место следующие события:

1 Периферийное устройство устанавливает прерывание INT2 и заставляет ЦП ввести ее таблицу векторов прерывания по адресу 0004h.

2 В адресах с 0004h по 0005h хранится инструкция перехода к ее GISR2.

3 GISR2 загружает аккумулятор содержимым соответствующего регистра IVR, сдвинутого влево на количество разрядов, которое требуется для получения адреса подпрограммы SISR.

Заметьте следующее:

– инструкция LACC использует прямую адресацию для доступа к IVR. Для этого указатель страниц DP в статусном регистре ST0 должен быть установлен на страницу, в которой находится IVR;

– можно сохранить значение аккумулятора перед загрузкой аккумулятора значением IVR;

– входной вектор умножается на 2, потому, что таблица векторов прерывания периферии должна поддерживать 2 слова для инструкции перехода по каждому периферийному прерыванию.

4 GISR2 переходит к адресу, указанному в аккумуляторе (начальный адрес подпрограммы SISR.

5 SISR выполняется. SISR включает инструкцию возврата, которая возвращает управление прерванной последовательности программного кода.

ISR для одного события на один уровень прерывания

ЦП может так конфигурироваться только в том случае, если единственное событие может вызывать конкретное маскируемое прерывание.

Или оба из перечисленных ниже условий должны выполняться:

– только одно периферийное устройство подключается к одному из уровней маскируемых прерываний(INT1, INT2, INT3, INT4, INT5 или INT6);

– это периферийное устройство имеет только одно событие, которое может запросить прерывание (то есть имеет один вектор).

Или только одно из множества событий связано с конкретным приоритетным уровнем прерывания.

И в той, и в другой ситуации нет необходимости во второй части ISR, подобной в методе 1 или методе 2. Имеется только одна простая ISR, которая содержит инструкцию перехода. Адрес ISR известен, то есть он не требует вычисления в подпрограмме. GISR и SISR становятся одним и тем же.

В таблице 122 приведен пример метода 3. В этом примере следующие события имеют место:

1 Периферийное устройство устанавливает INT1 и заставляет ЦП ввести таблицу вектора прерывания по адресу 0002h.

2 С адреса 0002h по 0003h ЦП читает инструкцию перехода, которая ведет прямо к SISR.

3 SISR выполняется, а инструкция возврата возвращает управление прерванной программе.

Таблица 122 – Метод 3

Число циклов	Адрес	Ассемблерный код	Комментарий
Таблица векторов прерывания ЦП			
0	0002h 0003h … SISRx Addr SISRx Addr+1	INT1: B ISR1 … ISR1: …. ….	;Переход к адресу главной ISR для INT2 ; ; ;Выполнение специальных действий
	…. SISRx Addr + n	RET:	;Возврат из ISR

Программирование ISR для немаскируемых прерываний (NMI)

Обычно, не может быть более одного события, способного генерировать запрос NMI, а в случаях, когда событий много, то они все делят одну и ту же ISR. Так, запрос по NMI не требует загрузки регистра вектора прерывания IVR, а переходит к вектору в ячейке 0024h, в которой находится переход только на NMI ISR. NMI может прервать маскируемое ISR после того как вектор загружается в IVR, но перед маскируемым ISR, читающим IVR. Поскольку NMI не загружает IVR, то NMI не вызывает переписывание смещения адреса вектора маскируемого прерывания. Однако разрешенное прерывание, следующее сразу за NMIISR, может переписать значение IVR перед прерванным ISR, который читал его. Следовательно NMIISR должен проверять VECRD (бит 0 в системном статусном регистре (SSR - system status register). Если VECRD = 1, чтение IVR завершается, и NMI ISR не должна разрешать маскируемые прерывания. NMIISR реализовывается как показано в таблице 123.

Дополнительные задачи ISR

Пока выполняются задачи, запрашиваемые прерыванием, ISR может также выполнить:

- сохранение и восстановление регистровых значений;
- управление ISR в пределах ISR.

Таблица 123 – Пример реализации NMIISR

Число циклов	Адрес	Ассемблерный код	Комментарий
Таблица векторов прерывания ЦП			
… 0 4	 0024h 0025h … NMIISRx Addr	 NMI: B NMI_ISR … NMIISR: …. …. BIT SSR, 15 RETC TC	; ; Переход к адресу NMI_ISR ; ; Старт ISR ; Тело ISR ; Проверка бита 0 (VECRD) SYSSR0 ; Если установлен, то возврат из ISR ;(без разрешения прерываний)
	….	CLRC INTM RET:	;Если не установлен, то разрешить ;прерывания ;Возврат из ISR

Сохранение и восстановление регистровых значений

Поскольку только увеличенное значение PC сохраняется автоматически ЦП перед выполнением ISR, то необходимо проектировать ISR таким образом, чтобы сохранить и затем восстановить значение важных регистров или значение управляющих бит. Можно использовать общую подпрограмму или подпрограммы индивидуальные для каждого прерывания, чтобы сохранить контекст процессора во время обработки прерывания. Можно управлять хранением стека так долго, как это возможно, чтобы стек не переполнил выделенную область памяти. Этот стек используется также и для вызова подпрограммы. ИС 1867ВЦ10Т позволяет вызывать подпрограмму из ISR. Инструкции PSHD и POPD могут передавать в память данных значения из и в стек.

Вложенные подпрограммы обработки прерываний (ISR)

Аппаратный стек ИС 1867ВЦ10Т позволяет обслуживать вложенные прерывания. В этом случае в подпрограмме обработки прерывания (ISR) может содержаться другая ISR, поэтому необходимо помнить следующее:

– если хотите, чтобы ISR прерывалось маскируемым прерыванием, то ISR должна размаскировать прерывание установкой подходящего бита маски в IMR и разрешить прерывания сбросом бита INTM (инструкция CLRC INTM);

– соответствующий регистр вектора прерывания должен быть прочитан GISR, чтобы получить смещение адреса вектора прерывания перед глобальным разрешением прерывания. Иначе, последовательность прерывания может вызвать перепись значения старого вектора прерывания;

– аппаратный стек ограничен восемью уровнями. Каждый раз, как только обслуживается прерывание или вызывается подпрограмма, то адрес возврата автоматически заталкивается (записывается) в стек. Этот механизм обеспечивает возврат контекста при выходе из прерывания. Стек содержит восемь ячеек, позволяющих прерываниям или вызовам подпрограммы вкладываться на глубину до восьми уровней (один уровень резервируется для отладки, для использования операций останова по адресу или пошаговому выполнению программы. Если отладка не используется, то это уровень может быть использован для внутренних целей). Если программа требует больше, чем восемь уровней стека, то можно использовать инструкции POPD и PSHD для расширения стека в памяти данных;

– если не использовать вложенные ISR, то можно избежать переполнение стека. ИС 1867ВЦ10Т. Имеет особенность, которая позволяет предотвратить неинициализированное вложение. Если прерывание происходит во время выполнения команду CLRC INTM, то ЦП всегда завершает команду CLRC INTM так же как следующую инструкцию перед обработкой незаконченного прерывания. Это гарантирует, что инструкция RET, помещенная сразу за инструкцией CLRC INTM, выполнится перед обработкой следующего прерывания. Процессор удаляет предыдущий адрес возврата из стека перед добавлением нового адреса возврата;

– если необходимо выполнить ISR внутри текущей ISR, а не после текущей ISR, то поместите инструкцию CLRC INTM более чем за одну инструкцию перед инструкцией возврата RET.

Время ожидания прерывания

Величина времени ожидания (задержка времени между сделанным запросом прерывания и когда оно обслужится) зависит от многих факторов. Этот раздел описывает факторы, которые определяют минимальное время ожидания и факторы, которые добавляют время ожидания. Максимальное время ожидания есть функция состояний ожидания и защите конвейера.

Имеется несколько компонентов времен задержки прерывания в ИС 1867ВЦ10Т:

– время синхронизации периферийного интерфейса;
– время ответа ЦП;
– время перехода к ISR.

Время синхронизации периферийного интерфейса

Время синхронизации периферийного интерфейса - это время, которое требуется запросу прерывания от периферийного устройства, чтобы распознаться периферийным интерфейсом, арбитрироваться и конвертироваться в запрос к ЦП. Это занимает до одного цикла SYSCLK для внутреннего прерывания от внутрикристальной периферии (периферийная шина работает с частотой SYSCL), другими словами в режиме деления на два или два цикла CPUCLK, или в режиме деления на четыре или четыре цикла CPUCLK. Внешние прерывания (NMI, INTx) имеют два цикла задержки синхронизации SYSCLK.

Запросы прерывания от периферийного менеджера событий не арбитрируются периферийным интерфейсом и есть только задержка на один цикл CPUCLK для распознавания прерывания.

Время ответа ЦП

Ответ ЦП – это время, которое требуется, чтобы ЦП распознал разрешенное прерывание, подтвердил прерывание, очистил конвейер и извлек первую инструкцию из таблицы векторов прерывания ЦП. Минимально время задержки ЦП – это четыре цикла CPUCLK. Если высокоприоритетное маскируемое прерывание запрашивается за время этого минимального периода времени ожидания, то оно маскируется, пока ISR для обслуживаемого прерывания завершено.

Время ожидания длиннее, если запрос прерывания происходит во время многоцикловой операции или других операций, которые не могут прерываться. Если высокоуровневое прерывание происходит во время дополнительного времени ожидания, то оно обслуживается перед низкоприоритеным прерыванием, в предположении, что оба прерывания разрешены.

Некоторые детали о влиянии многоцикловых инструкций:

– Состояния ожидания доступа к памяти. Инструкции, которые читают и пишут во внешнюю память, могут задерживаться состояниями ожидания, вызванные внешним сигналом READY или внутрикристальным генератором ожиданий. Эти состояния ожиданий могут влиять на выполнение инструкций во время запроса прерывания, и они могут затрагивать прерывание, если вектор прерывания должен выбираться из внешней памяти.

– Цикл повторения. Когда повторяется с RPT, инструкции выполняются параллельно в конвейере и контекст этих дополнительных параллельных операций не могут сохраняться в подпрограмме обработки прерывания. Чтобы защитить контекст инструкции повторения, ЦП блокирует все прерывания, за исключением сброса, до тех пор, пока цикл не выполнится полностью.

Примечание – Сброс не задерживается многоцикловыми инструкциями, а NMI может задерживаться. Если одно прерывание обслуживается, то имеется несколько факторов, которые задерживают обслуживание нового прерывания.

– Адрес возврата записывается в аппаратный стек каждый раз, когда следует другая программа обслуживания и подпрограмма ИС 1867ВЦ10Т имеет особенность, которая позволяет предотвратить неинициализированное вложение. Если прерывание происходит во время выполнения команды CLRC INTM, то ЦП всегда завершает команду CLRC INTM так же, как следующую инструкцию перед обработкой незаконченного прерывания. Это гарантирует, что инструкция RET, поме-

щенная сразу за инструкцией CLRC INTM, выполнится перед обработкой следующего прерывания. Инструкция возврата выталкивает предыдущий адрес возврата из верхушки стека перед записью нового адреса возврата в стек. Если прерывание произошло перед возвратом, то новый адрес возврата должен добавляться к аппаратному стеку, даже если стек уже полный.

– Прерывания также блокируются после инструкции RET, пока последняя команда в адресе возврата выполнится.

Время перехода к ISR

Время перехода ISR – это время, которое необходимо, чтобы получить специальную часть ISR. Эта величина зависит от реализации ISR. Для простейшей ситуации, в которой только один переход необходим ISR, то минимальное время перехода к ISR составляет четыре цикла ЦП (см. 14.6.3 «Прерывания»).

Итог операции прерывания

Если прерывание передалось ЦП, то он действует следующим образом (см. рисунок 256):

– Если запрашивается маскируемое прерывание:

1 Бит флага прерывания устанавливается в индивидуальном управляющем регистре. Если индивидуальный маскирующий бит установлен, то соответствующий бит IFR устанавливается.

2 Если бит IFR установлен, то проверяются условия подтверждения прерывания (INTM bit = 0 или IMR bit = 1). Если условия выполняются, то ЦП обслуживает прерывание, генерирует сигнал подтверждения, иначе он игнорирует прерывание и продолжает выполнять текущую кодовую последовательность.

3 Когда прерывание подтверждается, то бит IFR сбрасывается в 0 и бит INTM устанавливается в 1 (блокируя маскируемые прерывания). Бит флага соответствующего управляющего регистра сбрасывается.

4 Адрес возврата (увеличенное значение PC) сохраняется в стеке.

5 ЦП переходит и выполняет подпрограмму обслуживания прерывания ISR, которая включает инструкцию возврата, считывающую адрес возврата из стека. ЦП продолжает выполнять прерванную кодовую последовательность.

– Если запрашивается немаскируемое прерывание:

1 ЦП немедленно подтверждает прерывание, генерируя сигнал подтверждения.

Примечание – Когда прерывание запрашивается инструкцией INTR и соответствующий бит IFR устанавливается, то ЦП не сбрасывает его автоматически. Если приложение требует, чтобы этот флаг сбрасывался, то оно должно делать это самостоятельно.

2 Если прерывание было запрошено с вывода RESET# или вывода NMI#, или инструкцией NMI или инструкцией INTR, то бит INTM, устанавливается в 1, блокируя маскируемые аппаратные прерывания. Если прерывание запрашивается инструкцией TRAP, то бит INTM не устанавливается в 1.

3 Адрес возврата (увеличенное значение PC) сохраняется в стеке.

4 ЦП переходит к ISR и выполняет. ISR содержит инструкцию возврата, которая читает из стека адрес возврата. ЦП продолжает выполнять прерванную кодовую последовательность.

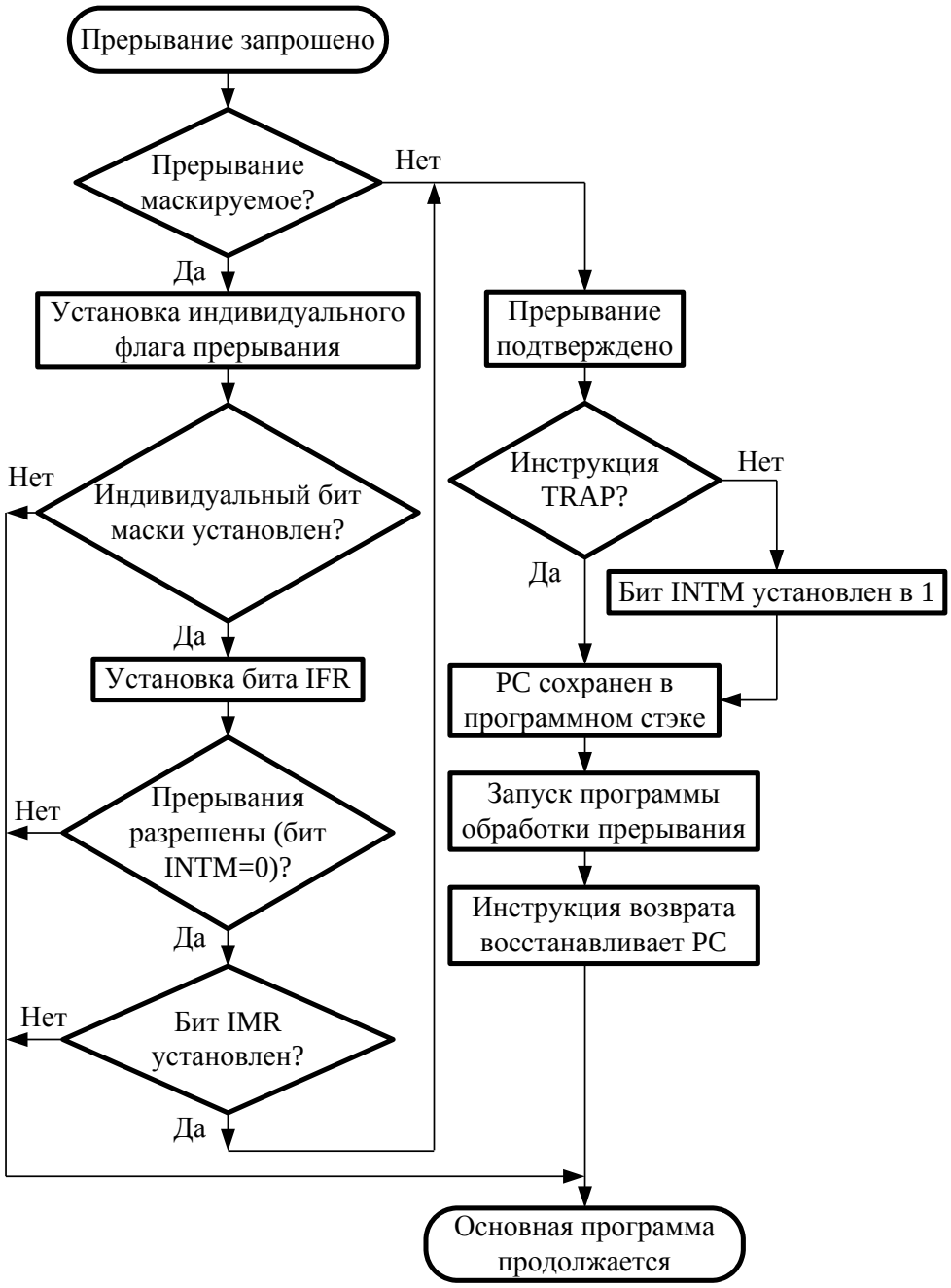


Рисунок 256 – Диаграмма операции прерывания

Управляющие регистры внешних прерываний

ИС 1867ВЦ10Т имеет шесть выводов для внешних прерываний с программируемой полярностью и в большинстве случаев приоритетом. Эти выводы программируют управляющие регистры типа A, B и C. До 14 дополнительных выводов прерывания может поддерживаться, используя два регистра. В некоторых

конфигурациях устройства ИС 1867ВЦ10Т регистры управления питанием могут использоваться для формирования прерывания в ответ на события, происходящие в периферии.

Реализации внешних прерываний

Адрес	Регистр	15	14	13	12	11	10	9	8
7070h	XINTA1 (Type A)	XINTA1 Flag	Резерв						
		7	6	5	4	3	2	1	0
		Резерв	XINTA1 Pin data	XINTA1 NMI†	Резерв		XINTA1 Polarity	XINTA1 Priority	XINTA1 Enable

Адрес	Регистр	15	14	13	12	11	10	9	8
7072h	XINTA–NMI (Type A)	XINTA–NMI Flag	Резерв						
		7	6	5	4	3	2	1	0
		Резерв	XINTA–NMI Pin data	XINTA–NMI NMI†	Резерв		XINTA–NMI Polarity	Резерв	

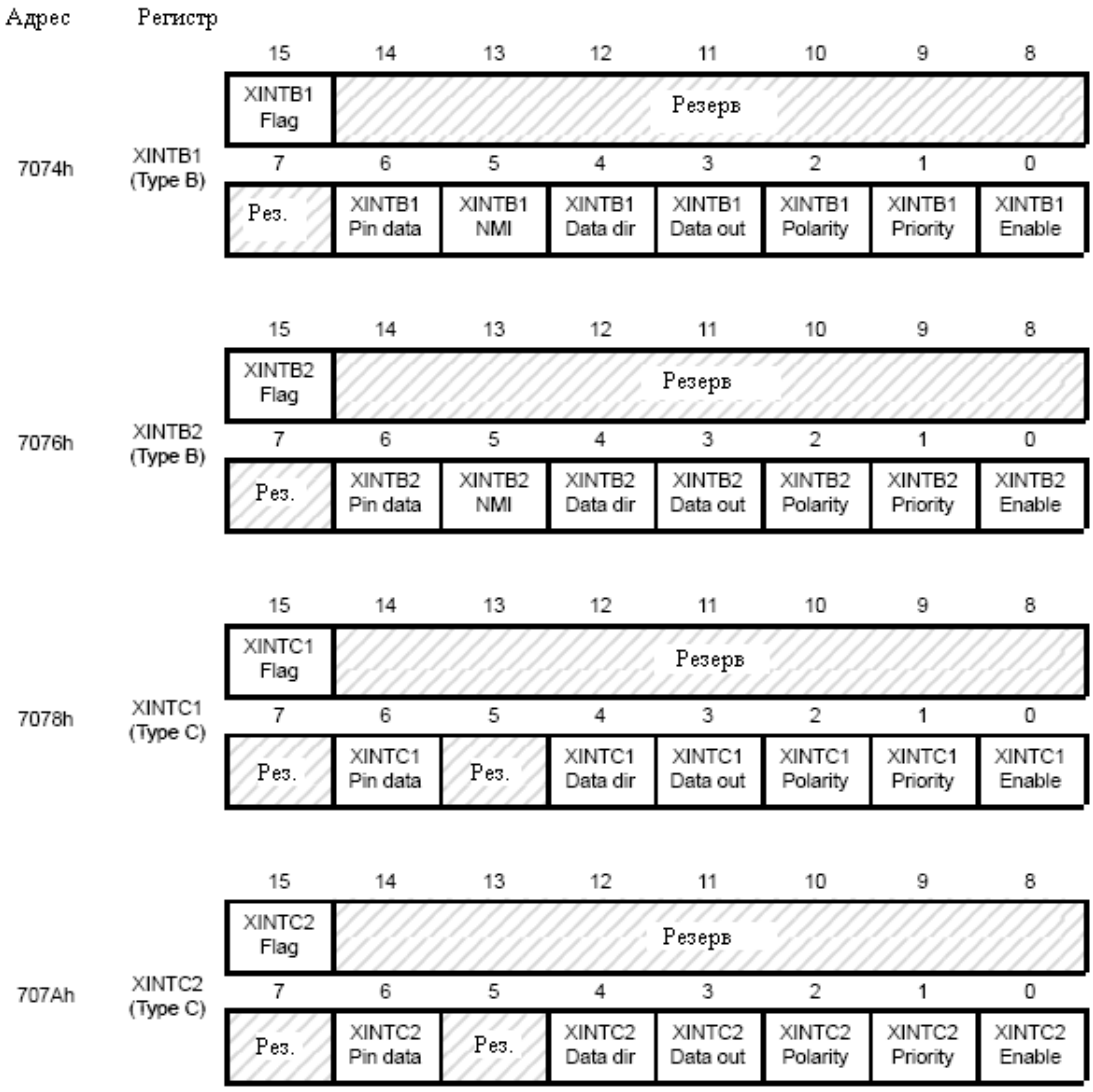


Рисунок 257 – Регистры управления внешним прерыванием

Имеется три типа внешних прерываний: тип А, тип В и тип С. ИС 1867ВЦ10Т обычно имеет два прерывания каждого типа, хотя они и не могут все использоваться. Действительное количество и тип выводов внешних прерываний зависит от конфигурации устройства (ИС 1867ВЦ10Т). На рисунке 257 показаны доступные регистры управления внешними прерываниями, включая и регистры прерывания модуля питания

Выводы прерываний типа А, типа В и типа С

В таблице 124 объединены типы выводов внешних прерываний:

– выводы типа А допускают только цифровой вход.

– один вывод типа А – это маскируемый вывод прерывания, а другой - немаскируемый вывод прерывания.

– Выводы типа В могут программироваться как маскируемые, так и не маскируемые прерывания,

– Выводы типа С могут быть только маскируемыми.

Все три типа могут активизироваться либо на подъем, либо на спад входного сигнала. Полярность программируется.

Все три типа прерывания могут конфигурироваться для высокого и низкого приоритетов запросов прерывания.

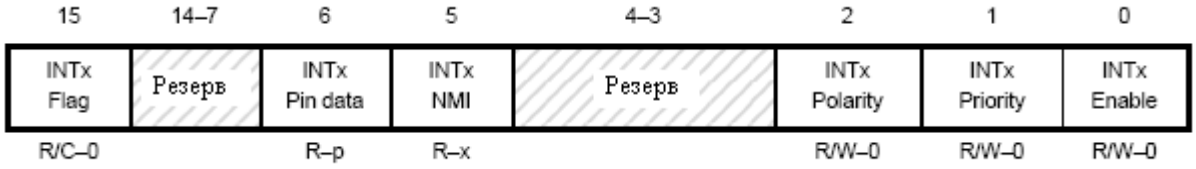
Все выводы прерывания конфигурируются как цифровые входы на сбросе.

Таблица 124 – Типы выводов внешнего прерывания

Тип вывода	NMI возможность	Доступное число	Цифровой ввод/вывод
Тип A	Да – аппаратный	1	Только вход
Тип B	Да – программируемый	2	Вход/выход
Тип C	Нет	2	Вход/выход

Выводы прерывания типа A

Выводы прерываний типа A могут использоваться как немаскируемые прерывания, обычно маскируемые прерывания или входные цифровые выводы. Управляющий регистр типа A имеет общую форму, показанную на рисунке 258. Описание бит этого регистра приведено в таблице 125.



Примечание – R – возможно чтение; W – возможна запись;
C – очистка только записью; n – значение после сброса (x – не изменяется сбросом); p – логический уровень вывода.

Рисунок 258 – Регистр управления прерыванием типа A

Таблица 125 – Описание бит регистра

Бит	Мнемоника	Описание
15	INTx Flag	Флаг прерывания x. Этот читаемый/очищаемый бит указывает, обнаружен ли на выводе прерывание x. Этот бит устанавливается или нет, если прерывание разрешается. Можно использовать этот бит для программного опроса, чтобы увидеть произошел или нет выбранный фронт. Этот бит сбрасывается программой или системным сбросом. Этот бит требует сброса, когда этот вывод используется для прерываний. Прерывание происходит для каждого фронта сигнала выбранного для этого вывода, даже если этот бит уже установлен. Очистка этого бита влечет сброс незавершенного прерывания от этого вывода. 0 – фронт сигнала не обнаружен. 1 – фронт сигнала обнаружен.

Бит	Мнемоника	Описание
06	INTxPD	INTx Pin Data. Бит данных вывода прерывания. Этот только читаемый бит отражает текущее состояние уровня на выводе прерывания, независимо от того, как вывод прерывания конфигурируется. 0 – вывод в логическом 0. 1 – вывод в логической 1.
05	INTxNMI	Бит разрешения немаскируемого прерывания. Этот только читаемый бит определяет, может или нет этот вывод генерировать немаскируемое прерывание. На большинстве ИС 1867ВЦ10Т регистр прерываний типа А имеет этот бит аппаратно установленным в 0 уровне. 0 – вывод для обычного прерывания или цифровой вход. 1 – вывод для немаскируемого прерывания.
04-03	Reserved	Резервный. Читается как неопределенный, запись не влияет
02	INTx Polarity	Бит полярности прерывания. Этот читаемый/записываемый бит определяет, какой полярности сигнал вызывает прерывание (переход входного сигнала с низкого на высокий или наоборот). 0 – прерывание генерируется на спаде сигнала (переход с высокого на низкий уровень) 1 - прерывание генерируется на подъеме сигнала (переход с низкого на высокий уровень)
01	INTx Priority	Бит приоритета прерывания. Этот читаемый/записываемый бит определяет, какой приоритет прерывания установлен. Этот бит не влияет, если бит NMI установлен. 0 – высокий приоритет. 1 – низкий приоритет.
00	INTx Enable	Бит разрешения прерывания. Этот читаемый/записываемый бит разрешает или запрещает маскируемое прерывание. Этот бит не влияет, если бит NMI установлен. 0 – запрещает прерывание (вывод используется как вход) 1 – разрешено прерывание

Модуль прерывания по питанию

Модуль прерываний по питанию может использоваться для связи сигналов от внутренней периферии, таких как модуль линеаризации сигнала (активные высокий или низкий), которые необходимы для запроса прерывания. Эти прерывания могут также использоваться с сигналами, идущими от внешних выводов. Каждый сигнал прерывания имеет один бит разрешения и один флаг прерывания. Каждый набор из семи внутренних прерываний имеет один вектор прерывания. Приоритетный уровень прерывания определяется производителем и не программируется. Каждый флаг прерывания модуля мощности имеет или активный низкий или высокий активный вход. Эти биты существуют в регистре прерывания модуля мощности, показаны на рисунках 259 и 260.

15	14	13	12	11	10	9	8
PM INT Flag	PM INT Status 6	PM INT Status 5	PM INT Status 4	PM INT Status 3	PM INT Status 2	PM INT Status 1	PM INT Status 0
R/C–0	R/C–0	R/C–0	R/C–0	R/C–0	R/C–0	R/C–0	R/C–0

Примечание – R – возможно чтение; W – возможна запись, C – очистка только записью; n – значение после сброса.

Рисунок 259 – Биты флага PM INT

7	6	5	4	3	2	1	0
PM INT ENA	PM INT Enable 6	PM INT Enable 5	PM INT Enable 4	PM INT Enable 3	PM INT Enable 2	PM INT Enable 1	PM INT Enable 0
R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0

Примечание – R – возможно чтение; W – возможна запись, n – значение после сброса.

Рисунок 260 – Биты разрешения PM INT

14.6.4 Операция сброса

Вывод RESET# генерирует немаскируемое внешнее прерывание, которое может использоваться в любое время, чтобы установить ИС 1867ВЦ10Т в известное состояние. Сброс имеет самый высокий приоритет, и никакие другие прерывания не могут его перекрыть. Обычно сброс используется после включения питания, когда ИС 1867ВЦ10Т находится в неизвестном состоянии. Поскольку сигнал сброса прекращает операции с памятью и инициализирует статусные биты, система должна реинициализироваться после каждого сброса. NMI прерывание может использоваться для «мягкого» сброса, поскольку оно не прекращает ни операции с памятью, ни инициализирует статусные биты. В зависимости от конфигурации ИС 1867ВЦ10Т, имеется до пяти случаев сброса ИС 1867ВЦ10Т, как показано на рисунке 261. В четырех из этих случаев сброс генерируются внутри, а в одном случае генерируется выводом RESET# и управляется внешним образом.

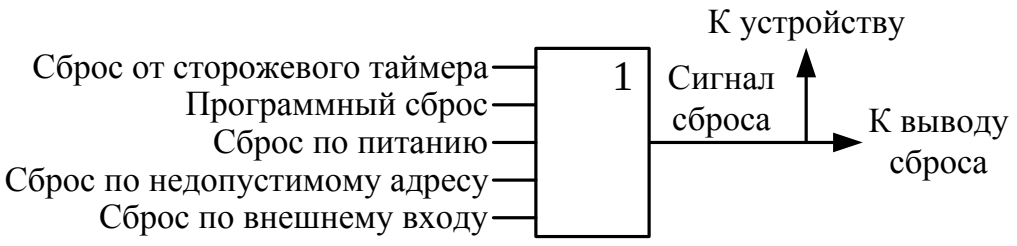


Рисунок 261 – Сигналы сброса

Пять возможных сигналов сброса генерируются следующим образом:

– Сигнал сброса от сторожевого таймера. Этот сброс формируется, если модуль сторожевого таймера переполнился или неправильное значение записано в регистр ключа модуля сторожевого таймера или в регистр управления модулем сторожевого таймера. (Когда на ИС 1867ВЦ10Т подается питание, то сторожевой таймер автоматически активизируется).

– Сброс, сгенерированный программно. Этот сброс реализуются системным управляющим регистром (SCR). Сбрасывая бит 14 (RESET0) или устанавливая бит 15 (RESET1), этот регистр вызывает системный сброс ИС 1867ВЦ10Т.

– Сброс при включении питания/V_{CC} вышел из допустимого диапазона. Этот сброс генерируется двумя источниками – внешним выводом сброса по питанию (PORESET) и схемой обнаружения понижения питания ниже нормы. Эта схема посылает сигнал, если ИС 1867ВЦ10Т работает при напряжении V_{CC}, вышедшей за рекомендуемые границы работы.

– Неверный адрес. ИС 1867ВЦ10Т содержит нереализованное адресное пространство, которое помечено как ‘Illegal’ (неверное). Любой доступ в это адресное пространство вызовет сброс по неправильному адресу.

– Вывод сброса активизирован. Для генерации внешнего импульса сброса на выводе RESET# длительность низкого уровня этого импульса достаточно мала и равняется несколько наносекунд, однако один цикл SYSCLK необходим, чтобы гарантировать распознавания ИС 1867ВЦ10Т сигнала сброса. Типовая схема сброса содержит 10 КОм повышающий резистор по питанию V_{CC} к выводу RESET#.
Как только источник сигнала сброс активизировался, внешний вывод RESET# должен быть активным низким как минимум восемь циклов SYSCLK. Это позволяет ИС 1867ВЦ10Т сбросить внешние устройства (периферию), подключенные к выводу RESET#. (Вывод RESET# имеет вход/выход с открытым стоком и должен иметь подключенный резистор подтяжки к V_{CC}). Дополнительно, если V_{CC} выходит за границы или вывод RESET# сохраняет низкий уровень, логика сброса удерживает ИС 1867ВЦ10Т в состоянии сброса до тех пор, пока эти условия сохраняются на этом выводе.

Когда сигнал сброса получен, то программа определяет источник сброса чтением содержимого системного статусного регистра (SYSSR). Регистр SYSSR содержит один статусный бит, для каждого из пяти источников, которые вызвали сброс.

Условия сброса заставляют ИС 1867ВЦ10Т завершить выполнение текущей программы и изменить различные регистры и статусные биты. Во время сброса содержимое ОЗУ остается неизменным, а все управляющие биты, которые затрагиваются сбросом, инициализируются. Процессор начинает выполнение программы с ячейки 0, которая обычно содержит инструкцию перехода к системной подпрограмме инициализации. Когда происходит сброс ИС 1867ВЦ10Т, выполняются следующие действия:

– Логический 0 загружается в бит CNF (configuration control) в статусном регистре ST1. Это картирует блок B0 DARAM в пространство данных.

– Программный счетчик сбрасывается в 0.

– Бит режима прерывания INTM (interrupt mode) устанавливается в 1, блокируя все маскируемые прерывания (прерывания RESET и NMI немаскируемые). Также сбрасывается регистр флагов прерывания IFR (interrupt flag register) и регистр масок прерывания IMR (interrupt mask register).

– Счетчик повторений RPTC (repeat counter) сбрасывается.

– Состояние ожидания (если используется интерфейс внешней памяти устанавливается для максимальной длительности).

– Биты периферийного регистра инициализируются как описано в разделе 14. Никакие другие регистры или биты ЦП (такие как аккумулятор, DP, ARP и ARx) не инициализируются.

14.6.5 Режимы пониженного потребления

ИС 1867ВЦ10Т имеет четыре режима пониженного энергопотребления, которые уменьшают рабочую мощность потребления, останавливая синхронизацию (и таким образом снижая потребление). Пока ИС 1867ВЦ10Т находится в режиме пониженного энергопотребления, все содержимое регистров сохраняется, а после завершения этого режима работа ИС 1867ВЦ10Т продолжается без изменения. Завершение режима пониженного энергопотребления осуществляется прерыванием. Содержимое всей внутрикристальной памяти остается неизменным. Однако, если режим пониженного энергопотребления заканчивается сбросом, то содержимое некоторых регистров меняется (регистры, содержимое которых меняется сбросом).

Имеется три различных области синхронизации, которые могут отключаться во время уменьшения энергопотребления:

– Область синхронизация ЦП. Все синхросигналы ЦП и памяти, за исключением регистра управления прерыванием.

– Область системной синхронизации. Вся синхронизация периферии (CPUCLK or SYSCLK), это синхросигналы для регистров прерывания ЦП и синхронизация аналогового модуля (ACLK).

– Синхронизация модуля сторожевого таймера (WDCLK). Обычно это 16 кГц синхросигнал используется для увеличения на 1 таймера в модуле сторожевого таймера (Watchdog Timer) и блока прерывания реального времени (Real Time Interrupt module).

Примечание – Термины CPUCLK и область синхронизации ЦП, SYSCLK и область синхронизации системы не являются одним и тем же.

Все типы возможных режимов пониженного энергопотребления имеют понижающие уровни энергопотребления и увеличивающиеся задержки выхода из режима низкого потребления при переходе от типа к типу пониженного энергопотребления. Все из возможных режимов потребления могут быть не реализованы для пониженного энергопотребления. Режим понижения потребления выполняется, когда ЦП выполняет инструкцию IDLE. Выбор одного из пяти возможных режимов определяется битами PLLPDM (1:0) регистра CKCR0 в модуле синхронизации.

Режимы пониженного энергопотребления в порядке уменьшения мощности и увеличения времени выхода из режима:

– Режим IDLE1. Останавливает синхросигнал в ЦП (область синхронизации ЦП), но синхросигналы для всей периферии (область системной синхронизации) продолжают работать. Выход из режима IDLE1 происходит немедленно следующим прерыванием или сбросом.

– Режим IDLE2. Останавливает синхроимпульсы в обоих областях – область синхронизации ЦП и область системной синхронизации. Выход из этого ре-

жима происходит немедленно следующим прерыванием, который запускает прерывание или сброс. Сторожевой таймер продолжает работать и, в конечном счете, время выходит, вызывая сброс.

– Режим PLL. Выключает PLL (если разрешается). Сторожевой таймер продолжает работать. Выход из этого режима может происходить прерыванием, которое «будит» или сбросом. ИС 1867ВЦ5Т не начинает полноскоростные операции, пока питание PLL не включится и он не подстроит частоту.

– Режим осциллятора. Этот режим выключает питание осциллятора (если разрешается). Этот режим имеет самый низкий режим потребления. Синхроимпульсы в устройстве отсутствуют. Прерывания, которые «будят» или сигнал сброса заставляют ИС 1867ВЦ10Т выйти из этого режима. Синхроимпульсы отсутствуют до тех пор, пока питание на осцилляторе не появится (это занимает время порядка миллисекунд; см. раздел 14). ИС 1867ВЦ5Т не стартует на полной скорости до тех пор, пока PLL не подстроится (это дополнительно займет несколько сот микросекунд).

В таблице 126 показывается состояние синхроимпульсов ЦП и периферии во время каждого из четырех режимов пониженного потребления.

Таблица 126 – Режимы пониженного энергопотребления

Режим	Синхронизация ЦП	Синхронизация системы	Синхронизация сторожевого таймера	Генератор
IDLE 1	Выключено	Включено	Включено	Включено
IDLE 2	Выключено	Выключено	Включено	Включено
Питание PLL отключено	Выключено	Выключено	Включено	Включено
Питание осциллятора отключено	Выключено	Выключено	Выключено	Выключен

Установка и получение режимов пониженного потребления

Все режимы пониженного энергопотребления инициализируются выполнением инструкции IDLE. Режим зависит от значения поля PLLPDM регистра управления блока синхронизации (CKCR0). В таблице 127 поясняется, как два бита PLLPDM регистра CKCR0 определяют режим пониженного энергопотребления.

Таблица 127 – Установка режимов пониженного потребления битами PLLPDM

Биты PLLPDM	Режим пониженного энергопотребления
00	IDLE 1
01	IDLE 2
10	Питание PLL отключено
11	Питание осциллятора отключено

Выход из режима пониженного потребления

В любом из четырех режимов пониженного энергопотребления (IDLE1, IDLE2, выключено питание PLL или выключено питание осциллятора) синхрони-

зация области ЦП выключается. Следовательно, программные прерывания не могут генерироваться для вывода процессора из этого состояния. Прерывания могут генерироваться только на внешних выводах или из внутрикристальной периферии. Ниже приведены сигналы, выводящие ИС 1867ВЦ10Т из пониженного энергопотребления.

Сброс (Reset). Сигнал сброса завершает режим пониженного энергопотребления следующим образом:

– вывод сброс (RESET#) вызывает выход ИС 1867ВЦ10Т из любого режима пониженного энергопотребления;

– сброс по тайм-ауту таймера модуля сторожевого таймера вызывает выход ИС 1867ВЦ10Т из режимов IDLE1, IDLE2 и выключению питания PLL. Таймер не увеличивается во время выключенного осциллятора, так как синхронизация сторожевого таймера останавливается.

Внешние прерывания. Внешние прерывания XINTn и NMI, а также прерывания, которые запускают прерывания. Это означает, что когда синхросигналы остановлены, то комбинационная логика передает сигналы с этих выводов для рестарта синхроимпульсов.

Внешние прерывания (XINTn) – это маскируемые прерывания, которые могут перевести ЦП из режима пониженного энергопотребления только, если они размаскированы. Если они маскируются, то они «будят» ИС 1867ВЦ10Т стартом всех синхросигналов, но ИС 1867ВЦ10Т остается в состоянии IDLE.

Соответствующая периферия также может генерировать «будящее» прерывание, когда синхроимпульсы в области системной синхронизации выключаются. Для примера, некоторые коммуникационные порты могут иметь возможность генерировать «будящее» прерывание в ответ на получение символа.

Режим выключенного осциллятора может завершиться только внешними прерываниями (NMI или XINTn). В этом режиме осциллятор выключен (никакие синхросигналы в ИС 1867ВЦ10Т не активны), таким образом, нет прерываний, которые могут сгенерироваться периферией и таймер модуля сторожевого таймера не может сгенерировать сигнал тайм-аута.

Периферийные прерывания. Режимы пониженного энергопотребления IDLE 1, IDLE2 и выключение PLL могут завершаться прерываниями от различной периферии при правильных условиях. В режиме IDLE1 все синхросигналы для периферийных устройств все еще работают, следовательно, любое размаскируемое периферийное прерывание завершает режим IDLE1. В режиме IDLE2 и выключеного PLL синхросигналы в области системных синхроимпульсов включены. Как результат, только размаскированные периферийные прерывания, не синхронизированные системным синхроимпульсом, могут перевести ЦП из режимов IDLE2 и выключеного PLL в рабочее состояние.

В режиме выключенного осциллятора периферийные синхроимпульсы и синхроимпульсы таймера модуля сторожевого таймера выключены. Поэтому, только размаскированные периферийные прерывания, несинхронизированные одним из этих синхроимпульсов, могут закончить режим выключенного осциллятора. В таблице 128 подведен итог состояний процессора в режимах пониженного энергопотребления и приведен список прерываний, которые завершают и не завершают это состояние.

Таблица 128 – Режимы пониженного энергопотребления и их окончание

Режим	Синхронизация ЦП	Системная синхронизация	Синхронизация сторожевого таймера	Генератор	Завершается	Не завершается
IDLE 1	Выкл.	Вкл.	Вкл.	Вкл.	Сбросом RESET. Сбросом от сторожевого NMI (на выводе). XINTx (на немаскируемом выводе). Любое периферийное прерывание (не маскируемое)	Маскируемые прерывания
IDLE 2	Выкл.	Выкл.	Вкл.	Вкл.	Сбросом RESET). Сбросом от сторожевого NMI (на выводе). XINTx (на немаскируемом выводе). Периферийные прерывания, которые «будят».	Маскируемые прерывания. Периферийные прерывания, зависящие от системной синхронизации
Питание PLL оключено (PPD)	Выкл.	Выкл.	Вкл.	Вкл.	Сбросом (RESET). Сбросом от сторожевого NMI# (на выводе). XINTx (на немаскируемом выводе). Периферийные прерывания, которые «будят»	Маскируемые прерывания. Периферийные прерывания, зависящие от системной синхронизации
Питание осцил-лятора отклю-чено (OPD)	Выкл.	Выкл.	Выкл.	Выкл.	Сбросом (RESET). Сбросом от сторожевого NMI (на выводе). XINT1, XINT2, XINT3 (на немаскируемом выводе). Периферийные прерывания, которые «будят»	Маскируемые прерывания. Периферийные прерывания

После выхода из режима пониженного энергопотребления

Когда решается вопрос о том, как запускать процессор, то нужно рассмотреть следующие возможные случаи:

– если используется сброс или NMI, то ЦП немедленно выполняет соответствующую подпрограмму прерывания;

– если используется маскируемое аппаратное прерывание, то следующие действия зависят от состояния бита (INTM) статусного регистра ST0:

– INTM = 0: Прерывание разрешается и ЦП выполняет соответствующую подпрограмму.

– INTM = 1. Прерывание запрещено и ЦП продолжает с инструкции после IDLE.

Если нет необходимости, чтобы ЦП выполнил подпрограмму перед продолжением прерванной программной последовательности, то нужно:

– не использовать сброс или NMI для перевода ЦП из состояния режима пониженного энергопотребления;

– убедиться, что программа установила бит INTM в 1 (SETC INTM) перед инструкцией IDLE.

Если необходимо, чтобы ЦП выполнил подпрограмму перед продолжением прерванной программы, то нужно:

– убедиться, что бит INTM стоит в 0 (CLRC INTM) перед инструкцией IDLE;

– убедиться, что разрешены прерывания соответствующих источников (локально в периферийных регистрах маскирования/размаскирования и глобально в регистре маски ЦП IMR).

Итог операций режима пониженного питания

Когда инструкция IDLE выполняется, то:

1 Программный счетчик инкрементируется один раз, так что когда режим пониженного энергопотребления завершается, то следующая инструкция – есть инструкция, которая следует за инструкцией IDLE (исключая случай сброса).

2 ИС 1867ВЦ10Т установит режим пониженного энергопотребления, выбранный битами PLLPM и сохранится в этом состоянии, пока он не получит соответствующего аппаратного прерывания.

3 По получении соответствующего прерывания ИС 1867ВЦ10Т выходит из режима пониженного энергопотребления.

4 Если используется NMI, чтобы «разбудить» ЦП, то ЦП выполняет соответствующую подпрограмму перед продолжением прерванной программной последовательности. Если используется маскируемое прерывание, то следующие действия ЦП зависят от значения бита INTM:

– Если INTM=0, то маскируемые прерывания разрешаются, ЦП первым выполняет подпрограмму этого прерывания, которое выводит его из режима пониженного энергопотребления. После этого ЦП продолжает работу с инструкцией, следующей за инструкцией IDLE.

– Если INTM=1, то маскируемые прерывания запрещены и ЦП продолжает выполнение инструкции, следующей за инструкцией IDLE.

В таблице 128 суммируются четыре режима пониженного энергопотребления, включая ориентировочные значения уровня потребления для каждого из режимов. В дополнение, в таблице 129, показан статус ИС 1867ВЦ10Т, когда она не находится в режиме пониженного энергопотребления.

Таблица 129 – Режимы РПЭ и рабочий режим

Режим	PLLPM +IDLE	Ток потребления	Синхронизация ЦП	Системная синхронизация	Синхронизация сторожевого таймера	PLL	Генератор

Режим	PLLPM +IDLE	Ток потребления	Синхронизация ЦП	Системная синхронизация	Синхронизация сторожевого таймера	PLL	Генератор
Рабочий	XX+нет IDLE	240мА	Вкл.	Вкл.	Вкл.	Вкл.	Вкл.
IDLE1	00+ IDLE	120мА	Выкл.	Вкл.	Вкл.	Вкл.	Вкл.
IDLE2	01+ IDLE	20мА	Выкл.	Выкл.	Вкл.	Вкл.	Вкл.
PPD	10+ IDLE	10мА	Выкл.	Выкл.	Вкл.	Выкл.	Вкл.
OPD	11+ IDLE	3 мкА	Выкл.	Выкл.	Выкл.	Выкл.	Выкл.

14.7 Режимы адресации

В этом разделе объясняется три основных режима адресации памяти, используемые инструкциями. ИС 1867ВЦ10Т имеет следующие три режима адресации:

- режим непосредственной адресации;
- режим прямой адресации;
- режим косвенной адресации.

В режиме непосредственной адресации константа, которой манипулирует инструкция, поставляется прямо как операнд этой инструкции. Доступны два типа непосредственной адресации – короткая непосредственная адресация (восемь, девять и тринадцать бит операнда включаются в слово инструкции) и длинная непосредственная адресация (использует 16-битный операнд).

Когда необходим доступ к памяти данных, то можно использовать или прямую или косвенную адресацию. Прямая адресация связывает семь младших бит слова инструкции с девятью битами указателя на страницу памяти данных (DP) для формирования 16 бит адреса. Косвенная адресация обращается к памяти данных через один из восьми 16-битных вспомогательных регистров.

14.7.1 Непосредственная адресация

В режиме непосредственной адресации константа, которой манипулирует инструкция, поставляется прямо как операнд этой инструкции.

Доступны два типа непосредственной адресации:

– Короткая непосредственная адресация. Инструкция, использующая этот вид адресации, берет восьми, девяти или тринадцати битные константы как операнды. Инструкция с короткой непосредственной адресацией требует однословной инструкции с константой, включенной в это слово.

– Длинная непосредственная адресация. Инструкция, использующая этот вид адресации, берет 16-битную константу как операнд и требует двухсловной инструкции. 16-битное значение может использоваться как абсолютная константа или как значение с дополнением до двух.

В примере 1 непосредственный операнд является частью слова инструкции RPT. Для этой инструкции регистр инструкции будет загружаться значением, показанным на рисунке 262. Непосредственным операндам предшествует символ #.

Пример 1 – Инструкция RPT, использующая короткую непосредственную адресацию.

```
RPT #99 ;Выполняет инструкцию, которая следует за RPT
        ;100 раз
```

Рисунок 262 – Содержимое регистра инструкций для примера 1

В примере 2 непосредственный операнд содержится в следующем слове. Регистр инструкции получает последовательно два 16-битных значения, как показано на рисунке 263.

Пример 2 – Инструкция ADD использует длинную непосредственную адресацию

```
ADD #16384,2  ;Сдвинуть значение 16384 влево на два
              ;бита и добавить результат к аккумуля-
           тору
```

Рисунок 263 – Два слова загружаются последовательно в регистр инструкции в примере 2

14.7.2 Режим прямой адресации

В режиме прямой адресации память данных адресуется блоками по 128 слов, которые называются страницами данных или просто страницами. 64К слов памяти данных содержат 512 страниц, помеченные от 0 до 511, как показано на

рисунке 264. Текущая страница определяется значением 9 бит указателя страниц данных (DP) в статусном регистре ST0. Для примера, если значение DP равно 0 0000 0000_2, то текущая станица равна 0. Если значение DP равно 0 0000 0000_2, то текущая страница равна 2.

Значение DP	Смещение	Память данных
0000 0000 0 … 0000 0000 0	000 0000 … 111 1111	Страница 0: 0000h–007Fh
0000 0000 1 … 0000 0000 1	000 0000 … 111 1111	Страница 1: 0080h–00FFh
0000 0001 0 … 0000 0001 0	000 0000 … 111 1111	Страница 2: 0100h–017Fh
… … …	… … …	… … …
1111 1111 1 … 1111 1111 1	000 0000 … 111 1111	Страница 511: FF80h–FFFFh

Рисунок 264 – Распределение страниц памяти данных

Для прямой адресации в дополнении к значению страницы данных процессор должен знать и адрес конкретного слова на этой странице. Этот адрес определяется 7-битным смещением. Смещение получается из семи последних бит регистра инструкции, который хранит опкод для выполнения следующей инструкции. В режиме прямой адресации содержимое регистра инструкции имеет формат, показанный на рисунке 265. Описание бит этого регистра приведено в таблице 130.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8 старших бит								0	7 младших бит						

Рисунок 265 – Содержимое регистра инструкции (IR) в режиме прямой адресации

Таблица 130 – Описание бит регистра

Биты	Мнемоника	Описание
15-08	8 MSBs	Эти биты индицируют тип инструкции (для примера инструкцию ADD) и также содержат информацию относительно сдвига значения данных, которые используются инструкцией.

Биты	Мнемоника	Описание
07	0	Индикатор прямой/косвенной адресации. 0 – прямая адресации. 1 – косвенная адресация.
06-00	7 LSBs	Эти биты показывают смещение для получения адреса памяти данных, на которую ссылается инструкция.

Для формирования полного 16-битного адреса, процессор подсоединяет значение DP и семь младших разрядов регистра инструкции, как показано на рисунке 266. DP обеспечивает девять старших бит адреса (номер страницы), а семь младших разрядов регистра инструкции обеспечивают семь младших разрядов адреса. Для примера, для доступа к адресу 003Fh необходимо установить DP равным 0 (DP = 0000 0000 0_2) и смещение равное 011 1111_2. Соединение (конкантенация) DP и смещения сформирует 16-битный адрес 0000 0000 0011 1111_2 или 003Fh или 63_{10}.

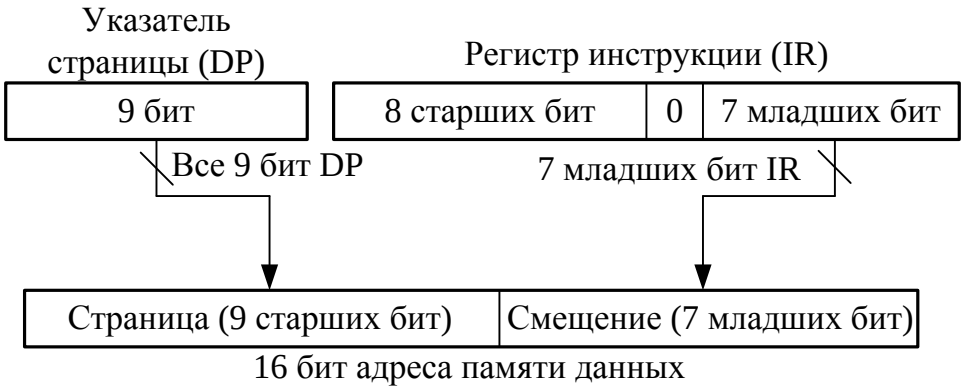


Рисунок 266 – Формирование адреса данных в режиме прямой адресации

ПРЕДУПРЕЖДЕНИЕ! Следует инициализировать DP во всех программах, так как DP не инициализируется сбросом и его значение не определено после включения питания. Инструментальные средства используют значение многих параметров по умолчанию, включая и DP. Тем не менее, программы, которые явно не используют DP, могут выполняться неправильно в зависимости от того, выполняются ли они под управлением инструментальных средств или нет.

Когда используется режим прямой адресации, то процессор использует DP, чтобы найти страницу данных и использует семь младших бит регистра инструкции, чтобы найти конкретный адрес на странице. Необходимо всегда выполнять следующее:

1 Установить страницу данных. Загрузить подходящее значение (от 0 до 511) в DP. DP может загружаться инструкцией LDP или инструкцией, которая может загружать значение в ST0. Инструкция LDP загружает значение прямо в DP, не влияя на остальные биты ST0. Для примера, для установки текущей страницы 32 (адрес 1000h–107Fh) можно использовать команду:

```
LDP #32 ;Инициализация указателя страницы
```

2 Указать смещение. Обеспечить семь бит смещения как операнд инструкции. Для примера,

```
ADD 1h ;Прибавить к аккумулятору значение по адресу со
       ;смещением 1 в текущей странице
```

Нет необходимости установки DP перед каждой инструкцией, которая использует прямую адресацию. Тем не менее, если меняется страница используемых данных, то необходимо менять значение в DP и устанавливать новую страницу.

Примеры прямой адресации

В примере 3 первая команда загружает DP значением 0000 0100_2,чтобы установить текущую страницу данных равной 4. Команда ADD ссылается на данные, адрес которых генерируется, как показано в следующем программном коде. Перед выполнением команды ADD опкоде загружается в регистр инструкций. DP и последние семь бит инструкции вместе формируют 16-битный адрес 0000 0010 0000 1001_2 (0209h). На рисунке 267 показано использование инструкцией ADD прямой адресации со сдвигом (от 0 до 15).

Пример 3 – Использование прямой адресации инструкцией ADD (сдвиг от 0 до 15)

```
    LDP #4   ;Установка страницы данных в 4(адрес 0200h-
027Fh).
    ADD 9h,5 ;Содержимое адреса данных 0209h
             ;сдвигается влево и прибавляется к
             ;содержимому аккумулятора
```

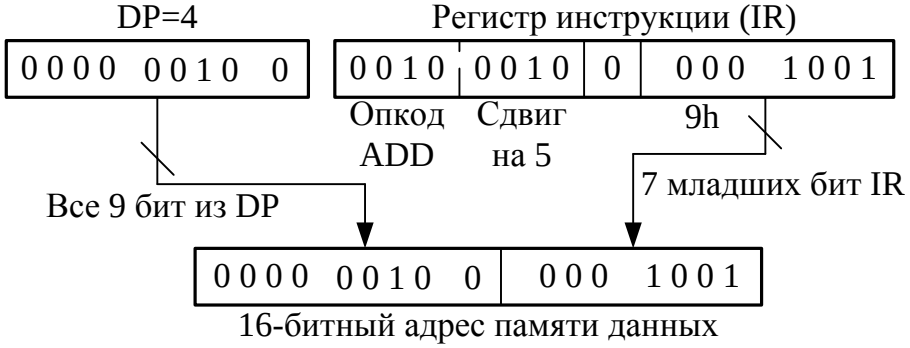


Рисунок 267 – Использование инструкцией ADD прямой адресации со сдвигом (от 0 до 15)

В примере 4 инструкция ADD ссылается на адрес памяти данных, который генерируется, как показано в следующем программном коде. Для любой инструкции, которая выполняет сдвиг на 16 бит, значение сдвига не включается прямо в слово инструкции; все восемь старших бит опкода определяют не только тип инструкции, но также определяют и сдвиг до 16 разрядов. На рисунке 268 показано использование инструкции ADD с прямой адресацией и со сдвигом на 16. В этом

примере 7-4 восемь бит слова инструкции определяют инструкцию ADD со сдвигом 16.

Пример 4 – Использование прямой адресации с ADD (сдвиг на 16).

```
    LDP  #5  ;Установка страницы данных на 5(адрес 0280h-
02FFh).
    ADD 9h,16 ;Содержимое адреса данных 0289h
              ;сдвигается влево на 16 бит и прибавляется к
              ;содержимому аккумулятора
```

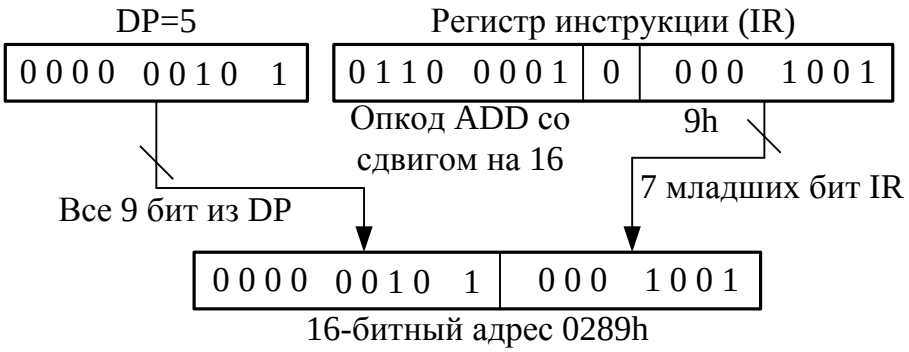


Рисунок 268 – Инструкция ADD с прямой адресацией и сдвигом на 16

В примере 5 инструкция ADDC ссылается на адрес памяти данных, который генерируется, как показано в следующем программном коде. Заметьте, что если инструкция не выполняет сдвиг, подобно этой инструкции ADDC, то все восемь бит содержат опкод типа инструкции. На рисунке 269 показано использование прямой адресации с инструкцией ADDC.

Пример 5 – Использование прямой адресации с инструкцией ADDC.

```
LDP #500 ;Установка страницы данных в 500
         ;(адрес FA00h-FA7Fh).
ADDC 6h  ;Содержимое адреса данных FA06h
         ;и значение бита переноса (C)
          ;прибавляется к содержимому аккумулятора.
```

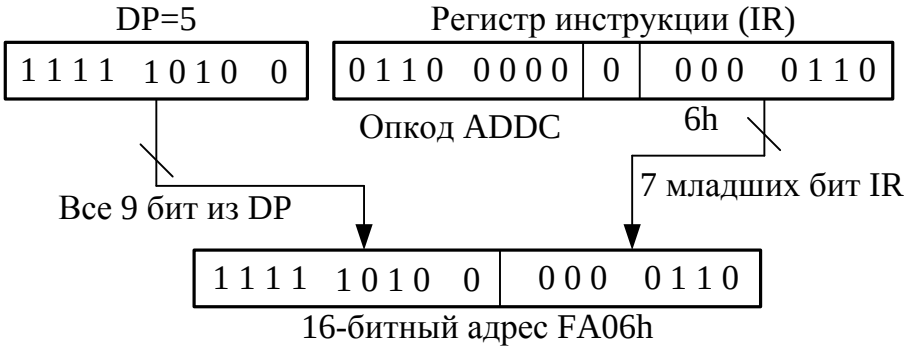


Рисунок 269 - Использование прямой адресации с инструкцией ADDC

14.7.3 Режим косвенной адресации

Восемь вспомогательных регистров (AR0–AR7) обеспечивают гибкий и мощный режим косвенной адресации (просто косвенной адресации). Любая ячейка в 64К слов памяти данных может быть доступна, используя 16-битный адрес, содержащийся во вспомогательном регистре.

Текущий вспомогательный регистр

Для выбора специфицированного вспомогательного регистра нужно загрузить 3-битный указатель вспомогательного регистра (ARP) статусного регистра ST0 значением от 0 до 7. ARP может загружаться как первичная операция инструкцией MAR или LST. ARP может загружаться как вторая операция любой инструкцией, которая поддерживает косвенную адресацию. Регистр, указанный в ARP, адресуется как текущий регистр или текущий AR. Во время выполнения инструкции содержимое текущего вспомогательного регистра используется как адрес на шине адреса чтения данных (DRAB), если инструкция требует чтение из памяти данных, или он передает адрес на адресную шину записи данных (DWAB), если инструкция требует запись данных в память данных. После этого инструкция использует значение текущего вспомогательного регистра как данные для ARAU, которая реализует 16-битную целочисленную беззнаковую арифметику для вычисления следующего значения вспомогательного регистра, например, увеличивать или уменьшать на 1. Обычно ARAU выполняет свои арифметические операции в фазе декодирования конвейера (когда инструкция, определяющая операции, декодируется).

Это позволяет адресу генерироваться перед фазой декодирования следующей инструкции. Имеется исключение для этого правила: во время выполнения инструкции NORM вспомогательный регистр и/или модификация ARP делается во время фазы выполнения конвейера. Для информации об операции конвейера
смотри 14.5.2.

Типы косвенной адресации

ИС 1867ВЦ10Т имеет четыре типа косвенной адресации:

– Безинкрементная или бездекрементная. Инструкция использует содержимое вспомогательного регистра как адрес памяти данных, но не инкрементирует, не декрементирует его содержимое.

– Инкремент или декремент на 1. Инструкция использует содержимое вспомогательного регистра как адрес памяти данных и после инкрементирует или декрементирует его содержимое на 1.

– Инкремент или декремент индексной величиной. Значение в AR0 – это величина индекса. Инструкция использует содержимое вспомогательного регистра как адрес памяти данных и после инкрементирует или декрементирует на величину индекса.

– Инкремент или декремент индексной величиной, используя реверсивный перенос. Значение в AR0 – это величина индекса. Инструкция использует содер-

жимое вспомогательного регистра как адрес памяти данных и после инкрементирует или декрементирует его содержимое на величину индекса. Сложение или вычитание в этом случае выполняется с реверсивным распространением переноса для реализации быстрого преобразования Фурье (БПФ).

Эти четыре типа адресации обеспечивают семь видов адресации в режиме косвенной адресации, перечисленные в таблице 131. Таблица 131 также показывает операнд инструкции, который соответствует каждому виду косвенной адресации и даны примеры, каким образом каждая опция используется.

Таблица 131 – Операнды косвенной адресации

Операнд	Опция	Пример
*	Без инкремента/ декремента	LT * загружает временный регистр (TREG) содержимым памяти данных, адресуемый текущим AR
*+	Инкремент на 1	LT *+ * загружает временный регистр (TREG) содержимым памяти данных, адресуемый текущим AR и после этого увеличивает на 1 содержимое текущего AR
*–	Декремент на 1	LT *– загружает временный регистр (TREG) содержимым памяти данных, адресуемый текущим AR и после этого уменьшает на 1 содержимое текущего AR
*0+	Инкремент на величину индекса	LT *0+ загружает временный регистр (TREG) содержимым памяти данных, адресуемый текущим AR и после этого добавляет содержимое AR0 к содержимому текущего AR
*0–	Декремент на величину индекса	LT *0– загружает временный регистр (TREG) содержимым памяти данных, адресуемый текущим AR и после этого вычитает содержимое AR0 из содержимого текущего AR
*BR0+	Инкремент на на величину индекса с реверсивным	LT *BR0+ загружает временный регистр (TREG) содержимым памяти данных, адресуемый текущим AR и после этого и после этого добавляет содержимое AR0 к содержимому текущего регистра AR с реверсивным распространением переноса

Окончание таблицы 131

Операнд	Опция	Пример
*BR0–	Декремент на величину индекса с реверсивным переносом	LT *BR0– загружает временный регистр (TREG) содержимым памяти данных, адресуемый текущим AR и после этого и после этого вычитает содержимое AR0 из содержимого текущего регистра AR с реверсивным распространением переноса

Все инкременты и декременты выполняются арифметическим устройством вспомогательных регистров (ARAU) в том же цикле, во время которого инструкция декодируется в конвейере. Бит-реверсная индексная адресация позволяет по-

высить эффективность операций ввода – вывода , переупорядочивая данные с основанием 2 для БПФ программ. Направление распространения переноса в ARAU реверсируется, когда выбирается адрес и AR0 добавляется или вычитается из текущего AR. Обычно использование этого режима адресации требует, чтобы AR0 был установлен первым в соответствующее значение половины размера области, а текущий AR должен быть установлен значением базового адреса данных (указатель на первые данные.

Следующий вспомогательный регистр

В дополнение к обновлению текущего вспомогательного регистра множество инструкций может также специфицировать следующий вспомогательный регистр AR. Этот регистр будет текущим вспомогательным регистром, когда инструкция выполнится полностью. Инструкции, которые указывают следующий текущий регистр, загружают ARP новым значением. Когда ARP загружается новым значением, то предыдущее значение ARP загружается в буфер указателя вспомогательного регистра (auxiliary register pointer buffer – ARB). Пример 6 иллюстрирует выбор следующего вспомогательного регистра, а также обсуждается другая косвенная адресация.

Пример 6 – Выбор нового текущего AR.

```
     MAR*,AR1 ;Загрузка ARP 1, чтобы сделать AR1 текущим
              ;вспомогательным регистром.
     LT *+,AR2 ;AR2 есть следующий вспомогательный регистр.
              ;Загрузка TREG содержимым адреса, указанного
AR1,
              ;добавление 1 к содержимому AR1, после этого
              ;сделать AR2 текущим вспомогательным реги-
         стром.
     MPY*     ;Умножение TREG на содержимое адреса
              ;указанного AR2.
```

Формат опкода инструкции с косвенной адресацией

На рисунке 270 показан формат слова инструкции, загружаемой в регистр инструкций, когда инструкция использует косвенную адресацию. Поля опкода описаны в таблицах 132 и 133.

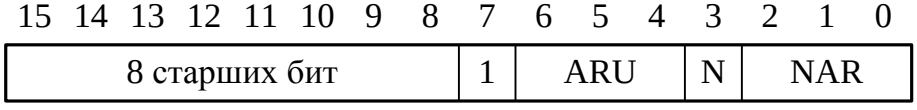


Рисунок 270 – Содержимое регистра инструкций с косвенной адресацией

Таблица 132 – Описание бит инструкции

Биты	Мнемоника	Описание

Биты	Мнемоника	Описание
15-08	8 MSBs	Тип инструкции (например, LT), а также содержит информацию относительно сдвигов данных
07	DIR/INDIR	Этот бит, установленный в 1, определяет косвенный режим адресации
06-04	ARU	Это поле определяет режим косвенной адресации – будет ли и как увеличиваться/уменьшаться текущий вспомогательный регистр
3	N	Этот бит определяет будет ли меняться значение ARP. 0 – содержимое ARP остается неизменным. 1 – содержимое NAR загружается в ARP и старое значение ARP загружается в ARB статусного регистра ST1
02-00	NAR	Это поле определяет следующий текущий вспомогательный регистра. Содержимое NAR загружается в ARP, если N = 1

Таблица 133 - Эффекты кода ARU на текущий вспомогательный регистр

Код ARU			Арифметическая операция, исполняемая на текущем AR
6	5	4	
0	0	0	Нет операций над текущим
0	0	1	Текущий AR - 1 → текущий AR
0	1	0	Текущий AR + 1 → текущий AR
0	1	1	Резерв
1	0	0	Текущий AR – AR0 → □□текущий AR (реверсивное/обратное распространение переноса)
1	0	1	Текущий AR – AR0 текущий AR
1	1	0	Текущий AR + AR0 →□□текущий
1	1	1	Текущий AR + AR0 →□□текущий (реверсивное.обратное распространение переноса)

В таблице 134 показаны биты поля опкода и нотация, используемая для косвенной адресации. Она также показывает соответствующие операции, выполняемые на текущем вспомогательном регистре и ARP.

Таблица 134 – Описание бит инструкции при косвенной адресации

Биты опкода инструкции			Операнды	Операция
15 8	7 6 5 4 3	2 1 0		
MSB	1 0 0 0 0	NAR	*	
MSB	1 0 0 0 1	NAR	*, ARn	NAR → AR
MSB	1 0 0 1 0	NAR	*-	Текущий AR – 1 → AR
MSB	1 0 0 1 1	NAR	*-, ARn	Текущий AR – 1 → AR, NAR → ARP
MSB	1 0 1 0 0	NAR	*+	Текущий AR + 1 → AR
MSB	1 0 1 0 1	NAR	*+, ARn	Текущий AR + 1 → AR, NAR → ARP

Биты опкода инструкции			Операнды	Операция
MSB	1 1 0 0 0	NAR	*BR0-	Текущий AR – rcAR0 → Текущий AR
MSB	1 1 0 0 1	NAR	*BR0-, ARn	Текущий AR – rcAR0 → Текущий AR, NAR → ARP
MSB	1 1 0 1 0	NAR	*0-	Текущий AR – AR0 → Текущий AR
MSB	1 1 0 1 1	NAR	*0-, ARn	Текущий AR – AR0 → Текущий AR, NAR → ARP
MSB	1 1 1 0 0	NAR	*0+	Текущий AR + AR0 → Текущий AR
MSB	1 1 1 0 1	NAR	*0+, ARn	Текущий AR + AR0 → Текущий AR, NAR → ARP
MSB	1 1 1 1 0	NAR	*BR0+	Текущий AR + rcAR0 → Текущий AR
MSB	1 1 1 1 1	NAR	*BR0+ ARn	Текущий AR + rcAR0 → Текущий AR, NAR → ARP

Примечание – rc – реверсивное распространение переноса
NAR – следующий AR
n – 0, 1, 2, …, 7
MSB - старшие значащие биты слова инструкции
→ - загружается в

Примеры косвенной адресации

В примере 7 инструкция ADD выбирается из программной памяти, а регистр инструкций загружается значением, показанным в примере. На рисунке 271 показана косвенная адресация без инкрементирования или декрементирования.

Пример 7 – Косвенная адресация без изменения текущего вспомогательного регистра

```
ADD *,8   ;Прибавить к аккумулятору Add содержимое
          ;программной памяти адресованной
          ;текущим вспомогательным регистром. Данные
          ;сдвигаются влево на восемь бит перед сложе-
    нием
```

Рисунок 271 – Косвенная адресация без инкрементирования или декрементирования

В примере 8 инструкция ADD выбирается из памяти программ, а регистр инструкций загружается значением, показанным в примере. На рисунке 272 показана косвенная адресация с инкрементом на 1.

Пример 8 – Косвенная адресация с инкрементом текущего вспомогательного регистра на 1

```
ADD *+,8,AR4  ;инструкция выполняется также как в
              ;примере 7, но в дополнение текущий
              ;вспомогательный регистр увеличивается на
         1
              ;AR4 выбирается как следующий вспомога-
         тельный
              ;регистр.
```

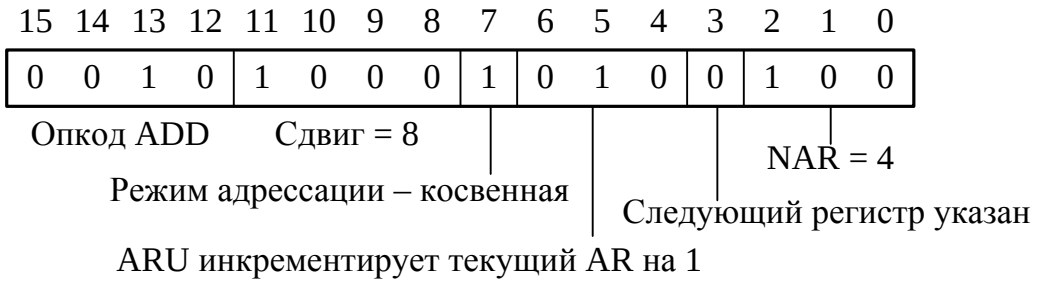


Рисунок 272 – Косвенная адресация с инкрементом на 1

Пример 9 – Косвенная адресация – декремент 1

```
ADD *-,8 ;операторы как в примере7, но в дополнение,
текущий
          ;вспомогательный регистр декрементируется 1.
```

Пример 10 – Косвенная адресация – инкрементируется индексом

```
ADD *0+,8 ;операторы как в примере 7, но в дополнение,
          ;содержимое регистра AR0 прибавляется к теку-
     щему
          ;вспомогательному регистру.
```

Пример 11 – Косвенная адресация – декремент индексом.

```
ADD *0+,8 ;операторы как в примере 7, но в дополнение,
          ;содержимое регистра AR0 вычитается из теку-
     щего
          ;вспомогательного регистра.
```

Пример 12 – Косвенная адресация – инкремент индексом с реверсивным распространением переноса.

```
    ADD *BR0+,8   ;операторы как в примере 10, но в допол-
нение,
                  ;содержимое регистра AR0 прибавляется к
                  ;текущему вспомогательному регистру с
                  ;реверсивным распространением переноса.
```

Пример 13 – Косвенная адресация – декремент индексом с реверсивным распространением переноса

```
    ADD *BR0-,8   ;операторы как в примере 11, но в допол-
нение,
                  ;содержимое регистра AR0 вычитается из
                  ;текущего вспомогательного регистра с
                  ;реверсивным распространением переноса.
```

Модификация содержимого вспомогательного регистра

Инструкции LAR, ADRK, SBRK и MAR – это специализированные инструкции для изменения содержимого вспомогательного регистра (AR):

– инструкция LAR загружает AR;

– инструкция ADRK добавляет непосредственное значение к AR; а SBRK вычитает непосредственное значение;

– инструкция MAR может инкрементировать или декрементировать AR значением 1 или индексом.

Однако, вспомогательные регистры могут модифицироваться любой инструкцией, которая поддерживает операнды с косвенной адресацией. (Косвенная адресация может использоваться со всеми инструкциями за исключением тех, которые имеют непосредственные операнды или без операндов).

14.8 Ассемблерные инструкции

В этом разделе описываются инструкции языка ассемблера ИС 1867ВЦ10Т. Набор инструкций ИС 1867ВЦ10Т совместим с набором инструкций ИС 1867ВМ2. Код, написанный для 1867ВМ2, может быть реассемблирован для работы на 1867ВЦ10Т.

Набор инструкций 1867ВЦ10Т является подмножеством набора инструкций 1867ВЦ2Т, поэтому после модернизации код для 1867ВЦ10Т может быть запущен на ИС 1867ВЦ2Т.

14.8.1 Список инструкций

В этом разделе представлены шесть таблиц (таблицы 135 – 140), в которых инструкции сгруппированы в соответствии с их функциональным назначением:

– Инструкции, оперирующие с аккумулятором, арифметические и логические инструкции (таблица 135).

– Инструкции, оперирующие с вспомогательными регистрами (таблица 136).

– Инструкции, оперирующие с регистрами TREG и PREG, и инструкции умножения (таблица 137).

– Инструкции переходов (таблица 138).

– Управляющие инструкции (таблица 139).

– Инструкции ввода-вывода и инструкции работы с памятью (таблица 140).

В таблицах инструкции расположены в алфавитном порядке. Число слов, которые инструкция занимает в памяти программ, указано в третьем столбце, а число циклов, которые инструкция требует для выполнения, - в четвертом столбце. Предполагается, что все инструкции выполняются из внутренней памяти программ (RAM) и внутренней памяти данных двойного доступа (DARAM). Количество циклов приведено для однократного выполнения инструкции, а не в режиме повтора. Подробная информация о каждой инструкции представлена в 14.8.3.

Ниже приведена справка о символах (сокращениях), используемых в таблицах этого раздела:

ACC	аккумулятор.
AR	вспомогательный регистр.
ARX	3-битное значение в инструкциях LAR и SAR для определения вспомогательного регистра, который будет загружаться инструкцией (LAR) или сохраняться инструкцией (SAR).
BITX	4-битное значение (называемое “код бита”), которое определяет, какой бит из указанного значения памяти данных будет тестироваться инструкцией BIT.
CM	2-битное значение. Определяет тип сравнения, выполняемого инструкцией CMPR: Если CM = 00, инструкция проверяет (текущий AR = AR0) Если CM = 01, инструкция проверяет (текущий AR < AR0) Если CM = 10, инструкция проверяет (текущий AR > AR0) Если CM = 11, инструкция проверяет (текущий AR ≠ AR0)
IAAA AAAA	(За I следует семь A). Бит I отражает способ адресации: I=0 – прямая адресация, I=1 – косвенная адресация. Когда используется прямая адресация, то 7 A - это младшие биты (LSB) адреса в странице памяти данных. Для косвенной адресации 7 A – это биты, определяющие режим косвенной адресации и способ манипуляции вспомогательными регистрами (см. 14.7.3 «Режим косвенной адресации»).
IIII IIII	(восемь I) - 8-битная константа, используемая в короткой непосредственной адресации.

I IIII IIII	(девять I) - 9-битная константа, используемая для короткой непосредственной адресации в инструкции LDP.
I IIII IIII IIII	(тринадцать I) - 13-битная константа, используемая в короткой непосредственной адресации для инструкции MPY.
INTR#	5-битное значение, представляет число от 0 до 31. Инструкция INTR использует это число для перехода программного управления по одному из 32 адресов векторов прерывания.
PM	2-битное значение, копируемое в PM-биты статусного регистра ST1 инструкцией SPM.
SHF	3-битное значение левого сдвига.
SHFT	4-битное значение левого сдвига.
TP	2-битное значение, используемое условными инструкциями для представления четырех условий: TP = 00 – вывод BIO низкий. TP = 01 – TC = 1. TP = 10 – TC=0. TP = 11 – без условий.
ZLVC ZLVC	два 4-битных поля, каждое из которых представляет следующие условия: Бит Условие Z ACC = 0 L ACC < 0 V Переполнение C Перенос Условная инструкция содержит два 4-битных поля. Младшие 4 бита инструкции – это поле маски. 1 в соответствующем бите маски указывает, какое из условий будет проверяться. Второе 4-битное поле (биты 4-7) указывает состояние тестируемых условий. Например, для проверки ACC ≥ 0 необходимо в младшем 4-битном поле установить биты Z и L в 1, а биты V и C установить в 0. Во втором поле бит Z устанавливается в 1 для тестирования условия ACC = 0, а бит L сбрасывается в 0 для тестирования условия ACC > 0. Возможные условия, задаваемые этими восемью битами, показаны в описании для инструкций BCND, CC и RETC.
+ 1 word	второе слово 2-словного опкода инструкции. Это второе слово содержит 16-битную константу. В зависимости от инструкции эта константа имеет либо длинное непосредственное значение,

адрес памяти программ или адрес порта ввода - вывода или картрируемый в память регистр ввода-вывода.

Таблица 135 - Инструкции, оперирующие с аккумулятором, арифметические и логические инструкци

Мнемоника	Описание	Слов	Циклов	Опкод
ABS	Абсолютное значение ACC	1	1	1011 1110 0000 0000
ADD	Прибавление к ACC операнда со сдвигом от 0 до 15, прямая/косвенная адресация	1	1	0010 SHFT IAAA AAAA
	Прибавление к ACC операнда со сдвигом от 0 до 15, длинная непосредственная адресация	2	2	1011 1111 1001 SHFT + 1 word
	Прибавление к ACC операнда со сдвигом 16, прямая/косвенная адресация	1	1	0110 0001 IAAA AAAA
	Прибавление к ACC операнда, короткая непосредственная адресация	1	1	1011 1000 IIII IIII
ADDC	Прибавление к ACC операнда с переносом, прямая/косвенная адресация	1	1	0110 0000 IAAA AAAA
ADDS	Прибавление к младшему слову ACC операнда с подавлением знакового расширения, прямая/косвенная адресация	1	1	0110 0010 IAAA AAAA
ADDT	Прибавление к ACC операнда со сдвигом от 0 до 15, указанным в TREG, прямая/косвенная адресация	1	1	0110 0011 IAAA AAAA
AND	Логическое «И» младшего слова ACC со значением памяти данных, прямая/косвенная адресация	1	1	0110 1110 IAAA AAAA
	Логическое «И» ACC с операндом со сдвигом от 0 до 15, длинная непосредственная адресация	2	2	1011 1111 1011 SHFT + 1 word
CMPL	Инверсия ACC	1	1	1011 1110 0000 0001

Продолжение таблицы 135

Мнемоника	Описание	Слов	Циклов	Опкод
LACC	Загрузка в ACC значения памяти данных со сдвигом от 0 до 15, прямая/косвенная адресация	1	1	0001 SHFT IAAA AAAA
	Загрузка в ACC операнда со сдвигом от 0 до 15, длинная непосредственная адресация	2	2	1011 1111 1000 SHFT + 1 word
LACL	Загрузка в младшее слово ACC операнда, прямая/косвенная адресация	1	1	0110 1001 IAAA AAAA

	Загрузка в младшего слова ACC операнда, короткая непосредственная адресация	1	1	1011 1001 IIII IIII
LACT	Загрузка в ACC операнда со сдвигом от 0 до 15, указанным в TREG, прямая/косвенная адресация	1	1	0110 1011 IAAA AAAA
NEG	Умножение ACC на –1 (изменение знака ACC)	1	1	1011 1110 0000 0010
NORM	Нормализация содержимого ACC, косвенная адресация	1	1	1010 0000 IAAA AAAA
OR	Логическое «ИЛИ» младшего слова ACC с операндом, прямая/косвенная адресация	1	1	0110 1101 IAAA AAAA
	Логическое «ИЛИ» ACC с операндом со сдвигом от 0 до 15, длинная непосредственная адресация	2	2	1011 1111 1100 SHFT + 1 word
	Логическое «ИЛИ» ACC с операндом со сдвигом 16, длинная непосредственная адресация	2	2	1011 1110 1000 0010 + 1 word
ROL	Циклический сдвиг ACC влево	1	1	1011 1110 0000 1100
ROR	Циклический сдвиг ACC вправо	1	1	1011 1110 0000 1101
SACH	Сохранение старшего слова ACC со сдвигом от 0 до 7, прямая/косвенная адресация	1	1	1001 1SHF IAAA AAAA
SACL	Сохранение младшего слова ACC со сдвигом от 0 до 7, прямая/косвенная адресация	1	1	1001 0SHF IAAA AAAA
SFL	Сдвиг ACC влево	1	1	1011 1110 0000 1001
SFR	Сдвиг ACC вправо	1	1	1011 1110 0000 1010
SUB	Вычитание из ACC операнда со сдвигом от 0 до 15, прямая/косвенная адресация	1	1	0011 SHFT IAAA AAAA
	Вычитание из ACC операнда со сдвигом от 0 до 15, длинная непосредственная адресация	2	2	1011 1111 1010 SHFT + 1 word

Окончание таблицы 135

Мнемоника	Описание	Слов	Циклов	Опкод
SUB	Вычитание из ACC операнда со сдвигом 16, прямая/косвенная адресация	1	1	0110 0101 IAAA AAAA
	Вычитание из ACC операнда, короткая непосредственная адресация	1	1	1011 1010 IIII IIII
SUBB	Вычитание из ACC операнда с заемом, прямая/косвенная адресация	1	1	0110 0100 IAAA AAAA
SUBC	Условное вычитание операнда, прямая/косвенная адресация	1	1	0000 1010 IAAA AAAA

SUBS	Вычитание из ACC операнда с запрещением расширения знака, прямая/косвенная адресация	1	1	0110 0110 IAAA AAAA
SUBT	Вычитание из ACC операнда со сдвигом, указанным в TREG, прямая/косвенная адресация	1	1	0110 0111 IAAA AAAA
XOR	«Исключающее ИЛИ» младшего слова ACC с операндом, прямая/косвенная адресация	1	1	0110 1100 IAAA AAAA
	«Исключающее ИЛИ» ACC с операндом со сдвигом от 0 до 15, длинная непосредственная адресация	2	2	1011 1111 1101 SHFT + 1 word
	«Исключающее ИЛИ» ACC с операндом со сдвигом 16, длинная непосредственная адресация	2	2	1011 1110 1000 0011 + 1 word
ZALR	Сброс младшего слова ACC и загрузка старшего слова ACC с округлением, прямая/косвенная адресация	1	1	0110 1000 IAAA AAAA

Таблица 136 – Инструкции вспомогательных регистров

Мнемоника	Описание	Слов	Циклов	Опкод
ADRK	Прибавление константы к текущему AR, короткая непосредственная адресация	1	1	0111 1000 IIII IIII
BANZ	Переход, если текущий AR не равен 0, косвенная адресация	2	4 (УВ) 2 (УНВ)	0111 1011 1AAA AAAA + 1 word
CMPR	Сравнение текущего AR с AR0	1	1	1011 1111 0100 01CM
LAR	Загрузка указанного AR операндом, прямая/косвенная адресация	1	2	0000 0ARX IAAA AAAA
	Загрузка указанного AR операндом, короткая непосредственная адресация	1	2	1011 0ARX IIII IIII
LAR	Загрузка указанного AR операндом, длинная непосредственная адресация	2	2	1011 1111 0000 1ARX + 1 word

Окончание таблицы 136

Мнемоника	Описание	Слов	Циклов	Опкод
MAR	Модификация текущего AR и/или ARP при косвенной адресации (при прямой адресации аналог NOP)	1	1	1000 1011 IAAA AAAA
SAR	Сохранение указанного AR по указанному адресу, прямая/косвенная адресация	1	1	1000 0ARX IAAA AAAA
SBRK	Вычитание операнда из текущего AR, короткая непосредственная адресация	1	1	0111 1100 IIII IIII

Примечание - УВ – условие выполнено, УНВ – условие не выполнено.

Таблица 137 – TREG, PREG и инструкции умножения

Мнемоника	Описание	Слов	Циклов	Опкод
APAC	Прибавить PREG к ACC	1	1	1011 1110 0000 0100
LPH	Загрузка старшего слова PREG операндом, прямая/косвенная адресация	1	1	0111 0101 IAAA AAAA
LT	Загрузка PREG операндом, прямая/косвенная адресация	1	1	0111 0011 IAAA AAAA
LTA	Загрузка TREG операндом, сложение ACC с PREG, прямая/косвенная адресация	1	1	0111 0000 IAAA AAAA
LTD	Загрузка TREG операндом, сложение ACC с PREG, пересылка операнда по адресу операнда + 1, прямая/косвенная адресация	1	1	0111 0010 IAAA AAAA
LTP	Загрузка TREG операнда, сохранение PREG в ACC, прямая/косвенная адресация	1	1	0111 0001 IAAA AAAA
LTS	Загрузка TREG операнда, вычитание PREG из ACC, прямая/косвенная адресация	1	1	0111 0100 IAAA AAAA
MAC	Умножение и накопление, прямая/косвенная адресация	2	3	1010 0010 IAAA AAAA + 1 word
MACD	Умножение и накопление, пересылка данных, прямая/косвенная адресация	2	3	1010 0011 IAAA AAAA + 1 word
MPY	Умножение TREG на операнд, прямая/косвенная адресация	1	1	0101 0100 IAAA AAAA
	Умножение TREG на операнд (13-битная константа), короткая непосредственная адресация	1	1	110I IIII IIII IIII
MPYA	Умножение с накоплением предыдущего произведения, прямая/косвенная адресация	1	1	0101 0000 IAAA AAAA
MPYU	Беззнаковое умножение, прямая/косвенная адресация	1	1	0101 0101 IAAA AAAA

Окончание таблицы 137

Мнемоника	Описание	Слов	Циклов	Опкод
MPYS	Умножение с вычитанием из предыдущего произведения, прямая/косвенная адресация	1	1	0101 0001 IAAA AAAA
PAC	Загрузка ACC значением регистра PREG	1	1	1011 1110 0000 0011
SPAC	Вычитание значениия регистра PREG из ACC	1	1	1011 1110 0000 0101

SPH	Сохранение старшего слова регистра PREG в памяти данных, прямая/косвенная адресация	1	1	1000 1101 IAAA AAAA
SPL	Сохранение младшего слова регистра PREG в памяти данных, прямая/косвенная адресация	1	1	1000 1100 IAAA AAAA
SPM	Установить режим сдвига произведения	1	1	1011 1111 0000 00PM
SQRA	Вычисление квадрата операнда с накоплением в ACC предыдущего произведения, прямая/косвенная адресация	1	1	0101 0010 IAAA AAAA
SQRS	Вычисление квадрата операнда, вычитание из ACC предыдущего произведения, прямая/косвенная адресация	1	1	0101 0011 IAAA AAAA

Таблица 138 - Инструкции перехода

Мнемоника	Описание	Слов	Циклов	Опкод
B	Безусловный переход, косвенная адресация	2	4	0111 1001 1AAA AAAA + 1 word
BACC	Переход по адресу, указанному в ACC	1	4	1011 1110 0010 0000
BANZ	Переход, если текущий AR не равен 0, косвенная адресация	2	4 (УВ) 2 (УНВ)	0111 1011 1AAA AAAA + 1 word
BCND	Условный переход	2	4 (УВ) 2 (УНВ	1110 00TP ZLVC ZLVC + 1 word
CALA	Вызов подпрограммы с адресом, указанным в ACC	1	4	1011 1110 0011 0000
CALL	Вызов подпрограммы, косвенная адресация	2	4	0111 1010 1AAA AAAA + 1 word
INTR	Программное прерывание	1	4	1011 1110 011I NTR#
NMI	Немаскируемое программное прерывание	1	4	1011 1110 0101 0010
RET	Возврат из подпрограммы	1	4	1110 1111 0000 0000
RETC	Условный возврат из подпрограммы	1	4 (УВ) 2 (УНВ	1110 11TP ZLVC ZLVC
TRAP	Программное прерывание	1	4	1011 1110 0101 0001
Примечание - УВ – условие выполнено, УНВ – условие не выполнено.				

Таблица 139 – Инструкции управления

Мнемоника	Описание	Слов	Циклов	Опкод
BIT	Проверка бита в операнде, код бита указан в инструкции, прямая/косвенная адресация	1	1	0100 BITX IAAA AAAA
BITT	Проверка бита в операнде, код бита указан в регистре TREG, прямая/косвенная адресация	1	1	0110 1111 IAAA AAAA
CLRC	Очистка управляющего бита			

Мнемоника	Описание		Слов	Циклов	Опкод
	CLRC C	Очистка бита C	1	1	1011 1110 0100 1110
	CLRC CNF	Очистка бита CNF	1	1	1011 1110 0100 0100
	CLRC INTM	Очистка бита INTM	1	1	1011 1110 0100 0000
	CLRC OVM	Очистка бита OVM	1	1	1011 1110 0100 0010
	CLRC SXM	Очистка бита SXM	1	1	1011 1110 0100 0110
	CLRC TC	Очистка бита TC	1	1	1011 1110 0100 1010
	CLRC XF	Очистка бита XF	1	1	1011 1110 0100 1100
IDLE	Ожидание прерывания		1	1	1011 1110 0010 0010
LDP	Загрузка указателя страниц, прямая/косвенная адресация		1	2	0000 1101 IAAA AAAA
	Загрузка указателя страниц, короткая непосредственная адресация		1	2	1011 110I IIII IIII
LST	Загрузка статусного регистра ST0, прямая/косвенная адресация		1	2	0000 1110 IAAA AAAA
	Загрузка статусного регистра ST1, прямая/косвенная адресация		1	2	0000 1111 IAAA AAAA
NOP	Нет операции		1	1	1000 1011 0000 0000
POP	Записать содержимое верхушки стека в младшее слово ACC		1	1	1011 1110 0011 0010
POPD	Записать содержимое верхушки стека в ячейку памяти данных, прямая/косвенная адресация		1	1	1000 1010 IAAA AAAA
PSHD	Записать значение ячейки памяти данных в стек, прямая/косвенная адресация		1	1	0111 0110 IAAA AAAA
PUSH	Записать младшее слово ACC в стек		1	1	1011 1110 0011 1100
RPT	Повтор следующей инструкции n+1 раз, n указано в операнде, прямая/косвенная адресация		1	1	0000 1011 IAAA AAAA
	Повтор следующей инструкции n+1 раз, n указано в операнде, короткая непосредственная адресация		1	1	1011 1011 IIII IIII
SETC	Установка управляющего бита				
	SETC C	Установка бита C	1	1	1011 1110 0100 1111
	SETC CNF	Установка бита CNF	1	1	1011 1110 0100 0101
	SETC INTM	Установка бита INTM	1	1	1011 1110 0100 0001
	SETC OVM	Установка бита OVM	1	1	1011 1110 0100 0011
	SETC SXM	Установка бита SXM	1	1	1011 1110 0100 0111

Окончание таблицы 139

Мнемоника	Описание		Слов	Циклов	Опкод
SETC	SETC TC	Установка бита TC	1	1	1011 1110 0100 1011
	SETC XF	Установка бита XF	1	1	1011 1110 0100 1101
SPM	Установка режима сдвига произведения		1	1	1011 1111 0000 00PM

SST	Сохранение статусного регистра ST0, прямая/косвенная адресация	1	1	1000 1110 IAAA AAAA
	Сохранение статусного регистра ST1, прямая/косвенная адресация	1	1	1000 1111 IAAA AAAA

Таблица 140 – Инструкции ввода - вывода и инструкции работы с памятью

Мнемоника	Описание	Слов	Циклов	Опкод
BLDD	Копирование содержимого памяти данных, адресуемой длинной непосредственной адресацией, в ячейку памяти данных, адресуемую прямо/косвенно	2	3	1010 1000 IAAA AAAA + 1 word
	Копирование содержимого памяти данных, адресуемой прямо/косвенно, в ячейку памяти данных, адресуемую длинной непосредственной адресацией	2	3	1010 1001 IAAA AAAA + 1 word
BLPD	Копирование содержимого памяти программ, адресуемой вторым словом инструкции, в ячейку память данных, адресуемую прямо/косвенно	2	3	1010 0101 IAAA AAAA + 1 word
DMOV	Пересылка данных из ячейки памяти данных, адресуемой прямо/косвенно в следующую ячейку памяти данных	1	1	0111 0111 IAAA AAAA
IN	Чтение данных из порта ввода - вывода, адресуемого 2 словом инструкции в память данных, адресуемую прямо/косвенно	2	2	1010 1111 IAAA AAAA + 1 word
OUT	Запись данных в порт ввода - вывода, адресуемый 2 словом инструкции из памяти данных, адресуемой прямо/косвенно	2	3	0000 1100 IAAA AAAA + 1 word
SPLK	Сохранение константы (второе слово инструкции) в памяти данных, адресуемой прямо/косвенно	2	2	1010 1110 IAAA AAAA + 1 word
TBLR	Копирование содержимого памяти программ, адресуемой младшим словом ACC, в память данных, адресуемую прямо/косвенно	1	3	1010 0110 IAAA AAAA

14.8.2 Использование описаний инструкций

В этом подраздел содержится подробная информация о наборе инструкций. Описание для каждой инструкции представляет собой следующие категории информации:

– Синтаксис.
– Код операции.
– Операнды
– Выполнение.
– Статусные биты.
– Описание.
– Слова (длина инструкции).
– Циклы.
– Примеры.

Синтаксис

Описание каждой инструкции начинается со списка синтаксически правильных выражений на языке ассемблера и типа(ов) режима адресации для каждого выражения. Для примера, описание для инструкции ADD начинается с:

```
ADD dma [, shift ]                ;Прямая адресация
ADD dma, 16                  ;Прямая адресация со сдвигом 16
ADD ind [, shift [, ARn] ]        ;Косвенная адресация
ADD ind, 16 [, ARn]          ;Косвенная адресация со сдвигом 16
ADD #k                       ;Короткая непосредственная адреса-
ция
ADD #lk [, shift ]           ;Длинная непосредственная адресация
```

Нотация, используемая в синтаксических выражениях:

– Символы курсивом в синтаксисе инструкции используются для обозначения переменной.

Для примера, синтаксис ADD dma позволяет использовать различные значения для dma (любое из значений, приведенных в примерах).

Примеры:

```
ADD DAT
ADD 15
```

– Жирные символы в синтаксисе инструкции должны набираться, как показано.

Пример:

Для синтаксиса

```
ADD dma, 16
```

можно использовать различные значения для dma, но слово ADD и число 16 должны набираться, как показано ниже:

```
ADD 7h, 16
ADD X, 16
```

– [, x] – необязательный операнд x

Пример:

Для синтаксиса

```
ADD dma [, shift]
```

можно подставить для dma как в инструкции, приведенной ниже:

```
ADD 7h
```

а можно добавить необязательный элемент, добавляя значение сдвига как в инструкции

```
ADD 7h, 5
```

– [, x1 [, x2]] – операнды x1 и x2 являются необязательными, но нельзя включить только элемент x2 без включения элемента x1.

Пример:

Для синтаксиса

```
ADD ind, [, shift [, ARn]]
```

необходимо включить ind как в инструкции

```
ADD *+
```

Можно включить необязательный сдвиг, как в инструкции

```
ADD *+, 5
```

Если нужно включить необязательный элемент ARn, то необходимо также включить и сдвиг как в инструкции

```
ADD *+, 0, AR2
```

– # - этот символ является префиксом для константы, используемой в непосредственной адресации. Для короткого/длинного непосредственного операнда этот символ используется в инструкции для устранения неоднозначности с другими режимами адресации.

Пример:

```
RPT #15
```

Инструкция использует короткую непосредственную адресацию. Эта инструкция повторяет следующую инструкцию 16 раз.

```
RPT 15
```

Инструкция использует прямую адресацию. Количество повторений следующей инструкции определяется значением в памяти данных.

В заключение рассмотрим пример кода:

```
MoveData BLDD DAT5, #310h ;пересылка данных из адреса
                          ;указаного в DAT5 в адрес
310h.
```

Обратите внимание, что необязательная метка MoveData используется как указатель перед началом мнемоники инструкции. Можно размещать метку или перед мнемоникой инструкции на той же самой линии, или на предыдущей линии в первой колонке. (Нужно убедиться, что в имени метки отсутствует пробел). Поле необязательного комментария может включать синтаксическое выражение. По

крайней мере, один пробел требуется между полями (метка, мнемоника, операнд или комментарий).

Операнды

Операнды могут быть константами или выражением, вычисляемым во время ассемблирования (компиляции), указывающими на память, порты ввода - вывода, адреса регистров, указатели, размер сдвига и на различные другие константы. Подраздел операндов в описании каждой инструкции определяет переменные, используемые в синтаксическом выражении.

Например, для инструкции ADD синтаксическая категория дает эти синтаксические выражения:

```
ADD dma [, shift ]                ;Прямая адресация
ADD dma, 16                   ;Прямая адресация со сдвигом 16
ADD ind [, shift [, ARn] ]        ;Косвенная адресация
ADD ind, 16 [, ARn]           ;Косвенная адресация со сдвигом 16
ADD #k                        ;Короткая непосредственная адреса-
ция
ADD #lk [, shift ]            ;Длинная непосредственная адресация
```

Категория операндов определяет переменные dma, shift, ind, n, k и lk. Для ind, косвенно адресуемой переменной, можно подставить один из семи символов:

```
*    *+    *-    *0+    *0-    *BR0+    *BR0-
```

Эти символы описаны в 14.7.3.

Код команды

Информация кода команды (опкода) в описании инструкции классифицирует различные поля битов, которые составляют каждое слово инструкции. Когда одно из полей содержит константу, полученную прямо из операнда, то поле имеет то же самое имя, что и этот операнд. Содержимое полей, которые не связаны напрямую с операндами, имеют другие имена; описание опкода или поясняет эти имена сразу или отсылает к разделу в этого документа, который поясняет их в деталях. Для примера, эти опкоды, данные для инструкции ADDC.

```
ADDC dma
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	0	0	dma						

```
ADDC ind [, ARn]
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	0	1	ARU			N	NAR		

Поле, называемое dma, содержит значение dma, которое определяется в категории операндов. Содержимое полей ARU, N и NAR заполняется из операндов ind и n, но прямо не соответствует этим операндам; однако, примечание направляет вас к подходящему разделу для детализации.

Выполнение

Исполнение (выполнение) показывает последовательность действий, происходящих при выполнении инструкции. Если выполняемое событие или события зависят от используемого метода адресации, то указывается какие события связываются с какими методами адресации. Далее приведена используемая нотация:

`(r)` содержимое регистра или ячейки r.
Пример: `(ACC)`
представляет значение аккумулятора.

`x → y` значение x назначается регистру или ячейке y
Пример: `(data-memory address) → ACC`
означает: содержимое из указанного адреса памяти данных записывается в аккумулятор.

`r(n:m)` биты от `n` до `m` регистра или ячейки r.
Пример: `ACC(15:0)`
представляет биты с 15 по 0 аккумулятора.

`(r(n:m))` содержимое бит от `n` до `m` регистра или ячейки r.
Пример: `ACC(31:16)`
представляет содержимое бит с 31 по 16 аккумулятора.

`nnh` указывает, что nn представляет собой шестнадцатиричное число.

Статусные биты

Биты в статусных регистрах ST0 и ST1 влияют на выполнение соответствующих инструкций и изменяются определенными инструкциями. Информация «Статусные биты» в описании каждой инструкции определяет, какие биты (если таковые имеются) влияют на выполнение инструкции и какие из бит (если таковые имеются) изменяются инструкцией.

Описание

«Описание» объясняет, что происходит во время выполнения инструкции, и ее влияние на весь процессор или на содержимое памяти. Здесь также обсуждаются различные ограничения на операнды, накладываемые процессором или ассемблером. Описание дополняет информацию, данную в «Выполнении».

Слова (длина инструкции)

«Длина инструкции в словах» указывает число слов памяти, требуемых для хранения инструкции (одно или два), указывает, какие режимы адресации требуют одно слово, а какие два.

Циклы

В описании каждой инструкции содержится информация, показывающая число машинных циклов (период CLKOUT1), требуемых для выполнения инструкции в данной конфигурации памяти как для одиночной, так и повторяемой инструкцией RPT.

Пример:

Циклов для одиночной инструкции

Операнд	Программа		
	ROM	DARAM	Внешняя
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов для выполнения инструкции под циклом RPT

Операнд	Программа		
	ROM	DARAM	Внешняя
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

Примечания

ROM Инструкция выполняется из внутрикристальной постоянной памяти,

DARAM Инструкция выполняется из внутренней памяти программ с двойным доступом за один машинный цикл.

Внешняя Инструкция выполняется из внешней памяти программ.

Если инструкция требует операнд из памяти, то строка в таблице указывает расположение операнда(ов) как показано ниже:

DARAM операнд находится во внутрикристальной памяти с двойным доступом за машинный цикл.

Внешняя операнд находится во внешней памяти.

Для циклического повторения инструкции под RPT, n – указывает число повторений данной инструкции инструкцией RPT. Дополнительные циклы (состояния ожидания) могут формироваться для доступа к памяти программ, памяти данных и пространству ввода - вывода генератором циклов ожидания (waitstate generator) или внешним сигналом READY. Эти циклы ожидания представлены следующими переменными:

p программные циклы ожидания. Представляет дополнительные циклы синхроимпульсов, которые ИС 1867ВЦ10Т ждет от внешней памяти программ при однократном доступе к ней.

d состояние ожидания памяти данных. Представляет число машинных циклов ожидания для ответа внешней памяти данных при однократном доступе к ней.

io состояние ожидания ввода-вывода. Представляет число машинных циклов ожидания для ответа устройств пространства ввода-вывода при однократном доступе.

n число повторений (где n > 2 для заполнения конвейера). Представляет число повторений выполнения инструкции.

Если имеется многократный доступ к одному и тому же пространству, то подходящий сомножитель предшествует переменной. Для примера, два доступа к внешней программной памяти требуют 2p циклов ожидания. Выше приведенные переменные могут также использовать подпись src, dst и code, чтобы указать источник, приемник или код, соответственно.

Все внешние чтения занимают, по крайней мере, один машинный цикл, в то время как все внешние записи занимают, по крайней мере, два машинных цикла. Однако, если внешняя запись немедленно следует или предшествует внешнему циклу чтения, тогда внешняя запись занимает три цикла. Если генератор ожиданий или вывод READY используется для добавления m (m > 0) циклов ожидания для внешнего доступа, тогда чтение требует m + 1 циклов, а внешняя запись требует m + 2 циклов.

Времена циклов инструкций основаны на следующих предположениях:

– По крайней мере четыре следующие инструкции выбираются из той же самой секции памяти (внутренней или внешней), которая была использована для выбора текущей инструкции (кроме случая, когда выполняется инструкция, нарушающая порядок увеличения PC на 1, например, такие как B, CALL и т.д.).

– В режиме однократного выполнения нет конфликтов в конвейере между текущей инструкцией и инструкцией немедленно предшествующей или следующей за текущей. Есть исключение только для конфликта между фазой выборки в конвейере и доступом чтения/записи в память (если имеется).

– В режиме повторения все конфликты, вызываемые конвейерным выполнением инструкции, обсуждаются.

Примеры

Для каждой инструкции включен пример кода, описано воздействие кода на память и регистры. Обсудим это на примере инструкции ADD:

ADD*+,0,AR0

Переменная	До выполнения инструкции	После выполнения инструкции
ARP	4	0
AR4	0302h	0303h
Data Memory		
302h	2h	2h
ACC	2h	04h
C	X	0

Имеются следующие факты и события, которые представлены в этом примере:

– Указатель вспомогательного регистра (ARP) указывает на текущий AR, потому что ARP = 4, следовательно, текущий регистр есть AR4.

– Когда выполняется сложение, то ЦП берет из AR4 адрес 0302h. Содержимое этого адреса, равное 2h, прибавляется к содержимому аккумулятора, которое

равно 2h. Результат, равный (4h), помещается в аккумулятор (потому, что второй операнд инструкции указывает левый сдвиг равный 0, значение памяти данных не сдвигается перед сложением со значением аккумулятора).

– Инструкция указывает инкремент на 1 содержимого текущего регистра (*+). Следовательно, после выполнения сложения, содержимое AR4 увеличивается до 0303h.

– Инструкция также указывает следующий вспомогательный регистр, следовательно, после выполнения инструкции ARP = 0.

– Поскольку не было переноса во время сложения, то бит переноса сбрасывается в 0.

14.8.3 Описание инструкций

В этом подразделе содержится детальная информация по набору инструкций для ИС 1867ВЦ10Т. Инструкции представлены в алфавитном порядке и описание для каждой инструкции представляет следующие категории информации:

– Синтаксис.
– Код операции.
– Операнды
– Выполнение.
– Статусные биты.
– Описание.
– Слова (длина инструкции).
– Циклы.
– Примеры.

Инструкция ABS

ABS – вычисление абсолютного значения аккумулятора.

СИНТАКСИС ABS

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
|(ACC)| -> ACC; 0 -> C
Зависит от OVM; изменяет OV и C. Не зависит от SXM.

ОПИСАНИЕ Если содержание аккумулятора больше или равно нулю, то после исполнения ABS оно не меняется. В противном случае аккумулятор загружается значением, которое является дополне-

нием до двух его исходного значения. После выполнения этой команды в бит переноса (C) всегда устанавливается ноль. Следует заметить, что значение 80000000h – это особый случай. Когда режим переполнения не установлен (OVM = 0), то ABS от 80000000h равен 80000000h. Когда режим переполнения установлен (OVM = 1), то ABS от 80000000h равен 7FFFFFFFh. В любом случае бит OV устанавливается в 1.

СЛОВА 1

ЦИКЛЫ

Циклы для однократного выполнения инструкции ABS

ROM	DARAM	Внешняя память
1	1	1+p

Циклы для многократного выполнения (RPT) инструкции ABS

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР 1

```
ABS
До выполнения                    После выполнения
ACC=1234h; C=X                     ACC=1234h; C=0
```

ПРИМЕР 2

```
ABS
До выполнения                    После выполнения
ACC=FFFFFFFFh; C=X               ACC=1h; C=0
```

ПРИМЕР 3

```
ABS ; (OVM = 1)
До выполнения                     После выполнения
ACC=80000000h; C=X               ACC=7FFFFFFFh; C=0
               OV=X                              OV=1
```

ПРИМЕР 4

```
ABS ; (OVM = 0)
До выполнения                    После выполнения
ACC=80000000h;  C=X              ACC=80000000h; C=0
                OV=X                              OV=1
```

Инструкция ADD

ADD – Сложение с аккумулятором со сдвигом.

СИНТАКСИС	ADD dma [, shift]	Прямая адресация
	ADD dma, 16	Прямая адресация с левым сдвигом на 16
	ADD ind [, shift [, ARn]]	Косвенная адресация
	ADD ind, 16 [, ARn]	Косвенная адресация с левым сдвигом на 16
	ADD #k	Короткая непосредственная адресация
	ADD #lk [, shift]	Длинная непосредственная адресация

ОПЕРАНДЫ

$0 \le dma \le 127$
$0 \le shift \le 15$ (по умолчанию 0)
$0 \le$ новый ARP ≤ 7
$0 \le k \le 255$
$-32768 \le lk \le 32767$

КОД КОМАНДЫ

Прямая адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	SHFT				0	dma						

Косвенная адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	SHFT				1	см. подраздел 14.7.3						

Прямая адресация со сдвигом 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1	0	dma						

Косвенная адресация со сдвигом 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1	1	см. подраздел 14.7.3						

Короткая непосредственная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	0	8-битная константа							

Длинная непосредственная адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	1	0	0	1	SHFT			
16-битная константа															

ВЫПОЛНЕНИЕ Прямая и косвенная адресации:
(PC) + 1 -> PC
(ACC) + [(dma)*$2^{сдвиг}$] -> ACC
Зависит от SXM и OVM; изменяет C и OV.
Короткая непосредственная адресация:
(PC) + 1 -> PC
(ACC) + k -> ACC
Зависит от OVM; изменяет на C и OV.
Длинная непосредственная адресация:
(PC) + 2 -> PC
(ACC) + lk*$2^{сдвиг}$ -> ACC
Зависит от OVM и SXM; изменяет на C и OV.

ОПИСАНИЕ Содержание адресуемой ячейки памяти данных или непосредственная константа сдвигается влево, в соответствии со значением сдвига, и прибавляется к аккумулятору. При выполнении сдвига младшие разряды прибавляемых данных заполняются нулями. В старшие разряды аккумулятора записывается значение знака, если SXM = 1, или они заполняются нулями, если SXM = 0. Результат запоминается в аккумуляторе. При модификации ARP, при косвенной адресации, необходимо указывать значение сдвига. Если сдвиг не используется, указывается 0: ADD *+,0,AR0. Короткий непосредственный 8-битный операнд добавляется к содержимому ACC как положительное значение. Результат операции сохраняется в ACC. В этом режиме не выполняется операция сдвига, и операция контроля знака через SXM бит. C бит устанавливается в 1, если результат операции генерирует перенос; иначе C бит устанавливается в 0. Если в инструкции задается сдвиг на 16, то C бит устанавливается только если результат добавления генерирует перенос; иначе бит C не модифицируется. Это позволяет корректно генерировать одиночный перенос при добавлении 32-битного числа к ACC.

СЛОВА 1 Прямая, косвенная или короткая непосредственная адресация.
2 Длинная непосредственная адресация.

ЦИКЛЫ

Циклов для однократного выполнения (прямая или косвенная адресация) инструкции ADD

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов для выполнения многократного выполнения(RPT) инструкции ADD

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

Циклов для однократного выполнения (короткая непосредственная адресация)

ROM	DARAM	Внешняя память
1	1	1+p

Циклов для однократного выполнения (длинная непосредственная адресация)

ROM	DARAM	Внешняя память
2	2	2+2p

ПРИМЕР 1

```
ADD DAT1,1  ;(DP = 6)
До выполнения                    После выполнения
Память данных                    Память данных
301h    1h                       301h    1h
ACC     2h;  C=X                 ACC     04h;  C=0
```

ПРИМЕР 2

```
ADD *+,0,AR0
До выполнения                    После выполнения
ARP     4                        ARP     0
AR4     0302h                    AR4     0303h
Память данных                    Память данных
302h    2h                       302h    2h
ACC     2h;  C=X                 ACC     04h;  C=0
```

ПРИМЕР 3

```
ADD #1h   ; короткий непосредственный операнд
До выполнения                    После выполнения
ACC     2h;  C=X                 ACC     03h;  C=0
```

ПРИМЕР 4

```
ADD #1111h,1   ; длинный непосредственный операнд
                                   со сдвигом 1
До выполнения                    После выполнения
ACC     2h; C=X                  ACC     2224h; C=0
```

Инструкция ADDC

ADDC – Сложение содержимого памяти данных и бита переноса с аккумулятором.

СИНТАКСИС
ADDC dma — Прямая адресация
ADDC ind [, ARn] — Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	0	0	dma						

Косвенная адресации

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	0	1	См. подраздел 14.7.3						

ОПЕРАНДЫ $0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(ACC) + dma + (C) -> ACC
Не зависит от OVM; влияет на C и OV.

ОПИСАНИЕ Содержимое адресуемой ячейки памяти данных и значение бита переноса прибавляется к аккумулятору. Результат сохраняется в аккумуляторе. Бит переноса устанавливается в 1, если результат генерирует перенос; иначе бит C устанавливается 0. Команда ADDC может быть использована для выполнения арифметических операций высокой точности.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнения инструкции под (RPT)

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
ADDC DAT0  ; (DP = 6)
До выполнения                    После выполнения
Память данных                    Память данных
300h    04h                      300h    04h
ACC     13h;  C=1                ACC     18h;  C=0
```

ПРИМЕР 2

```
ADDC *-,AR4 ; (OVM = 0)
До выполнения                    После выполнения
ARP     0                        ARP     4
AR0     300h                     AR0     299h
Память данных                    Память данных
300h    0h                       300h    0h
ACC     0FFFFFFFFh;  C=1         ACC     0h;    C=1
                     OV=X                       OV=0
```

Инструкция ADDS

ADDS – сложение с младшим словом аккумулятора без расширения знака.

СИНТАКСИС

ADDS dma — Прямая адресация
ADDS ind [, ARn] — Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	0	1	см. 14.7.3						

ОПЕРАНДЫ

$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ

(PC) + 1 -> PC
(ACC) + dma -> ACC
(dma) без знаковое 16-битное число
Зависит от OVM; изменяет на C и OV. Не зависит от SXM.

ОПИСАНИЕ

Значение адресуемой ячейки памяти данных прибавляется к аккумулятору с подавлением знакового расширения. Данные обрабатываются как 16-битные числа без знака, независимо от SXM. Содержание аккумулятора обрабатывается как число со знаком. Бит переноса устанавливается в 1, если результат генерирует перенос; иначе бит C устанавливается 0. При этом команда ADDS имеет такой же результат, что и команда ADD при SXM = 0 и сдвиге = 0.

СЛОВА 1

ЦИКЛЫ Циклов при однократном выполнении

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под(RPT)

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
ADDS DAT0  ; (DP = 6)
До выполнения                  После выполнения
Память данных                  Память данных
300h    F006h                  300h    F006h
ACC     3h;    C=X             ACC     F009h; C=0
```

ПРИМЕР 2

```
ADDS *
До выполнения                  После выполнения
ARP     0                      ARP     0
AR0     300h                   AR0     0300h
Память данных                  Память данных
300h    FFFFh                  300h   FFFFh
ACC     7FFF0000h;  C=X        ACC    7FFFFFFFh;C=0
```

Инструкция ADDT

ADDT – сложить с аккумулятором со сдвигом по значению TREG.

СИНТАКСИС

ADDT dma	Прямая адресация
ADDT ind [, ARn]	Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	1	1	см. 14.7.3						

ОПЕРАНДЫ $0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ (PC) + 1 -> PC

(ACC) + [(dma)*2(TREG(3-0))] -> (ACC)

ОПИСАНИЕ Значение памяти данных сдвигается влево и прибавляется к аккумулятору, результат помещается в аккумулятор. Сдвиг влево определяется четырьмя младшими битами TREG, от 0 до 15 бит. Знаковое расширение значения памяти данных управляется SXM. Бит переноса устанавливается в 1, если результат генерирует перенос; иначе бит C устанавливается 0.

СЛОВА 1

ЦИКЛЫ

Циклов для однократного выполнения

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под(RPT)

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
ADDT DAT127  ; (DP = 4, SXM = 0)
До выполнения                    После выполнения
027h    09h                      027h    09h
TREG1   0FF94h                   TREG1   0FF94h
ACC     0F715h; C=X              ACC     0F7A5h;C=0
```

ПРИМЕР 2

```
ADDT *-,AR4 ; (SXM = 0)
До выполнения                    После выполнения
ARP     0                        ARP    0
AR0     027Fh                    AR0    027Eh
Память данных                    Память данных
027Fh   09h                      027F   09h
TREG1   0FF94h                   TREG1  0FF94h
ACC     0F715h;  C=X             ACC    0F7A5h; C=0
```

Инструкция ADRK

ADRK – сложение вспомогательного регистра и короткого непосредственного операнда.

СИНТАКСИС ADRK #k Короткая непосредственная адресация

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	0	0	8-битная константа							

ОПЕРАНДЫ $0 \le k \le 255$

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
Текущий AR + 8-битная положительная константа -> текущего AR

ОПИСАНИЕ 8-битная непосредственная константа прибавляется к текущему дополнительному регистру (на который указывает ARP), результат сохраняется в текущий дополнительный регистр. В ARAU складываются 8-битные положительные целые значения. Следует отметить, что все арифметические операции незнаковые.

СЛОВА 1

ЦИКЛЫ

Циклов для однократного выполнения

ROM	DARAM	Внешняя память
1	1	1+p

ПРИМЕР 1

```
ADRK #80h
До выполнения                После выполнения
ARP     5                    ARP     5
AR5     4321h                AR5     43A1h
```

Инструкция AND

AND – логическое «И» с аккумулятором.

СИНТАКСИС

AND dma	Прямая адресация
AND ind [, ARn]	Косвенная адресация
AND #lk [, shift]	Длинная непосредственная адресация
AND #lk, 16	Длинная непосредственная адресация с левым сдвигом на 16

ОПЕРАНДЫ $0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7
lk: 16-битная константа
$0 \le$ сдвиг ≤ 16

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	0	1	См. подраздел 14.7.3						

Длинная непосредственная адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	1	0	1	1	SHFT			
16-битная константа															

Длинная непосредственная адресация со сдвигом 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	0	1	0	0	0	0	0	0	1
16-битная константа															

ВЫПОЛНЕНИЕ

Прямая и косвенная адресация:
(PC) + 1 -> PC
(ACC(15-0)) AND (dma) -> ACC(15-0)
0 -> ACC(31-16)

Непосредственная адресация:
(PC) + 2 -> PC
ACC(30-0) AND lk*2есдвиг -> ACC
Не зависит от SXM.

ОПИСАНИЕ

Если используется прямая или косвенная адресация над младшим словом аккумулятора и значением памяти данных, производится логическая операция AND, результат сохраняется в позиции младшего слова аккумулятора. Старшее слово аккумулятора обнуляется. Результат является логическим «И» значения операнда и содержимым аккумулятора.

СЛОВА

1 (Прямая или непосредственная адресация)
2 (Длинная непосредственная адресация)

ЦИКЛЫ

Циклов при однократном выполнении инструкции (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

Циклов при однократном выполнении инструкции

(непосредственная адресация)

ROM	DARAM	Внешняя память
2	2	2+2p

ПРИМЕР 1
```
AND DAT16 ;(DP = 4)
До выполнения                    После выполнения
Память данных                    Память данных
0210h   00FFh                    0210h    00FFh
ACC     12345678h                ACC      00000078h
```

ПРИМЕР 2
```
ADD *
До выполнения                    После выполнения
ARP     0                        ARP     0
AR0     0301h                    AR0     0301h
Память данных                    Память данных
301h    0FF0h                    0301h    0FF00h
ACC     12345678h                ACC     00005600h
```

ПРИМЕР 3
```
AND #00FFh,4
До выполнения                    После выполнения
ACC     12345678h                ACC      00000670h
```

Инструкция APAC

APAC – прибавляет содержимое P регистра к аккумулятору.

СИНТАКСИС APAC

ОПЕРАНДЫ Нет

КОД КОМАНДЫ

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

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(ACC) + (P-регистр со сдвигом) -> ACC
Зависит от OVM и PM; влияет на C и OV.
Не зависит от SXM.

ОПИСАНИЕ Содержание P регистра сдвигается как определено статусными битами PM и прибавляется к содержанию аккумулятора. Результат сохраняется в аккумуляторе. APAC не зависит от бита SXM; P регистр всегда знаковый. Бит переноса устанавливается в 1, если результат генерирует перенос; иначе бит C устанавливается 0.
Команда APAC является подфункцией LTA, LTD, MAC, MACD, MADS, MADD, MPYA и SQRA инструкций.

Режимы сдвига произведения

Биты PM		Результат
Бит 1	Бит 0	
0	0	Нет сдвига
0	1	Левый сдвиг на 1 бит
1	0	Левый сдвиг на 4 бита
1	1	Правый сдвиг на 6 бит

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под (RPT)

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР

```
APAC  ; (PM = 01)
До выполнения                    После выполнения
P       40h                      P       40h
ACC     20h;     C=X             ACC     A0h;     C=0
```

Инструкция B

B – безусловный переход.

СИТАКСИС B pma [, ind [, ARn]_] Косвенная адресация

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	D	0	1	1	См. 14.7.3						
16-битная константа															

ОПЕРАНДЫ $0 \le pma \le 65536$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ pma -> PC
Модифицирует текущий AR и ARP как задано.

ОПИСАНИЕ Текущий дополнительный регистр и ARP модифицируются заданным образом и управление передается по заданному адресу pma – или числовой или символический адрес.

СЛОВА 2

ЦИКЛЫ

Циклов при однократном выполнении инструкции		
ROM	DARAM	Внешняя память
4	4	4+4p

Примечание – Когда эта инструкция достигает фазы выполнения в конвейере, то два дополнительных слова вводятся в конвейер. Когда PC непоследовательно выбирается, то эти два слова инструкции выбрасываются.

ПРИМЕР
```
B               191, *+, AR1
```

В PC загружается 191 и выполнение программы продолжается с загруженного адреса. Текущий дополнительный регистр инкрементируется на 1 и ARP устанавливается в 1.

Инструкция BACC

BACC - переход по адресу, задаваемому аккумулятором.

СИНТАКСИС BACC

ОПЕРАНДЫ нет

КОД КОМАН- 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

ДЫ

1	0	1	1	1	1	1	0	0	0	1	0	0	0	0	0

ВЫПОЛНЕНИЕ ACC(15-0) -> PC

ОПИСАНИЕ Управление передается по 16-битному адресу, взятому из младшего слова аккумулятора.

СЛОВА 2

ЦИКЛЫ

Циклов при однократном выполнении инструкции		
ROM	DARAM	Внешняя память
4	4	4+3p

ПРИМЕР
```
BACC    ; (ACC содержит 191)
```

В PC загружается 191 и выполнение программы загруженного адреса.

Инструкция BANZ

BANZ - переход, если вспомогательный регистр ≠ 0.

СИНТАКСИС BANZ pma [, ind [, ARn]_] Косвенная адресация

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	D	1	1	1	см. подраздел 14.7.3						
16-битная константа															

ОПЕРАНДЫ $0 \le pma \le 65536$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
```
если ( текущий AR ≠ 0 )
     pma -> PC;
иначе
     (PC) + 2 -> PC;
```
Модифицирует текущий AR и ARP как задано

ОПИСАНИЕ Управление передается по адресу pma, если содержимое текущего AR ≠ 0. В противном случае управление передается к следующей инструкции. По умолчанию значение текущего AR уменьшается на 1. Для выполнения N операций цикла счетчик цикла перед входом в цикл устанавливают в (N-1).

СЛОВА 2

ЦИКЛЫ Циклов при однократном выполнении инструкции

Условие	ROM	DARAM	Внешняя память
Выполнено	4	4	4+4p
Не выполнено	2	2	2+2p

ПРИМЕР 1

```
BANZ PGM0
```

До выполнения		После выполнения	
ARP	0	ARP	0
AR0	5h	AR0	4h

В PC загружается 0 и выполнение программы продолжается с адреса 0.

ИЛИ

До выполнения		После выполнения	
ARP	0	ARP	0
AR0	0h	AR0	0ffffh

Значение PC увеличивается на два и выполнение программы продолжается с адреса, указанного в PC.

Так как содержимое AR0 ≠ 0, то программный адрес 0 загружается в программный счетчик (PC) и программа продолжает выполнение с этого адреса. Операция над вспомогательным регистром по умолчанию – это уменьшение текущего AR.

ПРИМЕР 2

```
        MAR    *, AR0
        LAR    AR1, #3
        LAR    AR0, #60H
PGM191  ADD    *+, AR1
        BANZ   PGM191, AR0
```

Содержимое памяти данных 60h – 63h прибавляется к аккумулятору.

Инструкция BCND

BCND - переход по условию.

СИНТАКСИС BCND pma, cond_1 [,cond_2] [,...]

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	0	TP		ZLVC				ZLVC			
16-битная константа															

ОПЕРАНДЫ $0 \le pma \le 65536$

Условия (cond):

ACC=0	EQ
ACC≠0	NEQ
ACC<0	LT
ACC<=0	LEQ
ACC>0	GT

ACC>=0	GEQ
C=0	NC
C=1	C
OV=0	NOV
OV=1	OV
BIO младшие	BIO
TC=0	NTC
TC=1	TC
Без условия	UNC

ВЫПОЛНЕНИЕ
If(условие(я))
Then pma -> PC
Else PC + 2 -> PC

ОПИСАНИЕ Управление передается по адресу pma, если условие выполняется (истинно). Следует отметить, что не все комбинации условий допускаются, а так же, что тестирование BIO сочетается исключительно с тестированием TC.

СЛОВА 2

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Условие	ROM	DARAM	Внешняя память
Выполняется	4	4	4+4p
Не выполняется	2	2	2+2p

ПРИМЕР
```
BCND  PG191, LEQ, C
```

Если содержание аккумулятора меньше или равно нулю и бит переноса установлен в 1, то программный адрес 191 загружается в PC и программа продолжает выполняться с этого адреса. Если условие не выполняется, то выполнение продолжается с адреса PC + 2.

Инструкция BIT

BIT - Тестирование бита.

СИНТАКСИС
BIT dma, bit code Прямая адресация
BIT ind, bit code [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	BITX				0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	BITX				1	см. подраздел 14.7.3						

ОПЕРАНДЫ $0 \leq pma \leq 65536$
$0 \leq$ новый ARP ≤ 7
$0 \leq$ битый код ≤ 15

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(dma-бит(15-битый код)) -> TC

Влияет на TC.

ОПИСАНИЕ Команда BIT копирует значения бита слова памяти данных в TC бит статусного регистра ST1.Следует заметить, что BIT, CMPR, LST1, APL, CPL, OPL, XPL и NORM команды тоже изменяют TC бит статусного регистра ST1. Значение битного кода, которое соответствует битной позиции, представлено в таблице.

Битный адрес	Битный код
(LSB) 0	1111
1	1110
2	1101
3	1100
4	1001
5	1010
6	1001
7	1000
8	0111
9	0110
10	0101
11	0100
12	0011
13	0010
14	0001
(MSB) 15	0000

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
BIT  0h,15; (DP = 6). Тестируется младший бит
ячейки 300h
        До выполнения  После выполнения
Память данных  Память данных
300h    4DC8h  300h    04DC8h
TC       0      TC      0
```

ПРИМЕР 2

```
BIT *,0,AR1 . Тестируется старший бит ячейки 310h
        До выполнения  После выполнения
ARP     0      ARP     1
AR0     310h   AR0     310h
Память данных  Память данных
310h    8000h  310h    8000h
TC      0      TC      1
```

Инструкция BITT

BITT - Тестирование бита по значению TREG.

СИНТАКСИС

BITT dma Прямая адресация
BITT ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	1	1	См. раздел 14.7.3						

ОПЕРАНДЫ $0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(dma-бит(15-TREG2(3-0))) -> TC

Влияет на TC.

ОПИСАНИЕ Команда BITT копирует значение бита слова памяти данных в TC бит статусного регистра ST1. Команды BIT, CMPR, LST1, APL, CPL, OPL, XPL и NORM тоже изменяют TC бит статусного регистра ST1. Битный адрес, определенный значением битного кода, содержится в четырех младших битах TREG, и представлен ниже в таблице.

Битный адрес	Битный код
(LSB) 0	1111
1	1110
2	1101
3	1100
4	1001
5	1010
6	1001
7	1000
8	0111
9	0110
10	0101
11	0100
12	0011
13	0010
14	0001
(MSB) 15	0000

СЛОВА 1

ЦИКЛЫ Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	N	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1
```
BITT  00h; (DP = 6). Тестируется 14 бит ячейки
данных 300h
        До выполнения  После выполнения
Память данных  Память данных
```

```
300h    4DC8h  300h   04DC8h
TREG    1h     TREG    1h
TC       0      TC      1
```

ПРИМЕР 2

```
BITT  *  ; (DP = 6). Тестируется 1 бит ячейки
данных 310h
        До выполнения  После выполнения
ARP     1      ARP     1
AR1     310h   AR1     310h
Память данных  Память данных
310h    8000h  310h   8000h
TREG    0Eh    TREG   0Eh
TC      0      TC     0
```

Инструкция BLDD

BLDD - Перемещение блока данных из памяти данных в память данных.

СИНТАКСИС

BLDD #lk, dma Прямая адресация с длинным непосредственным источником

BLDD #lk, ind [, ARn] Косвенная адресация с длинным непосредственным источником

BLDD dma, #lk Прямая адресация с длинным непосредственным приемником

BLDD ind, #lk [, ARn] Косвенная адресация с длинным непосредственным приемником

КОД КОМАНДЫ

Прямая адресация с SRC, определяемого длинной константой

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	0	0	0	0	dma						
16-битная константа															

Косвенная адресация с SRC, определяемого длинной константой

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	0	0	1	1	см. подраздел 14.7.3						
16-битная константа															

Прямая адресация с DST, определяемой длинной константой

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	0	0	1	0	dma						
16-битная константа															

Косвенная адресация с DST, определяемой длинной константой

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	0	0	1	1	см. подраздел 14.7.3						
16-битная константа															

ОПЕРАНДЫ

$0 \leq lk \leq 65535$

$0 \leq dma \leq 127$

$0 \leq$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ

Инкремент PC, после …

(PC) →MSTACK

lk →PC

(Источник) →Приемник

Для косвенной адресации (текущий AR) и (ARP) как указано

(PC) + 1 →PC

Пока (счетчик повторений) ≠0:

(Источник) →Приемник

Для косвенной адресации (текущий AR) и (ARP) как указано

(PC) + 1 →PC

(MSTACK) →PC

ОПИСАНИЕ

Слово из памяти данных, на которое указывает scr, копируется в слова памяти данных, на которое указывает dst. Слово источника и/или приемника может быть указано длинным непосредственным операндом или адресом памяти данных. Цикл RPT может быть использован с командой BLDD в режиме косвенной адресации для перемещения блоков данных. Число слов, которые будут перемещены, на единицу больше, чем число, содержащееся в счетчике повторений RPTC перед выполнением команды. Если при выполнении инструкции в цикле RPT данные адресуются длинной непосредственной константой, то адрес данных автоматически инкрементируется. При выполнении инструкции в цикле RPT dma адрес автоматически не инкрементируется. При этом блоки источника и приемника не должны быть полностью во внутренней или внешней памяти. Прерывания запрещены в течение исполнения операции BLDD в цикле RPT. Длинный непосредственный операнд не применим для адресации внутрикристальных регистров, картированных в памяти. Для адресации внутрикристальных регистров, картированных в памяти, используются прямая и косвенная адресации.

СЛОВА

2 (источник или приемник задается длинным непосредственным операндом)

ЦИКЛЫ

Циклов при однократном выполнении инструкции (SRC или DST длинная константа)

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM Приемник: DARAM	3	3	3+2p
Источник: Внешняя Приемник: DARAM	3+dsrc	3+dsrc	3+dsrc+2p
Источник: DARAM Приемник: Внешняя	4+ddst	4+ddst	6+ddst+2p
Источник: Внешняя Приемник: Внешняя	n+2	n+2	n+2+2p

Циклов при многократном выполнении инструкции под RPT (SRC или DST длинная константа)

Операнд	ROM	DARAM	Внешняя память
Источник: Внешняя Приемник: DARAM	n+2+ndsrc	n+2+ndsrc	n+2+ndsrc
Источник: DARAM Приемник: Внешняя	2n+2+nddst	2n+2+nddst	2n+2+nddst+2p

Источник: Внешняя Приемник: Внешняя	4n+ndsrc+nddst	4n+ndsrc+nddst 4n+2+ndsrc+nddst+2p

ПРИМЕР 1

```
BLDD #300h,20h  ; (DP = 6)
         До выполнения  После выполнения
память данных  память данных
300h     0h    300h     0h
320h     0Fh   320h     0h
```

ПРИМЕР 2

```
BLDD  *+,#321h,AR3
        До выполнения  После выполнения
ARP     2       ARP     3
AR2     301h    AR2     302h
память данных  память данных
301h    01h     301h    01h
321h    0Fh     321h    01h
```

ПРИМЕР 3

```
RPT 2
BLDD  #300h,*+
        До выполнения  После выполнения
ARP     0       ARP     0
AR0     320h    AR0     323h
память данных  память данных
300h    7F98h   300h    7F98h
301h    0FFE6h  301h    0FFE6h
302h    9522h   302h    9522h
320h    8DEEh   320h    7F98h
321h    9315h   321h    0FFE6h
322h    2531h   322h    9522h
```

Инструкция BLPD

BLPD - Перемещение блока данных из памяти данных в память программ.

СИНТАКСИС

BLPD #pma, dma Прямая адресация с длинным непосредственным источником

BLPD #pma, ind [, ARn] Косвенная адресация с длинным непосредственным источником

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	1	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ $0 \leq dma \leq 127$
$0 \leq$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ Инскремент PC, тогда
(PC) →MSTACK
pma → PC

ОПИСАНИЕ

Слово из памяти программ, на которое указывает scr, копируется в память данных, на которое указывает dst. Первое слово источника указывается длинным непосредственным операндом. Адрес получателя в памяти данных может быть указан непосредственно или через дополнительный регистр, при косвенной адресации.
Режим повтора может быть использован с командой BLPD для перемещения блоков данных. Число слов, которые будут перемещены, на единицу больше, чем число, содержащееся в счетчике повторений RPTC перед выполнением команды.
При выполнении инструкции в цикле RPT адрес источника памяти программ, длинный непосредственный операнд, заносится в PC. Так как PC инкрементируется на 1 при каждом повторении, возможен доступ к серии адресов памяти программ. При использовании косвенной адресации получателя (память данных), новый адрес памяти данных вычисляется при каждом повторении. При использовании прямой адресации указанный адрес памяти данных постоянен – не меняется при каждом повторении.
Блоки источника и приемника не должны быть полностью во внутренней или внешней памяти. Прерывания запрещены при выполнении операции BLPD в режиме повтора. В режиме повтора BLPD выполняется за один цикл.

СЛОВА

1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM Приемник: DARAM	2	2	2+p
Источник: Внешняя Приемник: DARAM	2+dsrc	2+dsrc	3+dsrc+pcode

Источник: DARAM Приемник: Внешняя	3+pdst	3+pdst	4+pdst+pcode
Источник: Внешняя Приемник: Внешняя	3+dsrc+pdst	3+dsrc+pdst	5+dsrc+pdst+pcode

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM Приемник: DARAM	n+1	n+1	n+1+pcode
Источник: Внешняя Приемник: DARAM	n+1+ndsrc	n+1+ndsrc	n+2+ndsrc+pcode
Источник: DARAM Приемник: Внешняя	2n+1+npdst	2n+1+npdst	2n+2+npdst+pcode
Источник: Внешняя Приемник: Внешняя	4n-1+ndsrc+npdst	4n-1+ndsrc+npdst	4n+1+ndsrc+npdst+pcode

ПРИМЕР 1

```
BLDP 00h      ; (DP = 6)
         До выполнения  После выполнения
Программная память              Программная память
800h     0Fh   800h   0Fh
память данных  память данных
300h      0h3   00h    0Fh
```

ПРИМЕР 2

```
BLDP *,AR7
        До выполнения  После выполнения
ARP     0      ARP     7
AR0     310h   AR0     310h
Программная память              Программная память
800h    1111h  800h   1111h
память данных  память данных
310h     0100h 310h    1111h
```

Инструкция CALA

CALA – Вызов подпрограммы по адресу задаваемым аккумулятором.

СИНТАКСИС CALA

КОД КОМАНДЫ

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

ОПЕРАНДЫ Нет

ВЫПОЛНЕНИЕ Без задержки: PC + 1 -> TOS
С задержкой: PC + 3 -> TOS
ACC(15-0) -> PC

ОПИСАНИЕ Текущее значение программного счетчика (PC) инкрементируется и заносится на верхушку стека (TOS). Потом содержание младшей половины аккумулятора загружается в PC. Исполнение программы продолжается с этого адреса.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
4	4	4+3p

ПРИМЕР

```
CALA
        До выполнения  После выполнения
PC      25h     PC      83h
ACC     83h     ACC     83h
TOS      100h    TOS      26h
```

Инструкция CALL

CALL - Безусловный вызов подпрограммы.

СИНТАКСИС CALL pma [, ind [, ARn]_] Косвенная адресация

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	1	0	1	см. подраздел 14.7.3						
16-битная константа															

ОПЕРАНДЫ $0 \leq pma \leq 65535$
$0 \leq$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ Без задержки: PC + 2 -> TOS
С задержкой: PC + 4 -> TOS
pma -> PC
Изменение текущего AR и ARP как определено.

ОПИСАНИЕ Текущее значение программного счетчика (PC) инкрементируется и заносится в верхушку стека (TOS). Потом значение, адресуемое программной памятью (pma), загружается в PC. Текущий вспомогательный регистр и ARP изменяются как определено.

СЛОВА 2

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
4	4	4+4p

ПРИМЕР

```
CALL  PRGG191,*+,AR0
          До выполнения  После выполнения
ARP       1       ARP      0
AR1       05h     AR1      06h
PC        30h     PC       0BFh
TOS       100h    TOS      32h
```

0BFh загружается в программный счетчик и исполнение программы продолжается с этого адреса.

Инструкция CC

CC - переход по условию.

СИНТАКСИС CC pma, cond_1 [,cond_2] [,...]

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	TP		ZLVC				ZLVC			
16-битная константа															

ОПЕРАНДЫ $0 \leq pma \leq 65535$

cond:

ACC = 0	EQ
ACC!=0	NEQ
ACC<0	LT
ACC<=0	LEQ
ACC>0	GT
ACC>=0	GEQ
C=0	NC
C=1	C
OV=0	NOV
OV=1	OV
BIO младшие	BIO
Без условия	UNC

ВЫПОЛНЕНИЕ

```
Если (условия(е))
    Без задержки:  PC + 2 -> TOS
    С задержкой:   PC + 4 -> TOS
    pma -> PC
Иначе
    PC + 2 -> PC
```

ОПИСАНИЕ Если заданное условие выполняется, управление передается по адресу pma. Текущее значение программного счетчика инкрементируется и заносится в верхушку стека (TOS). Следует отметить, что не все комбинации условий допустимы. К тому же, условия NTC, TC и BIO взаимно исключаются. Команда CC аналогична команде CALL, когда все условия истинны.

СЛОВА 2

ЦИКЛЫ

Циклов при однократном выполнении инструкции			
Условие	ROM	DARAM	Внешняя память
Выполнено	4	4	4+4p
Не выполнено	2	2	2+2p

ПРИМЕР

```
CC     PRGG191,LEQ,C
```

Если содержимое аккумулятора меньше или равно нулю и установлен в 1 бит переноса, то 0BFh (191) загружается в программный счетчик и исполнение программы продолжается с этого адреса. Если условия не выполнились, то выполнение продолжается с команды, следующей за командой CC.

Инструкция CLRC

CLRC – очистка управляющего бита.

СИНТАКСИС CLRC control bit

КОД КОМАНДЫ

CLRC C (Очистка C)

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

CLRC CNF (Очистка управления конфигурацией)

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

CLRC INTM (Очистка режима прерывания)

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

CLRC OVM (Очистка режима переполнения)

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

CLRC SXM (Очистка режима расширения знака)

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

CLRC TC (Очистка TC)

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

CLRC XF (Очистка вывода XF)

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

ОПЕРАНДЫ управляющий бит: ST0, ST1 биты (из следующего набора):
{C, CNF, INTM, OVM, TC, SXM, XF}

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
0 -> управляющий бит

ОПИСАНИЕ Указанный управляющий бит обнуляется. Команда LST может быть также использована для загрузки ST0 и ST1.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР

```
CLRC  TC  ; TC - 11 бит ST1
          До выполнения  После выполнения
ST1       x9xxh   ST1      x1xxh
```

Инструкция CMPL

CMPL – инверсия аккумулятора.

СИНТАКСИС CMPL

КОД КОМАНДЫ

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

ОПЕРАНДЫ Нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(~ACC) -> ACC

ОПИСАНИЕ Содержимое аккумулятора переписывается с логической инверсией (дополнение до единицы). Бит переноса не влияет.

СЛОВА 1

ЦИКЛЫ Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР

```
CMPL
          До выполнения              После выполнения
ACC       0F7982513h; C=X     ACC    0867DAECh;
C=X
```

Инструкция CMPR

CMPR – сравнение вспомгательного регистра с AR0.

СИНТАКСИС CMPR CM

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	0	1	0	0	0	1	CM	

ОПЕРАНДЫ $0 \leq CM \leq 3$

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
Результат сравнения текущего AR с AR0 заносится в TC бит статусного регистра ST1.

Влияет на TC; зависит от NDX.
Не зависит от SXM; не влияет на SXM.

ОПИСАНИЕ Команда CMPR выполняет сравнения, заданные значением CM:
При CM = 00 проверка (текущий AR) = (AR0)
При CM = 01 проверка (текущий AR) < (AR0)
При CM = 10 проверка (текущий AR) > (AR0)
При CM = 11 проверка (текущий AR) ≠ (AR0)
Если условие истинно, то в TC бит загружается 1. Если условие ложно, то в TC бит загружается 0.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР

```
CMPR 2
        До выполнения  После выполнения
APR     4       APR     4
ARCR    0FFFFh ARCR     0FFFFh
AR4     7FFFh  AR4      7FFFh
TC       1       TC       0
```

Инструкция DMOV

DMOV – перемещения данных в памяти данных.

СИНТАКСИС
DMOV dma Прямая адресация
DMOV ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	1	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \leq dma \leq 127$
$0 \leq$ новый $ARP \leq 7$

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(dma) -> dma + 1

Зависит от CNF и OVLY.

ОПИСАНИЕ
Содержимое заданного адреса (dma) памяти данных копируется в следующий (dma+1) адрес памяти данных.
DMOV выполнятся только для блоков внутрикристальной памяти, конфигурируемой как данные. Как дополнение, инструкция продолжит перемещение данных за границы адресного пространства блоков внутрикристальной памяти B0 и B1. Инструкция DMOV не может быть использована для внешней памяти или регистров, картированных в памяти данных. Если она будет использована для внешней памяти или регистров, картированных в памяти, то эффект выполнения инструкции не будет достигнут. Когда данные копируются из адресуемой ячейки в следующую ячейку, то содержание адресуемой ячейки остается неизменным.
Инструкция перемещения данных полезна для выполнения z^{-1} задержек, которые встречаются в цифровой обработке сигналов. Инструкция DMOV является подфункцией команд TD, MACD, MADD (см. эти инструкции).

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	2+2d	2+2d	5+2d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	4n-2+2nd	4n-2+2nd	4n+1+2nd+p

ПРИМЕР 1

```
DMOV  DAT8  ; (DP = 6)
        До выполнения  После выполнения
Память данных  Память данных
308h    43h    308h    43h
Память данных  Память данных
309h     2h     309h    43h
```

ПРИМЕР 2

```
DMOV   *,AR1
        До выполнения  После выполнения
ARP     0      ARP     1
AR1     30Ah   AR1     30Ah
Память данных  Память данных
30Ah    40h    30Ah    40h
Память данных  Память данных
30Bh    41h    30Bh    40h
```

Инструкция IDLE

IDLE – ожидание прерывания.

СИНТАКСИС IDLE

КОД КОМАНДЫ

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

ОПЕРАНДЫ Нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC

Зависит от INTM.

ОПИСАНИЕ Инструкция IDLE обеспечивает режим ожидания выполнения последовательности инструкций до появления немаскируемого прерывания (внешнего или внутреннего); PC увеличивается на 1 и процессор останавливается в состоянии ожидания до появления прерывания. Процессор выводится из состояния ожидания немаскируемым прерыванием, даже если INTM = 1. Если INTM = 1, то программа продолжит выполнение с команды, следующей за IDLE. Если INTM = 0, то вызывается программа обработки прерывания. Исполнение IDLE заставляет F240 входить в режим низкого потребления энергии. В течение режима ожидания IDLE таймер и последовательный порт активны. Поэтому прерывания таймера и последовательного порта выводят процессор из этого режима.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции		
ROM	DARAM	Внешняя память
1	1	1+p

ПРИМЕР 1

```
IDLE  ;Процессор ожидает сброс или немаскируемое
      ;прерывание.
```

Инструкция IN

IN – чтение из порта ввода-вывода.

СИНТАКСИС
IN dma, PA Прямая адресация
IN ind, PA [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	1	1	1	0	dma						
16-битная константа															

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	1	1	1	1	см. подраздел 14.7.3						
16-битная константа															

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7
$0 \le PA \le 65535$

ВЫПОЛНЕНИЕ
(PC) + 2 -> PC
Адрес порта -> адресная шины A15-A0
Шина данных D15-D0 -> dma

ОПИСАНИЕ Команда IN читает 16-битное значение из внешнего порта ввода-вывода в заданную ячейку памяти данных. Сигнал IS# устанавливается в 0, что обеспечивает выбор пространства ввода-вывода. Сигналы STRB#, WR/RD# и READY устанавливаются так же, как при чтении внешней памяти данных. Инструкция RPT может быть использована с инструкцией RPT для чтения последовательных слов из пространства ввода-вывода.

СЛОВА 2

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
Назначение: DARAM	2+iosrc	2+iosrc	3+iosrc+2pcode
Назначение: Внешняя	3+ddst+iosrc	3+ddst+iosrc	6+ddst+iosrc +2pcode

Операнд	ROM	DARAM	Внешняя память
Назначение: DARAM	2n+niosrc	2n+niosrc	2n+1+niosrc+2pcode
Назначение: Внешняя	4n-1+nddst +niosrc	4n-1+nddst +niosrc	4n+2+nddst+niosrc +2pcode

ПРИМЕР 1

```
IN DAT7, PA5 ;Чтение слова из адреса 5 порта
             ;ввода - вывода.Значение запоминается
              ;в 307h ячейки памяти(DP=6).
```

ПРИМЕР 2

```
IN  *,PA0 ;Чтение слова из адреса 0 порта
          ;ввода - вывода. Значение запоминается в
          ;ячейке памяти указанной текущим
          ;вспомогательном регистром.
```

Инструкция INTR

INTR – программное прерывание.

СИНТАКСИС INTR K

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	0	0	1	1	INTR#K				

ОПЕРАНДЫ $0 \le K \le 31$

ВЫПОЛНЕНИЕ (PC) + 1 -> стек
соответствующий вектор прерывания -> PC

Не зависит от INTM.

ОПИСАНИЕ Инструкция INTR – программное прерывание, которое передает программное управление по адресу вектора прерывания программной памяти, заданного K (смотри следующую таблицу). Эта команда позволяет исполнять любую сервисную программу обработки прерывания из программного обеспечения пользователя. Во время исполнения команды содержание PC + 1 заносится в стек.При этом программные прерывания INTR не маскируются. INTR для внешних прерываний(INT1-INT4) и выглядит точно так же, как внешнее прерывание (генерируется сигнал подтверждения прерывания, маскируемые прерывания глобально запрещаются INTM=1).

K	Прерывания	Ячейки	K	Прерывания	Ячейки
0	$\overline{RS}$	0h	16	Резервное	20h
1	$\overline{INT1}$	2h	17	TRAP	22h
2	$\overline{INT2}$	4h	18	$\overline{INT}$	24h
3	$\overline{INT3}$	6h	19	Резервное	26h
4	TINT	8h	20	Пользовательское	28h
5	RINT	Ah	21	Пользовательское	2Ah
6	XINT	Ch	22	Пользовательское	2Ch
7	TRNT	Eh	23	Пользовательское	2Eh
8	TXNT	10h	24	Пользовательское	30h
9	$\overline{INT4}$	12h	25	Пользовательское	32h
10	Резервное	14h	26	Пользовательское	34h
11	Резервное	16h	27	Пользовательское	36h
12	Резервное	18h	28	Пользовательское	38h
13	Резервное	1Ah	29	Пользовательское	3Ah
14	Резервное	1Ch	30	Пользовательское	3Ch
15	Резервное	1Eh	31	Пользовательское	3Eh

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции		
ROM	DARAM	Внешняя память
4	4	4+3p

ПРИМЕР 1

```
INTR    3  ;Управление передается ячейке 6h
           ;программной памяти, PC + 1 зано-
сится
           ;в стек.
```

Инструкция LACC

LACC – загрузка ACC со сдвигом.

СИНТАКСИС

LACC dma [, shift] Прямая адресация
LACC dma, 16 Прямая адресация с левым сдвигом на 16
LACC ind [, shift [, ARn]_] Косвенная адресация
LACC ind, 16[, ARn] Косвенная адресация с левым сдвигом на 16
LACC #lk [, shift] Длинная непосредственная адресация

КОД КОМАНДЫ

Прямая адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	SHFT				0	dma						

Косвенная адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	SHFT				1	см. подраздел 14.7.3						

Прямая адресация со сдвигом на 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	1	0	0	dma						

Косвенная адресация со сдвигом на 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	1	1	1	см. подраздел 14.7.3						

Длинная непосредственная адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	1	0	0	0	SHFT			
16-битная константа															

ОПЕРАНДЫ

$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7
$0 \le$ сдвиг ≤ 15 (по умолчанию 0)
$-32768 \le lk \le 32768$

ВЫПОЛНЕНИЕ Прямая адресация:
(PC) + 1 -> PC
(dma) * 2 ^{SHFT} ->ACC

Прямая или косвенная адресации:
(PC) + 1 -> PC
(dma) * 2 ^{16} ->ACC

Длинная непосредственная адресация:
(PC) + 2 -> PC
lk * 2 ^{SHFT} ->ACC

Зависит от SXM

ОПИСАНИЕ Содержимое указанного адреса памяти данных или 16-битная константа сдвигается влево и загружается в аккумулятор. При сдвиге младшие биты заполняются 0. В старших битах происходит расширение знака, если SXM = 1, иначе они заполняются 0.

СЛОВА 1 (Прямая или косвенная адресация)
2 (Длинная непосредственная адресация)

ЦИКЛЫ

Циклов при однократном выполнении инструкции (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

Циклов при однократном выполнении инструкции (длинная непосредственная адресация)

ROM	DARAM	Внешняя память
2	2	2+2p

ПРИМЕР 1

```
LACC DAT6, 4 ; (DP = 8,  SXM = 0)

        До выполнения           После выполнения
Память данных           Память данных
406h    01h             406h    01h
ACC     012345678h C=X  ACC     10h C=X
```

ПРИМЕР 2

```
LACC *, 4 ; (SXM = 0)
        До выполнения           После выполнения
ARP     2               ARP     2
AR2     0300h           AR2     0300h
Память данных           Память данных
300h    0ffh            300h    0ffh
ACC     012345678h;C=X  ACC     0ffh; C=X
```

ПРИМЕР 3

```
LACC    #f000h, 1 ; (SXM = 1)
        До выполнения           После выполнения
ACC     012345678h;C=X  ACC     0ffffe000h; C=X
```

Инструкция LACL

LACL – загрузка младшего слова и очистка старшего слова аккумулятора.

СИНТАКСИС

LACL dma	Прямая адресация
LACL ind [, ARn]	Косвенная адресация
LACL #k	Короткая непосредственная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	1	1	см. подраздел 14.7.3						

Короткая непосредственная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	0	0	1	см. подраздел 14.7.3							

ОПЕРАНДЫ

$0 \leq dma \leq 127$
$0 \leq$ новый ARP ≤ 7
$0 \leq k \leq 255$

ВЫПОЛНЕНИЕ

(PC) + 1 -> PC

Прямая или косвенная адресация:
0 -> ACC(31-16)
dma - >ACC(15-0)

Длинная непосредственная адресация:
0 -> ACC(31-16)
k - >ACC(15-0)

Не зависит от SXM.

ОПИСАНИЕ

Содержимое указанного адреса памяти данных или дополненная 0 в старших разрядах 8-битной константы загружается в младшее слово аккумулятора. Старшее слово аккумулятора заполняется 0. Данные интерпретируются как беззнаковые. Расширение знака не выполняется независимо от SXM.

СЛОВА

1

ЦИКЛЫ

Циклов при однократном выполнении инструкции (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	N+1+p+nd

Циклов при однократном выполнении инструкции (короткая непосредственная адресация)

ROM	DARAM	Внешняя память
1	1	1+p

ПРИМЕР 1

```
LACL    DAT1 ; (DP = 6)
        До выполнения           После выполнения
Память данных           Память данных
301h    01h               301h    01h ACC
7fffffffh; C=X  ACC       0h; C=X
```

ПРИМЕР 2

```
LACC    *-, AR4
        До выполнения           После выполнения
ARP     0               ARP     4
AR0     401h            AR20    400h
Память данных           Память данных
401h    0ffh            401h    0ffh
ACC     7fffffffh; C=X  ACC     0ffh; C=X
```

ПРИМЕР 3

```
LACC    #10h
        До выполнения           После выполнения
ACC     7fffffffh; C=X  ACC     010h; C=X
```

Инструкция LACT

LACT – загрузка аккумулятора со сдвигом, заданным TREG.

СИНТАКСИС
LACT dma Прямая адресация
LACT ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	1	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \leq dma \leq 127$
$0 \leq$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
dma << TREG1(3-0) -> ACC
if(SXM == 1)
Расширение знака dma;
else
Нет расширения знака dma;

Зависит от SXM.

ОПИСАНИЕ LACT загружает аккумулятор сдвинутым влево значением из памяти данных. Величина сдвига задана 4 младшими битами TREG (сдвиг на 1-15 разрядов). В старших битах происходит расширение знака, если SXM = 1, иначе они заполняются 0. LACT может использоваться для денормализации числа с плавающей точкой, если экспонента хранится в четырех младших битах регистра TREG, а мантисса считывается из памяти. Этот способ денормализации может использоваться, когда размер экспоненты 4 бита или меньше.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
LACT    DAT1 ; (DP = 6, SXM = 0)
        До выполнения          После выполнения
Память данных           Память данных
301h    1376h           301h    1376h
ACC     98f7ec83h C=X   ACC     13760h C=X
TREG1    14h              TREG1   14h
```

ПРИМЕР 2

```
LACC    *-, AR3 ; (SXM = 1)
        До выполнения          После выполнения
ARP     1               ARP     3
AR0     310h            AR20    300h
Память данных           Память данных
310h    0ff00h          310h    0ff00h
ACC     98f7ec83h C=X   ACC     0fffffe00h C=X
TREG1    11h              TREG1   11h
```

Инструкция LAR

LAR – загрузка вспомогательного регистра.

СИНТАКСИС	LAR ARx, dma	Прямая адресация
	LAR ARx, ind [, ARn]	Косвенная адресация
	LAR ARx, #k	Короткая непосредственная адресация
	LAR ARx, #lk	Длинная непосредственная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	ARX			0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	ARX			1	см. подраздел 14.7.3						

Короткая непосредственная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	ARX			8-битная константа							

Длинная непосредственная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	0	0	0	0	1	ARX		
16-битная константа															

ОПЕРАНДЫ

$0 \le dma \le 127$
$0 \le ARx \le 7$
$0 \le$ новый ARP ≤ 7
$0 \le k \le 255$
$0 \le lk \le 65535$

ВЫПОЛНЕНИЕ

Прямая или косвенная адресация:
(PC) + 1 -> PC
(dma) -> ARx

Короткая непосредственная адресация:
(PC) + 1 -> PC
k -> ARx

Длинная непосредственная адресация:
(PC) + 2 -> PC
lk -> ARx

ОПИСАНИЕ Содержимое заданного адреса памяти или 8- или 16-битная константы загружается в заданный вспомогательный регистр. Константы интерпретируются как беззнаковое число, независимо от SXM.
Инструкции LAR и SAR могут быть использованы для загрузки и сохранения вспомогательных регистров во время вызовов подпрограмм или прерываний. Если дополнительный регистр не будет использоваться для косвенной адресации, LAR и SAR позволяют использовать его как дополнительный регистр хранения данных, особенно для обмена значениями ячеек памяти без изменения содержимого ACC.

СЛОВА 1 (прямая, косвенная короткая непосредственная адресация)
2 (длинная непосредственная адресация)

ЦИКЛЫ

Циклов при однократном выполнении инструкции (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM	2	2	2+pcode
Источник: Внешняя	2+dsrc	2+dsrc	3+dsrc+pcode

Циклов при многократном выполнении инструкции под RPT (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM	2n	2n	2n+pcode
Источник: Внешняя	2n+ndsrc	2n+ndsrc	2n+1+ndsrc+pcode

Циклов при однократном выполнении инструкции (короткая непосредственная адресация)

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM	2	2	2+pcode
Источник: Внешняя	2+dsrc	2+dsrc	3+dsrc+pcode

Циклов при однократном выполнении инструкции (длинная непосредственная адресация)

ROM	DARAM	Внешняя память
2	2	2+2p

ПРИМЕР 1

```
LAR    AR0, DAT16 ; (DP = 6)
        До выполнения          После выполнения
Память данных          Память данных
310h    18h            310h    18h
AR0      6h              AR0     18h
```

ПРИМЕР 2

```
LAR    AR4, *-  ; (ARP = 4)
        До выполнения          После выполнения
Память данных           Память данных
300h    32h             300h    32h
AR4      300h             AR4      32h
```

ПРИМЕР 3

```
LAR    AR4, #01h
        До выполнения          После выполнения
AR4     0ff09h          AR4     01h
```

ПРИМЕР 4

```
LAR    AR4, #3fffh
        До выполнения          После выполнения
AR4     0h              AR4     3fffh
```

Инструкция LDP

LDP – загрузка указателя страницы памяти данных.

СИНТАКСИС

LDP dma	Прямая адресация
LDP ind [, ARn]	Косвенная адресация
LDP #k	Короткая непосредственная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	1	1	см. подраздел 14.7.3						

Короткая непосредственная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	0	9-битная константа								

ОПЕРАНДЫ

$0 \leq$ dma ≤ 127
$0 \leq$ новый ARP ≤ 7
$0 \leq k \leq 511$

ВЫПОЛНЕНИЕ

(PC) + 1 -> PC
Прямая или косвенная адресация:
9 младших бит (dma) -> DP
Короткая непосредственная адресация:
k -> DP

ОПИСАНИЕ

9 младших бит содержимого памяти данных или 9-битный непосредственный операнд загружается в регистр DP. Конкантенация DP с 7-битным адресом памяти данных формирует 16-битный адрес. DP может также загружаться командой LST.

СЛОВА

1

ЦИКЛЫ

Циклов при однократном выполнении инструкции (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM	2	2	2+pcode
Источник: Внешняя	2+dsrc	2+dsrc	3+dsrc+pcode

Циклов при многократном выполнении инструкции под RPT (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM	2n	2n	2n+pcode
Источник: Внешняя	2n+ndsrc	2n+ndsrc	2n+1+ndsrc+pcode

Циклов при однократном выполнении инструкции (короткая непосредственная адресация)

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM	2	2	2+pcode
Источник: Внешняя	2+dsrc	2+dsrc	3+dsrc+pcode

ПРИМЕР 1

```
LDP    DAT127 ; (DP = 511)
        До выполнения          После выполнения
Память данных          Память данных
0ffffh  0fedch         0ffffh  0fedch
DP       1ffh            DP       0dch
```

ПРИМЕР 2

```
LDP    #0h
        До выполнения          После выполнения
DP       1ffh            DP       0h
```

ПРИМЕР 3

```
LDP    *, AR5
        До выполнения          После выполнения
ARP     4              ARP     5
AR4     300h           AR4     300h
Память данных          Память данных
300h    06h            300h    06h
DP      1ffh           DP      06h
```

Инструкция LPH

LPH – загрузить старшее слово P регистра.

СИНТАКСИС
LPH dma Прямая адресация
LPH ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	0	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	0	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \leq dma \leq 127$
$0 \leq$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(dma) -> регистр P(31-16)

ОПИСАНИЕ В старшее слово регистра P загружается содержимое памяти данных. Младшее слово P не меняется.
LPH может использоваться для восстановления P регистра после прерывания или вызова подпрограммы.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
LPH    DAT0 ; (DP = 4)
          До выполнения          После выполнения
Память данных           Память данных
200h    0f79ch          200h    0f79ch
P        30079844h        P        0f79c9844h
```

ПРИМЕР 2

```
LPH    *, AR6
          До выполнения          После выполнения
ARP     5               ARP     6
AR5     200h            AR5     200h
Память данных           Память данных
200h    0f79ch          200h    0f79ch
P        30079844h        P        0f79c9844h
```

Инструкция LST

LST – загрузка статусного регистра.

СИНТАКСИС

LST #m, dma	Прямая адресация
LST #m, ind [, ARn]	Косвенная адресация

КОД КОМАНДЫ

Прямая адресация для LST #0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	0	dma						

Косвенная адресация для LST #0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	0	1	см. подраздел 14.7.3						

Прямая адресация для LST #1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	dma						

Косвенная адресация для LST #1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ

$0 \le dma \le 127$
$0 \le n \le 1$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ

(PC) + 1 -> PC
(dma) -> регистр статуса STn
dma (биты 13-15) -> ARP (независимо от нового ARP)

Влияет на ARB, ARP, OV, OVM, DP, CNF, TC, SXM, C, XF, PM.
Не изменяет INTM.

ОПИСАНИЕ

В статусный регистр STn (n = 0, 1) загружается значение из памяти данных. Следует обратить внимание, что бит INTM не изменяется LST #0. Кроме того, LST #0 не изменяет поле ARB в ST1, даже если загружается новое ARP. Если новое значение ARP задано в косвенном режиме адресации, оно игнорируется, а вместо него в ARP загружается значение из адреса памяти данных.
Когда загружается ST1, значение, загружаемое в ARB, также загружается в ARP.
LST может использоваться для восстановления ST0, ST1 после прерывания или вызова подпрограммы.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM	2	2	2+pcode
Источник: Внешняя	2+dsrc	2+dsrc	3+dsrc+pcode

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM	2n	2n	2n+pcode
Источник: Внешняя	2n+ndsrc	2n+ndsrc	2n+1+ndsrc+pcode

ПРИМЕР 1

```
MAR     *, AR0
LST     #0, *, AR1 ;Содержимое памяти данных,
                   ;адресуемой AR0, загружает-
ся в
                   ;ST0, исключая бит INTM.
Даже
                   ;когда задано новое значе-
ние
                   ;ARP игнорируется и старое
ARP
                   ;не загружается в ARB
```

ПРИМЕР 2

```
LST     #0, 60h ; (DP = 0)
        До выполнения          После выполнения
Память данных          Память данных
60h     2404h          60h     2404h
ST0     6e00h          ST0     2604h
ST1      0580h           ST1      0580h
```

ПРИМЕР 3

```
LST     #0, *-, AR1
        До выполнения          После выполнения
ARP     4              ARP     1
AR4     3ffh           AR4     2feh
Память данных          Память данных
3ffh    0ee04h         3ffh    0ee04h
ST0     0ee00h         ST0     0ee04h
ST1     f780h          ST1     f780h
```

ПРИМЕР 4

```
LST     #1, 0h  ;(DP = 6)
        До выполнения          После выполнения
Память данных          Память данных
300h    0e1bch         300h    0e1bch
ST0     0406h          ST0     0e406h
ST1     09a0h          ST1     0e1bch
```

Инструкция LT

LT – загрузка TREG.

СИНТАКСИС
LT dma — Прямая адресация
LT ind [, ARn] — Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	1	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(dma) -> TREG

ОПИСАНИЕ
В TREG загружается значение из памяти данных (dma). LT может использоваться для загрузки TREG при подготовке к умножению (см. LTA, LTD, LTP, LTS, MPY, MPYA, MPYS, MPYU).

СЛОВА
1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
LT    DAT24 ; (DP = 8)
       До выполнения          После выполнения
Память данных          Память данных
418h   62ch             418h   62ch
TREG    3h               TREG    62h
```

ПРИМЕР 2

```
LT    *, AR3 ;
       До выполнения          После выполнения
ARP     2              ARP     3
AR2     418h           AR2     418h
Память данных          Память данных
418h    62ch           418h    62ch
TREG     3h            TREG     62h
```

Инструкция LTA

LTA – загрузить TREG и аккумулировать предыдущий результат.

СИНТАКСИС
LTA dma Прямая адресация
LTA ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \leq dma \leq 127$
$0 \leq$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(dma) -> TREG
(ACC) + сдвинутый регистр P -> ACC

Зависит от OVM, PM; влияет на OV и C

ОПИСАНИЕ
В TREG загружается значение из памяти данных (dma). Содержимое регистра P, сдвинутое как определено битом статуса PM, прибавляется к ACC.
Функции LTA включаются в LTD.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
LTA    DAT36 ; (DP = 6, PM = 0)
        До выполнения          После выполнения
Память данных           Память данных
324h    62h             324h    62h
TREG    3h              TREG    62h
PREG    0fh             PREG    0fh
ACC      5h C=X           ACC     14h C=0
```

ПРИМЕР 2

```
LTA    *, AR5 ;
        До выполнения          После выполнения
ARP     4               ARP     5
AR2     324h            AR2     324h
Память данных           Память данных
324h    62h             324h    62h
TREG    3h              TREG    62h
PREG    0fh             PREG    0fh
ACC      5h C=X           ACC     14h C=0
```

Инструкция LTD

LTD – загрузка TREG, аккумулирование предыдущих данных и перемещение данных.

СИНТАКСИС
LTD dma Прямая адресация
LTD ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	1	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	1	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \leq dma \leq 127$
$0 \leq$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(dma) -> TREG
(dma) -> (dma + 1)
(ACC) + сдвинутый регистр PREG -> ACC

Зависит от OVM, PM; влияет на OV и C.

ОПИСАНИЕ В TREG загружается значение из памяти данных (dma). Содержимое регистра PREG, сдвинутое как определено полем PM, прибавляется к ACC. Содержимое указанной ячейки памяти копируется по следующему более старшему адресу. Эта инструкция допустима для всех блоков ОЗУ, сконфигурированных как память данных. Функция перемещения данных переходит через границы непрерывных блоков данных, но не может быть использована с внешней памятью данных или отображаемыми в память регистрами. Эта функция описана в описании инструкции DMOV. При работе с внешней памятью функция LTD идентична LTA.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	2+2d	2+2d	5+2d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	4n-2+2nd	4n-2+2nd	4n+1+2nd+p

ПРИМЕР 1

```
LTD    DAT126 ; (DP = 7, PM = 0)
        До выполнения          После выполнения
Память данных           Память данных
3feh    62h             3feh    62h
3ffh    0h              3fh     62h
TREG    3h              TREG    62h
PREG    0fh             PREG    0fh
ACC      5h C=X           ACC     14h C=0
```

ПРИМЕР 2

```
LTD     *, AR3
        До выполнения          После выполнения
ARP     1               ARP     3
AR1     3feh            AR1     3feh
Память данных           Память данных
3feh    62h             3feh    62h
3ffh    0h              3fh     62h
TREG    3h              TREG    62h
PREG    0fh             PREG    0fh
ACC      5h C=X           ACC     14h C=0
```

Инструкция LTP

LTP – загрузка TREG и запись PREG в аккумулятор.

СИНТАКСИС
LTP dma Прямая адресация
LTP ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	0	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \leq dma \leq 127$
$0 \leq$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(dma) -> TREG
сдвинутый регистр P -> ACC

Зависит от PM

ОПИСАНИЕ В TREG загружается значение из памяти данных (dma). Содержимое регистра PREG, сдвинутое как определено полем PM, загружается в ACC.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
LTP    DAT36 ; (DP = 6, PM = 0)
        До выполнения          После выполнения
Память данных           Память данных
324h    62h             324h    62h
324h    0h              324h    62h
TREG    3h              TREG   62h
P       0fh             P       0fh
ACC     5h C=X            ACC     0fh C=X
```

ПРИМЕР 2

```
LTP     *, AR5 ; (PM = 0)
        До выполнения          После выполнения
ARP     2               ARP     5
AR1     324h            AR2     324h
Память данных           Память данных
324h    62h             324h    62h
TREG    3h              TREG    62h
PREG    0fh             PREG    0fh
ACC     5h C=X            ACC     0fh C=X
```

Инструкция LTS

LTS – загрузить TREG и вычесть предыдущий результат.

СИНТАКСИС LTS dma Прямая адресация
LTS ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	0	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	0	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ $0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(dma) -> TREG
(ACC) – (сдвинутый регистр P) -> ACC

Зависит от OVM, PM; влияет на OV и C.

ОПИСАНИЕ В TREG загружается значение из памяти данных (dma).
Содержимое регистра PREG, сдвинутое как определено битом статуса PM, вычитается из ACC.
Бит переноса C сбрасывается (C=0), если результат вычитания генерирует заем, и устанавливается в 1 (C=1) в противном случае.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
LTS    DAT36 ; (DP = 6, PM = 0)
До выполнения          После выполнения
Память данных          Память данных
324h    62h            324h    62h
TREG    3h             TREG    62h
PREG    0fh            PREG    0fh
ACC     5h C=X           ACC     0FFFFFFF6h C=0
```

ПРИМЕР 2

```
LTS    *, AR2 ; (PM = 0)
До выполнения          После выполнения
ARP     1              ARP     2
AR2     324h           AR2     324h
Память данных          Память данных
324h    62ch           324h    62ch
TREG    3h             TREG    62h
PREG    0fh            PREG    0fh
ACC     5h C=X           ACC     0FFFFFFF6h C=0
```

Инструкция MAC

MAC – умножить и аккумулировать.

СИНТАКСИС MAC pma, dma Прямая адресация
MAC pma, ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	1	0	0	dma						
16-битная константа															

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	1	0	1	см. подраздел 14.7.3						
16-битная константа															

ОПЕРАНДЫ $0 \le dma \le 127$
$0 \le pma \le 65535$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ Инкремент PC, then
(PC) -> MSTACK
(pma) -> PC
ACC + (сдвинутый регистр PREG) ->ACC;
(dma) -> TREG;
(dma) * (pma, адресуемый PC) -> PREG;
Для косвенной адресации модифицировать текущий AR и ARP как задано;
(PC) + 1 -> PC;
While((счетчик_повтора) $\neq 0$)
ACC + (сдвинутый регистр PREG) ->ACC;
(dma) -> TREG;
(dma) * (pma, адресуемый PC) -> PREG;
Для косвенной адресации модифицировать текущий AR и ARP как задано;
(PC) + 1 -> PC;
(счетчик повтора) – 1 -> (счетчик повтора);

(MSTACK) -> PC;

Зависит от OVM, PM ; влияет на OV и C.

ОПИСАНИЕ

Содержимое регистра PREG, сдвинутое как определено полем PM, прибавляется к ACC. Бит переноса устанавливается (C=1), если результат сложения генерирует перенос и сбрасывается (C=0), в противном случае. Инструкция MAC перемножает значение из памяти данных (заданное dma) со значением из памяти программ (заданным pma). Результат умножения сохраняется в P-регистре.

Если программная память – это блок B0 внутрикристального ОЗУ, бит CNF должен быть установлен в 1. Старшие 8 бит адреса программной памяти должны быть 0FFh для адресации блока B0 внутрикристального ОЗУ программ. Старшие 8 бит адреса памяти данных должны быть 0 для доступа к адресам ниже 500h. В прямом режиме адресации dma не может быть изменен во время повторения инструкции. При повторении MAC адрес программной памяти, хранящийся в PC, увеличивается на 1 во время операции, что позволяет обрабатывать серии операндов в памяти. MAC применяется для вычисления длинных сумм произведений, т. к. выполняется как одноцикловая инструкция после старта конвейера PRT.

СЛОВА

2

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
1: DARAM/ROM 2: DARAM	3	3	3+2pcode
1: Внешняя 2: DARAM	3+pop1	3+pop1	3+pop1+2pcode
1: DARAM/ROM	3	3	3+2pcode
1: DARAM/ROM 2: Внешняя	3+dop2	3+dop2	3+dop2+2pcode
1: Внешняя 2: Внешняя	4+pop1+dop2	4+pop1+dop2	4+pop1+dop2+2pcode

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
1: DARAM/ROM 2: DARAM	n+2	n+2	n+2+2pcode
1: Внешняя 2: DARAM	n+2+npop1	n+2+npop1	n+2+npop1+2pcode
1: DARAM/ROM 2: Внешняя	n+2+ndop2	n+2+ndop2	n+2+ndop2+2pcode
1: Внешняя 2: Внешняя	2n+2+npop1 +ndop2	2n+2+npop1+ndop2	2n+2+npop1+ndop2 +2pcode

ПРИМЕР 1

```
MAC    0FF000h, 02h ; (DP = 6, PM = 0, CNF = 1)
До выполнения           После выполнения
Память данных           Память данных
302h    23h             302h    23h
Память программ         Память программ
0ff00h  4h              0ff00h  4h
TREG    45              TREG   23h
PREG    458972h         PREG   08ch
ACC     723EC41h C=X    ACC     76975B3h C=0
```

ПРИМЕР 2

```
MAC    0FF00h, *, AR5 ; (PM = 0, CNF = 1)
До выполнения           После выполнения
ARP     4               ARP     5
AR2     302h            AR2     302h
302h    23h             302h    23h
Память программ         Память программ
0ff00h  4h              0ff00h  4h
TREG    45              TREG    23h
PREG    458972h         PREG    08ch
ACC     723EC41h C=X    ACC     76975B3h C=0
```

Инструкция MACD

MACD – умножить с аккумулированием и перемещением данных.

СИНТАКСИС MACD pma, dma Прямая адресация
MACD pma, ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	1	1	0	dma						
16-битная константа															

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	1	1	1	см. подраздел 14.7.3						
16-битная константа															

ОПЕРАНДЫ $0 \le dma \le 127$
$0 \le pma \le 65535$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ (PC) + 2 -> PC
(PC) -> MCS
(pma) -> PC

Инкремент PC, then
(PC) -> MSTACK
(pma) -> PC
ACC + (сдвинутый регистр PREG) ->ACC;
(dma) -> TREG;
(dma) * (pma, адресуемый PC) -> PREG;
Для косвенной адресации модифицировать текущий AR и ARP как задано;
(PC) + 1 -> PC;
While((счетчик_повтора) $\neq$ 0)
ACC + (сдвинутый регистр PREG) ->ACC;
(dma) -> TREG;
(dma) * (pma, адресуемый PC) -> PREG;
Для косвенной адресации модифицировать текущий AR
и ARP как задано;
(PC) + 1 -> PC;
(dma) -> (dma + 1);
(счетчик_повтора) – 1 -> (счетчик_повтора);

(MSTACK) -> PC;

Зависит от OVM, PM; влияет на OV и C.

ОПИСАНИЕ Содержимое регистра PREG, сдвинутое как определено битами PM, прибавляется к ACC. Инструкция MACD перемножает значение из памяти данных (заданное dma) со значением из памяти программ (заданное pma).
Адрес памяти данных и программ может быть любым незарезервированным, внутри- и внекристальным. Если программная память – это блок B0 внутрикристального ОЗУ, бит CNF должен быть установлен в 1. Старшие 8 бит адреса программной памяти должны быть 0FFh для адресации блока B0 внутрикристального ОЗУ программ. Старшие 8 бит адреса памяти данных должны быть 0 для доступа к адресам ниже 500h. В прямом режиме адресации dma не может быть изменен во время повторения инструкции. Если MACD адресует отображаемый в память регистр или внешнюю память, эффект от MACD такой же, как от MAC (см. описание DMOV). MACD делает то же, что и MAC и дополнительно перемещает данные во внутренней памяти. Это делает MACD полезным в таких применениях, как вычисление свёрток.
При повторении MACD адрес программной памяти, хранящийся в PC, увеличивается на 1 во время операции, что позволяет обрабатывать серии операндов в памяти. MACD применяется для вычисления длинных сумм произведений, т. к. выполняется как одноцикловая инструкция после старта в цикле PRT.

СЛОВА 2

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
1: DARAM/ROM 2: DARAM	3	3	3+2pCode
1: Внешняя 2: DARAM	3+pop1	3+pop1	3+pop1+2pcode
1: DARAM/ROM 2: Внешняя[1)]	3+dop2	3+dop2	3+dop2+2pcode
1: Внешняя 2:Внешняя[1)]	4+pop1+dop2	4+pop1+dop2	4+pop1+dop2+2pcode

[1)] Операция по перемещению данных не выполняется, когда второй операнд во внешней памяти данных.

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
1:DARAM/ROM 2: DARAM	n+2	n+2	n+2+2pcode
1: Внешняя 2: DARAM	n+2+npop1	n+2+npop1	n+2+npop1+2pcode
1:DARAM/ROM 2: Внешняя[1)]	n+2+ndop2	n+2+ndop2	n+2+ndop2+2pcode
1: Внешняя 2:Внешняя[1)]	2n+2+npop1 +ndop2	2n+2+npop1+ndop2	2n+2+npop1+ndop2 +2pcode

[1)] Операция по перемещению данных не выполняется, когда операнд 2 во внешней памяти данных

ПРИМЕР 1

```
MACD   0FF000h, 02h ; (DP = 6, PM = 0, CNF = 1)
 До выполнения            После выполнения
 Память данных            Память данных
 308h    23h              308h    23h
 309h    18h              309h    23h
 Память программ          Память программ
 0ff00h  4h               0ff00h  4h
 TREG   45                TREG   23h
 P       458972h          P       08ch
 ACC     723EC41h C=X     ACC     76975B3h C=0
```

ПРИМЕР 2

```
 MACD    0FF00h, *, AR6 ; (PM = 0, CNF = 1)
  До выполнения           После выполнения
  ARP     5               ARP     6
  AR2     308h            AR2     308h
  Память  данных          Память данных
  308h    23h             308h    23h
  309h    18h             309h    23h
  Память программ         Память программ
```

```
0ff00h  4h              0ff00h  4h
TREG   45               TREG   23h
P       458972h         P       08ch
ACC     723EC41h C=X    ACC     76975B3h C=0
```

Примечание – Функция перемещения данных MACD работает только с внутрикристальным ОЗУ.

Инструкция MAR

MAR – модифицировать вспомогательный регистр.

СИНТАКСИС

MAR dma — Прямая адресация
MAR ind [, ARn] — Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ $0 \leq$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ (PC) + 1-> PC

Модифицирует ARP, AR, как задано при косвенной адресации. Работает как NOP в режиме прямой адресации

ОПИСАНИЕ В режиме косвенной адресации модифицируется дополнительный регистр и ARP; ссылка на память никак не используется. Старое значение ARP копируется в поле ARB регистра статуса ST1. Любая операция, выполняемая MAR, может быть выполнена любой инструкцией, использующей косвенную адресацию. ARP также может быть загружен инструкцией LST.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР 1

```
MAR    *, AR1 ; загрузка 1 в ARP
 До выполнения          После выполнения
 ARP     0              ARP     1
 ARB      7               ARB     0
```

ПРИМЕР 2

```
MAR    *+, AR5 ; увеличить текущий дополнительный
               ; регистр и загрузить 5 в ARP
 До выполнения          После выполнения
 AR1     34h            AR1     35h
 ARP     1              ARP     5
 ARB      0               ARB     1
```

Инструкция MPY

MPY – умножить.

СИНТАКСИС

MPY dma	Прямая адресация
MPY ind [, ARn]	Косвенная адресация
MPY #k	Короткая непосредственная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	0	1	см. подраздел 14.7.3						

Короткая непосредственная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	13-битная константа												

ОПЕРАНДЫ

$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7
$-4096 \le k \le 4095$

ВЫПОЛНЕНИЕ

Прямая или косвенная адресация:
(PC) + 1 -> PC
(TREG) * (dma) -> PREG

Короткая непосредственная адресация:
(PC) + 1 -> PC
(TREG) * k -> PREG

ОПИСАНИЕ

Содержимое TREG умножается на содержимое ячейки памяти данных. Результат пишется в регистр PREG. При короткой непосредственной адресации TREG умножается на знаковую 13-битную константу независимо от SXM.

СЛОВА

1 (Прямая, косвенная или короткая непосредственная адресация)

ЦИКЛЫ

Циклов при однократном выполнении инструкции (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

Циклов при однократном выполнении инструкции (короткая непосредственная адресация)

ROM	DARAM	Внешняя память
1	1	1+p

ПРИМЕР 1

```
MPY    DAT13 ; (DP = 8)
 До выполнения          После выполнения
 Память данных          Память данных
 40Dh    01h            40Dh    01h
 TREG    6h             TREG   6h
 P        36h             P        2Ah
```

ПРИМЕР 2

```
MPY    *, AR2
 До выполнения          После выполнения
 ARP     1              ARP     2
 AR2     40Dh           AR2     40Dh
 Память данных          Память данных
 40Dh    7h             40Dh    7h
 TREG    6h             TREG    6h
 P        36h             P        2Ah
```

ПРИМЕР 3

```
MPY    #031h
 До выполнения          После выполнения
 TREG    6h             TREG    6h
 P       36h            P       2Ah
```

Инструкция MPYA

MPYA – умножить и аккумулировать предыдущий результат.

СИНТАКСИС
MPYA dma — Прямая адресация
MPYA ind [, ARn] — Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(ACC) + (сдвинутый регистр PREG) -> ACC
(TREG) * (dma) -> PREG

Зависит от OVM, PM; влияет на OV и C.

ОПИСАНИЕ
Содержимое TREG умножается на содержимое ячейки памяти данных. Результат записывается в регистр P. Предыдущее значение P-регистра, сдвинутое как определено битами PM, прибавляется к ACC.

СЛОВА
1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
MPYA    DAT13 ; (DP = 6, PM = 0)
 До выполнения          После выполнения
 Память данных          Память данных
 30Dh    23h            30Dh    23h
 TREG    6h             TREG    6h
 PREG      36h          PREG    2Ah
 ACC      54h C=X         ACC      8Ah C=0
```

ПРИМЕР 2

```
MPYA    *, AR4 ; (PM = 0)
 До выполнения          После выполнения
 ARP     3              ARP     4
 AR3     30Dh           AR3     30Dh
 Память данных          Память данных
 30Dh    7h             30Dh    7h
 TREG    6h             TREG    6h
 PREG    36h            PREG    2Ah
 ACC      54h C=X         ACC      8Ah C=0
```

Инструкция MPYS

MPYS – умножить и вычесть предыдущий результат.

СИНТАКСИС
MPYS dma — Прямая адресация
MPYS ind [, ARn] — Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(ACC) – (сдвинутый регистр P) -> ACC
(TREG) * (dma) -> PREG

Зависит от OVM, PM; влияет на OV и C.

ОПИСАНИЕ Содержимое TREG умножается на содержимое ячейки памяти данных. Результат записывается в регистр P. Предыдущее значение P-регистра, сдвинутое как определено битами PM, прибавляется к ACC.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
MPYS    DAT13 ; (DP = 6, PM = 0)
 До выполнения          После выполнения
 Память данных          Память данных
 30Dh    7h             30Dh    7h
 TREG    6h             TREG    6h
 PREG    36h            PREG    2Ah
 ACC      54h C=X         ACC     1Eh C=1
```

ПРИМЕР 2

```
MPYS    *, AR5 ; (PM = 0)
 До выполнения          После выполнения
 ARP     4              ARP     5
 AR3     30Dh           AR3     30Dh
 Память данных          Память данных
 30Dh    7h             30Dh    7h
 TREG    6h             TREG    6h
 PREG    36h            PREG    2Ah
 ACC      54h C=X         ACC     1Eh C=1
```

Инструкция MPYU

MPYU – умножить без знака.

СИНТАКСИС
MPYU dma — Прямая адресация
MPYU ind [, ARn] — Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	1	0	1	1	См. раздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
беззнаковый(TREG) * беззнаковый(dma) -> PREG

Не зависит от SXM.

ОПИСАНИЕ
Беззнаковое содержимое TREG умножается на беззнаковое содержимое ячейки памяти данных. Результат пишется в регистр P. Умножитель обрабатывает операнды как знаковые 17-разрядные со старшим битом 0. Сдвигатель при чтении регистра P всегда выполняет расширение знака, когда PM = 3 (сдвиг на 6 бит вправо). Поэтому такой режим сдвига не может использоваться, если требуется беззнаковый результат. MPYU можно использовать для операций с повышенной точностью.

СЛОВА
1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
MPYU    DAT16 ; (DP = 4)
 До выполнения          После выполнения
 Память данных          Память данных
 210h    0FFFFh         210h    0FFFFh
 TREG    0FFFFh         TREG    0FFFFh
 PREG     1h              PREG    0FFFE0001h
```

ПРИМЕР 2

```
MPYU    *, AR2

 До выполнения          После выполнения
 ARP     5              ARP     6
 AR2     210h           AR2     210h
 Память данных          Память данных
 210h    0FFFFh         210h    0FFFFh
 TREG    0FFFFh         TREG    0FFFFh
 PREG     1h              PREG    0FFFE0001h
```

Инструкция NEG

NEG – изменить знак аккумулятора.

СИНТАКСИС NEG

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
ACC * -1 -> ACC

Зависит от OVM; влияет на OV и C.

ОПИСАНИЕ Содержимое аккумулятора заменяется его дополнением до 2. При выполнении NEG к 80000000h устанавливается бит OV. Если OVM = 1, содержимое аккумулятора заменяется 7FFFFFFFh. Если OVM = 0, результат 80000000h. Бит C устанавливается в 0 при ненулевом результате и в 1 при нулевом.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР 1
```
NEG     ;( OVM = X )
 До выполнения            После выполнения
 ACC 0FFFFF288h C=X,OV=X  ACC 0DD8h C=OV=0
```

ПРИМЕР 2
```
NEG     ;( OVM = 0 )
 До выполнения            После выполнения
 ACC 80000000h C=X,OV=X   ACC 80000000h
C=0,OV=1
```

ПРИМЕР 3
```
NEG     ;( OVM = 1 )
 До выполнения            После выполнения
 ACC 80000000h C=X,OV=X   ACC 7FFFFFFFh C=0,OV=1
```

Инструкция NMI

NMI – немаскируемое прерывание.

СИНТАКСИС NMI

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (PC) + 1 -> стек
24 -> PC

ОПИСАНИЕ Программный счетчик устанавливается на вектор немаскируемого прерывания 24h. Эффект от инструкции тот же, что и от аппаратного немаскируемого прерывания.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции		
ROM	DARAM	Внешняя память
4	4	4+3p

ПРИМЕР
```
NMI  ; Управление передается по адресу 24h и PC+1
     ; помещается в стек.
```

Инструкция NOP

NOP – нет операции.

СИНТАКСИС NOP

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC

ОПИСАНИЕ Никаких операций не выполняется. NOP влияет только на PC. NOP – это то же самое, что и MAR с прямой адресацией и dma = 0h. NOP используется как заполнитель или временная инструкция при создании программы.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР

```
NOP     ;не выполняются никакие операции
```

Инструкция NORM

NORM – нормализация содержимого аккумулятора.

СИНТАКСИС NORM {ind}

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7	6 5 4 3 2 1 0
1	0	1	0	0	0	0	0	1	см. подраздел 14.7.3

ОПЕРАНДЫ Операнд ind: Один из следующих методов индексации: *, *+, *–, *0+, *0–, *BR0+, *BR0–

ВЫПОЛНЕНИЕ
```
(PC) + 1 -> PC
 if(ACC == 0)
   TC -> 1;
 else, if (ACC(31) xor ACC(30)) = 0):
       TC -> 0,
       (ACC) * 2 -> ACC;
       Заданная модификация текущего AR;
     else
       TC -> 1.
```

Зависит от TC; изменяет TC

ОПИСАНИЕ Инструкция NORM предназначена для нормализации знакового содержимого аккумулятора. Нормализация числа с фиксированной точкой разделяет его на мантиссу и экспоненту. Для этого должен быть найден размер числа с расширенным знаком. Выполняется XOR бит 31 и 30 ACC, чтобы определить, является ли бит 30 частью числа или расширением знака. Если они одинаковы, оба бита указывают на знаковое расширение, поэтому аккумулятор сдвигается влево для удаления лишнего знакового бита.

Текущий AR модифицируется заданным образом для генерации размера экспоненты. Предполагается, что текущий AR инициализируется перед началом нормализации. Модификация текущего AR по умолчанию – инкремент. Для законченной нормализации 32-битного числа может потребоваться многократное выполнение NORM. Хотя использование NORM с RPT не обеспечивает выход из цикла при окончании нормализации, никакие операции не выполняются для остатка цикла. NORM работает с положительными и отрицательными числами с дополнением до двух.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР 1

```
NORM    *+
 До выполнения            После выполнения
 ARP     2                ARP     2
 AR2     0h               AR2     01h
 ACC 0FFFFF001h TC=X      ACC 0FFFE002h TC=0
```

ПРИМЕР 2

```
31-битная нормализация:
     MAR     *, APR1  ; AR1 используется для
                      ;хранения экспоненты
     LAR     AR1, #0h ; очистка счетчика
                      ;экспоненты
LOOP NORM    *+       ; нормализация одного бита
     BCND    LOOP, NTC; Если TC = 0, размер
еще
                         ;не найден
```

ПРИМЕР 3

```
15-битная нормализация
MAR   *, APR1  ; AR1 используется для хранения
               ;экспоненты
LAR   AR1, #0fh; инициализация счетчика
               ;экспоненты
RPT   #14      ; Задана 15-битная нормализация
               ;(4-битная экспонента и 16-битная
               ;мантисса
NORM  *-       ; NORM автоматически заканчивает
               ;сдвиг, когда найден первый
               ;значимый бит, выполняя NOP до
               ;выхода из цикла
```

Метод из примера 2 используется для нормализации 32-битного числа и задает 5-битную экспоненту. Метод из третьего примера используется для нормализации 16-битного числа и задает 4-битную экспоненту. Если число требует малой степени нормализации, метод во втором примере может быть предпочтительнее третьего. В примере 2 цикл выполняется до завершения нормализации. В примере 3 всегда выполняется 15 циклов. Пример 2 более эффективен, если требуемое количество сдвигов пять или меньше. Если требуемое количество сдвигов шесть или больше, пример 3 эффективнее. Результирующее значение в дополнительном регистре не будет действительной

экспонентой числа во всех случаях, однако оно может быть использовано для нахождения экспоненты.

Инструкция OR

OR – логическое «ИЛИ» с аккумлятором.

СИНТАКСИС	OR dma	Прямая адресация
	OR ind [, ARn]	Косвенная адресация
	OR #lk [, shift]	Длинная непосредственная адресация
	OR #lk, 16	Длинная непосредственная адресация с левым сдвигом на 16

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	0	0	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	1	0	0	1	1	см. подраздел 14.7.3						

Длинная непосредственная адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	1	1	0	0	SHFT			
16-битная константа															

Длинная непосредственная адресация со сдвигом на 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	0	1	0	0	0	0	0	1	0
16-битная константа															

ОПЕРАНДЫ $0 \leq dma \leq 127$
$0 \leq$ новый ARP ≤ 7
$0 \leq$ сдвиг ≤ 16
lk – 16-битная константа

ВЫПОЛНЕНИЕ Прямая или косвенная адресация:
(PC) + 1 -> PC
(ACC(15-0)) OR (dma) -> ACC(15-0)
(ACC(31-16)) -> ACC(31-16)

Непосредственная адресация:
(PC) + 2 -> PC
(ACC) OR lk<<сдвиг -> ACC

Не зависит от SXM

ОПИСАНИЕ Выполняется «ИЛИ» аккумулятора со значением памяти данных или со сдвинутой влево длинной константой. Все биты операнда, не занятые данными, заполняются 0 до 32-битного значения, независимо от SXM, так старшее слово аккумулятора не меняется при прямой или косвенной адресации или непосредственной адресации без сдвига. При сдвиге младшие биты операнда заполняются 0.

СЛОВА 1 (прямая или косвенная адресация)
2 (длинная непосредственная адресация)

ЦИКЛЫ

Циклов при однократном выполнении инструкции (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

Циклов при однократном выполнении инструкции (длинная непосредственная адресация)

ROM	DARAM	Внешняя память
2	2	2+2p

ПРИМЕР 1

```
OR     DAT8   ;(DP = 8)
 До выполнения          После выполнения
 Память данных          Память данных
 408h    0F000h         408h    0F000h
 ACC      100002h C=X     ACC      10F002H C=X
```

ПРИМЕР 2

```
OR     *, AR0
 До выполнения          После выполнения
 ARP     1              ARP     0
 AR1     300h           AR1     300h
 Память данных          Память данных
 300h    01111h         300h    01111h
 ACC      222h C=X        ACC      1333H C=X
```

ПРИМЕР 3

```
OR     #08111h, 8
 До выполнения          После выполнения
 ACC     0FF0000h C=X   ACC     0FF1100h C=X
```

Инструкция OUT

OUT – вывод данных в порт ввода - вывода.

СИНТАКСИС
OUT dma, PA Прямая адресация
OUT ind, PA [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	0	dma						
16-битная константа															

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	1	см. подраздел 14.7.3						
16-битная константа															

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7
$0 \le pa \le 65535$

ВЫПОЛНЕНИЕ
(PC) + 2 -> PC
адрес порта -> адресная шина (A15–A0)
(dma) -> шина данных (D15–D0)

ОПИСАНИЕ OUT пишет 16-битное значение из памяти данных в порт ввода - вывода. На линии IS# устанавливается низкий уровень для индикации доступа к портам ввода - вывода; STRB#, RD/WR# и READY работают так же, как при записи во внешнюю память. RPT может использоваться с инструкцией OUT для вывода последовательности слов из памяти в порт ввода-вывода.

СЛОВА 2

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM	3+iodst	3+iodst	5+iodst+2pcode
Источник: Внешняя	3+dsrc+iodst	3+dsrc+iodst	6+dsrc+iodst+2pcode

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM	3n+niodst	3n+niodst	3n+3+niodst+2pcode
Источник: Внешняя	5n–2+ndsrc+niodst	5n–2+ndsrc++niodst	5n+1+ndsrc+niodst+2pcode

ПРИМЕР 1

```
OUT DAT0, PA7 ; (DP = 4) Вывод слова данных из
              ;памяти данных по адресу 200h в
              ;порт 7
```

ПРИМЕР 2

```
OUT *, PA15   ;Вывод слова данных, указанного
              ;текущим дополнительным реги-
стром в
              ;порт 15
```

Инструкция PAC

PAC – загрузить PREG в аккумулятор.

СИНТАКСИС PAC

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(сдвинутый регистр P) -> ACC

Зависит от PM.

ОПИСАНИЕ Содержимое регистра PREG, сдвинутое как указано в поле PM, загружается в аккумулятор.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР

```
PAC
 До выполнения          После выполнения
 P       144h           P 144h
 ACC     23h C=X        ACC     144h C=X
```

Инструкция POP

POP – записать содержимое верхушки стека в аккумулятор.

СИНТАКСИС POP

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(TOS) -> ACC(0-15)
0 -> ACC(31-16)
Стек проталкивается на один уровень ниже

ОПИСАНИЕ Содержимое вершины стека (TOS) копируется в младшее слово аккумулятора, затем стек проталкивается на один уровень ниже. Старшее слово аккумулятора обнуляется. Реализация аппаратного стека – LIFO на восемь позиций. При POP каждое значение стека копируется вверх на одну позицию. После POP внизу стека будет два одинаковых значения. Поскольку каждое значение стека копируется, после более 7 POP (POP, POPD, RETE, RETI, RET) все семь уровней стека будут заполнены одним значением. Переполнение стека не обнаруживается.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР

```
POP
 До выполнения          После выполнения
 ACC 82H C=X            ACC     45h C=X

 Stack 45h              Stack 16h
       16h                    7h
       7h                     33h
       33h                    42h
       42h                    56h
       56h                    37h
       37h                    61h
        61h                      61h
```

Инструкция POPD

POPD – записать содержимое верхушки стека в память.

СИНТАКСИС

POPD dma — Прямая адресация
POPD ind [, ARn] — Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	0	1	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ

$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ

(PC) + 1 -> PC
(TOS) -> dma
Стек проталкивается на один уровень выше.

ОПИСАНИЕ

Содержимое вершины стека (TOS) копируется в заданную ячейку памяти, затем стек проталкивается на один уровень выше. Выход за границы стека не обнаруживается.

СЛОВА

1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	2+d	2+d	4+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	2n+nd	2n+nd	2n+2+nd+p

ПРИМЕР 1

```
POPD DAT10 ;(DP = 8)
 До выполнения          После выполнения
 Память данных          Память данных
 40Ah    55h            40Ah    45h
 Stack 45h              Stack 16h
       16h                     7h
       7h                    33h
       33h                   42h
       42h                   56h
       56h                   37h
       37h                   61h
        61h                     61h
```

ПРИМЕР 2

```
POPD *+, AR1
 До выполнения          После выполнения
 ARP     0              ARP     1
 AR0     300h           AR0     301h
 Память данных          Память данных
 300h   55h             300h    45h
 Stack 45h              Stack 16h
       16h                     7h
       7h                    33h
       33h                   42h
       42h                   56h
       56h                   37h
       37h                   61h
        61h                     61h
```

Инструкция PSHD

PSHD – записать содержимое памяти данных в верхушку стека.

СИНТАКСИС
PSHD dma — Прямая адресация
PSHD ind [, ARn] — Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	1	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	1	1	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(dma) -> TOS
(PC) + 1 -> PC
Все ячейки стека вниз на 1 уровень

ОПИСАНИЕ
Содержимое памяти данных помещается на вершину стека. Стек проталкивается вниз (см. описание PUSH). Первое значение стека теряется.

СЛОВА
1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
PSHD DAT10 ;(DP = 8)
 До выполнения          После выполнения
 Память данных          Память данных
 40Ah    55h            40Ah    55h
 Stack 45h              Stack 55h
       16h                    45h
       7h                     16h
       33h                     7h
       42h                    33h
       56h                    42h
       37h                    56h
        61h                      37h
```

ПРИМЕР 2

```
POHD *, AR1
 До выполнения          После выполнения
 ARP     0              ARP     1
 AR0     300h           AR0     300h
 Память данных          Память данных
 300h   55h             300h    55h
 Stack 45h              Stack 55h
       16h                    45h
       7h                     16h
       33h                     7h
       42h                    33h
       56h                    42h
       37h                    56h
        61h                      37h
```

Инструкция PUSH

PUSH – прочитать содержимое аккумулятора в верхушку стека.

СИНТАКСИС PUSH

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
Все ячейки стека вниз на один уровень.
ACC(15-0) -> TOS

ОПИСАНИЕ Содержимое младшего слова аккумулятора копируется в вершину аппаратного стека.
Реализация аппаратного стека – LIFO на восемь позиций. Если произведено более восьми PUSH (CALA, CALL, CC, PSHD, PUSH), первое записанное в стеке значение теряется.

СЛОВА 1

ЦИКЛЫ

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР

```
PUSH
 До выполнения          После выполнения
 ACC     7h C=X         ACC     7h C=X
 Stack 45h              Stack 7h
       16h                    45h
       7h                     16h
       33h                     7h
       42h                    33h
       56h                    42h
       37h                    56h
        61h                     37h
```

Инструкция RET

RET – возврат из подпрограммы.

СИНТАКСИС RET

КОД КОМАНДЫ

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

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (TOS) -> PC;
стек вверх на один уровень;

ОПИСАНИЕ Содержимое вершины стека копируется в PC. Стек перемещается вверх на один уровень. RET используется с CALA, CALL и CC для подпрограмм.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
4	4	4+3p

Циклов при однократном выполнении инструкции для RETD

ROM	DARAM	Внешняя память
2	2	2+p

ПРИМЕР

```
RET
 До выполнения          После выполнения
 PC      96h            PC      45h
 Stack 45h              Stack 16h
       16h                     7h
       7h                     33h
       33h                    42h
       42h                    56h
       56h                    37h
       37h                    61h
        61h                      61h
```

Инструкция RETC

RETC – возврат с условием.

СИНТАКСИС RETC cond_1 [, cond_2] [,...]

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	TP		ZLVC				ZLVC			

Примечание - Поля TP и ZLVC (см. 14.8.1).

ОПЕРАНДЫ cond:

ACC=0	EQ
ACC≠0	NEQ
ACC<0	LT
ACC<=0	LEQ
ACC>0	GT
ACC>=0	GEQ
C=0	NC
C=1	C
OV=0	NOV
OV=1	OV
~BIO	BIO
TC=0	NTC
TC=1	TC
Без условия	UNC

ВЫПОЛНЕНИЕ
```
if( условие (я) )
  (TOS) -> PC;
  стек вниз на 1 уровень;
  else
    продолжить
```

ОПИСАНИЕ Стандартный возврат RET происходит при выполнении условий. Должны выполняться все заданные условия.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Условие	ROM	DARAM	Внешняя память
Выполнено	4	4	4+3p
Не выполнено	2	2	2+p

ПРИМЕР
```
RETC GEQ, NOV ; Выполнить возврат при ACC ≤ 0 и
              ; OV = 0
```

Инструкция ROL

ROL – циклический сдвиг аккумулятора влево.

СИНТАКСИС ROL

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(ACC(31)) -> C
(ACC(30-0)) -> ACC(31-1)
(C до ROL) -> ACC(0)

Влияет на C, изменяет C.
Не зависит от SXM.

ОПИСАНИЕ ROL осуществляет циклический сдвиг аккумулятора влево. Старший бит аккумулятора сдвигается в бит переноса, старое значение C помещается в младший бит аккумулятора.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР

```
ROL
          До выполнения            После выполнения
 ACC      0B0001234h C=0  ACC      60002468h
C=1
```

Инструкция ROR

ROR – циклический сдвиг аккумулятора вправо.

СИНТАКСИС ROR

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(ACC(0)) -> C
(ACC(31-1)) -> ACC(30-0)
(C до ROL) -> ACC(31)

Влияет на C, зависит от C.
Не зависит от SXM.

ОПИСАНИЕ ROR осуществляет циклический сдвиг аккумулятора вправо. Младший бит аккумулятора сдвигается в бит переноса, старое значение C помещается в старший бит аккумулятора.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР

```
ROR
          До выполнения            После выполнения
 ACC      B0001235h C=0   ACC      5800091Ah C=1
```

Инструкция RPT

RPT – повторить инструкцию.

СИНТАКСИС

RPT dma	Прямая адресация
RPT ind [, ARn]	Косвенная адресация
RPT #k	Короткая непосредственная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	1	1	см. подраздел 14.7.3						

Короткая непосредственная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	0	1	1	8-битная константа							

ОПЕРАНДЫ

$0 \leq dma \leq 127$
$0 \leq$ новый ARP ≤ 7
$0 \leq k \leq 255$

ВЫПОЛНЕНИЕ

Прямая или косвенная адресация:
(PC) + 1 -> PC
(dma(7-0)) -> RPTC

Короткая непосредственная адресация:
(PC) + 1 -> PC
k -> RPTC

ОПИСАНИЕ

Счетчик повтора (RPTC) загружается в соответствии с используемым способом адресации. Команда, следующая за RPT, повторяется N=(RPTC)+1 раз. Т. к. RPTC не сохраняется при переключении контекста, циклы повтора, как и многоцикловые инструкции, не прерываются. При сбросе RPTC устанавливается в 0. RPT полезна для операций перемещения блоков, умножения-аккумулирования, нормализации и т. д.

СЛОВА

1 (Прямая, косвенная или короткая непосредственная адресация)

ЦИКЛЫ

Циклов при однократном выполнении инструкции			
Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при однократном выполнении инструкции		
ROM	DARAM	Внешняя память
1	1	1+p

ПРИМЕР 1

```
RPT DAT127  ;(DP = 31: адреса 0F80h–0FFFh)
            ;Повтор следующей инструкции 13
раз.
         До выполнения          После выполнения
 Память данных            Память данных
 0FFFh   0Ch              0FFFh  0Ch
 RPTC     0h                RPTC   0Ch
```

ПРИМЕР 2

```
RPT *,AR1 ; Повтор следующей инструкции 4096
раз.
         До выполнения          После выполнения
 ARP     0                ARP     1
 AR0     300h             AR0     300h
 Память данных            Память данных
 300h    0FFFh            300h    0FFFh
 RPTC     0h                RPTC    0FFFh
```

ПРИМЕР 3

```
RPT #1 ; Повтор следующей инструкции 2 раза.
         До выполнения          После выполнения
  PTC    0h               RPTC   1Ch
```

Инструкция SACH

SACH – сохранить старшее слово аккумулятора в памяти.

СИНТАКСИС
SACH dma [, shift2] Прямая адресация
SACH ind [, shift2 [, ARn]_] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	SHFT			0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	SHFT			1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7
$0 \le shift2 \le 7$ (по умолчанию 0)

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(ACC) << сдвиг -> dma

Не зависит от SXM

ОПИСАНИЕ SACH копирует весь аккумулятор в сдвигатель, где он сдвигается влево на 0-7 бит, а затем старшие 16 бит копируются в память. Значение аккумулятора не меняется.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	2+d	2+d	4+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	2n+nd	2n+nd	2n+2+nd+p

ПРИМЕР 1

```
SACH    DAT10, 1 ; (DP = 4)

До выполнения           После выполнения
ACC     4208001h C=X    ACC     4208001h C=X
Память данных           Память данных
```

```
                20Ah    00h             20Ah    0841h
```

ПРИМЕР 2

```
SACH    *+, 0, AR2
 До выполнения          После выполнения
 ARP     1              ARP     2
 AR2     0300h          AR2     0301h
 ACC     4208001h C=X   ACC     4208001h C=X
 Память данных          Память данных
 300h    0h             300h    0420h
```

Инструкция SACL

SACL – сохранить младшее слово аккумулятора в памяти данных.

СИНТАКСИС
SACL dma [, shift2] Прямая адресация
SACL ind [, shift2 [, ARn]_] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	SHF			0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	SHF			1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7
$0 \le shift2 \le 7$ (по умолчанию 0)

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
((ACC) << shift2)(15-0) -> dma

Не зависит от SXM

ОПИСАНИЕ Младшее слово аккумулятора сдвигается влево на 0 - 7 бит, а затем копируется в память. Младшие биты заполняются нулем. Значение аккумулятора не меняется.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	2+d	2+d	4+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	2n+nd	2n+nd	2n+2+nd+p

ПРИМЕР 1

```
SACL    DAT10, 1 ; (DP = 4)
 До выполнения          После выполнения
 ACC     7C638421 C=X   ACC     7C638421 C=X
 Память данных          Память данных
 20Ah    00h              20Ah    0841h
```

ПРИМЕР 2

```
SACL    *, 0, AR7
 До выполнения          После выполнения
 ARP     6              ARP     7
 AR2     0300h          AR2     0300h
 ACC     00FF8421 C=X   ACC     00FF8421 C=X
 Память данных          Память данных
 300h    5h               300h    8421h
```

Инструкция SAR

SAR – сохранить вспомогательный регистр в памяти.

СИНТАКСИС	SAR ARx, dma	Прямая адресация
	SAR ARx, ind [, ARn]	Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	ARX			0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	ARX			1	см. подраздел 14.7.3						

ОПЕРАНДЫ $0 \le dma \le 127$
$0 \le ar \le 7$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(ARx) -> (dma)

ОПИСАНИЕ Содержимое заданного вспомогательного регистра записывается по заданному адресу. При модификации содержимого текущего вспомогательного регистра в режиме косвенной адресации SAR ARx (где x = ARP) записывает значение вспомогательного регистра до его изменения. LAR и SAR могут быть использованы для загрузки и сохранения вспомогательных регистров во время вызова подпрограмм или прерываний. Если вспомогательный регистр не будет использоваться для косвенной адресации, LAR и SAR позволяют использовать его как дополнительный регистр хранения данных, особенно для обмена значениями ячеек памяти без изменения содержимого ACC.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	2+d	2+d	4+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	2n+nd	2n+nd	2n+2+nd+p

ПРИМЕР 1

```
SAR    AR0, DAT30 ; (DP = 6)
 До выполнения          После выполнения
 Память данных          Память данных
 31Eh    18h            31Eh    37h
 AR0      37h             AR0      37h
```

ПРИМЕР 2

```
SAR    AR4, *-
 До выполнения          После выполнения
 Память данных          Память данных
 401h     0h            401h    401h
 AR4      401h            AR4      402h
```

Инструкция SBRK

SBRK – вычесть короткое слово из вспомогательного регистра.

СИНТАКСИС SBRK #k

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7 6 5 4 3 2 1 0
0	1	1	1	1	1	0	0	8-битная константа

ОПЕРАНДЫ $0 \leq k \leq 255$

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
Текущий AR – 8-битная положительная константа -> текущий AR

ОПИСАНИЕ 8-битное непосредственное значение вычитается из текущего вспомогательного регистра; результат меняет старое содержимое вспомогательного регистра. Вычитание выполняется в ARAU, непосредственное значение считается 8-битным беззнаковым целым.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции		
ROM	DARAM	Внешняя память
1	1	1+p

ПРИМЕР

```
SBRK 0FFh
 До выполнения          После выполнения
 ARP     7              ARP     7
 AR7     0h               AR7     0FF01h
```

Инструкция SETC

SETC – установить управляющий бит.

СИНТАКСИС SETC control_bit

КОД КОМАНДЫ

SETC C (Установить бит переноса)

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

SETC CNF (Установить бит управления конфигурацией)

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

SETC: INTM (Установить бит режима прерывания)

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

SETC OVM (Установить бит режима переполнения)

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

SETC: SXM (Установить бит режима расширения знака)

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

SETC TC (Установить бит тестирования/управления)

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

SETC XF (Установить бит флага вывода XF)

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

ОПЕРАНДЫ control_bit : бит ST0 или ST1(из следующего набора) :
(C, CNF, IMTM, OVM, SXM, TC, XF)

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
1 -> control_bit

ОПИСАНИЕ Заданный управляющий бит устанавливается в 1. Чтобы загрузить ST0 и ST1, может также быть использована команда LST.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции		
ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT		
ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР 1

```
SETC    TC ; TC – бит 11 ST1
 До выполнения          После выполнения
 ST1      x1xxh           st1      x9xxh
```

Инструкция SFL

SFL – сдвиг аккумулятора влево.

СИНТАКСИС SFL

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(ACC(31)) -> C
(ACC(30-0)) -> ACC(31-1)
0 -> ACC(0)

Влияет на C.
Не зависит от SXM.

ОПИСАНИЕ SFL сдвигает весь ACC влево на 1 бит. Младший бит заполняется 0, а старший сдвигается в бит C. В отличии от SFR, инструкция SFL не зависит от SXM.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР 1

```
SFL
            До выполнения            После выполнения
 ACC      0B0001234h C=X  ACC      60002468h
C=1
```

Инструкция SFR

SFR – сдвиг аккумулятора вправо.

СИНТАКСИС SFR

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ
```
(PC) + 1 -> PC
 if(SXM == 0)
    0 -> ACC(31);
 (ACC(31-1)) -> ACC(30-0)
 ACC(0) -> C
```

Влияет на C.
Зависит от SXM.

ОПИСАНИЕ SFR сдвигает ACC вправо на 1 бит.
Если SXM = 1, выполняется арифметический сдвиг. Знаковый (старший) бит не меняется и копируется в 30 бит.
Если SXM = 0, выполняется логический сдвиг вправо. Старшие биты заполняется 0. Младший бит сдвигается в бит C.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР 1
```
SFR  ; (SXM = 0)
          До выполнения            После выполнения
 ACC      0B0001234h C=X  ACC      5800091Ah
C=0
```

ПРИМЕР 2
```
SFR  ; (SXM = 1)
          До выполнения            После выполнения
 ACC      0B0001234h C=X  ACC      D5800091Ah
C=0
```

Инструкция SPAC

SPAC – вычесть регистр PREG из ACC.

СИНТАКСИС SPAC

КОД КОМАНДЫ

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

ОПЕРАНДЫ нет

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
(ACC) – сдвинутый регистр PREG -> ACC

Зависит от OVM, PM; влияет на OV и C.
Не зависит от SXM.

ОПИСАНИЕ Содержимое регистра P PREG, сдвинутое как определено битами PM, вычитается из ACC. SPAC не зависит от SXM, всегда происходит расширение знака P PREG. SPAC является подфункцией инструкций LTS, MPYS, SQRS.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при многократном выполнении инструкции под RPT

ROM	DARAM	Внешняя память
n	n	n+p

ПРИМЕР

```
SPAC    ; (PM = 0)
          До выполнения          После выполнения
 PREG    10000000h       PREG    10000000h
 ACC      70000000h C=X   ACC      60000000h
C=1
```

Инструкция SPH

SPH – сохранить старшее слово регистра PREG в памяти.

СИНТАКСИС
SPH dma Прямая адресация
SPH ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	0	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	0	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \leq dma \leq 127$
$0 \leq$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(выход сдвигателя PREG(31-16)) -> dma

ОПИСАНИЕ Старшее слово регистра P, сдвинутое как определено битами PM, записывается в память данных. Ни регистр PREG, ни аккумулятор не меняются. В старших битах происходит расширение знака при выбранном режиме сдвига 6. При левом сдвиге младшие биты берутся из младшего слова PREG.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	2+d	2+d	4+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	2n+nd	2n+nd	2n+2+nd+p

ПРИМЕР 1

```
SPH    DAT3 ; (DP = 4, PM = 0)
 До выполнения          После выполнения
 PREG    0FE079844h     PREG   0FE079844h
 Память данных          Память данных
 203h    4567h            203h   0FE07h
```

ПРИМЕР 2

```
SPH    *, AR7 ; (PM=2)
         До выполнения          После выполнения
 ARP     6              ARP     7
 AR6     203h           AR6     203h
 Память данных          Память данных
 203h    4567h          203h   0E079h
 PREG    0FE079844h       PREG   0FE079844h
```

Инструкция SPL

SPL – сохранение младшего слова регистра PREG в памяти.

СИНТАКСИС
SPL dma Прямая адресация
SPL ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	0	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	0	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \leq dma \leq 127$
$0 \leq$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(выход сдвигателя PREG (15-0)) -> dma

ОПИСАНИЕ Младшее слово регистра PREG, сдвинутое как определено битами PM, записывается в память данных. Ни регистр PREG, ни аккумулятор не меняются. Старшие биты берутся из старшего слова PREG при выбранном режиме сдвига 6. При левом сдвиге младшие биты заполняются 0.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	2+d	2+d	4+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	2n+nd	2n+nd	2n+2+nd+p

ПРИМЕР 1

```
SPL    DAT3 ; (DP = 4, PM = 0)
          До выполнения          После выполнения
 Память данных           Память данных
 203h    4567h           203h   08440h
 PREG     0FE079844h       PREG   0FE079844h
```

ПРИМЕР 2

```
SPL    *, AR7 ;(PM=2)
          До выполнения          После выполнения
 ARP     6               ARP     7
 AR6     203h            AR6     203h
 Память данных           Память данных
 203h    4567h           203h   09844h
 PREG     0FE079844h       PREG   0FE079844h
```

Инструкция SPLK

SPLK – сохранить длинный непосредственный операнд.

СИНТАКСИС
SPLK #lk, dma Прямая адресация
SPLK #lk, ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	1	1	0	0	dma						
16-битная константа															

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	1	1	1	0	1	см. подраздел 14.7.3						
16-битная константа															

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7
lk : 16-битная константа

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
lk -> dma

ОПИСАНИЕ SPLK позволяет записать 16-битный операнд в любую ячейку памяти данных.

СЛОВА

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	2	2	2+2p
Внешняя	3+d	3+d	5+d+2p

ПРИМЕР 1

```
SPLK    #7FFFh, DAT3 ; (DP = 6)
        До выполнения          После выполнения
 Память данных           Память данных
 303h    7h                303h    7FFFh
```

ПРИМЕР 2

```
SPLK    #1111h, *+, AR4
         До выполнения          После выполнения
 ARP     0              ARP     4
 AR4      300h          AR4     301h
 Память данных          Память данных
 300h    7h              300h    1111h
```

Инструкция SPM

SPM – записать в поле PM (режим сдвига).

СИНТАКСИС SPM константа

КОД КОМАНДЫ

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	0	0	0	0	0	0	PM	

ОПЕРАНДЫ Константа: значение от 0 до 3 определяет режим сдвига

ВЫПОЛНЕНИЕ (PC) + 1 -> PC
константа -> PM

Влияет на PM.
Не зависит от SXM

ОПИСАНИЕ 2 младших бита инструкции копируются в поле PM регистра статуса ST1. PM управляет режимом сдвига регистра PREG. Сдвигатель может сдвигать регистр PREG на 1 или 4 бита влево, или на 6 бит вправо. Комбинации бит следующие:

PM	Значение
00	без сдвига
01	P << 1
10	P << 4
11	P >> 6 с расширением знака

Левый сдвиг позволяет выравнивать результат. Правый сдвиг на 6 введен для обеспечения до 128 умножений с накоплением без опасности переполнения. PM может быть загружен инструкцией LST #1.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции		
ROM	DARAM	Внешняя память
1	1	1+p

ПРИМЕР

```
SPM     3 ;PREG >> 6 при вводе в АЛУ
```

Инструкция SQRA

SQRA – возвести в квадрат и аккумулировать предыдущий результат.

СИНТАКСИС
SQRA dma Прямая адресация
SQRA ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	1	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	1	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
(ACC) + (сдвинутый регистр PREG) -> ACC
(dma) -> TREG
(dma) * (dma) -> PREG

Зависит от OVM, PM; влияет на OV и C.

ОПИСАНИЕ Значение из памяти загружается в TREG , возводится в квадрат и записывается в регистр P PREG. Предыдущий результат P - регистра, сдвинутый как определено битами PM, прибавляется к ACC.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
SQRA    DAT13 ; (DP = 6, PM = 0)
        До выполнения          После выполнения
Память данных           Память данных
30Dh    0Fh             30Dh    0Fh
TREG    6h              TREG  0Fh
P       12Ch            P      0E1h
ACC     1F4h C=X          ACC     320h C=0
```

ПРИМЕР 2

```
SQRA    *, AR4 ; (PM = 0)
         До выполнения          После выполнения
ARP     3               ARP     4
AR3     30Dh            AR3     30Dh
Память данных           Память данных
30Dh    Fh              30Dh    Fh
TREG    6h              TREG  0Fh
P       12Ch            P      0E1h
ACC     1F4h C=X          ACC     320h C=0
```

Инструкция SQRS

SQRS – вычисление квадрата.

СИНТАКСИС
SQRS dma Прямая адресация
SQRS ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	1	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + -> PC
(ACC) – (сдвинутый регистр PREG) -> ACC
(dma) -> TREG
(dma) * (dma) -> PREG

Зависит от OVM, PM; влияет на OV и C.

ОПИСАНИЕ Значение из памяти загружается в TREG , возводится в квадрат и записывается в регистр PREG. Предыдущий результат P - регистра, сдвинутый как определено битом статуса PM, вычитается из ACC.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
SQRS    DAT13 ; (DP = 6, PM = 0)
          До выполнения          После выполнения
 Память данных           Память данных
 30Dh    08h             30Dh    08h
 TREG   1124h            TREG   08h
 PREG   190h             PREG   40h
 ACC     1450h C=X         ACC     12C0h C=1
```

ПРИМЕР 2

```
SQRS    *, AR4 ; (PM = 0)
          До выполнения          После выполнения
 ARP     3               ARP     4
 AR3     30Dh            AR3     30Dh
 Память данных           Память данных
 30Dh    08h             30Dh    08h
 TREG   1124h            TREG   08h
 PREG   190h             PREG   40h
 ACC     1450h C=X         ACC     12C0h C=1
```

Инструкция SST

SST – сохранить статусный регистр в памяти данных.

СИНТАКСИС
SST #m, dma — Прямая адресация
SST #m, ind [, ARn] — Косвенная адресация

КОД КОМАНДЫ

Прямая адресация for SST#0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	1	0	0	dma						

Косвенная адресация for SST#0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	1	0	1	см. подраздел 14.7.3						

Прямая адресация for SST#1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	1	1	0	dma						

Косвенная адресация for SST#1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	1	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le m \le 1$
$0 \le$ новый ARP ≤ 7

ВЫПОЛНЕНИЕ
(PC) + -> PC
(регистр статуса STn) -> dma

ОПИСАНИЕ Регистр статуса STn пишется в память данных. В прямом режиме адресации STn всегда пишется в страницу 0, независимо от значения DP. Берется заданное в команде смещение от начала этой страницы. Регистр DP не меняется. Это позволяет записать регистр DP в памяти данных при прерывании и т. д. без необходимости изменения DP. В косвенном режиме адресации адрес памяти данных берется из выбранного дополнительного регистра (см. LST). В косвенном режиме адресации может быть адресована любая страница.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	2+d	2+d	4+d+p

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	2n+nd	2n+nd	2n+2+p+nd

ПРИМЕР 1

```
SST #0,96 (Прямая адресация: страница данных
0 с
          ;автоматическим доступом)
         До выполнения          После выполнения
 ST0      0A408h             ST0    0A408h
 Память данных             Память данных
 60h      0Ah                 60h    0A408h
```

ПРИМЕР 2

```
SST #1,*,AR7 (Косвенная адресация)
        До выполнения          После выполнения
 ARP     0                   ARP    7
 AR0     300h                AR0    300h
 ST1     2580h               ST1    2580h
 Память данных             Память данных
 300h    0h                  300h   2580h
```

Инструкция SUB

SUB – вычитание из аккумулятора.

СИНТАКСИС

SUB dma [, shift] Прямая адресация
SUB dma,16 Прямая адресация с левым сдвигом на 16
SUB ind [,shift [, ARn]_] Косвенная адресация
SUB ind,16[, ARn] Косвенная адресация с левым сдвигом на 16
SUB #k Короткая непосредственная адресация
SUB #lk [,shift] Длинная непосредственная адресация

КОД КОМАНДЫ

Прямая адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	SHFT				0	dma						

Косвенная адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	SHFT				1	см. подраздел 14.7.3						

Прямая адресация со сдвигом на 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	1	0	1	0	dma						

Косвенная адресация со сдвигом на 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	1	0	1	1	см. подраздел 14.7.3						

Короткая непосредственная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	0	1	0	8-битная константа							

Длинная непосредственная адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	1	0	1	0	SHFT			
16-битная константа															

ОПЕРАНДЫ

$0 \le dma \le 127$
$0 \le$ новый ARP ≤ 7
$0 \le k \le 255$
$-32768 \le lk \le 32767$
$0 \le$ SHFT t ≤ 15 (по умолчанию 0)

ВЫПОЛНЕНИЕ Прямая и косвенная адресация:
(PC) + 1 -> PC
(ACC) – [(dma) << shift] -> ACC
Зависит от SXM и OVM; влияет на C и OV.

Короткая непосредственная адресация:
(PC) + 1 -> PC
(ACC) – k -> ACC
Зависит от OVM; влияет на C и OV.

Длинная непосредственная адресация:
(PC) + 2 -> PC
(ACC) – lk << SHFT -> ACC
Зависит от OVM; влияет на C и OV.

ОПИСАНИЕ Содержание ячейки памяти данных или непосредственная константа сдвигаются влево и вычитаются из аккумулятора. В течении сдвига младшие разряды бит заполняются нулями. В старших разрядах происходит расширение знака, если SXM = 1, или они заполняются нулями, если SXM = 0. Результат запоминается в аккумуляторе.
Когда используется короткая непосредственная адресация, из аккумулятора вычитается 8-битная константа, сдвиг не может быть задан, вычитание не зависит от SXM и инструкция неповторяема.
Если вычитание генерирует заем, бит переноса устанавливается в 0, в противном случае в 1. Если задан 16-битый сдвиг, инструкция может лишь установить в 1 бит переноса, если вычитание генерирует перенос; в противном случае C не меняется.

СЛОВА 1 (Прямая, косвенная или короткая непосредственная адресация).
2 (Длинная непосредственная адресация).

ЦИКЛЫ

Циклов при однократном выполнении инструкции
(прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT
(прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

Циклов при однократном выполнении инструкции
(короткая непосредственная адресация)

ROM	DARAM	Внешняя память
1	1	1+p

Циклов при однократном выполнении инструкции
(длинная непосредственная адресация)

ROM	DARAM	Внешняя память
2	2	2+2p

ПРИМЕР 1

```
SUB DAT80,1  ;(DP = 8,SXM = 0)
        До выполнения   После выполнения
 Память данных   Память данных
 450h   11h      450h   11h
 ACC     24h  C=X ACC      13h  C=1
```

ПРИМЕР 2

```
SUB *-,1,AR0  ;(SXM = 0)
         До выполнения   После выполнения
 ARP     7       ARP     0
 AR7     0301h   AR7     0300h
 Память данных   Память данных
 301h    4h      301h   42h
 ACC      9h  C=X ACC      01h  C=1
```

ПРИМЕР 3

```
SUB #8h   ; (SXM = 1)
         До выполнения   После выполнения
 ACC     7h  C=X ACC     0FFFFFFFh  C=0
```

Инструкция SUBB

SUBB – вычесть из аккумулятора значение памяти данных и бит C.

СИНТАКСИС
SUBB dma Прямая адресация
SUBB ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	1	0	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	1	0	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le n \le 7$

ВЫПОЛНЕНИЕ Прямая и косвенная адресация:
(PC) + 1 -> PC
(ACC) – (dma) – (логическая инверсия C) -> ACC
Зависит от OVM; не зависит от SXM; влияет на C и OV.

ОПИСАНИЕ Содержание ячейки памяти данных (dma) и логически инвертированный бит C вычитаются из содержимого аккумулятора с подавлением знакового расширения. Результат сохраняется в аккумуляторе. Бит C очищается, если результат вычитания формирует заём, иначе бит C устанавливается в 1.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
SUBB DAT5  ;(DP = 8)
        До выполнения   После выполнения
 Память данных          Память данных
 405h   06h             405h  06h
 ACC     06h  C=0         ACC   FFFF FFFFh  C=0
```

ПРИМЕР 2

```
SUBB *
         До выполнения          После выполнения
 ARP     6              ARP     6
 AR6     301h           AR7     301h
 301h    02h            301h    02h
 ACC      04h  C=1       ACC      02h  C=1
```

Инструкция SUBC

SUBC – условное вычитание из аккумулятора содержимого памяти данных.

СИНТАКСИС
SUBC dma Прямая адресация
SUBC ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	1	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le n \le 7$

ВЫПОЛНЕНИЕ Прямая и косвенная адресация:
(PC) + 1 -> PC
(ACC) – ((dma)x 2^{15}) -> ALU выход

Если выход ALU ≥ 0: (выход ALU)x 2 + 1 -> ACC
Иначе (ACC)x 2 -> ACC

Не зависит от OVM и SXM; влияет на C и OV.

ОПИСАНИЕ Инструкция SUBC выполняет вычитание по условию, которое может быть использовано для деления. Делитель в памяти данных. После завершения последней инструкции SUBC: частное от деления в ACCL и остаток в ACCH.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
SUBC DAT2  ;(DP = 6)
        До выполнения          После выполнения
 Память данных           Память данных
 302h   01h              302h  01h
 ACC     04h  C=X          ACC   08h  C=0
```

ПРИМЕР 2

```
RPT     #15
SUBC    *
         До выполнения          После выполне-
ния
 ARP     3               ARP     3
 AR3     1000h           AR3     1000h
 Память данных           Память данных
 1000h   07h             1000h   07h
 ACC     41h  C=X        ACC     20009h  C=1
```

Инструкция SUBS

SUBS – вычитание из аккумулятора содержимого памяти данных с запрещением расширения знака.

СИНТАКСИС
SUBS dma Прямая адресация
SUBS ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	1	1	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	1	1	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le n \le 7$

ВЫПОЛНЕНИЕ Прямая и косвенная адресация:
(PC) + 1 -> PC
(ACC) – (dma) -> ACC
(dma) как беззнаковое 16-битное число

Зависит от OVM; не зависит от SXM; влияет на C и OV.

ОПИСАНИЕ Содержимое ячейки памяти данных (dma) вычитается из содержимого аккумулятора (ACC) с подавлением знакового расширения. Результат сохраняется в аккумуляторе. Полученное содержимое аккумулятора - знаковое число. Бит C очищается, если вычитание вызывает заём, иначе бит C устанавливается в 1.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
SUBS DAT2  ;(DP = 16, SXM = 1)
        До выполнения            После выполнения
 Память данных          Память данных
 802h   F003h           802h       F003h
 ACC     F105h  C=X     ACC          102h  C=1
```

ПРИМЕР 2

```
SUBS * ;(SXM = 1)
         До выполнения            После выполнения
 ARP     0                ARP     0
 AR0     310h             AR0     310h
 Память данных     Память данных
 310h    F003h            310h    F003h
 ACC      0FFF F105h  C=X ACC      0FFF 0102h
C=1
```

Инструкция SUBT

SUBT – вычитание из аккумулятора содержимого памяти данных со сдвигом, определенным регистром TREG.

СИНТАКСИС
SUBT dma Прямая адресация
SUBT ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	1	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	0	1	1	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ $0 \le dma \le 127$
$0 \le n \le 7$

ВЫПОЛНЕНИЕ Прямая и косвенная адресация:
(PC) + 1 -> PC
(ACC) – ((dma)x $2^{TREG(3-0)}$) -> ACC
(dma) как беззнаковое 16-битное число

Если SXM = 1: (dma) знакорасширенное
Если SXM = 0: (dma) не знакорасширенное

Зависит от OVM, SXM; влияет на C и OV.

ОПИСАНИЕ Содержимое ячейки памяти данных (dma) сдвигается влево от 0 до 15 бит, как определено 4 младшими битами регистра TREG, и вычитается из содержимого аккумулятора (ACC). Результат сохраняется в аккумуляторе. Знаковое расширение в значении dma управляется битом SXM. Бит C очищается, если результат вычитания вызывает заём, иначе бит C устанавливается в 1.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p

	Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1 SUBT DAT127 ;(DP = 4)

До выполнения		После выполнения	
Память данных		Память данных	
2FFh	06h	2FFh	06h
TREG1	08h	TREG1	08h
ACC	FDA5h C=X	ACC	F7A5h C=1

ПРИМЕР 2 SUBT *

До выполнения		После выполнения	
ARP	1	ARP	1
AR1	800h	AR1	800h
Память данных		Память данных	
TREG1	08h	TREG1	08h
ACC	0h C=X	ACC	FFFF FF00h C=0

Инструкция TBLR

TBLR – чтение таблицы.

СИНТАКСИС
TBLR dma Прямая адресация
TBLR ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	1	1	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	1	1	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \leq dma \leq 127$
$0 \leq n \leq 7$

ВЫПОЛНЕНИЕ Прямая и косвенная адресация:
(PC) + 1 -> PC
(PC) -> MSTACK
(ACC(15-0)) -> PC

if (счётчик повторов)≠ 0:
pma, адресуемый через PC) -> dma.
Изменение текущего AR и ARP, как указано
(PC) + 1 -> PC
(счётчик повторов) – 1 -> счётчик повторов.
Else: (pma, адресуемый через PC) -> dma.
Изменение текущего AR и ARP, как указано
(MSTACK) -> PC.

ОПИСАНИЕ Содержимое ячейки памяти программ (pma) пересылается в ячейку памяти данных (dma). Pma определяется содержимым младшего слова аккумулятора (ACCL) и dma определяется из инструкции. Инструкция TBLR в цикле RPT становится одно-цикловой инструкцией с автоинкрементированием pma.

СЛОВА 1

ЦИКЛЫ Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM/ROM Приемник: DARAM	3	3	3+pcode
Источник: Внешняя Приемник: DARAM	3+psrc	3+psrc	3+psrc+pcode
Источник: DARAM/ROM Приемник: Внешняя	4+ddst	4+ddst	6+ddst+pcode
Источник: Внешняя Приемник: Внешняя	4+psrc+ddst	4+psrc+ddst	6+psrc+ddst+pcode

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM/ ROM Приемник: DARAM	n+2	n+2	n+2+pcode
Источник: Внешняя Приемник: DARAM	n+2+npsrc	n+2+npsrc	n+2+npsrc+pcode
Источник: DARAM/ ROM Приемник: Внешняя	2n+2+nddst	2n+2+nddst	2n+4+nddst+pcode
Источник: Внешняя Приемник: Внешняя	4n+npsrc+nddst	4n+npsrc+nddst	4n+2+npsrc+nddst +pcode

ПРИМЕР 1

```
TBLR DAT6  ;(DP = 4)
        До выполнения                    После выпол-
нения
 ACC    23h                      ACC   23h
 Память программ                 Память программ
 23h    306h                     23h   306h
 Память данных                   Память данных
 206h    75h                        206h  306h
```

ПРИМЕР 2

```
TBLR *,AR7
         До выполнения                   После выпол-
нения
 ARP     0                      ARP     7
 AR0     300h                   AR0     300h
 ACC     24h                    ACC     24h
 Память программ                Память программ
 24h     307h                   24h     307h
 Память данных                  Память данных
 300h     75h                     300h    307h
```

Инструкция TBLW

TBLW – запись таблицы.

СИНТАКСИС
TBLW dma Прямая адресация
TBLW ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	1	1	1	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	1	1	1	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \leq dma \leq 127$
$0 \leq n \leq 7$

ВЫПОЛНЕНИЕ Прямая и косвенная адресация:
(PC) + 1 -> PC
(PC) -> MSTACK
(ACC(15-0) -> PC

Если (счётчик повторов)≠ 0:
(pma, адресуемый через PC) -> pma.
Изменение текущего AR и ARP как указано
(PC) + 1 -> PC
(счётчик повторов) – 1 -> счётчик повторов.
Иначе: (dma, адресуемый через PC) -> pma.
Изменение текущего AR и ARP как указано
(MSTACK) -> PC.

ОПИСАНИЕ Содержание ячейки памяти данных (dma) передаётся в память программ (pma). dma определяется инструкцией, pma определяется содержимым младшего байта аккумулятора ACCL). Когда используют с RPT инструкцией, TBLW становится одноцикловой инструкцией с автоинкрементированием pma.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM Приемник: DARAM	3	3	3+pcode
Источник Внешняя Приемник: DARAM	3+dsrc	3+dsrc	3+dsrc+pcode
Источник: DARAM Приемник: Внешняя	4+pdst	4+pdst	5+pdst+pcode
Источник: Внешняя Приемник: Внешняя	4+dsrc+pdst	4+dsrc+pdst	5+dsrc+pdst+pcode
Источник: DARAM Приемник: DARAM	n+2	n+2	n+2+pcode
Источник: Внешняя Приемник: DARAM	n+2+ndsrc	n+2+ndsrc	n+2+ndsrc+pcode

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
Источник: DARAM Приемник: Внешняя	2n+2+npdst	2n+2+npdst	2n+3+npdst+pcode
Источник: Внешняя Приемник: Внешняя	4n+ndsrc+npdst	4n+ndsrc+npdst	4n+1+ndsrc+npdst+pcode

ПРИМЕР 1

```
TBLW DAT5  ;(DP = 32)
        До выполнения   После выполнения
 ACC    257h     ACC    257h
 Память данных   Память данных
 1905h  4339h    1905h  4339h
 Память программ Память программ
 257h    306h     257h   4399h
```

ПРИМЕР 2

```
TBLW *
        До выполнения   После выполнения
 ARP     6       ARP     6
 AR6     1006h   AR6     1006h
 ACC     258h    ACC     258h
 Память данных   Память данных
 1006h   4340h   1006h   4340h
 Память программ Память программ
 258h    307h    258h    4340h
```

Инструкция TRAP

TRAP – программное прерывание.

СИНТАКСИС TRAP

КОД КОМАНДЫ

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

ОПЕРАНДЫ Нет

ВЫПОЛНЕНИЕ (PC) + 1 -> stack
22h -> PC

Не зависит от INTM; не влияет на INTM.

ОПИСАНИЕ TRAP вызывает последовательность инструкций программы, расположенной в ячейке 22h. Текущий программный счётчик (PC) инкрементируется и сохраняется в стеке. Адрес 22h загружается в программный счётчик. По адресу 22h может находиться инструкция перехода к управлению определённым TRAP. Инструкция TRAP не маскируется.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции		
ROM	DARAM	Внешняя память
4	4	4+3p

ПРИМЕР

```
TRAP ;контроль прохождения ячейки 22h памяти
     ;программ и задвигание PC + 1 в стэк.
```

Инструкция XOR

XOR – «ИСКЛЮЧАЮЩЕЕ «ИЛИ» аккумулятора с содержимым памяти данных.

СИНТАКСИС

XOR dma	Прямая адресация
XOR ind [, ARn]	Косвенная адресация
XOR #lk, [, shift]	Длинная непосредственная адресация
XOR #lk,16	Длинная непосредственная адресация с левым сдвигом на 16

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	0	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	0	0	1	см. подраздел 14.7.3						

Длинная непосредственная адресация со сдвигом

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	1	1	0	1	SHFT			
16-битная константа															

Длинная непосредственная адресация со сдвигом на 16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	0	1	0	0	0	0	0	1	1
16-битная константа															

ОПЕРАНДЫ

$0 \leq dma \leq 127$
$0 \leq n \leq 7$
lk: 16-битная константа
$0 \leq$ сдвиг ≤ 15

ВЫПОЛНЕНИЕ

Прямая и косвенная адресация:
(PC) + 1 -> PC
(ACC(15-0)) XOR (dma) -> ACC(15-0)
(ACC(31-16)) -> ACC(31-16)

Длинная непосредственная константа со сдвигом:
(PC) + 2 -> PC
(ACC(31-0)) XOR (lk x 2^{shift}) -> ACC(31-0)

Не влияет на C; не зависит от SXM (длинная непосредственная константа).

ОПИСАНИЕ Если длинная непосредственная константа определена, то константа сдвигается влево, как определено сдвиговым кодом, дополняется до 32 бит 0 в старшие и младшие биты и выполняется XOR с содержимым аккумулятора (ACC). Результат сохраняется в аккумуляторе. Если константа не определена, выполняется операция XOR содержимого ячейки памяти данных (dma) с содержимым младшего слова аккумулятора (ACCL). Результат сохраняется в ACCL и содержимое старшего слова аккумулятора (ACCH) не меняется.

СЛОВА 1 (прямая и косвенная адресация)
2 (длинная непосредственная адресация)

ЦИКЛЫ Циклов при многократном выполнении инструкции под RPT (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT (прямая или косвенная адресация)

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

Циклов при однократном выполнении инструкции

ROM	DARAM	Внешняя память
2	2	2+2p

ПРИМЕР 1
```
XOR DAT127  ;(DP = 51
          До выполнения            После выполнения
 Память данных              Память данных
 FFFFh   F0F0h              FFFFh  F0F0h
 ACC      1234 5678h  C=X  ACC     1234 A688h
C=X
```

ПРИМЕР 2
```
XOR *+,AR0
          До выполнения            После выполнения
 ARP     7                  ARP     0
 AR7     300h               AR7     301h
 Память данных              Память данных
 300h    FFFFh              300h    FFFFh
 ACC      1234 F0F0h  C=X ACC      1234 0F0Fh
C=X
```

ПРИМЕР 3
```
XOR #0F0F0h,4
 ACC     1111 1010h  C=X ACC     111E 1F10h  C=X
```

Инструкция ZALR

ZALR - обнуление старшего слова и загрузка младшего слова аккумулятора из памяти данных с округлением.

СИНТАКСИС
ZALR dma Прямая адресация
ZALR ind [, ARn] Косвенная адресация

КОД КОМАНДЫ

Прямая адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	0	0	dma						

Косвенная адресация

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	0	1	см. подраздел 14.7.3						

ОПЕРАНДЫ
$0 \le dma \le 127$
$0 \le n \le 7$

ВЫПОЛНЕНИЕ
(PC) + 1 -> PC
8000h -> ACC(15-0)
(dma) -> ACC(31-16)
Не влияет на C

ОПИСАНИЕ Содержимое ячейки памяти данных (dma) загружается в старший байт аккумулятора. 15 младших бит (биты 0-14) аккумулятора устанавливаются в 0, бит 15 ACCL устанавливается в 1.

СЛОВА 1

ЦИКЛЫ

Циклов при однократном выполнении инструкции

Операнд	ROM	DARAM	Внешняя память
DARAM	1	1	1+p
Внешняя	1+d	1+d	2+d+p

Циклов при многократном выполнении инструкции под RPT

Операнд	ROM	DARAM	Внешняя память
DARAM	n	n	n+p
Внешняя	n+nd	n+nd	n+1+p+nd

ПРИМЕР 1

```
ZALR DAT3 ;(DP = 32)
 До выполнения                    После выполнения
 Память данных            Память данных
 1003h   3F01h            1003h  3F01h
 ACC      0077 FFFFh  C=X  ACC    3F01 8000h
C=X
```

ПРИМЕР 2

```
ZALR *-,AR4
 До выполнения                    После выполнения
 ARP    7h                ARP    4h
 AR7    FF00h             AR7    FEFFh
 Память данных            Память данных
 FF00h  E0E0h             FF00h  E0E0h
 ACC     0010 7777h  C=X   ACC    E0E0 8000h
C=X
```

15 Блок внутрикристальной памяти ЭСППЗУ

15.1 Назначение

Блок внутрикристальной памяти ЭСППЗУ обеспечивает микроконтроллер энергонезависимым (flash) типом памяти программ с возможностью перепрограммирования.

15.2 Состав

Блок внутрикристальной памяти ЭСППЗУ состоит из
- модуля FLASH памяти, объёмом 64K слов ×16 бит;
- модуля «теневого» ОЗУ, объёмом 64K слов ×16 бит;
- модуля управления копированием данных;
- модуля программирования и тестирования FLASH памяти.

Модуль FLASH памяти состоит из восьми банков по 8K слов ×16 бит.

Модуль FLASH памяти обеспечивает 20000 циклов перезаписи (минимум), сохранность данных более 10 лет, время доступа 32 нс (максимум).

Модуль «теневого» ОЗУ является скоростным буфером между ЦПУ микроконтроллера и FLASH памятью и обеспечивает ЦПУ доступ на чтение к данным из FLASH памяти на полной скорости без состояний ожидания. Состоит из восьми банков SARAM по 8K слов ×16 бит.

Модуль управления копированием данных осуществляет копирование информации из FLASH памяти в «теневое» ОЗУ.

Модуль программирования и тестирования FLASH памяти предназначен для тестирования и изменения содержимого FLASH памяти (запись, стирание) с помощью внешней специальной программы через стандартный IEEE 1149.1 (JTAG) порт микроконтроллера.

15.3 Функционирование

После окончания сигнала сброса PORESET# модуль управления копированием данных осуществляет перенос всего содержимого FLASH памяти в «теневое» ОЗУ. Копирование производится восемью параллельными потоками на частоте внешнего тактового сигнала. Во время копирования микроконтроллер удерживается в состоянии «сброс» сигналом от модуля управления копированием. После завершения копирования модуль управления снимает внутренний сигнал «сброс» микроконтроллера и переводит FLASH память в режим низкого энергопотребления, а «теневое» ОЗУ - в режим «только чтение», имитируя ПЗУ.

ЦПУ микроконтроллера после процедуры сброса может обращаться к копии содержимого FLASH памяти в «теневом» ОЗУ только для чтения данных и только при низком уровне на выводе MP/MC# (режим «мирокомпьютер»).

16 Модуль последовательного интерфейса связи CAN

16.1 Ведение в протокол CAN

Controller Area Network (CAN) – последовательный интерфейс связи (далее – модуль CAN), эффективно поддерживающий распределенное управление в реальном масштабе времени с высокой помехозащищенностью, показан на рисунке 273. Протокол связи полностью определен Robert Bosch GmbH в спецификациях CAN 1.2, CAN 2.0A, CAN 2.0B (активная и пассивная версии).

Протокол CAN оптимизирован для систем, в которых должно передаваться относительно небольшое количество информации (по сравнению с Ethernet или USB) к любому или всем узлам сети. Множественный доступ с опросом состояния шины позволяет каждому узлу получить доступ к шине с учетом приоритетов. Неадресатная структура сообщений позволяет организовать многоабонентскую доставку данных с сокращением трафика шины. Быстрая устойчивая передача информации, с системой контроля ошибок, позволяет отключать неисправные узлы от шины, что гарантирует доставку критических по времени сообщений.

Область применения CAN протокола: от высокоскоростных сетей связи до электропроводов в автомобиле. Высокая скорость передачи данных до 1 Мбит/с, высокая помехозащищенность протокола, защита от неисправности узлов делают шину CAN подходящей для индустриальных приложений управления типа DeviceNet.

CAN имеет асинхронную последовательную структуру шины с одним логическим сегментом сети. CAN сеть может состоять из двух или более узлов, с возможностью подключения/отключения узлов от шины без перенастройки других устройств.

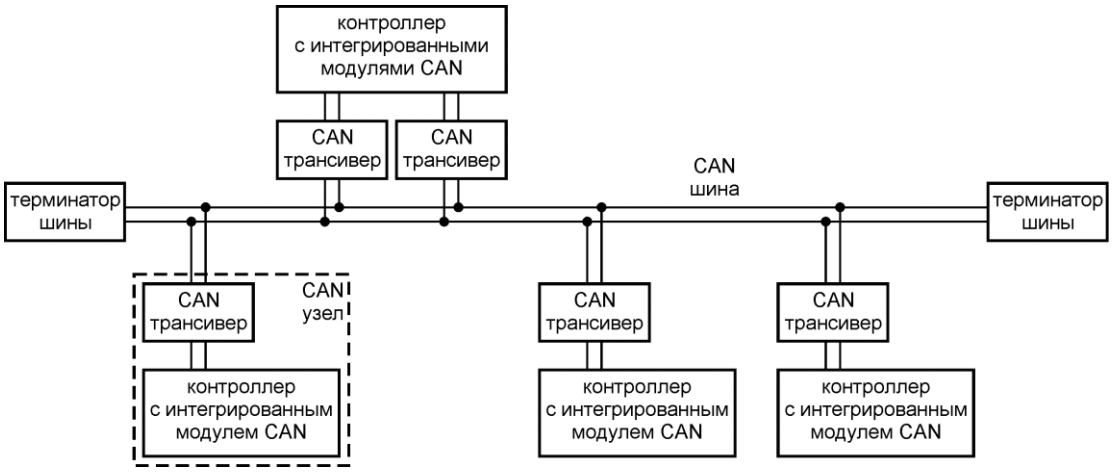


Рисунок 273 – Общая структура CAN сети

Логика шины работает по механизму «монтажное И», в котором рецессивный (recessive) бит соответствует логической единице «1», а доминантный (dominant) – логическому нулю «0». Пока ни один узел не формирует доминантный бит, шина находится в рецессивном состоянии. Появление на шине доминантного бита (выставленного одним или несколькими узлами) создает доминантное состояние шины. Отсюда следует, что при выборе среды передачи данных необхо-

димо точно определить, какое состояние будет доминантным, а какое – рецессивным. Одним из наиболее распространенных и «дешевых» вариантов является пара скрученных проводов. Линии шины тогда называются CANH и CANL и могут быть подключены непосредственно к устройствам. Не существует никакого дополнительного стандарта на среду передачи данных.

При использовании в качестве линии связи пары скрученных проводов с нагрузочными резисторами на концах можно получить максимальную скорость передачи данных 1 Мбит/с при длине линии до 40 м. Для линий связи протяженностью более 40 м необходимо снизить скорость передачи данных (для линии в 1 000 м скорость шины должна быть не более 40 Кбит/с). Из-за дифференциального характера линии связи шина CAN мало чувствительна к электромагнитным помехам. Экранирование шины значительно снизит воздействие внешнего электромагнитного поля, что особенно важно для высокоскоростных режимов работы.

Двоичная информация кодируется, при этом доминантным является низкий уровень, рецессивным – высокий уровень. Для гарантированной синхронизации данных всеми узлами шины используется наполняющий бит. При последовательной передаче пяти бит одинаковой полярности передатчик вставляет один дополнительный бит противоположной полярности перед передачей остальных битов. Приемник также проверяет полярность и удаляет дополнительные биты.

В CAN протоколе при передаче данных приемные узлы не адресуются, а указывается идентификатор передатчика. С помощью идентификатора указывается содержание сообщения (например, применительно автомобиля – обороты, температура двигателя и т. д.). Идентификатор дополнительно указывает приоритет сообщения. Более высокий приоритет у идентификатора, имеющего меньшее бинарное значение.

При коллективном доступе к шине используется неразрушающий арбитраж с опросом состояния шины. Перед началом передачи данных узел проверяет состояние шины (отсутствие активности на шине). При начале передачи сообщения узел становится управляющим шины, все остальные узлы переходят в режим приема. Каждый узел выдает подтверждение приема, проверяет идентификатор сообщения, обрабатывает или удаляет принятые данные. Если два или более узлов начинают передачу данных одновременно, поразрядный арбитраж позволяет избежать конфликта на шине. Каждый узел выдает на шину свой идентификатор (старший бит формируется первым) и контролирует ее состояние. Если узел посылает «1», а читает «0», то арбитраж потерян и узел переключается в режим приема. Это происходит тогда, когда идентификатор конкурирующего узла имеет меньшее бинарное значение. Таким образом, узел с высоким приоритетом выигрывает арбитраж, без необходимости повторять сообщение. Все остальные узлы будут пытаться передать сообщение после освобождения шины. Данный механизм не позволяет передавать одновременно сообщения с одинаковым идентификатором, избегая, таким образом, дальнейших ошибок.

Оригинальная спецификация в версии CAN 2.0B (так называемая расширенная версия CAN) определяет возможность идентификатора иметь длину 11 или 29 бит.

16.2 Функциональные возможности модуля CAN

Модуль CAN выполняет все функции для приема и передачи сообщений на шине. Данные, предназначенные для передачи, записываются в соответствующие регистры. Состояние модуля CAN и возникшие ошибки проверяются чтением регистров состояния. Любое сообщение, передаваемое по шине, проверяется на наличие ошибок, согласуется по фильтрам и сохраняется в регистрах приемного буфера при выполнении всех условий.

CAN модуль поддерживает следующие типы сообщений:

- сообщения данных;
- удаленный запрос данных;
- сообщение об ошибке;
- сообщение о перезагрузке.

Существует два типа сообщений данных:

- стандартное сообщение;
- расширенное сообщение.

16.2.1 Стандартное сообщение данных

Стандартное сообщение данных формируется, когда узел желает передать данные, (показано на рисунке 274).

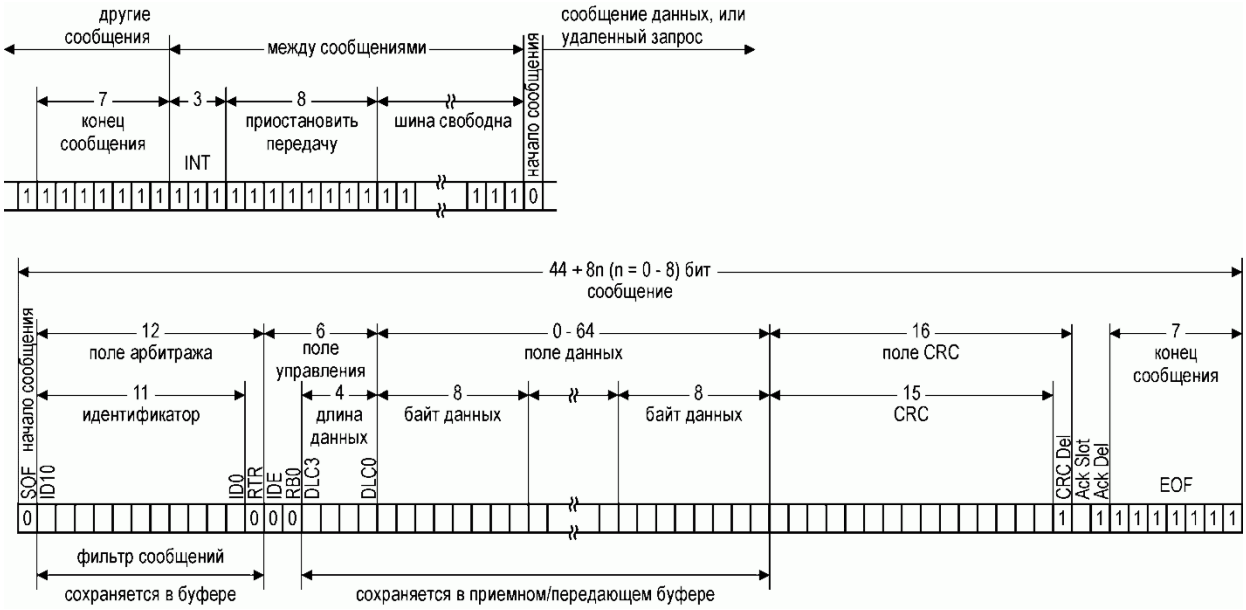


Рисунок 274 – Стандартное сообщение данных

Стандартное сообщение данных имеет в своем составе:

- бит SOF – доминантный «0» бит начала сообщения для жесткой синхронизации всех узлов;
- поле арбитража 12 бит, включающее:
  - поле ID идентификатора 11 бит;
- бит RTR передачи по удаленному запросу (RTR = 0_B соответствует сообщению данных, RTR = 1_B соответствует удаленному запросу);

- поле управления 6 бит, включающее:

- бит IDE указатель расширенного идентификатора (IDE = 0_B соответствует стандартному идентификатору, IDE = 1_B соответствует расширенному идентификатору);

- бит RB0 резервный доминантный бит;

- поле DLC числа байт данных 4 бита, которое указывает, сколько байт данных содержится в сообщении (допустимые значения – от 0 до 8, другие значения использоваться не могут);

- поле данных от 0 до 64 бит, содержащее целое число байт данных;

- поле контрольной суммы CRC 16 бит, включающее:

- поле CRC 15 бит, используемое для обнаружения возможных ошибок передачи данных;

- бит CRC Del рецессивный разделитель CRC;

- поле подтверждения 2 бита, включающее:

- бит ACK Slot подтверждения передачи, передающий узел выдает рецессивный бит, а любой узел, который принял сообщение без ошибок, заменяет его сформированным доминантным битом;

- бит ACK Del рецессивный разделитель подтверждения;

- поле EOF конца сообщения 7 бит.

Между передачами двух любых сообщений шина должна оставаться в рецессивном состоянии, как минимум, в течение времени появления 3 битов (поле INT простоя). Если после появления трех рецессивных битов (поля INT) ни один узел не начал передачу, шина переходит в состояние ожидания (режим Idle) и находится в рецессивном состоянии до появления доминантного бита сообщения.

16.2.2 Расширенное сообщение данных

Расширенное сообщение данных формируется, когда узел желает передать данные, показан на рисунке 275.

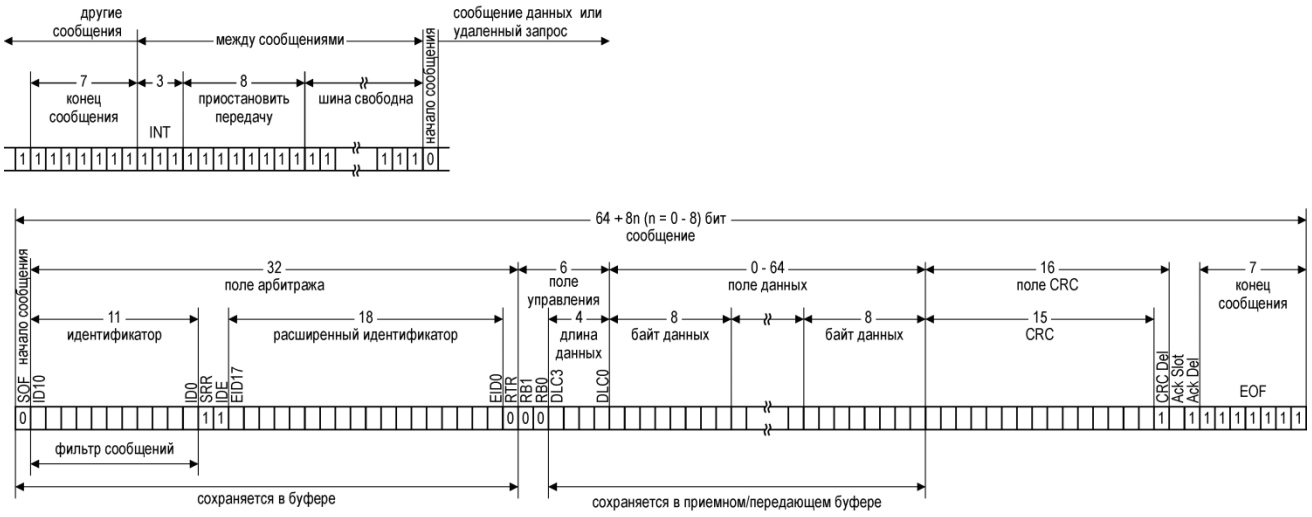


Рисунок 275 – Расширенное сообщение данных

Расширенное сообщение данных имеет в своем составе:

- бит SOF – доминантный «0» бит начала сообщения для жесткой синхронизации всех узлов;
- поле арбитража 38 бит, включающее:
  - поле стандартного идентификатора 11 бит;
  - бит SRR заменитель удаленного запроса;
  - бит IDE указатель расширенного идентификатора (рецессивный, что соответствует расширенному идентификатору);
  - поле расширенного идентификатора 18 бит;
  - бит RTR передачи по удаленному запросу (RTR = 0_B соответствует сообщению данных, RTR = 1_2 соответствует удаленному запросу);
- поле управления 6 бит, включающее:
  - бит RB0 резервный доминантный бит;
  - бит RB1 резервный доминантный бит;
- поле DLC числа байт данных 4 бита, которое указывает, сколько байт данных содержится в сообщении (допустимые значения – от 0 до 8, другие значения использоваться не могут);
- поле данных от 0 до 64 бит, содержащее целое число байт данных;
- поле контрольной суммы CRC 16 бит, включающее:
  - поле CRC 15 бит используемое для обнаружения возможных ошибок передачи данных;
  - бит CRC Del рецессивный разделитель CRC;
- поле подтверждения 2 бита, включающее:
  - бит ACK Slot подтверждения передачи (передающий узел выдает рецессивный бит, а любой узел, который принял сообщение без ошибок, заменяет его сформированным доминантным битом);
  - бит ACK Del рецессивный разделитель подтверждения;
- поле EOF конца сообщения 7 бит.

16.2.3 Удаленный запрос данных

Удаленный запрос данных формируется, когда узлу требуются данные другого узла. Узел назначения посылает удаленный запрос с идентификатором источника. Соответствующий узел источника, распознавший свой идентификатор, посылает стандартное или расширенное сообщение в ответ на запрос. Удаленный запрос данных существует в стандартном и расширенном вариантах. На рисунке 276 представлен вариант удаленного запроса со стандартным идентификатором. Имеются только два отличия содержимого удаленного запроса от сообщения данных:
- бит RTR в удаленном запросе передается в рецессивном состоянии;
- поле данных отсутствует (в сообщении не передается никаких данных, значение в поле DLC любое в пределах от 0 до 8).

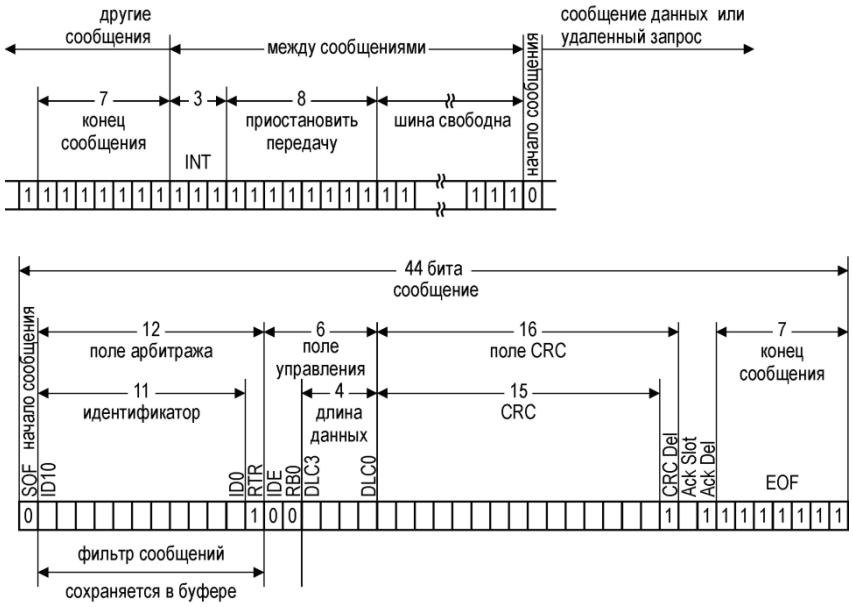


Рисунок 276 – Удаленный запрос данных (стандартный формат)

В самом маловероятном случае, когда одновременно формируется удаленный запрос и устройство пытается передать данные с одинаковыми идентификаторами, арбитраж будет выигран устройством, передающим данные, из-за доминантного состояния бита RTR.

Узел, который посылал запрос, получает данные немедленно.

16.2.4 Сообщение об ошибке

Формируется любым узлом, который обнаруживает ошибку на шине.

Сообщение об ошибке может формироваться как активным, так и пассивным узлом.

Если ошибку обнаружил активный узел, представленный на рисунке 277, тогда он прерывает передачу текущего сообщения, формируя флаг активной ошибки. Флаг активной ошибки состоит из 6 последовательных доминантных битов, которые нарушают правила бит-стаффинга (правила наполнения и передачи битов на шине). Остальные узлы также обнаруживают ошибку и начинают формировать сообщение об ошибке. Таким образом, поле флага ошибки может содержать от 6 до 12 доминантных битов. Ошибка сопровождается разделителем ошибки, состоящим из восьми рецессивных битов и позволяющим перезапускать связь с шиной.

После перехода шины в нормальное состояние узлы возобновляют передачу данных.

Если ошибку обнаружил пассивный узел, тогда он формирует флаг пассивной ошибки, состоящий из шести последовательных рецессивных битов, и затем разделитель ошибки. Таким образом, сообщение о пассивной ошибке состоит из 14 рецессивных битов. Это не нарушает правила бит-стаффинга на шине и не оказывает влияния на передачи других узлов. Исключение составляет узел, который передает данные узлу, обнаружившему ошибку. В этом случае правила бит-стаффинга нарушаются, и передача данных прекращается. После передачи пассив-

ной ошибки, узел должен ожидать шесть последовательных рецессивных битов для восстановления связи с шиной.

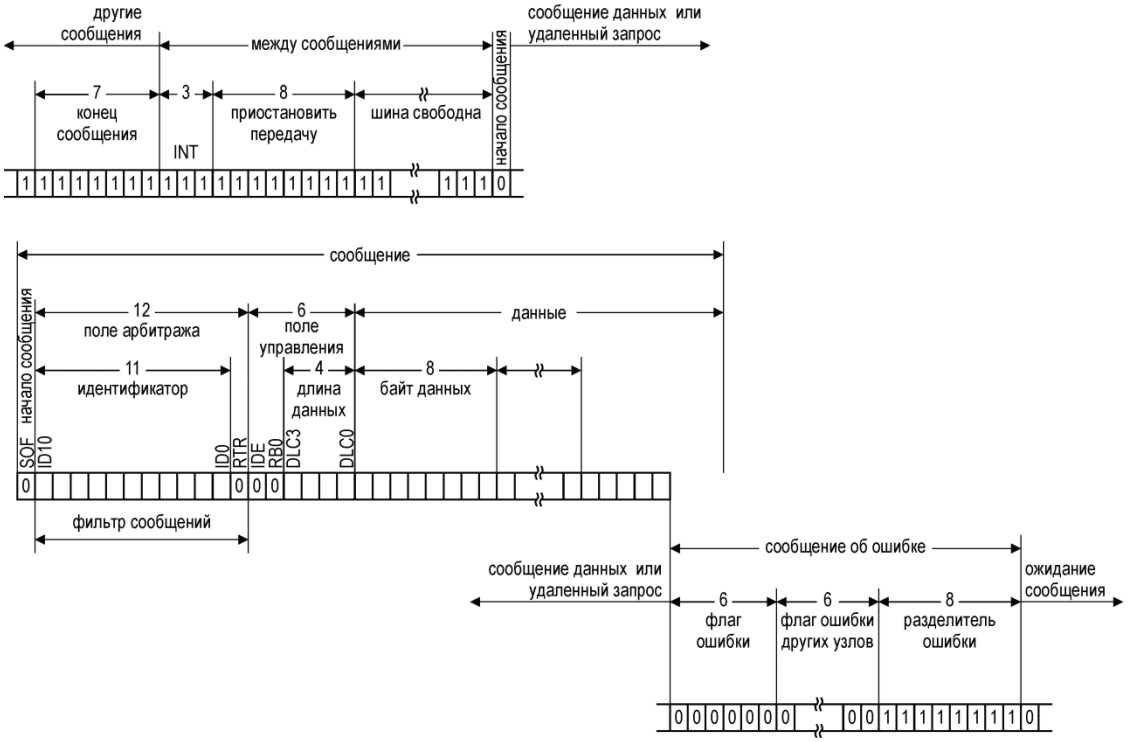


Рисунок 277 – Сообщение об активной ошибке

16.2.5 Сообщение о перезагрузке

Формат сообщения о перезагрузке, представленный на рисунке 278, аналогичен формату сообщения об ошибке, но может быть сформирован только, когда шина простаивает.

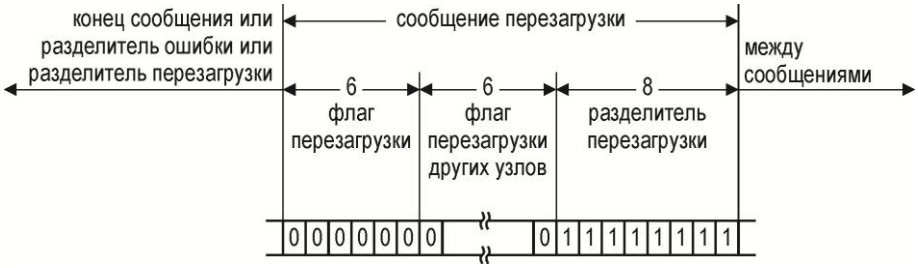


Рисунок 278 – Сообщение о перезагрузке

Флаг перезагрузки состоит из 6 последовательных доминантных битов. Другие узлы обнаруживают перезагрузку и начинают формировать ее самостоятельно. Поэтому на шине, во время выполнения перезагрузки, может быть до12 доминантных битов.

Разделитель перезагрузки состоит из 8 последовательных рецессивных битов.

Узел может сформировать сообщение о перезагрузке в двух случаях:

- между сообщениями обнаружен доминантный бит, что является ненормальным во время простоя шины;

- для задержки передачи нового сообщения.

Узел может последовательно сформировать не более двух сообщений перезагрузки.

16.2.6 Простой шины

Между сообщениями шина находится в рецессивном состоянии. Для выполнения условий «простой шины» необходимо, чтобы было получено, как минимум, 3 рецессивных бита после завершения передачи/приема очередного сообщения.

16.3 ISO/OSI модель

ISO/OSI - эталонная модель - представлена на рисунке 279. Она используется для того, чтобы определить уровни протокола системы связи. На самом верхнем уровне модели находятся приложения, которые должны связываться друг с другом. На самом низком уровне модели расположена физическая среда, обеспечивающая электрическую сигнализацию связи.

Верхние уровни сетевой модели CAN реализуются программно. В технических требованиях к CAN протоколу нет никаких дополнительных требований к содержанию сообщений.

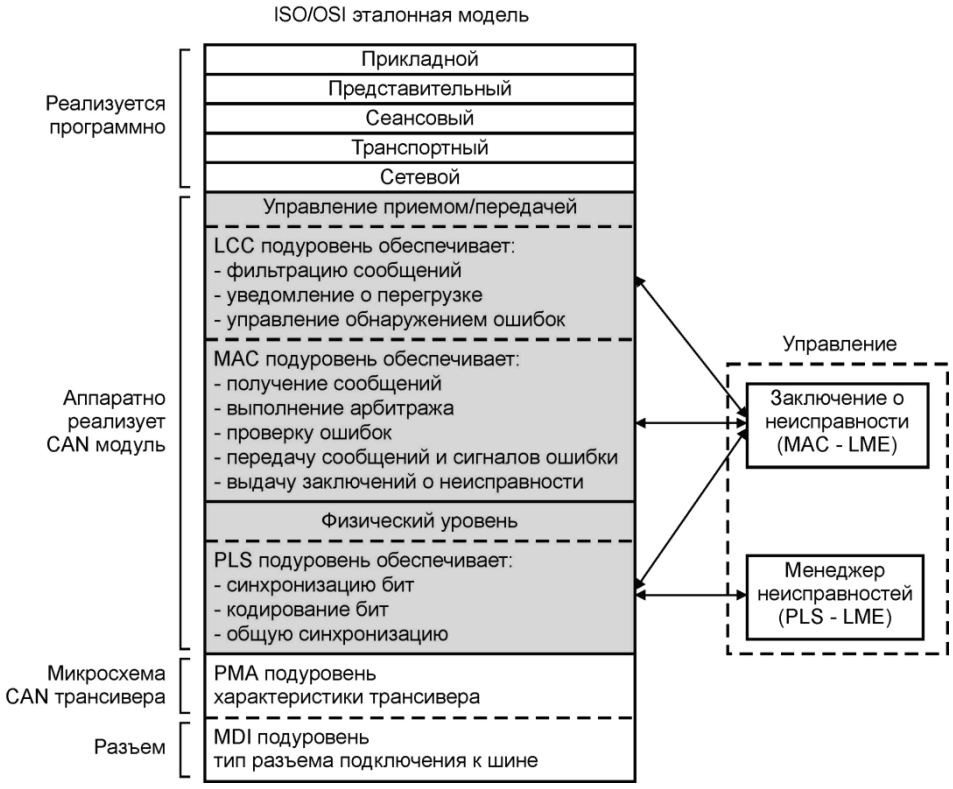


Рисунок 279 – ISO/OSI модель

Полнофункциональный CAN модуль поддерживает два нижних уровня эталонной модели сети:

- управление приемом/передачей данных:
- управление логической связью – LCC подуровень,
- среднее управление доступом – MAC подуровень;
- физический уровень: физическая сигнализация – PLS подуровень.

LCC подуровень обеспечивает:

- фильтрацию сообщений;
- оповещение о перегрузке;
- управление обнаружением ошибок.

MAC подуровень обеспечивает управление средой передачи:

- получение сообщений;
- выполнение арбитража;
- проверку ошибок;
- передачу сообщений и сигналов ошибки;
- выдачу заключений о неисправности.

MAC подуровень получает сообщения от LCC для передачи по шине и передает сообщения, принятые с шины в LCC подуровень. В пределах подуровня MAC определяется отсутствие активности на шине CAN для начала передачи сообщения. Заключение о неисправности является механизмом самоконтроля при постоянном возникновении кратковременных ошибок.

Физический уровень определяет фактическую передачу бит между различными устройствами с выполнением всех электрических требований.

PLS подуровень определяет физическую структуру сигнала и контролирует:

- синхронизацию бит;
- кодирование бит;
- общую синхронизацию.

Низшие уровни протокола определяют фактический тип интерфейса связи: витая пара, волоконный световод, т. д. В пределах одной сети физический уровень должен быть одинаков для всех узлов. Характеристики драйверов физического уровня не определены в спецификации на CAN протокол, чтобы позволить оптимизировать среду передачи для конкретных применений.

16.4 Модуль интерфейса CAN

В микроконтроллере 1867ВЦ10Т находится модуль интерфейса CAN, который представлен на рисунке 280, включает в себя два идентичных независимых блока CAN (в дальнейшем CAN узел 0 и CAN узел 1), с общим ОЗУ, между которыми есть отличие лишь в подключении к выводам.

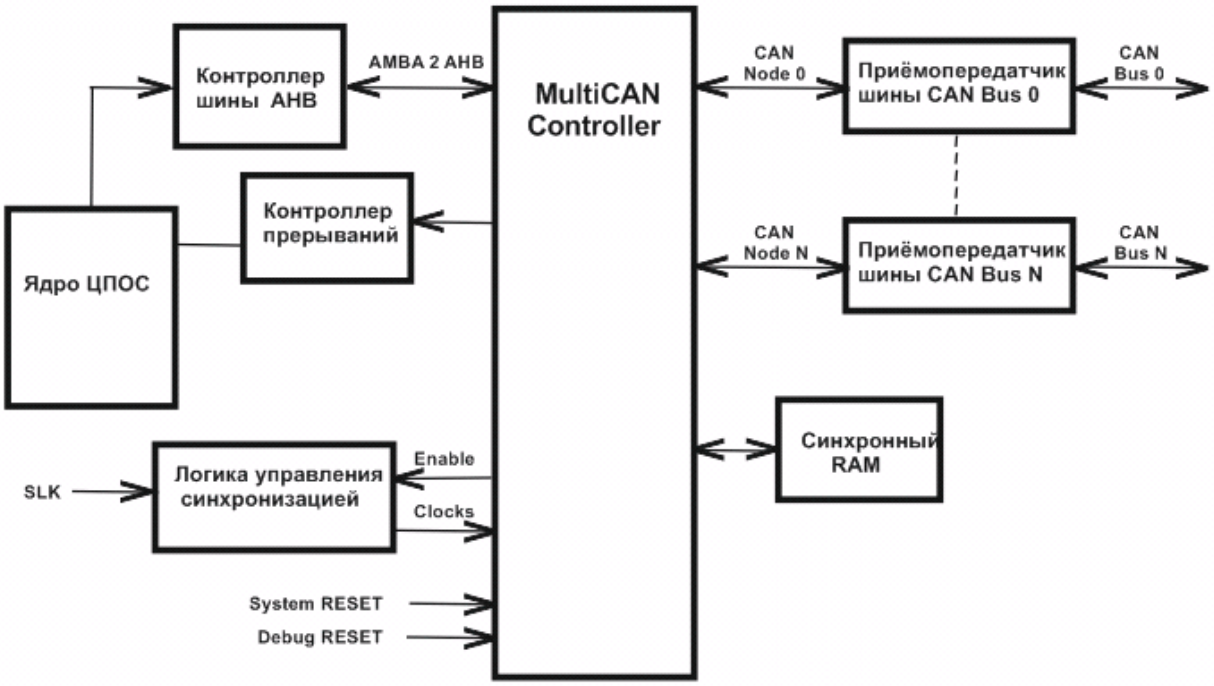
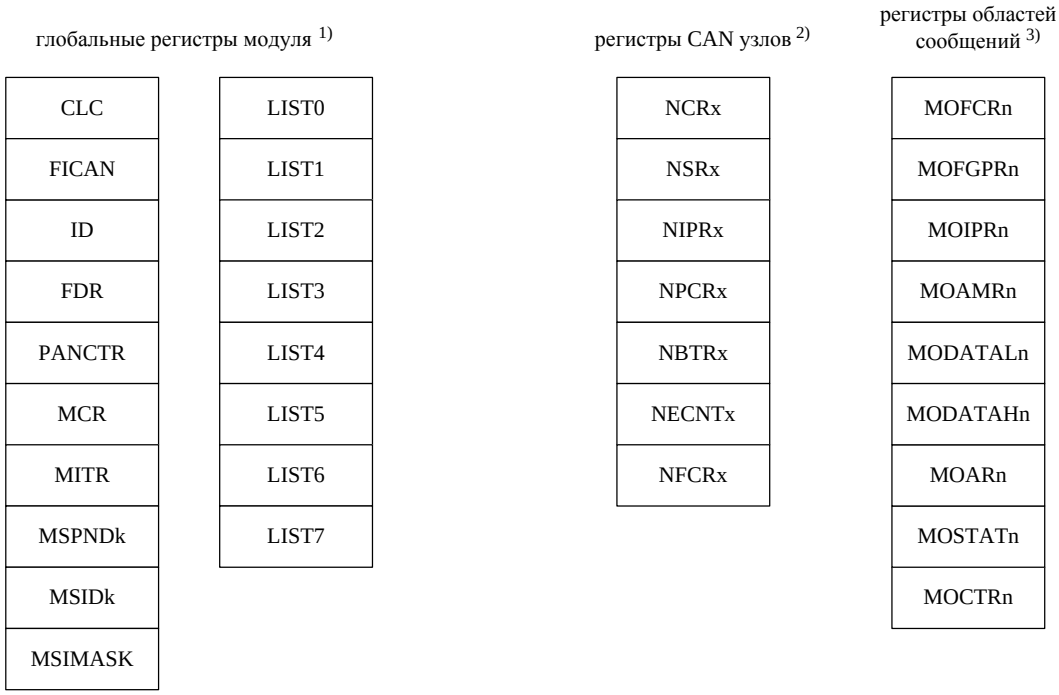


Рисунок 280 – Модуль интерфейса CAN

Помимо стандартной передачи сообщений, когда каждый узел передает на свои выходы, в контроллере реализовано логическое «И» передач CAN узлов (вывод результата сложения сигналов передачи обоих узлов на один выход). Список регистров модуля CAN представлен на рисунке 281.



Пояснения:

CLC – регистр управления частотой;
FICAN – регистр флагов прерываний;
ID – регистр идентификации;
FDR – регистр делителя;
PANCTR – регистр панели команд;
MCR – регистр управления;
MITR – регистр прерываний;
MSPNDk – регистр k ждущих прерываний;
MSIDk – регистр k индекса сообщения;
MSIMASK – регистр маски индекса сообщения узла x.

LIST0 – регистр списка №0 нераспределенных областей сообщений;
LIST1 – регистр списка №1 узла 0;
LIST2 – регистр списка №2 узла 1;
LIST3 – регистр списка №3 узла 2 (если реализован);
LIST4 – регистр списка №4 узла 3 (если реализован);
LIST5 – регистр списка №5 узла 4 (если реализован);
LIST6 – регистр списка №6 узла 5 (если реализован);
LIST7 – регистр свободного списка №7.

NCRx – регистр управления узла x;
NSRx – регистр состояния узла x;
NIPRx – регистр указателя прерываний узла x;
NPCRx – регистр управления портом узла x;
NBTRx – регистр синхронизации битов узла x;
NECNTx – регистр счетчика ошибок узла x;
NFCRx – регистр счетчика сообщений узла x;

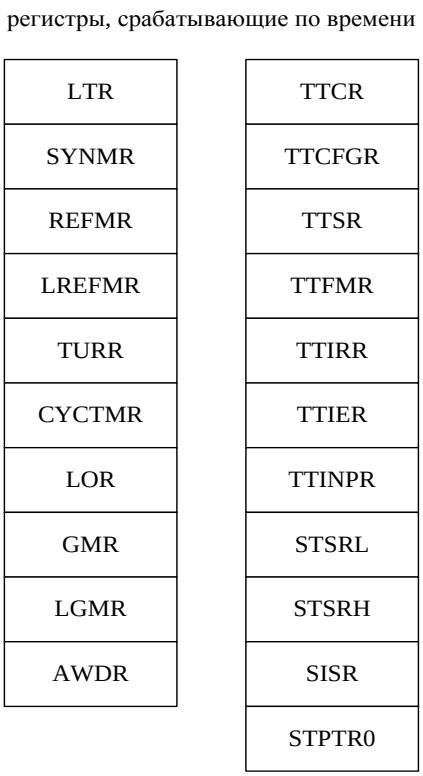
MOFCRn – регистр управления функционированием области сообщения n;
MOFGPRn – регистр указателя FIFO/шлюза области сообщения n;
MOIPRn – регистр указателя прерываний области сообщения n;
MOAMRn – регистр маски области сообщения n;
MODATALn – регистр (младший) данных области сообщения n;
MODATAHn – регистр (старший) данных области сообщения n;
MOARn – регистр арбитража области сообщения n;
MOSTATn – регистр состояния области сообщения n;
MOCTRn – регистр управления областью сообщения n.

---

1) k – номер глобального регистра, k = 0 … 7.

2) x – номер CAN узла равен 0 или 1.

3) n – номер области сообщения, n = 0 … 255.



LTR – регистр локального времени;
SYNMR – регистр синхронизации отметки;
REFMR – регистр ссылки отметки;
LREFMR – регистр последней ссылки отметки;
TURR – регистр коэффициента единицы времени;
CYCTMR – регистр длительности цикла;
LOR – регистр локального сдвига;
GMR – регистр глобальной отметки;
LGMR – регистр последней глобальной отметки;
AWDR – регистр использования сторожевого таймера;
TTCR – регистр управления, срабатывающий по времени;
TTCFGR – регистр конфигурации, срабатывающий по времени;
TTSR – регистр состояния, срабатывающий по времени;
TTFMR – регистр модификации флага, срабатывающий по времени;
TTIRR – регистр запроса прерываний, срабатывающий по времени;
TTIER – регистр разрешения прерываний, срабатывающий по времени;
TTINPR – регистр прерываний в указанном узле, срабатывающий по времени;
STSRL – регистр (младший) состояния запланированного времени;
STSRH – регистр (старший) состояния запланированного времени;
SISR – регистр состояния запланированных инструкций;
STPTR0 – регистр запланированного указанного старта в узле 0.

Рисунок 281 – Регистры модуля CAN

Характеристики модуля CAN:

- соответствие ISO 11898;
- функционирование согласно спецификации CAN 2.0B активная версия;
- управляющие регистры для каждого CAN узла;
- программируемая скорость передачи информации до 1 Мбит/с;
- гибкий и полный контроль передачи сообщений и обработки ошибок;
- 32 области сообщений (message object);
- полнофункциональная реализация CAN, в которой любая область сообщения может быть:
- выделена для любого из узлов;
- сконфигурирована как область «для передачи» или область «для приема»;
- сконфигурирована с 11- или 29-битным идентификатором;
- сконфигурирована для удаленных запросов;
- расширенная фильтрация входящих сообщений:
- каждая область сообщения имеет свою индивидуальную маску для фильтрации принимаемых сообщений;
- каждая область сообщения может быть сконфигурирована как для приема только стандартного или только расширенного сообщения, так и для приема сообщений обоих типов;
- области сообщений могут быть объединены в классы, каждый из которых может иметь один из четырех уровней приоритета для приема и передачи;
- распределение приоритета сообщений при передаче согласно правилам арбитража (см. спецификацию CAN 2.0B) или согласно их позициям в списке;
- расширенные возможности областей сообщений:
- области сообщений могут быть объединены для построения FIFO структур с произвольными размерами, ограниченными только количеством областей сообщений;
- области сообщений CAN узлов могут соединяться попарно с образованием так называемого шлюза, что позволяет автоматическую передачу сообщений между CAN шинами. Между CAN узлами может быть сформировано произвольное число шлюзов;
- расширенное управление данными:
- организация областей сообщений в двухсвязные списки;
- переорганизация списков возможна в любой момент времени, также во время функционирования CAN узлов;
- управляемый «Контроллер списка» отвечает за организацию структуры списка и отслеживает его наполнение;
- FIFO области сообщений организуются в списки, размеры которых в любой момент могут быть изменены;
- жестко распределенные команды делают модуль CAN совместимым с TwinCAN устройствами, которые не имеют в своей структуре списков;
- расширенное обслуживание прерываний:
- доступно до 16 линий прерываний. Каждый запрос прерывания может быть индивидуально скоммутирован на любую из 16 линий;
- информация о последующей обработке сообщения отражается в специальном регистре, состоящем из битов оповещения.

16.5 CAN узел

CAN узел состоит из нескольких функциональных блоков, показанных на рисунке 282.

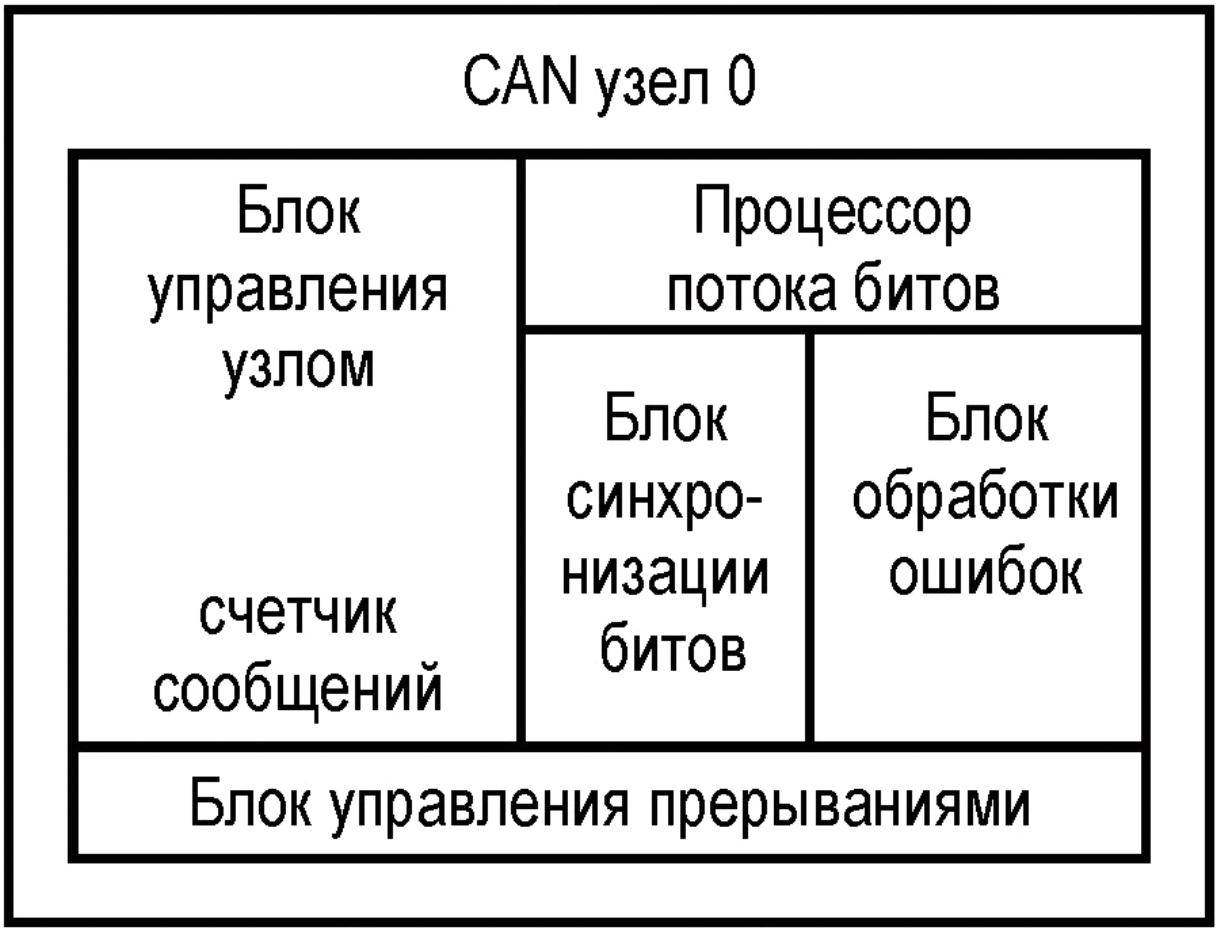


Рисунок 282 – Структура CAN узла

Процессор потока битов

Процессор потока битов управляет операциями с сообщениями данных, удаленного запроса, ошибки и перезагрузки (соответствует стандарту ISO 11898), также выполняет роль преобразователя между последовательным потоком данных и регистрами ввода-вывода.

Блок синхронизации битов

Блок синхронизации битов определяет длительность времени бита и положение точки выборки, согласно запрограммированным установкам, управляет вставкой задержки и ошибкой сдвига фазы. Также отвечает за ресинхронизацию.

Блок обработки ошибок

Блок обработки ошибок управляет счетчиками ошибок приема и передачи. В зависимости от состояния обоих счетчиков, CAN узел может находиться в состоянии активной ошибки, пассивной ошибки или быть отключенным от шины.

Блок управления узлом

Блок управления узлом координирует работу CAN узла:

- разрешает/запрещает действия узла на шине;
- разрешает/запрещает и генерирует различные события, касающиеся работы узла (например, «ошибка на шине», «успешное завершение передачи»), которые приводят к формированию запросов на прерывания;
- управляет счетчиком сообщений.

Блок управления прерываниями

Блок управления прерываниями управляет генерированием прерываний в течение работы CAN узла.

Контроллер сообщений

Контроллер сообщений управляет обменом сообщениями между CAN узлами и памятью сообщений и выполняет следующие функции:

- фильтрацию входящих сообщений для определения корректной области сообщения для сохранения полученных данных;
- определение области сообщения, содержимое которой будет передано в первую очередь (для каждого узла индивидуально);
- передачу содержимого области сообщения к CAN узлу, с параллельной вставкой в сообщение битов управления и состояния;
- осуществление буферизации FIFO и функционирования шлюза;
- объединение битов оповещения ждущих обработки сообщений.

Контроллер списка

Контроллер списка выполняет все операции по модификации списков областей сообщений. Только контроллер списка может изменять структуру списка. Распределение и перераспределение областей сообщений возможно посредством командного интерфейса (панели команд), остальные операции конечный автомат контроллера списка выполняет автономно.

Управление прерываниями

На рисунке 283 показана структура формирования запроса на прерывание.

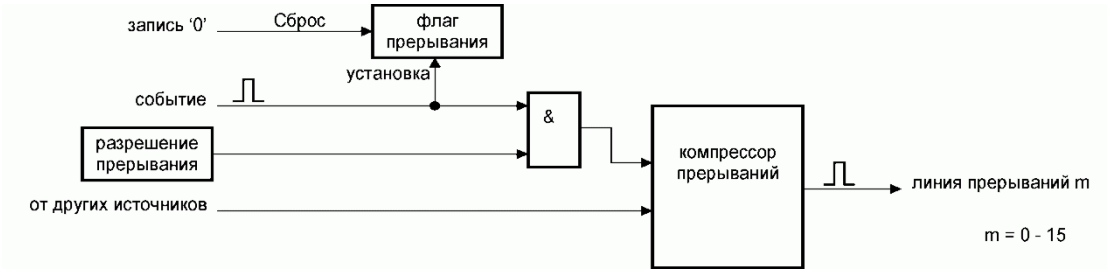


Рисунок 283 – Структура формирования запроса на прерывание

Событие, по которому должен быть сгенерирован запрос на прерывание, устанавливает флаг прерывания и, если это разрешено, формирует запрос на прерывание на одной, выбранной из 16, линии прерываний. Импульс запроса на пре-

рывание генерируется независимо от состояния флага прерывания. Флаг прерывания может быть сброшен программно, записью «0».

Если к одной линии прерываний подключены несколько источников прерываний, то появление импульса от любого источника сформирует запрос на прерывание.

Логика управления прерываниями использует схему компрессии прерываний (показана на рисунке 284).

Источниками прерываний являются:

- CAN узлы, 8 источников по четыре для каждого узла;
- области сообщений, 64 источника по два для каждой области;
- программное прерывание, один источник (регистр MITR).

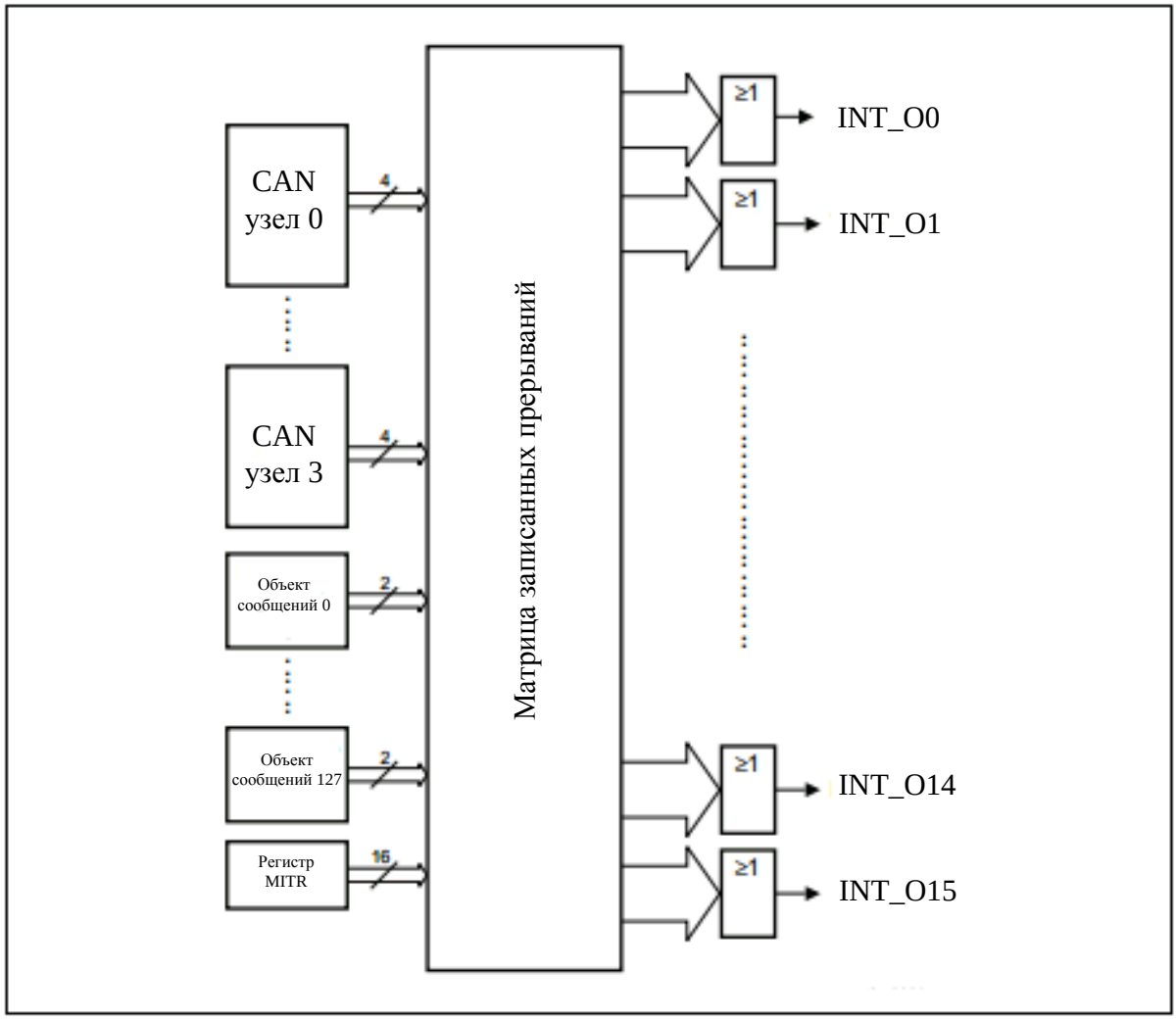


Рисунок 284– Компрессор прерываний

Каждый аппаратный источник прерывания управляется 4 битами указателя прерываний, который определяет для него одну из 16 линий прерываний, что позволяет коммутировать на одну линию несколько источников прерываний.

Описание бит регистров указателя прерываний узла NIPR0 и NIPR1 представлено на рисунке 285, коды выбора линии прерываний для полей ALINP, LECINP, TRINP, CFCINP регистров NIPR0 и NIPR1 – в таблице 141.

31 … 16
Зарезервировано
R-0

15	14	13	12	11	10	9	8
CFCINP3	CFCINP2	CFCINP1	CFCINP0	TRINP3	TRINP2	TRINP1	TRINP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
LECINP3	LECINP3	LECINP3	LECINP3	ALINP3	ALINP2	ALINP1	ALINP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W - доступ записи, -0 – значение после сброса

Рисунок 285 – Регистр указателя прерываний узла x модуля CAN (NIPRx) – адрес 0A08h + x*100h (x = 0 или 1)

Биты 31-16 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Биты 15-12 CFCINP3 – CFCINP0. Указатель узла для прерывания счетчика сообщений. Это битовое поле выбирает выходную линию прерываний (одну из 16) для прерывания счетчика сообщений CAN узла. Кодировка в таблице 141.

Биты 11-8 TRINP3 – TRINP0. Указатель узла для прерывания после успешной пересылки сообщения. Это битовое поле выбирает выходную линию прерываний (одну из 16) для прерывания, формируемого после успешной пересылки сообщения CAN узла. Кодировка в таблице 141.

Биты 7-4 LECINP3 – LECINP0. Указатель узла для прерывания кода последней ошибки. Это битовое поле выбирает выходную линию прерываний (одну из 16) для прерывания кода последней ошибки CAN узла. Кодировка в таблице 141.

Биты 3-0 ALINP3 – ALINP0. Указатель узла для ALERT прерывания. Это битовое поле выбирает выходную линию прерываний (одну из 16) для ALERT прерывания CAN узла. Кодировка в таблице 141.

Таблица 141 – Коды выбора линии прерываний для полей ALINP, LECINP, TRINP, CFCINP регистров NIPR0 и NIPR1

Код	Номер выходной линии прерываний
0000_2	0
0001_2	1
0010_2	2
0011_2	3
0100_2	4
0101_2	5
0110_2	6
0111_2	7
1000_2	8
1001_2	9
1010_2	10
1011_2	11
1100_2	12
1101_2	13
1110_2	14
1111_2	15

Описание бит регистров указателя прерываний области сообщений MOIPRn, где n = 0 – 255, представлен на рисунке 286, коды выбора линии прерываний для полей TXINP и RXINP регистров MOIPRn – в таблице 142.

31	30	29	28	27	26	25	24
CFCVAL15	CFCVAL14	CFCVAL13	CFCVAL12	CFCVAL11	CFCVAL10	CFCVAL9	CFCVAL8
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0

23	22	21	20	19	18	17	16
CFCVAL7	CFCVAL6	CFCVAL5	CFCVAL4	CFCVAL3	CFCVAL2	CFCVAL1	CFCVAL0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0

15	14	13	12	11	10	9	8
MPN7	MPN6	MPN5	MPN4	MPN3	MPN2	MPN1	MPN0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
TXINP3	TXINP2	TXINP1	TXINP0	RXINP3	RXINP2	RXINP1	RXINP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 286 – Регистр указателя прерываний области сообщений n

модуля CAN (MOIPRn) – адрес 1808h + n*20h (n = 0 ÷ 255)

Биты 31-16 CFCVAL15 – CFCVAL0. Значение счетчика сообщений. После того, как полученное сообщение сохранено в области сообщения n или после успешной передачи данных из области сообщения n, значение битового поля счетчика сообщений CFC (регистра NFCR) копируется в битовое поле CFCVAL.

Биты 15-8 MPN7 – MPN0. Номер ждущего сообщения. Это битовое поле указывает позицию флага в регистре ждущих прерываний MSPND0/ MSPND1, который выставляется в ответ на формирование прерывания областью сообщения n по окончании приема/передачи сообщения.

Биты 7-4 TXINP3 – TXINP0. Указатель прерывания передачи. Это битовое поле выбирает выходную линию прерываний (одну из 16) для прерывания, формируемого областью сообщения n после передачи сообщения. Кодировка в таблице 142.

RXINP3 – RXINP0. Указатель прерывания приема. Это битовое поле выбирает выходную линию прерываний (одну из 16) для прерывания, формируемого областью сообщения n после приема сообщения. Кодировка в таблице 142.

Таблица 142 – Коды выбора линии прерываний для полей TXINP и RXINP регистров MOIPRn, где n = 0 – 255

Код	Номер выходной линии прерываний
0000_2	0
0001_2	1
0010_2	2
0011_2	3
0100_2	4
0101_2	5
0110_2	6
0111_2	7
1000_2	8
1001_2	9
1010_2	10
1011_2	11
1100_2	12
1101_2	13
1110_2	14
1111_2	15

Описание бит регистра прерываний MITR представлено на рисунке 287.

R – доступ чтения, W - доступ записи, -0 – значение после сброса

Рисунок 287 – Регистр прерываний модуля CAN (MITR) – адрес 09CCh

Биты 31-16 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Биты 15-0 IT15 – IT0. Генерирование прерывания. Для того, чтобы на желаемой (одной из 16) линии прерываний сформировался запрос на прерывание, необходимо записать «1» в бит, соответствующий этой линии прерываний. Чтобы сформировать запросы на прерывания одновременно на нескольких линиях прерываний, необходимо одновременно записать «1» в биты, соответствующие этим линиям прерываний. Номер позиции бита в поле IT совпадает с номером соответствующей ему линии прерываний. Запись «0» в биты поля IT игнорируется. Чтение регистра всегда возвращает нули.

Когда область сообщения n генерирует запрос на прерывание по окончании приема или передачи сообщения, запрос передается на линию прерываний, выбранную в битовом поле RXINP или TXINP регистра MOIPRn области сообщения n. Если количество областей сообщений больше, чем количество линий прерываний, то на одну линию могут приходить несколько запросов прерываний. Для разрешения конфликтов на линиях прерываний в модуле CAN предусмотрен механизм распределения приоритетов для областей сообщений.

16.6 Управление модулем CAN

В случаях, когда модуль CAN не используется, он может быть принудительно выключен установкой бита DISR в регистре CLC, при этом бит состояния DISS того же регистра всегда показывает, выключен ли модуль CAN DISS = 1_2 или

включен DISS = 0_2. Описание бит регистра управления частотой CLC представлен на рисунке 288.

По умолчанию состояние модуля CAN после сброса – «выключен» DISS = 1_2.

R – доступ чтения, W - доступ записи, -0 – значение после сброса

Рисунок 288 – Регистр управления частотой модуля CAN (CLC) – адрес 0800h

Биты 31-6 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 5 FSOE[1)]. Бит разрешения быстрого выключения модуля CAN. Разрешает/запрещает возможность быстрого выключения модуля для системы отладки схемы.
0 = Выключение модуля запрещено.
1 = Выключение модуля разрешено.

Бит 4 SBWE[1)]. Бит разрешения для битов управления приостановкой работы для системы отладки схемы. Определяет возможность записи в биты SPEN и FSOE. Бит SBWE доступен только для записи, но при этом значение, записываемое в этот бит, не сохраняется. Чтение SBWE всегда возвращает «0».
0 = Запись в биты запрещена.
1 = Запись в биты разрешена.

Бит 3 EDIS. Бит внешнего обращения. Управляет ответной реакцией модуля CAN на внешнее указание перехода в режим сна.
Примечание – Режим сна для данного модуля CAN недоступен.
0 = Режим сна разрешен.
1 = Режим сна запрещен.

Бит 2 SPEN[1)]. Бит разрешения режима приостановки работы (Suspend) для системы отладки схемы. Разрешает переход модуля CAN в режим приостановки работы. Бит SPEN может быть доступен для за-

писи, только если бит SBWE = 1_2.
0 = Режим запрещен (переход модуля в режим приостановки работы запрещен).
1 = Режим разрешен (переход модуля в режим приостановки работы разрешен).

Бит 1 DISS. Бит состояния модуля. Отражает текущее состояние модуля CAN.
0 = Модуль находится в рабочем состоянии.
1 = Модуль выключен.

Бит 0 DISR. Бит запроса выключения модуля. Управляет включением/выключением модуля CAN.
0 = Выключение модуля не требуется.
1 = Требуется выключение модуля. После записи «1» в DISR начинается процесс выключения модуля CAN.

Примечание – Когда модуль CAN находится в выключенном состоянии, только регистр CLC доступен для записи/чтения. Доступ к остальным регистрам модуля запрещен.

_____
[1)] Биты доступны только в отладочном режиме работы схемы.

Когда выключенный модуль CAN включается записью «0» в CLC.DISR, бит состояния DISS меняется из «1» в «0». Следует программно проверять состояние модуля CAN чтением бита DISS перед началом программирования регистров модуля.

Примечание – После сброса тактирование модуля CAN внешней частотой f_{CLC} должно быть разрешено записью «0» в DISR до начала задания тактовой частоты таймера модуля CAN в регистре делителя FDR.

Как уже было сказано выше, для выключения модуля CAN следует записать «1» в бит DISR регистр CLC.

При выключении модуль CAN, в первую очередь, завершает все текущие операции. После того, как все блоки переходят в состояние готовности к выключению, модуль CAN утверждает состояние бита DISS и отключает внутреннее тактирование. После этого логика управления частотой контроллера отключает подачу тактового сигнала к модулю CAN.

16.6.1 Делитель частоты

Делитель частоты модуля CAN позволяет генерировать и выводить f_{OUT}, программируемую внутреннюю частоту f_{CAN} из внешней частоты f_{CLC} делением на любое число от 0 до 1023. Делитель частоты контролируется регистром делителя FDR и может быть сконфигурирован для работы в двух режимах:

- нормальном;
- дробного деления.

Описание бит регистра делителя частоты FDR представлен на рисунке 289.

31	30	29	28	27 … 26	25	24
DISCLK	ENHW	SUSREQ	SUSACK	Зарезервировано	RESULT9	RESULT8
RWH-0	RW-0	RH-0	RH-0	R-0	RH-0	RH-0

23	22	21	20	19	18	17	16
RESULT7	RESULT6	RESULT5	RESULT4	RESULT3	RESULT2	RESULT1	RESULT0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

15	14	13	12	11	10	9	8
DM1	DM0	SC1	SC0	SM	Зарезер-вировано	STEP9	STEP8
RW-0	RW-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0

7	6	5	4	3	2	1	0
STEP7	STEP6	STEP5	STEP4	STEP3	STEP2	STEP1	STEP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 289 – Регистр делителя частоты модуля CAN (FDR) – адрес 080Ch

Бит 31 DISCLK. Бит запрета тактирования. Аппаратно управляемый запрет выходного сигнала f_{CAN} делителя. В случае конфликта между аппаратным сбросом и программной установкой бита DISCLK приоритет остается за программой. Операций записи/чтения желательно избегать.
0 = Генерирование частоты f_{CAN} разрешено, согласно установкам (см. битовое поле DM).
1 = Делитель выключен. Частота f_{CAN} не генерируется.

Бит 30 ENHW. Бит разрешения аппаратного контроля частоты. Управляет подачей внешней тактовой частоты и битом DISCLK.
Примечание – Сигнал разрешения внешней тактовой частоты для делителя модуля CAN аппаратно удерживается равным «0».
0 = Бит DISCLK не может быть очищен аппаратно в ответ на появление высокого уровня сигнала разрешения внешней тактовой частоты.
1 = Бит DISCLK может быть очищен аппаратно в ответ на появление высокого уровня сигнала разрешения внешней тактовой частоты.

Бит 29 SUSREQ[1)]. Бит запроса включения режима приостановки работы.
Примечание – Модуль CAN переходит в режим приостановки работы, когда оба бита SUSREQ и SUSACK равны «1».
0 = Включение режима приостановки работы не требуется.
1 = Требуется включение режима приостановки работы.

Бит 28 SUSACK[1)]. Бит подтверждения режима приостановки работы (Suspend).
Примечание – Модуль CAN переходит в режим приостановки работы, когда оба бита SUSREQ и SUSACK равны «1».
0 = Режим приостановки не подтверждается.
1 = Режим приостановки подтверждается.

Биты 27-26 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Биты 25-16 RESULT9 – RESULT0. Результат.
- в нормальном режиме RESULT работает как перезагружаемый счетчик с инкрементированием на 1;
- в режиме дробного делителя RESULT работает как счетчик с инкрементированием на значение, хранящееся в битовом поле STEP;
- если DM = 01_2 или DM = 10_2, в RESULT загружается значение $3FF_{16}$ (максимальное значение счетчика).

Биты 15-14 DM1, DM0. Выбор режима делителя.
00 = Делитель выключен. Частота f_{CAN} не генерируется. Сигнал сброса внешнего делителя RED равен «1». Значение поля RESULT не изменяется (остается равным значению после сброса).
01 = Делитель в нормальном режиме работы.
10 = Делитель в режиме дробного деления.
11 = Делитель выключен. Частота f_{CAN} не генерируется. Значение поля RESULT не изменяется.

Биты 13-12 SC[1)]. Управление в режиме приостановки работы. Определяет поведение делителя в режиме приостановки работы.
00 = Нормальное генерирование тактовой частоты.
01 = Генерирование тактовой частоты останавливается. f_{CAN} не генерируется. Значение RESULT не изменяется, за исключением случая, когда DM = 01_2 или DM = 10_2.
10 = Генерирование тактовой частоты останавливается. f_{CAN} не генерируется. В RESULT загружается значение $3FF_{16}$.
11 = Так же как при SC = 10_2, но значение сигнала сброса внешнего делителя RED равно «1» (независимо от состояния битового поля DM).

Бит 11 SM[1)]. Бит выбора режима приостановки работы (Suspend).
0 = Промежуточный режим приостановки работы.
1 = Основной режим приостановки работы.

Бит 10 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Биты 9-0 STEP. Значение шага.
В нормальном режиме работы делителя в STEP хранится значение перезагрузки для битового поля RESULT.
В режиме дробного деления в STEP хранится значение, прибавляемое к значению битового поля RESULT.

[1)] Биты доступны только в отладочном режиме работы схемы.

16.6.2 Нормальный режим делителя

В нормальном режиме FDR.DM = 01_2 делитель работает как обычный перезагружаемый счетчик. Шаг инкрементирования равен 1 (RESULT = RESULT + 1). Каждый раз, при переполнении RESULT = $3FF_{16}$, формируется импульс тактовой частоты f_{OUT}, счетчик сбрасывается и в него загружается значение, хранящееся в FDR.STEP.

Выходная частота определяется по формуле

$$f_{OUT} = f_{IN} \times 1/n, \quad (2.1)$$

где n = 1024 – STEP (в десятичном формате).

Для получения частоты $f_{OUT} = f_{IN}$ значение STEP должно быть $3FF_{16}$ (1023 в десятичном формате). На рисунке 290 показано формирование тактовой частоты f_{OUT} при значении STEP = $3FD_{16}$ (1021). После подстановки значения 1021 в формулу получим:

$$f_{OUT} = f_{IN} \times 1/(1024 - 1021) = f_{IN} / 3.$$

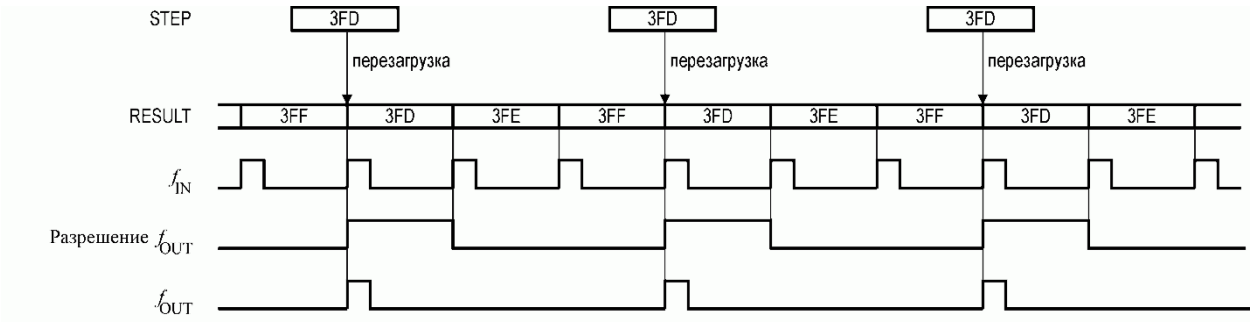


Рисунок 290 – Формирование тактовой частоты в нормальном режиме

16.6.3 Режим дробного деления

В режиме дробного деления FDR.DM = 10_2 выходная частота f_{OUT} формируется из входной частоты f_{IN} умножением на дробь n/1024, где n может принимать значения от 0 до 1023. В целом, режим дробного деления позволяет программировать частоту с более высокой точностью, чем нормальный режим.

В режиме дробного деления делитель также работает как перезагружаемый счетчик, но шаг инкрементирования равен значению STEP (RESULT = RESULT + STEP). Если результат сложения значений RESULT и STEP превышает $3FF_{16}$, возникает переполнение счетчика и генерируется импульс тактовой частоты f_{OUT}. Значение, на которое результат сложения превысил $3FF_{16}$, загружается в счетчик (RESULT) и счет продолжается.

Следует помнить, что в режиме дробного деления частота f_{OUT} может иметь джиттер периода, не превышающий одного периода f_{IN}.

Выходная частота определяется по формуле

$$f_{OUT} = f_{IN} \times n/1024, \qquad (2.2)$$

где n от 0 до 1023.

На рисунке 291 показано формирование тактовой частоты f_{OUT} при значении STEP = 234_{16} (564 в десятичном формате). После подстановки значения 564 в формулу (2.2) получим:

$$f_{OUT} = f_{IN} \times 564/1024 = 0{,}55 \times f_{IN}$$

Сигнал тактовой частоты f_{OUT} получается из логического «И» сигналов разрешения f_{OUT} и f_{IN}.

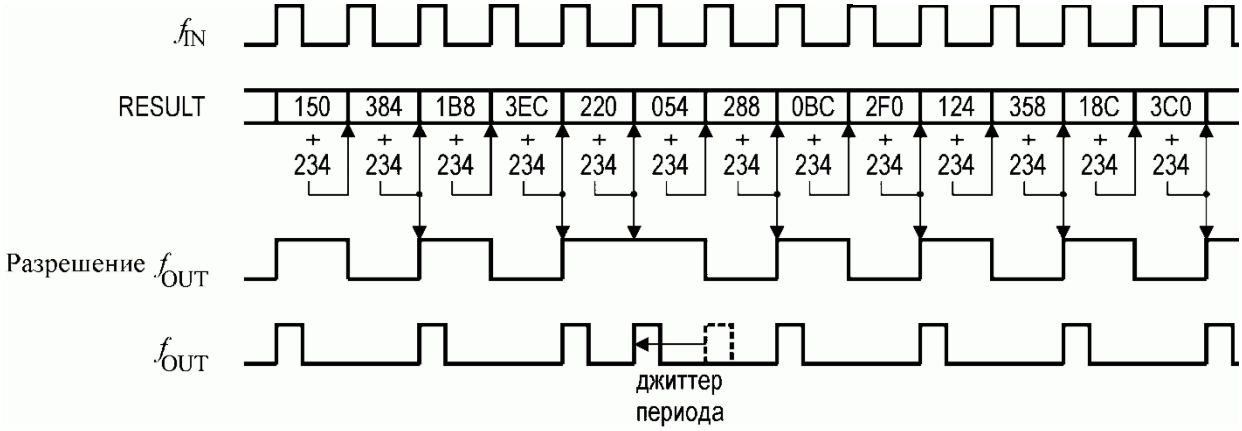


Рисунок 291 – Формирование тактовой частоты в режиме дробного деления

Таблица 143 – Сводная таблица сигналов управления функционирования делителя

SC	DM	DISS	RESULT	FOUT	Режим делителя
–	00_2	1	Без изменений	Не формируется	Выключен
	01_2	0	Загрузка[1)]	Формируется	Нормальный
	10_2				Дробный делитель
	11_2		Без изменений	Не формируется	Выключен

[1)] Каждый раз при записи в поле DM значения 01_2 или 10_2, значение RESULT устанавливается в $3FF_{16}$.

16.6.4 Управление выводами для приема

С каждым CAN узлом связано 8 входных линий. Программно выбрать линию для приема сообщений можно посредством поля RXSEL регистра NPCR0/NPCR1 (для каждого CAN узла свой регистр).

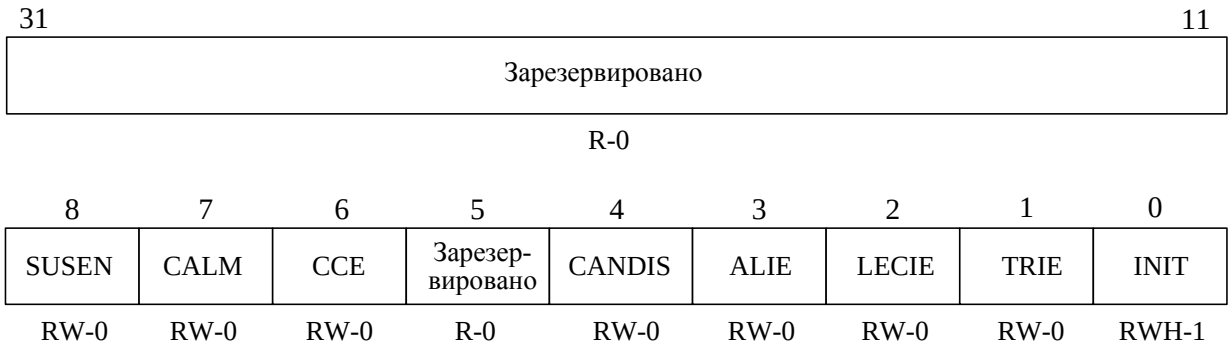
16.6.5 Управление CAN узлами

Каждый CAN узел имеет свою собственную логику управления и выдачи информации о состоянии и может быть сконфигурирован и работать независимо от другого узла.

Примечание – Далее в описании номера CAN узлов будут опущены. В названиях регистров под «x» будет подразумеваться номер CAN узла (0 и 1). Под «n» будет подразумеваться номер области сообщения (0 – 255).

Поскольку CAN узел 0 и CAN узел 1 идентичны, все нижесказанное справедливо для каждого из узлов.

Каждый их двух CAN узлов имеет свою группу регистров. Регистры в группах идентичны и отличаются только адресами. Описание бит регистра управления CAN узла NCRx представлено на рисунке 292. Описание бит регистра состсояния CAN узла NSRx представлено на рисунке 293.



R – доступ чтения, W - доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 292 – Регистр управления узла x модуля CAN (NCRx) – адрес 0A00h + x*100h (x = 0 или 1)

Биты 31-9 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 8 SUSEN. Разрешение режима приостановки. Бит позволяет перевести CAN узел в режим приостановки посредством отладочной системы OCDS. Бит SUSEN сбрасывается по сигналу «Сброса» системы OCDS.
0 = Переход в режим приостановки невозможен.
1 = Переход в режим приостановки возможен. По мере того, как CAN узел перейдет в состояние «простоя» или «отключен от шины», бит INIT будет автоматически установлен в «1». Действующее состояние бита INIT остается без изменений.

Бит 7 CALM. Режим анализа CAN. Если этот бит установлен, модуль CAN функционирует в режиме анализа. Это означает, что сообщения могут только приниматься и не могут быть переданы. Бит подтверждения не посылается после успешного приема сообщения. Флаг активной ошибки посылается рецессивным вместо доминантного. На линии отправки сообщений поддерживается рецессивный «1» уровень. Бит CALM может быть установлен только в течение времени, когда установлен бит INIT.

Бит 6 CCE. Разрешение изменения конфигурации.
0 = Регистры синхронизации битов NBTRx, управления портом NPCRx и счетчика ошибок NECNTx узла доступны только для чтения. Все попытки записи в эти регистры игнорируются.
1 = Регистры NBTRx, NPCRx и NECNTx доступны как для чтения, так и для записи.

Бит 5 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 4 CANDIS. Выключение узла. Установка этого бита выключает CAN узел. В первую очередь, узел переходит в состояние «простоя» или «отключен от шины», после чего автоматически устанавливается бит IDLE и, если разрешено (ALIE = 1_2), генерируется ALERT прерывание.

Бит 3 ALIE. Разрешение ALERT прерывания узла. ALERT прерывание, если разрешено, может быть сформировано любым из следующих событий:
- изменение состояния бита BOFF регистра NSRx;
- изменение состояния бита EWRN регистра NSRx;
- ошибка длины списка, которая также выставляет бит LLE регистра NSR;
- ошибка элемента списка, которая также выставляет бит LOE регистра NSR;
- бит INIT выставлен аппаратно.

Битовое поле ALINP регистра NIPRx выбирает, какая из линий прерываний соответствует данному типу прерывания.
0 = Формирование прерывания ALERT запрещено.
1 = Формирование прерывания ALERT разрешено.

Бит 2 LECIE. Разрешение прерывания узла при возникновении ошибки кода последней ошибки. Прерывание формируется каждый раз при изменении/ обновлении битового поля LEC регистра NSRx, если LEC > 0 (ошибка CAN протокола). Битовое поле LECINP регистра NIPRx выбирает, какая из линий прерываний соответствует данному типу прерывания.
0 = Формирование прерывания кода последней ошибки запрещено.
1 = Формирование прерывания кода последней ошибки разрешено.

Бит 1 TRIE. Разрешение прерывания узла после пересылки сообщения. После успешной пересылки сообщения (передача/ прием), если разрешено, генерируется прерывание. Битовое поле TRINP регистра NIPRx выбирает, какая из линий прерываний соответствует данному типу прерывания.
0 = Формирование прерывания запрещено.
1 = Формирование прерывания разрешено.

Бит 0 INIT. Инициализация узла. Бит INIT автоматически устанавливается, если узел переходит в состояние «отключен от шины».
0 = Сброс бита INIT разрешает участие узла в трафике CAN шины. Если на момент сброса бита INIT узел находился в состоянии «отключен от шины», идет процесс выхода их этого состояния (независимо от бита INIT). После получения необходимого количества последовательностей битов, узел выходит из состояния «отключен от шины» и снова включается в трафик. Если на момент сброса бита INIT, узел не находился в состоянии «отключен от шины», то узел ожидает обнаружение последовательности 11 рецессивных бит на шине и включается в трафик.
1 = Установка бита INIT прекращает участие узла в трафике CAN шины. Все текущие передачи останавливаются, и линии передач переходят в рецессивное состояние. Если на момент установки бита INIT узел находился в состоянии «отключен от шины», процесс выхода их этого состояния продолжается. Если после завершения процесса выхода из состояния «отключен от шины» (после обнаружения 128 последовательностей – каждая из 11 рецессивных битов) бит INIT остается активным, узел выходит из состояния «отключен от шины», но остается неактивным до тех пор, пока INIT = 1_2.

R – доступ чтения, W - доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 293 – Регистр состояния узла x модуля CAN (NSRx) – адрес 0A04h + x*100h (x = 0 или 1)

Биты 31-11 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 10 SUSACK. Бит подтверждения перехода узла CAN в режим приостановки.
0 = CAN узел не находится в режиме приостановки или ожидает перехода в режим приостановки, но еще не достиг состояния «простоя» или «отключен от шины».
1 = CAN узел находится в режиме приостановки. CAN узел неакти-

вен (бит INIT аппаратно установлен в «1») из-за использования его системой OCDS.

Бит 9 LOE. Флаг ошибки элемента списка. Флаг LOE должен сбрасываться программно записью «0». Запись «1» в бит LOE игнорируется.
0 = С момента сброса флага LOE не было обнаружено ни одной ошибки элемента списка.
1 = Ошибка элемента списка обнаружена при приеме сообщения. Обнаружена операция внесения в список области сообщения с несоответствующим номером списка LIST в регистре MOCTRn.

Бит 8 LLE. Флаг ошибки длины списка. Флаг LLE должен сбрасываться программно (записью «0»). Запись «1» в бит LLE игнорируется.
0 = С момента сброса флага LLE не было обнаружено ни одной ошибки длины списка.
1 = Ошибка длины списка обнаружена при приеме сообщения. В списке, который принадлежит принимающему CAN узлу, количество элементов отличается от указанного в битовом поле SIZE.

Бит 7 BOFF. Флаг состояния «отключен от шины».
0 = Контроллер CAN не находится в состоянии «отключен от шины».
1 = Контроллер CAN находится в состоянии «отключен от шины».

Бит 6 EWRN. Флаг критического количества ошибок.
0 = Лимит ошибок еще не достигнут.
1 = По крайней мере, один из счетчиков ошибок (REC, TEC) достиг лимита ошибок заданного битовым полем EWRNLVL.

Бит 5 ALERT. ALERT предупреждение. Флаг выставляется в следующих случаях (не взаимоисключающих):
- изменение бита BOFF регистра NSRx;
- изменение бита LOE регистра NSRx;
- ошибка длины списка, которая также выставляет бит
LLE регистра NSR;
- ошибка элемента списка, которая также выставляет бит LOE регистра NSR;
- бит INIT выставлен аппаратно.
Флаг ALERT должен сбрасываться программно (записью «0»). Запись «1» в бит ALERT игнорируется.

Бит 4 RXOK. Успешный прием сообщения. Флаг RXOK должен сбрасываться программно (записью «0»). Запись «1» в бит RXOK игнорируется
0 = С момента сброса флага RXOK не было получено ни одного сообщения.
1 = Сообщение получено успешно.

Бит 3 TXOK. Успешная передача сообщения. Флаг TXOK должен сбрасываться программно (записью «0»). Запись «1» в бит TXOK игнорируется.
0 = С момента сброса флага TXOK не было осуществлено ни одной успешной передачи.
1 = Сообщение передано успешно (т. е. без ошибок и с получением бита подтверждения от другого узла).

Биты 2-0 LEC2 – LEC0. Код последней ошибки. Показывает тип последней (из обнаруженных) CAN ошибки (кодировка в таблице 144).
Примечание – Битовое поле LEC может быть изменено программно, только если в него записывается код «111_2», в остальных случаях запись игнорируется.

Таблица 144 – Коды битового поля LEC

Значение LEC	Значение
1	2
000_2	Ошибок нет. На шине не было обнаружено ни одной ошибки с момента передачи последнего сообщения до настоящего времени
001_2	Ошибка стаффинга (заполнения, STUFF ERROR). Может быть обнаружена во время передачи шестого бита из последовательности шести одинаковых бит в поле сообщения, которое должно быть кодировано методом разрядного заполнения (заключается в том, что после передачи пяти битов одинаковой полярности, шестой бит должен иметь противоположную полярность и вставляться передатчиком в поток данных автоматически, приемник пропускает этот бит)
010_2	Ошибка формы (FORM ERROR). Обнаруживается, если: - в битовом поле фиксированного формата содержится количество битов, отличающееся от установленного; - на месте рецессивного бита находятся доминантный или наоборот. Исключение – для приемника доминантный бит в течение последнего бита поля «конец фрейма» не интерпретируется как ошибка формы

Окончание таблицы 144

1	2
011_2	Ошибка подтверждения (ACKNOWLEGMENT ERROR). Обнаруживается передатчиком всякий раз, когда он не обнаруживает доминантный бит АСК в «области подтверждения»
100_2	Разрядная ошибка или ошибка бита 1 (BIT 1 ERROR). Узел, который передает данные на шину, осуществляет мониторинг шины. Ошибка бита 1 имеет место, если при передаче рецессивного «1» бита (за исключением битов полей арбитража и подтверждения) на шине обнаруживается доминантный «0» бит
101_2	Разрядная ошибка или ошибка бита 0 (BIT 0 ERROR). Ошибка возникает в случаях: - во время передачи сообщения (или бита подтверждения, флага активной ошибки, флага перезагрузки), узел передает доминантный бит «0», но на шине обнаруживается рецессивный «1»; - во время выхода из состояния «отключен от шины» при каждом обнаружении последовательности из 11 рецессивных битов. В этом случае ЦП может использовать код 101_B для отслеживания длительного простоя шины
110_2	Ошибка циклического избыточного кода (CRC ERROR). Передатчик, по установленному алгоритму, вычисляет значение контрольной суммы (собственно CRC) для передаваемых данных и вставляет ее в сообщение. Приемник, после получения данных, вычисляет CRC по тому же алгоритму, что и передатчик и сравнивает вычисленное значение с принятым значением CRC. В случае несовпадения возникает ошибка
111_2	ЦП производит запись в LEC. ЦП может записать код 111_B в LEC в любой момент времени. Попытки записи в LEC любого другого значения игнорируются

Режим конфигурации включается установкой в «1» бита CCE регистра управления CAN узла NCRx. Режим конфигурации позволяет изменять параметры синхронизации битов и состояния счетчиков ошибок.

Режим анализа включается установкой в «1» бита CALM регистра NCRx. В режиме анализа сообщения данных и удаленные запросы отслеживаются без участия узла в операциях на шине (выход TXDC узла находится в рецессивном состоянии). Входящие удаленные запросы сохраняются в соответствующих областях «для передачи», а входящие сообщения данных – в соответствующих областях «для приема».

Так же в режиме анализа полная информация о конфигурации полученных сообщений сохраняется в соответствующих областях сообщений и может быть обработана ЦП. В ответ на входящие сообщения не выдается подтверждение, и не генерируются сообщения об ошибках. На удаленные запросы не выдаются сообщения данных, а сами сообщения данных не могут быть переданы установкой бита запроса передачи TXRQ регистра состояния области сообщения MOSTATn. Прерывания после приема генерируются (если это разрешено) для всех принятых сообщений, не содержащих ошибок.

Конфигурация прерываний определяется логикой управления узла посредством битов TRIE, ALIE и LECIE регистра NCRx:

- если бит управления TRIE = 1_2, прерывание после передачи генерируется, когда содержимое регистра NCRx меняется (после каждой успешной передачи сообщения);

- если бит управления ALIE = 1_2, прерывание ошибки генерируется, когда устанавливается состояние «отключен от шины» или превышено допустимое количество ошибок, или обнаруживается «недобор» данных;

- если бит управления LECIE = 1_B, прерывание кода последней ошибки генерируется, когда в битовое поле LEC регистра NSRx аппаратно записывается значение кода ошибки больше нуля. В таблице 144 представлены коды битового поля LEC.

Регистр состояния CAN узла NSRx отражает текущее состояние, содержит информацию о передачах и ошибках соответствующего узла.

Счетчик сообщений может использоваться для проверки последовательности передаваемых битов области сообщения или получения информации о завершении передачи/приема сообщения соответствующего CAN узла. Подсчет сообщений осуществляется 16-разрядным счетчиком, который управляется регистром счетчика сообщений CAN узла NFCRx. Битовые поля CFMOD и CFSEL определяют режим работы и событие для инкрементирования счетчика.

16.6.6 Блок синхронизации битов

Описание бит регистра синхронизации битов CAN узла NBTRx представлено на рисунке 294.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Зарезервировано															
R-0															

15	14	13	12	11	10	9	8
DIV8	TSEG2.2	TSEG2.1	TSEG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP	BRP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W - доступ записи, -0 – значение после сброса

Рисунок 294 – Регистр синхронизации битов x модуля CAN (NBTRx) – адрес 0A10h + x*100h (x = 0 или 1)

Биты 31-16 Зарезервировано. При чтении возвращает «0». При записи сле-

дует писать «0».

Бит 15 — DIV8. Делитель частоты на 8:
0 = Длительность кванта времени равна (BRP + 1) тактам частоты.
1 = Длительность кванта времени равна 8×(BRP + 1) тактам частоты.

Биты 14-12 — TSEG2.2 – TSEG2.0. Параметр 2. Временной сегмент от точки выборки до точки передачи, определяемый пользователем. Длительность сегмента равна (TSEG2 + 1) квантам времени и может быть уменьшена за счет ресинхронизации. Допустимые значения для TSEG1 от 1 до 7.

Биты 11-8 — TSEG1.3 – TSEG1.0. Параметр 1. Временной сегмент от сегмента синхронизации до точки выборки, определяемый пользователем и включающий в себя сегмент распространения. Длительность сегмента равна (TSEG1 + 1) квантам времени и может быть увеличена за счет ресинхронизации. Допустимые значения для TSEG1 от двух до 15.

Биты 7-6 — SJW1, SJW0. Ширина перехода ресинхронизации. Длительность ширины перехода ресинхронизации равна (SJW + 1) квантам времени.

Биты 5-0 — BRP5 – BRP0. Скорости передачи.
Если DIV8 = 0_B, тогда длительность одного кванта времени равна (BRP + 1) тактам частоты.
Если DIV8 = 1_B, тогда длительность одного кванта времени равна 8×(BRP + 1) тактам частоты.

Примечание – Регистр NBTRx может быть записан, если только бит CCE регистра NCRx установлен.

Согласно стандарта ISO 11898, время передачи одного бита разделено на сегменты, которые, в свою очередь, составлены из целочисленных отрезков времени, называемых квантами времени t_q, показанные на рисунке 295. Квант времени – фиксированная единица времени, получаемая из частоты синхронизации и делителя модуля CAN.

Сегмент синхронизации T_{Sync}. Позволяет синхронизировать начало обмена данными между передатчиком и приемником. Длительность сегмента всегда равна одному кванту времени t_q.

Сегмент распространения T_{Prop}. Используется для компенсации физического времени запаздывания сигнала в пределах сети. Длительность сегмента рассчитывается с учетом времени прохождения сигнала от передатчика к приемнику и обратно, входной задержки компаратора и задержки выхода драйвера и может составлять от 1 до 8 квантов времени.

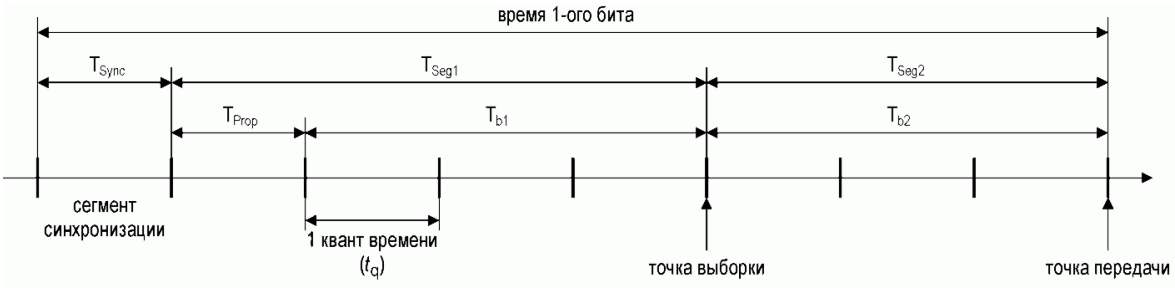


Рисунок 295 – Структура одного бита

Сегменты буфера фазы 1 и буфера фазы 2 (T_{b1} и T_{b2}), расположенные до и после точки выборки, используются для компенсации смещения фазы тактовых частот источника и приемника, обнаруживаемой после появления сегмента синхронизации, а также для оптимального расположения точки выборки полученного бита.

Точка выборки – момент, когда читается состояние шины для определения принятого бита. Как правило, длительность временного интервала от начала бита до точки выборки составляет (60 – 70) % времени бита, в зависимости от системных параметров.

Сегмент распространения и сегмент буфера фазы 1 вместе составляют сегмент параметра 1 (T_{Seg1}), который определяется битовым полем TSEG1 регистра синхронизации битов NBTRx (может быть записан, только если бит CCE регистра NCRx установлен). Согласно стандарту ISO, минимальная длительность сегмента параметра 1 должна составлять 3 кванта времени.

Сегмент параметра 2 (T_{Seg2}) определяется битовым полем TSEG2 регистра NBTRx и охватывает сегмент буфера фазы 2. Минимальная длительность сегмента параметра 2 составляет 2 кванта времени.

Согласно стандарту ISO, минимальная длительность одного бита, получающаяся сложением сегментов T_{Sync}, T_{Seg1} и T_{Seg2}, не должна быть менее 8 квантов времени.

Максимальная длительность бита – 25 квантов времени.

Примечание – Минимальное номинальное время передачи одного бита составляет 1 мкс, что соответствует скорости передачи 1 Мбит/с.

Формулы вычисления значений сегментов и времени одного бита:

$$t_q = (BRP + 1) / f_{CAN}, \text{ если DIV8} = 0_B, \text{ или} \quad (2.3)$$

$$t_q = 8 \times (BRP + 1) / f_{CAN}, \text{ если DIV8} = 1_B, \quad (2.4)$$

$$T_{Sync} = 1 \times t_q, \quad (2.5)$$

$$T_{Seg1} = (TSEG1 + 1) \times t_q \text{ (минимум 3 } t_q), \quad (2.6)$$

$$T_{Seg2} = (TSEG2 + 1) \times t_q \text{ (минимум 2 } t_q), \quad (2.7)$$

$$\text{Время бита} = T_{Sync} + T_{Seg1} + T_{Seg2} \text{ (минимум 8 } t_q). \quad (2.8)$$

Чтобы компенсировать смещение фазы между частотами генераторов различных узлов шины, каждый CAN модуль должен синхронизироваться по фронту смены уровня сигнала на шине от рецессивного к доминантному. Как только фронт обнаруживается, логика синхронизации сравнивает его текущее размещение с ожидаемым и выполняет настройку значений параметров 1 и 2 (T_{Seg1} и T_{Seg2}).

Есть два механизма синхронизации:

- аппаратная (жесткая синхронизация);
- ресинхронизация (синхронизация с восстановлением тактовых интервалов).

Аппаратная синхронизация выполняется по каждому фронту смены уровня сигнала на шине от рецессивного к доминантному. При аппаратной синхронизации временные интервалы сегментов, из которых складываются времена битов, не изменяются в течение всего сообщения.

Ресинхронизация выполняется автоматическим удлинением сегмента параметра 1 или укорачиванием сегмента параметра 2. Максимальное значение изменения сегментов параметров колеблется в пределах от 1 до 4 квантов времени. Синхронизация выполняется только при появлении фронта смены уровня сигнала на шине от рецессивного к доминантному. Фиксированное значение максимального числа последовательных бит одинаковой полярности гарантирует своевременное восстановление синхронизации. Смещение фазы фронта смены уровня сигнала на шине отслеживается относительно сегмента синхронизации и измеряется в квантах времени t_q.

Если величина фазового смещения меньше или равна запрограммированному значению ширины перехода ресинхронизации T_{SJW}, выполняется аппаратная синхронизация.

Если величина смещения фазы больше, чем ширина перехода ресинхронизации T_{SJW} и фазовое смещение положительно, то удлиняется сегмент параметра 1.

Если величина смещения фазы больше, чем ширина перехода ресинхронизации T_{SJW} и фазовое смещение отрицательно, то укорачивается сегмент параметра 2.

Значение T_{SJW} определяется полем SJW регистра NBTRx по формуле

$$T_{SJW} = (SJW + 1) \times t_q . \qquad (2.9)$$

Помимо прочего, должны соблюдаться следующие правила:

$$T_{Seg1} \geq T_{SJW} + T_{Prop}, \qquad (2.10)$$

$$T_{Seg2} \geq T_{SJW}. \qquad (2.11)$$

Соотношения между максимальным отклонением частоты f_{CAN} и сегментами буферов фаз и шириной перехода ресинхронизации следующие:

$$df_{CAN} \leq \text{миним.}[T_{b1}, T_{b2}] / 2 \times (13 \times \text{время бита} - T_{b2}), \qquad (2.12)$$

$$df_{CAN} \leq T_{SJW} / 20 \times \text{время бита}. \qquad (2.13)$$

В итоге имеем:

- сегмент синхронизации составляет 1 квант времени;
- сегмент распространения – от 1 до 8 квантов времени;
- сегмент буфера фазы 1 – от 1 до 8 квантов времени;
- сегмент буфера фазы 2 выбирается равным двум квантам времени или равным сегменту буфера фазы 1, если его значение более 2 квантов времени;
- ширина перехода ресинхронизации может составлять максимально 4 кванта времени, однако в типовых применениях достаточно 1.

Корректные значения параметров синхронизации битов должны быть записаны в регистр NBTRx до окончания инициализации (сброс бита INIT регистра управления CAN узла NCRx), т. е. до начала работы CAN узла.

Примечание – Следует помнить, что регистр NBTRx может быть записан, только если бит CCE регистра NCRx установлен.

16.6.7 Процессор потока битов

Процессор потока битов формирует (на основе содержимого областей сообщений) сообщения данных и удаленные запросы непосредственно перед отправкой на шину CAN. Процессор потока управляет генератором CRC и добавляет контрольную сумму к сообщению. После вставки битов начала SOF и конца EOF сообщения процессор потока начинает передачу сообщения по правилам арбитража шины CAN. В течение всего времени передачи сообщения процессор потока битов ведет мониторинг шины. Если обнаруживается несовпадение текущего (определяемого мониторингом) и ожидаемого (выдаваемого CAN узлом) уровня напряжения на шине, генерируется ошибка и соответствующий ей запрос на прерывание. Код возникшей ошибки отражается в битовом поле LEC регистра состояния CAN узла NSRx.

Корректность получаемых данных проверяется и подтверждается или не подтверждается кодом CRC. В случае не подтверждения возникает ошибка, генерируется запрос на прерывание, и код ошибки выставляется в регистре NSRx, кроме этого на шину выдается сообщение об ошибке.

После получения сообщения, не содержащего ошибок, и разбиения его на идентификатор и пакет данных, полученная информация записывается в буфер блока обработки сообщений, формируется соответствующее прерывание, и обновляются регистры состояния.

16.6.8 Блок обработки ошибок

Блок обработки ошибок CAN узла предназначен для выявления ошибок в работе устройств узла. В составе блока есть два счетчика: счетчик ошибок приема REC и счетчик ошибок передачи TEC. Счетчикам соответствуют битовые поля REC и TEC регистра счетчика ошибок CAN узла NECNTx. Инкрементированием и декрементированием счетчиков управляет процессор потока битов. В зависимости от значений счетчиков ошибок, CAN узел может находиться в одном из трех состояний:

- активной ошибки;

- пассивной ошибки;
- отключен от шины.

Узел в состоянии «активной ошибки» присоединен к шине и посылает флаг активной ошибки при обнаружении ошибок.

Узел в состоянии «пассивной ошибки» не должен посылать флаг активной ошибки. Он подключен к шине, но при обнаружении ошибок посылает флаг пассивной ошибки. После передачи узел в состоянии «пассивной ошибки» будет ждать инициализации дальнейшей передачи.

Узел в состоянии «отключения от шины» не может работать с шиной (выходные передатчики отключены). Описание бит регистра счетчика ошибок CAN узла NECNTx представлено на рисунке 296.

31 … 25	25	24
Зарезервировано	LEINC	LETD
R-0	RH-0	RH-0

23	22	21	20	19	18	17	16
EWRNLVL7	EWRNLVL6	EWRNLVL5	EWRNLVL4	EWRNLVL3	EWRNLVL2	EWRNLVL1	EWRNLVL0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

15	14	13	12	11	10	9	8
TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0

7	6	5	4	3	2	1	0
REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 296 – Регистр счетчика ошибок x модуля CAN (NECNTx) – адрес 0A14h + x*100h (x = 0 или 1)

Биты 31-26 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 25 LEINC. Инкрементирование при последней ошибке.
0 = Последняя ошибка приводит к инкрементированию счетчика ошибок на 1.
1 = Последняя ошибка приводит к инкрементированию счетчика ошибок на 8.

Бит 24 LETD. Последняя ошибка передачи.
0 = Последняя ошибка была обнаружена при приеме сообщения (с

инкрементированием REC).
1 = Последняя ошибка была обнаружена при передаче сообщения (с инкрементированием TEC).

Биты 23-16 EWRNLVL7 – EWRNLVL0. Промежуточный лимит ошибок. Поле определяет лимит ошибок (по умолчанию – 96), по достижении которого выставляется соответствующий флаг ошибки EWRN.

Биты 15-8 TEC7 – TEC0. Счетчик ошибок передачи. Поле хранит значение счетчика ошибок передачи сообщений CAN узла.

Биты 7-0 REC7 – REC0. Счетчик ошибок приема. Поле хранит значение счетчика ошибок приема сообщений CAN узла.

Существует 5 различных типов ошибок (не взаимоисключающих).

Разрядная ошибка или ошибка бита BIT ERROR

Разрядная ошибка или ошибка бита BIT ERROR – узел, который передает данные на шину, осуществляет мониторинг шины. Ошибка бита имеет место, если значение бита на шине отличается от переданного значения. Исключение – посылка рецессивного бита в течение битового потока поля арбитража или поля подтверждения. Передатчик, посылающий флаг пассивной ошибки и обнаруживший доминантный бит, не интерпретирует это как ошибку бита.

Ошибка стаффинга (заполнения) STUFF ERROR

Ошибка стаффинга (заполнения) STUFF ERROR может быть обнаружена во время передачи шестого бита из последовательности шести одинаковых бит в поле сообщения, которое должно быть кодировано методом разрядного заполнения. Метод разрядного заполнения заключается в том, что после передачи 5 бит одинаковой полярности, шестой бит должен иметь противоположную полярность и вставляться передатчиком в поток данных автоматически, приемник пропускает этот бит.

Ошибка циклического избыточного кода CRC ERROR

Ошибка циклического избыточного кода CRC ERROR – передатчик по установленному алгоритму, вычисляет значение контрольной суммы (собственно CRC) для передаваемых данных и вставляет ее в сообщение. Приемник после получения данных вычисляет CRC по тому же алгоритму, что и передатчик и сравнивает вычисленное значение с принятым значением CRC. В случае несовпадения возникает ошибка.

Ошибка формы FORM ERROR

Ошибка формы FORM ERROR обнаруживается, если:

- в битовом поле фиксированного формата содержится количество битов, отличающееся от установленного количества;

- на месте рецессивного бита находится доминантный бит или наоборот.

Примечание – Для приемника доминантный бит в течение последнего бита поля «конец фрейма» не интерпретируется как ошибка формы.

Ошибка подтверждения ACKNOWLEGMENT ERROR

Ошибка подтверждения ACKNOWLEGMENT ERROR обнаруживается передатчиком всякий раз, когда он не обнаруживает доминантный бит АСК в «области подтверждения».

Всякий раз, когда каким-либо узлом обнаружена ошибка бита, ошибка стаффинга, ошибка формы или ошибка подтверждения, со следующего бита начинается передача флага ошибки соответствующим узлом.

Всякий раз, когда обнаружена ошибка CRC, передача флага ошибки начинается с бита, следующего после разделителя подтверждения, если не передается флаг ошибки другого условия.

Как уже было сказано выше, имеются два счетчика ошибок.

Эти счетчики функционируют согласно перечисленным ниже правилам (при передаче сообщения может применяться более одного правила одновременно).

1 Когда принимающий узел обнаруживает ошибку, счетчик ошибок приема увеличивается на единицу, за исключением разрядной ошибки во время передачи флага активной ошибки или флага перезагрузки.

2 Когда принимающий узел обнаруживает появление доминантного бита в качестве первого после передачи флага ошибки, счетчик ошибок приема увеличивается на восемь.

3 Когда передающий узел посылает флаг ошибки, счетчик ошибок передачи увеличивается на восемь.

Исключения, при которых значение счетчика ошибок передачи не изменяется:

- узел находится в состоянии «пассивной ошибки» и обнаруживает ошибку подтверждения (не обнаружен доминантный бит ACK) в течение передачи флага пассивной ошибки;

- передающий узел посылает флаг ошибки вследствие обнаружения ошибки стаффинга во время арбитража из-за того, что бит, находящийся перед битом RTR, должен быть рецессивным, был передан как рецессивный, но детектируется как доминантный.

4 Если передающий узел обнаруживает разрядную ошибку при передаче флага активной ошибки или флага перезагрузки, счетчик ошибок передачи увеличивается на восемь.

5 Если принимающий узел обнаруживает разрядную ошибку во время передачи флага активной ошибки или флага перезагрузки, счетчик ошибок приема увеличивается на восемь.

6 Любой узел сети допускает до семи последовательных доминантных бит после передачи флага активной ошибки, флага пассивной ошибки или флага перезагрузки.

Счетчик ошибок передачи увеличивается на восемь (в случае передачи) и счетчик ошибок приема увеличивается на восемь (в случае приема), если:

- обнаружена последовательность из 14 доминантных битов (в случае флага активной ошибки или флага перезагрузки);

- обнаружено восемь доминантных битов вслед за флагом пассивной ошибки;

- обнаружена последовательность (любая) из восьми доминантных битов при передаче.

7 После успешной передачи сообщения (получение бита ACK и отсутствие ошибок), счетчик ошибок передачи уменьшается на единицу, если его значение не равно нулю.

8 После успешного приема сообщения (прием без ошибок, соответствующее значение поля ACK Slot и отправка бита ACK), счетчик ошибок приема уменьшается на 1, если его значение было между единицей и 127. Если в счетчике ошибок приема ноль, счетчик остается без изменений. Если значение счетчика ошибок приема больше чем 127, он примет значение между 119 и 127.

9 Узел находится в состоянии «пассивной ошибки», когда один из счетчиков (ошибок передачи/приема) больше или равен 128. Возникновение ошибки, вследствие чего узел принял состояние «пассивной ошибки», является причиной того, что узел передает флаг активной ошибки.

10 Узел переходит в состояние «отключен от шины», если значение счетчика ошибок передачи равно или больше 256.

11 Узел, находившийся в состоянии «пассивной ошибки», снова переходит в состояние активной ошибки, если оба счетчика (ошибок передачи/приема) меньше или равны 127.

12 Узлу, который находится в состоянии «отключен от шины», разрешается перейти в состояние «активной ошибки», с установкой обоих счетчиков в «0», после того, как на шине будут обнаружены 128 последовательностей из 11 рецессивных битов.

Примечания

1 Величина счетчика ошибки большая, чем 96, указывает на серьезные нарушения на шине. Это можно использовать для проверки состояния шины.

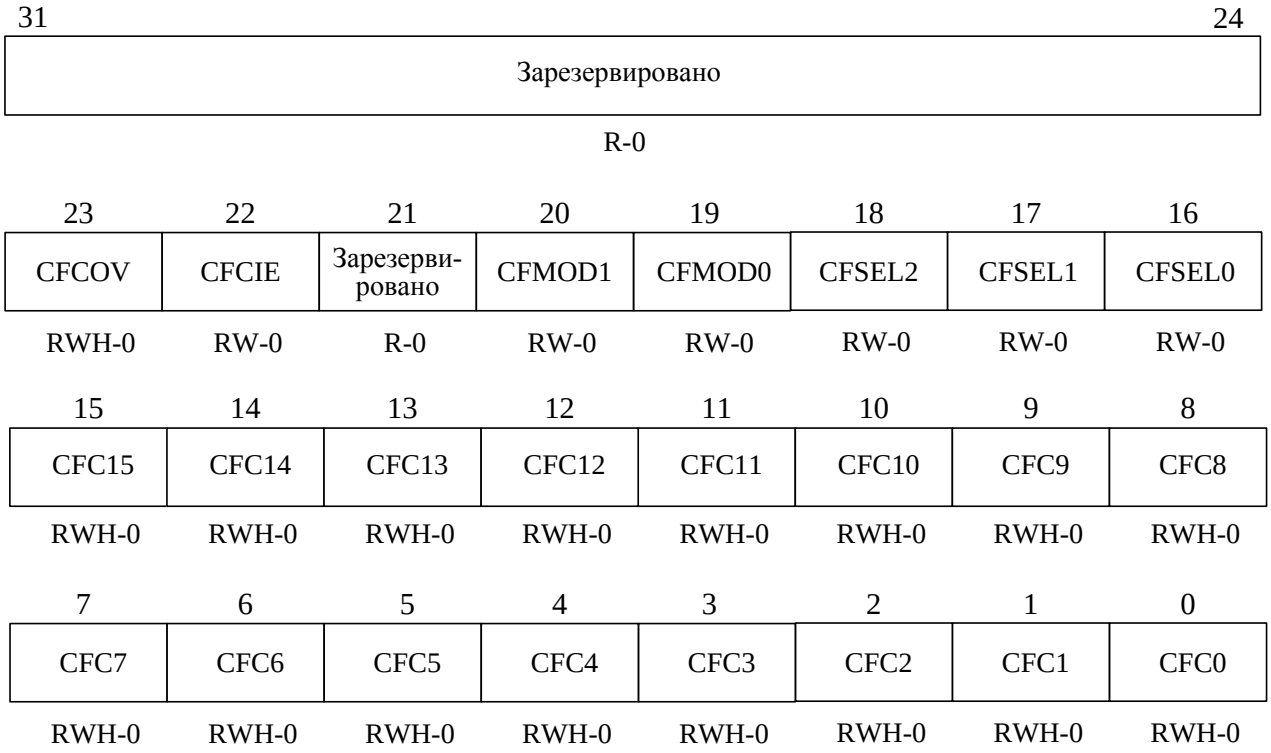
2 Возможна ситуация, когда после запуска узла, он окажется единственным на шине (других узлов нет или они отключены от шины). В этом случае, если узел начнет передачу, то по ее окончании он не получит подтверждения. Это будет воспринято как ошибка передачи, и сообщение будет передано заново. Вследствие возникновения такой ситуации узел может перейти в состояние «пассивной ошибки», но не может перейти в состояние «отключен от шины».

О том, что CAN узел находится в состоянии «отключен от шины», сигнализирует флаг BOFF регистра NSRx.

Флаг EWRN регистра NSRx устанавливается, когда хотя бы один из счетчиков достиг или превысил лимит ошибок, определенный в битовом поле EWRNLVL регистра NECNTx. Как только значения обоих счетчиков не будут превышать лимит ошибок, флаг EWRN сбросится.

16.6.9 Счетчик сообщений

Каждый CAN узел имеет в своем составе 16-разрядный счетчик сообщений/синхросчетчик для подсчета количества принятых и переданных сообщений. Счетчик управляется регистром счетчика сообщений CAN узла NFCRx. Описание бит регистра счетчика сообщений CAN узла NFCRx представлено на рисунке 297.



R – доступ чтения, W - доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 297 – Регистр счетчика сообщений x модуля CAN (NFCRx) – адрес 0A18h + x*100h (x = 0 или 1)

Биты 31-24 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 23 CFCOV. Флаг переполнения счетчика сообщений. Устанавливается при переполнении счетчика сообщений. В режиме синхросчетчика CFCOV устанавливается при изменении CFC. Если разрешено CFCIE = 1_B, при переполнении счетчика генерируется прерывание. Флаг CFCOV сбрасывается программно.
0 = Нет переполнения счетчика.
1 = Счетчик переполнился

Бит 22 CFCIE. Разрешение прерывания счетчика сообщений. Бит разрешения генерирования прерывания счетчика сообщения при его переполнении. Линия прерываний (одна из 16) для прерывания счетчика сообщений выбирается битовым полем CFCINP регистра NIPRx.
0 = Генерирование прерывания запрещено.
1 = Генерирование прерывания разрешено.

Бит 21 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Биты 20-19 CFMOD1, CFMOD0. Режим счетчика сообщений.
00 = Режим счетчика сообщений. Счетчик сообщений инкрементируется после каждого успешного приема/передачи сообщения.
01 = Режим подсчета битов.
10 = Режим синхросчетчика. Счетчик сообщений используется для анализа синхронизации битов.

Биты 18-16 CFSEL2 – CFSEL0. Выбор объекта счета.
Режим счетчика сообщений:
- если CFSEL.0 установлен, CFC инкрементируется каждый раз при получении сообщения, не имеющего соответствующей области сообщения;
- если CFSEL.1 установлен, CFC инкрементируется каждый раз при получении сообщения, имеющего соответствующую область сообщения;
- если CFSEL.2 установлен, CFC инкрементируется после каждой успешной отправки сообщения.
Режим подсчета битов:
000 = Счетчик сообщений инкрементируется каждый раз с началом очередного бита, начиная от начала передачи сообщения. Значение счетчика сохраняется в CFC.
Режим синхросчетчика:
Доступные режимы сведены в таблицу 145. Если CFCIE установлен, то при изменении CFC генерируются прерывания соответствующего CAN узла.

Биты 15-0 CFC15 – CFC0. Счетчик сообщений.
- в режиме счетчика сообщений CFMOD = 00_2 CFC хранит значение счетчика сообщений;
- в режиме подсчета битов CFMOD = 01_2 CFC хранит значение счетчика, который инкрементируется с началом очередного бита, начиная от начала передачи сообщения;
- в режиме синхросчетчика CFMOD = 10_B CFC всегда хранит количество тактов частоты f_{CLC} (результат замера) минус один.
Например, если значение CFC равно 34 в режиме измерения CFSEL = 000_2, то это означает, что между появлениями передних фронтов двух доминантных битов (при приеме сообщения) прошло 35 тактов частоты f_{CLC}.

Таблица 145 – Режимы синхросчетчика

CFSEL	Описание
1	2
000_2	Всякий раз, когда доминантный фронт (перепад сигнала из «1» в «0») обнаруживается на принимающем входе, время, измеренное в тактах частоты f_{CLC} между этим фронтом и новым доминантным фронтом, сохраняется в CFC
001_2	Всякий раз, когда рецессивный фронт (перепад сигнала из «0» в «1») обнаруживается на принимающем входе, время, измеренное в тактах частоты f_{CLC} между этим фронтом и новым рецессивным фронтом, сохраняется в CFC
010_2	Всякий раз, когда принимается доминантный фронт как результат переданного доминантного фронта, время, измеренное в тактах частоты f_{CLC} между этими фронтами, сохраняется в CFC
011_2	Всякий раз, когда принимается рецессивный фронт как результат переданного рецессивного фронта, время, измеренное в тактах частоты f_{CLC} между этими фронтами, сохраняется в CFC
100_2	Всякий раз, когда на принимающем входе обнаруживается доминантный фронт, предназначенный для синхронизации, время, измеренное в тактах частоты f_{CLC} между этим фронтом и новой точкой выборки, сохраняется в CFC
101_2	Всякий раз в момент точки выборки время, измеренное в тактах частоты f_{CLC} между началом нового бита и началом предыдущего бита, сохраняется в CFC.11 – CFC.0. Также в момент точки выборки в CFC.15 – CFC.12 сохраняется дополнительная информация: - CFC.15: передаваемое значение текущего времени бита; - CFC.14: принимаемое значение выборки текущего времени бита; - CFC.13, CFC.12: информация CAN шины подробно – в таблице 146
110_2	Зарезервировано
111_2	Зарезервировано

Таблица 146 – Информация о состоянии CAN шины

CFC13, CFC12	Состояние шины
1	2
00_2	Нет бита. CAN шина находится в состоянии простоя, осуществляет (де-) стаффинг бита или имеет место один из сегментов: SOF, SRR, CRC, разделители, первые 6 битов EOF, IFS
01_2	Новый бит. Этот код представляет первый бит нового сегмента сообщения. Текущий бит является первым битом одного из следующих сегментов: 10-й бит стандартного идентификатора (только при передаче), RTR, зарезервированные биты, IDE, DLC, седьмой бит каждого байта данных и первый бит расширения идентификатора ID
10_2	Бит. Этот код представляет бит (не первый бит сегмента, который определяется как «Новый» бит) в пределах сегмента (длина сегмента должна быть более одного бита) сообщения. Текущий бит является битом одного из следующих сегментов: идентификатор (за исключением первого бита стандартного идентификатора и первого бита расширения идентификатора), DLC (3 LSB) и биты с 6 по 0 каждого байта данных
11_2	Завершающий бит. Текущий бит является битом одного из следующих сегментов: бит подтверждения, последний бит EOF, сообщение об активной/пассивной ошибке, сообщение о перезагрузке. Два или более следующих друг за другом «завершающих» бита формируют сигнал ошибки сообщения

Битовое поле CFSEL регистра NFCRx определяет один из трех режимов работы счетчика.

Режим подсчета сообщений

После успешной передачи и/или успешного приема сообщения, содержимое счетчика копируется в битовое поле CFCVAL регистра MOIPRn (регистр указателя прерываний области сообщения n) области сообщения, участвующей в пересылке данных. После чего счетчик сообщений инкрементируется.

Режим подсчета битов

Счетчик инкрементируется с началом очередного бита. С началом передачи/приема сообщения значение счетчика захватывается и сохраняется в битовом поле CFC регистра NFCRx. После окончания передачи/приема сообщения сохраненное значение копируется в битовое поле CFCVAL регистра MOIPRn области сообщения, участвующей в пересылке данных.

Режим синхросчетчика

Используется для отслеживания скорости потока битов и анализа синхронизации битов.

16.6.10 Прерывания CAN узла

Каждый CAN узел имеет четыре типа прерываний, каждое от своего источника, которые представлены на рисунке 298. Источники прерываний:

- успешная передача или прием сообщения;
- код последней ошибки;
- ALERT – состояние, возникающее в случаях:

а) когда хотя бы один из счетчиков ошибок CAN узла достигает значения своего лимита;

б) изменяется состояние «отключен от шины»;

в) возникает ошибка длины списка или ошибка списка областей;

- переполнение счетчика сообщений.

В дополнение к аппаратным прерываниям есть возможность программного генерирования прерываний с использованием регистра прерываний MITR. Запись «1» в n-й разряд битового поля IT регистра MITR генерирует сигнал запроса прерывания на соответствующей ему n-ой линии прерываний. Запись в регистр MITR с установкой нескольких битов в битовом поле IT приводит к параллельному генерированию запросов прерываний на соответствующих установленным битам линиях прерываний.

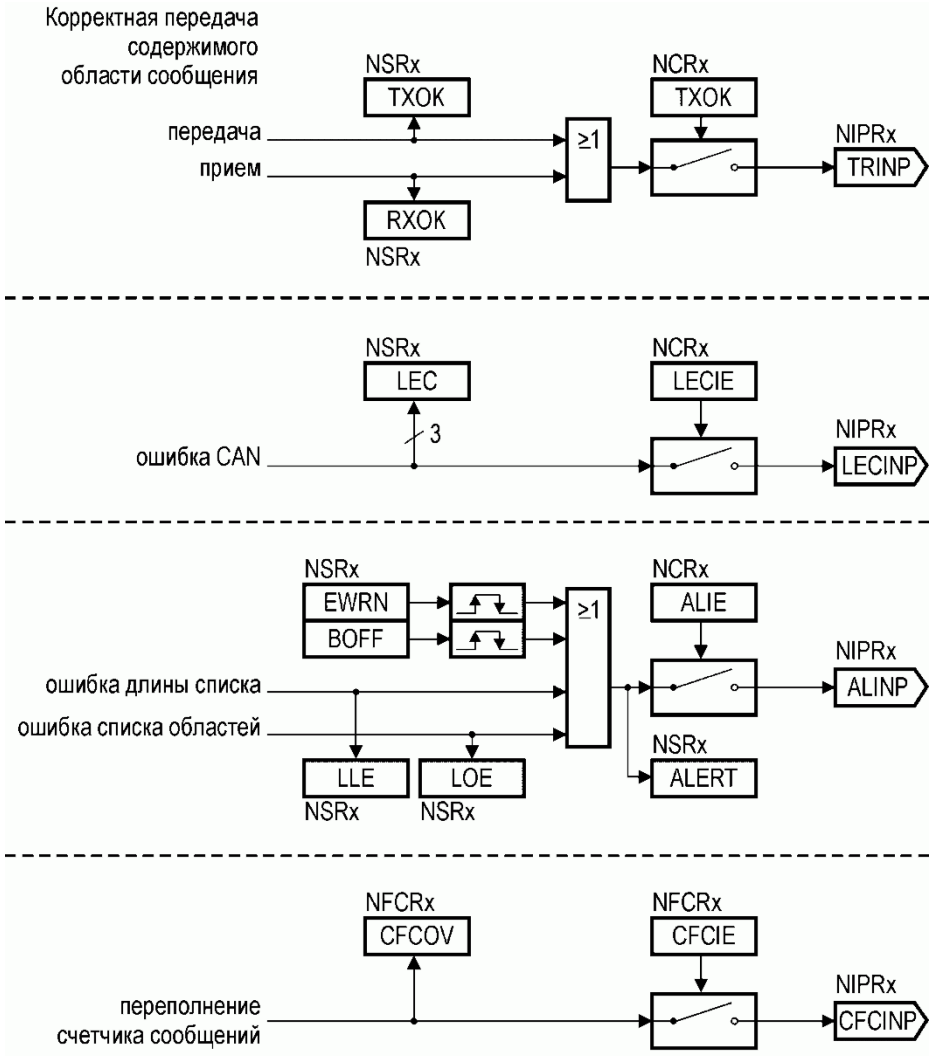
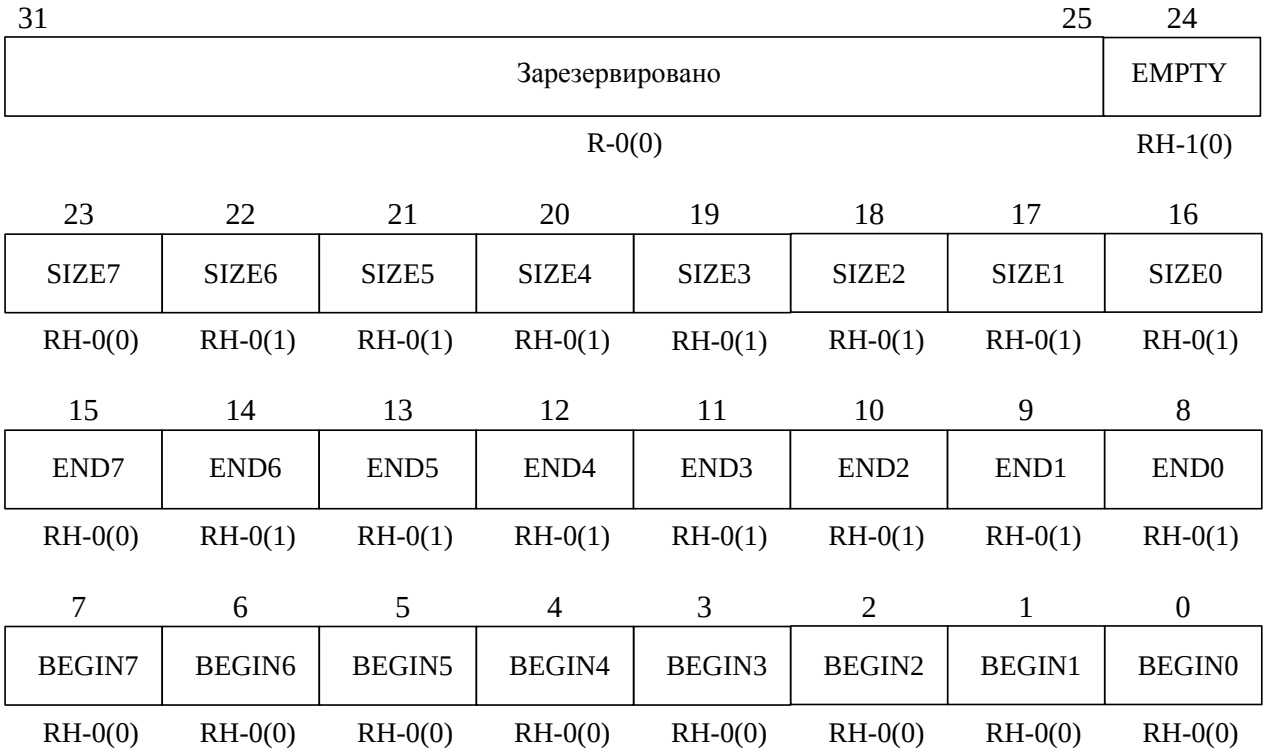


Рисунок 298 – Прерывания CAN узла

16.6.11 Структура списка областей сообщений

Области сообщений модуля CAN могут быть организованы в восемь двухсвязных списков. Каждая область сообщения может быть отнесена к одному из списков, каждый CAN узел имеет свой список и соответствующий регистр списка: LIST1 – для списка № 1 узла 0, LIST2 – для списка № 2 узла 1, и имеет указатели на предшествующую ей и следующую за ней области. Не распределенные между CAN узлами области сообщений организуются в отдельный список № 0, который управляется регистром LIST0. Остальные пять списков, от списка № 3 до списка № 7, являются свободными и управляются регистрами LIST3 – LIST7. Описание бит регистра списка CAN узла LISTy представлено на рисунке 299.



R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса
для LIST1 – LIST7, - (0) – значение после сброса для LIST0

Рисунок 299 – Регистр списка y модуля CAN
(LISTy) – адрес 0900h + y*4h (y = 0÷7)

Биты 31-25 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 24 EMPTY. Индикатор пустого списка. Бит EMPTY является индикатором того, что список пуст.
0 = По крайней мере, один элемент присутствует в списке.
1 = Список пуст.

Бит 23-16 SIZE7 – SIZE0. Размер списка. Количество областей сообщений, занесенных в список. Значение SIZE на единицу меньше количе-

ства элементов списка. Если список пуст, значение SIZE равно «0_H».

Бит 15-8 END7 – END0. Номер последнего элемента списка. Номер области сообщения, расположенной в конце списка.

Бит 7-0 BEGIN7 – BEGIN0. Номер первого элемента списка. Номер области сообщения, расположенной в начале списка.

На рисунке 300 представлен вариант, когда области сообщений с номерами 3, 5 и 16 занесены в список № 2, принадлежащий CAN узлу 1 (соответствующий регистр списка – LIST2).

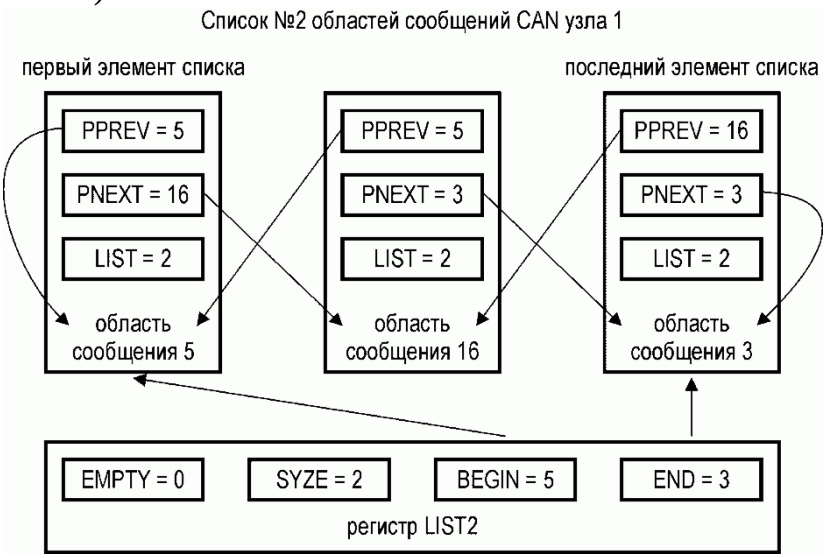


Рисунок 300 – Пример списка областей сообщений

Значение битового поля BEGIN регистра LIST(x+1) указывает на первый элемент списка (к примеру, на рисунке 300 это область сообщения 5). Значение битового поля END указывает на последний элемент списка, на рисунке 300 это область сообщения 3. Количество элементов списка (количество областей сообщений в списке) отражается в битовом поле SIZE (значение SIZE на единицу меньше количества элементов списка – так в нашем случае на рисунке 300, SYZE = 2_{16} для трех областей сообщений). Бит EMPTY является индикатором заполнения списка. Если EMPTY = 1_2, значит лист пустой (в случае на рисунке 300, EMPTY = 0_2, потому что список не пустой).

Каждая область сообщения n имеет указатель (битовое поле PNEXT регистра состояния области сообщения n MOSTATn) на следующую по списку область сообщения, и указатель (битовое поле PPREV регистра состояния области сообщения MOSTATn) на предыдущую по списку область сообщения.

PPREV первой по списку области сообщения указывает на эту же область, в примере на рисунке 300 область сообщения 5 является первой в списке и поэтому ее указатель PPREV = 5_{16}.

PNEXT последней по списку области сообщения указывает на эту же область, в примере на рисунке 300 область сообщения 3 является последней в списке и поэтому ее указатель PNEXT = 3_{16}.

Битовое поле LIST регистра MOSTATn указывает номер списка, к которому относится область сообщения n. Для случая, рассмотренного на рисунке 300, значение LIST всех трех областей сообщений будет равно 2.

Если битовому полю LIST регистра MOSTATn области сообщения n присвоено значение 0, то эта область относится к списку № 0 нераспределенных областей сообщений.

После сброса все области сообщений считаются нераспределенными и битовые поля LIST, в соответствующих им регистрах, равны 0. По умолчанию, порядок элементов списка № 0 следующий: область сообщений (n – 1) является предыдущей области сообщения n, а область сообщения (n + 1) – последующей.

Если предположить, что все области сообщения занесены в один список в порядке возрастания их номеров, то значения битовых полей PNEXT и PPREV регистров MOSTATn будут соответствовать указанным значениям в таблице 147.

Таблица 147 – Пояснительная таблица к описанию регистра MOSTATn

Область сообщения	PNEXT	PPREV	Значение после сброса
1	2	3	4
0	1_H	0_H	$0100\ 0000_H$
1	2_H	0_H	$0200\ 0000_H$
2	3_H	1_H	$0301\ 0000_H$
3	4_H	2_H	$0402\ 0000_H$
4	5_H	3_H	$0503\ 0000_H$
5	6_H	4_H	$0604\ 0000_H$
6	7_H	5_H	$0705\ 0000_H$
7	8_H	6_H	$0806\ 0000_H$
8	9_H	7_H	$0907\ 0000_H$
9	10_H	8_H	$0A08\ 0000_H$
10	11_H	9_H	$0B09\ 0000_H$
11	12_H	10_H	$0C0A\ 0000_H$
12	13_H	11_H	$0D0B\ 0000_H$
13	14_H	12_H	$0E0C\ 0000_H$
14	15_H	13_H	$0F0D\ 0000_H$
15	16_H	14_H	$100E\ 0000_H$
16	17_H	15_H	$111F\ 0000_H$
17	18_H	16_H	$1210\ 0000_H$
18	19_H	17_H	$1311\ 0000_H$
19	20_H	18_H	$1412\ 0000_H$
20	21_H	19_H	$1513\ 0000_H$
21	22_H	20_H	$1614\ 0000_H$
22	23_H	21_H	$1715\ 0000_H$
23	24_H	22_H	$1816\ 0000_H$
24	25_H	23_H	$1917\ 0000_H$

Окончание таблицы 147

1	2	3	4
25	26_H	24_H	$1A18\ 0000_H$

26	27 $_{н}$	25 $_{н}$	1B19 0000$_{H}$
27	28 $_{н}$	26 $_{н}$	1C1A 0000$_{H}$
28	29 $_{н}$	27 $_{н}$	1D1B 0000$_{H}$
29	30 $_{н}$	28 $_{н}$	1E1C 0000$_{H}$
30	31 $_{н}$	29 $_{н}$	1F1D 0000$_{H}$
31	32 $_{н}$	30 $_{н}$	201E 0000$_{H}$
32	33 $_{н}$	31 $_{н}$	211F 0000$_{H}$
33	34 $_{н}$	32 $_{н}$	2220 0000$_{H}$
34	35 $_{н}$	33 $_{н}$	2321 0000$_{H}$
35	36 $_{н}$	34 $_{н}$	2422 0000$_{H}$
36	37 $_{н}$	35 $_{н}$	2523 0000$_{H}$
37	38 $_{н}$	36 $_{н}$	2624 0000$_{H}$
38	39 $_{н}$	37 $_{н}$	2725 0000$_{H}$
39	40 $_{н}$	38 $_{н}$	2826 0000$_{H}$
40	41 $_{н}$	39 $_{н}$	2927 0000$_{H}$
41	42 $_{н}$	40 $_{н}$	2A28 0000$_{H}$
42	43 $_{н}$	41 $_{н}$	2B29 0000$_{H}$
43	44 $_{н}$	42 $_{н}$	2C2A 0000$_{H}$
44	45 $_{н}$	43 $_{н}$	2D2B 0000$_{H}$
45	46 $_{н}$	44 $_{н}$	2E2C 0000$_{H}$
46	47 $_{н}$	45 $_{н}$	2F2B 0000$_{H}$
47	48 $_{н}$	46 $_{н}$	302E 0000$_{H}$
48	49 $_{н}$	47 $_{н}$	312F 0000$_{H}$
49	50 $_{н}$	48 $_{н}$	3230 0000$_{H}$
50	51 $_{н}$	49 $_{н}$	3331 0000$_{H}$
51	52 $_{н}$	50 $_{н}$	3432 0000$_{H}$
52	53 $_{н}$	51 $_{н}$	3533 0000$_{H}$
53	54 $_{н}$	52 $_{н}$	3634 0000$_{H}$
54	55 $_{н}$	53 $_{н}$	3735 0000$_{H}$
55	56 $_{н}$	54 $_{н}$	3836 0000$_{H}$
56	57 $_{н}$	55 $_{н}$	3937 0000$_{H}$
57	58 $_{н}$	56 $_{н}$	3A38 0000$_{H}$
58	59 $_{н}$	57 $_{н}$	3B39 0000$_{H}$
59	60 $_{н}$	58 $_{н}$	3C3A 0000$_{H}$
60	61 $_{н}$	59 $_{н}$	3D3B 0000$_{H}$
61	62 $_{н}$	60 $_{н}$	3E3C 0000$_{H}$
62	63 $_{н}$	61 $_{н}$	3F3D 0000$_{H}$
63	63 $_{н}$	62 $_{н}$	3F3E 0000$_{H}$

Подключение списков областей сообщений

CAN узел может оперировать только с теми областями сообщений, которые занесены в принадлежащий ему список. Так CAN узлу 0 принадлежит список № 1, а CAN узлу 1 принадлежит список № 2.

В модуле CAN имеется восемь списков, которые показаны на рисунке 301:

- список № 0 нераспределенных сообщений;

- список № 1 CAN узла 0;
- список № 2 CAN узла 1;
- списки № 3 – № 7 – свободные списки, не принадлежащие ни одному из двух CAN узлов.

Области сообщений, распределенные в списки № 3 – № 7, не могут быть использованы CAN узлами.

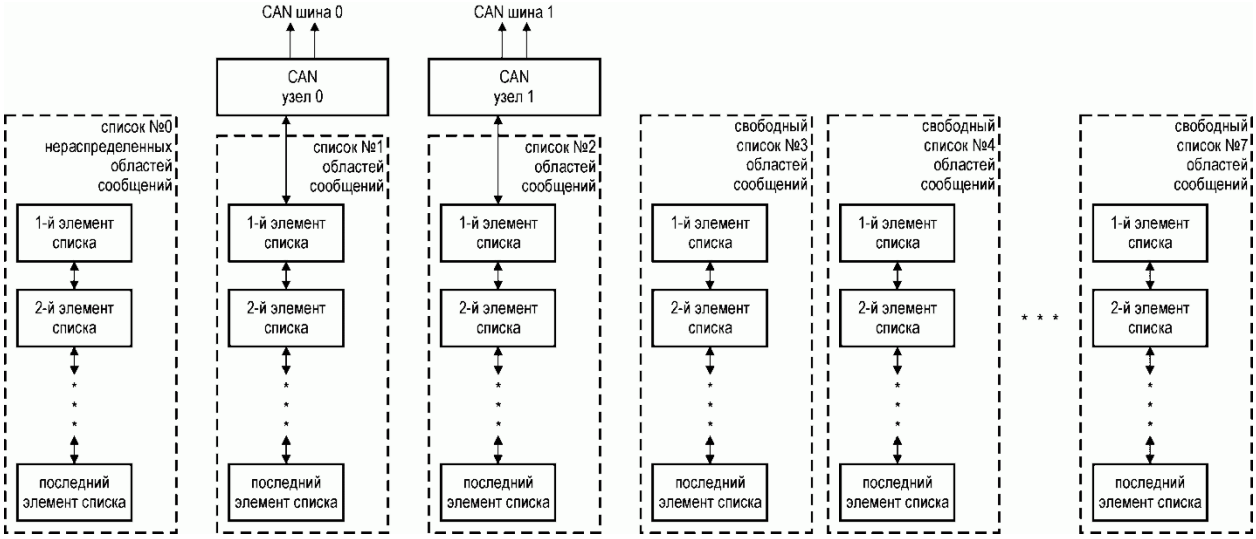


Рисунок 301 – Списки модуля CAN

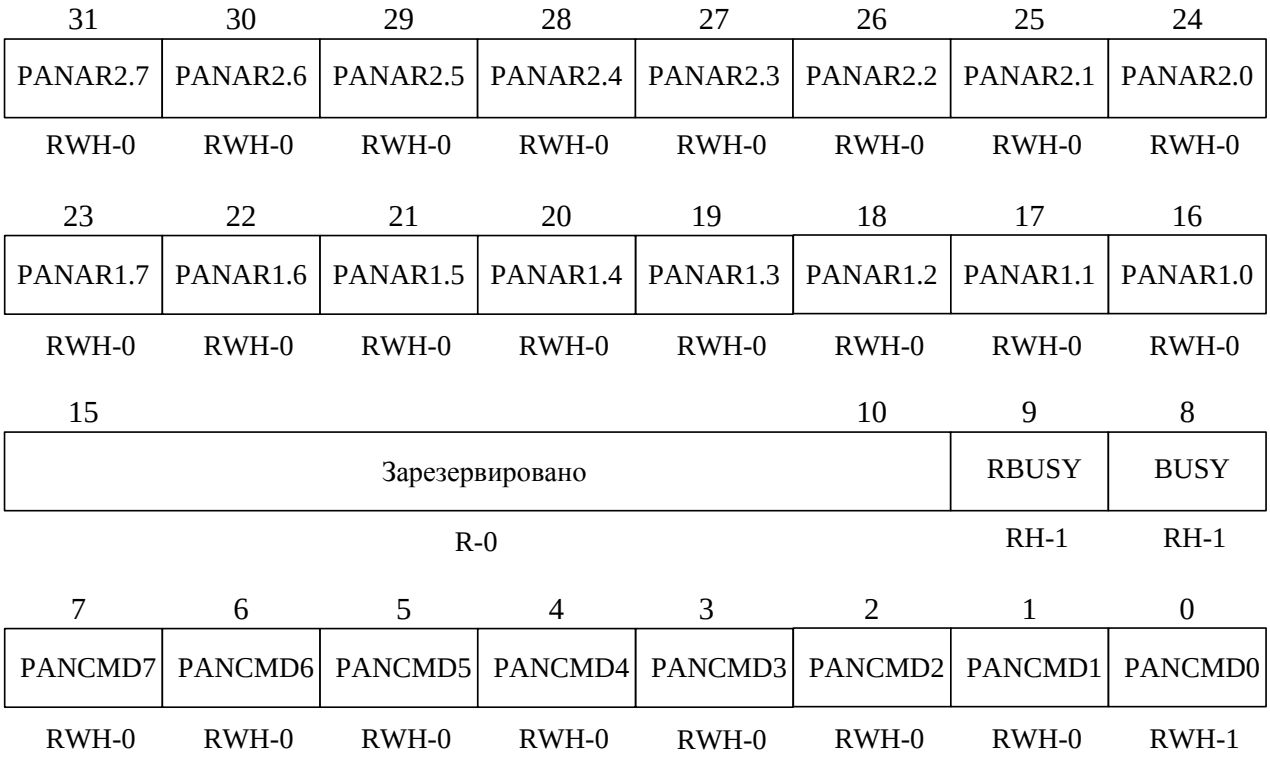
Механизмы FIFO и шлюза оперируют с областями сообщений, независимо от их распределения по спискам, что, ко всему прочему, дает возможность работы со всеми восемью списками. Следовательно, при использовании механизмов FIFO и шлюза следует следить за тем, действительно ли используемые области сообщений принадлежат желаемым спискам.

16.6.12 Панель команд списка

Для просмотра структуры списка областей сообщений CAN узла достаточно обратиться к соответствующим регистрам LIST(x+1) и MOSTATn.

Структура списка управляется и изменяется посредством контроллера списка, который, в свою очередь, управляется панелью команд, основное назначение которой – упрощение внесения изменений в структуру списка, отслеживание этих изменений и проверка их корректности.

Панель команд имеет регистр PANCTR, полностью определяющий действия контроллера списка по построению и/или реорганизации списков областей сообщений. Описание бит регистра панель команд PANCTR представлено на рисунке 302.



R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 302 –Регистр панель команд модуля CAN (PANCTR) – адрес 09C4h

Биты 31-24 PANAR2.7 - PANAR2.0. Панель аргумента 2. Описание в таблице 148.

Биты 23-16 PANAR1.7 – PANAR1.0. Панель аргумента 1. Описание в таблице 148.

Биты 15-10 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 9 RBUSY. Флаг занятости панелей аргументов.
0 = Изменение битовых полей PANAR1 и PANAR2 контроллером списка не планируется.
1 = Выполняется команда списка BUSY = 1_2, результаты выполнения которой будут записаны в битовые поля PANAR1 и PANAR2, но пока не доступны.

Бит 8 BUSY. Флаг занятости панели.
0 = Панель занята, выполняется команда.
1 = Панель готова для записи команды.

Биты 7-0 PANCMD7 – PANCMD0. Панель команд. Это битовое поле предназначено для записи кодов команд. После выполнения команды код «нет операции» автоматически записывается в PANCMD. Допусти-

мые коды команд сведены в таблицу 148.

Операция панели состоит из кода команды PANCMD и двух панелей аргументов PANAR1 и PANAR2. Допустимые коды команды PANCMD представлены в таблице 148. Команды, которые имеют возвращаемое значение, записывают это значение в битовое поле PANAR1. Команды, которые возвращают флаг ошибки, записывают его в 31-й бит регистра PANCTR (7-й бит битового поля PANAR2).

Таблица 148 – Допустимые коды команды PANCMD

PANCMD	PANAR2	PANAR1	Описание команды
1	2	3	4
00_{16}	–	–	Нет операции. Запись 00_{16} безрезультатна. Никаких действий не предпринимается.
01_{16}	Результат: 7-й бит – ошибка, 6-й бит – не определен	–	Инициализация списков. Запуск инициализации для очистки битовых полей CTRL и LIST всех областей сообщений. Регистры списков LIST0 – LIST8 устанавливаются в свои значения после сброса. Это приводит к переносу всех областей сообщений в список № 0 нераспределенных областей сообщений. Инициализация списков требует, чтобы биты INIT и CCE регистра NCRx были установлены для обоих CAN узлов. 7-й бит битового поля PANAR2 сигнализирует о результате операции: 0 = Инициализация завершена успешно. 1 = Не все биты INIT и CCE были установлены. Инициализация не завершена. Команда инициализации списков автоматически запускается при каждом сбросе модуля CAN, за исключением случая, когда все регистры областей сообщений уже сброшены

Продолжение таблицы 148

1	2	3	4
02_{16}	Аргу-	Аргу-	Статическое распределение области сообще-

	мент: номер списка	мент: номер области сообщения	ния. Независимо от того, какому списку принадлежала область сообщения, она переносится в указанный список (битовое поле PANAR2) и добавляется в его конец. Эта команда так же используется для дераспределения (перенос области сообщения в список №0 нераспределенных областей сообщений) области сообщения. Для этого значение в битовом поле PANAR2 должно быть равно 0_{16}
03_{16}	Аргумент: номер списка Результат: 7-й бит – ошибка, 6-й бит – не определен	Результат: номер области сообщения	Динамическое распределение области сообщения. Занесение области сообщения, являющейся первым элементом списка №0 нераспределенных областей сообщений в указанный (битовое поле PANAR2) список. Область сообщения добавляется в конец списка. Номер (по списку) добавленной области сообщения возвращается в битовом поле PANAR1. 7-й бит битового поля PANAR2 сигнализирует о результате операции: 0 = Занесение в список прошло успешно. 1 = Команда не выполнена, поскольку список №0 пуст
04_{16}[1)]	Аргумент: новый номер области сообщения	Аргумент: текущий номер области сообщения	Перемещение до указанной позиции. Перенос области сообщения (номер области указывается в поле PANAR1) из текущей позиции на позицию, предшествующую занимаемой области сообщения, указанной в битовом поле PANAR2

Продолжение таблицы 148

1	2	3	4
05_{16}[2)]	Аргу-	Резуль-	Занесение в список до указанной позиции.

	мент: новый номер области сообщения Результат: 7-й бит– ошибка, 6-й бит – не определен	тат: номер добавленной области сообщения	Перенос области сообщения из списка № 0 (выбирается первый по списку элемент) на позицию, предшествующую занимаемой области сообщения, указанной в битовом поле PANAR2. Номер добавленной области сообщения возвращается в битовом поле PANAR1. 7-й бит битового поля PANAR2 сигнализирует о результате операции: 0 = Занесение в список прошло успешно. 1 = Команда не выполнена, поскольку список № 0 пуст
06_{16}[1)]	Аргумент: новый номер области сообщения	Аргумент: текущий номер области сообщения	Перемещение на позицию, после указанной позиции. Перенос области сообщения (номер области указывается в поле PANAR1) из текущей позиции на позицию, следующую за занимаемой областью сообщения с номером, указанным в PANAR2
07_{16}[2)]	Аргумент: новый номер области сообщения Результат: 7-й бит – ошибка, 6-й бит – не определен	Результат: номер добавленной области сообщения	Занесение в список на позицию, после указанной позиции. Перенос области сообщения из списка № 0 (выбирается первый по списку элемент) на позицию, следующую за занимаемой областью сообщения, с номером указанным в битовом поле PANAR2. Номер добавленной области сообщения возвращается в битовом поле PANAR1. 7-й бит битового поля PANAR2 сигнализирует о результате операции: 0 = Занесение в список прошло успешно. 1 = Команда не выполнена, поскольку список №0 пуст
08_{16} – FF_{16}	–	–	Зарезервировано

Окончание таблицы 148

[1)] Перемещаемая область сообщения и область сообщения до/после пози-

ции которой происходит перемещение, могут принадлежать как одному списку, так и разным. В случае если области сообщений принадлежат разным спискам, то исключение элемента из одного списка и занесение в другой происходит автоматически. Для перемещения области сообщений достаточно знать ее номер (0, …, 255).

[2)] Для перемещения области сообщения из списка № 0 достаточно знать номер области сообщения, до/после позиции которой происходит перемещение. Занесение нового элемента в список, которому принадлежит область сообщения, до/после позиции которой происходит перемещение, осуществляется автоматически, с выдачей номера области сообщения (0, …, 255), заносимой в список.

Панель команд запускается записью соответствующей команды в битовое поле PANCMD регистра PANCTR. Соответствующие аргументы команд должны быть записаны в битовые поля PANAR1 и PANAR2 до записи кода команды в поле PANCMD.

Примечание – Запись новых значений в битовые поля PANAR1 и PANAR2 не изменяет сразу их содержимого. Новые значения сначала попадают в специальный теневой регистр. Далее, одновременно с записью кода команды в битовое поле PANCMD, новые значения, хранящиеся в теневом регистре, переносятся в битовые поля PANAR1 и PANAR2.

С записью корректного кода команды выставляется флаг BUSY регистра PANCTR и в дальнейшем все попытки записи в регистр PANCTR игнорируются. Флаг BUSY остается активным, а панель команд заблокированной до тех пор, пока не завершится выполнение записанной команды.

После сброса, контроллер списка формирует список № 0 нераспределенных областей сообщений. Во время этой операции флаг BUSY = 1_2 и все обращения к ОЗУ областей сообщений запрещены. ОЗУ становиться доступным только после сброса флага BUSY.

Примечание – ОЗУ областей сообщений автоматически инициализируется после сброса контроллером списка для обеспечения каждой области сообщения корректным указателем на список. По окончании этой операции флаг BUSY сбрасывается.

В случае появления команды динамического распределения, по которой какой-либо элемент забирается из списка № 0 и переноситься в другой указанный список, наряду с битом BUSY устанавливается бит RBUSY (RBUSY = BUSY = 1_2). Это указывает на то, что значения битовых полей PANAR1 и PANAR2 будут обновлены контроллером списка, следующим образом:

- номер области сообщения, забираемой из списка № 0 нераспределенных областей сообщений, записывается в PANAR1;

- если бит ERR = 1_2 (7-й бит поля PANAR2), значит, список № 0 пуст и выполнение команды завершается; если бит ERR = 0_2, значит, список № 0 не пуст и команда выполняется.

Результаты выполнения команды динамического распределения записываются до того, как контроллер списка начнет процесс распределения. Как только результаты станут доступны, бит RBUSY снова становиться равным «0» (т. е. неактивным), в то время как бит BUSY остается активным (= 1_2) до завершения выполнения команды. Это позволяет пользователю запрограммировать настройки

желаемой области сообщения, в то время как контроллер списка распределяет области. Во время операций со списками доступ к областям сообщений не ограничен, но следует помнить, что любой доступ к регистрам ОЗУ областей сообщений в течение процесса распределения областей вносит задержку (в процесс), равную длительности доступа.

По завершении процесса выполнения команды, флаг BUSY сбрасывается и запись в регистр PANCTR снова становиться возможной.

Код команды «нет операции» автоматически записывается в битовое поле PANCMD.

Новая команда может быть записана в любое время, когда бит BUSY = 0_2.

Все битовые поля регистра PANCTR, исключая биты BUSY и RBUSY, могут быть записаны программно, что делает возможным сохранять и восстанавливать значения регистра PANCTR, если панель команд используется независимой подпрограммой обработки прерываний. Если возникает такая ситуация, то любые задачи, которые используют панель команд и которые могут прерывать выполнение других задач, тоже использующих панель команд, будут опрашивать состояние флага BUSY. До тех пор, пока флаг BUSY будет оставаться активным, содержимое регистра PANCTR будет сохранено в соответствующей области памяти до операции восстановления. Как только подпрограмма обработки прерываний закончится, содержимое регистра PANCTR будет восстановлено.

До того, как область сообщения, занесенная в список активного CAN узла, будет перенесена на другую позицию этого же списка или перенесена в другой список, бит MSGVAL регистра MOSTATn области сообщения n должен быть очищен.

16.7 Анализ работы CAN узла

Доступны три режима анализа:
- режим общего анализа;
- режим внутренней петли;
- режим анализа синхронизации битов.

16.7.1 Режим общего анализа

Этот режим позволяет осуществлять независимый мониторинг работы CAN узла, не затрагивая CAN шину. Режим общего анализа выбирается битом CALM регистра NCRx.

В режиме общего анализа выходы CAN узла находятся в рецессивном состоянии. Узел может получать сообщения данных, сообщения удаленных запросов и сообщения об ошибках, но работа узла на передачу запрещена. Полученные сообщения данных/удаленных запросов остаются без подтверждения (бит подтверждения остается в рецессивном состоянии), но принимаются и сохраняются (при совпадении идентификаторов) в соответствующих областях сообщений.

16.7.2 Режим внутренней петли

Этот режим позволяет проводить внутреннее тестирование модуля CAN, а также отладку управляющей программы без доступа к внешней CAN шине.

Внутренняя петля состоит из внутренней CAN шины (внутри модуля CAN) и переключателя выбора шины для каждого CAN узла, которые показаны на рисунке 303. С помощью переключателя каждый CAN узел может быть подключен либо к внутренней CAN шине (режим внутренней петли) или к внешней CAN шине (нормальный режим работы). Если выбран режим внутренней петли, то на внешнем передающем выводе CAN узла поддерживается рецессивный уровень сигнала, а состояние принимающего вывода игнорируется.

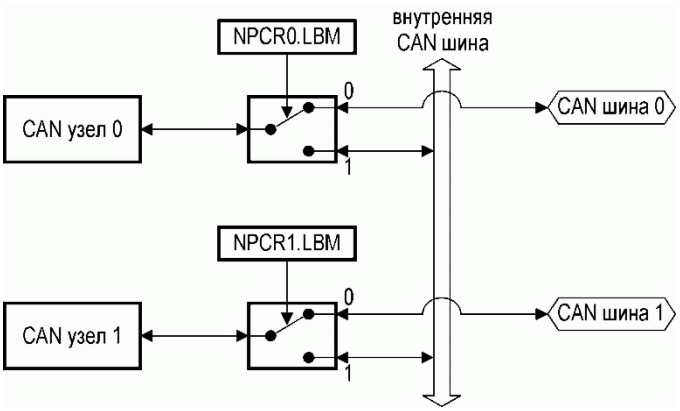


Рисунок 303 – Режим внутренней петли

Режим внутренней петли для CAN узла может быть активирован битом LBM регистра NPCRx.

Если оба CAN узла функционируют в режиме внутренней петли, они взаимодействуют друг с другом посредством внутренней CAN шины.

16.7.3 Режим анализа синхронизации битов

Детальный анализ синхронизации битов для CAN узла может быть осуществлен посредством режима анализа счетчика сообщений этого CAN узла. Анализ синхронизации при помощи счетчика сообщений может быть использован для автоматического детектирования потока данных, а также для временного анализа CAN шины.

Режим анализа синхронизации битов для CAN узла x выбирается, когда CFMOD = 10_2 (регистр NFCRx). Режим анализа синхронизации битов не оказывает влияния на работу CAN узла.

Результаты измерений в режиме анализа синхронизации битов записываются в битовое поле CFC регистра NFCRx. Каждый раз, когда битовое поле CFC изменяется (в режиме анализа синхронизации битов), устанавливается флаг CFCOV регистра NFCRx. Если разрешено и флаг CFCIE регистра NFCRx установлен, может быть сгенерирован запрос на прерывание.

Автоматический контроль скорости потока битов

Для автоматического контроля скорости потока битов на CAN шине необходимо измерять время между появлениями фронтов передаваемых доминантных

битов. Эти измерения осуществляются автоматически, если CFSEL = 000_B (регистр NFCRx). Время (измеренное в тактах частоты f_{CLC}) между появлениями фронтов двух доминантных битов сохраняется в битовом поле CFC.

Анализ синхронизации

Анализ синхронизации времени бита осуществляется, если CFSEL = 010_B (регистр NFCRx). Время между фронтом первого доминантного сигнала и точкой выборки измеряется и сохраняется в битовом поле CFC. Смещение синхронизации бита может быть получено из этого времени, как только появится первый фронт сигнала после точки выборки. Существует только одна синхронизация между последовательными точками выборки.

Анализ синхронизации, например, может быть использован во время приема первого сообщения для более точной настройки скорости потока битов.

Измерение задержки драйвера

Задержка между появлением на шине переднего фронта сигнала передаваемого бита и приемом этого фронта измеряется, когда CFSEL = 011_B (доминантный – доминантный) и CFSEL = 100_B (рецессивный – рецессивный). Эта задержка показывает, какое время необходимо для определения значения нового бита в зависимости от физической реализации CAN шины.

16.8 Фильтрация сообщений

Фильтрация используется в модуле CAN для контроля приема и передачи сообщений. Описание бит регистра маски области сообщения MOAMRn представлено на рисунке 304. Описание бит регистра арбитража области сообщения MOARn представлено на рисунке 305. Распределение приоритетов областей сообщений на основе правил арбитража представлено в таблице 149.

31 30	29	28	27	26	25	24
Зарезервировано	MIDE	AM28	AM27	AM26	AM25	AM24
RW-0	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1

23	22	21	20	19	18	17	16
AM23	AM22	AM21	AM20	AM19	AM18	AM17	AM16
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1

15	14	13	12	11	10	9	8
AM15	AM14	AM13	AM12	AM11	AM10	AM9	AM8
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1

7	6	5	4	3	2	1	0
AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1

R – доступ чтения, W – доступ записи, -1 – значение после сброса

Рисунок 304 – Регистр маски области сообщения n модуля CAN (MOAMRn) – адрес 180Ch + n*20h (n = 0÷255)

Биты 31-30 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 29 MIDE. Маска бита IDE.
0 = Область сообщения n может получать как стандартные, так и расширенные сообщения.
1 = Область сообщения n может получать только те сообщения, в которых IDE = 1_2.

Биты 28-0 AM28 – AM0. Маска идентификатора. 29 битная маска идентификатора для фильтрации принимаемых сообщений. В случае приема стандартного идентификатора используется битовое поле AM28 – AM8 (при этом состояние битов AM17 – AM0 любое). В случае приема расширенного идентификатора используется битовое поле AM28 – AM0.

31	30	29	28	27	26	25	24
PRI1	PRI0	IDE	ID28	ID27	ID26	ID25	ID24
RW-0	RW-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
23	22	21	20	19	18	17	16
ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
15	14	13	12	11	10	9	8
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
7	6	5	4	3	2	1	0
ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 305 – Регистр арбитража области сообщения n модуля CAN (MOARn) – адрес 1818h + n*20h (n = 0÷255)

Биты 31-30 PRI1, PRI0. Класс приоритета. Это битовое поле определяет один из четырех классов приоритета для области сообщения. Меньший номер класса, указанный в PRI, соответствует большему приоритету. В случае если несколько областей сообщений имеют один класс приоритета, тогда приоритет среди них определяется согласно значениям идентификаторов и позициям в списке. Помимо прочего, битовое поле PRI определяет метод фильтрации для передачи сообщений.

00 = Зарезервировано.

01 = Фильтрация на основе положения в списке. Область сообщения имеет приоритет для передачи/приема в случае, если выше по списку нет других областей сообщений, ждущих передачу MSGVAL & TXEN0 & TXEN1 = 1_2.

10 = Фильтрация на основе значения идентификатора. Область сообщения имеет приоритет (для передачи/приема) в случае, если в списке нет других областей сообщений с более высоким приоритетом, определяемым полем «ID + IDE + DIR», согласно правилам арбитража CAN шины (подробнее в таблице 150).

11 = Фильтрация на основе положения в списке (аналогично PRI = 01_2).

Бит 29 IDE. Бит расширения идентификатора.
0 = Область сообщения поддерживает стандартные сообщения с 11-битными идентификаторами.
1 = Область сообщения поддерживает расширенные сообщения с 29-битными идентификаторами.

Бит 28-0 ID28 – ID0. Идентификатор области сообщения. Битовое поле хранит идентификатор (стандартный или расширенный) сообщения. В случае стандартного идентификатора используются биты ID28 – ID18 (при этом состояние битов ID17 – ID0 любое). В случае расширенного идентификатора используются биты ID28 – ID0.

Таблица 149 – Распределение приоритетов областей сообщений на основе правил арбитража

Установка приоритетов областей сообщений A и B (A имеет больший приоритет, чем B)	Пояснение
1	2
A.MOAR28–A.MOAR18 < B.MOAR28–B.MOAR18 (11-битный стандартный идентификатор области A меньше по числовому значению, чем 11-битный стандартный идентификатор области B)	Сообщение со стандартным идентификатором, имеющим меньшее значение, обладает более высоким приоритетом. MOAR28 – старший бит стандартного идентификатора (в случае 11 битного идентификатора). MOAR18 – младший бит стандартного идентификатора
A.MOAR28–A.MOAR18 = B.MOAR28–B.MOAR18 A.MOAR.IDE = 0_2 – стандартный идентификатор B.MOAR.IDE = 1_2 – расширенный идентификатор	При равенстве значений идентификаторов, стандартный идентификатор имеет приоритет перед расширенным идентификатором
A.MOAR28–A.MOAR18 = B.MOAR28–B.MOAR18 A.MOAR.IDE = B.MOAR.IDE = 0_2 A.MOAR.DIR = 1_2 – сообщение данных B.MOAR.DIR = 0_2 – сообщение удаленного запроса	При равенстве значений идентификаторов, стандартное сообщение данных имеет приоритет перед стандартным сообщением удаленного запроса
A.MOAR28–A.MOAR18 = B.MOAR28–B.MOAR18 A.MOAR.IDE = B.MOAR.IDE = 1_2 A.MOAR.DIR = 1_2 – сообщение данных B.MOAR.DIR = 0_2 – сообщение удаленного запроса	При равенстве значений идентификаторов, расширенное сообщение данных имеет приоритет перед расширенным сообщением удаленного запроса

Окончание таблицы 149

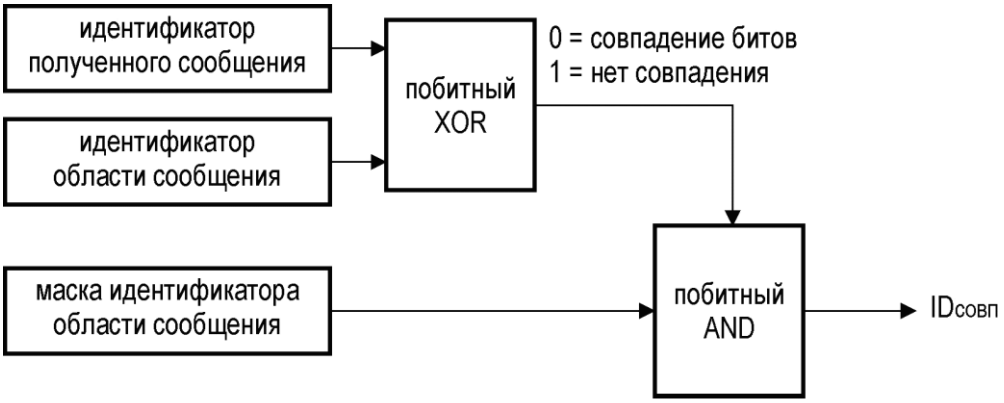
1	2
A.MOAR28–A.MOAR0 < B.MOAR28 – B.MOAR0 A.MOAR.IDE = B.MOAR.IDE = 1_2 – 29-битный идентификатор	Сообщение с расширенным идентификатором, имеющим меньшее значение, обладает более высоким приоритетом. MOAR.28 – старший бит расширенного идентификатора в случае 29-битного идентификатора, состоящего из стандартного идентификатора MOAR.28 – MOAR.18 и расширения MOAR.17 – MOAR.0. MOAR.0 – младший бит расширенного идентификатора

16.8.1 Фильтрация при получении сообщений

При получении CAN узлом сообщения, определяется область сообщения, в которой будут сохранены получаемые данные в случае успешного приема.

Область сообщения считается корректной для приема, если одновременно соблюдаются условия:

- область сообщения распределена в список областей сообщений CAN узла, которым принимается сообщение;
- флаг MSGVAL установлен (регистр MOSTATn);
- флаг RXEN установлен (регистр MOSTATn);
- бит DIR (регистр MOSTATn) равен биту RTR принимаемого сообщения. Если бит DIR = 1_2 (область-для-передачи), область сообщения может принять только сообщение удаленного запроса. Если бит DIR = 0_2 (область-для-приема), область сообщения может принять только сообщение данных;
- если бит MIDE = 1_2 (регистр MOAMRn), бит IDE получаемого сообщения оказывает следующее влияние:
  - если IDE = 1_2 (регистра MOARn), бит IDE принимаемого сообщения должен быть равен «1» (указатель расширенного идентификатора);
  - если IDE = 0_2 (регистра MOARn), бит IDE принимаемого сообщения должен быть равен «0» (указатель стандартного идентификатора);
  - если бит MIDE = 0_2 (регистр MOAMRn), значение бита IDE принимаемого сообщения неважно. В этом случае допускаются сообщения как со стандартным, так и с расширенным идентификатором;
- идентификатор полученного сообщения полностью (побитно) совпадает с идентификатором, хранящимся в регистре MOARn области сообщения n, за исключением битов, закрытых маской регистра MOAMRn, значение которых не важно. На рисунке 306 показан пример проверки идентификатора.



Примечание -
IDсовп = 0: идентификатор ID полученного сообщения совпал с ID области сообщения.
IDсовп > 0: идентификатор ID полученного сообщения не совпал с ID области сообщения.

Рисунок 306 – Проверка идентификатора полученного сообщения

Среди всех областей сообщений, которые отвечают указанным выше критериям, для сохранения полученного сообщения выбирается область с наивысшим приоритетом.

Для областей сообщений установлена схема определения приоритета.

Область сообщения A (далее здесь OCA) имеет приоритет над областью сообщения B (OCB), если выполняются следующие условия:

- OCA имеет более высокий класс приоритета, чем OCB (MOARA.PRI ≤ MOARB.PRI);

- в списке областей сообщений OCA занимает позицию, предшествующую позиции OCB (в случае MOARA.PRI = MOARB.PRI).

16.8.2 Фильтрация при передаче сообщений

Когда требуется передача содержимого какой-либо области сообщения, в соответствующих управляющих регистрах выставляются флаги, указывающие на необходимость передачи. Может возникнуть ситуация, когда передачи требуют одновременно несколько областей сообщений. Для выбора области сообщения, содержимое которой будет передано в первую очередь, существует система приоритетов.

Область сообщения считается корректной для передачи, если одновременно соблюдаются условия:

- область сообщения распределена в список областей сообщений CAN узла;

- флаг MSGVAL установлен (регистр MOSTATn);

- флаг TXRQ установлен (регистр MOSTATn);

- флаги TXEN0 и TXEN1 установлены (регистр MOSTATn).

Среди всех областей сообщений, которые отвечают указанным выше критериям, для передачи выбирается область с наивысшим приоритетом.

Схема определения приоритета следующая.

Область сообщения A (далее здесь OCA) и область сообщения B (OCB) являются областями, одновременно требующими передачи. В списке областей сообщений OCA находится перед OCB.

Если OCA и OCB имеют один класс приоритета (MOARA.PRI = MOARB.PRI), OCA будет иметь приоритет над OCB, если будут соблюдаться следующие условия:

- PRI = 10_2 и сообщение, хранящееся в OCA, имеет приоритет (согласно правилам арбитража) над сообщением, хранящемся в OCB;

- PRI = 01_2 или PRI = 11_2 (приоритет по положению в списке).

Область сообщения, являющаяся корректной для передачи и имеющая приоритет, будет передана первой. Остальные области сообщений будут переданы по очереди, согласно их приоритетов, аналогично описанному выше. На рисунке 307 приведено формирование запроса на передачу области сообщения.

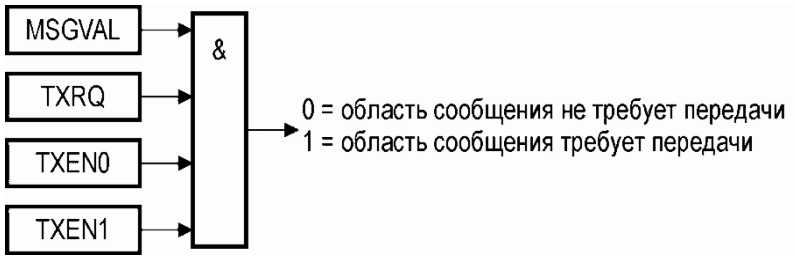


Рисунок 307 – Формирование запроса на передачу области сообщения

16.8.3 Заключительные операции при работе с сообщениями

После успешного прима или успешной передачи сообщения, ЦП получает оповещение о завершении операции для определения дальнейших действий, связанных с областью сообщения. Заключительная операция модуля CAN состоит из двух частей: формирование прерывания; сбор прерываний, ожидающих обработки в общую структуру.

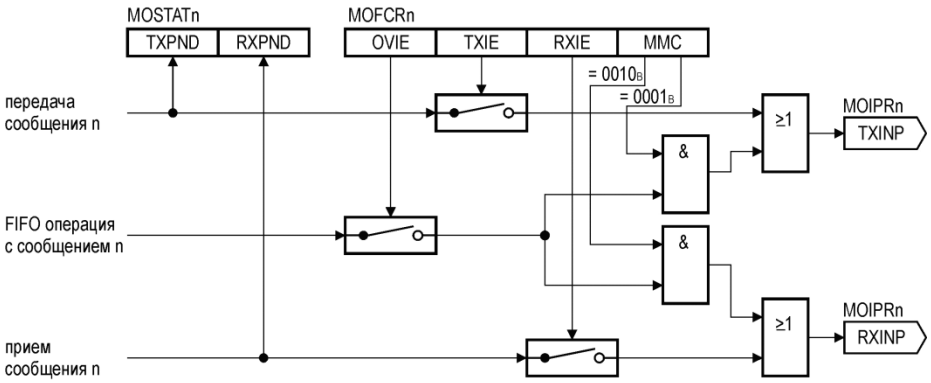
16.8.4 Прерывания областей сообщений

После успешного сохранения принятого сообщения в области сообщения или успешной передачи, формируется соответствующее прерывание. Каждая область сообщения может формировать прерывания. Каждое такое прерывание после распределения попадает на одну из 16 выходных линий прерываний. Прерывания приема (после сохранения сообщения) также формируются после операций FIFO и шлюзовых операций. Флаги TXPND и RXPND (MOSTATn) всегда устанавливаются после успешной операции передачи/приема, независимо от состояния соответствующих флагов разрешения прерываний.

Область сообщения может формировать FIFO прерывания. Если флаг OVIE регистра MOFCRn установлен, формирование FIFO прерывания будет зависеть от настоящего типа области сообщения.

В случае если область сообщения является базовой областью FIFO приема (MOFCRn.MMC = 0001_2), выходная линя прерываний (одна из 16) для этой области определяется битовым полем TXINP регистра MOIPRn.

В случае. если область сообщения является базовой областью FIFO передачи (MOFCRn.MMC = 0010_2), выходная линя прерываний (одна из 16) для этой области определяется битовым полем RXINP регистра MOIPRn, который показан на рисунке 308.



Примечание -
MMC = 0001_B: область сообщения n является базовой областью FIFO приема.
MMC = 0010_B: область сообщения n является базовой областью FIFO передачи.

Рисунок 308 – Распределение прерываний

16.8.5 Ждущие сообщения

Когда генерируется запрос на прерывание (после приема/передачи сообщения), в одном из двух регистров ждущих прерываний MSPND0 или MSPND1 выставляется флаг ждущего сообщения. Два 32 разрядных регистра образуют область из 64 битов – по два бита (один бит для операций приема и один бит для операций передачи) для каждой из областей сообщений. Позиция флага ждущего сообщения определяется двумя демультиплексорами.

На рисунке 309 верхний 1-битный демультиплексор выбирает один из двух регистров MSPND0 или MSPND1, а нижний 5-разрядный – позицию флага в выбранном регистре. Описание бит регистра управления MCR представлено на рисунке 310. Описание бит регистра ждущих прерываний MSPNDk представлено на рисунке 311.

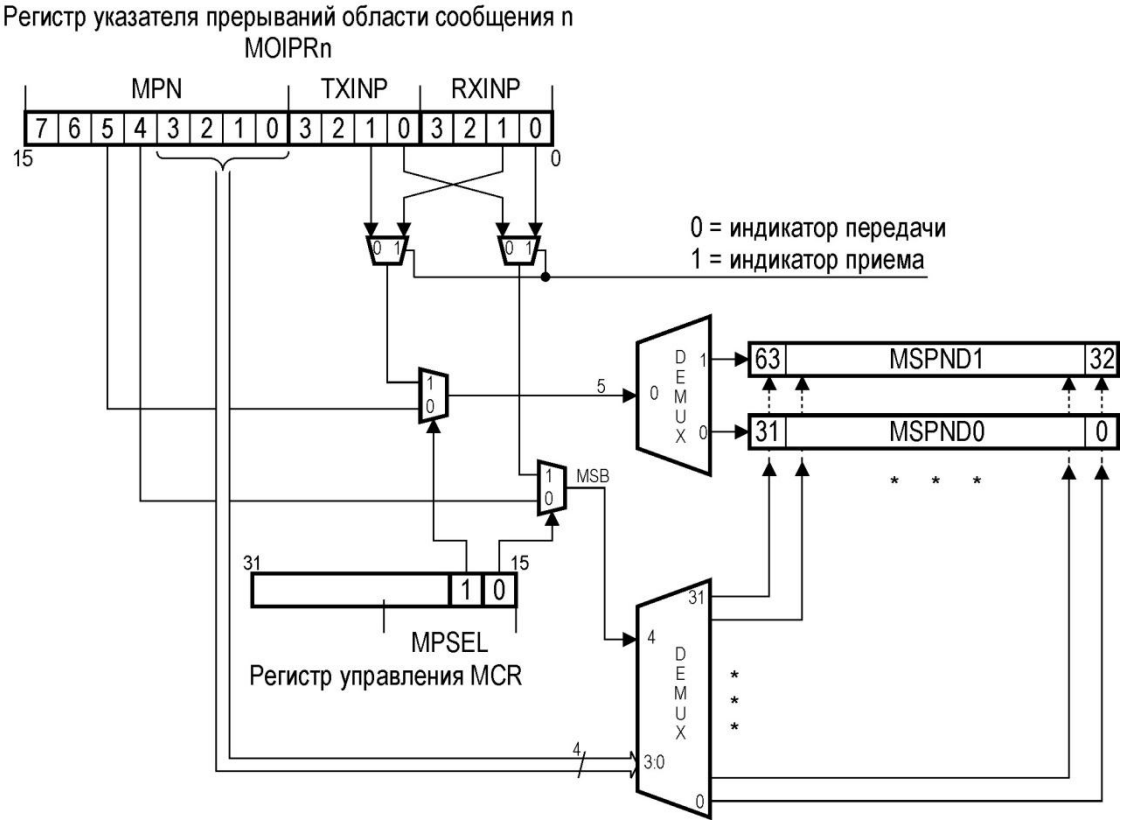


Рисунок 309 – Механизм выбора и установки флагов ждущих сообщений

31	16
Зарезервировано	
R-0	

15	14	13	12	11	0
MPSEL3	MPSEL2	MPSEL1	MPSEL0	Зарезервировано	
RW-0	RW-0	RW-0	RW-0	R-0	

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 310 – Регистр управления модулем CAN (MCR) – адрес 09C8h

Биты 31-16 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Биты 15-12 MPSEL3 – MPSEL0. Выбор ждущего сообщения. Это битовое поле позволяет выбрать позицию для бита ждущего сообщения после его приема/передачи комбинацией битовых полей RXINP, TXINP и MPN регистра MOIPRn.

Биты 11-0 Зарезервировано. При чтении возвращает «0». При записи следует

писать «0».

31	30	29	28	27	26	25	24
PND31	PND30	PND29	PND28	PND27	PND26	PND25	PND24
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
23	22	21	20	19	18	17	16
PND23	PND22	PND21	PND20	PND19	PND18	PND17	PND16
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
15	14	13	12	11	10	9	8
PND15	PND14	PND13	PND12	PND11	PND10	PND9	PND8
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
7	6	5	4	3	2	1	0
PND7	PND6	PND5	PND4	PND3	PND2	PND1	PND0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 311 – Регистр ждущих прерываний k модуля CAN (MSPNDk) – адрес 0940h + k*4h (k = 0÷7)

Биты 31-0 PND31 – PND0. Поле ожидания. Когда возникает запрос на прерывание, область сообщения n устанавливает соответствующий ей бит в поле PND одного из регистров MSPNDk. Выбор регистра MSPNDk определяется битом MPN, а позиция устанавливаемого бита определяется битами MPN4 – MPN0 регистра MOIPRn. Установленные биты могут быть очищены программно записью «0». Запись «1» в биты регистра игнорируется.

Каждому регистру MSPNDk соответствует регистр индекса сообщения MSIDk. Регистр MSIDk показывает индекс активного (установленного) бита ожидания, занимающего низшую из всех позиций активных битов поля PND регистров MSPNDk. Описание бит регистра индекса сообщений MSIDk представлено на рисунке 312.

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 312 – Регистр индекса сообщений k модуля CAN (MSIDk) – адрес 0980h + k*4h (k = 0÷7)

Биты 31-6 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Биты 5-0 INDEX5 – INDEX0. Индекс бита. Показывает индекс активного бита i, удовлетворяющего условиям:
- MSPNDk.i & IM.i = 1;
- i = 0 или MSPNDk.(i–1) – MSPNDk.0 & IM.(i–1)–IM.0 = 0_2.
Если в регистре MSPND0 или MSPND1 нет битов, удовлетворяющих этим условиям, тогда чтение битового поля INDEX возвращает значение 100000_2. Битовое поле INDEX показывает позицию первого бита ожидания регистра MSPND0 или MSPND1, в котором обслуживаются только те биты, которые не закрыты маской регистра MSIMASK.

Регистр маски индекса сообщения MSIMASK содержит маску для регистра MSPNDk, в котором обслуживаются только те биты, которые не закрыты маской.

Регистр MSIMASK используется для регистров MSPNDk и соответствующих им регистров MSIDk. Возможны два варианта определения позиции флага в регистрах MSPNDk. Описание бит регистра маски индекса сообщений MSIMASK представлен на рисунке 313.

31	30	29	28	27	26	25	24
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 313 – Регистр маски индекса сообщений модуля CAN (MSIMASK) – адрес 09C0h

Биты 31-0 IM. Маска индекса. При вычислении индекса бита во внимание принимаются только те биты регистра MSPNDk, для которых соответствующие биты поля IM равны «1».

В первом варианте битовое поле MPSEL = 0000_2 (регистр MCR) и установка флага ждущего сообщения происходит следующим образом:

- пятый бит поля MPN (MPN5) выбирает регистр MSPNDk, в котором будет установлен флаг ждущего сообщения;

- младшие 5 битов поля MPN (MPN4 – MPN0) с помощью демультиплексора выбирают позицию флага (31-0), который будет установлен в выбранном регистре.

Во втором варианте при определении позиции флага ждущего сообщения принимаются в расчет значения битового поля MPSEL (регистр MCR) и битовых полей указателей узлов прерываний (для приема – MOIPRn.RXINP, для передачи – MOIPRn.TXINP). При этом для флагов операций приема и передачи используются разные биты выбранного регистра MSPNDk.

Таким образом, при MPSEL = 1111_2, установка флага ждущего сообщения происходит следующим образом:

- при передаче первый бит поля TXINP (TXINP1 регистра MOIPRn) определяет регистр MSPNDk, в котором будет установлен флаг ждущего сообщения. При приеме регистр определяется первым битом поля RXINP (RXINP1 регистра MOIPRn);

- позиция флага (31-0) в выбранном регистре выбирается младшим битом поля TXINP (TXINP0 при передаче) или младшим битом поля RXINP (RXINP0 при приеме) и четырьмя младшими битами поля MPN (MPN3 – MPN0).

16.8.6 Общие замечания

Регистры MSPNDk могут быть записаны программно. Биты, в которые записываются «1», остаются без изменений, а биты, в которые записываются «0», очищаются. Такой механизм записи позволяет очищать любой бит регистра независимо и за один такт записи, что позволяет избежать конфликта между одновременной аппаратной установкой и программной очисткой битов регистра.

Каждый регистр MSPNDk связан с соответствующим регистром индекса сообщения MSIDk, который отражает позицию самого младшего бита из всех установленных (равных «1») в регистре MSPNDk. Регистры k доступны только для чтения и обновляются незамедлительно после изменения (аппаратного, либо программного) содержимого соответствующих регистров MSPNDk.

16.9 Операции с данными областей сообщений

16.9.1 Прием сообщения

После получения сообщение сохраняется в области сообщения в соответствии со схемой, показанной на рисунке 314. Помимо сохранения данных в области сообщения, модуль CAN осуществляет обмен данными с ЦП.

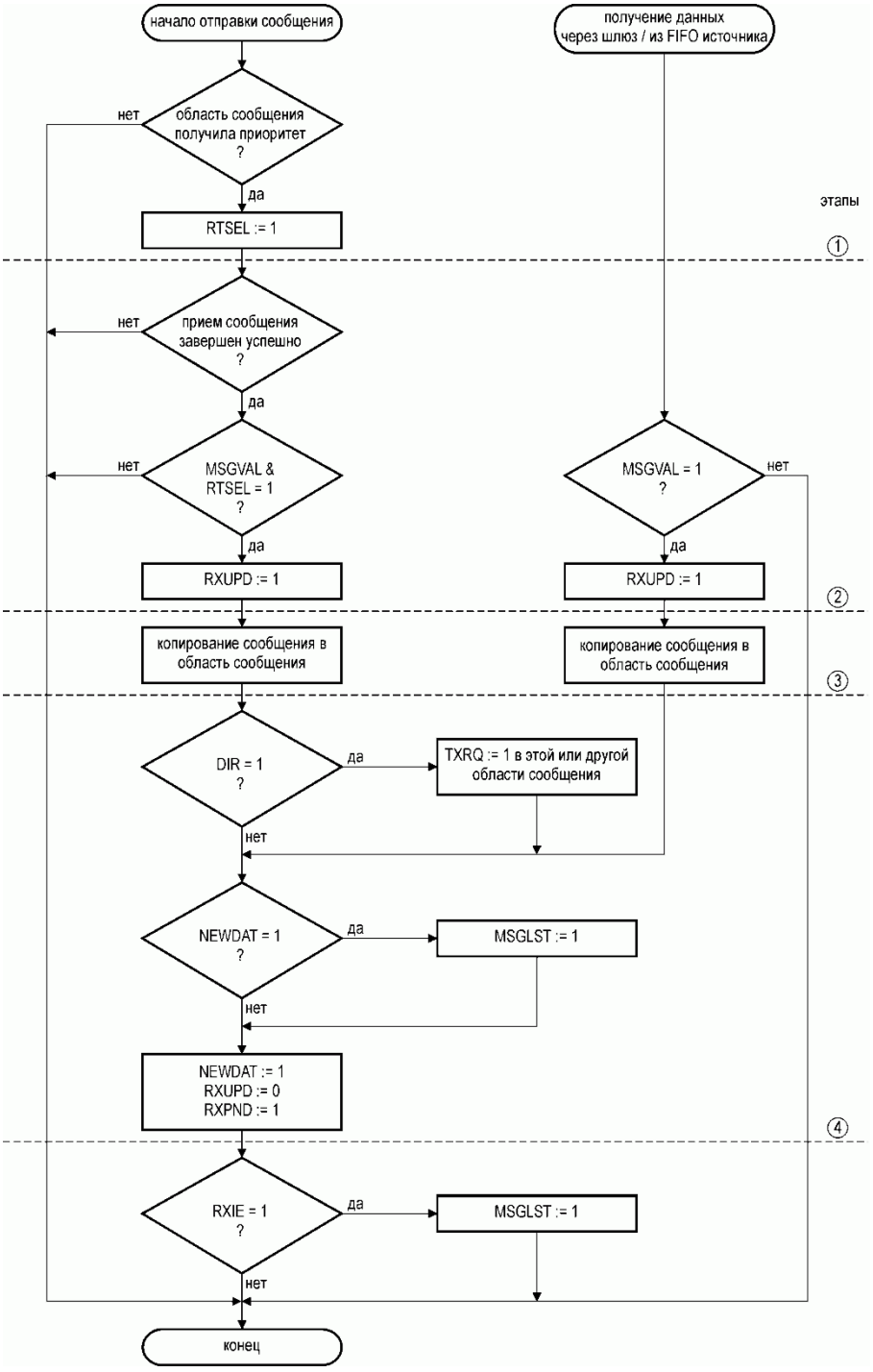


Рисунок 314 – Прием сообщения

Флаг корректности области сообщения MSGVAL

При приеме сообщения информация сохраняется в области сообщения только в том случае, если бит MSGVAL = 1_2, регистр MOSTATn. Если ЦП очищает бит MSGVAL, модуль CAN останавливает запись в область сообщения, и далее область сообщения может быть реконфигурирована центральным процессором с последующей записью в нее информации без участия модуля CAN.

Флаг распределения области сообщения RTSEL

Реконфигурация области сообщения центральным процессором во время работы модуля CAN (например, сброс MSGVAL, изменение области сообщения и повторная установка MSGVAL) происходят следующим образом:

1. Область сообщения получает приоритет.
2. ЦП очищает MSGVAL для реконфигурации области сообщения.
3. После реконфигурации ЦП снова устанавливает MSGVAL.
4. Завершение получения сообщения. Если MSGVAL установлен, полученные данные сохраняются в области сообщения, генерируется запрос на прерывание, и (если сконфигурировано) производятся шлюзовые и FIFO операции.

После реконфигурации области сообщения дальнейшее сохранение данных (см. шаг 4 выше) может быть нежелательным. Запретить запись данных в область сообщения можно посредством бита RTSEL.

После получения областью сообщения n приоритета, ее бит RTSELn устанавливается модулем CAN, открывая, таким образом, область сообщения n для записи. После приема сообщения модуль CAN дополнительно проверяет возможность записи в область сообщения n, а именно – установлен ли все еще бит RTSELn. И только в том случае, если RTSELn = 1_2, полученные данные сохраняются в области сообщения n (вместе со всеми последующими действиями, такими как прерывания области сообщения, шлюзовые и FIFO операции, установка/ сброс флагов).

Если во время операций модуля CAN область сообщения становится некорректной (сброс флага MSGVAL), флаг RTSEL должен быть сброшен до того, как флаг MSGVAL будет установлен снова, или, по крайней мере, одновременно с ним. Это необходимо для предотвращения сохранения старой информации в области сообщения. Итак, реконфигурация области сообщения должна происходить следующим образом:

1. Сброс флага MSGVAL.
2. Реконфигурация области сообщения, пока MSGVAL = 0_2.
3. Сброс флага RTSEL и далее установка флага MSGVAL.

Флаг разрешения приема RXEN

Полученное с CAN шины сообщение может быть сохранено в области сообщения только в случае RXEN = 1_2. Модуль CAN проверяет состояние флага RXEN только во время фильтрации принимаемого сообщения. После того, как сообщение принято, состояние флага не имеет значения и не оказывает влияния на дальнейшее сохранение данных в области сообщения.

Флаг RXEN позволяет управлять закрытием области сообщения: после сброса RXEN полученное сообщение сохраняется в области сообщения, которая получила приоритет, но в сохранении последующих сообщений эта область не принимает участия.

Флаги изменения RXUPD, завершенных изменений NEWDAT и потери сообщения MSGLST

Индикатором процесса сохранения (изменения) данных в области сообщения является флаг RXUPD, который выставляется с началом процесса сохранения (изменения) и сбрасывается с его окончанием.

После сохранения полученного сообщения (идентификатора, бита IDE, кода длины данных; поля данных, в случае сообщения данных) выставляется флаг NEWDAT. Если к моменту выставления (завершение сохранения/изменения данных) флаг NEWDAT был уже установлен, выставляется флаг MSGLST, который говорит о том, что произошла потеря данных.

Флаги RXUPD и NEWDAT позволяют произвести чтение корректных данных из области сообщения во время текущих операций модуля CAN. Рекомендуемая последовательность шагов следующая:

1. Сброс флага NEWDAT;

2. Чтение данных (идентификатор, данные и т. д.) из области сообщения.

3. Проверка флагов. Оба флага NEWDAT и RXUPD должны быть равны «0». В случае невыполнения этого условия – возвращение к шагу 1;

4. Если условие шага 3 выполнено, содержимое области сообщения корректно и не используется модулем CAN в течение операции чтения;

Поведение флагов RXUPD, NEWDAT и MSGLST идентично как для сообщений данных, так и для сообщений удаленных запросов.

16.9.2 Регистры управления и состояния областей сообщений

В состав каждой области сообщения входят девять 32-разрядных регистра. Расположение регистров представлено на рисунке 315, где для примера взята пятая область сообщения. Расположение регистров остальных областей сообщений идентично.

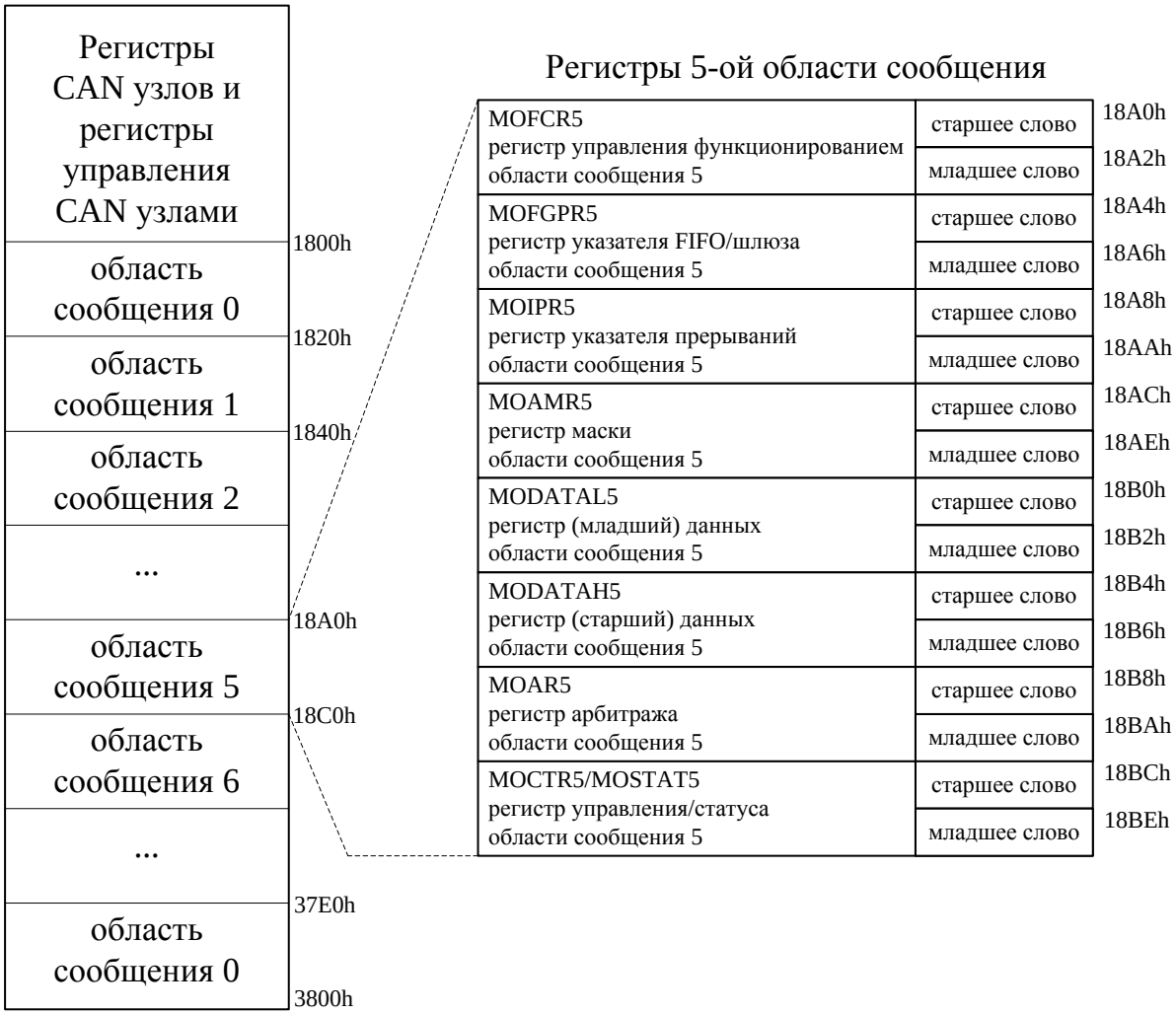


Рисунок 315 – Структура областей сообщений

Регистры состояния области сообщения MOSTATn доступны только для чтения и содержат информацию о состоянии списка, в том числе, номер списка, которому принадлежит текущая область сообщения, и номера предшествующей и следующей за ней областей.

Регистры управления областью сообщения MOCTRn и регистры состояния области сообщения MOSTATn расположены по одному адресу. Регистры MOCTRn доступны только для записи и позволяют программно устанавливать и сбрасывать соответствующие управляющие биты. Описание бит регистра состояния области сообщения MOSTATn представлено на рисунке 316. Описание бит регистра управления областью сообщения MOCTRn представлено на рисунке 317.

31	30	29	28	27	26	25	24
PNEXT7	PNEXT6	PNEXT5	PNEXT4	PNEXT3	PNEXT2	PNEXT1	PNEXT0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-1
23	22	21	20	19	18	17	16
PPREV7	PPREV6	PPREV5	PPREV4	PPREV3	PPREV2	PPREV1	PPREV0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
15	14	13	12	11	10	9	8
LIST3	LIST2	LIST1	LIST0	DIR	TXEN1	TXEN0	TXRQ
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
7	6	5	4	3	2	1	0
RXEN	RTSEL	MSGVAL	MSGLST	NEWDAT	RXUPD	TXPND	RXPND
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса
для MOSTAT0, -((n+1)*01000000h)+((n-1)+00010000h) – значение после сброса для MOSTATn (n=1÷255)

Рисунок 316 – Регистр состояния области сообщения n модуля CAN (MOSTATn) – адрес 181Ch + n*20h (n=0÷255)

Биты 31-24 PNEXT7 – PNEXT0. Указатель на следующий элемент списка. Битовое поле хранит номер области сообщения, следующей за настоящей областью сообщения n в списке.

Биты 23-16 PPREV7 – PPREV0. Указатель на предыдущий элемент списка. Битовое поле хранит номер области сообщения, предшествующей настоящей области сообщения n в списке.

Биты 15-12 LIST3 – LIST0. Номер списка. Битовое поле отражает номер списка областей сообщений, которому принадлежит область сообщения n. Битовое поле LIST изменяется аппаратно каждый раз, когда происходит перераспределение области сообщения n посредством панели команд.

Бит 11 DIR. Флаг распределения.
0 = Область для приема. При TXRQ = 1_2 для передачи формируется сообщение удаленного запроса с идентификатором области сообщения n. Полученное в ответ на запрос сообщение данных сохраняется в области сообщения n.
1 = Область для приема. Если TXRQ = 1_B, содержимое области сообщения предназначается для передачи в качестве сообщения данных. При получении сообщения удаленного запроса с корректным идентификатором устанавливается флаг TXRQ.

Бит 10 TXEN1. Флаг 1 разрешения передачи. Содержимое области сообщения n может быть передано только при условии, что оба флага TXEN0 и TXEN1 установлены. Модуль CAN использует флаг TXEN1 для выбора активной области сообщения при FIFO передачах.
0 = Область сообщения n не доступна для передачи сообщения.
1 = Область сообщения n доступна для передачи сообщения.

Бит 9 TXEN0. Флаг 0 разрешения передачи. Содержимое области сообщения n может быть передано только при условии, что оба флага TXEN0 и TXEN1 установлены. Пользователь может сбросить флаг TXEN0 для запрета передачи сообщения, которое в текущий момент корректируется, или же для запрета автоматической передачи сообщения в ответ на удаленный запрос.
0 = Область сообщения n не доступна для передачи сообщения.
1 = Область сообщения n доступна для передачи сообщения.

Бит 8 TXRQ. Запрос на передачу сообщения. Запрос на передачу содержимого области сообщения имеет силу только в случае, если установлены флаги TXRQ, TXEN0, TXEN1 и MSGVAL. TXRQ устанавливается аппаратно в случае получения корректного удаленного запроса. После успешной передачи сообщения и, если флаг NEWDAT повторно не выставлен программно, флаг TXRQ аппаратно сбрасывается.
0 = Передача содержимого области сообщения n не требуется.
1 = Требуется передача содержимого области сообщения n.

Бит 7 RXEN. Флаг разрешения приема. Состояние флага RXEN важно только при фильтрации принимаемых сообщений.
0 = Область сообщения n не доступна для приема сообщений.
1 = Область сообщения n доступна для приема сообщений.

Бит 6 RTSEL. Флаг распределения области сообщения.
Прием сообщения:
Флаг RTSEL устанавливается аппаратно, когда полученное сообщение прошло проверку идентификации для сохранения в области сообщения n. До того, как принятое сообщение будет сохранено в области сообщения n, проверяется состояние флага RTSEL (установлен ли). Это используется ЦП следующим образом – сбрасывая программно флаг RTSEL, ЦП может запрещать запись полученного сообщения в область сообщения n.
Передача сообщения:
Если область сообщения n прошла проверку идентификации и содержимое готово к отправке, устанавливается флаг RTSEL. До того, как будет начата отправка сообщения, проверяется состояние флага RTSEL (установлен ли) и состояние флага NEWDAT (сброшен ли). Также проверяется, чтобы флаг RTSEL оставался установленным до окончания передачи сообщения.
Проверка состояния флага RTSEL может потребоваться для того, чтобы избежать конфликта, возникающего при одновременной попытке передать и изменить содержимое области сообщения n. В остальных случаях состояние флага RTSEL игнорируется. Флаг RTSEL не имеет отношения к фильтрации сообщений и не сбрасывается аппаратно.
0 = Область сообщения n не выбрана для операций приема/передачи.
1 = Область сообщения n выбрана для операций приема/передачи.

Бит 5 MSGVAL. Флаг корректности области сообщения. В работе CAN узла принимают участие только корректные области сообщений.
0 = Область сообщения n некорректна.
1 = Область сообщения n корректна.

Бит 4 MSGLST. Флаг потери сообщения.
0 = Потерянных сообщений нет.
1 = Было потеряно сообщение. В область сообщения n были записаны новые данные, в то время как был выставлен флаг NEWDAT, что повлекло за собой потерю ранее хранившегося сообщения.

Бит 3 NEWDAT. Флаг завершенных изменений. Флаг NEWDAT сбрасывается аппаратно после начала передачи содержимого области сообщения n. Для предотвращения аппаратного сброса флага TXRQ в конце текущей передачи данных области сообщения n, флаг NEWDAT должен быть установлен программно после начала передачи.
0 = Содержимое области сообщения n не изменялось с момента сброса флага NEWDAT.
1 = Содержимое области сообщения n было изменено. Флаг NEWDAT выставляется аппаратно после того, как принятое сообщение было сохранено в области сообщения n.

Бит 2 RXUPD. Флаг изменения.
0 = Никаких изменений не происходит.
1 = В текущий момент происходит изменение идентификатора сообщения, DLC и данных области сообщения.

Бит 1 TXPND. Флаг ожидания передачи. Флаг TXPND должен устанавливаться программно. Сброс флага TXPND происходит аппаратно.
0 = Не было получено ни одного сообщения.
1 = Сообщение из области сообщения n было передано посредством CAN шины.

Бит 0 RXPND. Флаг ожидания приема. Флаг RXPND не сбрасывается аппаратно и должен сбрасываться программно.
0 = Не было получено ни одного сообщения.
1 = Было получено (непосредственно с CAN шины или копированием через шлюз) сообщение и записано в область сообщения n.

31 … 28	27	26	25	24
Зарезервировано	SETDIR	SETTXEN1	SETTXEN0	SETTXRQ
W-0	W-0	W-0	W-0	W-0

23	22	21	20	19	18	17	16
SETRXEN	SETRTSEL	SETMSGVAL	SETMSGLST	SETNEWDAT	SETRXUPD	SETTXPND	SETRXPND
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

15 … 12	11	10	9	8
Зарезервировано	RESDIR	RESTXEN1	RESTXEN0	RESTXRQ
W-0	W-0	W-0	W-0	W-0

7	6	5	4	3	2	1	0
RESRXEN	RESRTSEL	RESMSGVAL	RESMSGLST	RESNEWDAT	RESRXUPD	RESTXPND	RESRXPND
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W – доступ записи, -0 – значение после сброса для MOCTR0,
-((n+1)*01000000h)+((n-1)+00010000h) – значение после сброса для MOCTRn (n=1÷255)

Рисунок 317 – Регистр управления областью сообщения n модуля CAN (MOCTRn) – адрес 181Ch + n*20h (n=0÷255)

Биты 31-28 Зарезервировано. При записи следует писать «0».

Бит 27 SETDIR. Установка бита DIR регистра MOSTATn

Бит 26 SETTXEN1. Установка бита TXEN1 регистра MOSTATn
Бит 25 SETTXEN0. Установка бита TXEN0 регистра MOSTATn

Бит 24 SETTXRQ. Установка бита TXRQ регистра MOSTATn

Бит 23 SETRXEN. Установка бита RXEN регистра MOSTATn

Бит 22 SETRTSEL. Установка бита RTSEL регистра MOSTATn

Бит 21 SETMSGVAL. Установка бита MSGVAL регистра MOSTATn

Бит 20 SETMSGLST. Установка бита MSGLST регистра MOSTATn

Бит 19 SETNEWDAT. Установка бита NEWDAT регистра MOSTATn

Бит 18 SETRXUPD. Установка бита RXUPD регистра MOSTATn

Бит 17 SETTXPND. Установка бита TXPND регистра MOSTATn

Бит 16 SETRXPND. Установка бита RXPND регистра MOSTATn

Биты 15-12 Зарезервировано. При записи следует писать «0»

Бит 11 RESDIR. Сброс бита DIR регистра MOSTATn

Бит 10 RESTXEN1. Сброс TXEN1 регистра MOSTATn

Бит 9 RESTXEN0. Сброс TXEN0 регистра MOSTATn

Бит 8 RESTXRQ. Сброс бита TXRQ регистра MOSTATn

Бит 7 RESRXEN. Сброс бита RXEN регистра MOSTATn

Бит 6 RESRTSEL. Сброс бита RTSEL регистра MOSTATn

Бит 5 RESMSGVAL. Сброс бита MSGVAL регистра MOSTATn

Бит 4 RESMSGLST. Сброс бита MSGLST регистра MOSTATn

Бит 3 RESNEWDAT. Сброс бита NEWDAT регистра MOSTATn

Бит 2 RESRXUPD. Сброс бита RXUPD регистра MOSTATn

Бит 1 RESTXPND. Сброс бита TXPND регистра MOSTATn

Бит 0 RESRXPND. Сброс бита RXPND регистра MOSTATn

Примечание – Правила сброса/установки для битов регистров MOCTRn, где n = 0 – 255, сведены в таблицу 150.

Таблица 150 – Состояния битов сброса/установки регистров MOCTRn и их влияние на соответствующие биты регистров MOSTATn, где n = 0 – 31

Бит RESy[1)]	Бит SETy[1)]	Действие, оказываемое на бит «y», регистра MOSTATn
0	0 Нет записи	Без изменений
Нет записи 1	0 1	
1	0 Нет записи	Сброс бита
0 Нет записи	1	Установка бита

_____

[1)] Параметр «y» подразумевает вторую часть названия бита RXPND, TXPND и т. д. до DIR.

16.9.3 Передача данных

Алгоритм передачи сообщений показан на рисунке 318. Одновременно с копированием данных: идентификатора, бита IDE, бита RTR, равного биту DIR, кода длины данных и собственно данных, из области сообщения, содержимое которой должно быть передано во внутренний передающий буфер соответствующего CAN узла, для контроля соблюдения четкой последовательности выполнения всех операций выставляются флаги состояния.

Процессы передачи сообщений данных и сообщений удаленных запросов идентичны, показанным на рисунке 318.

Флаги потери сообщения MSGVAL, разрешения передачи TXEN0, TXEN1 и запрос на передачу сообщения TXRQ

Сообщение может быть передано только в случае, когда все четыре бита MSGVAL, TXEN0, TXEN1, TXRQ равны 1_2, что показано в таблице 151.

Таблица 151 – Описание битов, управляющих передачей сообщений

Бит (флаг)	Описание
1	2
MSGVAL	Флаг корректности области сообщения. Основной бит доступа к области сообщения
TXRQ	Запрос на передачу сообщения. Бит устанавливается, когда требуется передача содержимого области сообщения. По окончании успешной передачи сообщения, этот бит сбрасывается аппаратно, за исключением случая, когда возникает необходимость передачи новых данных (бит NEWDAT = 1_2). Если установлен бит STT регистра MOFCRn, бит TXRQ очищается, когда данные из области сообщения копируются в передающий буфер CAN узла. Полученный удаленный запрос (по окончании получения сообщения удаленного запроса) устанавливает бит TXRQ для формирования запроса на передачу запрашиваемых данных
TXEN0	Флаг 0 разрешения передачи. Этот флаг может быть временно сброшен программно для запрета передачи содержимого области сообщения, в которую записываются новые данные. Это позволяет избежать передачи некорректных смешанных данных. Если TXEN0 = 0_2, допускается передача сообщения удаленного запроса, но передача сообщения данных станет возможной только после программной установки флага TXEN0
TXEN1	Флаг 1 разрешения передачи. Этот флаг используется при FIFO передачах для выбора активной области в структуре FIFO. Для областей сообщений, не являющихся элементами FIFO структуры, флаг TXEN1 может быть или постоянно установленным (= 1_2), или использоваться как второй независимый бит разрешения передачи

Флаг распределения области сообщения RTSEL

Флаг RTSEL выставляется после того, как область сообщения получает приоритет для передачи своего содержимого.

Когда данные области сообщения копируются в передающий буфер, флаг RTSEL проверяется, и сообщение передается только в случае, если RTSEL = 1_2. После успешной передачи сообщения, флаг RTSEL проверяется снова, и если RTSEL = 1_2, осуществляются дальнейшие операции.

Для полной и завершенной реконфигурации корректной области сообщения должны быть выполнены следующие шаги:

1 Очистка бита MSGVAL.

2 Реконфигурация области сообщения пока MSGVAL = 0_2.

3 Сброс флага RTSEL и установка флага MSGVAL.

Сброс флага RTSEL гарантирует как полное отключение области сообщения от текущей передачи, так и то, что никакие операции (копирование данных в передающий буфер, включая сброс флага NEWDAT, очистка бита TXRQ, прерывание сообщения и т. д.), относящиеся к старой конфигурации этой области сообщения,

не повлияют на новую конфигурацию после установки корректности области (флаг MSGVAL).

Флаг завершенных изменений NEWDAT

После завершения передачи содержимого области сообщения в передающий буфер CAN узла, флаг NEWDAT аппаратно сбрасывается, тем самым обозначая, что область сообщения открыта для записи новых данных.

Если после успешной передачи сообщения (на CAN шину) бит NEWDAT все еще остается равным «0» (в область сообщения не были записаны новые данные), флаг TXRQ автоматически аппаратно сбрасывается. Если же бит NEWDAT был снова установлен программно (потому что требуется передача новых данных), флаг TXRQ не сбрасывается, тем самым разрешая передачу новых данных.

16.10 Функции области сообщения

Регистр управления функционированием области сообщения n MOFCRn включает в себя биты, которые конфигурируют действия области сообщения, и хранит код длины данных сообщения. Описание бит регистра управления функционированием области сообщения MOFCRn представлено на рисунке 319.

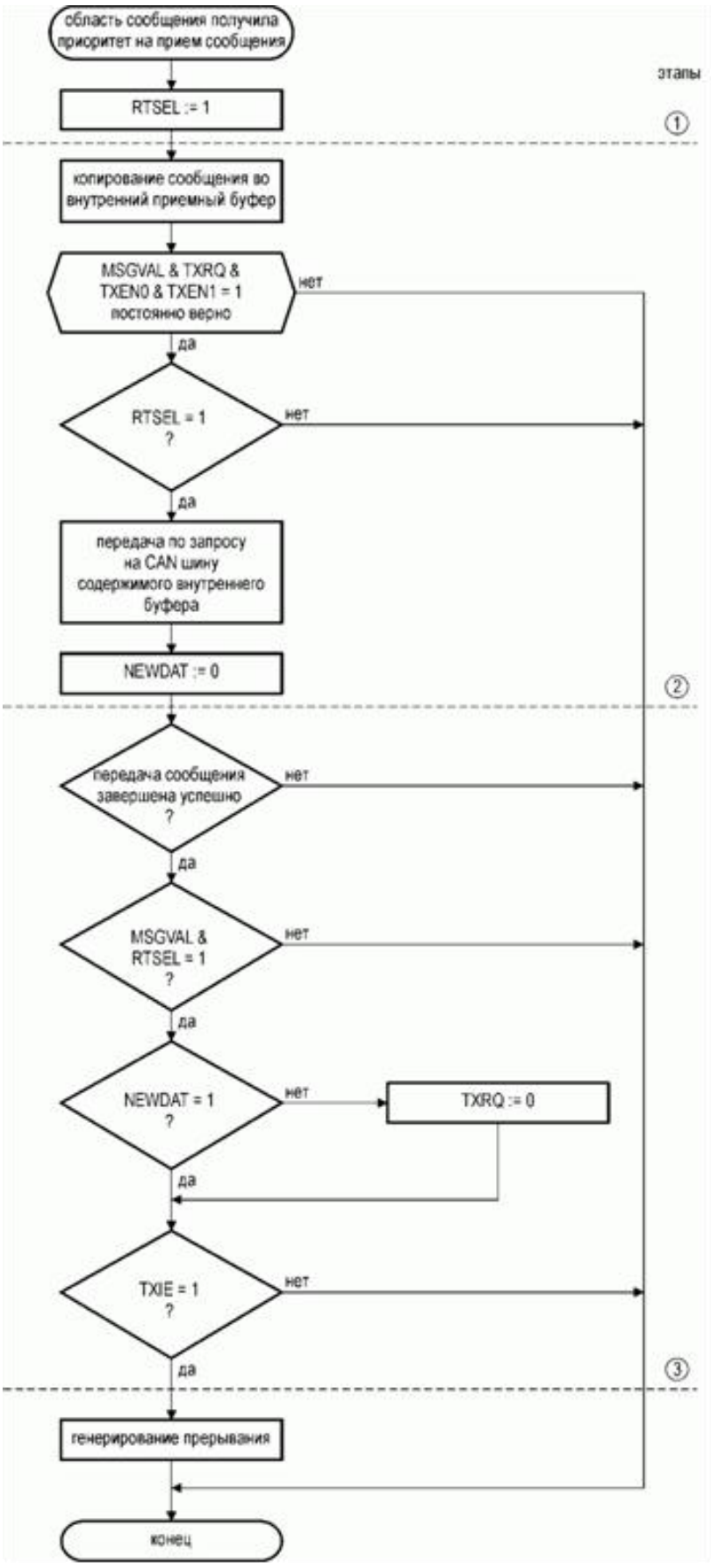
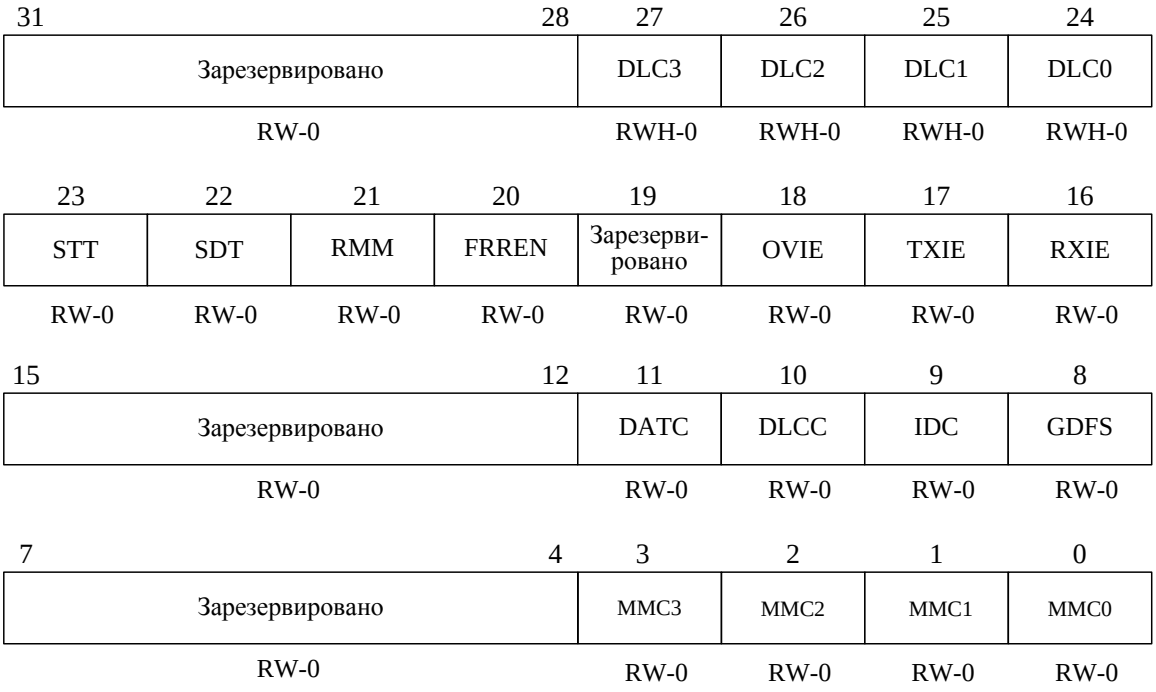


Рисунок 318 – Передача сообщения



R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 319 – Регистр управления функционированием области сообщения n модуля CAN (MOFCRn) – адрес 1800h + n*20h (n=0÷255)

Биты 31-28 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Биты 27-24 DLC3 – DLC0. Код длины данных. Это битовое поле определяет количество байт данных, хранящихся в области сообщения n. Допустимые значения – от 0 до 8. Запись любого числа больше восьми в битовое поле DLC будет означать, что длина данных – 8 байт. Если получено сообщение с DLC > 8, полученное значение сохраняется в области сообщения.

Бит 23 STT. Бит однократной пересылки данных. Если этот бит установлен, то TXRQ сбрасывается с началом передачи содержимого области сообщения n. Таким образом, при неудачной пересылке не будет осуществлена повторная передача сообщения.

Бит 22 SDT. Бит блокирования повторной пересылки данных.
- если SDT = 1_2 и область сообщения n не является базовой областью FIFO, тогда флаг MSGVAL сбрасывается, когда область сообщения n принимает участие в пересылке данных (прием или передача);
- если SDT = 1_2, а область сообщения n является базовой областью FIFO, тогда флаг MSGVAL сбрасывается, когда указатель CUR на текущую область сообщения n достигает значения битового поля SEL регистра MOFGPRn;
- если SDT = 0_2, на флаг MSGVAL не оказывается влияния.

Бит 21 RMM. Бит удаленного мониторинга. Бит RMM оказывает влияние только в случае, если область сообщения n является областью для передачи.
0 = Удаленный мониторинг выключен. Идентификатор, бит IDE и DLC области сообщения n остаются без изменений после приема корректного удаленного запроса.
1 = Удаленный мониторинг включен. Идентификатор, бит IDE и DLC корректного удаленного запроса копируются в область для передачи n.

Бит 20 FRREN. Бит разрешения удаленного запроса. Этот бит определяет, устанавливается ли флаг TXRQ в регистре области сообщения n или в регистре другой области сообщения, на которую указывает указатель CUR.
0 = При приеме корректного сообщения удаленного запроса устанавливается флаг TXRQ области сообщения n.
1 = При приеме корректного сообщения удаленного запроса устанавливается флаг TXRQ области сообщения, указанной битовым полем CUR.

Бит 19 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 18 OVIE. Флаг разрешения прерывания переполнения. Этот бит разрешает формирование FIFO прерывания области сообщения n. Это прерывание генерируется, когда указатель на текущую область сообщения CUR достигает значения битового поля SEL регистра указателя FIFO/шлюза области сообщения n (MOFGPRn). Если область сообщения n является базовой областью FIFO приема, выходную линию прерываний (одну из 16) для данного типа прерывания выбирает битовое поле TXINP регистра MOIPRn. Если область сообщения n является базовой областью FIFO передачи, выходную линию прерываний (одну из 16) для данного типа прерывания выбирает битовое поле RXINP регистра MOIPRn. На все остальные типы областей сообщений флаг OVIE не оказывает влияния.
0 = FIFO прерывание запрещено.
1 = FIFO прерывание разрешено.

Бит 17 TXIE. Бит разрешения прерывания передачи. Бит разрешает формирование прерывания областью сообщения n после передачи сообщения. Прерывание генерируется после успешной передачи сообщения.
Битовое поле TXINP регистра MOIPRn выбирает выходную линию прерывания (одну из 16) для данного типа прерывания.
0 = Формирование прерывания запрещено.
1 = Формирование прерывания разрешено.

Бит 16 RXIE. Бит разрешения прерывания приема. Бит разрешает формирование прерывания областью сообщения n после приема сообщения. Прерывание генерируется после получения сообщения (независимо от того, было ли сообщение получено посредством CAN шины или через шлюз). Битовое поле RXINP регистра MOIPRn выбирает выходную линию прерывания (одну из 16) для данного типа прерывания.
0 = Формирование прерывания запрещено.
1 = Формирование прерывания разрешено.

Биты 15-12 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 11 DATC. Флаг копирования данных. Флаг DATC применим только к шлюзовой области-источник и игнорируется остальными.
0 = Поле данных не копируется.
1 = Поля данных регистров MODATALn и MODATAHn шлюзовой области-источника (после сохранения полученного сообщения в области-источнике) копируется в шлюзовую область-приемник.

Бит 10 DLCC. Флаг копирования кода длины данных. Флаг DLCC применим только к шлюзовой области-источник и игнорируется остальными.
0 = Код длины данных не копируется.
1 = Код длины данных шлюзовой области-источник (после сохранения полученного сообщения в области-источник) копируется в шлюзовую область-приемник.

Бит 9 IDC. Флаг копирования идентификатора. Флаг IDC применим только к шлюзовой области-источник и игнорируется остальными.
0 = Идентификатор шлюзовой области-источник не копируется.
1 = Идентификатор шлюзовой области-источник
(после сохранения полученного сообщения в области-источник) копируется в шлюзовую область-приемник.

Бит 8 GDFS. Флаг пересылки сообщения через шлюз. Флаг GDFS применим только к шлюзовой области-источник и игнорируется остальными.
0 = Флаг TXRQ области-приемник не изменяется.
1 = Флаг TXRQ шлюзовой области-приемник[2)] устанавливается после успешной пересылки сообщения из шлюзовой области-источник в шлюзовую область-приемник.

Биты 7-4 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Биты 3-0 MMC3 – MCC0. Выбор типа области сообщения.

0000_2 = Стандартная область сообщения.
0001_2 = Базовая область сообщения FIFO приема.
0010_2 = Базовая область сообщения FIFO передачи.
0011_2 = Вспомогательная область сообщения для FIFO передачи.
0100_2 = Шлюзовая область-источник[1)].
0101_2– 1111_2 = Зарезервированы.

_____

[1)] Область сообщения, используемая в качестве источника сообщения для пересылки через шлюз.

[2)] Область сообщения, используемая в качестве приемника сообщения при пересылке через шлюз.

16.10.1 Стандартная область сообщения

Область сообщения определяется как стандартная область сообщения, если битовое поле MMC = 0000_B (регистр MOFCRn). Стандартная область сообщения может принимать и передавать сообщения, согласно правилам, описанным выше. Дополнительно имеются два режима, каждый из которых может быть выбран индивидуально:

- режим передачи данных с защитой от повторений;
- режим однократной пересылки данных.

Режим передачи данных с защитой от повторений

Выбирается установкой бита SDT регистра MOFCRn.

Режим передачи данных, в котором имеется механизм защиты от дублирования передачи информации.

Прием сообщений в режиме передачи данных с защитой от повторений.

После приема сообщения данных, принятые данные сохраняются в выбранной области сообщения, в связи с чем предыдущее содержимое этой области теряется. При дальнейшем приеме, очередные новые данные могут быть записаны в ту же область сообщения, что в свою очередь опять вызовет потерю хранившихся данных.

Если SDT = 1_2 (режим включен), то после приема сообщения и сохранения его в выбранной области сообщения, флаг MSGVAL этой области автоматически аппаратно сбрасывается, что делает область некорректной, а, следовательно, недоступной для последующей записи.

После приема сообщения удаленного запроса, автоматического сброса флага MSGVAL не происходит.

Передача сообщений в режиме передачи данных с защитой от повторений

Если область сообщения получает серию удаленных запросов, она, в ответ на это, передает несколько сообщений данных. Если в промежутки времени между передачами содержимое области сообщения не изменялось, то на CAN шину будут несколько раз переданы одни и те же данные.

Если SDT = 1_2, то сразу после однократной успешной передачи данных флаг MSGVAL передающей области сообщения будет сброшен и область станет недоступной для передачи.

После передачи сообщения удаленного запроса, автоматического сброса флага MSGVAL не происходит.

16.10.2 Режим однократной пересылки данных

Выбирается установкой бита STT регистра MOFCRn.

Если бит STT = 1_2, то бит TXRQ сбрасывается, когда содержимое области сообщения копируется в передающий буфер CAN узла. Таким образом, при дальнейшей неудачной (вследствие ошибок) пересылке сообщения по CAN шине повторной передачи не будет.

16.10.3 FIFO структура области сообщения

Регистр указателя FIFO/шлюза области сообщения MOFGPRn содержит установки указателей на области сообщений, которые используются при операциях FIFO и шлюзовых операциях. Описание бит регистра указателя FIFO/шлюза области сообщения MOFGPRn представлено на рисунке 320.

31	30	29	28	27	26	25	24
SEL7	SEL6	SEL5	SEL4	SEL3	SEL2	SEL1	SEL0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
CUR7	CUR6	CUR5	CUR4	CUR3	CUR2	CUR1	CUR0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
15	14	13	12	11	10	9	8
TOP7	TOP6	TOP5	TOP4	TOP3	TOP2	TOP1	TOP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
BOT7	BOT6	BOT5	BOT4	BOT3	BOT2	BOT1	BOT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 320 – Регистр указателя FIFO/шлюза области сообщения n модуля CAN (MOFCRn) – адрес 1804h + n*20h (n=0÷255)

Биты 31-24 SEL7 – SEL0. Указатель выбора области. Это битовое поле является вторым (программным) указателем для дополнения аппаратного указателя CUR в структуре FIFO. Битовое поле SEL используется для мониторинга возникающих задач (генерирование прерываний FIFO).

Биты 23-16 CUR7 – CUR0. Указатель текущей области. Это битовое поле хранит значение, которое указывает на выбранный в настоящий момент элемент в пределах списка структуры FIFO/шлюза. После каждой операции FIFO (также шлюзовой операции), битовое поле CUR изменяет свое значение – в него записывается номер следующей по списку области сообщения (указанной в битовом поле PNEXT регистра MOSTATn). Так происходит до тех пор, пока не будет достигнут последний элемент списка (битовое поле TOP), после чего в CUR заноситься номер первого элемента (битовое поле BOT).

Биты 15-8 TOP7 – TOP0. Указатель конца. Это битовое поле хранит значение, которое указывает на последний элемент в структуре FIFO.

Биты 7-0 BOT7 – BOT0. Указатель начала. Это битовое поле хранит значение, которое указывает на первый элемент в структуре FIFO.

В случае сильной загрузки ЦП может быть затруднительным обработать серию сообщений модуля CAN в рамках необходимого времени. Такие ситуации могут возникать вследствие получения и/или передачи большого числа сообщений за малые промежутки времени.

Для таких случаев предусмотрена система буферов быстрого ввода-вывода, так называемая FIFO структура, которая позволяет избежать потери принимаемых сообщений и минимизировать время подготовки сообщений к отправке. Помимо этого, FIFO структура используется для автоматического приема/передачи серий сообщений и генерирования однократных прерываний по окончании операций с ними.

Допускается организация нескольких параллельных FIFO структур. Число структур и их составляющих зависит только от количества доступных областей сообщений. FIFO структура может быть создана, изменена и удалена в любой момент времени, даже во время операций модуля CAN.

На рисунке 321 представлена основная FIFO структура. Она состоит из одной базовой области и n-ого числа вспомогательных областей. Вспомогательные области объединяются последовательно в списки (подобно спискам областей сообщений). Базовая область может быть занесена в любой список. Хотя на рисунке базовая область не относится ни к одному из списков, она может быть вставлена в любую последовательность вспомогательных областей. Это означает, что базовая область одновременно является и вспомогательной областью (шлюзовые операции не возможны). Порядковые номера областей сообщений (0, 1, 2 и т. д.) не имеют никакого значения при FIFO операциях с областями.

Базовая область не нуждается в обязательном занесении ее в какой-либо список, в отличие от вспомогательных областей, которые должны быть определены в общий список (так как они последовательно связаны). С помощью указателей (битовые поля BOT, CUR и TOP регистра MOFGPRn) можно присоединять базовую область к вспомогательным областям, независимо от того принадлежат базовая и вспомогательные области одному списку или разным спискам.

Минимальная FIFO структура может состоять из одной области сообщения, которая будет одновременно являться и базовой и вспомогательной (фактически не используется). Максимальная FIFO структура может включать в себя все 255 области сообщений модуля CAN. В остальном размеры FIFO структуры варьируются между этими границами.

В базовой области FIFO границы установлены. Битовое поле BOT регистра MOFGPRn базовой области указывает на самый младший элемент FIFO структуры. Битовое поле TOP регистра MOFGPRn базовой области указывает на самый старший элемент FIFO структуры. Битовое поле CUR регистра MOFGPRn базовой области указывает на вспомогательную область, которая в настоящий момент выбрана модулем CAN для передачи сообщения. Как только начинается передача, в CUR записывается номер следующей по списку вспомогательной области сообщения (CUR = PNEXT используемой области). Если значение битового поля CUR достигло номера старшего элемента списка (CUR = TOP), то следующее значение, которое будет записано в поле CUR, – это BOT (реализация автоматического перехода в начало списка). Таким образом, реализуется замкнутая FIFO структура, в которой битовые поля TOP и BOT устанавливают связь между началом и концом списка.

Битовое поле SEL регистра MOFGPRn позволяет определить вспомогательную область в пределах списка, для которой генерируется прерывание всякий раз, когда указатель CUR достигает значения указателя SEL. Так же, битовое поле SEL позволяет отследить окончание запланированной передачи серии сообщений или выдать прерывание, предупреждающее о том, что FIFO структура становится заполненной.

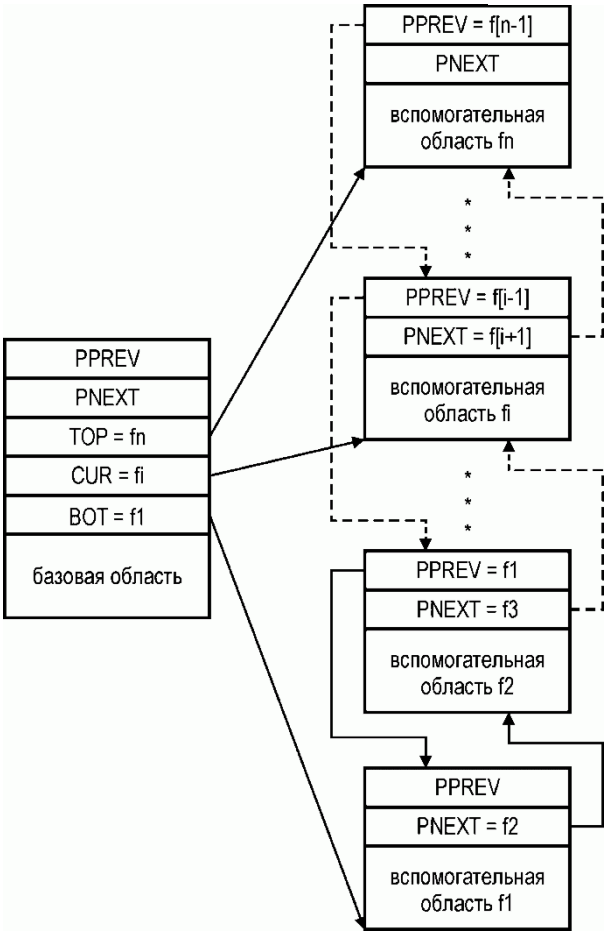


Рисунок 321 – FIFO структура с базовой областью и n вспомогательными областями

16.10.4 FIFO структура приема

FIFO структура приема используется для буферизации входящих (полученных) сообщений данных и удаленных запросов.

FIFO структура приема активируется записью значения 0001_2 в битовое поле MMC регистра MOFCRn базовой области. Установка MMC = 0001_2 автоматически определяет область сообщения как базовую область FIFO приема. Типы вспомогательных областей FIFO не имеют значения при операциях FIFO структуры приема.

Когда базовая область FIFO получает сообщение от CAN узла, которому она принадлежит, сообщение сохраняется не в этой базовой области, а в вспомогательной области сообщения, на которую указывает битовое поле CUR (регистра MOGPRn базовой области). При этом по умолчанию предполагается, что для вспомогательной области битовое поле MMC = 0000_2 (действительное значение MMC игнорируется) и никаких операций фильтрации принимаемого сообщения не производится.

Одновременно с приемом сообщения, текущее значение указателя CUR базовой области меняется на номер следующей по списку вспомогательной области FIFO структуры. Эта вспомогательная область будет использована для приема следующего сообщения.

Если установлен флаг OVIE регистра MOFCRn базовой области, и значение указателя CUR становится равным значению указателя SEL, генерируется прерывание переполнения. Это прерывание генерируется на узле прерываний с указателем TXINP базовой области сразу после сохранения полученного сообщения в вспомогательной области.

Прерывания передач генерируются, если это разрешено битом TXIE регистра MOFCRn.

Следует помнить, что сообщение сохраняется в базовой и вспомогательной области FIFO, только если MSGVAL = 1_2.

Во избежание непосредственного приема сообщения вспомогательной областью, как если бы она была независимой областью сообщения и не принадлежала FIFO структуре, флаги RXEN всех вспомогательных областей должны быть сброшены. Состояние флага RXEN неважно в случае, когда вспомогательная область занесена в список, не связанный с CAN узлом.

16.10.5 FIFO структура передачи

FIFO структура передачи используется для буферизации серий сообщений данных или удаленных запросов, которые должны быть отправлены. FIFO структура передачи состоит из базовой области и одной или более вспомогательных областей.

FIFO структура передачи активируется записью значения 0010_2 в битовое поле MMC регистра MOFCRn базовой области. В отличие от FIFO структуры приема, в битовые поля MMC вспомогательных областей (FIFO структуры передачи) должно быть записано значение 0011_2. Указатели CUR всех вспомогательных областей должны указывать на базовую область FIFO передачи (чтобы инициализироваться программно).

Флаги TXEN1 всех вспомогательных областей сообщений, за исключением одной, на которую указывает CUR базовой области, должны быть программно сброшены. Флаг TXEN1 указанной области должен быть установлен. Указатель CUR базовой области может быть инициализирован для любой вспомогательной области.

При определении корректности областей сообщений FIFO структуры для начала FIFO операции базовая область должна быть первой определена, как корректная (MSGVAL = 1_2).

В случае удаления FIFO структуры, прежде чем начнется операция удаления, все вспомогательные области, принадлежащие этой FIFO структуре, должны быть определены как некорректные (MSGVAL = 0_2).

FIFO структура передачи использует флаги TXEN1 всех своих областей для выбора сообщения для передачи. В результате фильтрации право передавать сообщение получает та область, у которой выставлен флаг TXEN1. После передачи сообщения флаг TXEN1 аппаратно сбрасывается, а в указатель CUR записывается номер следующей области, требующей отправки сообщения, для которой уже выставлен (аппаратно) свой флаг TXEN1, и так далее для всей FIFO структуры.

Если установлен флаг OVIE регистра MOFCRn базовой области, и значение указателя CUR становится равным значению указателя SEL, генерируется преры-

вание переполнения. Это прерывание генерируется на узле прерываний с указателем RXINP базовой области после завершения операций получения сообщения.

Прерывания приема базовой области FIFO передачи генерируются, если это разрешено битом RXIE регистра MOFCRn.

16.10.6 Режим шлюза

Режим позволяет реализовывать автоматическую передачу информации (через шлюз) между двумя независимыми CAN шинами без участия ЦП.

Шлюз можно сформировать на уровне областей сообщений и осуществлять передачу информации между CAN узлами. Шлюз может быть сформирован между двумя любыми областями сообщений, принадлежащими разным CAN узлам. Количество шлюзов зависит только от количества областей сообщений, допускающих формирование структуры шлюзов.

Режим шлюза активируется записью значения 0100_2 в битовое поле MMC регистра MOFCRn области сообщения и инициализирует ее как шлюзовую область-источник. Область сообщения, которая будет являться шлюзовой областью-приемником, выбирается указателем CUR области-источника. Для формирования шлюза достаточно, чтобы область-приемник была корректной (ее флаг MSGVAL = 1_2). Остальные параметры не влияют на возможность осуществления передачи между областями от источника к приемнику.

Шлюзовая область-источник, показанная на рисунке 322, функционирует как обычная область сообщения с тем отличием, что возможны дополнительные действия модуля CAN при приеме и сохранении сообщения в области-приемнике:

1 Если установлен флаг DLCC регистра MOFCRn области-источника, код длины данных MOFCRn.DLC копируется из шлюзовой области-источника в шлюзовую область-приемник.

2 Если установлен флаг IDC регистра MOFCRn области-источника, идентификатор MOARn.ID и расширение идентификатора MOARn.IDE копируются из шлюзовой области-источника в шлюзовую область-приемник.

3 Если установлен флаг DATC регистра MOFCRn области-источника, байты данных, хранящиеся в двух регистрах MODATALn и MODATAHn области-источника, копируются из шлюзовой области-источника в шлюзовую область-приемник. Копируются все 8 байт данных, вне зависимости от значения поля DLC.

4 Если флаг GDFS регистра MOFCRn области-источника установлен, то устанавливается бит запроса передачи TXRQ области-приемника.

5 Устанавливаются флаги RXPND и NEWDAT регистр MOSTATn области-приемника.

6 Если установлен флаг RXIE регистра MOSTATn области-приемника, то генерируется запрос на прерывание.

7 Указатель области CUR регистра MOFGPRn области-источника переводится на следующую область-приемник по правилам FIFO структуры. Сформировать шлюз между областью-источником и одной областью-приемником (значение указателя CUR будет оставаться неизменным) возможно установкой:

MOFGPRn.TOP = MOFGPRn.BOT = MOFGPRn.CUR = область-приемник.

Организация шлюзового соединения «область-источник – область-приемник» аналогична организации FIFO структуры «базовая область – вспомога-

тельная область». Это означает, что имеется возможность формирования шлюза с интегрированным FIFO-приемником. На рисунке 322 область сообщения слева – шлюзовая область-источник, а область сообщения справа – шлюзовая область-приемник.

При получении сообщения данных (область-источник является областью для приема, DIR = 0_2) и при получении удаленного запроса (область-источник является областью для передачи) через шлюз используется один и тот же механизм.

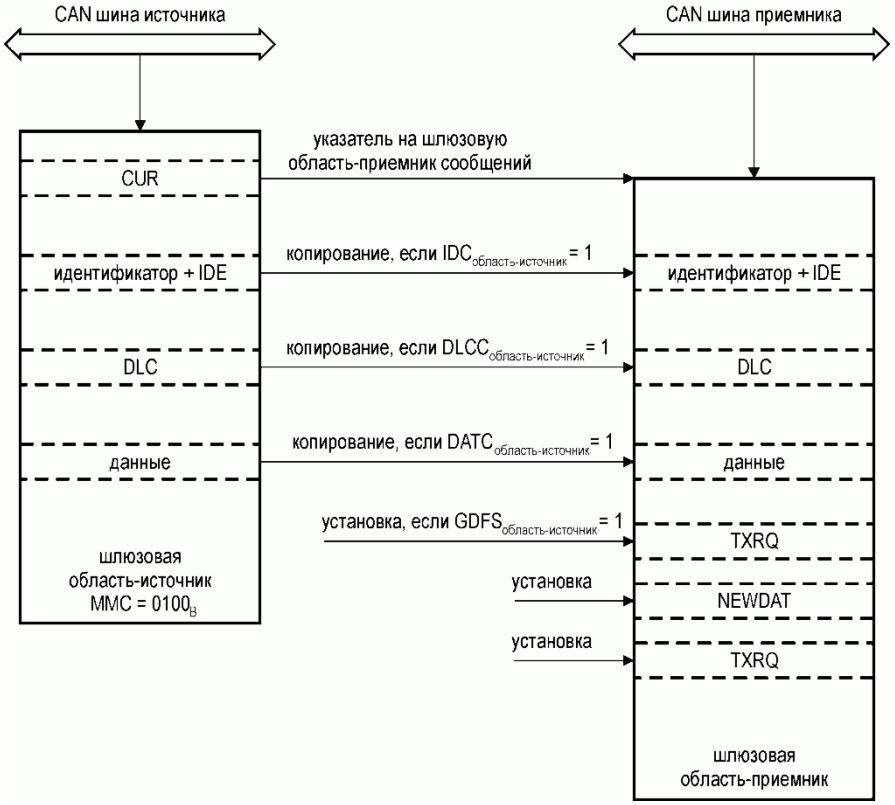


Рисунок 322 – Передача через шлюз от источника к приемнику

16.10.7 Удаленные запросы

После получения CAN узлом сообщения удаленного запроса и сохранения его в области сообщения, выставляется бит запроса передачи для ответа на удаленный запрос (отправка сообщения данных) или для автоматического повторения запроса. Если бит FRREN = 0_2 регистра MOFCRn области сообщения, в которую было сохранено сообщение удаленного запроса, то выставляется флаг TXRQ регистра MOSTATn этой области.

Если бит FRREN = 1_2 области сообщения, в которую было сохранено сообщение удаленного запроса, то выставляется флаг TXRQ той области сообщения, на которую указывает CUR. При этом указатель CUR (принадлежащий регистру MOFGPRn области сообщения, в которую было сохранено сообщение удаленного запроса) не меняет своего значения.

Несмотря на то, что механизм удаленных запросов работает независимо от типа области сообщения, он наиболее полезен при использовании шлюзов, для формирования удаленных запросов на шлюзовой шине источника после получения

удаленного запроса на шлюзовой шине приемника. В зависимости от значения бита FRREN шлюзовой области-приемника, есть два варианта обработки удаленного запроса, возникшего с той стороны шлюза, где расположена область-приемник (при условии, что происходит передача из области-источника в область-приемник, т. е. $DIR_{область\text{-}источник} = 0_2$ и $DIR_{область\text{-}приемник} = 1_2$).

1 Обработка запроса, если шлюзовая область-приемник с FRREN = 0_2:

- сообщение удаленного запроса принимается шлюзовой областью-приемником;
- бит TXRQ шлюзовой области-приемника устанавливается автоматически;
- сообщение данных с текущей информацией, хранящейся в области-приемнике, передается на шину приемника.

2 Обработка запроса, если шлюзовая область-приемник с FRREN = 1_2:

- сообщение удаленного запроса принимается шлюзовой областью-приемником;
- бит TXRQ шлюзовой области-источника (должна быть указана в поле CUR регистра области-приемника) устанавливается автоматически;
- сообщение данных передается областью-источником на CAN шину источника;
- получатель удаленного запроса в ответ выдает сообщение данных на шину источника;
- сообщение данных сохраняется в области-источнике;
- сообщение данных копируется в область-приемник (через шлюз);
- выставляется бит TXRQ области-приемника (при условии, что $GDFS_{область\text{-}источник} = 1_2$);
- новые данные, сохраненные в области-приемнике, передаются на шину приемника, в ответ удаленный запрос на шине приемника.

16.10.8 Дополнительная информация по регистрам модуля CAN

Модуль CAN включает три группы регистров, см. рисунки 323 – 324.

- регистры модуля;
- регистры узлов (для каждого узла своя группа регистров);
- регистры областей сообщений (для каждой области сообщения своя группа регистров).

Каждый регистр состоит из двух слов (старшего и младшего).

Допускается запись в регистры пословно и побайтно.

В зарезервированные биты (обозначены «0») всегда следует записывать «0».

16.11 Регистры модуля CAN

Регистр управления портом CAN узла NPCRx конфигурирует порты прима/передачи. Регистр NPCRx может быть доступен для записи, только если установлен бит CCE регистра NCRx. Описание бит регистра идентификации модуля ID представлено на рисунке 323. Описание бит регистра управления портом CAN узла NPCRx представлено на рисунке 324.

31	30	29	28	27	26	25	24
MOD_ NUMBER15	MOD_ NUMBER14	MOD_ NUMBER13	MOD_ NUMBER12	MOD_ NUMBER11	MOD_ NUMBER10	MOD_ NUMBER9	MOD_ NUMBER8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
MOD_ NUMBER7	MOD_ NUMBER6	MOD_ NUMBER5	MOD_ NUMBER4	MOD_ NUMBER3	MOD_ NUMBER2	MOD_ NUMBER1	MOD_ NUMBER0
R-0	R-0	R-1	R-0	R-1	R-0	R-1	R-1
15	14	13	12	11	10	9	8
MOD_ TYPE7	MOD_ TYPE6	MOD_ TYPE5	MOD_ TYPE4	MOD_ TYPE3	MOD_ TYPE2	MOD_ TYPE1	MOD_ TYPE0
R-1	R-1	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
MOD_ REV7	MOD_ REV6	MOD_ REV5	MOD_ REV4	MOD_ REV3	MOD_ REV2	MOD_ REV1	MOD_ REV0
RWH-0	RWH-1	RWH-0	RWH-1	RWH-0	RWH-0	RWH-0	RWH-1

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 323 – Регистр идентификации модуля CAN (ID) – адрес 0808h

Биты 31-16 MOD_NUMBER15 - MOD_NUMBER0. Номер модуля CAN. Идентификационный номер модуля CAN. По умолчанию – $002B_{16}$.

Биты 15-8 MOD_TYPE7 - MOD_TYPE0. Разрядность модуля CAN. По умолчанию, значение равно $C0_{16}$. Модуль 32-разрядный.

Биты 7-0 MOD_REV7 - MOD_REV0. Число модификаций модуля CAN. Число модификаций модуля, начиная со значения 01_{16} (первая модификация).

Биты 7-0 MOD_REV7 - MOD_REV0. Число модификаций модуля CAN. Число модификаций модуля, начиная со значения 01_{16} (первая модификация).

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 324 – Регистр управления портом узла x модуля CAN (NPCRx) – адрес 0A0Ch + x*100h (x=0 или 1)

Биты 31-9 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Бит 8 LBM. Режим внутренней петли.
0 = Режим внутренней петли выключен.
1 = Режим внутренней петли включен. В этом режиме CAN узел подсоединяется к внутренней (виртуальной) CAN шине. CAN узлы, для которых включен режим внутренней петли, объединяются виртуальной CAN шиной и могут виртуально взаимодействовать друг с другом. В режиме внутренней петли на внешних выводах CAN узла (соединенных с внешней CAN шиной) поддерживается рецессивный уровень сигнала, т. е. узел неактивен.

Биты 7-3 Зарезервировано. При чтении возвращает «0». При записи следует писать «0».

Биты 2-0 RXSEL2 – RXSEL0. Выбор входа. Битовое поле выбирает один из выводов CAN узла, для приема сообщений. Например, для CAN узла 0:
000 = выбирается can_rxd_i[0]
001 = выбирается can_rxd_i[1]
…
111 = выбирается can_rxd_i[7]
Дополнительные CAN узлы выбираются также. Например, для CAN узла 1:
000 = выбирается can_rxd_i[8]
001 = выбирается can_rxd_i[9]
…
111 = выбирается can_rxd_i[15]
Сигнал CAN принимается только через выбранный вход.

Младший регистр данных области сообщения MODATALn хранит 4 младших байта данных области сообщения n. Неиспользуемые байты данных заполняются нулями и не участвуют в операциях приема/передачи. Описание бит младшего регистра данных области сообщения MODATALn представлено на рисунке 325.

31	30	29	28	27	26	25	24
DB3.7	DB3.6	DB3.5	DB3.4	DB3.3	DB3.2	DB3.1	DB3.0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
23	22	21	20	19	18	17	16
DB2.7	DB2.6	DB2.5	DB2.4	DB2.3	DB2.2	DB2.1	DB2.0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
15	14	13	12	11	10	9	8
DB1.7	DB1.6	DB1.5	DB1.4	DB1.3	DB1.2	DB1.1	DB1.0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
7	6	5	4	3	2	1	0
DB0.7	DB0.6	DB0.5	DB0.4	DB0.3	DB0.2	DB0.1	DB0.0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 325 – Регистр (младший) данных области сообщения n модуля CAN (MODATALn) – адрес 1810h + n*20h (n=0÷255)

Биты 31-24 DB3.7 – DB3.0. Третий байт данных области сообщения n.

Биты 23-16 DB2.7 – DB2.0. Второй байт данных области сообщения n.

Биты 15-8 DB1.7 – DB1.0. Первый байт данных области сообщения n.

Биты 7-0 DB0.7 – DB0.0. Нулевой байт данных области сообщения n.

Старший регистр данных области сообщения MODATAHn хранит 4 старших байта данных области сообщения n. Неиспользуемые байты данных заполняются нулями и не участвуют в операциях приема/передачи. Описание бит младшего регистра данных области сообщения MODATAHn представлено на рисунке 326.

31	30	29	28	27	26	25	24
DB7.7	DB7.6	DB7.5	DB7.4	DB7.3	DB7.2	DB7.1	DB7.0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
23	22	21	20	19	18	17	16
DB6.7	DB6.6	DB6.5	DB6.4	DB6.3	DB6.2	DB6.1	DB6.0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
15	14	13	12	11	10	9	8
DB5.7	DB5.6	DB5.5	DB5.4	DB5.3	DB5.2	DB5.1	DB5.0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
7	6	5	4	3	2	1	0
DB4.7	DB4.6	DB4.5	DB4.4	DB4.3	DB4.2	DB4.1	DB4.0
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 326 – Регистр (старший) данных области сообщения n модуля CAN (MODATAHn) – адрес 1814h + n*20h (n=0÷255)

Биты 31-24 DB7.7 – DB7.0. Седьмой байт данных области сообщения n.

Биты 23-16 DB6.7 – DB6.0. Шестой байт данных области сообщения n.

Биты 15-8 DB5.7 – DB5.0. Пятый байт данных области сообщения n.

Биты 7-0 DB4.7 – DB4.0. Четвертый байт данных области сообщения n.

16.12 CAN расширение срабатывания по времени (TTCAN)

В дополнение к функциональности CAN по событию детерминированное поведение может быть достигнуто CAN узлом 0 с помощью модуля расширения, который поддерживает функциональность TTCAN. TTCAN протокол соответствует утвержденному плану стандартизации для ISO 11898-4 и полностью соответствует существующему протоколу CAN.

Функциональность управления по времени добавляется в качестве расширения более высокого уровня (уровня сеанса) в протокол CAN может для того, чтобы иметь возможность работать в критичных к безопасности приложениях. Новые функции позволяют детерминировать поведение сети CAN и синхронизацию сетей. Доступно глобальное информационное время. Управление по времени сраба-

тывает на основе механизма планировщика с блоком управления синхронизацией и выделенной синхронизацией части данных.

16.12.1 Особенности TTCAN

Особенностями TTCAN являются:

- Полная поддержка основного цикла и функциональности системной матрицы.
- Поддержка справочных сообщений первого и второго уровней.
- Может использоваться как ведущий (Master) по времени.
- Арбитражные окна поддерживается в режиме управления по времени.
- Доступно глобальное информационное время.
- CAN узел 0 может быть сконфигурирован либо для режима по событию или управления по времени.
- Встроенный механизм планировщика и блок синхронизации времени.
- Записанная защита памяти данных планировщика синхронизации.
- Связанная по времени функции прерывания.
- Защита четности для планировщика памяти.

16.12.2 Генерация локального времени

Номинальная единица времени (NTU) в TTCAN контроллере генерируется с помощью таймера NTU с тактами, полученными от частичного деления (действие происходит с NTU таймером t_q или кратное $1/f_{CAN}$). Сумма NTU и есть локальное время LT, которым является 16-битное значение (следует рассматривать как целое число), а дробная часть LTFR, что показано на рисунке 327.

В уровне 1 TTCAN, LT счетчик увеличивается раз в CAN-разрядное время. Дробная часть LTFR локального времени и TUR не будут приняты во внимание. В уровне 2 TTCAN, LT продлевается своей дробной частью LTFR. Значение TUR добавляется к значению LTFR с периодом обновления t_{upd} для генерации нового значения LT. Если случается переполнение (происходит перенос) после добавления LTFR и TUR, значение LT увеличивается на 1. В результате значение LT можно изменить только с шагом 1, в то время как значение LTFR изменяет с шагом TUR.

Для того, чтобы охватить широкий спектр CAN скорости передачи данных с тем же диапазоном счета локального таймера, подсчет может проводиться каждый CAN доли времени t_q ($t_{upd} = t_q$). Если желательно другое разрешение NTU, локальное время генерации может также быть основано на тактах f_{CAN}, разделенного с помощью программируемого коэффициента, который не зависит от CAN-разрядного времени. Если после периода обновления t_{upd} прошло N раз, то разница в локальном времени d_t прошла на:

$$d_t = \text{TURR.TUR} / 1024 \times N \times t_{upd}. \qquad (2.14)$$

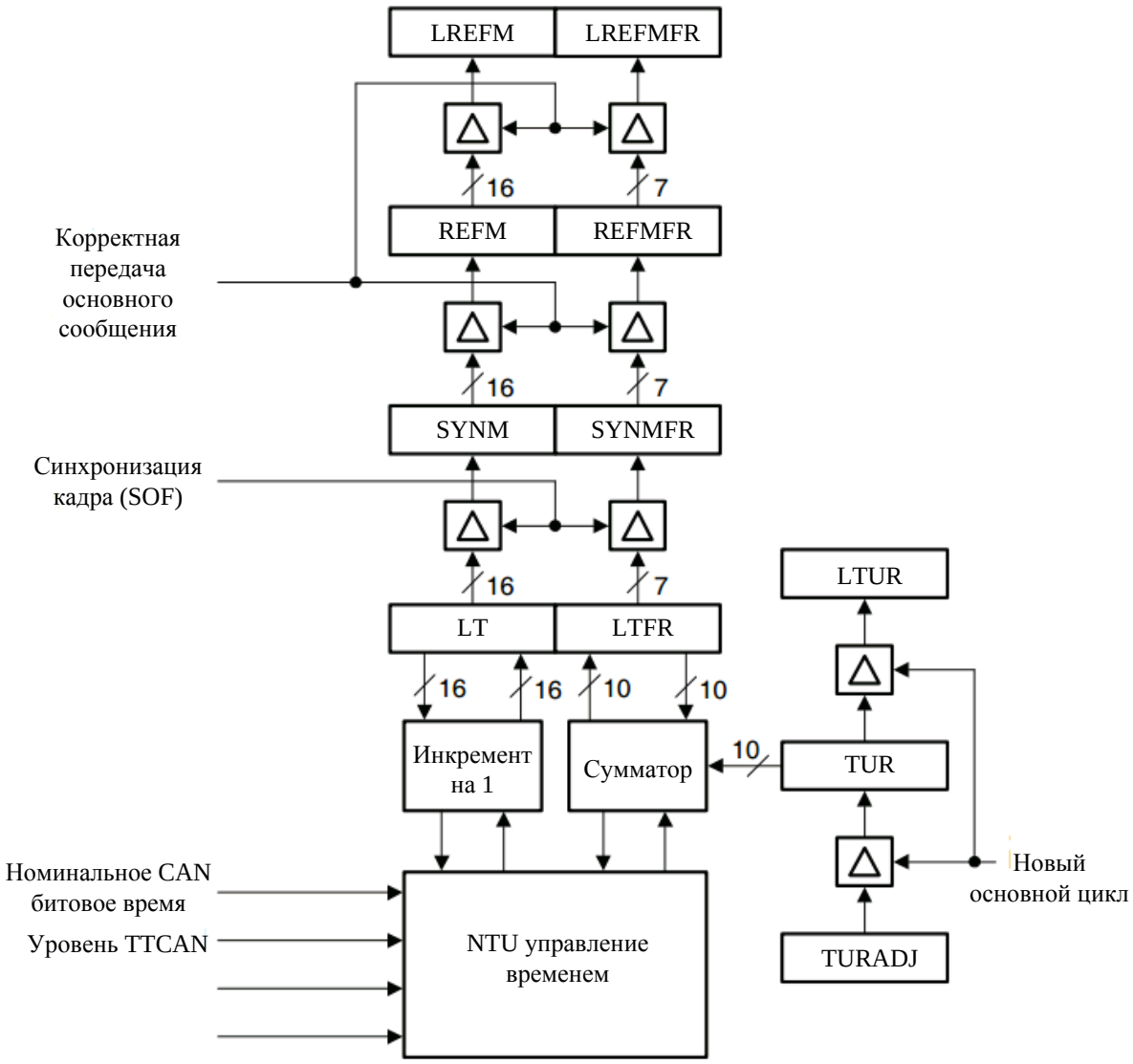


Рисунок 327 – Генерация локального времени

Примечание - Регистр LTUR используется для внутреннего автоматического расчета нового значения TURADJ. Он не виден для пользователя.

16.12.3 Автоматическая настройка TUR

Можно автоматически рассчитать новое значение, которое записывается в TURADJ для регулировки правильного значения для локального времени TTCAN второго уровня. Каждый раз, когда новая ссылка сообщения принята правильно, вычисляется разница между значениями времени в основном сообщении (GMR) и предыдущем основном сообщении (LGMR). Эта разница делится на количество собственного времени модуля NTU для завершения основного цикла и умножается на LTUR, обеспечивая значение TURADJ (в том числе дробных частей).

$$TURADJ = LTUR \times (GM - LGM) / (REFM - LREFM) \qquad (2.15)$$

Автоматическая настройка TUR не делается, когда был пропуск сигнала между основными сообщениями. Автоматический расчет нового значения TURADJ происходит во время основного цикла.

Значение TUR обновляется в начале каждого основного цикла. Новое значение регулировки времени, что записано в TUR, сохраняется в битовом поле TURADJ.

16.12.4 Время цикла

Время цикла всегда является положительным и представляет собой время, прошедшее в текущем основном цикле, начиная с 0.

16.12.5 Локальное время и метки синхронизации

Время цикла - это разница между локальным временем (хранится в битовом поле LT) и его последней основной меткой (хранится в битовом поле REFM). Результат доступен в битовом поле CYCTM. Значение локального времени LT захватывается в начале каждого сообщения с меткой синхронизации SYNM. Если правильно принятое сообщение было основным сообщением, захваченное значение становится меткой REFM. Значение REFM предыдущего отсчёта сохраняется в LREFM.

Время цикла увеличивается еще в то время, как получено основное сообщение. После окончания правильно принятого основного сообщения, время цикла принимает значение, соответствующее длине этого основного сообщения. В результате кажется, что время цикла начинается с нуля при каждом запуске основного сообщения (с каждым новым основным циклом). Генерация времени цикла представлена на рисунке 328.

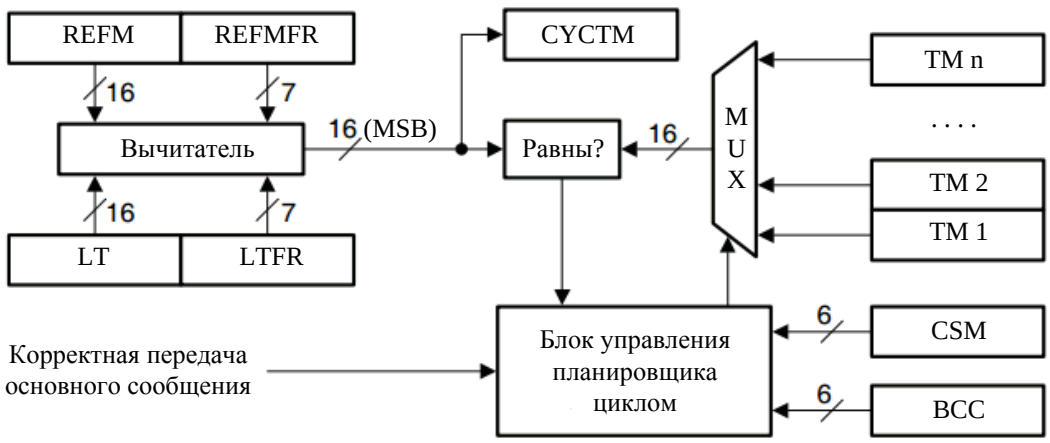


Рисунок 328 – Генерация времени цикла

Блок управления циклом также предоставляет значения для обработки основного цикла и матричного цикла и обеспечивает возможность генерировать прерывание, если начался новый основной цикл или новый матричный цикл. Если TTCAN узел получает основное сообщение, значение BCC берется из эталонного сообщения.

Временная метка

Каждый раз, когда основное сообщение получено правильно, то блок управления циклом начинает снова сравнивать время цикла с первой временной отметкой (TM 1). Если временная метка будет достигнута, она продолжится со следующей единицы. Для того, чтобы иметь возможность сделать "равные" сравнения, временные метки должны быть запрограммированы в порядке возрастания и в первый раз длина метки не должна быть ниже, чем длина основной.

Время цикла используется для определения различных столбцов матрицы системы (CSM = столбец матрицы системы). Эти столбцы определяются временными метками. Эти временные метки могут быть запрограммированы и действительны для всех столбцов матрицы системы.

Блок управления циклом также проверяет номер текущего основного цикла, который указан в счетчике основного цикла BCC. Значение BCC и значение CSM, взятые вместе, четко определяют каждое окно времени в матричном цикле.

В ведущем (Master) режиме, значение BCC передается как часть основного сообщения.

Сторожевой триггер

Сторожевой триггер позволяет проверить, если по истечению времени с момента последнего основного сообщения другое было получено слишком долго. Это делается путем сравнения значения запрограммированного сторожевого триггера до времени цикла. Эта информация может генерировать прерывание к системе. Значение сторожевого триггера определяется записью планировщиком для конца основного цикла.

16.12.6 Ведущая (Master) основная метка (только второй уровень)

Ведущий (Master) по времени отправляет сумму его основной метки SYNM (целая часть) и SYNMFR (дробная часть) и его локального смещения LOF (целая часть) и LOFFR (дробная часть) как ведущий (Master) основной метки (MRM), которая хранится в регистре глобальной метки GMR. Если такты ведущего (Master) или его локальное время было исправлено и был введен пропуск, то бит пропуска DISC автоматически устанавливается до следующего запуска основного сообщения. Бит пропуска устанавливается, если происходит доступ для записи LOF или LOFFR. В результате основное сообщение будет содержать новую основную метку и бит пропуска DISC, установленного на текущего ведущего (Master) по времени.

16.12.7 Окно разрешения передачи

Окно разрешения передачи определяет количество CAN-бит времени, что может пройти между запускаемым событием передачи (меток времени в плани-

ровщике) и реальном стартом сообщения на шине. Если по истечении этого времени не начинается передача запускаемого сообщения, то не будет новой попытки передачи в этом временном окне (если это было исключительно одно или короткое арбитражное одно). Запускаемая передача удаляется и может генерироваться прерывание. Если временное окно долго, окно объединенного арбитража, окно разрешения передачи не учитывается.

16.12.8 Локальное смещение и глобальное время

Для второго уровня TTCAN локальное смещение в узле TTCAN (не ведущего (Master) по времени) определяется как разница между Master_Ref_Mark (также по имени Global_Ref_Mark) и основной меткой узла Ref_Mark. Для ведущего (Master) по времени сумма локального времени и локального смещения хранится в Global_Sync_Mark, когда достигается синхронизация фрейма. Global_Sync_Mark (для ведущего (Master) по времени) и Global_Ref_Mark (для всех других узлов) хранятся в регистре глобальной метки GMR. Local_Offset хранится в регистре локального смещения LOR, показанных на рисунке 329.

Когда TTCAN узел получает основное сообщение, регистр глобальной метки обновляется с полученным значением. Значение, которое было получено в предыдущем основном сообщении, автоматически переводится из регистра глобальной метки до последнего регистра глобальной метки. Разница между этими двумя значениями представляет собой время, прошедшее между двумя последними основными сообщениями с точки зрения ведущего (Master) по времени.

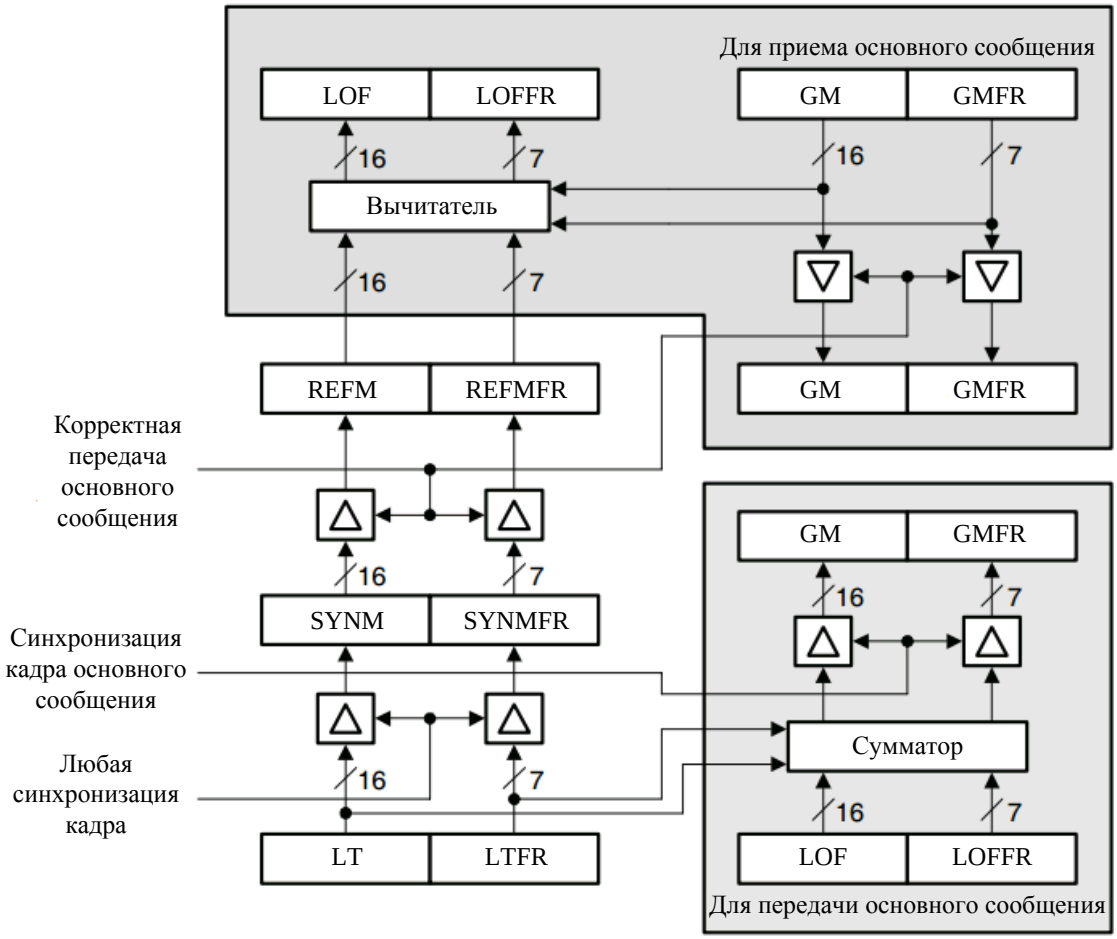


Рисунок 329 - Локальное смещение и глобальное время

В связи с возможными вариациями значений LOF от одной основной метки к другой текущее глобальное время может содержать пропуски. Чтобы получить гладкий (непрерывный) режим, программный фильтр может быть использован для получения узлов. Глобальное время можно рассчитать, как показано ниже:

Локальное время + Локальное смещение (2.16)

или

Время цикла + Основная метка + Локальное смещение (2.17)

16.12.9 Запуск передачи

Особенность TTCAN CAN узла предполагает подсчет запросов передачи для сообщений, которые будут отправлены в исключительных временных окнах. Количество сообщений, которые будут переданы в исключительных временных окнах, определяются системой проектирования приложения для всей матричной системы (ожидание запуска передачи EXPTT). Это теоретическое значение используется для проверки, если реальный TTCAN узел также запрашивает такое же количество передач. Если меньше или больше запускаемых передач были подсчи-

таны с помощью счетчика запуска передачи TTCNT в конце матричной системы, может быть получено прерывание ошибки запуска передачи (если включен TTIEN). Об этом свидетельствуют биты опустошения запуска передачи TTUF и переполнения запуска передачи TTOF. Устройство отслеживания запуска передачи показано на рисунке 330.

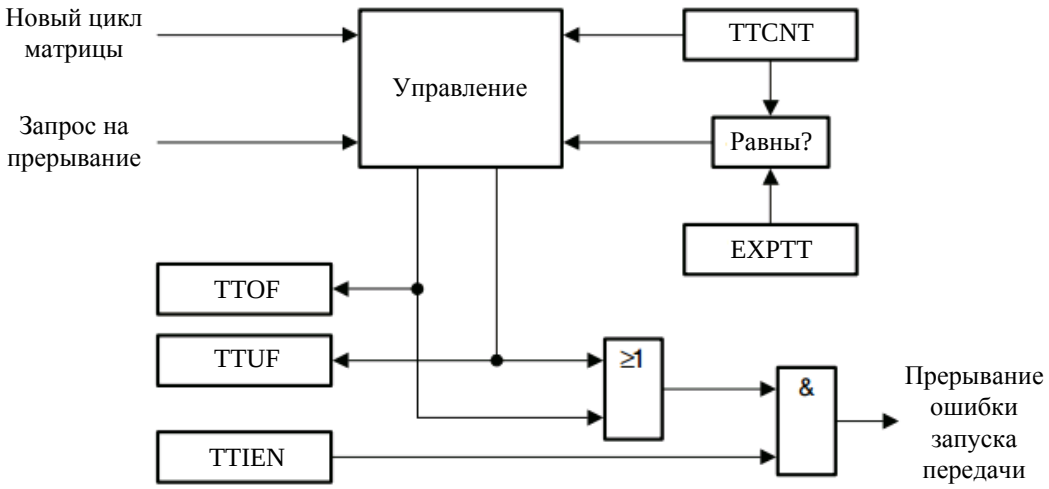


Рисунок 330 – Отслеживание запуска передачи

16.12.10 Основное сообщение

Основная область сообщения имеет особую роль в системе TTCAN. Это первая запись в списке области сообщения для TTCAN узла (узел 0, если узел 0 сконфигурирован для TTCAN).

Различия в нормальных CAN сообщениях

Есть несколько различий между режимом основного сообщения и нормального сообщения CAN:

1. Если основное сообщение нарушается из-за ошибки во время передачи, оно сразу же повторно передается.

2. Если основное сообщение теряет арбитраж, то оно не передается повторно, но сообщение на шине принимается (арбитраж потенциальных ведущих (Master) по времени).

3. Для получения сообщения биты идентификатора (за исключением 3 младших бит ID[2:0]) не используют маску приема (должна быть установлена программно). Полный идентификатор последнего правильного опорного сообщения на шине сохраняется в основной области сообщения.

4. Для получения сообщения три младших бита идентификатора всегда используют маску приема 000_2 (должна быть установлена программно).

5. Запрос на передачу обозначается флагом TTSR.REFTRG и относится только к основной временной метке или внешнему событию (для синхронизации). Если правильно получено основное сообщение во время установленного REFTRG, REFTRG автоматически сбрасывается.

6. Для передачи берутся биты идентификатора основной области сообщения, за исключением трех младших битов (ID[2:0]). Они взяты из битового поля TMCR.TMPRIO.

7. Соответствующие части 8-байтных данных непосредственно связаны с TTCAN внутренними значениями (cycle_count, master_ref_mark и т.д.) в соответствии с выбранным уровнем TTCAN протокола. Остальные байты доступны для свободного использования.

8. DLC основного сообщения, которое будет разослано, определяется битовом полем TTCR.RMDLC. Это битовое поле должно быть по крайней мере установлено в 1 в первом уровне TTCAN и по крайней мере установлено в 4 во втором уровне TTCAN.

Основная область сообщения может быть выбрана ведущим (Master) по времени (потенциальный ведущий (Master) по времени) или приемным устройством (не ведущий (Master) по времени). Основная область сообщения должна быть инициализирована как объект приема (но передача этого объекта приводит к системе отсчета). Это связано с тем, что получение основного сообщения определяет фильтрацию о принятии нормального приема, в то время как передача определяется действием, вызванным TTCAN расширением. Основное сообщение также может быть передано ведущим (Master) по времени, которое находится в состоянии ошибки S2 или было достигнуто после Init_Watch_Trigger.

16.12.11 Запуск передачи для основного сообщения

Передача основного сообщения может быть запущена временной меткой или внешним событием. Если это запущено временной меткой, бит "Следующий промежуток" (TTSR.NIG) из предыдущего основного сообщения будет 0. Если это вызвано внешним событием, бит "Следующий промежуток" из предыдущего основного сообщения будет 1.

Программно можно установить бит TTSR.NIG, чтобы инициировать синхронизацию основного сообщения на события. Это событие может быть крайним на внешнем входе или программное действие (запись в бит TTFMR.STE = 1, программный запуск события).

При установлении TTSR.NIG, он будет передан в следующее основное сообщение, чтобы указать, что промежуток синхронизации может следовать к другому CAN узлу. Значение TTSR.NIG копируется в бит TTSR.ETR (запрос внешнего запуска) и очищается автоматически, когда основное сообщение было передано правильно. Бит TTSR.ETR = 1 указывает, что следующее основное сообщение будет передано, как только будет достигнута соответствующая временная метка (включая промежуток), или когда выбор запуска события (ETREV) находится в ожидании. Вычисление ETREV не стартует после 1 CAN битного времени и после остановки разрешения окна передачи в последнее окно передачи.

Логика генерации запуска передачи для основного сообщения показана на рисунке 331. Запуск события может быть выбран в битовом поле ETESEL (выбор

внешнего запуска события). Только запускаемые события, которые были обнаружены после начала текущего основного цикла, учитываются для передачи основного сообщения. Часть процесса обнаружения для ttc_ectt_i [7:1] (ECTT1-ECTT7 на рисунке 331) содержит этап синхронизации.

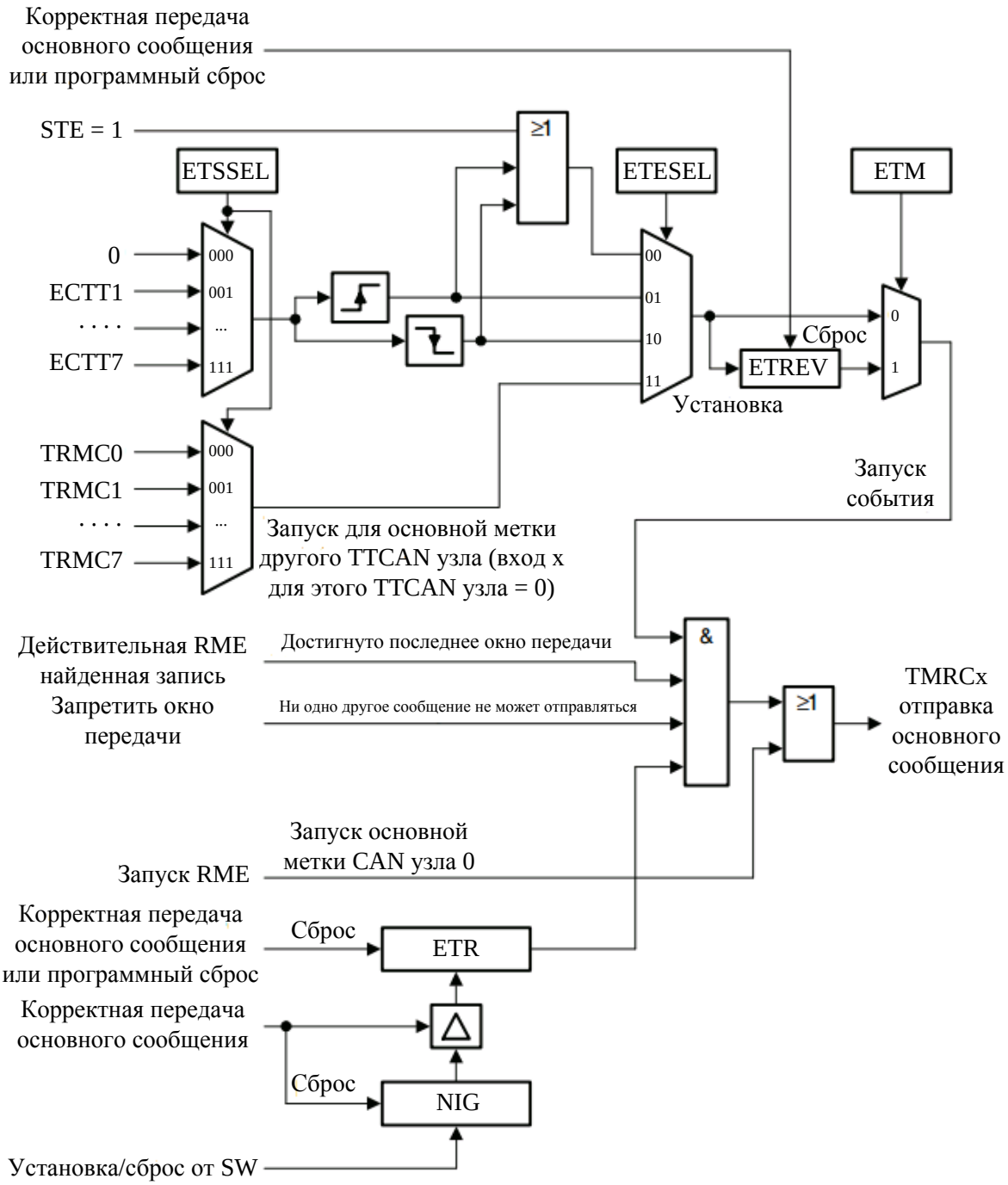


Рисунок 331 – Генерация внешнего запуска для основного сообщения

16.13 Планировщик TTCAN

Передача сообщения и прием в TTCAN контролируется механизмом планировщика. Этот планировщик основывается на работе по времени цикла и обеспечивается временными метками. Временные метки определяются временными метками входных данных TMEx. Всякий раз, когда будет достигнута временная метка, могут совершаться программируемые действия (определенные команды входных данных памяти планировщика INSTRxy). Типовые инструкции включают в себя начало передачи сообщения, проверку, если было получено сообщение, открытия или закрытия окна арбитража или генерация прерываний. На рисунке 332 показана структура планировщика.

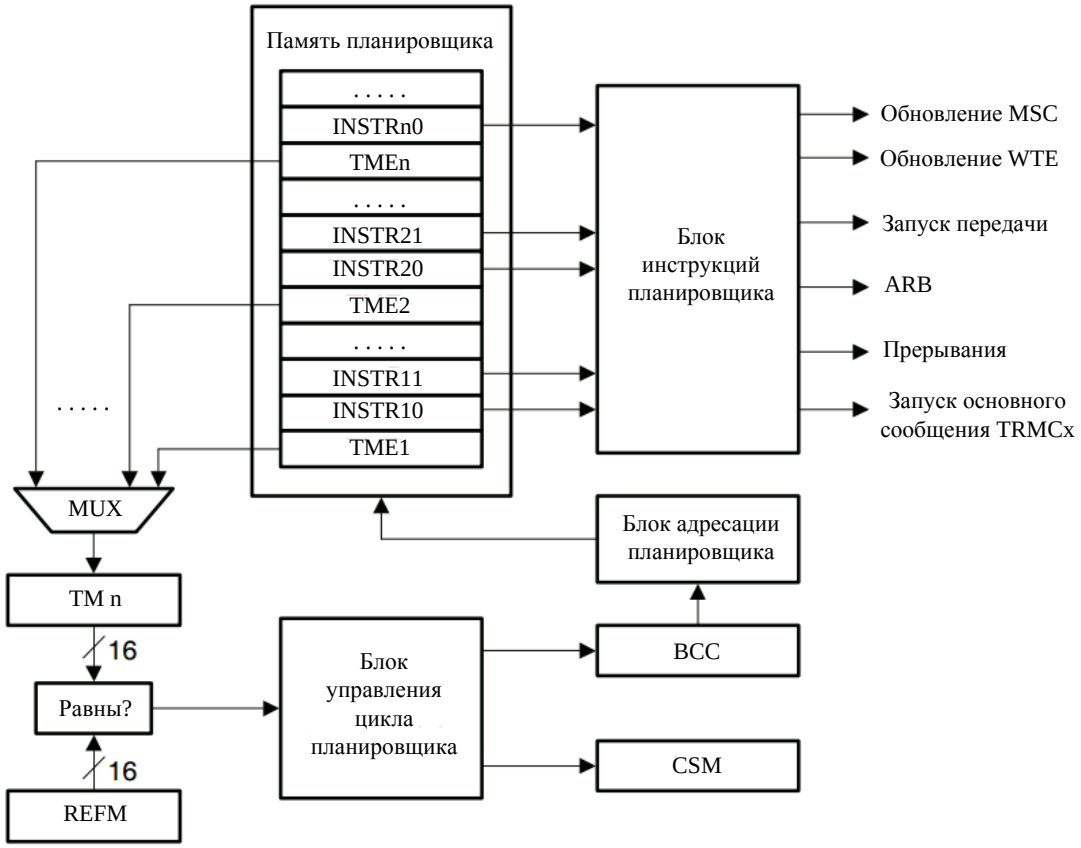


Рисунок 332 – Обзор планировщика

Инструкции, следующие за временной меткой, считываются планировщиком, пока следующая временная метка входных данных не будет найдена. Затем процесс набора инструкции останавливается, пока не будет достигнута следующая временная метка.

Примечание - Значения временной метки входных данных в планировщике должно быть в порядке возрастания.

16.13.1 Память планировщика

Планировщик содержит блок памяти, в котором записаны временные метки входных данных (TMEx) и хранятся команды входных данных планировщика (INSTRnx). Каждая временная метка входных данных может сопровождаться рядом инструкций планировщика. Память планировщика организована как 32-битная память, хранящая 32-битную временную метку входных данных и 32-битную инструкцию планировщика.

Общее количество временных меток входных данных и команд планировщика, которые могут храниться, ограничивается размером памяти планировщика (128 32-разрядных слов). На рисунке 333 представлена структура памяти планировщика.

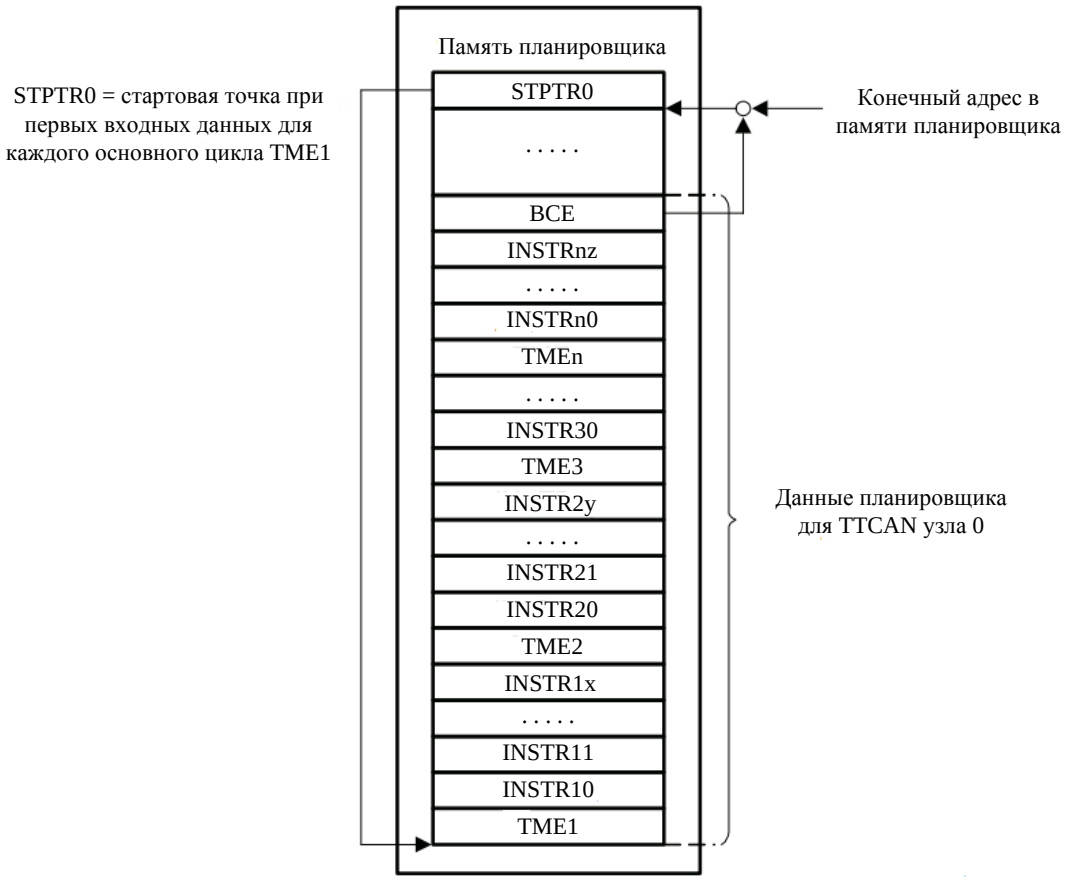


Рисунок 333 – Память планировщика TTCAN

Последнее адресное слово в памяти планировщика зарезервировано для стартового указателя STPTR0. Значение, записанное по этому адресу, определяет начальную точку первых входных данных для TTCAN узла (узел CAN 0, если сконфигурирован для TTCAN). STPTR0 указывает, на сколько 32-битных (слово) входных данных ниже SPTR0 находится первая временная отметка (TME1).

Когда заканчивается основной цикл входных данных, BCE (с GM = 0) читается планировщиком, и он готовит следующие инструкции планировщика (читает из памяти планировщика), начиная снова с TME1. Таким образом, первое чтение входных данных после BCE является STPTR0 для того, чтобы получить адрес, где находится TME1.

Типы планировщика входных данных

Инструкции входных данных планировщика размещены после временной метки входных данных. Временная метка входных данных определяет поведение TTCAN, когда будет достигнута следующая временная метка. Инструкции планировщика после временной метки входных данных действительны для следующей временной метки. Планировщик регулярно считывает временные метки входных данных и устанавливает сравнение (сравнивает между новой временной меткой и временем цикла). Затем, планировщик считывает следующие инструкции и устанавливает режим передачи TTCAN узла. Планировщик прекращает чтение из памяти планировщика, когда будет найдена новая временная метка входных данных.

Настройка режима передачи может меняться с каждой инструкцией планировщика. Нет никакой встроенной проверки достоверности для инструкций планировщика. Ранее прочитанные инструкции планировщиком могут быть перезаписаны последующими. С каждой временной меткой, настройки сбрасываются, и должны быть установлены по новой инструкции планировщика (при желании).

Полная информация для планировщика должна быть передана в последний основной цикл входных данных, который фиксируется значение для сторожевого триггера. Каждые входные данные в памяти планировщика содержат 4-битное поле, которое определяет тип входных данных. Таблица 152 перечисляет возможные типы входных данных в планировщике памяти.

Таблица 152 – Типы входных данных в планировщике памяти

Биты входного кода EC [31:28]	Обозначение	Тип входных данных
0001_2	TME	Временная метка входных данных. Подробно на рисунке 334
0010_2	ICE	Управление прерываниями входных данных. Подробно на рисунке 335
0011_2	ARBE	Арбитраж входных данных. Подробно на рисунке 336
0100_2	TCE	Управление передачей входных данных. Подробно на рисунке 337
0101_2	RCE	Управление приемом входных данных. Подробно на рисунке 338
0110_2	RME	Основное сообщение входных данных. Подробно на рисунке 339
0111_2	BCE	Последний основной цикл входных данных. Подробно на рисунке 340
Другие комбинации	EOS	Последние входные данные в памяти планировщика

Общий запуск управления передачей для областей сообщений должен быть передан, когда достигаемая временная метка осуществляется самими областями сообщений. Для повышения гибкости в результате приема проделанная фильтрация с областями сообщений может быть отклонена контроллером входных данных. Основное сообщение может быть передано, только когда CAN узел включен в качестве ведущего (Master) по времени.

Примечание - Количество входных данных планировщика, следующих за временной меткой, не должно превышать 10. Чтобы свести к минимуму требуемый доступ к памяти планировщика по части контроля планировщика, пользователь должен тщательно настроить систему.

16.13.2 Описание типов входных данных планировщика

Временная метка входных данных (TME) определяется следующим способом:

31	30	29	28	27 … 26	25	24
EC3	EC2	EC1	EC0	Зарезервировано	ARBM1	ARBM0
RW-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0

23	22	21	20	19	18	17	16
IENRECF1	IENRECF0	IENTRAF1	IENTRAF0	INP3	INP2	INP1	INP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

15	14	13	12	11	10	9	8
TMV15	TMV14	TMV13	TMV12	TMV11	TMV10	TMV9	TMV8
RW-0	RW-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0

7	6	5	4	3	2	1	0
TMV7	TMV6	TMV5	TMV4	TMV3	TMV2	TMV1	TMV0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 334 – Временная метка входных данных

Биты 31-28 EC3 – EC0. Код входных данных. EC = 0001_2 определяет входные данные в памяти планировщика как временная метка входных данных.

Биты 27-26 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0001_2.

Биты 25-24 ARBM1, ARBM0. Режим арбитража. Выполняет действие с арбитражным окном:

00 = Нет принятых действий; статус арбитражного окна не изменился.
01 = Объединенное окно арбитража открыто. Если оно уже открыто, то остается открытым.
10 = Объединенное (длинное) окно передачи закрыто. Закрытие объединенного окна передачи приводит к одному (короткому) окну арбитража (передача возможна только при разрешенном окне передачи). Короткое окно арбитража автоматически закрывается после одного временного окна. Если нет открытого окна арбитража, эти входные данные игнорируется.
11 = Одно (короткое) окно арбитража (передача возможна только во время разрешенного окна передачи) открыто. Короткое окно арбитража автоматически закрывается после одного временного окна.

Бит 23 IENRECF1. Разрешение прерывания, если RECF = 1. Разрешает генерацию прерывания, когда бит TTSR.RECF = 1 и обнаруживать совпадение между временем цикла и значение временной метки, определяемой TMV:
0 = Генерация прерывания запрещена.
1 = Генерация прерывания разрешена.

Бит 22 IENRECF0. Разрешение прерывания, если RECF = 0. Разрешает генерацию прерывания, когда бит TTSR.RECF = 0 и обнаруживать совпадение между временем цикла и значение временной метки, определяемой TMV:
0 = Генерация прерывания запрещена.
1 = Генерация прерывания разрешена.

Бит 21 IENTRAF1. Разрешение прерывания, если TRAF = 1. Разрешает генерацию прерывания, когда бит TTSR.TRAF = 1 и обнаруживать совпадение между временем цикла и значение временной метки, определяемой TMV:
0 = Генерация прерывания запрещена.
1 = Генерация прерывания разрешена.

Бит 20 IENTRAF0. Разрешение прерывания, если TRAF = 0. Разрешает генерацию прерывания, когда бит TTSR.TRAF = 0 и обнаруживать совпадение между временем цикла и значение временной метки, определяемой TMV:
0 = Генерация прерывания запрещена.
1 = Генерация прерывания разрешена.

Биты 19-16 INP3 – INP0. Узел указателя прерывания. Выбирает выходную линию прерывания (int_req_o[15: 0]), которая будет активироваться при обнаружении совпадения между временем цикла и значением временной метки, определяемой TMV, и если по крайней мере одно из четырех условий прерывания выполняется и разрешено:
0000_2 = Выбрана выходная линия прерывания int_req_o[0].
0001_2 = Выбрана выходная линия прерывания int_req_o[1].

…
1110_2 = Выбрана выходная линия прерывания int_req_o[14].
1111_2 = Выбрана выходная линия прерывания int_req_o[15].

Биты 15-0[1)] TMV15 – TMV0. Значение временной метки. Определяет сравниваемое значение, используемое для следующего сравнительного действия со временем цикла.

[1)] Чтобы иметь возможность обнаруживать каждый раз правильную временную метку, значение первой временной метки основного цикла не должно быть меньше длины основного сообщения при наихудших условиях.

Примечание - Настройка прерывания данных по временной метке входных данных может быть отменена последующим действительным управлением прерывания входных данных. Настройки действительны до того, когда будет достигнута следующая временная метка. За время после следующей временной метки настройки считываются, когда следующая временная метка входных данных становится действительной.

Управление прерываниями входных данных (ICE) определяется следующим способом:

31	30	29	28	27 … 24
EC3	EC2	EC1	EC0	Зарезервировано
RW-0	RW-0	RW-0	RW-0	R-0

23	22	21	20	19	18	17	16
IENRECF1	IENRECF0	IENTRAF1	IENTRAF0	INP3	INP2	INP1	INP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

15 … 14	13	12	11	10	9	8
Зарезервировано	MCYCLE5	MCYCLE4	MCYCLE3	MCYCLE2	MCYCLE1	MCYCLE0
R-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0

7 … 6	5	4	3	2	1	0
Зарезервировано	CYCLE5	CYCLE4	CYCLE3	CYCLE2	CYCLE1	CYCLE0
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 335 – Управление прерываниями входных данных

Биты 31-28 EC3 – EC0. Код входных данных. EC = 0010_2 определяет входные данные в памяти планировщика как управление прерываниями вход-

ных данных.

Биты 27-24 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0010_2.

Бит 23 IENRECF1. Разрешение прерывания, если RECF = 1. Разрешает генерацию прерывания, когда бит TTSR.RECF = 1 и обнаруживать совпадение между временем цикла и значение временной метки, определяемой TMV:
0 = Генерация прерывания запрещена.
1 = Генерация прерывания разрешена.

Бит 22 IENRECF0. Разрешение прерывания, если RECF = 0. Разрешает генерацию прерывания, когда бит TTSR.RECF = 0 и обнаруживать совпадение между временем цикла и значение временной метки, определяемой TMV:
0 = Генерация прерывания запрещена.
1 = Генерация прерывания разрешена.

Бит 21 IENTRAF1. Разрешение прерывания, если TRAF = 1. Разрешает генерацию прерывания, когда бит TTSR.TRAF = 1 и обнаруживать совпадение между временем цикла и значение временной метки, определяемой TMV:
0 = Генерация прерывания запрещена.
1 = Генерация прерывания разрешена.

Бит 20 IENTRAF0. Разрешение прерывания, если TRAF = 0. Разрешает генерацию прерывания, когда бит TTSR.TRAF = 0 и обнаруживать совпадение между временем цикла и значение временной метки, определяемой TMV:
0 = Генерация прерывания запрещена.
1 = Генерация прерывания разрешена.

Биты 19-16 INP3 – INP0. Узел указателя прерывания. Выбирает выходную линию прерывания (int_req_o[15: 0]), которая будет активироваться при обнаружении совпадения между временем цикла и значением временной метки, определяемой TMV, и если по крайней мере одно из четырех условий прерывания выполняется и разрешено:
0000_2 = Выбрана выходная линия прерывания int_req_o[0].
0001_2 = Выбрана выходная линия прерывания int_req_o[1].
…
1110_2 = Выбрана выходная линия прерывания int_req_o[14].
1111_2 = Выбрана выходная линия прерывания int_req_o[15].

Биты 15-14 Зарезервировано. Читаются как «0». При записи следует писать «0».

Биты безразличны для операций планировщика, когда EC = 0010_2.

Биты 13-8 MCYCLE5 – MCYCLE0. Маска для цикла сравнения. Определяет маску, используемую для определения частоты повторения для этого планировщика входных данных. Это битное поле эквивалентно соответствующей части битного поля MOAMRn.AM.

Биты 7-6 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0010_2.

Биты 5-0 CYCLE5 – CYCLE0. Количество основных циклов. Определяет количество основного цикла, в течение которого действует управление прерываниями входных данных. Значение CYCLE сравнивается (побитово) с текущим значением битового поля CYCTMR.BCC. Затем результат маскируется со значением, заданным в битовом поле MCYCLE, чтобы определить частоту повторения для данного планировщика входных данных внутри матричного цикла.

Примечание – Управление прерываниями входных данных может отменить ранее сохраненные настройки прерываниями для следующей временной метки. В результате, следует избегать использования более чем одного сообщения управления прерываниями входных данных между двумя временными метками входных данных.

Арбитраж входных данных (ARBE) определяется следующим способом:

31	30	29	28	27 – 26	25	24
EC3	EC2	EC1	EC0	Зарезервировано	ARBM1	ARBM0
RW-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0

23 – 14	13	12	11	10	9	8
Зарезервировано	MCYCLE5	MCYCLE4	MCYCLE3	MCYCLE2	MCYCLE1	MCYCLE0
R-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0

7 – 6	5	4	3	2	1	0
Зарезервировано	CYCLE5	CYCLE4	CYCLE3	CYCLE2	CYCLE1	CYCLE0
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 336 – Арбитраж входных данных

Биты 31-28 EC3 – EC0. Код входных данных. EC = 0011_2 определяет входные данные в памяти планировщика как арбитраж входных данных.

Биты 27-26 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0011_2.

Биты 25-24 ARBM1, ARBM0. Режим арбитража. Выполняет действие с арбитражным окном:
00 = Нет принятых действий; статус арбитражного окна не изменился.
01 = Объединенное окно арбитража открыто. Если оно уже открыто, то остается открытым.
10 = Объединенное (длинное) окно передачи закрыто. Закрытие объединенного окна передачи приводит к одному (короткому) окну арбитража (передача возможна только при разрешенном окне передачи). Короткое окно арбитража автоматически закрывается после одного временного окна. Если нет открытого окна арбитраж, эти входные данные игнорируется.
11 = Одно (короткое) арбитраж окно (передача возможна только во время разрешенного окна передачи) открыто. Короткое окно арбитража автоматически закрывается после одного временного окна.

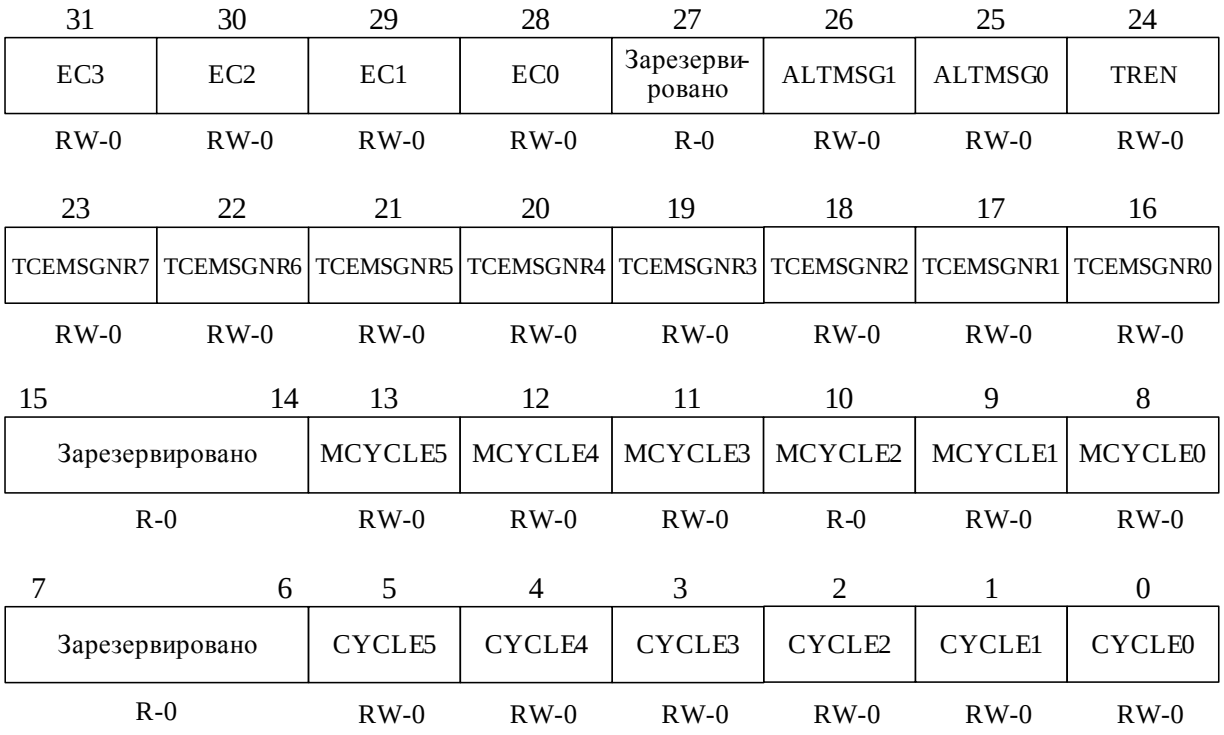
Бит 23-14 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0011_2.

Биты 13-8 MCYCLE5 – MCYCLE0. Маска для цикла сравнения. Определяет маску, используемую для определения частоты повторения для этого арбитража входных данных. Это битное поле эквивалентно соответствующей части битного поля MOAMRn.AM (см. рисунок 304).

Биты 5-0 CYCLE5 – CYCLE0. Количество основных циклов. Определяет количество основного цикла, в течение которого действует арбитраж входных данных. Значение CYCLE сравнивается (побитово) с текущим значением битового поля CYCTMR.BCC. Затем результат маскируется со значением, заданным в битовом поле MCYCLE, чтобы определить частоту повторения для данного планировщика входных данных внутри матричного цикла.

Примечание - Арбитраж входных данных может отменить ранее сохраненные настройки арбитража для следующей временной метки. В результате, следует избегать использования более чем одного сообщения арбитража входных данных между двумя временными метками входных данных.

Управление передачей входных данных (TCE) определяется следующим способом:



31	30	29	28	27	26	25	24
EC3	EC2	EC1	EC0	Зарезерви-ровано	ALTMSG1	ALTMSG0	TREN
RW-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0	RW-0

23	22	21	20	19	18	17	16
TCEMSGNR7	TCEMSGNR6	TCEMSGNR5	TCEMSGNR4	TCEMSGNR3	TCEMSGNR2	TCEMSGNR1	TCEMSGNR0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

15 14	13	12	11	10	9	8
Зарезервировано	MCYCLE5	MCYCLE4	MCYCLE3	MCYCLE2	MCYCLE1	MCYCLE0
R-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0

7 6	5	4	3	2	1	0
Зарезервировано	CYCLE5	CYCLE4	CYCLE3	CYCLE2	CYCLE1	CYCLE0
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 337 – Управление передачей входных данных

Биты 31-28 EC3 – EC0. Код входных данных. EC = 0100_2 определяет входные данные в памяти планировщика как арбитраж входных данных.

Бит 27 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0100_2.

Биты 26-25 ALTMSG1, ALTMSG0. Альтернативное сообщение. Определяет, из какой области сообщения будет передаваться:
00 = Если номер области сообщения доставленного этим элементом не действителен для передачи, сообщение не будет отправлено.
01 = Номер области сообщения доставленных входных данных не будет принят во внимание для передачи, из области сообщения, находящейся в фильтровании приема переданного будет отправлено, если оно действительно.
10 = Если пришло из области сообщения, что входные данные не действительны для передачи, в то время как фильтрование приема переданного из области сообщения выдает правильный объект, то одно последнее будет отослано наружу.
11 = Зарезервировано

Бит 24 TREN. Разрешение передачи. Разрешает передачу из области сообщения в следующее временное окно вместе с запуском передачи TTCAN.
0 = Ни запуск передачи TTCAN, ни область сообщения приема-передачи приведет к передаче из области сообщения в следующее временное окно.
1 = Если запуск передачи TTCAN или область сообщения, находящуюся в фильтровании приема переданного будет действительной областью сообщения, то передача из области сообщения в следующем временном окне будет разрешена.

Биты 23-16 TCEMSGNR7 – TCEMSGNR0. Номер сообщения TCE. Определяет номер области сообщения, из которой должно быть передано, когда будет достигнута следующая временная метка. Из области сообщения будет передаваться, только если оно действительно нужно для передачи. С планировщиком инструкций номер области сообщения определяется непосредственно, без принятия во внимание, которое назначено текущему узлу. Даже если присваивается другой CAN узел, из выбранной области сообщения будет передаваться.
Если CAN фильтрование приема переданного может передать номер области, оно будет отклонено действующим управлением передачей входных данных в соответствии с ALTMSG.
Если из области сообщения сообщение, доставленное в управление передачей входными данными, не действительно для передачи, передача не будет запущена.
Если выполнение поддержки контроллера CAN меньше, чем 256 областей сообщений, более высокие значения для MSGNR, чем количество фактически поддерживаемых областей сообщений, сообщение будет рассматриваться с работой по модулю (для примера, если поддерживаются 128 объектов, значение MSGNR = 135 равносильно MSGNR = 7).

Биты 15-14 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0100_2.

Биты 13-8 MCYCLE5 – MCYCLE0. Маска для цикла сравнения. Определяет маску, используемую для определения частоты повторения для этого управления передачей входных данных. Это битное поле эквивалентно соответствующей части битного поля MOAMRn.AM (см. рисунок 304).

Биты 7-6 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0100_2.

Биты 5-0 CYCLE5 – CYCLE0. Количество основных циклов. Определяет количество основного цикла, в течение которого действует управление передачей входных данных. Значение CYCLE сравнивается (побитово) с текущим значением битового поля CYCTMR.BCC. Затем результат маскируется со значением, заданным в битовом поле MCYCLE, чтобы определить частоту повторения для управления передачей входных данных внутри матричного цикла.

Примечание - Управление передачей входных данных может отменить ранее сохраненные настройки передачи и также может отменить фильтрование приема переданных сообщений из области сообщения для следующей временной метки (в зависимости от разрядов ALTMSG). В результате, следует избегать использование более чем одного сообщения управления передачей входных данных между двумя временными метками входных данных.

Управление приемом входных данных (RCE) определяется следующим способом:

31	30	29	28	27 – 25	24
EC3	EC2	EC1	EC0	Зарезервировано	CHEN
RW-0	RW-0	RW-0	RW-0	R-0	RW-0

23	22	21	20	19	18	17	16
RCEMSGNR7	RCEMSGNR6	RCEMSGNR5	RCEMSGNR4	RCEMSGNR3	RCEMSGNR2	RCEMSGNR1	RCEMSGNR0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

15 – 14	13	12	11	10	9	8
Зарезервировано	MCYCLE5	MCYCLE4	MCYCLE3	MCYCLE2	MCYCLE1	MCYCLE0
R-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0

7 – 6	5	4	3	2	1	0
Зарезервировано	CYCLE5	CYCLE4	CYCLE3	CYCLE2	CYCLE1	CYCLE0
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 338 – Управление приемом входных данных

Биты 31-28 EC3 – EC0. Код входных данных. EC = 0101_2 определяет входные данные в памяти планировщика как арбитраж входных данных.

Биты 27-25 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0101_2.

Бит 24 CHEN. Проверка разрешения. Только временные окна с активным управлением приема входных данных и CHEN = 1 будут обрабатываться с обновлением битового поля MSC в указателе сообщения объекта MSGNR. Если был получен правильный фрейм и также хранится в указанной области сообщения, то прием считается правильным; в противном случае, это ошибка:
0 = Битовое поле MSC из указанной области сообщения не обновляется.
1 = Битовое поле MSC из указанной области сообщения обновляется.

Биты 23-16 RCEMSGNR7 – RCEMSGNR0. Номер сообщения RCE. Определяет номер области сообщения, что проверяется для правильности приема сообщения в течение последнего окна передачи. Полученное сообщение всегда сохраняется в области сообщения, которое определяется путем фильтрации приема. При достижении новой временной метки, RCE может быть использовано для проверки, если требуемое сообщение фактически было получено в нужной области сообщения.

Биты 15-14 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0100_2.

Биты 13-8 MCYCLE5 – MCYCLE0. Маска для цикла сравнения. Определяет маску, используемую для определения частоты повторения для этого планировщика входных сообщений. Это битное поле эквивалентно соответствующей части битного поля MOAMRn.AM (см. рисунок 304).

Биты 7-6 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0100_2.

Биты 5-0 CYCLE5 – CYCLE0. Количество основных циклов. Определяет количество основного цикла, в течение которого действует управление приемом входных данных. Значение CYCLE сравнивается (побитово) с текущим значением битового поля CYCTMR.BCC. Затем результат маскируется со значением, заданным в битовом поле MCYCLE, чтобы определить частоту повторения для управления передачей входных данных внутри матричного цикла.

Примечание - Управление приемом входных данных может отменить ранее сохраненную информацию. В результате, следует избегать использования более чем одного сообщения управления приемом входных данных между двумя временными метками входных данных.

Примечание - Если не найдено корректное управление приемом входных данных только для текущего временного окна, прием и хранение сообщений зависит только от фильтрования приема полученного сообщения из области сообщения. Только области сообщения, получающие корректные сообщения при управлении приемом входных данных, изменяют битовое поле MSC соответственно. Битовое поле MSC приемной области неизменно без корректного управления

приемом.

Основное сообщение входных данных (RME) определяется следующим способом:

31	30	29	28	24	26 … 16
EC3	EC2	EC1	EC0	GM	Зарезервировано
RW-0	RW-0	RW-0	RW-0	RW-0	R-0

15	14	13	12	11	10	9	8
TMV15	TMV14	TMV13	TMV12	TMV11	TMV10	TMV9	TMV8
RW-0	RW-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0

7	6	5	4	3	2	1	0
TMV7	TMV6	TMV5	TMV4	TMV3	TMV2	TMV1	TMV0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 339 – Основное сообщение входных данных

Биты 31-28 EC3 – EC0. Код входных данных. EC = 0110_2 определяет входные данные в памяти планировщика как арбитраж входных данных.

Бит 27 GM. Режим пропуска. Определяет, как планировщик переходит в режим ведущего (Master) по времени, если TTCAN узел в «режиме пропуска».
0 = Основное сообщение будет отправлено как основное сообщение входных данных (без соблюдения возможного пропуска).
1 = Если TTCAN узел в «режиме пропуска», основное сообщение входных данных отбрасывается и читаются следующие входные данные.

Биты 26-16 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0110_2.

Биты 15-0 TMV15 – TMV0. Значение временной метки. Определяет сравниваемое значение, используемое для следующего сравнительного действия со временем цикла. Если CAN узел является ведущим (Master) по времени, то основное сообщение должно быть отправлено, когда достигается временная метка (см. также TMR.RTO).

Примечание - Эти входные данные учитываются для ведущего (Master) по времени. В случае обнаружения более корректных основных сообщений вход-

ных данных, настройки ранее найденных основных сообщений входных данных отменяются следующими обнаруженными.

Если TTCAN узел не настроен как ведущий по времени и планировщик считывает RME, генерируется ошибка конфигурации. В то время как система находится в состоянии синхронизации, учитывается основное сообщение входных данных с GM = 1.

Последний основной цикл входных данных

Последний основной цикл входных данных определяется значением временной метки, которая используется в качестве временной метки сторожевого триггера в том случае, если система TTCAN ждет основное сообщение, а система, как правило, синхронизирована (не в «режиме пропуска», не дожидаясь другого события при синхронизации). Если система находится в «режиме пропуска» с установленным GM, можно прочитать следующие входные данные.

Последний основной цикл входных данных (BCE) определяется следующим способом:

31	30	29	28	24	26 … 16
EC3	EC2	EC1	EC0	GM	Зарезервировано
RW-0	RW-0	RW-0	RW-0	RW-0	R-0

15	14	13	12	11	10	9	8
TMV15	TMV14	TMV13	TMV12	TMV11	TMV10	TMV9	TMV8
RW-0	RW-0	RW-0	RW-0	RW-0	R-0	RW-0	RW-0

7	6	5	4	3	2	1	0
TMV7	TMV6	TMV5	TMV4	TMV3	TMV2	TMV1	TMV0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 340 – Последний основной цикл входных данных

Биты 31-28 EC3 – EC0. Код входных данных. EC = 0111_2 определяет входные данные в памяти планировщика как арбитраж входных данных.

Бит 27 GM. Режим пропуска. Определяет, как планировщик переходит в режим ведущего (Master) по времени и ждет возникновения события, чтобы отправить основное сообщение (для текущего ведущего (Master) по времени), или когда он ждет приема основного сообщения:
0 = Событие для сторожевого триггера будет сформировано в соответствии с последним основным циклом входных данных (без соблюдения возможного пропуска).

1 = Если TTCAN узел в «режиме пропуска», последний основной цикл входных данных отбрасывается и читаются следующие входные данные.

Биты 26-16 Зарезервировано. Читаются как «0». При записи следует писать «0». Биты безразличны для операций планировщика, когда EC = 0111_2.

Биты 15-0 TMV15 – TMV0. Значение временной метки. Определяет сравниваемое значение, используемое для следующего сравнительного действия со временем цикла.

Примечание - Последний основной цикл определяет конечные входные данные планировщика в памяти планировщика, когда система, как правило, синхронизирована. Когда временная метка этих входных данных будет достигнута, TTCAN узел автоматически переходит в режим настройки. TTCAN узел продолжает с первой временной метки, когда основное сообщение было правильно передано по CAN шине.

В то время как система находится в состоянии синхронизации, также принимается во внимание и последний основной цикл входных данных с GM = 1.

Последние входные данные в памяти планировщика

Последние входные данные в памяти планировщика EOS определяются входными кодами EC = 0000_2 или EC = $1XXX_2$ (таблица 152). EOS немедленно останавливает чтение входных данных в памяти планировщика и устанавливает TTCAN узел в режим конфигурирования.

EOS можно рассматривать как своего рода аварийная остановка входных данных. Он не должен быть использован для управления планировщиком, а представляет собой механизм безопасности для случая, когда планировщик входных данных не показывает корректное битовое поле EC.

16.13.3 Установка планировщика входных данных

Входные данные в памяти планировщика можно настроить только тогда, когда NCR0.CCE = 1. Действия записи в память планировщика, когда NCR0.CCE = 0, не учитываются. Планировщика входных данных всегда можно прочитать для проверочных целей.

16.13.4 Чтение планировщика входных данных

После достигнутой временной метки, инструкции планировщика для следующего временного окна читаются планировщиком. Эта "собранная" информация может быть считана из статуса планировщика синхронизацией и от регистра статуса команд планировщика. Информация становится действительной, когда будет достигнута следующая временная метка. Собранная информация между временной меткой n-1 и временной меткой n становится действительной, когда достигается временная метка n, что показано на рисунке 341.

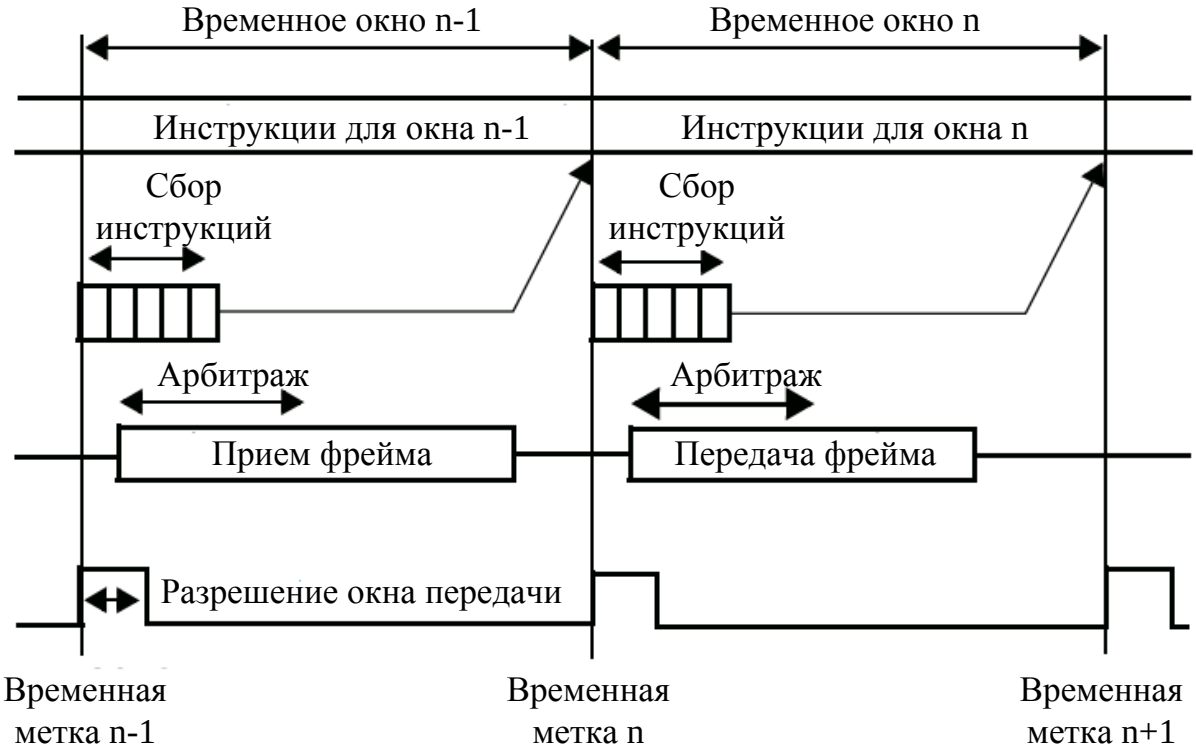


Рисунок 341 – Сбор инструкций

Инструкции во время основного цикла

Обработка входных данных, собранных между отметками времени n-1 и n, определяется следующим образом:

а) Временная метка входных данных – определенное значение временной метки сравнивается со значением временной метки n.

б) Прерывание входных данных - действующее прерывание, информация (INP + 4 бита разрешения) может генерировать прерывание, когда достигается временная метка n, в зависимости от флагов RECF и TRAF (они автоматически удаляются, когда достигается временная метка). В результате, это прерывание означает, что сообщение было отправлено/получено во временном окне между временными метками n-1 и n.

в) Управление приемом входных данных - действующее управление приемом информации (CHEN, RCEMSGNR) контролирует хранение сообщения во временном окне между временными метками n-1 и n.

г) Арбитраж входных данных - действующий арбитраж, информация (ARBM) определяет режим временного окна, начиная с временной метки n.

д) Управление передачей входных данных - действующее управление приемом информации (TPEH, ALTMSG, TCEMSGNR) определяет сообщение, которое будет передано после временной метки n. Начало передачи возможно в то время, как разрешено окно передачи для исключительного окна или втечение короткого (одного) арбитражного окна или в конце объединенного арбитражного окна. Для объединенного (длинного) арбитражного окна передача может начаться в полном временном окне.

е) Основное сообщение входных данных - не используется в этом временном окне.

ж) Последний основной цикл входных данных - не используется в этом временном окне.

Значение TMV, собранное после временной метки n-1, становится новым сравниваемым значением для временной метки n.

Значение CYCTMR.CSM равно номеру временной метки достигнутой последней временной метки, (CSM = n после достигнутой временной метки n). Значение CYCTMR.BCC является номером текущего основного цикла. Эта информация необходима, чтобы правильно настроить планировщик входных данных (битовые поля CYCLE, MCYCLE) и области сообщений (битовые поля CYCLE, MCYCLE, COLUMN, MCOLUMN).

Инструкции при окончании основного цикла

Сбор инструкций для последнего временного окна основного цикла начинается прежде, чем будет достигнута временная метка, и останавливается, когда TME2 входных данных найдено. Обработка входных данных, собранных втечение последнего временного окна основного цикла, выглядит следующим образом:

1. Временная метка входных данных - после прочтения действующего последнего основного цикла входных данных читается первая временная метка TME1 и ее определенное значение сравнивается со значением первой временной метки (первый временная метка после основного сообщения = начало нового основного цикла).

2. Прерывание входных данных – действующее прерывание информации (INP + 4 бита разрешения), собранной в ходе последнего временного окна основного цикла может генерировать прерывание, когда достигается временная метка 1 в зависимости от флагов RECF и TRAF (они автоматически удаляются, когда достигается временная метка). Эти флаги не применяются к основному сообщению (передача основного сообщения не изменяет эти флаги). В результате это прерывание означает, что сообщение было отправлено/получено в последнем временном окне основного цикла. ICE для последнего временного окна основного цикла должно быть записано в память планировщика после TME1 входных данных.

3. Управление приемом входных данных - действующее управление приемом информации (CHEN, RCEMSGNR), собранной во время последнего временного окна основного цикла, управляет хранением сообщения в последнем временном окне (не применяется для основного сообщения). RCE для последнего временного окна основного цикла должно быть записано в память планировщика после TME1 входных данных.

4. Арбитраж входных данных - действующая арбитражная информация (ARBM) определяет режим временного окна, начиная с первой временной метки. Эти входные данные должны быть записаны в память планировщика после TME1 входных данных.

5. Управление передачей входных данных - действующее управление передачи информации (TPEH, ALTMSG, TCEMSGNR) определяет сообщение, которое будет передано после первой временной метки. Начало передачи возможно в то время, когда активно разрешение окна передачи для исключительного окна или в течение короткого (одного) арбитражного окна. Для объединенного (длинного) ар-

битражного окна передача может начаться в полном временном окне. Эти входные данные должны быть записаны в память планировщика после TME1 входных данных.

6. Основное сообщение входных данных - значение временной метки действующей RME является сравнивающим значением для основной временной метки. Когда эта основная временная метка плюс основное смещение запуска будет достигнута, будет отправлено основное сообщение наружу (в зависимости от режима системного пропуска). Эти входные данные должны быть записаны в память планировщика перед действительными BCE входными данными.

7. Последний основной цикл входных данных - временной метки действующего BCE - является сравнивающим значением для сторожевого триггера по событию. При достижении этого значения времени сообщение из сторожевого триггера по событию может быть сгенерировано (в зависимости от режима системного пропуска). Действующий BCE всегда заканчивает входные данные планировщика для каждого TTCAN узла.

Перед основным сообщением можно принять действительное:

- Значение CYCTMR.CSM равно номеру достигнутой последней временной метки, (CSM = n после достигнутой временной метки n).

- Значение CYCTMR.BCC является номером текущего основного цикла.

Эта информация необходима, чтобы правильно настроить планировщик входных данных RCE и ICE (битовые поля CYCLE, MCYCLE).

После правильно полученного основного сообщения, но временная метка 1 еще не достигнута:

- Значение CYCTMR.CSM = 0.

- Значение CYCTMR.BCC - это номер нового основного цикла.

Эта информация необходима, чтобы правильно настроить планировщик входных сообщений ARBE и TCE (битовые поля CYCLE, MCYCLE) и области сообщений (битовые поля CYCLE, MCYCLE, COLUMN, MCOLUMN).

В результате ICE и RCE, описывающие прием/передачу сообщения в последнем временном окне основного цикла, приводят к действиям (например, прерывания), когда достигается временная метка 1.

16.13.5 Последовательность инструкций планировщика

Текущий основной цикл (BCC) и столбец системной матрицы (CSM)

Чтобы правильно настроить инструкции планировщика, значения CSM и BCC в регистре CYCTMR должны тщательно соблюдаться, особенно для последнего временного окна основного цикла.

При выходе из режима настройки, BCC и CSM равны 0 и планировщик стартует с первой временной метки. Значения BCC и CSM на рисунке 342 представляют внутренние значения, которые обновляются после корректной передачи основного сообщения.

Запущенное сообщение во время последней временной метки в основном цикле является основной меткой (см. описание RME). На рисунке 342 это соответствует сообщению после временного окна n. Основная временная метка следует

после временного окна n и начинается основное сообщение в новом основном цикле.

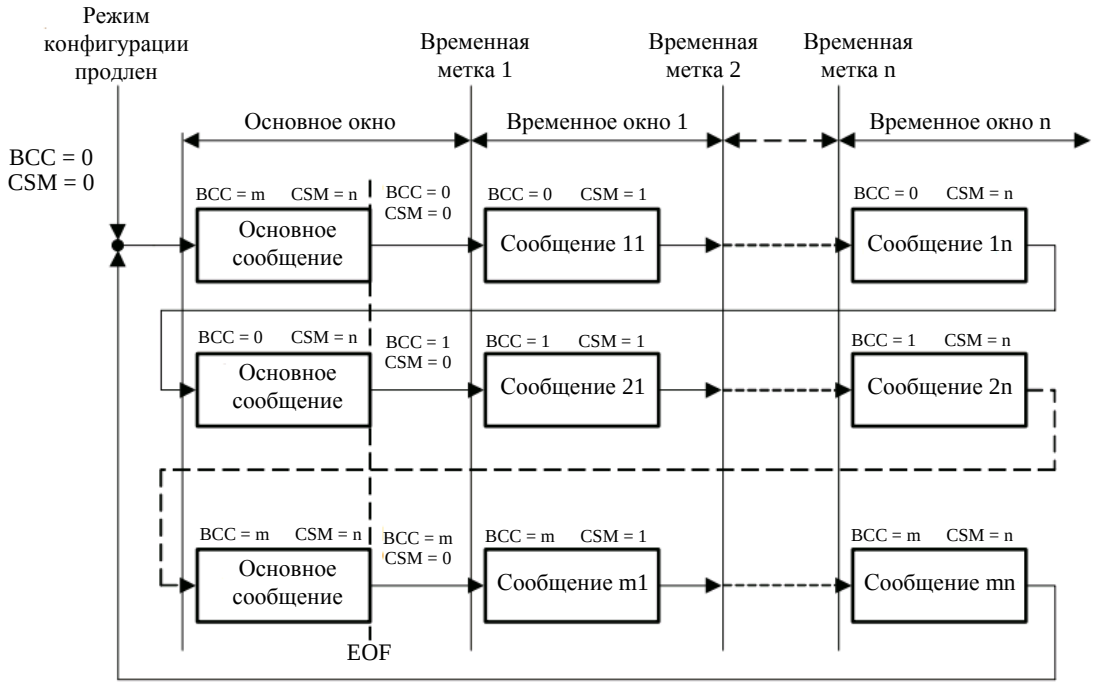


Рисунок 342 - Текущий основной цикл (BCC) и столбец системной матрицы (CSM) в матричном цикле

Время цикла каждого основного цикла начинается с 0 (виртуальное значение). Начало нового основного цикла может быть обнаружено только после корректного приема основного сообщения. В результате, значение времени цикла можно рассматривать как значение после того, как основное сообщение было передано правильно.

Общие правила последовательности инструкций.

Настройки уже прочитанных инструкций планировщика могут быть перезаписаны следующими действительными инструкциями планировщика. Следующий порядок типов планировщика входных данных должен соблюдаться, чтобы правильно настроить планировщик (начиная с первого):

1. TME, затем RCE или ICE, или TCE, или ARBE;
2. RME, затем BCE (для ведущего (Master) по времени);
3. BCE (для ведомых (Slave) устройств).

Планировщик входных данных всегда должен быть отключен, когда BCE и GM = 0.

Для ведущих по времени, следующая последовательность может быть установлена, чтобы отключить планировщик входных данных:

RME (GM=1), затем BCE (GM=1), затем RME (GM=0), затем BCE (GM=0) (пункты RME (GM = 0) и BCE (GM = 0) являются обязательными).

Если системный пропуск, первые RME и BCE входные данные (как с GM = 1) не принимаются во внимание. С помощью этих входных данных, стандартные (не пропущенные) значения синхронизации могут быть скорректированы. Второй RME (с GM = 0) может быть настроен на передачу вынужденного основного сообщения, когда есть системный пропуск, и процесс синхронизации занимает слишком много времени. Планировщик входных данных затем снова отключается с BCE с GM = 0 (например, с большим значением синхронизации в качестве критерия паузы для отсутствующего процесса синхронизации).

Пример планировщика последовательностей

Приведенные значения для CSM и BCC (оба битовых поля находятся в регистре CYCTMR) приводят к действующему планировщику входных данных RCE, TBK, ICE и ARBE (битовые поля CYCLE, MCYCLE) и управлением передачей областей сообщений (битовые поля CYCLE, MCYCLE , COLUMN, MCOLUMN). В случае запрограммированных значения для CYCLE, MCYCLE и так далее, не соответствующие заданным значениям CSM и BCC, соответствующие входные данные считаются недействительными и их информация не принимается во внимание для соответствующего временного окна. В следующем примере показана последовательность команд планировщика для основного цикла m с n временной меткой входных данных:

- TME1;
- RCE, ICE (CSM = n, BCC = m-1 или CSM = BCC = 0 когда продлен режим конфигурации);
- TCE, ARBE (CSM = 0, BCC = m);
- TME2;
- RCE, ICE (CSM = 1, BCC = m);
- TCE, ARBE (CSM = 0, BCC = m);
- TME3;
- RCE, ICE (CSM = 2, BCC = m);
- TCE, ARBE (CSM = 2, BCC = m);
- другие временные метки и инструкции;
- TMEn;
- RCE, ICE (CSM = n-1, BCC = m);
- TCE, ARBE (CSM = n-1, BCC = m);
- RME (GM = 1, только для ведущих (Masters) по времени);
- BCE (GM = 1, для ведущих (Masters) по времени и ведомых (Slaves));
- RME (GM = 0, только для ведущих (Masters) по времени);
- BCE (GM = 0, для ведущих (Masters) по времени и ведомых (Slaves));

16.14 Операции TTCAN

16.14.1 Конфигурация

После операции сброса должно быть настроено TTCAN расширение. Режим конфигурации вводится автоматически после сброса или должен быть инициирован программно, записав TTFMR.CFGMEL = 01_2. Режим конфигурации может быть продлен только программно, записав TTFMR.CFGMEL = 10_2. Флаг состояния

TTSR.CFGM указывает, является или нет режим конфигурации активным. TTSR.CFGM автоматически устанавливается, когда соответствующий CAN узел отключает тактирование (отключить запрос).

В режиме конфигурации TTCAN узла (CAN узел 0, если сконфигурирован для TTCAN), следующие действия должны быть выполнены, как указано, программно или аппаратно для всех узлов:

1. Локальное время, глобальное время и время цикла устанавливается в 0 (аппаратно).

2. Передача или прием сообщений из TTCAN узла невозможны, потому что результат фильтрации приема не разрешен (аппаратно).

3. Соответствующее значение TUR должно быть записано в битовое поле TURR.TURADJ (программно). Это значение автоматически передается в TURR.TUR (аппаратно).

4. Планировщик памяти входных данных должен быть инициализирован (программно).

5. Управление информацией TTCAN (например, ID основного сообщения) и самого TTCAN узла должно быть полностью установлено и включено для передачи CAN сообщений (программно).

6. После завершения конфигурации (программно), TTFMR.CFGMEL должен быть установлен в 10_2 (программно).

7. Локальное время начинается после выхода из режима конфигурации (аппаратно).

8. Фаза синхронизации вводится автоматически, когда продлен режим конфигурации (аппаратно). Это показывают битовые поля TTSR.SYNCS = 01_2 (аппаратно).

9. Для ведущего (Master) по времени планируется передача основных сообщений, в то время как TTCAN узел в режиме пропуска находится в состоянии "синхронизации". Планировщик входных данных RME с GM = 1 учитывается только тогда, когда TTCAN узел в состоянии "планирование".

10. Для ведущих (Master) и ведомых (Slave) по времени планировщик входных данных BCE с GM = 1 учитывается только тогда, когда TTCAN узел в состоянии "планирование".

Ошибка конфигурации

Во время действий планировщика некоторые условия приводят к ошибке конфигурации:

1. В конце основного цикла объединённое арбитражное окно остается открытым.

2. Найдено RME входных данных для ведомых устройств.

3. Ни одна временная метка не найдена во время сбора инструкций.

4. RCE, ICE, TCE или ARBE были найдены, прежде чем TME1 (после начала или в конце основного цикла).

5. Другие входные данные (кроме EOS), прежде чем RME или BCE были найдены после первой RME.

6. Основная область сообщения не действительна, когда требуется для передачи.

16.14.2 Фаза синхронизации

При запуске системы TTCAN (например, после сброса), вся система должна быть синхронизирована после завершения фазы конфигурации. Во время фазы синхронизации (обозначены в битовом поле TTSR.SYNCS = 01_2) должны быть приняты следующие действия (или принимаются автоматически):

1. Init_Watch_Trigger учитывается до тех пор, пока первое сообщение было правильно принято или передано. Обнаруживается событие Init_Watch_Trigger, когда время цикла достигает значения 2^{16}-1. Это событие свидетельствует IWTE = 1 и приводит только к прерыванию.

2. Значение сторожевого триггера задает последний основной цикл входных данных, учитывая только после первого правильного переданного сообщения на шину. Когда происходит событие на сторожевом триггере после правильной передачи сообщения на шину, передача и прием фреймов данных или удаленных фреймов отключены (установлен CAN узел в режим конфигурации).

3. Во время фазы синхронизации планировщик входных данных RME или с GM = 1 не учитывается.

16.14.3 Ведущий по времени

Состояние ведущего по времени

Потенциальный ведущий (Master) по времени - это устройство, которое может передавать основное сообщение. Резервный ведущий (Master) по времени - это потенциальный ведущий (Master) по времени для приема основного сообщения, если нет основного ведущего (Master) по времени. Текущий ведущий (Master) по времени - это устройство, которое успешно отправило свое опорное сообщение. Существует только один текущий ведущий (Master) по времени в системе TTCAN.

Если текущий ведущий (Master) по времени получает основное сообщение, которое не свое, то он становится резервным ведущим (Master) по времени. Если резервный ведущий (Master) по времени успешно передал свое основное сообщение, оно становится новым текущим ведущим (Master) по времени.

При обнаруженном конфликте приоритета между TTCAN узлом и фактическим ведущим (Master) по времени значение RTO автоматически изменяется (декремент на 1). Это означает, что приоритет фактического ведущий (Master) по времени (задается битами ID[2:0]) последнего принятого основного сообщения ниже запрограммированного значения TMPRIO.

Режим строгого запуска по времени

Режим строгого запуска по времени (пропуски не допускаются) может быть достигнут, если используются только планировщик входных данных RME и BCE с GM = 0. В этом случае условие пропуска "игнорируется" для передачи основного сообщения. Чтобы избежать непреднамеренных передач основных сообщений, в этом случае должна быть выключена внешняя генерация запуска.

16.14.4 Обработка ошибок

Обработка ошибок TTCAN различает четыре степени ошибок:

1. S0 - Нет ошибки: Нормальная функциональность;

2. S1 - Внимание: только оповещение об использовании прерывания (ERRS1).

Источник: Ошибка 1 Планировщика (MSCmax – MSCmin > 2 или, когда MSC области приема равно 7), Tx_Underflow;

3. S2 - Ошибка: Оповещение об использовании прерывания (ERRS2). Все передачи будут отключены (за исключением основных сообщений), TTCFGR.RTO установлен в 127.

Источник: Ошибка 1 Планировщика (MSC области приема равно 7), Tx_overflow.

Следствие: Не будет открыто разрешенное окно передачи (для исключительных и арбитражных окон).

4. S3 - Серьезная ошибка: Оповещение об использовании прерывания (ERRS3). Все действия CAN на шине останавливаются (второстепенные значения передаются по шине). Стадия конфигурации вводится автоматически (установлен CFGM).

Источник: Примененный сторожевой таймер, отключенная шина, ошибка конфигурации, событие сторожевого таймера.

Следствие: Будет установлен бит INIT в CAN узле (останавливаются действия CAN узла на шине).

Примечание - Любое изменение состояния ошибки может генерировать прерывание.

16.14.5 Приложение сторожевого таймера

Приложение сторожевого таймера обеспечивает возможность проверки для TTCAN модуля, если основная система по-прежнему работает. Приложение сторожевого таймера считает с шагом 256 NTU.

Каждый раз, когда прошло 256 NTU, значение AWDR.AWDV уменьшается на единицу. Если достигается значение 0 после декремента, ошибка S3 сигнализирует битом AWDERR. Спадающий фронт бита LTR.LT.7 (переход от 1 до 0) указывает, что время 256 NTU прошло и значение приложения сторожевого таймера уменьшается. Приложение сторожевого таймера обслуживается программно, записав новое значение в битовое поле AWDR.AWDE.

16.14.6 Обращение к MSC

Битовое поле MSC обновляется только для тех областей сообщений, в которых будут указанны действительные RCE или TCE входных данных, в то время как установлен соответствующий бит MSGVAL. Номер сообщения учитывается для определенного обновления в соответствующих битовых полях RCEMSGNR или TCEMSGNR.

Проверка на состояние приема/передачи происходит, когда достигается конец разрешенного окна передачи следующих временных меток. Поскольку длина разрешенного окна передачи известна после каждой временной метки (даже если никакое сообщение не может быть отправлено на выход), этот момент времени может быть использован для проверки приема и передачи. Эта особенность дает

возможность иметь положительный результат проверки, даже если сообщение в предыдущем временном окне заканчивается в течение разрешенного окна передачи.

Попытка передачи определяется как не удавшаяся, если действительная область сообщения для передачи не может быть передана на шину CAN. Запуск передачи, должный быть активным без действительной области сообщения, при обнаружении не рассматривается как попытка передачи.

16.14.7 Управление прерываниями TTCAN

Рисунки 343 и 344 показывают конфигурации трех прерываний TTCAN.

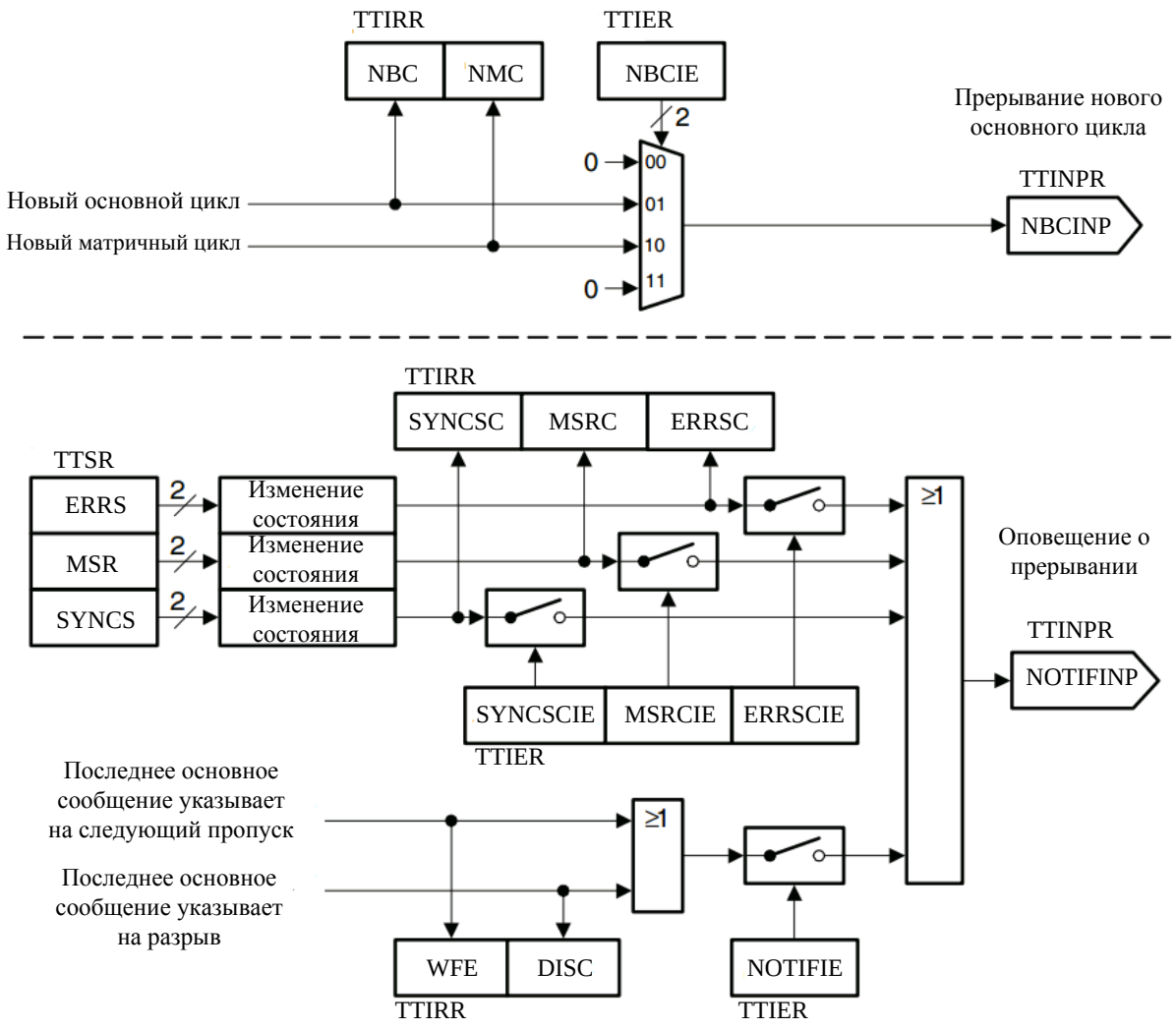


Рисунок 343 – Структура генерации нового основного цикла и прерывания оповещения

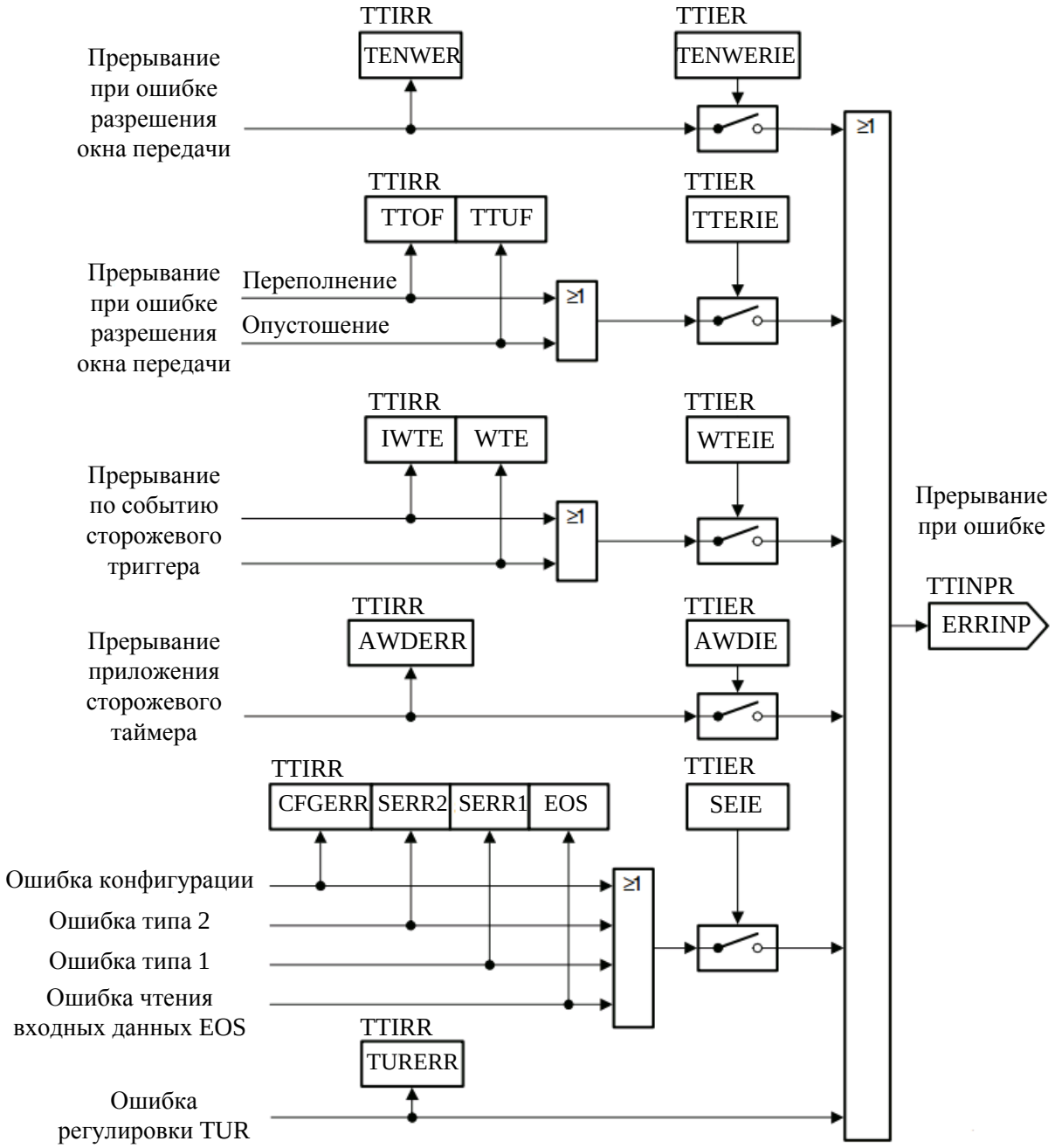


Рисунок 344 – Структура генерации прерывания при ошибке

.

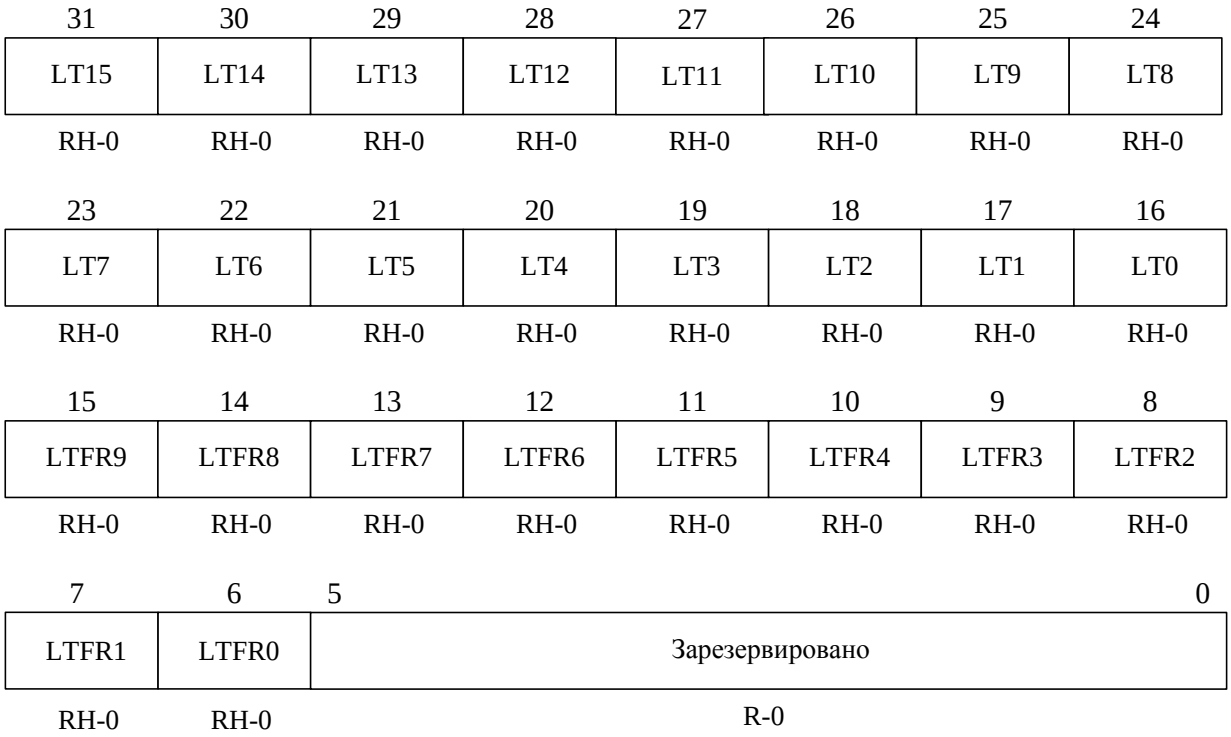
16.15 Регистры TTCAN

В таблицу 153 сведены все программируемые регистры TTCAN.

Таблица 153 - Программируемые регистры TTCAN

Адрес	Регистр	Имя регистра
0A80h	LTR	Регистр локального времени
0A84h	SYNMR	Регистр метки синхронизации
0A88h	REFMR	Регистр основной метки
0A8Ch	LREFMR	Регистр последней основной метки
0A90h	TURR	Регистр коэффициента единицы времени
0A94h	CYCTMR	Регистр времени цикла
0A98h	LOR	Регистр локального смещения
0A9Ch	GMR	Регистр глобальной метки
0AA0h	LGMR	Регистр последней глобальной метки
0AA4h	AWDR	Регистр использования сторожевого таймера
0AC0h	TTCR	Регистр управления, срабатывающий по времени
0AC4h	TTCFGR	Регистр конфигурации, срабатывающий по времени
0AC8h	TTSR	Регистр состояния, срабатывающий по времени
0ACCh	TTFMR	Регистр флага модификации, срабатывающий по времени
0AD0h	TTIRR	Регистр запроса на прерывание, срабатывающий по времени
0AD4h	TTIER	Регистр разрешения прерывания, срабатывающий по времени
0AD8h	TTINPR	Регистр прерывания указателя узла, срабатывающий по времени
0AF0h	STSRL	Регистр (младший) состояния планировщика времени
0AF4h	STSRH	Регистр (старший) состояния планировщика времени
0AF8h	SISR	Регистр состояния инструкций планировщика
47FCh	STPTR0	Регистр планировщика стартового указателя узла 0

Описание бит регистра локального времени LTR представлено на рисунке 345.



R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 345 – Регистр локального времени модуля CAN (LTR) – адрес 0A80h

Биты 31-16 LT15 – LT0. Локальное время. Содержит целую часть счетчика NTU. В TTCAN уровне 1, LT увеличивается с каждым CAN бит времени. В TTCAN уровне 2, LT увеличивается каждый раз, когда переполняется дробная часть LTFR.

Биты 15-6 LTFR9 – LTRFR0. Часть локального времени. Содержит дробную часть счетчика NTU (только TTCAN уровень 2). В случае переполнения после добавления значения TUR, LT увеличивается на 1.

Биты 5-0 Зарезервировано. Читаются как «0». При записи следует писать «0».

Описание бит регистра метки синхронизации SYNMR представлено на рисунке 346.

31	30	29	28	27	26	25	24
SYNM15	SYNM14	SYNM13	SYNM12	SYNM11	SYNM10	SYNM9	SYNM8
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
23	22	21	20	19	18	17	16
SYNM7	SYNM6	SYNM5	SYNM4	SYNM3	SYNM2	SYNM1	SYNM0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
15	14	13	12	11	10	9	8 0
SYNMFR6	SYNMFR5	SYNMFR4	SYNMFR3	SYNMFR2	SYNMFR1	SYNMFR0	Зарезервировано
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	R-0

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 346 – Регистр метки синхронизации модуля CAN (SYNMR) – адрес 0A84h

Биты 31-16 SYNM15 – SYNM0. Метка синхронизации. Содержит целую часть метки синхронизации. SYNM является битовом полем LTR.LT с захваченным импульсом синхронизации фреймов.

Биты 15-9 SYNMFR6 – SYNMFR0. Часть метки синхронизации. Содержит дробную часть метки синхронизации. SYNMFR является битовым полем LTR.LTFR с захваченным импульсом синхронизации фреймов.

Биты 8-0 Зарезервировано. Читаются как «0». При записи следует писать «0».

Примечание - Если CAN узел является ведущим (Master) по времени, содержимое регистра SYNMR используется в качестве синхронизирующих данных, передаваемых в основном сообщении.

Описание бит регистра основной метки REFMR представлено на рисунке 347.

31	30	29	28	27	26	25	24
REFM15	REFM14	REFM13	REFM12	REFM11	REFM10	REFM9	REFM8
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
23	22	21	20	19	18	17	16
REFM7	REFM6	REFM5	REFM4	REFM3	REFM2	REFM1	REFM0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
15	14	13	12	11	10	9	8 0
REFMFR6	REFMFR5	REFMFR4	REFMFR3	REFMFR2	REFMFR1	REFMFR0	Зарезерви-ровано
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	R-0

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 347 – Регистр основной метки модуля CAN (REFMR) – адрес 0A88h

Биты 31-16 REFM15 – REFM0. Основная метка. Содержит целую часть основной метки. REFM является битовым полем SYNMR.SYNM с захваченной правильной концовкой основного сообщения.

Биты 15-9 REFMFR6 – REFMFR0. Часть основной метки. Содержит дробную часть основной метки. REFMFR является битовом полем SYNMR.SYNMFR с захваченной правильной концовкой основного сообщения.

Биты 8-0 Зарезервировано. Читаются как «0». При записи следует писать «0».

Описание бит регистра последней основной метки LREFMR представлено на рисунке 348.

31	30	29	28	27	26	25	24
LREFM15	LREFM14	LREFM13	LREFM12	LREFM11	LREFM10	LREFM9	LREFM8
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
23	22	21	20	19	18	17	16
LREFM7	LREFM6	LREFM5	LREFM4	LREFM3	LREFM2	LREFM1	LREFM0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
15	14	13	12	11	10	9	8 0
LREFMFR6	LREFMFR5	LREFMFR4	LREFMFR3	LREFMFR2	LREFMFR1	LREFMFR0	Зарезервировано
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	R-0

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 348 – Регистр последней основной метки модуля CAN (LREFMR) – адрес 0A8Ch

Биты 31-16 LREFM15 – LREFM0. Последняя основная метка. Содержит целую часть основной метки. LREFM является битовом полем REFMR.REFM с захваченной правильной концовкой основного сообщения.

Биты 15-9 LREFMFR6 – LREFMFR0. Часть последней основной метки. Содержит дробную часть последней основной метки. LREFMR является битовом полем REFMR.REFMFR с захваченной правильной концовкой основного сообщения.

Биты 8-0 Зарезервировано. Читаются как «0». При записи следует писать «0».

Описание бит регистра коэффициента единицы времени TURR представлено на рисунке 349.

31	30	29	28	27	26	25	24
TUR9	TUR8	TUR7	TUR6	TUR5	TUR4	TUR3	TUR2
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
23	22	21	20	19	18	17	16
TUR1	TUR0	Зарезерви-ровано	LTDIV2	LTDIV1	LTDIV0	LTCS	VAL
RH-0	RH-0	R-0	RW-0	RW-0	RW-0	RW-0	RH-0
15	14	13	12	11	10	9	8
TURADJ9	TURADJ8	TURADJ7	TURADJ6	TURADJ5	TURADJ4	TURADJ3	TURADJ2
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0
7	6	5				1	0
TURADJ1	TURADJ0	Зарезервировано					ADJEN
RWH-0	RWH-0	R-0					RWH-0

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 349 – Регистр коэффициента единицы времени модуля CAN (TURR) – адрес 0A90h

Биты 31-22 TUR9 – TUR0. Содержит в данный момент значение отношения активной единицы времени измерения.

Бит 21 Зарезервировано. Читаются как «0». При записи следует писать «0».

Биты 20-18 LTDIV2 – LTDIV0. Делитель локального времени. Определяет коэффициент делителя для генерации локального времени (если f_{CAN}, то выбирается LTCS = 1). Коэффициент делителя задается 2^{LTDIV}:

000_2 = Коэффициент делителя равен 1
001_2 = Коэффициент делителя равен 2
010_2 = Коэффициент делителя равен 4
011_2 = Коэффициент делителя равен 8
100_2 = Коэффициент делителя равен 16
101_2 = Коэффициент делителя равен 32
110_2 = Коэффициент делителя равен 64
111_2 = Коэффициент делителя равен 128

Бит 17 LTCS. Источник тактирования локального времени. Определяет источник синхронизации для генерации локального времени (добавление TUP к LT, LTFR) для TTCAN уровня 2:
0 = Новое значение локального времени генерируется с каждым квантом времени t_q, соответствующий CAN узел (в зависимости от CAN битового времени).
1 = Генерация локального времени основывается на тактах CAN модуля f_{CAN}.
Скорость обновления t_{upd} основана на коэффициенте делителя заданного битового поля LTDIV в соответствии с $t_{upd} = 2^{LTDIV} / f_{CAN}$.

Бит 16 Valid. Подтверждение. Указывает, что новое значение TURADJ доступно. VAL устанавливается, когда CAN аппаратно завершил свой автоматический расчет (с правильным результатом без переполнения или потери значимости) или когда программно пишет TURADJ. Автоматический расчет не состоится, когда VAL = 1. VAL очищается, когда значение TURADJ копируется в TUR:
0 = Значение TURADJ не было обновлено и не происходит автоматическое обновление TUR значением TURADJ.
1 = Значение TURADJ было обновлено программно или аппаратно и оно будет скопировано в TUR в начале следующего основного цикла.

Биты 15-6 TURADJ9 – TURADJ0. Регулировка коэффициента единицы измерения. Удерживает значение коэффициента единицы измерения, которое будет использоваться на следующий основной цикл. В случае автоматического расчета коэффициента единицы измерения с опустошением (переполнением), TURADJ записывает с 0c (1c), ADJEN очищается и генерируется прерывание ошибки регулировки коэффициента единицы измерения. Аппаратное расчетное значение TURADJ может быть перезаписано программно.

Биты 5-1 Зарезервировано. Читаются как «0». При записи следует писать «0».

Бит 0 ADJEN. Разрешение регулировки. Включает автоматическое вычисление нового значения TURADJ, когда было получено новое основное сообщение (не для текущего ведущего (Master) по времени). При генерации прерывания ошибки регулировки коэффициента единицы измерения бит ADJEN будет автоматически очищаться, когда встречается переполнение или опустошение автоматически рассчитанного значения:
0 = Автоматический расчет TUR запрещен. Новое значение времени для коэффициента единицы измерения для TURADJ должно быть рассчитано и записано программно.
1 = Автоматический расчет TUR разрешен. После получения основного сообщения, новое значение для TURADJ рассчитывается аппаратно. Рассчитанное значение проверяется на переполнение или

опустошение.

Описание бит регистра времени цикла CYCTMR представлено на рисунке 350.

31 30	29	28	27	26	25	24
Зарезервировано	CSM5	CSM4	CSM3	CSM2	CSM1	CSM0
R-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

23 22	21	20	19	18	17	16
Зарезервировано	BCC5	BCC4	BCC3	BCC2	BCC1	BCC0
R-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

15	14	13	12	11	10	9	8
CYCTM15	CYCTM14	CYCTM13	CYCTM12	CYCTM11	CYCTM10	CYCTM9	CYCTM8
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

7	6	5	4	3	2	1	0
SYNMFR7	SYNMFR6	SYNMFR5	SYNMFR4	SYNMFR3	SYNMFR2	SYNMFR1	CYCTM0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 350 – Регистр времени цикла модуля CAN (CYCTMR) – адрес 0A94h

Биты 31-30 Зарезервировано. Читаются как «0». При записи следует писать «0».

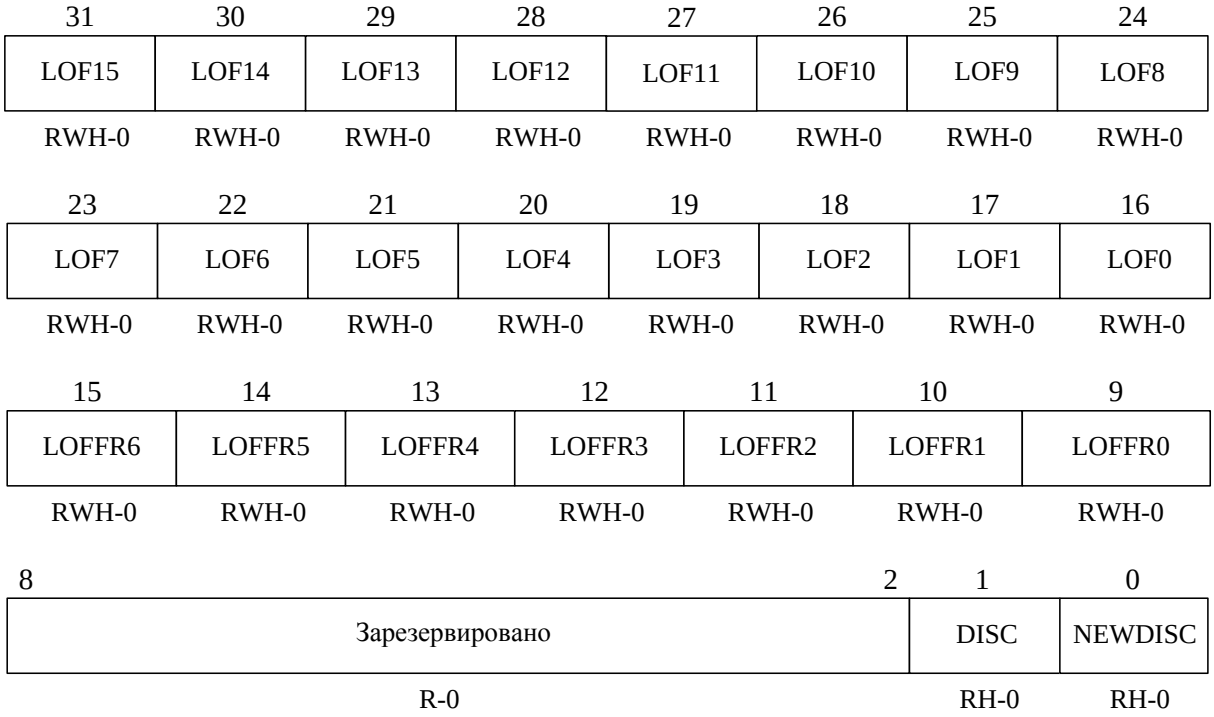
Биты 29-24 CSM5 – CSM0. Столбец матричной системы. Показывает номер текущего столбца в системной матрице. CSM увеличивается, когда текущее время цикла становится равным сохраненной временной метки (с указанием начала следующего столбца).

Биты 23-22 Зарезервировано. Читаются как «0». При записи следует писать «0».

Биты 21-16 BCC6 – BCC0. Счетчик основных циклов. Показывает номер текущего основного цикла. BCC увеличивается после каждого правильно переданного основного сообщения.

Биты 15-0 CYCTM15 – CYCTM0. Время цикла. Показывает время, уже прошедшее в текущем основном цикле. CYCTM рассчитывается путем LTR.LT - REFMR.REFM. В случае включения отрицательного результата (переполнение LTR.LT), результат корректируется.

Описание бит регистра локального смещения LOR представлено на рисунке 351.



R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 351 – Регистр локального смещения модуля CAN (LOR) – адрес 0A98h

Биты 31-16 LOF15 – LOF0. Локальное смещение. Содержит целую часть локального смещения:
- ведущий (Master) по времени (передача основных сообщений): сумма локального смещения и локального времени хранится в глобальном времени в регистре GTR.
- не ведущий (Master) по времени (получение основных сообщений): Локальное смещение получается из информации глобального времени минус локальная основная метка (в REFMR).

Биты 15-9 LOFFR6 – LOFFR0. Часть локального смещения. Содержит дробную часть локального смещения.

Биты 8-2 Зарезервировано. Читаются как «0». При записи следует писать «0».

Бит 1 DISC. Бит разрыва. Содержит бит DISC основного сообщения, которое послано наружу (учитываются для ведущего (Master) по времени). Этот бит устанавливается, если бит NEWDISC равен «1», когда захвачено глобальное время (в SOF) для передачи основного сообщения. Он сбрасывается, когда основное сообщение было передано правильно:

0 = Бит DISC в основном сообщении текущего ведущего (Master) по времени «0».
1 = Бит DISC в основном сообщении текущего ведущего (Master) по времени «1».

Бит 0 NEWDISC. Бит нового разрыва. Указывает, что регистр LOR доступен для записи для ведущего по времени (и что содержание LOF или LOFFR можно было бы изменить). NEWDISC указывает, что следующее основное сообщение будет отправлено с разрывом в основной метке. NEWDISC устанавливается автоматически, когда обнаружена операция записи в LOR. NEWDISC очищается, когда захватывается локальное время (в SOF) для передачи основного сообщения:
0 = LOR не доступен для записи.
1 = LOR доступен для записи.

Примечание - Регистр LOR можно записать программно, только если CAN узел - текущий ведущий (Master) по времени или в режиме настройки; в противном случае, операция записи не учитывается, потому что записанное локальное смещение учитывается только для передачи основного сообщения, бит DISC устанавливается автоматически.

Описание бит регистра глобальной метки GMR представлено на рисунке 352.

31	30	29	28	27	26	25	24
GM15	GM14	GM13	GM12	GM11	GM10	GM9	GM8
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
23	22	21	20	19	18	17	16
GM7	GM6	GM5	GM4	GM3	GM2	GM1	GM0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
15	14	13	12	11	10	9	8 0
GMFR6	GMFR5	GMFR4	GMFR3	GMFR2	GMFR1	GMFR0	Зарезервировано
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	R-0

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 352 – Регистр глобальной метки модуля CAN (GMR) – адрес 0A9Ch

Биты 31-16 GM15 – GM0. Глобальная метка. Содержит целую часть глобальной метки:
- ведущий (Master) по времени (передача основных сообщений): сумма локального смещения и локального времени хранится в глобальном времени (Global_Sync_Mark) в регистре GTR в начале ос-

новного сообщения.
- не ведущий (Master) по времени (получение основных сообщений): Информация глобального времени в GMR, полученная Global_Ref_Mark в основном сообщении.

Биты 15-9 GMFR6 – GMFRFR0. Часть глобальной метки. Содержит дробную часть глобальной метки.

Биты 8-0 Зарезервировано. Читаются как «0». При записи следует писать «0».

Описание бит регистра последней глобальной метки LGMR представлено на рисунке 353.

31	30	29	28	27	26	25	24
LGM15	LGM14	LGM13	LGM12	LGM11	LGM10	LGM9	LGM8
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
23	22	21	20	19	18	17	16
LGM7	LGM6	LGM5	LGM4	LGM3	LGM2	LGM1	LGM0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
15	14	13	12	11	10	9	8 0
LGMFR6	LGMFR5	LGMFR4	LGMFR3	LGMFR2	LGMFR1	LGMFR0	Зарезервировано
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	R-0

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 353 – Регистр последней глобальной метки модуля CAN (LGMR) – адрес 0AA0h

Биты 31-16 LGM15 – LGM0. Последняя глобальная метка. Содержит значение GM последней основной метки.

Биты 15-9 LGMFR6 – LGMFRFR0. Часть последней глобальной метки. Содержит значение GMFR последней основной метки.

Биты 8-0 Зарезервировано. Читаются как «0». При записи следует писать «0».

Примечание - Разница между действительной глобальной меткой и последней может быть использована для определения величины, необходимой для обновления TURR.TUR (не для действующего ведущего (Master) по времени).

Описание бит регистра использования сторожевого таймера AWDR представлено на рисунке 354.

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

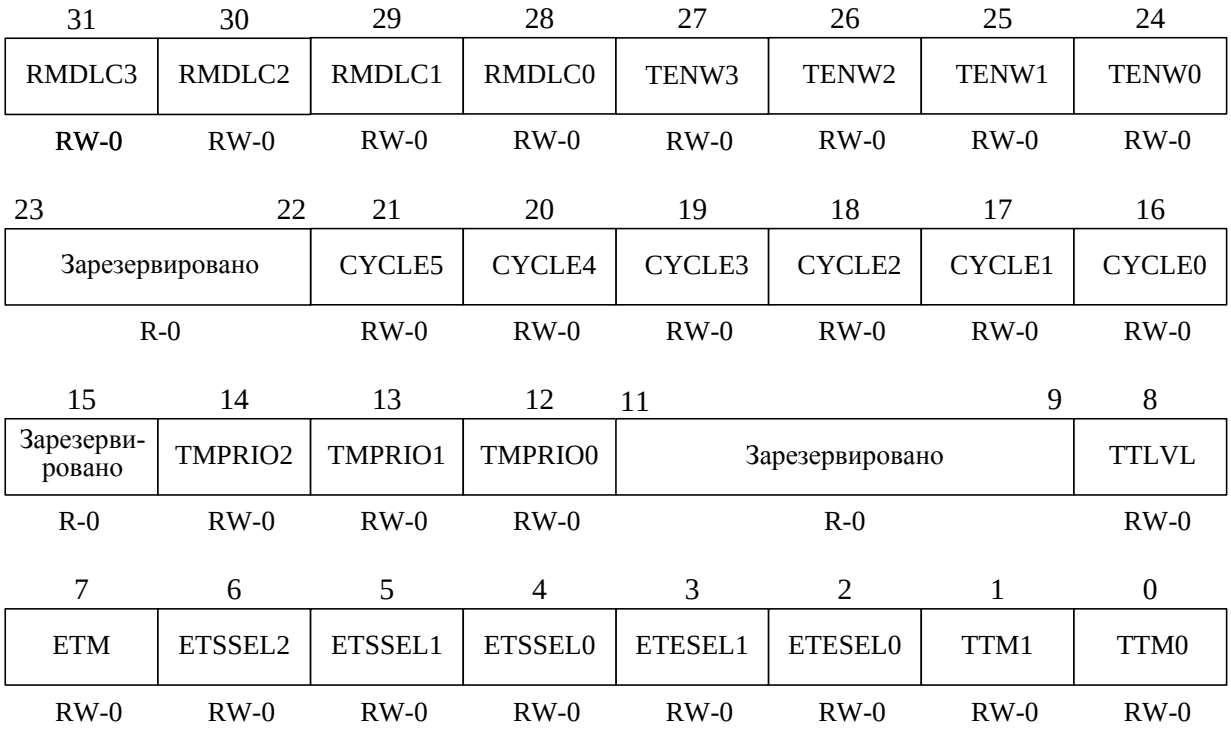
Рисунок 354 – Регистр использования сторожевого таймера модуля CAN (AWDR) – адрес 0AA4h

Биты 31- 8 Зарезервировано. Читаются как «0». При записи следует писать «0».

Биты 7-0 AWDV6 – AWDV0. Применение сторожевого таймера. Содержит текущее значение приложения сторожевого таймера. Падающий фронт бита LTR.LT[7] (перехода из «1» в «0») указывает, что время 256 NTU истекло, и приложение сторожевого таймера автоматически уменьшается на 1. Если значение «0» становится после уменьшения, ошибку S3 сигнализирует бит TTIRR.AWDERR. AWDV не уменьшается ниже 0. Приложение сторожевого таймера обслуживается программно, записав новое значение битового поля AWDV. В случае коллизии программной записи и автоматического декремента, учитывается запись. В связи с тем, что счетчик подсчета NTU (регистр LTR) не сбрасывается, когда обслуживается приложение сторожевого таймера, общее время между записью в AWDV и сигнализации ошибки S3 имеет неопределенность один шаг AWDV.

При применении сторожевого таймера происходит событие, когда значение AWDV уменьшается и достигает нуля. Когда программа пишет 00_{16} в AWDV, приложение сторожевого таймера отключается без генерации события при применении сторожевого таймера.

Описание бит регистра управления, срабатывающего по времени, TTCR представлено на рисунке 355.



31	30	29	28	27	26	25	24
RMDLC3	RMDLC2	RMDLC1	RMDLC0	TENW3	TENW2	TENW1	TENW0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

23 22	21	20	19	18	17	16
Зарезервировано	CYCLE5	CYCLE4	CYCLE3	CYCLE2	CYCLE1	CYCLE0
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

15	14	13	12	11 9	8
Зарезерви-ровано	TMPRIO2	TMPRIO1	TMPRIO0	Зарезервировано	TTLVL
R-0	RW-0	RW-0	RW-0	R-0	RW-0

7	6	5	4	3	2	1	0
ETM	ETSSEL2	ETSSEL1	ETSSEL0	ETESEL1	ETESEL0	TTM1	TTM0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 355 – Регистр управления, срабатывающий по времени, модуля CAN (TTCR) – адрес 0AC0h

Биты 31- 28 RMDLC3 – RMDLC0. Длина кода данных основного сообщения. Определяет длину кода данных DLC в основном сообщении, посланное наружу из CAN узла, если он ведущий (Master) по времени. Битовое поле DLC в основной области сообщения содержит DLC из ранее полученного основного сообщения. Обратите внимание, что RMDLC должен быть запрограммирован только значений от 1 до 8 для TTCAN уровня 1 и значений от 4 до 8 для TTCAN уровня 2. Другие значения не должны быть запрограммированы.

Биты 27-24 TENW3 – TENW0. Разрешить окно передачи. Определяет, сколько может CAN бит времени пройти, прежде чем отложенный запуск передачи сбрасывается:
0000_2 = Может пройти один CAN бит времени;
0001_2 = Может пройти два CAN бит времени;
. . . . .
1110_2 = Может пройти пятнадцать CAN бит времени;
1111_2 = Может пройти шестнадцать CAN бит времени;

Биты 23-22 Зарезервировано. Читаются как «0». При записи следует писать «0».

Биты 21-16 CYCLE5 – CYCLE0. Номер основного цикла. Определяет номер последнего основного цикла в матричном цикле. Для ведущих (Master) по времени значение CYCLE по сравнению с текущим значением битового поля CYCTMR.BCC. Если при сравнении обнаруживается совпадение, основное сообщение посылается наружу с новым основным циклом счета, начиная с 0. Для ведомых (Slaves) устройств значение используется для обнаружения ожидаемого конца матричного цикла.

Бит 15 Зарезервировано. Читаются как «0». При записи следует писать «0».

Биты 14-12 TMPRIO3 – TMPRIO0. Приоритет ведущего по времени. Определяет приоритет действительного ведущего по времени. Это значение используется для передачи в бит идентификатора [2:0] основного сообщения. В том случае, когда TTCAN узел теряет арбитраж в пользу основной метки на шине или получает основную метку, в то время как основной запуск установлен, основной запуск сбрасывается, и принятый идентификатор сохраняется в основной области сообщения ID битного поля.

Биты 11-9 Зарезервировано. Читаются как «0». При записи следует писать «0».

Бит 8 TTLVL. Уровень срабатывания по времени. Определяет уровень функциональности TTCAN.
0 = Выбирается TTCAN уровень 1.
1 = Выбирается TTCAN уровень 2.

Бит 7 ETM. Режим внешнего запуска. Определяет режим внешнего запуска. Передача основного сообщения запускается, если выполняется выбранное условие запуска, когда шина находится в режиме ожидания в конце разрешенного окна передачи в последнем временном окне основного цикла. Событие запуска сохраняется в ETREV.
0 = Событие триггера само учитывается после окончания разрешенного окна передачи в последнем временном окне основного цикла.
1 = Событие запуска хранится в ETREV и учитывается после окончания разрешенного окна передачи в последнем временном окне основного цикла.

Биты 6-4 ETSSEL2 – ETSSEL0. Выбор источника внешнего запуска. Выбор входного источника для внешнего запуска основного сообщения.
000_2 = Выбор входной линии внешнего запуска ttc_ecctt_i[1];
001_2 = Выбор входной линии внешнего запуска ttc_ecctt_i[2];
. . . . .
110_2 = Выбор входной линии внешнего запуска ttc_ecctt_i[6];
111_2 = Выбор входной линии внешнего запуска ttc_ecctt_i[7];

Биты 3-2 ETESEL1, ETESEL0. Выбор внешнего запуска события. - Определяет источник внешнего запуска события, который может быть использован для запуска передачи основного сообщения. Передача основного сообщения будет срабатывать только тогда, когда обнаружено внешнее событие и установлен бит запроса внешнего запуска TTSR.ETR (= 1).
00 = Зарезервировано; внешний запуск не запускает передачу основного сообщения.
01 = Отрицательный фронт на входной линии внешнего запуска ttc_ecctt_i[n] (n = 1-7 выбирается ETSSEL) запускает передачу основного сообщения.
10 = Положительный фронт на входной линии внешнего запуска ttc_ecctt_i[n] (n = 1-7 выбирается ETSSEL) запускает передачу основного сообщения.
11 = Отрицательный или положительный фронт на входной линии внешнего запуска ttc_ecctt_i[n] (n = 1-7 выбирается ETSSEL) запускает передачу основного сообщения.

Биты 1-0 TTM1, TTM0. Режим срабатывания по времени. Определяет режим TTCAN узла относительно срабатывания по времени и функциональность текущего (Master) по времени.
00 = CAN узел отключен для работы TTCAN и работает в режиме запуска по событию (другие настройки для TTCAN неактивны). Основная область сообщения работает как приемник или передатчик, как и любая другая область сообщения.
01 = CAN узел включен для работы TTCAN как приемное устройство. Основные сообщения не могут быть переданы в CAN узел.
10 = CAN узел включен для работы TTCAN как действующий или возможный ведущий (Master) по времени. CAN узел может передавать основное сообщение.
11 = Зарезервировано. Комбинация не используется.

Описание бит регистра конфигурации, срабатывающего по времени, TTCFGR представлено на рисунке 356.

31	30	29	28	27	26	25	24
TTCNT7	TTCNT6	TTCNT5	TTCNT4	TTCNT3	TTCNT2	TTCNT1	TTCNT0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
23	22	21	20	19	18	17	16
RTO7	RTO6	RTO5	RTO4	RTO3	RTO2	RTO1	RTO0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
15	14	13	12	11	10	9	8
EXPTT7	EXPTT6	EXPTT5	EXPTT4	EXPTT3	EXPTT2	EXPTT1	EXPTT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
Зарезерви-ровано	IRO6	IRO5	IRO4	IRO3	IRO2	IRO1	IRO0
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 356 – Регистр конфигурации, срабатывающий по времени, модуля CAN (TTCFGR) – адрес 0AC4h

Биты 31- 24 TTCNT7 – TTCNT0. Счетчик запуска передачи. Увеличивается на 1 каждый раз, когда запрашивается запуск передачи для TTCAN узла. При достижении значения EXPTT, больше не увеличивается, и будущие запросы на передачу не обслуживаются (сообщения не передаются). TTCNT сбрасывается в начале каждого нового матричного цикла (после правильной передачи в основном сообщении основного цикла «0»).

Биты 23-16 RTO7 – RTO0. Смещение основного запуска. Показывает текущее смещение основного запуска. Это значение считается как 2 с дополнением и может достигать значений между -127 и 127. Оно добавляется к временной метке (задается последними входными данными основного цикла в памяти планировщика) для запуска основного сообщения. Изменение и соответствующие условия приведены в таблице 153.

Биты 15-8 EXPTT7 – EXPTT0. Ожидаемые запуски передач. Определяет, сколько запросов на передачу ожидается в матричном цикле.

Бит 7 Зарезервировано. Читаются как «0». При записи следует писать «0».

Биты 6-0 IRO6 – IRO0. Начальное основное смещение. Определяет начальное исходное смещение запуска. Значение считается как 2 с дополнени-

ем и может достигать значений от 0 до 127.

Таблица 153 – Аппаратное изменение битового поля RTO

Изменение	Условия
Установка TTCFGR.IRO	Во время сброса (TTCFGR.IRO = 0) или режима конфигурации (программная запись TTCFGR.IRO).
	Каждый раз, когда возможный ведущий (Master) по времени получает основное сообщение с более высоким приоритетом, чем собственный.
Уменьшение на 1	Каждый раз, когда возможный ведущий (Master) по времени получает основное сообщение с более низким приоритетом, чем собственный, до достижения -127.
Установка 127	Когда входит в состояние S2.
Установка 0	Когда TTCAN узел правильно передается свое основное сообщение (если это текущий ведущий (Master) по времени).
	Когда возможный ведущий (Master) получает основное сообщение с приоритетом ниже, чем он сам и его RTO положительно.

Регистр состояния, срабатывающего по времени, содержит информацию о состоянии срабатывания по времени, которое не приводит непосредственно к прерыванию событий. Описание бит регистра состояния, срабатывающего по времени, TTSR представлено на рисунке 357.

31	30	29	28	27	26	25	24
Зарезервировано					ETREV	ETR	NIG
R-0					RH-0	RH-0	RH-0

23	22	21	20	19	18	17	16
Зарезервировано	MSCMAX2	MSCMAX1	MSCMAX0	Зарезервировано	MSCMIN2	MSCMIN1	MSCMIN0
R-0	RH-0	RH-0	RH-0	R-0	RH-0	RH-0	RH-0

15	14	13	12	11	10	9	8
RECF	TRAF	TMPC	CFGM	ARB	REFTRG	EFF	EFI
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

7	6	5	4	3	2	1	0
Зарезервировано		SYNCS1	SYNCS0	MSR1	MSR0	ERRS1	ERRS0
R-0		RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 357 – Регистр состояния, срабатывающий по времени, модуля CAN (TTSR) – адрес 0AC8h

Биты 31- 27 Зарезервировано. Читаются как «0». При записи следует писать «0».

Бит 26 ETREV. Запуск внешнего события. Указывает, что был обнаружен запуск внешнего события. ETREV автоматически очищается, когда основное сообщение было правильно передано на шину.
0 = Запуск выбранного внешнего события еще не был обнаружен.
1 = Запуск выбранного внешнего события был обнаружен.

Бит 25 ETR. Запрос внешнего запуска. Показывает состояние, ведущее к передаче следующего основного сообщения (только для текущего ведущего (Master) по времени). Значение NIG копируется в ETR, когда передано правильно основное сообщение. Бит автоматически очищается, когда правильно принято следующее основное сообщение.
0 = Следующее основное сообщение будет передано, когда достигнет соответствующей временной метки.
1 = Следующее основное сообщение будет передано, когда выбран запуск или происходит событие запуска RME.

Бит 24 NIG. Пропуск следующего. Показывает состояние, ведущее к передаче основного сообщения после следующего (только для текущего ведущего по времени). NIG будет передаваться со следующим основным сообщением.
0 = Основное сообщение будет передаваться после следующего при достижении соответствующей временной метки.
1 = Основное сообщение будет передаваться после следующего, когда происходит выбранное событие запуска. Бит NIG будет очищен и бит ETR будет установлен при следующем правильно переданном основном сообщении (прием или передача).

Бит 23 Зарезервировано. Читаются как «0». При записи следует писать «0».

Биты 22-20 MSCMAX2 – MSCMAX0. Максимум битового поля MSC. Указывает максимальное значение битового поля MSC задействованной области сообщения в исключительных временных окнах. Это значение равно 0 в начале нового матричного цикла. Оно обновляется в соответствии с: если MSC_cur > MSCMAX, то MSCMAX: = MSC_cur. Значение MSC_cur - это значение MSC в текущей задействованной области сообщения.

Бит 19 Зарезервировано. Читаются как «0». При записи следует писать «0».

Биты 18-16 MSCMIN2 – MSCMIN0. Минимум битового поля MSC. Указывает минимальное значение битового поля MSC задействованной области сообщения в исключительных временных окнах. Это значение равно 7 в начале нового матричного цикла. Оно обновляется в соответствии с: если MSC_cur < MSCMIN, то MSCMIN: = MSC_cur. Значение MSC_cur - это значение MSC в текущей задействованной области сообщения.

Бит 15 RECF. Флаг завершения приема. Устанавливается, когда CAN узел корректно завершает прием сообщения (корректное сообщение было замечено на шине). RECF очищается, когда будет достигнута следующая временная метка. RECF используется для генерации прерывания входных данных временной метки и прерывания входных данных.
0 = После достижения последней временной метки, CAN узел пока не принял сообщение.
1 = После достижения последней временной метки, CAN узел пока не принял сообщение.

Бит 14 TRAF. Флаг завершения передачи. Устанавливается, когда CAN узел корректно завершает передачу сообщения. TRAF очищается, когда будет достигнута следующая временная метка. TRAF используется для генерации прерывания входных данных временной метки и прерывания входных данных.
0 = После достижения последней временной метки, CAN узел пока не корректно завершил передачу.
1 = После достижения последней временной метки, CAN узел пока не корректно завершил передачу.

Бит 13 TMPC. Конфликт приоритета ведущего (Master) по времени. Показывает, есть ли столкновение приоритетов между ведущим (Master) по времени и TTCAN узла. Конфликт приоритетов возникает, если текущий ведущий (Master) по времени имеет более низкий приоритет, чем сам TTCAN узел (возможный ведущий (Master) по времени).
0 = Конфликт приоритета не был обнаружен в последнем основном сообщении.
1 = TTCAN узел является возможным ведущим (Master) по времени с конфликтом приоритета (последнее основное сообщение было получено с более низким приоритетом, чем TTCR.TMPRIO).

Бит 12 CFGM. Флаг режима конфигурации. Указывает, активен или нет режим конфигурации:
0 = TTCAN узел не в режиме конфигурации.
1 = TTCAN узел в режиме конфигурации. Не происходит передача сообщений и обновления битового поля MSC.

Бит 11 ARB. Флаг арбитражного окна. Устанавливается, когда открыто арбитражное окно. ARB очищается, когда закрыто арбитражное окно
0 = Текущее временное окно является окном, срабатывающим по времени.
1 = Текущее временное окно является арбитражным окном.

Бит 10 REFTRG. Флаг основного запуска. Устанавливается, когда основное сообщение TTCAN узла предназначено для отправки. REFTRG очи-

щается, когда основное сообщение было корректно отправлено на выход или было правильно принято основное сообщение (отправленное другим ведущим (Master) по времени).

Бит 9 EFF. Флаг ошибки фрейма. Отслеживает индикацию ошибки фрейма EFI, когда достигается временная метка.
0 = Ошибка CAN не обнаружена в последнем временном окне.
1 = Ошибка CAN обнаружена в последнем временном окне.

Бит 8 EFI. Индикация ошибки фрейма. Устанавливается, когда обнаружена ошибка состояния в битовом потоке CAN фреймов и ошибка фреймов послана на выход. EFI автоматически очищается, когда будет достигнута следующая временная метка и его состояние на данный момент отслеживается битом EFF.
0 = После достижения последней временной метки, CAN узел не обнаруживает ошибку состояния в битовом потоке CAN.
1 = После достижения последней временной метки, CAN узел обнаруживает ошибку состояния в битовом потоке CAN. Когда устанавливается EFI, CAN узлом может быть сгенерировано прерывание LEC.

Биты 7-6 Зарезервировано. Читаются как «0». При записи следует писать «0».

Биты 5-4 SYNCS1, SYNCS0. Состояние синхронизации. Показывает текущее состояние синхронизации TTCAN узла:
00 = Синхронизация отключена.
01 = Синхронизация.
10 = В режиме пропуска.
11 = В режиме планировщика.

Биты 3-2 MSR1, MSR0. Отношение ведущий (Master) – ведомый (Slave). Показывает текущее отношение главного (Master) к подчиненному (Slave) для TTCAN узла.
00 = Ведущий (Master) отключен.
01 = Ведомый (Master) (устройство приема).
10 = Возможный ведущий (Master) по времени.
11 = Текущий ведущий (Master) по времени.

Биты 1-0 ERRS1, ERSS0. Состояние ошибки. Показывает текущий уровень серьезности ошибки.
00 = Ошибки нет.
01 = Опасность.
10 = Ошибка.
10 = Серьезная ошибка.

Состояния битов/флагов TTCAN могут быть изменены при выполнении операции записи в регистр флага модификации, срабатывающего по времени. Описание бит регистра флага модификации, срабатывающего по времени, TTFMR представлено на рисунке 358.

R – доступ чтения, W - доступ записи, -0 – значение после сброса

Рисунок 358 – Регистр флага модификации, срабатывающий по времени, модуля CAN (TTFMR) – адрес 0ACCh

Биты 31- 6 Зарезервировано. Читаются как «0». При записи следует писать «0».

Бит 5 ETREVR. Сброс запуска внешнего события. Очищает флаг запуска внешнего события.
0 = Нет действия.
1 = Очищается бит TTSR.ETREV.

Бит 4 STE. Программный запуск события. Используется для программной синхронизации TTCAN узла.
0 = Нет действий.
1 = Запускается передача основного сообщения, если TTSR.ETREV = 11_2 и система в режиме пропуска.

Биты 3-2 NIGSR1, NIGSR0. Флаг установки/сброса пропуска следующего. Используется для установки/очищения бита TTSR.NIG и очищения бита TTSR.ETR.
00 = Нет действий.
01 = Бит TTSR.NIG может быть очищен.
10 = Биты TTSR.NIG и TTSR.ETR могут быть очищены.
11 = Нет действий.

Биты 1-0 CFGMEL1, CFGVEL0. Вход/продление режима конфигурации. Используется для входа продления режима конфигурации.
00 = Нет действий.
01 = Возможен вход в режим конфигурации (устанавливается TTSR.CFGM).
10 = Возможно продление режима конфигурации (очищается TTSR.CFGM).

11 = Нет действий.

Регистр запроса на прерывание, срабатывающего по времени, содержит информацию о состоянии срабатывания по времени вместе с прерываниями событий. Обратите внимание, что все биты в TTIRR могут быть очищены программно, записав «0». Запись «1» в бит не имеет никакого эффекта. Описание бит регистра запроса на прерывание, срабатывающего по времени, TTIRR представлено на рисунке 359.

31						18	17	16
Зарезервировано							TURERR	CFGERR
R-0							RWH-0	RWH-0
15	14	13	12	11	10	9	8	
SERR2	SERR1	DISC	WFE	EOS	SYNCSC	MSRC	ERRSC	
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	
7	6	5	4	3	2	1	0	
AWDERR	IWTE	WTE	TTOF	TTUF	TENWER	NBC	NMC	
RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	RWH-0	

R – доступ чтения, W – доступ записи, H – аппаратное влияние, -0 – значение после сброса

Рисунок 359 – Регистр запроса на прерывание, срабатывающий по времени, модуля CAN (TTIRR) – адрес 0AD0h

Биты 31- 18 Зарезервировано. Читаются как «0». При записи следует писать «0».

Бит 17 TURERR. Ошибка настройки TUR. Указывает, что была обнаружена ошибка настройки TUR. Ошибка настройки TUR обнаруживается, когда включена автоматическая регулировка TUR и случается переполнение или опустошение рассчитанного значения TURADJ.
0 = Ошибка настройки TUR не была обнаружена.
1 = Ошибка настройки TUR была обнаружена.

Бит 16 CFGERR. Ошибка конфигурации. Указывает, была ли обнаружена планировщиком ошибка конфигурации. Ошибка конфигурации обнаруживается, когда арбитражное окно не закрывается при достижении BCE. Когда RME.TMV достигнута, ни одни другие входные данные планировщика, как RME и BCE, не допускаются. Смотрите также EOSERR.
0 = Ошибка конфигурации не была обнаружена.
1 = Ошибка конфигурации была обнаружена (состояние ошибки S3).

Бит 15 SERR2. Флаг ошибки 2 планировщика. Устанавливается, когда обнаружена ошибка планировщика второго типа. Ошибка планировщика может изменить ошибку состояния TTCAN и, как следствие, может быть указано прерывание изменения ошибки состояния.
0 = Ошибка планировщика второго типа не была обнаружена.
1 = Ошибка планировщика второго типа была обнаружена.

Бит 14 SERR1. Флаг ошибки 1 планировщика. Устанавливается, когда обнаружена ошибка планировщика первого типа. Ошибка планировщика может изменить ошибку состояния TTCAN и, как следствие, может быть указано прерывание изменения ошибки состояния.
0 = Ошибка планировщика первого типа не была обнаружена.
1 = Ошибка планировщика первого типа была обнаружена.

Бит 13 DISC. Флаг разрыва. Устанавливается, когда принято основное сообщение с указанием разрыва (не для текущего ведущего (Master) по времени).
0 = Последнее полученное основное сообщение не указывает на разрыв.
1 = Последнее полученное основное сообщение указывает на разрыв. Это событие может генерировать прерывание оповещения.

Бит 12 WFE. Флаг ожидания события. Устанавливается, когда принято основное сообщение с указанием на следующий пропуск (не для текущего ведущего (Master) по времени).
0 = Последнее полученное основное сообщение не указывает на следующий пропуск.
1 = Последнее полученное основное сообщение указывает на следующий пропуск. Это событие может генерировать прерывание оповещения.

Бит 11 EOS. Флаг окончания входных данных планировщика. Устанавливается, когда TTCAN планировщик считывает входные данные EOS.
0 = TTCAN планировщик не считал входные данные EOS.
1 = TTCAN планировщик считал входные данные EOS. В этом случае автоматически устанавливается TTSR.CFGM.

Бит 10 SYNCSC. Изменение состояния синхронизации. Указывает изменение битового поля TTSR.SYNCS.
0 = TTSR.SYNCS не изменилось.
1 = TTSR.SYNCS изменилось.

Бит 9 MSRC. Изменение отношения ведущий (Master) – ведомый (Slave). Указывает изменение битового поля MSR.
0 = MSR не изменилось.
1 = MSR изменилось.

Бит 8 ERRSC. Изменение состояния ошибки. Указывает изменение битового поля TTSR.ERRS.
0 = TTSR.ERRS не изменилось.
1 = TTSR.ERRS изменилось.

Бит 7 AWDERR. Ошибка применения сторожевого таймера. Указывает, что приложение сторожевого таймера, уменьшенное до 0, не обслуживается.
0 = AWDR.AWDV еще не достиг значения 00_{16}.
1 = AWDR.AWDV достиг значения 00_{16} (состояние ошибки S3).

Бит 6 IWTE(2). Инициализация сторожевого триггера по событию. Указывает на сторожевой триггер по событию WTE со значением Init_Watch_Trigger. Это событие обнаруживается, когда время цикла в CYCTMR.CYCTM становится равным значению инициализированного сторожевого триггера 2^{16}-1.
0 = Время цикла пока не равно 2^{16}-1.
1 = Время цикла равно 2^{16}-1.

Бит 5 WTE(1). Сторожевой триггер по событию. Указывает на сторожевой триггер по событию WTE. Это событие определяется, когда время цикла в CYCTMR.CYCTM становится равным значению сторожевого триггера, определенного по временной метке входных данных BCE.
0 = Время цикла не равно WTE.
1 = Время цикла равно WTE.

Бит 4 TTOF. Переполнение запуска передачи. Указывает, что было предложено больше запусков передачи во время матричного цикла, чем указано в TTCFGR.EXPTT.
0 = Было предложено ожидаемое (или меньшее) число запусков передачи.
1 = Было предложено большее число запусков передачи.

Бит 3 TTUF. Опустошение запуска передачи. Указывает, что было предложено меньше запусков передачи во время матричного цикла, чем указано в TTCFGR.EXPTT.
0 = Было предложено ожидаемое (или больше) число запусков передачи.
1 = Было предложено меньше число запусков передачи.

Бит 2 TENWER. Ошибка разрешения окна передачи. Указывает, что заданное время истекло после запуска передачи без запуска передачи сообщения.
0 = Запускаемые сообщения были разосланы до истекшего разрешения окна передачи.

1 = Запуск сообщений не начался до истечения разрешения окна передачи.

Бит 1 NBC. Новый основной цикл. Указывает на старт нового основного цикла.
0 = Новый основной цикл не был обнаружен.
1 = Новый основной цикл был обнаружен.

Бит 0 NMC. Новый матричный цикл. Указывает на старт нового матричного цикла.
0 = Новый матричный цикл не был обнаружен.
1 = Новый матричный цикл был обнаружен.

_____

[1] WTE может быть сгенерирован только после того, как первое сообщение было передано на шину.

[2] IWTE может быть сгенерирован только до того, как первое сообщение было передано на шину.

Описание бит регистра разрешения прерывания, срабатывающего по времени, TTIER представлено на рисунке 360.

31 … 13	12	11	10	9	8
Зарезервировано	NOTIFIE	SEIE	SYNCSEIE	MSRCIE	ERRSCIE
R-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5 … 4	3	2	1	0
AWDIE	WTEIE	Зарезервировано	TTERIE	TENWERIE	NBCIE1	NBCIE0
RW-0	RW-0	R-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 360 – Регистр разрешения прерывания, срабатывающий по времени, модуля CAN (TTIER) – адрес 0AD4h

Биты 31- 18 Зарезервировано. Читаются как «0». При записи следует писать «0».

Бит 12 NOTIFIE. Разрешение прерывания оповещения. Это прерывание генерируется всякий раз, когда бит TTIRR.WFE или TTIRR.DISC аппаратно устанавливается. Битовое поле TTINPR.NOTIFINP выбирает выходную линию прерывания, которая активируется при этом типе прерывания.
0 = Прерывание оповещения запрещено.
1 = Прерывание оповещения разрешено.

Бит 11 SEIE. Разрешение прерывания ошибки планировщика. Это прерывание генерируется всякий раз, когда биты EOS, или SERR1, или SERR2, или CFGERR регистра TTSR аппаратно устанавливаются.

Битовое поле TTINPR.ERRINP выбирает выходную линию прерывания, которая активируется при этом типе прерывания.
0 = Генерация прерывания ошибки планировщика запрещена.
1 = Генерация прерывания ошибки планировщика разрешена.

Бит 10 SYNCSEIE. Разрешение прерывания изменения состояния синхронизации. Это прерывание генерируется, когда изменяется состояние битового поля TTSR.SYNCSC. Битовое поле TTINPR.NOTIFINP выбирает выходную линию прерывания, которая активируется при этом типе прерывания.
0 = Прерывание изменения состояния синхронизации запрещено.
1 = Прерывание изменения состояния синхронизации разрешено.

Бит 9 MSRCIE. Разрешение прерывания изменения отношения ведущий (Master) – ведомый (Slave). Это прерывание генерируется, когда битовое поле TTSR.MSRC изменяет свое состояние. Битовое поле TTINPR.NOTIFINP выбирает выходную линию прерывания, которая активируется при этом типе прерывания.
0 = Прерывание изменения отношения ведущий (Master)-ведомый (Slave) запрещено.
1 = Прерывание изменения отношения ведущий (Master)-ведомый (Slave) разрешено.

Бит 8 ERRSIE. Разрешение прерывания изменения состояния ошибки. Это прерывание генерируется, когда битовое поле TTSR.ERRSC изменяет свое состояние. Битовое поле TTINPR.NOTIFINP выбирает выходную линию прерывания, которая активируется при этом типе прерывания.
0 = Прерывание изменения состояния ошибки запрещено.
1 = Прерывание изменения состояния ошибки разрешено.

Бит 7 AWDIE. Разрешение прерывания применения сторожевого таймера. Это прерывание генерируется, когда устанавливается TTIRR.AWDERR (независимо от его текущего состояния). Битовое поле TTINPR.ERRINP выбирает выходную линию прерывания, которая активируется при этом типе прерывания.
0 = Прерывание применения сторожевого таймера запрещено.
1 = Прерывание применения сторожевого таймера разрешено.

Бит 6 WTEIE. Разрешение прерывания сторожевого триггера по событию. Это прерывание генерируется, когда устанавливается TTIRR.IWTE или TTIRR.WTE (независимо от его текущего состояния). Битовое поле TTINPR.ERRINP выбирает выходную линию прерывания, которая активируется при этом типе прерывания.
0 = Прерывание сторожевого триггера по событию запрещено.
1 = Прерывание сторожевого триггера по событию разрешено.

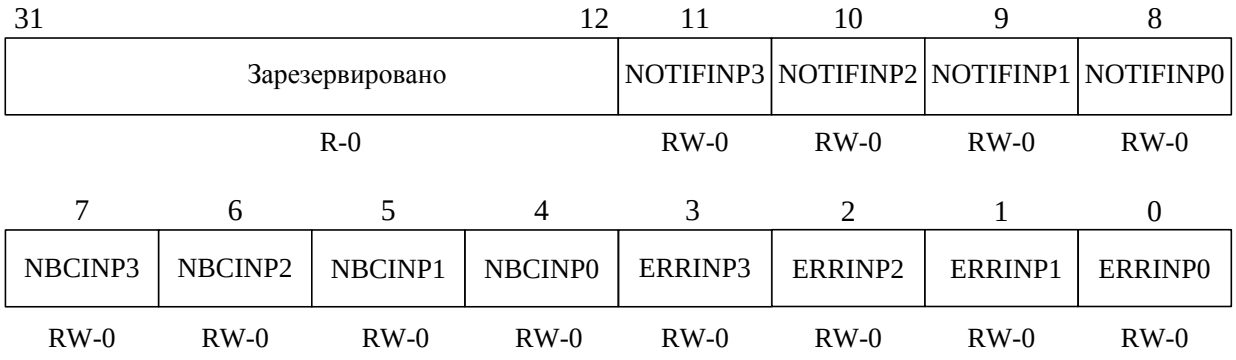
Биты 5-4 Зарезервировано. Читаются как «0». При записи следует писать «0».

Бит 3 TTERIE. Разрешение прерывания ошибки запуска передачи. Это прерывание генерируется, когда устанавливается TTIRR.TTOF или TTIRR.TTUF (независимо от его текущего состояния). Битовое поле TTINPR.ERRINP выбирает выходную линию прерывания, которая активируется при этом типе прерывания.
0 = Прерывание ошибки запуска передачи запрещено.
1 = Прерывание ошибки запуска передачи разрешено.

Бит 2 TENWERIE. Разрешение прерывания ошибки разрешения запуска окна передачи. Это прерывание генерируется, когда устанавливается бит TTIRR. TENWER (независимо от его текущего состояния). Битовое поле TTINPR.ERRINP выбирает выходную линию прерывания, которая активируется при этом типе прерывания.
0 = Прерывание прерывания ошибки разрешения запуска окна передачи запрещено.
1 = Прерывание прерывания ошибки разрешения запуска окна передачи разрешено.

Биты 1-0 NBCIE0, NBCIE1. Разрешение прерывания нового основного цикла. Разрешаются прерывания нового основного или матричного цикла. Это прерывание генерируется, когда устанавливается бит TTIRR.NBC или бит TTIRR.NMC (независимо от его текущего состояния). Битовое поле TTINPR.NBCINP выбирает выходную линию прерывания, которая активируется при этом типе прерывания.
00 = Прерывание нового основного или матричного цикла запрещено.
01 = Прерывание основного цикла генерируется всякий раз, когда устанавливается TTIRR.NBC.
10 = Прерывание матричного цикла генерируется всякий раз, когда устанавливается TTIRR.NMC.
11 = Зарезервировано.

Описание бит регистра прерывания указателя узла, срабатывающего по времени, TTINPR представлено на рисунке 361.



R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 361 – Регистр прерывания указателя узла, срабатывающий по времени, модуля CAN (TTINPR) – адрес 0AD8h

Биты 31- 12 Зарезервировано. Читаются как «0». При записи следует писать «0».

Биты 11-8 NOTIFINP3 – NOTIFINP0. Уведомление о прерывании указателя узла. Выбор выходной линии прерывания (int_req_o[15:0]) для уведомления прерывания.
0000_2 = Выбирается выходная линия прерывания int_req_o[0];
0001_2 = Выбирается выходная линия прерывания int_req_o[1];
. . .
1110_2 = Выбирается выходная линия прерывания int_req_o[14];
1111_2 = Выбирается выходная линия прерывания int_req_o[15].
Для прерывания указателя узла возможными уведомлениями являются:
- изменения битового поля TTSR.ERRS;
- изменения битового поля TTSR.MSR;
- изменения битового поля TTSR.SYNCS;
- установка бита TTIRR.WFE;
- установка бита TTIRR.DISC.

Биты 7-4 NBCINP3 – NBCINP0. Прерывание нового основного цикла указателя узла. Выбор выходной линии прерывания (int_req_o[15:0]) для прерывания нового базового или матричного цикла.
0000_2 = Выбирается выходная линия прерывания int_req_o[0];
0001_2 = Выбирается выходная линия прерывания int_req_o[1];
. . .
1110_2 = Выбирается выходная линия прерывания int_req_o[14];
1111_2 = Выбирается выходная линия прерывания int_req_o[15].
Это прерывание может быть сгенерировано только тогда, когда принято основное сообщение. Прерывание для передаваемых основных сообщений контролируется основным сообщением входных данных в планировщике.

Биты 3-0 ERRINP3 – ERRINP0. Прерывание ошибки указателя узла. Выбор выходной линии прерывания (int_req_o[15:0]) для прерывания ошиб-

ки.
0000_2 = Выбирается выходная линия прерывания int_req_o[0];
0001_2 = Выбирается выходная линия прерывания int_req_o[1];
. . .
1110_2 = Выбирается выходная линия прерывания int_req_o[14];
1111_2 = Выбирается выходная линия прерывания int_req_o[15].
Для прерывания указателя узла возможными ошибками являются:
- ошибка разрешения окна передачи;
- ошибка запуска передачи;
- (инициализация) сторожевой триггер по событию;
- применение сторожевого таймера;
- ошибка планировщика.

Битовые поля TMV в регистрах состояния планировщика времени отслеживают информацию о времени метки для запуска следующего временного окна после этапа сбора инструкций планировщиком (собранной из TME, RME или BCE входных данных). Описание бит регистра (младший) состояния планировщика времени STSRL представлено на рисунке 362.

31	30	29	28	27	26	25	24
BCETMV15	BCETMV14	BCETMV13	BCETMV12	BCETMV11	BCETMV10	BCETMV9	BCETMV8
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
23	22	21	20	19	18	17	16
BCETMV7	BCETMV6	BCETMV5	BCETMV4	BCETMV3	BCETMV2	BCETMV1	BCETMV0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
15	14	13	12	11	10	9	8
RMETMV15	RMETMV14	RMETMV13	RMETMV12	RMETMV11	RMETMV10	RMETMV9	RMETMV8
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0
7	6	5	4	3	2	1	0
RMETMV7	RMETMV6	RMETMV5	RMETMV4	RMETMV3	RMETMV2	RMETMV1	RMETMV0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 362 – Регистр (младший) состояния планировщика времени модуля CAN (STSRL) – адрес 0AF0h

Биты 31- 16 BCETMV15 – BCETMV0. Значение временной метки для BCE. Указанное значение сравнения для следующей временной метки определенно BCE. Это значение является действительным, только если SISR.ICF = 1.

Биты 15-0 RMETMV15 – RMETMV0. Значение временной метки для RME. Указанное значение сравнения для следующей временной метки

определенно RME. Это значение является действительным, только если SISR.ICF = 1 и SISR.RMEV = 1.

Описание бит регистра (старший) состояния планировщика времени STSRH представлено на рисунке 363.

31	30	29	28	27	26	25	24
TCEMSGNR7	TCEMSGNR6	TCEMSGNR5	TCEMSGNR4	TCEMSGNR3	TCEMSGNR2	TCEMSGNR1	TCEMSGNR0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

23	22	21	20	19	18	17	16
RCEMSGNR7	RCEMSGNR6	RCEMSGNR5	RCEMSGNR4	RCEMSGNR3	RCEMSGNR2	RCEMSGNR1	RCEMSGNR0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

15	14	13	12	11	10	9	8
TMETMV15	TMETMV14	TMETMV13	TMETMV12	TMETMV11	TMETMV10	TMETMV9	TMETMV8
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

7	6	5	4	3	2	1	0
TMETMV7	TMETMV6	TMETMV5	TMETMV4	TMETMV3	TMETMV2	TMETMV1	TMETMV0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 363 – Регистр (старший) состояния планировщика времени модуля CAN (STSRH) – адрес 0AF4h

Биты 31- 24 TCEMSGNR7 – TCEMSGNR0. Количество сообщений входных данных управления передачей. Указывает на собранную информацию TCEMSGNR от TCR. Это значение учитывается, только если SISR.ICF = 1 и SISR.TCEV = 1.

Биты 23-16 RCEMSGNR7 – RCEMSGNR0. Количество сообщений входных данных управления приемом. Указывает на собранную информацию RCEMSGNR от TCR. Это значение учитывается, только если SISR.ICF = 1 и SISR.RCEV = 1.

Биты 15-0 TMETMV15 – TMETMV0. Значение временной метки для TME. Указанное значение сравнения для следующей временной метки определенно TME. Это значение является действительным, только если SISR.ICF = 1 и SISR.TMEV = 1.

Биты в регистре состояния инструкций планировщика отслеживают информацию во время и после этапа сбора инструкций планировщика. Эти значения вступят в силу, когда будет достигнута следующая временная метка. Описание бит регистра состояния инструкций планировщика SISR представлено на рисунке 364.

31 – 23	22	21	20	19	18	17	16
Зарезервировано	BCEV	RMEV	TMEV	ARBV	ICEV	TCEV	RCEV
R-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

15	14	13	12	11	10	9	8
ICF	GM	ARBM1	ARBM0	ALTMSG1	ALTMSG0	TREN	CHEN
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

7	6	5	4	3	2	1	0
IENRECF1	IENRECF0	IENTRAF1	IENTRAF0	INP3	INP2	INP1	INP0
RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0	RH-0

R – доступ чтения, H – аппаратное влияние, -0 – значение после сброса

Рисунок 364 – Регистр состояния инструкций планировщика модуля CAN (SISR) – адрес 0AF8h

Биты 31- 23 Зарезервировано. Читаются как «0». При записи следует писать «0».

Бит 22 BCEV. Действующее завершение основного цикла входных данных. Указывает, что действующее завершение основного цикла входных данных было найдено во время сбора команд для данного временного окна. BCEV автоматически очищается при достижении временной метки.
0 = Действующий BCE не был найден.
1 = Действующий BCE был найден.

Бит 21 RMEV. Действующая основная метка входных данных. Указывает, что действующая основная метка входных данных была найдена во время сбора команд для данного временного окна. RMEV автоматически очищается при достижении временной метки.
0 = Действующая RME не была найдена.
1 = Действующая RME была найдена.

Бит 20 TMEV. Действующая временная метка входных данных. Указывает, что действующая временная метка входных данных была найдена во время сбора команд для данного временного окна. TMEV автоматически очищается при достижении временной метки.

0 = Действующая TME не была найдена.
1 = Действующая TME была найдена.

Бит 19 ARBV. Действующий арбитраж входных данных. Указывает, что действующий арбитраж входных данных был найден во время сбора команд для данного временного окна. ARBV автоматически очищается при достижении временной метки.
0 = Действующий ARBV не был найден.
1 = Действующий ARBV был найден.

Бит 18 ICEV. Действующее управление прерыванием входных данных. Указывает, что действующее управление прерыванием входных данных было найдено во время сбора команд для данного временного окна. ICEV автоматически очищается при достижении временной метки.
0 = Действующее ICE не было найдено.
1 = Действующее ICE было найдено.

Бит 17 TCEV. Действующее управление передачей входных данных. Указывает, что действующее управление передачей входных данных было найдено во время сбора команд для данного временного окна. TCEV автоматически очищается при достижении временной метки.
0 = Битовые поля TREN, ALTMSG и TCEMSGNR недействительны. Они не принимаются во внимание для следующего временного окна.
1 = Битовые поля TREN, ALTMSG и TCEMSGNR действительны. Они принимаются во внимание для следующего временного окна.

Бит 16 RCEV. Действующее управление приемом входных данных. Указывает, что действующее управление приемом входных данных было найдено во время сбора команд для данного временного окна. RCEV автоматически очищается при достижении временной метки.
0 = Битовые поля CHEN и RCEMSGNR недействительны. Они не принимаются во внимание для следующего временного окна.
1 = Битовые поля CHEN и RCEMSGNR действительны. Они принимаются во внимание для следующего временного окна.

Бит 15 ICF. Завершение сбора инструкций. Указывает, что сбор инструкций закончен для данного временного окна. ICF автоматически очищается при достижении временной метки. ICF устанавливается, когда прекращается сбор инструкций.
0 = Сбор инструкций еще не прекращен. Все остальные значения в регистрах SISR, STSR0 и STSR4 недействительны.
1 = Сбор инструкций прекращен. Все остальные значения в регистрах SISR, STSR0 и STSR4 действительны.

Бит 14 GM. Режим разрыва. Указывает на собранную информацию GM от RME или BCE (действует, только если были найдены RME или

BCE).

Биты 13-12 ARBM1, ARBM0. Режим арбитража. Указывает на собранную информацию ARBM от TME или ARBE.

Биты 11-10 ALTMSG1, ALTMSG0. Альтернативное сообщение. Указывает на собранную информацию ALTMSG от TCE.

Бит 9 TREN. Разрешение передачи. Указывает на собранную информацию TREN от TCE.

Бит 8 CHEN. Проверка разрешения. Указывает на собранную информацию CHEN от RCE. CHEN учитывается, только если EC указывает на RCE.

Бит 7 IENRECF1. Разрешение прерывания, если RECF = 1. Указывает на собранную информацию IENRECF1.

Бит 6 IENRECF0. Разрешение прерывания, если RECF = 0. Указывает на собранную информацию IENRECF0.

Бит 5 IENTRAF1. Разрешение прерывания, если TRAF = 1. Указывает на собранную информацию IENTRAF1.

Бит 4 IENTRAF0. Разрешение прерывания, если TRAF = 0. Указывает на собранную информацию IENTRAF0.

Биты 3-0 INP3 – INP0. Прерывание указателя узла. Указывает на собранную информацию INP. Это значение учитывается только тогда, когда по крайней мере один из четырех запросов прерывания включен.

Примечание - Этот регистр сбрасывается в начале новой фазы сбора инструкции.

Описание бит регистра планировщика стартового указателя узла STPTR0 представлено на рисунке 365.

31 7	6	5	4	3	2	1	0
Зарезервировано	STPTR6	STPTR5	STPTR4	STPTR3	STPTR2	STPTR1	STPTR0
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W – доступ записи, -0 – значение после сброса

Рисунок 365 – Регистр планировщика стартового указателя узла

модуля CAN (STPTR0) – адрес 47FCh

Биты 31-7	Зарезервировано. Читаются как «0». При записи следует писать «0».
Бит 6-0	STPTR6 –STPTR0. Начальный указатель. Определяет расположение первого планировщика входных данных для TTCAN узла 0. Значение определяет сколько входных данных (учитывается в единицах по 32 бит) первого TME входных данных (TME1) для этого TTCAN узла расположено ниже последнего адреса в памяти планировщика. Если реализовано два или более TTCAN узла, все начальные указатели обращаются в конец (последний адрес) памяти планировщика.

17 Модуль интерфейса I2C

Модуль интерфейса I2C (далее – модуль I2C) – это последовательный двухпроводной интерфейс, совместимый с шиной SMBus на физическим уровне. Он также совместим с шинами ACCESS.Bus и Philips' I2C. I2C может быть сконфигурирован как ведущая (Master) шина или ведомая (Slave), и может поддерживать двухстороннюю связь как с ведущими Master, так и ведомыми Slave устройствами.

Функциональные возможности модуля:

- Совместимость с SMBus (Version 1.1 и 2.0), ACCESS.Bus, I2C (Version 2.1).

- Поддержка стандартного (Standard), скоростного (F/S) и высокоскоростного (Hs) режимов.

- Программирование интерфейса для работы в режиме ведущего (Master) или ведомого (Slave).

- Возможность подключения к шине нескольких ведущих устройств (режим Multi-Master).

- Один программно определяемый 7/10-битный адрес ведомого (Slave) устройства.

- Возможность одновременного обращения ко всем устройствам шины, так называемый «общий вызов» (global call).

- Особые возможности SMBus:

  - отслеживание времени ожидания линии SCL;

  - наличие функции PEC (Packet Error Checking – отслеживание ошибок в пакетах данных) с использованием стандартного полинома SMBus;

  - возможность обращения одного или нескольких ведомых (Slave) к ведущему (Master) с получением отклика от последнего.

- Возможность последовательного опроса.

- Функционирование под управлением прерываний.

Структурная схема модуля изображена на рисунке 366.

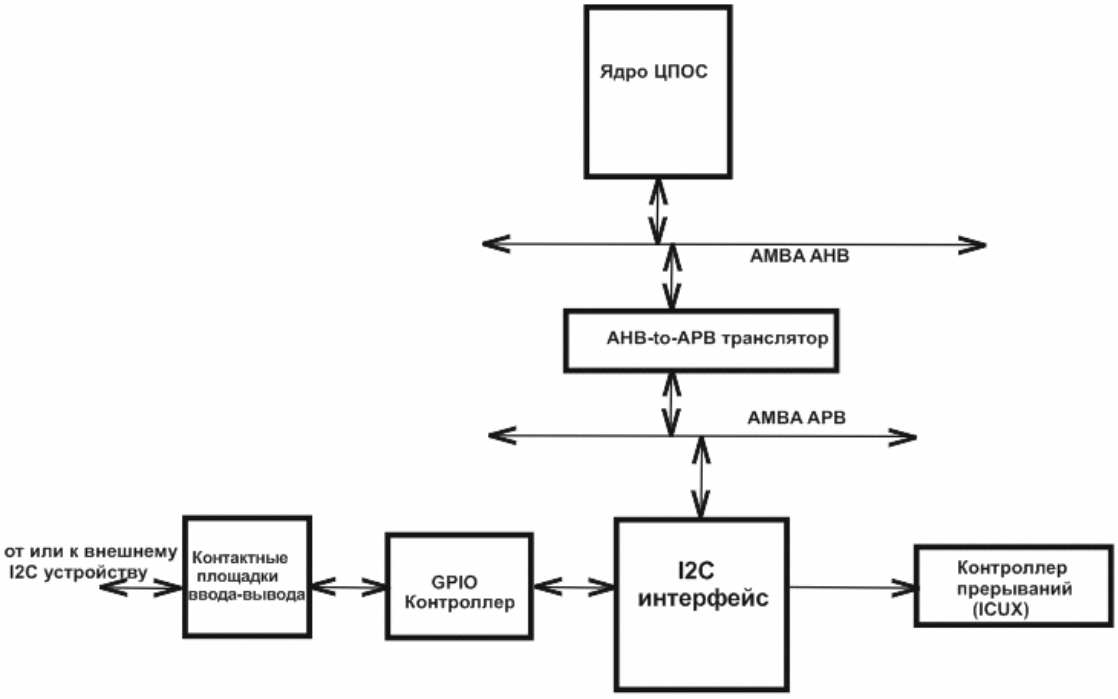


Рисунок 366 – Структурная схема модуля интерфейса I2C

17.1 Протокол шины

Модуль интерфейса I2C (далее – I2C) обеспечивает полную поддержку двухпроводного последовательного синхронного интерфейса I2C/SMBUS. Он также совместим с шинами ACCESS.Bus и Philips' I2C. Результат такой совместимости – легкое соединение со многими запоминающими устройствами и устройствами ввода-вывода, включая EEPROM, SRAM, счетчики, АЦП, ЦАП, периферийные устройства.

Протокол I2C использует двухпроводной интерфейс для двусторонней связи между устройствами, подключенными к шине. Двунаправленная шина состоит из двух линий: линии данных (Serial Data Line – SDA) и линии тактового сигнала (Serial Clock Line – SCL). Эти линии подключены к источнику питания через подтягивающие резисторы. Шинные формирователи любых устройств, подключаемых к шине, выполняются по схеме с открытым коллектором или открытым стоком. Устройства могут выставить только низкий уровень на соответствующей линии. Следовательно, обе линии SDA и SCL реализуют функцию «монтажное И».

Протокол поддерживает режим Multi-Master – одно или несколько устройств, подключенных к шине, могут контролировать шину. Каждому устройству, подключенному к шине, присваивается собственный адрес, оно может выступать как передатчик или приемник, хотя некоторые периферийные устройства являются только приемниками.

17.1.1 Операции с данными

Устройство, которое начинает передачу данных, становится ведущим (Master). Ведущий (Master) генерирует тактовый сигнал SCL, а также инициирует и завершает передачу данных по шине.

Один бит данных передается по SDA в течение каждого импульса на SCL, показанный на рисунке 367. Данные стабильны, пока уровень сигнала на линии SCL высокий, могут меняться пока уровень сигнала на линии SCL низкий.

Каждая передача данных состоит из начального состояния «Старт», ряда передач байтов данных и состояния «Стоп» при окончании передачи данных. Первым передается старший разряд (Most Significant Bit – MSB). После каждого байта (8 бит) должен следовать сигнал подтверждения ACK.

Рисунок 367 – Передача бита данных

Ведомое (Slave) устройство может увеличивать паузу между тактовыми импульсами, удерживая на линии SCL сигнал низкого уровня, пока происходит обработка принятых данных или подготовка данных для следующей передачи. Этот процесс может происходить после передачи любого бита.

17.1.2 Состояния «Старт» и «Стоп»

Ведущий (Master) формирует состояния «Старт» и «Стоп», как показано на рисунке 368. После того, как было сформировано состояние «Старт», шина считается занятой и другие ведущие не должны пытаться управлять ей. Такой статус шина сохраняет до тех пор, пока не будет сформировано состояние «Стоп»:

а) Состояние «Старт» S – отрицательный перепад на SDA, когда сигнал на SCL высокого уровня.

б) Состояние «Стоп» P – положительный перепад на SDA, когда сигнал на SCL высокого уровня.

в) Состояние «Повторного старта» Sr, см. рисунок 369, – может быть сформировано в середине передачи, чтобы разрешить другому ведомому (Slave) обратиться к ведущему (Master) или чтобы изменить направление передачи данных без потери контроля над шиной.

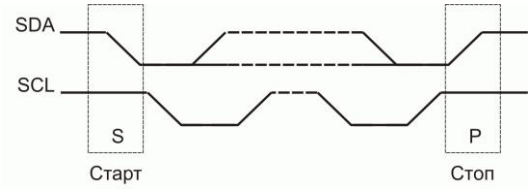


Рисунок 368 – Состояния «Старт» и «Стоп»

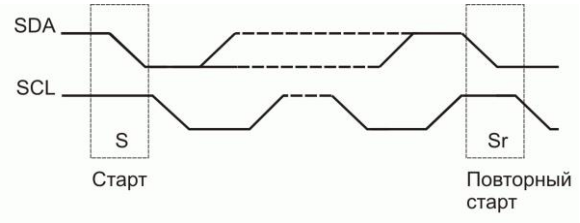


Рисунок 369 – Состояние «Повторный старт»

17.1.3 Цикл подтверждения

Цикл подтверждения, показанный на рисунках 370 и 371, включает в себя два сигнала:

- тактовый импульс подтверждения, который передается ведущим (Master) с каждым байтом данных;
- сигнал подтверждения ACK, посылаемый приемным устройством.

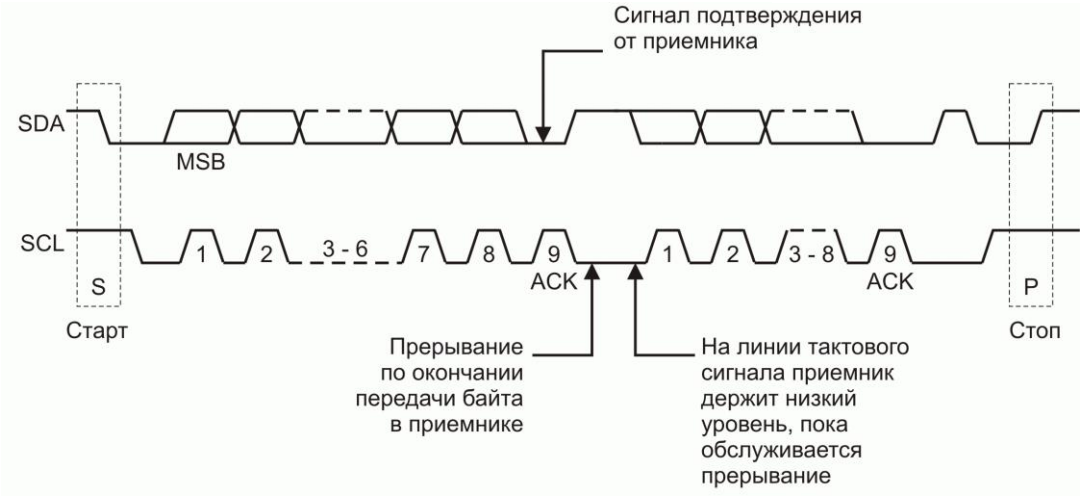


Рисунок 370 – Операции с данными

Во время девятого тактового импульса при передаче данных ведущий (Master) генерирует импульс подтверждения. Передатчик освобождает линию SDA (удерживает ее в высоком уровне), чтобы позволить приемнику отправить сигнал подтверждения. В течение импульса подтверждения приемник должен сформировать сигнал подтверждения, указывая тем самым на корректный прием последнего байта данных и готовность к приему следующего байта данных.

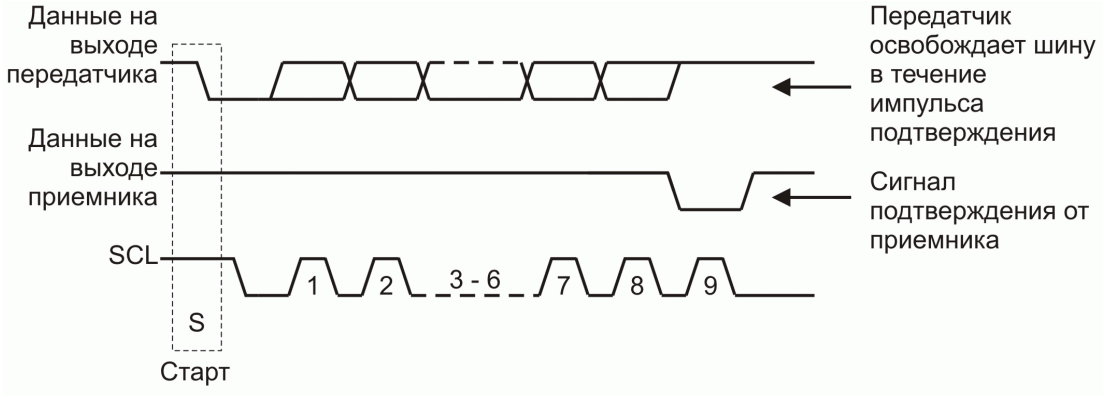


Рисунок 371 – Цикл подтверждения

17.1.4 Правило «Подтверждение после каждого байта»

Ведущий (Master) генерирует импульс подтверждения после каждого переданного байта, приемник должен формировать сигнал подтверждения после каждого принятого байта.

Существует два исключения из правила «Подтверждение после каждого байта»:

- когда ведущий (Master) является приемником, то при завершении передачи данных во время девятого тактового импульса он формирует инверсный сигнал подтверждения NACK;

- если возникли проблемы при приеме или приемник переполнен, он формирует инверсный сигнал подтверждения NACK, показывая, что не может принять дополнительные данные.

17.1.5 Формат передачи данных с 7-битной адресацией

На рисунке 372 показана завершенная передача данных. Каждому устройству, подключенному к шине, присваивается уникальный 7-битный адрес. Первые семь бит, передаваемые после состояния «Старт» – адрес ведомого (Slave), восьмой бит R/W# указывает направление передачи данных (передача ведомым (Slave) устройством – чтение, передача ведомому (Slave) устройству – запись).

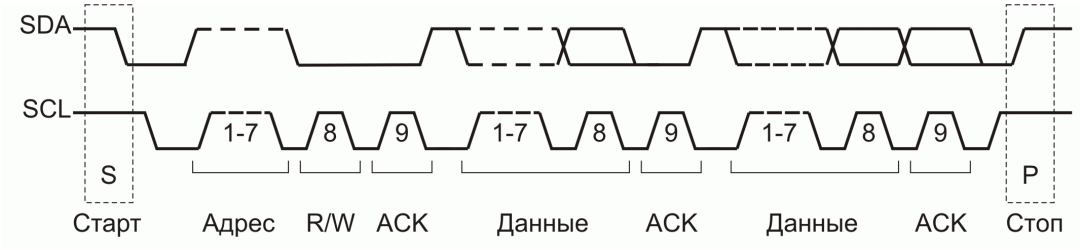


Рисунок 372 – Завершенная передача данных

Каждое устройство, подключенное к шине, сравнивает принятый адрес со своим собственным адресом. Если адрес устройства совпадает с принятым им адресом, то устройством формируется сигнал подтверждения. В зависимости от значения бита R/W# (1_2 – чтение, 0_2 – запись), адресуемое устройство становится ведомым (Slave) передатчиком или ведомым (Slave) приемником.

Протокол I2C предоставляет возможность одновременного обращения к устройствам, подключенным к шине при помощи адреса общего вызова. В первом передаваемом байте определяется специальный адрес общего вызова 00_{16}, во втором – значение общего вызова. Ведомые (Slave), готовые к приему данных, формируют сигнал подтверждения, остальные игнорируют общий вызов.

Протокол I2C также допускает работу в режиме Alert-отклика (Slave Transmit Alert Response Mode). Любой ведомый (Slave), или несколько ведомых (Slave), может послать сигнал запроса SMBAlert, с целью передать данные к ведущему (Master). Ведущий (Master), получив сигнал SMBAlert, в ответ выдает на шину специальный адрес отклика 0001100_2 с битом R/W#, равным 1_2 (чтение).

Устройство (устройства), пославшее сигнал SMBAlert, получив адрес отклика, выдает на шину свой 7-битный адрес (последний бит передаваемого байта может быть равным как 0_2, так и 1_2). Если несколько ведомых (Slave) выдают свои адреса на шину, то передачу осуществляет устройство, выигравшее арбитраж.

Как только в ответ на появление адреса отклика одним из устройств будет выдан на шину корректный адрес, сигнал SMBAlert должен быть сброшен программно (поддержка режима Alert-отклика является уникальной особенностью устройств SMBus).

17.1.6 Арбитраж

Процесс распределения приоритетов (арбитраж) на линии SDA происходит в то время, когда сигнал на линии SCL высокого уровня. Арбитраж возникает в случае, когда два ведущих (Master) одновременно сформировали состояние

«Старт», и продолжается до тех пор, пока одно устройство удерживает высокий уровень на SDA, в то время как другое удерживает низкий уровень на SDA (уровень сигнала на SCL – высокий). Ведущий (Master), удерживающий высокий уровень на SDA, теряет приоритет. Если ведущий (Master) теряет приоритет во время передачи адреса ведомого (Slave) (первый байт, следующий после формирования состояния «Старт»), то он переключается в режим «ведомый (Slave) приемник» и начинает сравнивать передаваемые адреса со своим адресом. Приоритет также может быть потерян в режиме «ведущий (Master) приемник» в течение цикла подтверждения или в режиме «ведомый (Slave) передатчик» во время получения сигнала отклика.

В случае потери приоритета устанавливаются соответствующие биты в статусном регистре SMBST (SMBST.MODE) и генерируется прерывание.

17.1.7 Синхронизация тактового сигнала

В случае, когда два и более ведущих (Master) пытаются одновременно начать передачу, необходима синхронизация их тактовых сигналов. Задача синхронизации решается просто благодаря тому, что все устройства подключаются к линии SCL по схеме «монтажное И». В результате длительность тактовых импульсов на линии SCL определяется тактовым сигналом с наименьшей длительностью импульсов, а пауза между тактовыми импульсами – тактовым сигналом с наибольшей паузой между импульсами.

17.1.8 Формат передачи данных с 10-битной адресацией

10-битная адресация, показанная на рисунке 373, позволяет использовать на шине I2C 1024 адреса ведомых (Slave). После формирования состояния «Старт» передается зарезервированный адрес $11110XX_2$. 10-битная адресация полностью совместима с 7-битной адресацией. Таким образом, ведомый (Slave) с 7- и 10-битной адресацией могут быть подключены к одной и той же шине.

Общее понятие заключается в формировании 10-битного адреса к ведомому (Slave) устройству первыми двумя байтами, следующими за стартовым битом. Первый байт после стартового бита содержит $11110yy0_2$, где yy обозначает два самых значимых бита (MSB) адреса ведомого (Slave). R/W# бит первого байта должен быть «0», чтобы указать, что второй байт адреса ведомого (Slave) устройства, содержащий младшие 8 бит, будет следовать за адресом ведомого устройства.

Чтобы прочитать 10-битное ведомое (Slave) устройство, стартовый бит повторно следует за вторым байтом адреса, который повторно передает $11110yy1_2$. Другими словами, отправляется первая часть адреса ведомого (Slave) устройства, но R/W# бит 1.

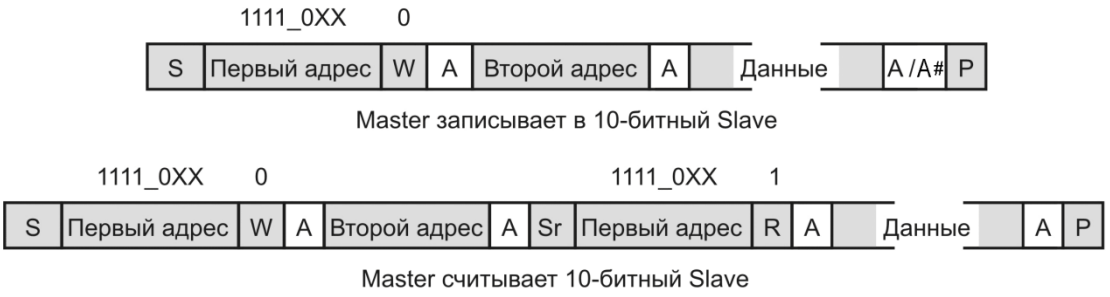


Рисунок 373 – 10-битная адресация Slave

17.1.9 Высокоскоростной режим (Hs)

По сравнению с режимом «Быстрый/Стандартный» (F/S) в высокоскоростном режиме повышается скорость передачи данных по шине. В режиме «Быстрый/Стандартный» скорость передачи ограничена 400 кГц и 100 кГц, соответственно, а в высокоскоростном режиме – 3,4 МГц, таким образом, пропускная способность увеличивается почти в 10 раз.

Шина переходит в высокоскоростной режим (Hs) из режима «Быстрый/Стандартный» с помощью следующих действий:

а) Формирование состояния «Старт» S;

б) 8-битный адрес ведущего (Master) $00001yyy_2$, где yyy – уникальный адрес каждого мастера, подключенного к шине;

в) Инверсный сигнал подтверждения A#, передаваемый для ведомого (Slave).

Так как адрес для каждого ведущего (Master) является уникальным, то процесс распределения приоритетов происходит только в момент передачи адреса ведущего (Master). После передачи адреса один ведущий (Master) получает приоритет в работе с шиной. Таким образом, арбитраж и синхронизация тактового сигнала не выполняются в высокоскоростном режиме.

После выполнения вышеуказанных действий устройства, поддерживающие высокоскоростной режим (Hs), подключаются к шине. Далее ведущий (Master) формирует состояние «Повторного старта» Sr, затем передается адрес ведомого (Slave) и бит направления передачи R/W#.

Формат операций с данными в высокоскоростном режиме (Hs) аналогичен режиму F/S, таким образом, режим Hs полностью совместим с устройствами, работающими в режиме F/S.

Завершение высокоскоростного режима (Hs), показанного на рисунке 374, производится формированием состояния «Стоп» P, после чего все устройства переключаются в режим F/S.

Рисунок 374 – Высокоскоростной режим (Hs)

17.2 Функциональное описание

На рисунке 375 показано строение модуля I2C.

17.2.1 Входные и выходные каскады линий SDA и SCL

Выводы SDA и SCL используются для подключения модуля к одноименным линиям шины I2C. Входные каскады, подключенные к выводам SDA и SCL, содержат фильтры, подавляющие помехи. В режиме F/S эти фильтры подавляют все входные сигналы с длительностью меньше, чем период тактового сигнала. Выходные каскады содержат устройства с предустановкой, выполненные по схеме с открытым стоком. Работа входных и выходных каскадов разрешается при разрешении работы модуля I2C (установлен бит SMBCTL2.ENABLE).

17.2.2 Регистр адреса и компаратор

В регистр адреса SMBADDR может быть записан 7-битный адрес, на который I2C отвечает, когда является ведомым (Slave) устройством на шине. Установка старшего бита SAEN регистра SMBADDR разрешает распознавание адреса ведомого (Slave).

Компаратор адреса сравнивает принятый 7-битный адрес с адресом, записанным в регистр SMBADDR. При разрешении общего вызова, когда установлен бит SMBCTL1.GCMEN, первый принятый байт сравнивается с величиной 00_{16}.

В режиме Alert-отклика, если установлен бит SMBCTL1.SMBARE, первый принятый байт сравнивается с величиной 0001100_2.

При 10-битной адресации (одновременно установлены биты SMBADDR.SAEN и SMBCTL3.S10EN) старшие 5 бит первого принятого адреса сравниваются с величиной 11110_2, младшие 2 бита принятого адреса сравниваются с величиной, записанной в SMBCTL3S10.2 – SMBCTL3S10.1. Старший бит второго принятого адреса сравнивается с SMBCTL3.SA10.0, младшие 7 бит - с SMBADDR.ADDR.

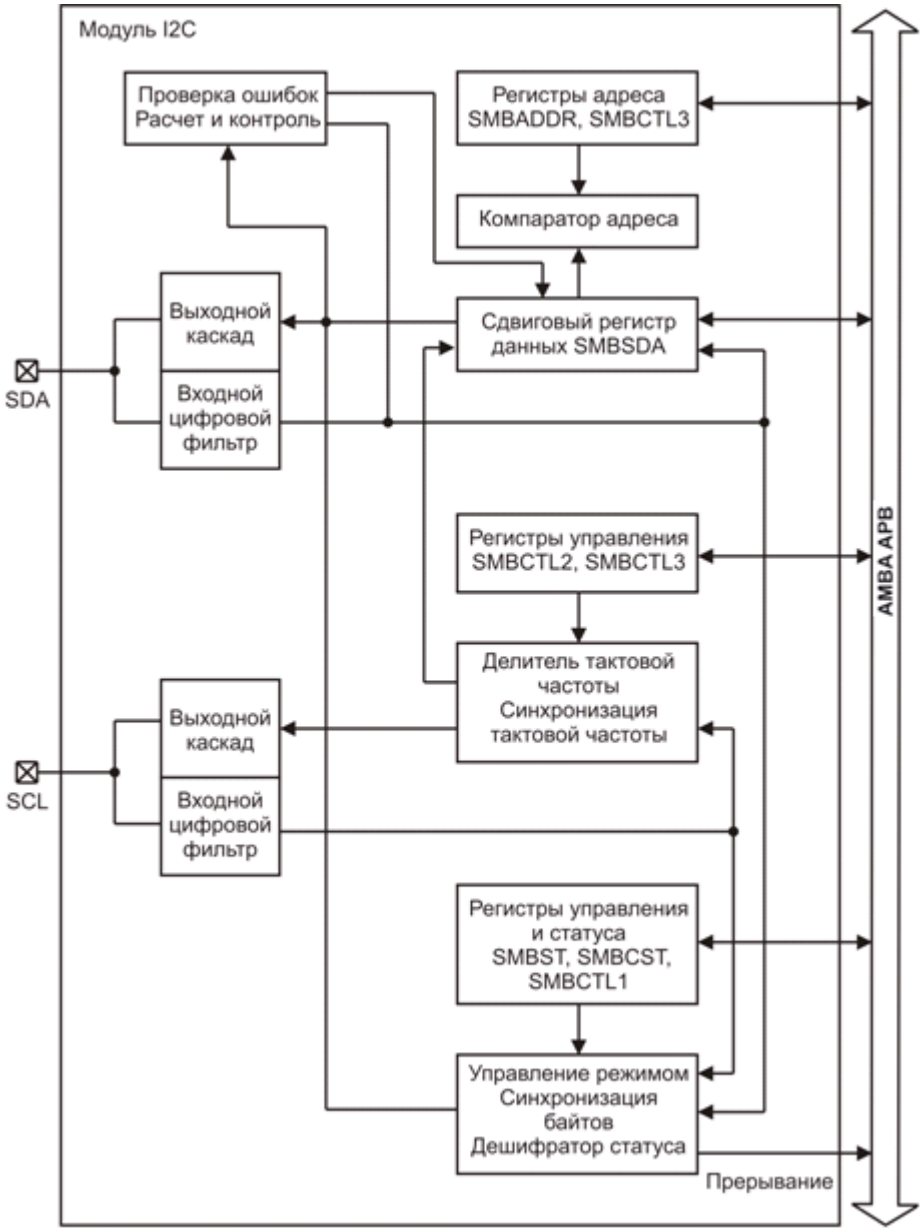


Рисунок 375 – Строение модуля I2C

17.2.3 Сдвиговый регистр данных

Сдвиговый регистр данных SMBSDA содержит байт данных, который может быть как принятым байтом данных, так и байтом данных, который необходимо передать. Данные одновременно принимаются и передаются, следовательно, SMBSDA всегда содержит последний байт данных, которые передавались по шине. Поэтому даже если произошла потеря приоритета, то корректные данные будут сохраняться в SMBSDA до момента окончания операции с данными. Данные передаются по заднему фронту SCL (первым передается MSB), принимаются – по переднему фронту SCL.

Процесс распределения приоритетов проходит во время передачи адреса, формирования состояний «Старт» и «Стоп». Если принятые данные отличаются от переданных, например, передана «1», а принят «0», то приоритет будет потерян.

17.2.4 Генерация тактового сигнала и синхронизация

Последовательный тактовый сигнал, выходной сигнал I2C в режиме ведущего (Master), формируется генератором из системного тактового сигнала.

В режиме F/S используется 7-битный делитель. Значение старших 6 бит определяется полем SMBCTL2.SCLFRQ, младший бит всегда равен нулю (деление производится только на четное число). Минимальный коэффициент деления равен восьми, максимальный – 128.

Блок синхронизации тактового сигнала производит синхронизацию генератора тактового сигнала и выходного сигнала SCL с тактовым сигналом других устройств, подключенных к шине.

В высокоскоростном режиме (Hs) коэффициент деления определяется величиной, записанной в поле SMBCTL3.HSCDIV. Делитель уменьшен до 4 бит, значение младшего бита равно нулю.

17.2.5 Управление режимом и статус

Флаг прерывания находится в регистре SMBST. Также этот регистр содержит биты, показывающие режим работы I2C (ведущий (Master) или ведомый (Slave), приемник или передатчик).

Регистр SMBCST является регистром управления и статуса, он показывает статус шины, возврат после ошибки.

Регистр SMBCTL1 управляет формированием состояний «Старт», «Повторный старт», «Стоп», а также сигнала подтверждения приема данных.

В регистрах SMBCTL2 и SMBCLT3 содержатся коэффициенты деления тактового сигнала, которые используются при работе I2C в режиме ведущего (Master). Также регистр SMBCLT3 управляет 10-битной адресацией.

17.3 Функционирование

I2C поддерживает следующие режимы работы:
- режим «Неадресованный ведомый (Slave)»;
- режим «Ведущий (Master) передатчик»;
- режим «Ведущий (Master) приемник»;
- режим «Ведомый (Slave) приемник»;
- режим «Ведомый (Slave) передатчик».

17.3.1 Инициализация

Для начала работы шины I2C требуется произвести инициализацию. Для этого необходимо выполнение нижеследующих действий:

1) Разрешить работу I2C установкой бита SMBCTL2.ENABLE.

2) При работе в режиме ведущего (Master) для выбора периода тактового сигнала SCL необходимо записать нужный коэффициент деления в поле

SMBCTL2.SCLFRQ. В высокоскоростном режиме (Hs) необходимо записать коэффициент деления в поле SMBCTL3.HSDIV. При работе в режиме ведомого (Slave) значения в полях SMBCTL2.SCLFRQ и SMBCTL3.HSDIV не учитываются.

3) Для работы в режиме ведомого, настроить ведомое для адреса, по которому данное устройство должно ответить:

а) Для работы в режиме ведомого (Slave) необходимо записать в регистр SMBADDR значение адреса ведомого (Slave) и разрешить распознавание адреса установкой бита SMBADDR.SAEN.

б) Для разрешения расширенной 10-битной адресации записать значение старших разрядов адреса в поле SMBCTL3.S10AD и установить бит SMBCTL3.S10EN.

в) Установить бит SMBCTL1.GCMEN для разрешения распознавания адреса общего вызова (дополнительная функция).

г) Установить бит SMBCTL1.SMBARE для распознавания адреса отклика (дополнительная функция).

4) Если установлены требования к времени ожидания шины SMBus – записать соответствующую величину в биты SMBCST.TOCDIV и в регистр SMBTOPR для проверки времени активной работы линии SCL (максимум 25 мс). Запись в биты SMBCST.TOCDIV величины, отличной от нуля, автоматически разрешает проверку времени ожидания шины.

5) Для разрешения прерывания I2C установить бит SMBCTL1.INTEN.

17.3.2 Режим «Неадресованный ведомый (Slave)»

Режим «Неадресованный ведомый (Slave)» (общий режим) является режимом работы по умолчанию SMBST.MODE = IDLE (00_{16}). После разрешения работы I2C шина начинает работать в режиме «Неадресованный ведомый (Slave)». I2C непрерывно отслеживает состояние шины и при формировании состояния «Старт» или «Повторный старт» переходит в режим «Ведомый (Slave) приемник». Из этого режима есть возможность перейти в режим «Ведущий (Master) передатчик» после успешного формирования состояния «Старт».

I2C возвращается в режим «Неадресованный ведомый (Slave)» при выполнении следующих условий:

- состояние «Старт» не было успешно сформировано, так как другое устройство удерживает линию SCL в низком уровне;
- произошла потеря приоритета при передаче пакета данных в режиме «Ведущий (Master) передатчик» или при передаче бита R/W# в режиме «Ведущий (Master) приемник»;
- произошла потеря приоритета во время отклика;
- отправлен инверсный сигнал подтверждения в режиме «Ведомый (Slave) приемник» (было запрещено распознавание адреса или адрес ведомого (Slave) не совпал);
- получен инверсный сигнал подтверждения в режиме «Ведомый (Slave) передатчик»;
- сформировано состояние «Стоп»;
- ошибочное состояние на шине;
- сброс I2C;

- запрещение работы I2C.

17.3.3 Режим «Ведущий (Master) передатчик»

Переход в режим «Ведущий (Master) передатчик»

1) Переход в режим «Ведущий (Master) передатчик» производится после формирования сигнала «Старт». Первый байт, следующий после формирования сигнала «Старт», передается ведущим (Master) устройством и содержит адрес ведомого (Slave) и бит направления передачи.

2) В зависимости от значения бита направления передачи R/W#, I2C может как остаться в режиме «Ведущий (Master) передатчик», так и перейти в режим «Ведущий (Master) приемник», если R/W# = 1_2. Вместо последующей передачи адреса ведомого (Slave), I2C может передать адрес ведущего (Master) ($00001xxx_2$) для перехода в высокоскоростной режим (Hs).

3) I2C может перейти в режим ведущего (Master) с помощью установки бита SMBCTL1.START. I2C проверяет, свободна ли шина (бит SMBCST.BB = 0_2), и формирует состояние «Старт». Если шина занята (SMBCST.BB = 1_2), то состояние «Старт» будет сформировано в тот момент, когда шина будет свободна и сигнал на линии SCL будет в высоком уровне.

4) При успешном формировании состояния «Старт», бит SMBCTL1.START станет равным нулю, будет установлен флаг SMBST.INT и линия SCL будет удерживаться в низком уровне, пока флаг не будет сброшен (SMBST.MODE = 01_{16} – код статуса шины STDONE). Прерывание будет сгенерировано в случае, если установлен бит SMBCTL1.INTEN.

Передача адреса и данных

1) Если бит SMBST.INT установлен, необходимо программно записать адрес ведомого (Slave) и направление передачи в сдвиговый регистр данных SMBSDA.

2) После записи в регистр SMBSDA необходимо сбросить флаг прерывания SMBST.INT – очистить бит SMBCTL1.CLRST.

3) Линия SCL будет освобождена после сброса флага SMBST.INT и времени установки данных. Затем содержимое регистра SMBSDA будет передано на шину.

4) После передачи всех битов данных и получения сигнала подтверждения (после девятого тактового импульса), бит подтверждения анализируется системой, биты SMBST.MODE показывают код статуса шины I2C в зависимости от принятого адреса.

5) В течение передачи данных на линиях SDA и SCL отслеживаются потенциальные конфликты с другими устройствами, подключенными к шине. Если зафиксирован конфликт, передача данных прекращается. Код статуса шины определяется как SRAAPA (SMBST.MODE = 11_H), если потерян приоритет в режиме «Ведомый (Slave) приемник» или как IDLARL (SMBST.MODE = 03_H), если потерян приоритет в режиме «Неадресованный ведомый (Slave)».

6) Если бит направления передачи R/W# = 1_2, I2C переходит в режим «Ведущий (Master) приемник», если не произошло ошибок.

7) Если выбрано требуемое направление передачи и передача адреса ведомого (Slave) успешно завершена (код статуса шины не MTADNA SMBST.MODE = 05_{16} и не BERROR SMBST.MODE = $1F_{16}$), то устанавливается флаг SMBST.INT, показывающий необходимость записи первого байта данных в регистр SMBSDA. Пока установлен флаг SMBST.INT, SCL удерживается в низком уровне, прерывание будет сгенерировано, если установлен бит SMBCTL1.INTEN.

8) В случае, когда после передачи данных ведомый (Slave) приемник отправляет инверсный сигнал подтверждения, на SCL будет удерживаться низкий уровень, прерывание будет сгенерировано, если установлен бит SMBCTL1.INTEN (код статуса шины MRDANA SMBST.MODE = $0B_{16}$).

Отслеживание ошибок в пакетах данных

При работе I2C в режиме «Ведущий (Master) передатчик», показанный в таблице 154 и на рисунке 376, после установки бита SMBCST.PECNEXT произойдет передача содержимого регистра Packet Error Checking в регистр SMBSDA и начало передачи байта Cyclic Redundancy Check (CRC) для ведомого (Slave) устройства. Это произойдет после передачи последнего байта данных и до того, как будут сформированы состояния «Стоп» и «Повторный старт».

Таблица 154 – Режим F/S «Ведущий (Master) передатчик», действия и статус

Код	Название	Описание	Данные		Управление SMBCTL1				Следующее действие
			R/W	SMBSDA	clrst	ask	stop	start	
1	2	3	4	5	6	7	8	9	10
01_{16}	STDONE	Сформировано состояние «Старт»	W	Адрес ведущего (Master)	1	0	0	0	Передача кода ведущего (Master), переход в режим Hs
				SADR/ RW = 0_2					Передача SADR/RW – ACK или NACK от ведомого (Slave)
02_{16}	RSDONE	Сформировано состояние «Повторный Старт»	W	SADR/ RW = 0_2	1	0	0	0	Передача SADR/RW – ACK или NACK от ведомого (Slave)
				SADR/ RW = 1_2					Передача SADR/RW – ACK или NACK от ведомого (Slave), I2C переходит в режим «Ведущий (Master)

									приемник»

Окончание таблицы 154

	1	2	3		4	5	6	7	8
03_{16}	IDLARL	Потеря приоритета, переход в режим «Неадресованный ведомый (Slave)»	–		1	0	0	0	Режим «Неадресованный ведомый (Slave)»
04_{16}	MTADPA	Отправлен адрес ведомого (Slave), сигнал ACK	W	Данные	1	0	0	0	Передача байта данных
			–		1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» и последующий «Повторный старт»
05_{16}	MTADNA	Отправлен адрес ведомого (Slave), сигнал NACK	–		1	0	0	1	«Повторный старт»
					1	0	1	0	Состояние «Стоп»
					1	0	1	1	«Стоп» и последующий «Повторный старт»
06_{16}	MTDAPA	Отправлен байт данных, сигнал ACK	W	Данные	1	0	0	0	Передача байта данных
			–		1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» и последующий «Старт»
07_{16}	MTDANA	Отправлен байт данных, сигнал NACK	–		1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» и последующий

								«Повторный старт»

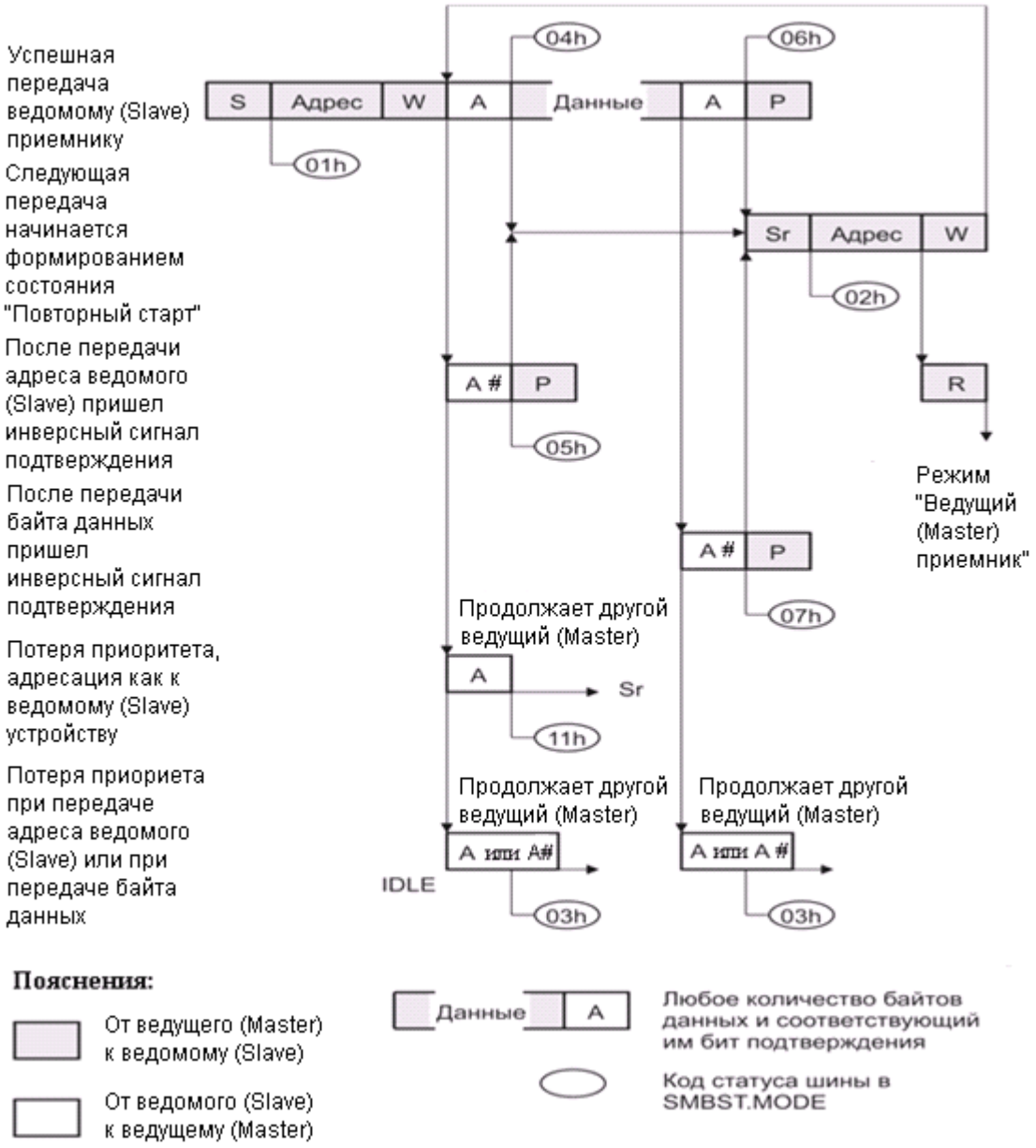


Рисунок 376 – Режим F/S «Ведущий (Master) передатчик», действия и статус

Повторный старт

I2C при работе в режиме «Ведущий (Master) передатчик» осуществляет полный контроль над шиной и может адресовать другому ведомому (Slave) изменить направление передачи без потери контроля над шиной.

Для формирования состояния «Повторный старт» необходимо:

- установить бит SMBCTL1.START;

- в режиме «Ведущий (Master) приемник» считать последний элемент данных из регистра SMBSDA;

- сбросить ожидаемый флаг прерывания установкой бита SMBCTL1.CLRST;

- впоследствии линия SCL будет освобождена, будет сгенерировано состояние «Повторный старт» и прерывание I2C (код статуса шины RSDONE).

Выход из режима «Ведущий (Master) передатчик»

Выход из режима «Ведущий (Master) передатчик» осуществляется формированием сигнала «Стоп». Чтобы сформировать сигнал «Стоп» необходимо:

- установить бит SMBCTL1.STOP;

- в режиме «Ведущий (Master) приемник» считать последний элемент данных из регистра SMBSDA;

- сбросить ожидаемый флаг прерывания установкой бита SMBCTL1.CLRST.

Выполнение данных условий вызывает немедленное формирование I2C состояния «Стоп» и обнуление бита SMBCTL1.STOP. Состояние «Стоп» может быть сформировано только в том случае, когда I2C является активным ведущим (Master) устройством шины (SMBST.MODE = 01_{16} – $0B_{16}$).

Высокоскоростной режим (Hs) «Ведущий (Master) передатчик»

Вход в высокоскоростной режим (Hs) «Ведущий (Master) передатчик» осуществляется формированием состояния «Старт» с последующей передачей адреса ведущего (Master) ($00001xxx_2$) вместо адреса ведомого (Slave). После того, как адрес ведущего (Master) будет передан, установится флаг SMBST.INT, сгенерируется прерывание (если был установлен бит SMBCTL1.INTEN). После успешной передачи адреса ведущего (Master) код статуса шины станет HMTMOK. В этот момент I2C входит в высокоскоростной режим (Hs) «Ведущий (Master) передатчик», показанный в таблице 155 и на рисунке 377.

Необходимо сформировать состояние «Повторный старт» установкой бита SMBCTL1.START и сбросить флаг прерывания установкой бита SMBCTL1.CLRST.

После формирования сигнала «Повторный старт» установится флаг SMBST.INT, код статуса в SMBST.MODE станет HRSDONE. Затем последует передача адреса и данных (см. раздел 17.3.3.2).

Таблица 155 – Высокоскоростной режим (Hs) «Ведущий (Master) передатчик», действия и статус

Код	Название	Описание	Данные		Управление: SMBCTL1				Следующее действие
			R/W	SMBSDA	clrst	ask	stop	start	
1	2	3	4	5	6	7	8	9	10
$0C_{16}$	MTMCER	Передан код ведущего (Master), найдена ошибка	–		1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» с последующим «Повторным стартом»

		(сигнал ACK)							

Продолжение таблицы 155

1	2	3	4	5	6	7	8	9	10
21_{16}	HMTMCOK	Адрес ведущего (Master) успешно передан, переход в высокоскоростной режим (Hs)	–		1	0	0	1	«Повторный старт»
22_{16}	HRSDONE	Сформирован «Повторный старт»	W	SADR/RW = 0	1	0	0	0	Передача SADR/RW – сигнал ACK или NACK от ведомого (Slave)
				SADR/RW = 0					Передача SADR/RW – сигнал ACK или NACK от ведомого (Slave), I2C перейдет в высокоскоростной режим (Hs) «Ведущий (Master) приемник»
23_{16}	HIDLARL	Потеря приоритета, переход в высокоскоростной режим (Hs) «Неадресованный ведомый (Slave)»	–		1	0	0	0	Режим «Неадресованный ведомый (Slave)»
24_{16}	HMTADPA	Отправлен адрес ведомого (Slave), сигнал	W	Данные	1	0	0	0	Передача байта данных
			–		1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» и последу-

		ACK						ющий «Повторный старт»

Окончание таблицы 155

1	2	3	4	5	6	7	8	9	10
25_{16}	HMTADNA	Отправлен адрес ведомого (Slave), сигнал NACK	–		1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» и последующий «Повторный старт»
26_{16}	HMTDAPA	Отправлен байт данных, сигнал ACK	W	Данные	1	0	0	0	Передача байта данных
			–		1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» и последующий «Повторный старт»
27_{16}	HMTDANA	Отправлен байт данных, сигнал NACK	–		1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» и последующий «Повторный старт»

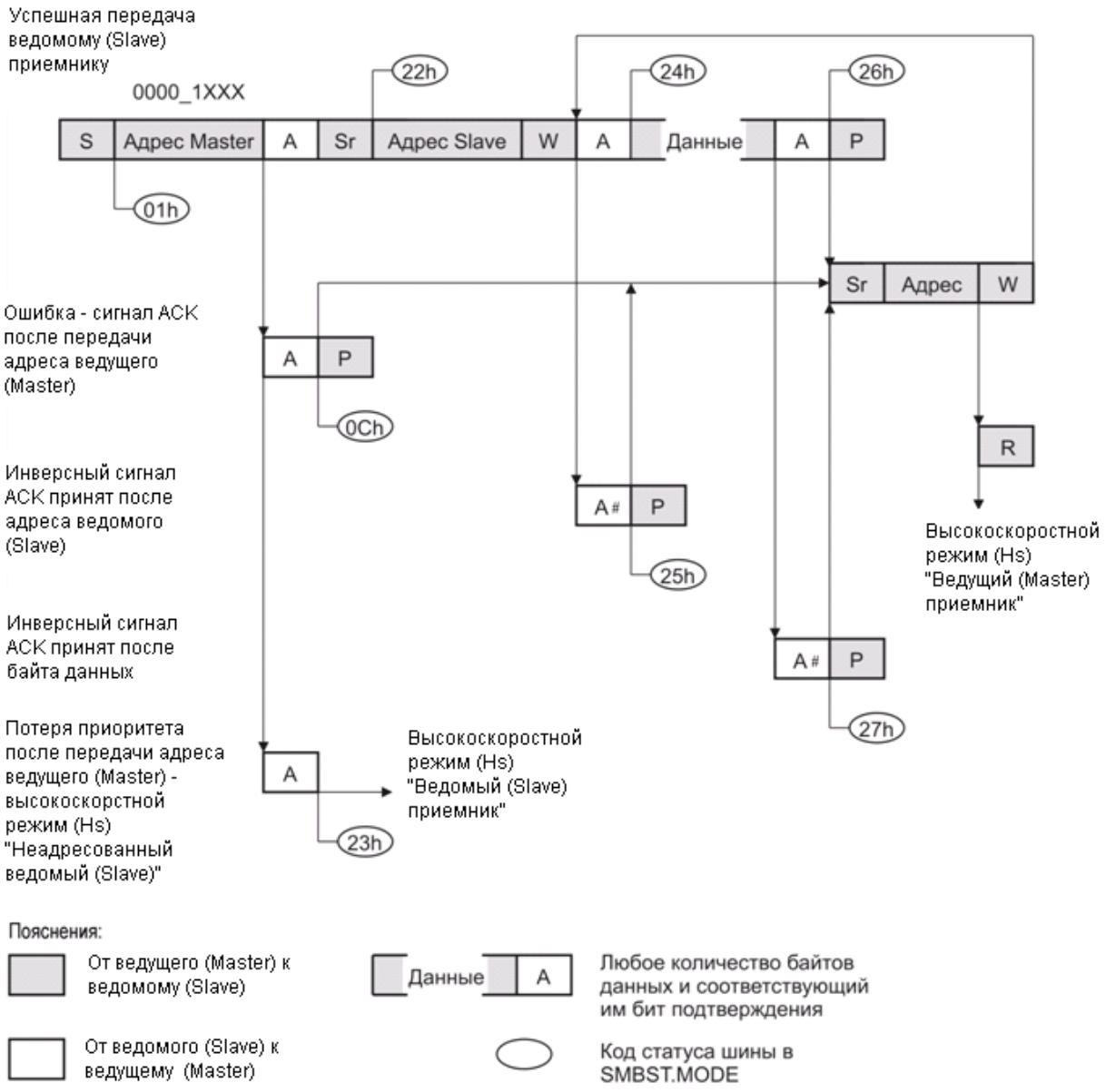


Рисунок 377 – Высокоскоростной режим (Hs)
«Ведущий (Master) передатчик», действия и статус

17.3.4 Режим «Ведущий (Master) приемник»

Вход в режим «Ведущий (Master) приемник» осуществляется после успешной передачи адреса ведомого (Slave) с битом направления передачи, равным единице R/W# = 1_2. В режиме «Ведущий (Master) приемник» I2C считывает с ведомого (Slave) устройства, подключенного к шине, и, тем не менее, продолжает контролировать линию SDA. I2C продолжает генерировать импульсы SCL и подтверждать каждый принятый байт данных.

После каждого принятого байта устанавливается бит SMBST.INT и данные программно считываются из регистра SMBSDA. Линия SCL удерживается в низком уровне, пока установлен флаг SMBST.INT. Установка бита SMBCTL1.CLRST очищает бит SMBST.INT, разрешая прием следующего байта данных.

Протокол шины устанавливает, что состояния «Повторный старт» или «Стоп» не должны формироваться, пока работа ведется в режиме «Ведущий (Master) приемник» и I2C уже не является единственным устройством, контролирующем SDA. Для восстановления полного контроля над шиной в соответствии с протоколом шины ведущему (Master) приемнику необходимо отправить инверсный сигнал подтверждения после приема последнего байта данных.

Для этого необходимо очистить бит SMBCTL1.ACK до сброса флага SMBST.INT во время приема предпоследнего байта. В то же самое время должен быть установлен бит SMBCST.PECNEXT, если есть запрос на PEC. Когда флаг SMBST.INT будет сброшен, линия SCL станет свободной и будет принят последний байт данных, в течение девятого тактового импульса SCL будет отправлен инверсный сигнал ACK. Затем I2C вернется в режим «Ведущий (Master) передатчик» и сможет сформировать на шине состояния «Повторный старт» или «Стоп». Работа режима «Ведущий (Master) приемник» показана в таблице 156 и на рисунке 378.

Проверка ошибок пакета данных

При разрешении проверки ошибок пакета данных PEC, последний байт, принятый от передающего ведомого (Slave) устройства, будет байт PEC. Если результат вычислений CRC отличен от нуля, то будет установлен бит SMBCST.PECFAULT, указывающий на ошибку.

Таблица 156 – Режим F/S «Ведущий (Master) приемник»

Код	Название	Описание	Данные		Управление: SMBCTL1				Следующее действие
			R/W	SMBSDA	clrst	ask	stop	start	
1	2	3	4	5	6	7	8	9	10
08_{16}	MRADPA	Передан адрес ведомого (Slave), сигнал ACK	–		1	0	0	0	Принят байт данных, сформирован сигнал ACK
					1	1	0	0	Принят байт данных, сформирован сигнал NACK
09_{16}	MRADNA	Передан адрес ведомого (Slave), сигнал NACK	–		1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» с последующим «Повторным стартом»
$0A_{16}$	MRDAPA	Принят байт данных, сигнал ACK	R	Данные	1	0	0	0	Прием байта данных, формирование сигнала ACK
					1	1	0	0	Прием байта данных, формирование сигнала NACK

Окончание таблицы 156

1	2	3	4	5	6	7	8	9	10
$0B_{16}$	MRDANA	Принят байт данных, сигнал NACK	R	Данные	1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» с последующим «Повторным стартом»

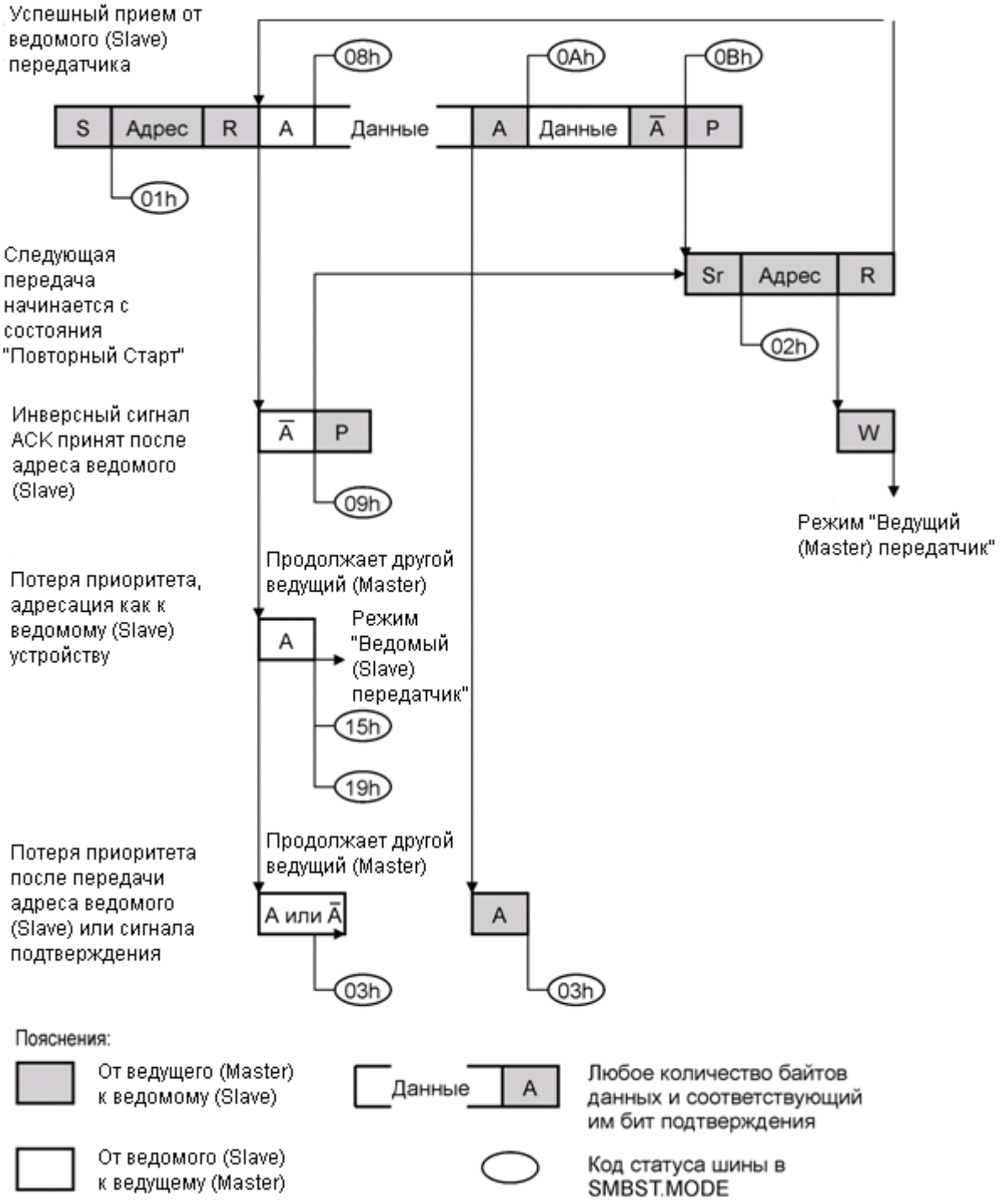


Рисунок 378 – Режим F/S «Ведущий (Master) приемник»

Высокоскоростной режим (Hs) «Ведущий (Master) приемник»

Вход в высокоскоростной режим (Hs) «Ведущий (Master) приемник» осуществляется, если после передачи адреса ведущего (Master) следует адрес ведомого (Slave) с битом направления передачи, равным единице (R/W# = 1_2), а затем формируется состояние «Повторный старт». После входа I2C в высокоскоростной режим (Hs) «Ведущий (Master) приемник» устанавливается флаг SMBST.INT, коды статуса шины показаны в таблице 157 и на рисунке 379.

Таблица 157 – Высокоскоростной режим (Hs) «Ведущий (Master) приемник», действия и статус

Код	Название	Описание	Данные		Управление: SMBCTL1				Следующее действие
			R/W	SMBSDA	clrst	ask	stop	start	
1	2	3	4	5	6	7	8	9	10
28_{16}	HMRADPA	Передан адрес ведомого (Slave), сигнал ACK	–		1	0	0	0	Принят байт данных, сформирован сигнал ACK
					1	1	0	0	Принят байт данных, сформирован сигнал NACK
29_{16}	HMRADNA	Передан адрес ведомого (Slave), сигнал NACK	–		1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» с последующим «Повторным стартом»
$2A_{16}$	HMRDAPA	Принят байт данных, сигнал ACK	R	Данные	1	0	0	0	Прием байта данных, формирование сигнала ACK
					1	1	0	0	Прием байта данных, формирование сигнала NACK
$2B_{16}$	HMRDANA	Принят байт данных, сигнал NACK	R	Данные	1	0	0	1	«Повторный старт»
					1	0	1	0	«Стоп»
					1	0	1	1	«Стоп» с последующим «Повторным стартом»

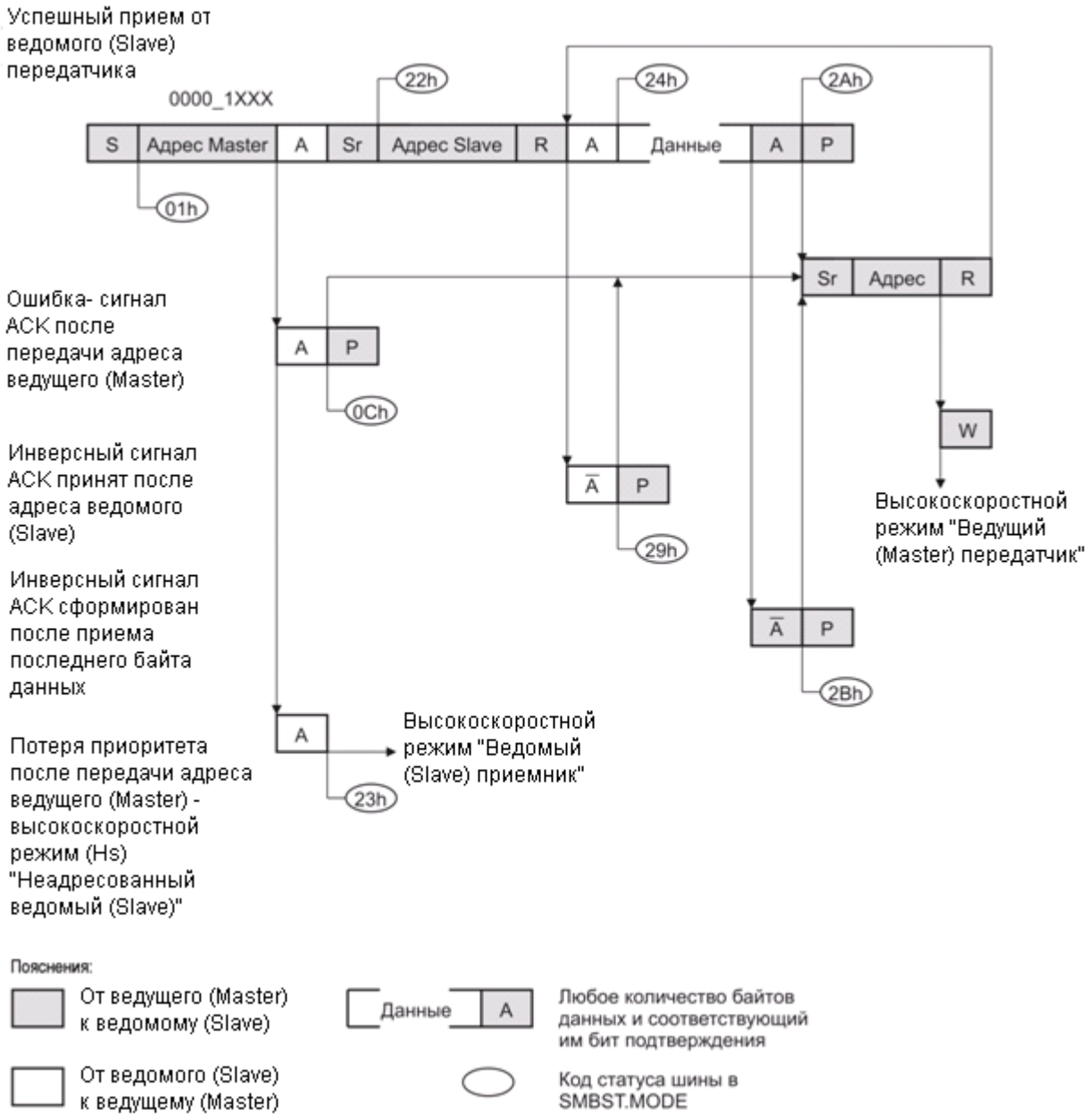


Рисунок 379 – Высокоскоростной режим (Hs) «Ведущий (Master) приемник»

17.3.5 Режим «Ведомый (Slave) приемник»

В режиме «Ведомый (Slave) приемник» данные принимаются от ведущего (Master) передатчика. Сигнал подтверждения формируется после приема каждого байта.

Вход в режим «Ведомый (Slave) приемник»

После разрешения на работу I2C продолжает контролировать шину. После обнаружения состояния «Старт», I2C входит в режим «Ведомый (Slave) приемник», показанный в таблице 158 и на рисунке 380, и начинает прием 7-битного адреса и бита направления передачи R/W# от ведущего (Master). Переход в режим

«Ведомый (Slave) приемник» также может произойти из режима «Ведущий (Master) передатчик» в случае потери приоритета при передаче адреса или данных.

После приема полного адреса I2C сравнивает принятый адрес:

- с полем SMBADDR.ADDR, если установлен бит SMBADDR.SAEN;
- с адресом общего вызова (00_{16}), если установлен бит SMBCTL1.GCMEN;
- с адресом отклика (0001100_2), если установлен бит SMBCTL1.SMBARE.

Сигнал подтверждения формируется в течение девятого тактового импульса на SCL, если принятый адрес совпал с адресом, записанным в регистр, адресом общего вызова или адресом отклика. После того, как произошло совпадение, устанавливается флаг SMBST.INT, изменяется значение SMBST.MODE. Если установлен бит SMBCTL1.INTEN, то генерируется прерывание. В регистре SMBSDA содержится принятый байт, включающий адрес ведомого (Slave) и бит направления передачи.

В зависимости от принятого бита направления передачи I2C может перейти в режим «Ведомый (Slave) передатчик» R/W# = 1_2 или остаться в режиме «Ведомый (Slave) приемник» R/W# = 0_2.

После каждого принятого байта данных устанавливается флаг SMBST.INT, показывая необходимость считать данные из регистра SMBSDA, линия SCL удерживается в низком уровне. Необходимо программно считать принятый байт данных, а затем установить бит SMBCTL1.CLRST, чтобы сбросить флаг SMBST.INT и освободить линию SCL.

Таблица 158 – Режим F/S «Ведомый (Slave) приемник», действия и статус

Код	Название	Описание	Данные		Управление: SMBCTL1				Следующее действие
			R/W	SMBSDA	clrst	ask	stop	start	
1	2	3	4	5	6	7	8	9	10
10_{16}	SRADPA	Принят адрес ведомого (Slave), сигнал ACK	–		1	0	0	0	Прием байта данных, формирование сигнала ACK
					1	1	0	0	Прием байта данных, формирование сигнала NACK
11_{16}	SRAAPA	Принят адрес ведомого (Slave) после потери приоритета, сигнал ACK	–		1	0	0	0	Прием байта данных, формирование сигнала ACK
					1	1	0	0	Прием байта данных, формирование сигнала NACK

Окончание таблицы 158

1	2	3	4	5	6	7	8	9	10
12_{16}	SRDAPA	Принят байт данных, сигнал ACK	R	Данные	1	0	0	0	Прием байта данных, формирование сигнала ACK
					1	1	0	0	Прием байта данных, формирование сигнала NACK
13_{16}	SRDANA	Принят байт данных, сигнал NACK	R	Данные	1	0	0	0	Режим «Неадресованный ведомый (Slave)»
					1	0	0	1	Режим «Неадресованый ведомый (Slave)», формирование сигнала «Старт» после того, как шина станет свободна
$1C_{16}$	SSTOP	Режим ведомого (Slave), обнаружено состояние «Стоп»	–		1	0	0	0	Режим «Неадресованный ведомый (Slave)»
					1	0	0	1	Режим «Неадресованый ведомый (Slave)», формирование сигнала «Старт» после того, как шина станет свободна
$1D_{16}$	SGADPA	Принят адрес общего вызова, сигнал ACK	–		1	0	0	0	Прием байта данных, формирование сигнала ACK
					1	1	0	0	Прием байта данных, формирование сигнала NACK
$1E_{16}$	SGAAPA	Принят адрес общего вызова после потери приоритета, сигнал ACK	–		1	0	0	0	Прием байта данных, формирование сигнала ACK
					1	1	0	0	Прием байта данных, формирование сигнала NACK

Рисунок 380 – Режим F/S «Ведомый (Slave) приемник», действия и статус

Режим «Ведомый (Slave) приемник» – 10-битная адресация

Для использования внешней (10-битной) адресации необходимо установить бит SMBCTL2.S10EN и бит SMBADDR.SAEN. Старшие биты адреса необходимо записать в поле SMBCTL3.S10AD. После формирования сигнала «Старт» и приема первого байта, I2C сравнивает содержимое сдвигового регистра данных с величиной $11110yy_2$, где yy являются двумя наиболее значащими битами адреса ведомого (Slave), хранимыми в двух старших битах поля SMBCTL3.S10AD. Бит направления передачи должен быть равен нулю R/W# = 0_2, показывая ведомому (Slave) устройству то, что будет принята вторая часть 10-битного адреса.

Если две величины равны, то I2C формирует сигнал подтверждения и продолжает принимать младшие биты адреса. После приема первого байта не генерируется прерывание.

Ведущий (Master) передатчик затем передает второй байт 10-битного адреса ведомого (Slave). Старший бит принятого адреса сравнивается с младшим битом поля SMBCTL3.S10AD, а младшие 7 бит принятого адреса сравниваются с величиной, определяемой полем SMBADDR.SADDR. Если величины равны, то формируется сигнал подтверждения, при этом установится флаг SMBST.INT, код статуса шины станет соответственно SRADPA или SRAPAA.

Проверка ошибок пакета данных

При разрешенной проверке ошибок пакета данных PEC последний байт, принятый от ведущего (Master), будет байтом PEC. Если результат вычислений CRC не равен нулю, будет установлен бит SMBCST.PECFAULT, указывая на произошедшую ошибку, будет передан инверсный сигнал подтверждения. Необходимо точно знать количество байтов, передаваемых ведущим (Master) устройством шины, и при чтении предпоследнего байта из регистра SMBSDA необходимо установить бит SMBCST.PECNEXT, а затем установить SMBCTL1.CLRST.

Выход из режима «Ведомый (Slave) приемник»

Если ведомый (Slave) приемник больше не может принимать данные от ведущего (Master), необходимо установить бит SMBCTL1.ACK до того, как будет установлен бит SMBCTL1.CLRST, сбрасывающий флаг SMBST.INT. Это послужит причиной тому, что после принятия последнего байта данных от ведущего (Master) будет сформирован инверсный сигнал подтверждения. После принятия последнего байта данных установится флаг SMBST.INT. Затем необходимо программно считать последний принятый байт из регистра SMBSDA и записать единицу в бит SMBCTL1.CLRST, чтобы сбросить флаг прерывания SMBST.INT и освободить шину SCL.

Высокоскоростной режим (Hs) «Ведомый (Slave) приемник»

Вход в высокоскоростной режим (Hs) «Ведомый (Slave) приемник» осуществляется после приема адреса ведущего (Master) ($00001xxx_2$). После адреса ведущего (Master) следует состояние «Повторный старт» и адрес ведомого (Slave), включающий бит направления, равный нулю R/W# = 0_2. После приема адреса ведомого (Slave), I2C производит сравнение адресов, показанное на рисунке 381. Более подробные действия указаны в таблице 159.

Таблица 159 – Высокоскоростной режим (Hs) «Ведомый (Slave) приемник»

Код	Название	Описание	Данные		Управление: SMBCTL1				Следующее действие
			R/W	SMBSDA	clrst	ask	stop	start	
1	2	3	4	5	6	7	8	9	10
30_{16}	HSRADPA	Принят адрес ведомого (Slave), сигнал ACK	–		1	0	0	0	Прием байта данных, формирование сигнала ACK
					1	1	0	0	Прием байта данных, формирование сигнала NACK

Окончание таблицы 159

1	2	3	4	5	6	7	8	9	10
32_{16}	HSRDAPA	Принят байт данных, сигнал ACK	R	Данные	1	0	0	0	Прием байта данных, формирование сигнала ACK
					1	1	0	0	Прием байта данных, формирование сигнала NACK
33_{16}	HSRDANA	Принят байт данных, сигнал NACK	R	Данные	1	0	0	0	«Неадресованный ведомый (Slave)»
					1	0	0	1	«Неадресованный ведомый (Slave)», «Старт» после освобождения шины

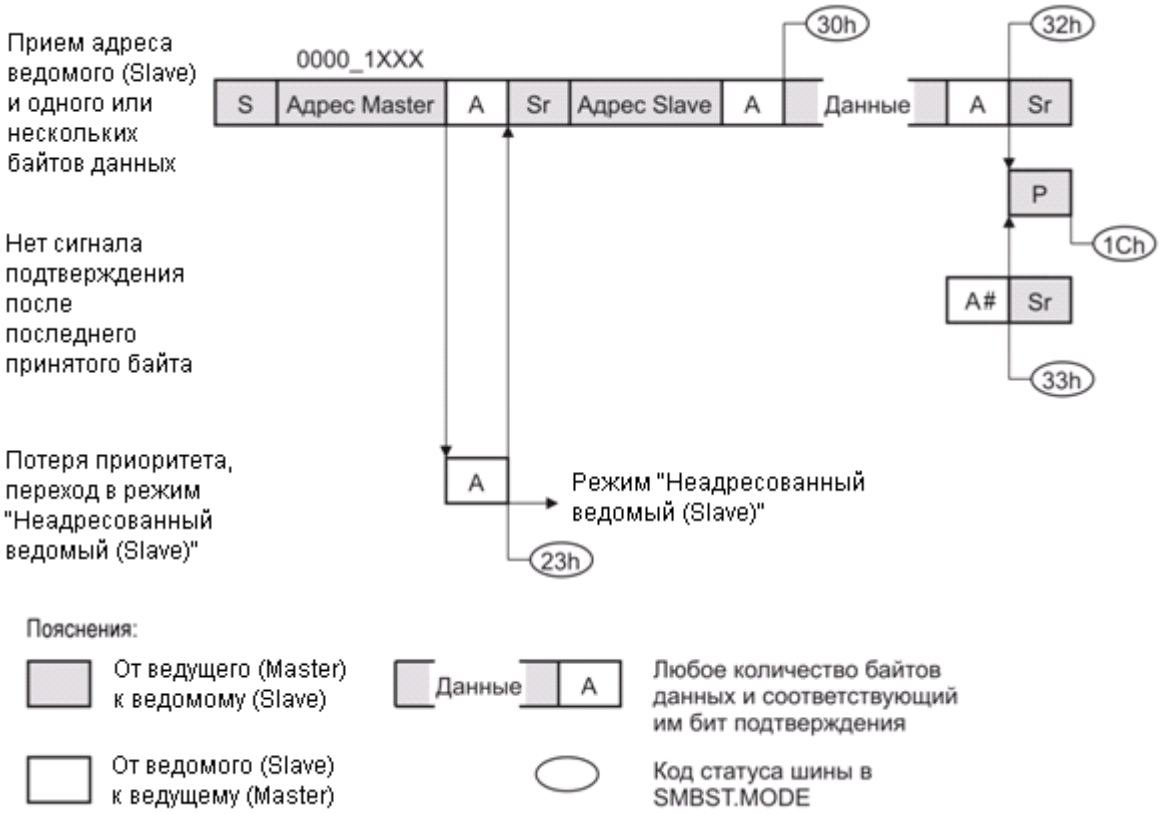


Рисунок 381 – Высокоскоростной режим (Hs) «Ведомый (Slave) приемник»

17.3.6 Режим «Ведомый (Slave) передатчик»

В режиме «Ведомый (Slave) передатчик» любое количество байтов данных может быть передано от I2C ведущему (Master) приемнику. I2C проверяет бит подтверждения в течение девятого тактового импульса при передаче данных.

Вход в режим «Ведомый (Slave) передатчик»

Вход в режим «Ведомый (Slave) передатчик» осуществляется из режима «Ведомый (Slave) приемник» после адресации I2C «Ведущий (Master) передатчиком». Бит направления передачи, следующий после адреса ведомого (Slave), должен быть равным единице R/W# = 1_2. После этого установится флаг SMBST.INT, показывая необходимость записи данных в регистр SMBSDA. Линия SCL будет удерживаться в низком уровне, пока флаг SMBST.INT не будет сброшен. Когда первый байт передаваемых данных будет записан в регистр SMBSDA, необходимо программно установить бит SMBCTL1.CLRST, который сбросит флаг SMBST.INT. Затем линия SCL станет свободной, и будет передан байт данных.

Передача данных в этом режиме сходна с передачей в режиме «Ведомый (Slave) передатчик». Флаг SMBST.INT устанавливается после каждой успешно завершенной передачи данных, SMBST.MODE содержит соответствующий код статуса шины. Каждый следующий байт данных необходимо записывать в регистр SMBSDA, если SMBST.MODE не содержит код статуса шины STDANA, показывающий, что ведущий (Master) не может принять дополнительных данных.

Выход из режима «Ведомый (Slave) передатчик» может быть осуществлен только с помощью ведущего (Master) приемника. В этом случае ведущий (Master) приемник формирует инверсный сигнал подтверждения после последнего принятого байта. После обнаружения инверсного сигнала подтверждения I2C переходит в режим «Неадресованный ведомый (Slave)» (SMBST.MODE = IDLE) и ждет формирования на шине сигнала «Старт», показанные в таблице 160.

Таблица 160 – Режим F/S «Ведомый (Slave) передатчик»

Код	Название	Описание	Данные		Управление: SMBCTL1				Следующее действие
			R/W	SMBSDA	clrst	ask	stop	start	
1	2	3	4	5	6	7	8	9	10
14_{16}	STADTA	Принят адрес ведомого (Slave), сигнал ACK	W	Данные	1	X	0	0	Передача байта данных
15_{16}	STADPA	Принят адрес ведомого (Slave) после потери приоритета, сигнал ACK	W	Данные	1	X	0	0	Передача байта данных
16_{16}	STDAPA	Передача байта данных, сигнал ACK	W	Данные	1	X	0	0	Передача байта данных

Окончание таблицы 160

1	2	3	4	5	6	7	8	9	10
17_{16}	STDANA	Передача байта данных, сигнал NACK	–		1	X	0	0	«Неадресованный ведомый (Slave)»
					1	X	0	1	«Неадресованный ведомый (Slave)», «Старт» после того, как шина станет свободна
18_{16}	SATADP	Принят адрес отклика, сигнал ACK	W	Данные	1	X	0	0	Передача байта данных
19_{16}	SATAAP	Принят адрес отклика после потери приоритета, сигнал ACK	W	Данные	1	X	0	0	Передача байта данных
$1A_{16}$	SATDAP	Адресация с адресом отклика, передача байта данных, сигнал ACK	W	Данные	1	X	0	0	Передача байта данных
$1B_{16}$	SATDAN	Адресация с адресом отклика, передача байта данных, сигнал NACK	–		1	X	0	0	«Неадресованный ведомый (Slave)»
					1	X	0	1	«Неадресованный ведомый (Slave)», «Старт» после того, как шина станет свободна

Режим «Ведомый (Slave) передатчик» – 10-битная адресация

Для использования в режиме «Ведомый (Slave) передатчик» 10-битной адресации должен быть записан адрес ведомого (Slave) и разрешена 10-битная адресация (см. пункт «Режим «Ведомый (Slave) приемник» – 10-битная адресация» и рисунок 382).

Сначала I2C входит в режим «Ведомый (Slave) приемник», чтобы принять полный 10-битный адрес ведомого (Slave). Флаг SMBST.INT не будет установлен, линия SCL не будет удерживаться в низком уровне, статус кода шины в SMBST.MODE не изменится.

После формирования состояния «Повторный старт» последует второй байт адреса ведомого (Slave), а затем – повторная передача старших битов адреса с битом направления передачи, равным единице R/W# = 1_2. Если принятый адрес совпадет с адресом, записанным в регистрах, то будет установлен флаг SMBST.INT и I2C перейдет в режим «Ведомый (Slave) передатчик», код статуса шины в поле SMBST.MODE будет STADPA или STAAPA.

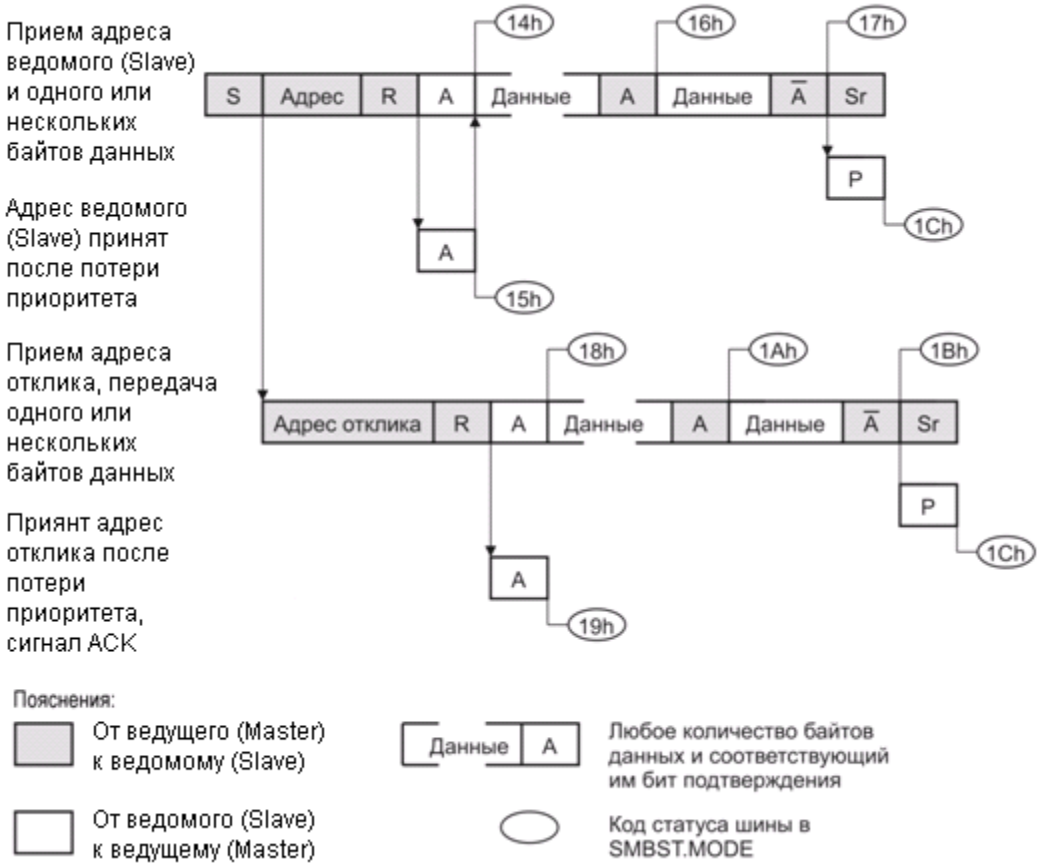


Рисунок 382 – Режим F/S «Ведомый (Slave) передатчик»

Проверка ошибок пакета данных

При разрешении проверки ошибок пакета данных PEC последний байт, переданный ведущим (Master) устройством, будет байтом PEC. Необходимо установить бит SMBCST.PECNEXT для записи байта PEC в регистр SMBSDA.

Режим Alert-отклика

Любой ведомый (Slave) (или несколько ведомых (Slave)) может послать сигнал запроса SMBAlert#, с целью передать данные к ведущему (Master). Ведущий (Master), получив сигнал SMBAlert#, в ответ инициализирует цикл отклика – выдает на шину специальный адрес отклика 0001100_2 с битом R/W#, равным 1_2 (чтение).

Устройство (устройства), пославшее сигнал SMBAlert# и получившее адрес отклика, выдает на шину свой 7-битный адрес (последний бит передаваемого байта может быть равным как 0_2, так и 1_2). Если несколько ведомых (Slave) выдают свои адреса на шину, то передачу осуществляет устройство, выигравшее арбитраж.

Ведущий (Master), который не послал адрес отклика (не инициализировал цикл отклика) в ответ на сигнал SMBAlert#, может сделать это через некоторое время.

Переход в режим Alert-отклика осуществляется после появления на шине SMBus адреса отклика 0001100_2, если установлен бит SMBCTL1.SMBARE.

Этот режим является единственным случаем, когда при функционировании I2C в качестве ведомого (Slave) используется арбитраж при передаче адреса.

Высокоскоростной режим (Hs) «Slave передатчик»

После передачи адреса ведущий (Master) $00001xxx_2$ осуществляется вход в высокоскоростной режим (Hs). Затем следует формирование состояния «Повторный старт», передача адреса ведомого (Slave) с битом направления передачи, равным единице R/W# = 1_2. После этого I2C переходит в режим «Ведомый (Slave) передатчик». Работа в высокоскоростном режиме (Hs) «Ведомый (Slave) передатчик» почти идентична работе в режиме F/S, отличие заключается только в более высокой скорости передачи данных и в кодах статуса шины, показанных в таблице 161 и на рисунке 383.

Таблица 161 – Высокоскоростной режим (Hs) «Ведомый (Slave) передатчик»

Код	Название	Описание	Данные		Управление: SMBCTL1				Следующее действие
			R/W	SMBSDA	clrst	ask	stop	start	
1	2	3	4	5	6	7	8	9	10
34_{16}	HSTADPA	Принят адрес ведомый (Slave), сигнал ACK	W	Данные	1	X	0	0	Передача байта данных
36_{16}	HSTDAPA	Передача байта данных, сигнал ACK	W	Данные	1	X	0	0	Передача байта данных
37_{16}	HSTDANA	Передача байта данных, сигнал NACK	–		1	X	0	0	«Неадресованный ведомый (Slave)»
					1	X	0	1	«Неадресованный ведомый (Slave)», «Старт» после того, как шина станет свободна

Рисунок 383 – Высокоскоростной режим (Hs) «Ведомый (Slave) передатчик»

17.3.7 Арбитраж и обнаружение ошибок на шине

Арбитраж

Приоритет в режиме «Ведущий (Master) передатчик» может быть потерян в случаях, когда два ведущих (Master) одновременно формируют состояние «Старт» и начинают передачу данных. I2C по-разному реагирует на потерю приоритета в случаях, когда она произошла при передаче байта адреса или при передаче байта данных:

1) В случае потери приоритета при передаче байта адреса, I2C переходит в режим «Ведомый (Slave) приемник» и начинает отслеживать передачу своего ведомого (Slave) адреса на шине. Если адреса совпали, I2C остается в режиме ведомого (Slave). Если адреса не совпали, I2C переходит в режим «Неадресованный ведомый (Slave)».

2) Как только произошла потеря приоритета при передаче байта данных, I2C переходит в режим «Неадресованный ведомый (Slave)».

Обнаружение ошибок на шине

Ошибки на шине случаются, если во время передачи адреса, данных, сигнала подтверждения обнаруживаются состояния «Старт» или «Стоп», показанные в таблице 162. При нахождении ошибки на шине, I2C действует следующим образом:

а) Код статуса шины в SMBST.MODE становится BERROR ($1F_{16}$).

б) Генерируется прерывание, если был установлен бит SMBCTL1.INTEN.

в) I2C переходит из данного режима в режим «Неадресованный ведомый (Slave)».

г) Линии SDA и SCL становятся свободны.

Таблица 162 – Ошибки на шине, действия и статус

Код	Название	Описание	Данные		Управление: SMBCTL1				Следующее действие
			R/W	SMBSDA	clrst	ask	stop	start	
1	2	3	4	5	6	7	8	9	10
00_{16}	IDLE	Информация о режиме отсутствует	–		–				Ожидание завершения текущей передачи данных
$1F_{16}$	BERROR	Обнаружены некорректные состояния «Старт» или «Стоп»	–		1	0	0	0	Режим «Неадресованный Slave»

Исправление ошибок на шине

В некоторых случаях I2C и другое устройство, подключенное к шине, могут обнаружить ошибку и вызвать простой шины. В этом случае может быть не завершено формирование состояния «Старт» и I2C перестанет функционировать.

Для возврата из этого состояния необходимо выполнить следующие действия:

- запретить и снова разрешить работу I2C (сначала обнулить бит SMBCTL2.ENABLE, а затем записать туда единицу);
- во время простоя проверить, не подключен ли другой активный ведущий (Master) к шине (бит SMBCST.BB остается равным «0»).

В этот момент некоторые ведомые (Slave) могут не распознать ошибку на шине. Для исправления ошибки I2C становится ведущим (Master) устройством шины, формируя состояние «Старт» и передавая адрес. Затем I2C формирует состояние «Стоп», чтобы синхронизировать все ведомые (Slave) устройства.

17.3.8 Режимы останова (Halt) и ожидания (Idle)

В режимах ожидания (Idle) или останова (Halt) системный тактовый сигнал остановлен и на шине не производятся операции с данными. Таким образом, вход в режимы ожидания (Idle) или останова (Halt) не должен осуществляться при работе I2C в качестве ведущего (Master) или ведомого (Slave) при передаче или приеме данных. Следовательно, I2C должна работать в режиме «Неадресованный ведомый (Slave)» при входе в режимы ожидания (Idle) или останова (Halt) (биты SMBST.MODE читаются как 000_2). Это гарантирует то, что I2C не сформирует сигнал подтверждения при передаче адреса и при дальнейших действиях на шине.

При входе шины в режим ожидания (Idle) или останова (Halt) работа I2C запрещается (обнуляется бит SMBCTL2.ENABLE). При этом регистры SMBCTL1,

SMBST и SMBCST очищаются, генерация тактовых сигналов приостанавливается, чтобы гарантировать правильное возобновление работы I2C в дальнейшем.

17.3.9 Конфигурация частоты тактового сигнала SCL

Когда I2C работает в режиме ведущего (Master) устройства шины, то можно программно задавать частоту тактового сигнала на линии SCL.

Режим «Быстрый/Стандартный» (F/S)

Величина, содержащаяся в поле SMBCTL2.SCLFRQ, определяет период тактового сигнала SCL относительно системного тактового сигнала. Также учитывается, что длительность тактовых импульсов на линии SCL определяется тактовым сигналом ведущего (Master) устройства с наименьшей длительностью импульсов, а длительность паузы между тактовыми импульсами определяется тактовым сигналом ведущего (Master) устройства с наибольшей паузой между импульсами.

Высокоскоростной режим (Hs)

Величина, содержащаяся в поле SMBCTL3.HSDIV, определяет коэффициент деления тактового сигнала SCL относительно системного тактового сигнала. В соответствии с протоколом шины, синхронизация тактового сигнала не применяется при работе в режиме ведущего (Master), то есть другое ведущее (Master) устройство не может уменьшить длительность тактового импульса. Величина в SMBCTL3.HSDIV определяет длительность тактового импульса на SCL, длительность паузы между тактовыми импульсами равна удвоенной длительности импульсов.

17.3.10 Время ожидания на линии SCL

Спецификация SMBus определяет наименьшее время ожидания на линии как TTIMEOUT = 25 мс. Если пауза между двумя тактовыми импульсами превысила TTIMEOUT, то устройство должно прервать текущую передачу. Ведущий (Master) должен сформировать состояние «Старт» в процессе передачи или после ее окончания. Ведомый (Slave) должен освободить шину. Устройства, обнаружившие данное состояние, должны восстановить свои соединения и ожидать формирования состояния «Старт» в пределах 10 мс.

Функциональное описание счетчика времени ожидания SCL

I2C содержит счетчик времени ожидания для обнаружения и уведомления о простое на шине, показанный на рисунке 384.

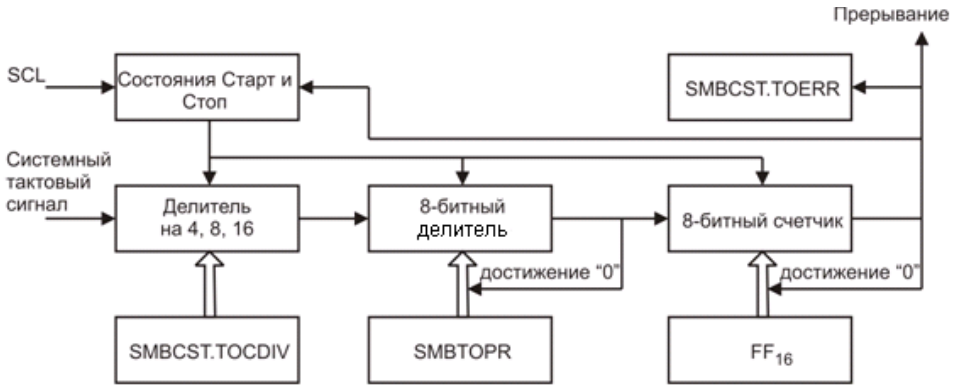


Рисунок 384 – Обнаружение времени ожидания

Счетчик времени ожидания включает в себя делитель тактовой частоты, 8-битный программируемый делитель и 8-битный основной счетчик. Все элементы счетчика времени ожидания начинают работу по отрицательному фронту на SCL (если работа счетчика разрешена). Положительный фронт на SCL сбрасывает значения делителя тактовой частоты, делителя и основного счетчика. Делитель начинает отчитывать «вниз», начиная с величины, записанной в регистр SMBTOPR. После достижения нуля счетчиком делителя, он перезагружает значение из SMBTOPR. Основной счетчик считает «вниз» от величины FF_{16}. Каждое достижения нуля счетчиком делителя декрементирует значение основного счетчика. Переход основного счетчика от нуля к FF_{16} вызывает остановку основного счетчика, делителя счетчика и тактового делителя, при этом устанавливается флаг SMBCST.TOERR. Дополнительно устанавливается флаг SMBST.INT. Прерывание генерируется, если был установлен бит SMBCTL1.INTEN.

Период времени ожидания определяется следующим выражением:

$$T_{сист} \times Div \times (SMBTOPR + 1) \times 256,$$

где $T_{сист}$ – период системного тактового сигнала;

Div – величина, определяемая SMBCST.TOCDIV (4, 8 или 16).

17.4 Описание регистров

17.4.1 Последовательных регистр данных SMB модуля I2C (SMBSDA)

Последовательных регистр данных SMBSDA является 8-битным сдвиговым регистром, используемым для передачи и приема данных. Наиболее значимый бит MSB передается (принимается) первым, а наименее значимый бит LSB передается (принимается) последним. Описание бит SMBSDA показано на рисунке 385.

7	6	5	4	3	2	1	0
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X

R – доступ чтения, W - доступ записи, -X – значение после сброса (неопределенный)

Рисунок 385 – Последовательный регистр данных SMB модуля I2C (SMBSDA) – адрес 4800h

Биты 7-0 DATA7 – DATA0. Запись в SMBSDA возможна только в случае, когда установлен флаг SMBST.INT. Попытки записи регистра в других случаях будут проигнорированы. Регистр SMBSDA не очищается после сброса, а содержит случайные данные до того, как будет записан программно или до приема и сдвига данных.

17.4.2 Статусный регистр SMB модуля I2C (SMBST)

Статусный регистр SMBST является 8-битным сдвиговым регистром, содержащий текущее значение статуса I2C. Во время запрещения работы I2C, SMBST принимает значение 00_{16}. Описание бит SMBST показано на рисунке 386.

7	6	5	4	3	2	1	0
INT	Зарезервировано	MODE5	MODE4	MODE3	MODE2	MODE1	MODE0
R-0		R-0	R-0	R-0	R-0	R-0	R-0

R – доступ чтения, -0 – значение после сброса

Рисунок 386 – Статусный регистр SMB модуля I2C (SMBST) – адрес 4804h

Бит 7 INT. Флаг прерывания.
0 = Устанавливается после записи «1» в бит SMBCTL1.CRST или при запрещении работы I2C при обнулении бита SMBCTL2.ENABLE.
1 = Устанавливается после девятого тактового импульса на SCL (SCL – в низком уровне), в любое время разрешено программное вмешательство.

Бит 6 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.

Биты 5-0 MODE5 – MODE0. Текущий статус и режим работы I2C. Расшифровка кода статуса шины приведена в таблице 163.

Условия выставления флага прерывания:

- при работе в режимах «Ведущий (Master) передатчик», «Ведущий (Master) приемник», «Ведомый (Slave) приемник» или «Ведомый (Slave) передатчик»;
- после обнаружения совпадения адреса (совпадение адреса ведомого (Slave), адреса отклика, адреса общего вызова); необходимо программно проверять содержимое регистра SMBSDA для определения, какое совпадение произошло;
- после успешного формирования состояний «Старт» или «Повторный старт»;
- если был получен инверсного сигнала подтверждения NACK;
- при успешном формировании состояний «Стоп» или «Повторный старт»;
- после обнаружения ошибки на шине.

Линия SCL удерживается в низком уровне, пока установлен флаг прерывания INT регистра SMBST. Следующие условия устанавливают флаг прерывания, но не приводят к удерживанию SCL в низком уровне:

- ожидание на линии SCL;
- состояние «Стоп» в режиме ведомого (Slave) поле MODE = SSTOP ($1C_{16}$);

- потеря приоритета приводит к входу модуля I2C в режим «Неадресованный ведомый (Slave)» поле MODE = IDLARL (03_{16}) или MODE = HIDLARL (23_{16});
- передан байт данных, инверсный сигнал подтверждения NACK (17_{16}).

Таблица 163 – Режимы работы модуля I2C

Режим	Код	Мнемонический код	Описание
1	2	3	4
Общий	00_{16}	IDLE	Ожидание (Idle), не доступна достоверная информация о статусе
F/S «Ведущий (Master)»	01_{16}	STDONE	Сформировано состояние «Старт»
	02_{16}	RSDONE	Сформировано состояние «Повторный старт»
	03_{16}	IDLARL	Потеря приоритета, вход в режим «Неадресованный ведомый (Slave)»
F/S «Ведущий (Master) передатчик»	04_{16}	MTADPA	Отправлен адрес ведомого (Slave), сигнал ACK
	05_{16}	MTADNA	Отправлен адрес ведомого (Slave), сигнал NACK
	06_{16}	MTDAPA	Отправлен байт данных, сигнал ACK
	07_{16}	MTDANA	Отправлен байт данных, сигнал NACK
F/S «Ведущий (Master) приемник»	08_{16}	MRADPA	Отправлен адрес ведомого (Slave), сигнал ACK
	09_{16}	MRADNA	Отправлен адрес ведомого (Slave), сигнал NACK
	$0A_{16}$	MRDAPA	Принят байт данных, сигнал ACK
	$0B_{16}$	MRDANA	Принят байт данных, сигнал NACK
F/S «Ведущий (Master)»	$0C_{16}$	MTMCER	Адрес ведущего (Master) передан с ошибкой (сигнал ACK)
–	$0D_{16}$ – $0F_{16}$	–	Не используются
F/S «Ведомый (Slave) приемник»	10_{16}	SRADPA	Принят адрес ведомого (Slave), сигнал ACK
	11_{16}	SRAAPA	Принят адрес ведомого (Slave) после потери приоритета, сигнал ACK
	12_{16}	SRDAPA	Принят байт данных, сигнал ACK
	13_{16}	SRDANA	Принят байт данных, сигнал NACK
F/S «Ведомый (Slave) передатчик»	14_{16}	STADPA	Принят адрес ведомого (Slave), сигнал ACK
	15_{16}	STAAPA	Принят адрес ведомого (Slave) после потери приоритета, сигнал ACK
	16_{16}	STDAPA	Отправлен байт данных, сигнал ACK
	17_{16}	STDANA	Отправлен байт данных, сигнал NACK
F/S «Ведомый (Slave) передатчик» отклик	18_{16}	SATADP	Принят адрес отклика, сигнал ACK
	19_{16}	SATAAP	Принят адрес отклика после потери приоритета, сигнал ACK
	$1A_{16}$	SATDAP	Отправлены данные отклика, сигнал ACK
	$1B_{16}$	SATDAN	Отправлены данные отклика, сигнал NACK
F/S «Ведомый (Slave)»	$1C_{16}$	SSTOP	В режиме ведомого (Slave) зафиксировано состояние «Стоп»
	$1D_{16}$	SGADPA	Принят адрес общего вызова, сигнал ACK
	$1E_{16}$	SDAAPA	Принят адрес общего вызова после потери приоритета, сигнал ACK

Продолжение таблицы 163

1	2	3	4
Общий	$1F_{16}$	BERROR	Ошибка на шине (некорректное формирование состояний «Старт» или «Стоп»)
Hs «Ведущий (Master)»	20_{16}	–	Не используется
	21_{16}	HMTMCOK	Адрес ведущего (Master) передан – переход в высокоскоростной режим (Hs)
	22_{16}	HRSDONE	Формирование состояния «Повторный старт»
	23_{16}	HIDLARL	Потеря приоритета, переход в высокоскоростной режим (Hs) «Неадресованный ведомый (Slave)»
Hs «Ведущий (Master) передатчик»	24_{16}	HMTADPA	Отправлен адрес ведомого (Slave), сигнал ACK
	25_{16}	HMTADNA	Отправлен адрес ведомого (Slave), сигнал NACK
	26_{16}	HMTDAPA	Отправлен байт данных, сигнал ACK
	27_{16}	HMTDANA	Отправлен байт данных, сигнал NACK
Hs «Ведущий (Master) приемник»	28_{16}	HMRADPA	Отправлен адрес ведомого (Slave), сигнал ACK
	29_{16}	HMRADNA	Отправлен адрес ведомого (Slave), сигнал NACK
	$2A_{16}$	HMRDAPA	Принят байт данных, сигнал ACK
	$2B_{16}$	HMRDANA	Принят байт данных, сигнал NACK
–	$2C_{16}$ – $2F_{16}$	–	Не используются
Hs «Ведомый (Slave) приемник»	30_{16}	HSRADPA	Принят адрес ведомого (Slave), сигнал ACK
	31_{16}	–	Не используется
	32_{16}	HSRDAPA	Принят байт данных, сигнал ACK
	33_{16}	HSRDANA	Принят байт данных, сигнал NACK
Hs «Ведомый (Slave) приемник»	34_{16}	HSTADPA	Принят адрес ведомого (Slave), сигнал ACK
	35_{16}	–	Не используется
	36_{16}	HSTDAPA	Отправлен байт данных, сигнал ACK
	37_{16}	HSTDANA	Отправлен байт данных, сигнал NACK
–	38_{16} – $3F_{16}$	–	Не используется

17.4.3 Статусный регистр управления SMB модуля I2C (SMBCST)

Статусный регистр управления SMBCST является 8-битным регистром, содержащим значение статуса модуля I2C. Во время запрещения работы I2C все биты регистра становятся равными нулю. Описание бит SMBCST показано на рисунке 387.

7	6	5	4	3	2	1	0
PECFAULT	PECNEXT	TGSCL	TSDA	TOERR	TOCDIV1	TOVDIV0	BB
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	

R – доступ чтения, W - доступ записи, -0 – значение после сброса

Рисунок 387 – Статусный регистр управления SMB модуля I2C (SMBCST) – адрес 4808h

Бит 7 PECFAULT. Ошибка пакета данных. Если установлена «1», то при последней передаче произошла ошибка пакета данных (значение регистра Packet Error Checking Register не равно нулю).

Бит 6 PECNEXT. Следующий байт PEC. Установка этого бита означает, что следующий байт на шине будет байт PEC. Отклик на этот бит зависит от текущего режима работы I2C. Когда I2C находится в режиме передачи, то установка этого бита передает результат вычисления PEC в SMBSDA регистр. После того, как флаг INT будет очищен, передача байта PEC снова будет начата. Когда I2C действует в качестве приемника, этот бит указывает в PEC логике, что следующий байт, который будет получен – это байт PEC. В режиме ведомого приемника I2C будет автоматически отправлять ACK или NACK в зависимости от того, была ли обнаружена ошибка PEC, и, следовательно, снимать флаг SMBCTL1.ACK. В режиме ведущего приемника, для I2C по-прежнему правильная передача определяется значением бита SMBCTL1.ACK.

Бит 5 TGSCL. Удерживание SCL в неактивном уровне. Разрешает удерживание SCL в неактивном уровне на период исправления ошибки. Когда SDA низкий, запись «1» удерживает SCL в неактивном уровне на один цикл. Попытка записи в этот бит при высоком уровне сигнала на SDA игнорируется. Бит очищается при завершении удерживания SCL.

Бит 4 TSDA. Тест SDA. Содержит текущее значение SDA. Этот бит может быть использован при исправлении ошибки на шине. Этот бит только для чтения, попытка записи в него игнорируется.

Бит 3 TOERR. Ошибка ожидания на шине.
0 = Бит очищается при записи «1» в бит SMBCTL1.CLRST.
1 = Зафиксировано состояние ожидания на линии SCL. Бит устанавливается при достижении нуля основным счетчиком времени ожидания.

Биты 2-1 TOCDIV1, TOCDIV0. Эти биты определяют коэффициент деления

для делителя времени ожидания SCL.
00 = Запрещение работы, нет тактового сигнала.
01 = Деление на 4.
10 = Деление на 8.
11 = Деление на 16.

Бит 0 BB. Шина занята.
0 = Бит обнуляется при обнаружении состояния «Стоп» или при запрещении работы I2C.
1 = Устанавливается, когда шина активна (линии SDA и SCL в низком уровне) или про формировании сигнала «Старт». Показывает, что шина занята.

17.4.4 Регистр управления 1 SMB модуля I2C (SMBCTL1)

Регистр управления SMBCTL1 является 8-битным регистром конфигурации и управления I2C. При запрещении работы модуля I2C SMBCTL2.ENABLE = 0_2 регистр принимает значение 00_2. Описание бит SMBCTL1 показано на рисунке 388.

7	6	5	4	3	2	1	0
CLRST	SMBARE	GCMEN	ACK	Зарезервировано	INTEN	STOP	START
RW-0	RW-0	RW-0	RW-0		RW-0	RW-0	RW-0

R – доступ чтения, W - доступ записи, -0 – значение после сброса

Рисунок 388 – Регистр управления 1 SMB модуля I2C (SMBCTL1) – адрес 480Ch

Бит 7 CLRST. Очистка статуса
0 = Запись в этот бит не имеет эффекта, этот бит доступен только для записи, всегда читается как «0».
1 = Очищает бит статусный бит SMBST.INT.

Бит 6 SMBARE. Разрешение распознавания адреса отклика.
0 = Модуль I2C не отвечает на адрес отклика на шине. Бит очищается при выходе модуля I2C из режимов останова (Halt) или ожидания (Idle), независимо от его состояния до входа в режим останова (Halt) или ожидания (Idle).
1 = Разрешение распознавания среди принятых адресов адреса отклика 0001100_2 в режиме ведомого (Slave).

Бит 5 GCMEN. Разрешение распознавания адреса общего вызова.
0 = I2C не распознает адрес общего вызова. Бит очищается при выходе модуля I2C из режимов останова (Halt) или ожидания (Idle).
1 = Разрешение распознавания среди принятых адресов адреса общего вызова 00_2 в режиме ведомого (Slave)

Бит 4 ACK. Бит подтверждения. ACK игнорируется в режиме передатчика.
Когда I2C действует в качестве приемника (ведущего или ведомого), ACK показывает значение, что I2C пошлет в следующий цикл подтверждения.
0 = ACK сбрасывается после первого цикла подтверждения.
1 = Сообщает передающему устройству о необходимости остановки передачи данных, поскольку приемнику либо не нужно, либо он не может получить больше данных.

Бит 3 Зарезервировано. Чтения неопределенны, записи не имеют эффекта.

Бит 2 INTEN. Разрешение прерывания.
0 = Прерывание запрещено.
1 = Прерывание разрешено.

Бит 1 STOP. «Стоп».
0 = Очищается после завершения формирования состояния «Стоп».
1 = При работе в режиме ведущего (Master) – формирование состояния «Стоп», что завершит или прервет текущую передачу данных.

Бит 0 START. «Старт».
0 = Очищается при завершении формирования состояния «Старт» или после обнаружения ошибки на шине SMBST.MODE = $1F_{16}$.
1 = Устанавливается при необходимости формирования состояния «Старт» на шине.

17.4.5 Регистр собственного адреса SMB модуля I2C (SMBADDR)

Регистр собственного адреса SMBADDR является 8-битным регистром, показывающий адрес I2C/SMBus. Описание бит SMBCTL1 показано на рисунке 389.

7	6	5	4	3	2	1	0
SAEN	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X

R – доступ чтения, W - доступ записи, -X – значение после сброса (неопределенный)

Рисунок 389 – Регистр собственного адреса SMB модуля I2C (SMBADDR) – адрес 4810h

Бит 7 SAEN. Разрешение адреса ведомого (Slave). Когда установлен, то показывает, что поле ADDR содержит значение адреса, разрешает сравнение ADDR и принятого байта адреса .

Биты 6-0 ADDR6 – ADDR0. Собственный адрес. Содержит 7-битный адрес I2C. При работе I2C в режиме ведомого (Slave) первые 7 бит, принятые после состояния «Старт», сравниваются с этим полем. Если значение ADDR совпадает с принятым адресом, то SMBST.MODE принимает значение 001_2.

17.4.6 Регистр управления 2 SMB модуля I2C (SMBCTL2)

Регистр управления SMBCTL2 является 8-битным регистром, который разрешает или запрещает работу I2C и определяет тактовую частоту I2C. Описание бит SMBCTL2 показано на рисунке 390.

7	6	5	4	3	2	1	0
SCLFRQ6	SCLFRQ5	SCLFRQ4	SCLFRQ3	SCLFRQ2	SCLFRQ1	SCLFRQ0	ENABLE
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W - доступ записи, -0 – значение после сброса

Рисунок 390 – Регистр управления 2 SMB модуля I2C (SMBCTL2) – адрес 4814h

Быты 7-1 SCLFRQ6 – SCLFRQ0. Частота SCL. Это поле определяет значение периода SCL при работе I2C в режиме ведущего (Master). Низкая и высокая длительность такта определяется следующим образом:

$$t_{SCL1} = t_{SCLh} = 2 \times SCLFRQ \times t_{CLK},$$

где t_{CLK} является системным тактом I2C цикла pclk.
В поле SCLFRQ может быть записано значение от 4 до 127. При выборе значения меньше 4, оно автоматически дополняется до 4.

Бит 0 ENABLE. Разрешение работы.
0 = Запрещение работы I2C. При очищении бита ENABLE регистры SMBCTL1, SMBST, SMBCST обнуляются, прекращается генерация тактового сигнала.
1 = Разрешение работы I2C.

17.4.7 Регистр делителя времени ожидания SCL модуля I2C (SMBTORP)

Регистр делителя времени ожидания SMBTORP является 8-битным регистром. Описание бит SMBTORP показано на рисунке 391.

7	6	5	4	3	2	1	0
SMBTORP7	SMBTORP6	SMBTORP5	SMBTORP4	SMBTORP3	SMBTORP2	SMBTORP1	SMBTORP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W - доступ записи, -0 – значение после сброса

Рисунок 391 – Регистр делителя времени ожидания SCL модуля I2C (SMBTORP) – адрес 4818h

Биты 7-0 SMBTOPR7 – SMBTOPR0. Делитель времени ожидания SCL. Содержит значение делителя времени ожидания.

17.4.8 Регистр управления 3 SMB модуля I2C (SMBCTL3)

Регистр управления SMBCTL3 является 8-битным регистром, для управления 10-битной адресацией, делителем тактового сигнала в высокоскоростном режиме (Hs). Описание бит SMBCTL3 показано на рисунке 392.

7	6	5	4	3	2	1	0
HSDIV3	HSDIV2	HSDIV1	HSDIV0	S10EN	S10ADR2	S10ADR1	S10ADR0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

R – доступ чтения, W - доступ записи, -0 – значение после сброса

Рисунок 392– Регистр управления 1 SMB модуля I2C (SMBCTL3) – адрес 481Ch

Биты 7-4 HSDIV3 – HSDIV0. Делитель тактового сигнала SCL в высокоскоростном режиме (Hs). Определяет значение периода SCL при работе I2C в высокоскоростном режиме. Тактовый период определяется следующим образом:

$$T_{HSLh} = HSDIV \times t_{CLK},$$

$$T_{HSL1} = 2 \times HSDIV \times t_{CLK},$$

где t_{CLK} является системным тактом I2C цикла pclk.
В поле HSDIV может быть записано значение от 2 до 16. При записи числа, меньшего 2, к начальному значению по умолчанию добавляется 2. То есть, при выборе значения делителя, равного 1, результирующим будет значение, равное 3.

Бит 3 S10EN. Разрешение 10-битной адресации. Когда установлена «1» разрешает распознавание 10-битного адреса ведомого (Slave). Для распознавания 10-битного адреса требуется одновременная установка битов S10EN и SMBADDR.SAEN.

Биты 2-0 S10ADR3 - S10ADR0. 10-битный адрес ведомого (Slave). Содержит старшие биты 10-битного адреса ведомого (Slave). Старшие 5 бит первого принятого адреса сравниваются с величиной 1110_{B}, младшие 3 бита – с величиной S10ADR.2 – S10ADR.1. Старший бит второго принятого адреса сравнивается с величиной S10ADR.0, младшие 7 бит – со значением поля SMBADDR.SADR.

Приложение А
(обязательное)

Сравнение набора инструкций ИС М1867ВМ1, 1867ВМ2, 1867ВЦ2Т, 1867ВЦ10Т

Набор инструкций ИС 1867ВЦ10Т идентичен тому, который имеется в ИС М1867ВМ1, 1867ВМ2, 1867ВЦ2Т. Все ссылки к ПЦОС в этом приложении также применимы и к 1867ВЦ10Т. Это приложение содержит таблицу А.4, в которой сравниваются инструкции, расположенные в алфавитном порядке для ИС М1867ВМ1, 1867ВМ2, 1867ВЦ2Т, 1867ВЦ10Т.

Каждый пункт в таблице А.4 показывает синтаксис инструкции, указывает, какие ИС поддерживают эту инструкцию и кратко описывает работу инструкции.

В подразделе А.1 приведен образец пункта таблицы А.4 и описаны символы и аббревиатура, используемые в таблице А.4.

ИС 1867ВМ2, 1867ВЦ2Т, 1867ВЦ10Т имеют расширенные инструкции. Для этих инструкций существует одна мнемоника, которая служит для описания функций нескольких простых инструкций. В подразделе А.2 подводится итог этих расширенных инструкций.

А.1 Использование сравнительной таблицы А.4 набора инструкций

Данный подраздел показывает пример пункта таблицы А.4 и список акронимов для прочтения таблицы сравнения инструкций.

А.1.1 Пример элемента таблицы А.4

В случаях, когда используется более чем один синтаксис, первый синтаксис используется обычно для прямой адресации, а второй – обычно для косвенной адресации. Где присутствует более одного синтаксиса, там каждый синтаксис снабжен указанием на устройство, для которого он допустим.

В таблице А.1 приведен пример описания инструкции AND.

Таблица А.1 – Пример описания инструкции AND

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
AND dma AND {ind} [, next ARP] AND #lk [, shift]	V V	V V	V V V	V V V	AND с аккумулятором. ИС M1867BM1 и 1867BM2: AND содержимого адреса ячейки памяти данных с 16 LSB аккумулятора. 16 MSB аккумулятора AND с 0. ИС 1867ВЦ2Т и 1867ВЦ10Т: AND содержимого адреса ячейки памяти данных или с 16-битным непосредственным значением с 16 LSB. 16 MSB аккумулятора AND с 0. Если указан сдвиг, то выполняется сдвиг константы перед AND. Сдвинутые значения младших бит внизу и старших бит вверху обрабатываются как 0.

Первый столбец – Синтаксис. Показывает мнемонику и синтаксис для инструкции AND.

Второй – пятый столбцы. Обозначение в следующих пяти колонках M1867BM1, 1867BM2, 1867ВЦ10Т и 1867ВЦ2Т указывает на ИС, в которой может использоваться этот синтаксис.

В примере таблицы А.1 можно использовать первые два синтаксиса с ИС M1867BM1, 1867BM2, 1867ВЦ10Т и 1867ВЦ2Т, но последний синтаксис можно использовать только с 1867ВЦ10Т и 1867ВЦ2Т.

Шестой столбец – Описание. Кратко описывает то, как инструкция выполняется. Часто функции устройств немного отличаются для различных устройств, поэтому необходимо читать пункт «Описание» перед использованием инструкции.

А.1.2 Символы и аббревиатура, используемая в таблице

В таблице А.2 перечислен набор инструкций и акронимов, используемых в этом приложении.

Таблица А.2 – Символы и акронимы, используемые в приложении

Символ	Описание	Символ	Описание
lk	16-битное непосредственное значение	INTM	Бит маски прерывания
k	8-битное немедленное значение	INTR	Бит режима прерывания
{ind}	Косвенный адрес Indirect address	OV	Бит переполнения Overflow bit
ACC	Аккумулятор Accumulator	P	Программная шина Program bus
ACCB	Буфер аккумулятора Accumulator buffer	PA	Адрес порта Port address
AR	Вспомогательный регистр Auxiliary register	PC	Программный счетчик Program counter
ARCR	Сравнение вспомогательного регистра Auxiliary register compare	PM	Режим сдвига произведения Product shifter mode
ARP	Указатель вспомогательного регистра Auxiliary register pointer	pma	Адрес программной памяти Program-memory address
BMAR	Регистр адреса пересылки блока Block move address register	RPTC	Счетчик повторений Repeat counter
BRCR	Регистр счета повторения блока Block repeat count register	shift, shiftn	Значение сдвига Shift value
C	Бит переноса Carry bit	src	Адрес источника Source address
DBMR	Регистр динамического манипулирования битом Dynamic bit manipulation register	ST	Статусный регистр Status register
dma	Адрес памяти данных Data-memory address	SXM	Бит режима расширения знака Sign-extension mode bit
DP	Указатель страницы памяти данных Data-memory page pointer	TC	Тест/Управляющий бит Test/control bit
dst	Адрес назначения Destination address	T	Временной регистр Temporary register
FO	Список формата статуса Format status list	TREGn	1867ВЦ2Т временный регистр (0–2)
FSX	Внешний импульс фрейма External framing pulse	TXM	Статусный регистр режима передачи Transmit mode status register
IMR	Регистр маски прерывания	XF	XF вывод статусного бита pin status bit

Ниже показано как интерпретируются операнд {ind} в зависимости от типа устройства:

{ind}

для ИС M1867BM1: { * | *+ | *– }

для ИС 1867BM2: { * | *+ | *– | *0+| *0– | *BR0+ | *BR0– }

для ИС 1867ВЦ10Т: { * | *+ | *– | *0+| *0– | *BR0+ | *BR0– }

для ИС 1867ВЦ2Т: { * | *+ | *– | *0+| *0– | *BR0+ | *BR0– }

Возможный выбор разделяется вертикальным слешем (|). Для примера ADD {ind} интерпретируется как:

для ИС M1867BM1 ADD { * | *+ | *– }

для ИС 1867BM2 ADD { * | *+ | *– | *0+ | *0– | *BR0+ | *BR0– }

для ИС 1867ВЦ10Т ADD { * | *+ | *– | *0+ | *0– | *BR0+ | *BR0– }

для ИС 1867ВЦ2Т ADD { * | *+ | *– | *0+ | *0– | *BR0+ | *BR0– }

Ниже показано, как устанавливаются значения для shift, shift1 и shift2 в зависимости от типа ИС:

shift

- для ИС M1867BM1: 0–15 (shift of 0–15 bits)
- для ИС 1867BM2: 0–15 (shift of 0–15 bits)
- для ИС 1867ВЦ10Т: 0–16 (shift of 0–16 bits)
- для ИС 1867ВЦ2Т: 0–16 (shift of 0–16 bits)

shift1

- для ИС M1867BM1: нет
- для ИС 1867BM2: 0–15 (shift of 0–15 bits)
- для ИС 1867ВЦ10Т: 0–16 (shift of 0–16 bits)
- для ИС 1867ВЦ2Т: 0–16 (shift of 0–16 bits)

shift2

- для ИС M1867BM1: нет
- для ИС 1867BM2: нет
- для ИС 1867ВЦ10Т: 0–15 (shift of 0–15 bits)
- для ИС 1867ВЦ2Т: 0–15 (shift of 0–15 bits)

В некоторых случаях значения проще, в этих случаях значения даются в столбце «Описание» таблицы.

А.2 Расширенные инструкции

Расширенная инструкция – это одна мнемоника, которая выполняет функции нескольких простых функций. Для примера, расширенная инструкция ADD выполняет функции ADD, ADDH, ADDK и ADLK и заменяет любые из этих инструкций другими инструкциями во время ассемблирования. Эти расширенные инструкции допустимы для ИС 1867BM2, 1867ВЦ10Т 1867ВЦ2Т (но не для M1867BM1).

В таблице А.3 показаны расширенные инструкции и функции, которые они выполняют (основывается на мнемонике для ИС M1867BM1, 1867BM2).

Таблица А.3 – Расширенные инструкции

Расширенная инструкция	Включает эти операции
ADD	ADD, ADDH, ADDK, ADLK
AND	AND, ANDK
BCND	BBNZ, BBZ, BC, BCND, BGEZ, BGZ, BIOZ, BLEZ, BLZ, BNC, BNV, BNZ, BV, BZ
BLDD	BLDD, BLKD
BLDP	BLDP, BLKP
CLRC	CLRC, CNFD, EINT, RC, RHM, ROVM, RSXM, RTC, RXF
LACC	LAC, LACC, LALK, ZALH
LACL	LACK, LACL, ZAC, ZALS
LAR	LAR, LARK, LRLK
LDP	LDP, LDPK
LST	LST, LST1
MAR	LARP, MAR
MPY	MPY, MPYK
OR	OR, ORK
RPT	RPT, RPTK
SETC	CNFP, DINT, SC, SETC, SHM, SOVM, SSXM, STC, SXF
SUB	SUB, SUBH, SUBK

А.3 Сравнительная таблица набора инструкций

В таблице А.4 содержится сравнение инструкций для устройств М1867ВМ1, 1867ВМ2, 1867ВЦ10Т и 1867ВЦ2Т.

Таблица А.4 – Сравнение инструкций

Синтаксис	M1867BM1	1867BM2	1867BЦ10T	1867BЦ2T	Описание
1	2	3	4	5	6
ABS	V	V	V	V	Абсолютное значение аккумулятора. Если содержимое аккумулятора меньше чем 0, то заменяет дополнением до 2 содержимого. Если содержимое есть 0, то не влияет
ADCB				V	Прибавляет ACCB к аккумулятору с переносом. Добавляет содержимое ACCB и значение бита переноса к аккумулятору. Если результат сложения формирует перенос из старших бит аккумулятора, тогда бит переноса C устанавливается в 1
ADD dma [, shift] ADD {ind} [, shift [, next ARP]] ADD #k ADD # lk [, shift2]	V V	V V	V V V V	V V V V	Прибавляет к аккумулятору со сдвигом. Устройства M1867BM1 и 1867BM2: прибавляют содержимое ячейки памяти данных к аккумулятору. Если указан сдвиг, то выполняется левый сдвиг содержимого ячейки перед сложением. Во время сдвига младшие биты заполняются 0, а старшие биты расширяют знак. Устройства 1867BЦ10T и 1867BЦ2T: прибавляют содержимое ячейки памяти данных или непосредственное значение к аккумулятору. Если указан сдвиг, то выполняется левый сдвиг содержимого ячейки перед сложением. Во время сдвига младшие биты заполняются 0, а старшие биты расширяют знак, если SXM = 1.
ADDB				V	Прибавляет ACCB к аккумулятору. Прибавляет содержимое ACCB к аккумулятору
ADDC dma ADDC {ind} [, next ARP]		V V	V V	V V	Прибавляет к аккумулятору с переносом. Прибавляет содержимое адресуемой ячейки памяти данных и бит переноса к аккумулятору.
ADDH dma ADDH {ind} [, next ARP]	V V	V V	V V	V V	Прибавляет к старшему слову аккумулятора. Прибавляет содержимое адресуемой ячейки памяти данных к 16 старшим разрядам аккумулятора. Младшие разряды не затрагиваются. Если результат сложения генерирует перенос, то бит переноса устанавливается в 1.

Продолжение таблицы А.4

1	2	3	4	5	6
ADDK #k		V	V	V	Прибавляет к аккумулятору короткое немедленное значение. Устройство М1867ВМ1: прибавляет 8-битное непосредственное значение к аккумулятору. Устройства 1867ВМ2, 1867ВЦ10Т и 1867ВЦ2Т: прибавляет 8-битное непосредственное значение с правым выравниванием к аккумулятору с результатом, заменяющим содержимое аккумулятора. Непосредственное значение обрабатывается как 8-битное положительное число. Расширение знака подавляется
ADDS dma ADDS {ind} [, next ARP]	V V	V V	V V	V V	Прибавление к аккумулятору с подавлением расширения знака. Прибавляет содержимое адресуемой ячейки памяти данных к аккумулятору. Значение обрабатывается как 16-битное беззнаковое число. Расширение знака подавляется
ADDT dma ADDT {ind} [, next ARP]		V V	V V	V V	Прибавляет к аккумулятору со сдвигом, указанным в Т регистре. Сдвинутое влево на величину 4 младших разрядов Т регистра содержимое адресуемой ячейки памяти данных прибавляется к аккумулятору. Если указан сдвиг, то левый сдвиг данных выполняется перед сложением. Во время сдвига младшие биты заполняются 0, а старшие биты знаков расширяются, если SXM = 1. Устройства 1867ВЦ10Т и 1867ВЦ2Т: Если результат сложения формирует перенос из старших бит аккумулятора, то бит переноса С устанавливается в 1
ADLK #lk [, shift]		V	V	V	Прибавляет к аккумулятору длинное непосредственное число со сдвигом. Прибавляет 16-битное значение к аккумулятору; если указан сдвиг, то левый сдвиг значения выполняется перед сложением. Во время сдвига младшие биты заполняются 0, а старшие биты знаково расширяются, если SXM = 1
ADRK #k		V	V	V	Сложение 8-битного значения к текущему вспомогательному регистру. Добавляет 8-битное значение к текущему вспомогательному регистру

Продолжение таблицы А.4

1	2	3	4	5	6
AND dma AND {ind} [, next ARP] AND #lk [, shift]	V V	V V	V V V	V V V	AND с аккумулятором. Устройства М1867ВМ1 и 1867ВМ2: AND содержимого адресуемой ячейки памяти данных с 16 младшими разрядами аккумулятора. Старшие разряды аккумулятора AND с 0. Устройства 1867ВЦ10Т и 1867ВЦ2Т: AND содержимого адресуемой ячейки памяти данных или непосредственное значение с содержимым аккумулятора. 16 старших разрядов аккумулятора AND с 0. Если указывается сдвиг, то константа сдвигается влево перед AND. Сдвинутые значения младших бит внизу и старших бит вверху обрабатываются как 0
ANDB				V	AND ACCB к аккумулятору. AND содержимого ACCB к аккумулятору
ANDK #lk [, shift]		V	V	V	AND непосредственного значения к аккумулятору со сдвигом. AND 16-битного непосредственного значения с содержимым аккумулятора, если указан сдвиг, то левый сдвиг константы выполняется перед AND.
APAC	V	V	V	V	Сложение P регистра к аккумулятору. Добавляет содержимое P регистра к аккумулятору. Устройства 1867ВМ2, 1867ВЦ10Т и 1867ВЦ2Т: Перед сложением выполняется левый сдвиг содержимого P регистра, как определено PM битами
APL [#lk] ,dma APL [#lk,] {ind} [, next ARP]				V V	AND значения памяти данных с DBMR или длинной константой. AND значения памяти данных с содержимым DBMR или длинной константой. Если длинная константа указана, то она AND с содержимым ячейки памяти данных. Результат записывается обратно в ячейку памяти данных прежде хранящую первый операнд. Если результат равен 0, то бит TC установлен в 1, иначе бит TC сбрасывается в 0
B pma B pma [, {ind} [, next ARP]]	V	V	V		Безусловный переход. Переход к указанному адресу памяти программ. Устройства 1867ВМ2 и 1867ВЦ10Т: Модифицирует AR и ARP как указано

Продолжение таблицы А.4

1	2	3	4	5	6
B[D] pma [, {ind} [, next ARP]]				V	Безусловный переход с необязательной задержкой. Модификация текущего AR и ARP как указывается и передает управление в обозначенный адрес памяти программ. Если указан переход с задержкой (BD), то следующие 2 слова инструкции (2 однословных и одна 2-словная инструкция) выбирается и выполняется перед переходом
BACC		V	V		Переход по адресу, указанному в аккумуляторе. Переход к ячейке, указанной в 16 младших разрядах аккумулятора
BACC[D]				V	Переход по адресу, указанному в аккумуляторе, с необязательной задержкой. Переход к ячейке, указанной в 16 младших разрядах аккумулятора. Если указан переход с задержкой (BACCD), то следующие два слова инструкции (2 однословных и одна 2-словная инструкция) выбирается и выполняется перед переходом.
BANZ pma BANZ pma [, {ind} [, next ARP]]	V	V	V		Переход по вспомогательному регистру, если не 0. Если содержимое 9 младших бит текущего вспомогательного регистра (М1867ВМ1) или содержимое целого регистра (1867ВМ2) ≠ 0, то выполняется переход к указанному адресу программной памяти. Устройства 1867ВМ2 и 1867ВЦ10Т: модифицирует текущий регистр AR и ARP (если указывается) или декрементируется текущий AR (по умолчанию). Устройство М1867ВМ1: декрементируется текущий AR
BANZ[D] pma [, {ind} [, next ARP]]				V	Переход, если вспомогательный регистр не равен 0 с задержкой. Если содержимое текущего вспомогательного регистра ≠ 0, то выполняется переход к указанному адресу программной памяти. Модифицирует текущий AR и ARP как указывается или декрементируется текущий AR. Если указывается задержка перехода (BANZD), то следующие два слова инструкции (2 одно словных и одна 2-словная инструкция) выбираются и выполняются перед переходом

Продолжение таблицы А.4

1	2	3	4	5	6
BBNZ pma [, {ind} [, next ARP]]		V	V	V	Переход по биту TC ≠ 0. Если бит TC =1, то переход к указанному адресу программы. Устройство 1867ВМ2: модифицирует текущий AR и ARP, как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
BBZ pma [, {ind} [, next ARP]] BBZ pma		V	V	V V	Переход по биту TC = 0. Если бит TC = 0, то переход к указанному адресу программы. Устройство 1867ВМ2: модифицирует текущий AR и ARP, как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
BC pma [, {ind} [, next ARP]] BC pma		V	 V	V V	Переход по переносу. Если бит C =1, то переход к указанному адресу программы. Устройство 1867ВМ2: модифицирует текущий AR и ARP, как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
BCND pma, cond1 [, cond2] [, ...]			V		Условный переход. Переход к указанному адресу памяти программ, если встречается указанное условие. Не все условные комбинации допустимы
BCND[D] pma, cond1 [, cond2] [, ...]				V	Условный переход с задержкой. Переход к указанному адресу памяти программ, если встречается указанное условие. Не все условные комбинации допустимы. Если указана задержка перехода (BCNDD), то следующие два слова инструкции (два однословных и одна двухсловная инструкция) выбираются и выполняются перед переходом

Продолжение таблицы А.4

1	2	3	4	5	6
BGEZ pma BGEZ pma [, {ind} [, next ARP]]	V	 V	V	V V	Переход, если аккумулятор равен 0. Если содержимое аккумулятора равно 0, то выполняется переход к указанному адресу памяти программ. Устройство 1867ВМ2: модифицирует текущий AR и ARP как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: Если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
BGZ pma BGZ pma [, {ind} [, next ARP]]	V	 V	V	V V	Переход, если аккумулятор > 0. Если содержимое аккумулятора > 0, то выполняется переход к указанному адресу памяти программ. Устройство 1867ВМ2: модифицирует текущий AR и ARP как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
BIOZ pma BIOZ pma [, {ind} [, next ARP]]	V	 V	V	V V	Переход по состоянию вывода BIO = 0. Если состояние вывода BIO равно низкому уровню, то выполняется переход к указанному адресу памяти программ. Устройство 1867ВМ2: модифицирует текущий AR и ARP как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
BIT dma, bit code BIT {ind}, bit code [, next ARP]		V V	V V	V V	Тест/проверка бита. Копирует указанный бит в ячейке памяти данных в бит TC статусного регистра ST1
BITT dma BITT {ind} [, next ARP]		V V	V V	V V	Тест бита, указанного T регистром. Устройства 1867ВМ2 и 1867ВЦ10Т: копирует указанный бит значения ячейки памяти данных в бит TC статусного регистра ST1. 4 младших разряда T регистра указывают какой бит копируется. Устройство 1867ВЦ2Т: копирует, указанный бит значения ячейки памяти данных в бит TC статусного регистра ST1. Четыре младших разряда TREG2 регистра указывают какой бит копируется

Продолжение таблицы А.4

1	2	3	4	5	6
BLDD #lk, dma BLDD #lk, {ind} [, next ARP] BLDD dma, #lk BLDD {ind}, #lk [, next ARP] BLDD BMAR, dma BLDD BMAR, {ind} [, next ARP] BLDD dma BMAR BLDD {ind}, BMAR [, next ARP]			V V V V	V V V V V V V V	Пересылка блока из памяти данных в память данных. Копирует блок памяти данных в память данных. Копируемый блок памяти данных указывается src, а приемный блок указывается dst. Устройство 1867ВЦ10Т: слово источника и/или приемника может указываться значением длинной непосредственной адресации. Можно использовать инструкцию RPT с BLDD для пересылки последовательных слов, указанных косвенно в памяти данных для непрерывной программной области. Количество пересылаемых слов равно на 1 больше, чем число, содержащееся в RPTC в начале инструкции. Устройство 1867ВЦ2Т: слово пространства источника и/или приемника может указываться длинным непосредственным значением содержимого BMAR или адреса памяти данных. Можно использовать инструкцию RPT с BLDD для пересылки последовательных слов, указанных косвенно в памяти данных для непрерывной программной области. Количество пересылаемых слов на 1 больше чем число, содержащееся в RPTC в начале инструкции
BLDP dma BLDP {ind} [, next ARP]				V V	Пересылка блока из памяти данных в память программ. Копирует блок памяти данных в память программ, указанных BMAR. Можно использовать инструкцию RPT с BLPD для пересылки последовательных слов, указанных косвенно в памяти данных для непрерывной программной области памяти, указанной BMAR

Продолжение таблицы А.4

1	2	3	4	5	6
BLEZ pma BLEZ pma [, {ind} [, next ARP]]	V	 V	V V	V V	Переход, если аккумулятор ≤ 0. Если содержимое аккумулятора ≤ 0, то выполняется переход к указанному адресу в программной памяти. Устройство 1867ВМ2: модифицирует текущий AR и ARP как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
BLKD dma1, dma2 BLKD dma1, {ind} [, next ARP]		V V	V V	V V	Пересылка из памяти данных в память данных. Пересылка блока слов из одной ячейки в памяти данных в другую ячейку памяти данных. Модифицирует текущий AR и ARP как указывается. RPT или RPTK должны использоваться с инструкцией BLKD с режимом косвенной адресации, если пересылается больше чем одно слово, для пересылки последовательных слов, указанных косвенно в памяти данных для непрерывной области памяти данных. Количество пересылаемых слов на 1 больше чем число, содержащееся в RPTC в начале инструкции
BLKP pma, dma BLKP pma, {ind} [, next ARP]		V V	V V	V V	Пересылка блока слов программной памяти в память данных. Пересылка блока слов из ячейки в программной памяти в ячейку памяти данных. Модифицирует текущий AR и ARP как указывается. RPT или RPTK должны использоваться с инструкцией BLKP с режимом косвенной адресации, если пересылается больше чем одно слово, для пересылки последовательных слов, указанных косвенно в памяти данных для непрерывной области памяти данных. Количество пересылаемых слов на 1 больше чем число, содержащееся в RPTC в начале инструкции

Продолжение таблицы А.4

1	2	3	4	5	6
BLPD #pma, dma BLPD #pma, {ind} [, next ARP] BLPD BMAR, dma BLPD BMAR, {ind} [, next ARP]			V V	V V V V	Пересылка блока слов программной памяти в память данных. Копирует блок программной памяти в память данных. Блок программной памяти указывается src, а блок назначения в памяти данных указывается dst. Устройство 1867ВЦ10Т: слово пространства источника указывается длинным немедленным значением. Можно использовать инструкцию RPT с BLPD для пересылки последовательных слов, указанных косвенно в памяти данных для непрерывной программной области памяти. Устройство 1867ВЦ2Т: слово пространства источника может указываться длинным немедленным значением или содержимым BMAR. Можно использовать инструкцию RPT с BLPD для пересылки последовательных слов, указанных косвенно в памяти данных для непрерывной программной области памяти
BLZ pma BLZ pma [, {ind} [, next ARP]]	V	 V	V V	V	Переход, если аккумулятор < 0. Если содержимое аккумулятора ≤ 0, то выполняется переход к указанному адресу в программной памяти. Устройство 1867ВМ2: модифицирует текущий AR и ARP как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
BNC pma [, {ind} [, next ARP]]		V	V	V	Переход, если нет переноса. Если бит C =0, то переход к указанному адресу программы. Устройство 1867ВМ2: модифицирует текущий AR и ARP как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается

Продолжение таблицы А.4

1	2	3	4	5	6
BNV pma [, {ind} [, next ARP]]		V	V	V	Переход, если нет переполнения. Если бит OV =0, то переход к указанному адресу программы. Устройство 1867ВМ2: модифицирует текущий AR и ARP как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
BNZ pma BNZ pma [, {ind} [, next ARP]]	V	V	V	V	Переход, если аккумулятор ≠ 0. Если содержимое аккумулятора ≠ 0, то выполняется переход к указанному адресу в программной памяти. Устройство 1867ВМ2: модифицирует текущий AR и ARP как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
BSAR [shift] □□Barrel Shift				V	Циклический сдвиг. В одном цикле выполняет от 1 до 16 арифметических циклических сдвигов вправо аккумулятора. Знаковое расширение определяется битом SXM статусного регистра ST1
BV pma BV pma [, {ind} [, next ARP]]	V	V	V	V	Переход, если есть переполнение. Если бит OV =1, то переход к указанному адресу программы и очистка бита OV. Устройство 1867ВМ2: модифицирует текущий AR и ARP как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
BZ pma BZ pma [, {ind} [, next ARP]]	V	V	V	V	Переход, если аккумулятор = 0. Если содержимое аккумулятора = 0, то выполняется переход к указанному адресу в программной памяти. Устройство 1867ВМ2: модифицирует текущий AR и ARP как указывается. Устройства 1867ВЦ10Т и 1867ВЦ2Т: если использован ключ –p (портирование), то модифицирует текущий AR и ARP как указывается
CALA	V	V	V		Косвенный вызов подпрограммы. Содержимое аккумулятора указывает адрес подпрограммы. Инкремент PC, запись PC в стек и загрузка 12 (М1867ВМ1) или 16 (1867ВМ2, 1867ВЦ2Т) младших разрядов аккумулятора в PC

Продолжение таблицы А.4

1	2	3	4	5	6
CALA[D]				V	Косвенный вызов подпрограммы с задержкой. Содержимое аккумулятора указывает адрес подпрограммы. Инкремент PC, запись PC в стек и загрузка 16 младших разрядов аккумулятора в PC. Если указана задержка (CALAD), то следующие два слова инструкции (две однословные инструкции или одна двухсловная инструкция) выбираются и выполняются перед вызовом
CALL pma CALL pma [,{ind} [, next ARP]]	V	V	V		Вызов подпрограммы. Содержимое адресуемой ячейки программной памяти указывает на адрес подпрограммы. Увеличивает PC на 2, записывает PC в стек и после этого загружает указанный адрес программной памяти в PC. Устройства 1867ВМ2 и 1867ВЦ10Т: модифицирует текущий AR и ARP как указывается
CALL[D] pma [, {ind} [, next ARP]]				V	Безусловный вызов подпрограммы с задержкой. Содержимое адресуемой ячейки программной памяти указывает на адрес подпрограммы. Увеличивает PC и записывает PC в стек, после этого загружает указанный адрес подпрограммы в памяти программ (символический или числовой) в PC. Модифицирует AR и ARP как указано. Можно указать переход с задержкой (CALLD), следующие два слова инструкции (две 1-словные инструкции или одна 2-словная инструкция) выбираются и выполняются перед вызовом
CC pma, cond1 [, cond2] [, ...]			V		Условный вызов. Если указанные условия выполняются, управление передается по pma. Не все комбинации условий допустимы
CC[D] pma, cond1 [, cond2] [, ...]				V	Условный вызов с задержкой. Если указанные условия выполняются, управление передается по pma. Не все комбинации условий допустимы. Если указан вызов с задержкой (CCD), то следующие два слова инструкции (две 1-словных инструкции или одна 2-словная инструкция) выбираются и выполняются перед вызовом

Продолжение таблицы А.4

1	2	3	4	5	6
CLRC control bit □□□□			V	V	Сброс управляющего бита. Установка указанного управляющего бита в логический 0. Маскируемые прерывания разрешаются немедленно после выполнения инструкции CLRC
CMPL		V	V	V	Дополнение аккумулятора. Дополнение содержимого аккумулятора до 1.
CMPR CM		V	V	V	Сравнение вспомогательного регистра с AR0. Сравнение содержимого вспомогательного регистра с AR0, базируемое на следующих случаях: Если CM = 00, проверка было ли AR(ARP) = AR0. Если CM = 01, проверка было ли AR(ARP) < AR0. Если CM = 10, проверка было ли AR(ARP) > AR0. Если CM = 11, проверка было ли AR(ARP) ≠ AR0. Если результат проверки истинен, то загружается 1 в TC, иначе загружается 0 в TC. Сравнение не влияет на тестируемый регистр. Устройство 1867ВЦ2Т: сравнивает содержимое вспомогательного регистра с ARCR
CNFD		V	V	V	Конфигурирование внутрикристальной памяти (блок B0) как память данных. Блок B0 внутрикристальной памяти конфигурируется как память данных. Блок B0 картрируется в память данных в ячейки 512h - 767h. Устройство 1867ВЦ2Т: блок B0 картируется в ячейки 512h–1023h
CNFP		V	V	V	Конфигурирование блока B0 внутрикристальной памяти как программную память. Конфигурирование блока B0 внутрикристальной памяти как программную память. Блок B0 картируется в ячейки программной памяти 65280h–65535h. Устройство 1867ВЦ2Т: блок B0 картируется в ячейки программной 65024h–65535h
CONF 2-битная константа		V			Конфигурирование блоков B0/B1/B2/B3 внутрикристальной как программной памяти. Конфигурирование блоков B0/B1/B2/B3 внутрикристальной как программной памяти. Для большей информации см. описание на 1867ВМ2

Продолжение таблицы А.4

1	2	3	4	5	6
CPL [#lk,] dma CPL [#lk,] {ind} [, next ARP]				V V	Сравнение DBMR или длинной константы со значением данных. Сравнение значений: если значения равны, то бит TC устанавливается в 1, иначе бит TC сбрасывается в 0
CRGT				V	Проверка соотношения ACC > ACCB. Сравнивает содержимое ACC с содержимым ACCB и после этого загружает большее значение в оба регистра и модифицирует бит переноса C в соответствии с результатом сравнения. Если содержимое ACC больше или равно содержимому ACCB , то бит переноса устанавливается в 1
CRLT				V	Проверка соотношения ACC < ACCB. Сравнивает содержимое ACC с содержимым ACCB и после этого загружает меньшее значение в оба регистра. Модифицирует бит переноса C в соответствии с результатом сравнения. Если содержимое ACC меньше содержимого ACCB , то бит переноса сбрасывается в 0
DINT	V	V	V	V	Запрещение прерывания. Запрещение всех прерываний. Устанавливает INTM в 1. Маскируемые прерывания немедленно запрещаются после выполнения инструкции DINT. DINT не запрещает немаскируемое прерывание RESET. DINT не влияет на IMR
DMOV dma DMOV {ind} [, next ARP]	V V	V V	V V	V V	Пересылка данных в память данных. Копирует содержимое адресуемой ячейки памяти данных в следующую старшую ячейку. DMOV пересылает данные только в блоках внутрикристальной памяти. Устройства 1867ВМ2, 1867ВЦ10Т и 1867ВЦ2Т: Блоки внутрикристальной памяти – это B0 (когда конфигурируется как память данных) B1 и B2
EINT	V	V	V	V	Разрешение прерывания. Разрешает все прерывания. Сбрасывает бит INTM в 0. Маскируемые прерывания разрешаются немедленно после выполнения инструкции INTM
EXAR				V	Обмен ACCB с ACC. Обмен содержимого ACC с содержимым ACCB
FORT 1-битная константа		V			Формат регистров последовательного порта. Загружает либо 0, либо 1. Если FO = 0, тогда регистры конфигурируются для приема/передачи 16-битных слов. Если FO = 1, то регистры конфигурируются для приема/передачи 8-битных байтов
IDLE		V	V	V	Ожидание пока не поступит прерывание. Заставляет приостанавливать выполнение программы и ждет, пока не поступит сброс или прерывание. Устройство сохраняет состояние ожидания до тех пор, пока не придет прерывание

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
IDLE2				V	Ожидание до прерывания – малоэнерго-потребляемый режим. Останавливает функциональную синхронизацию внутренних устройств. Это позволяет перевести устройство в режим низкого потребления. Инструкция IDLE2 форсирует перевод выполнения программы в состояние останова и ждет, пока будет получен сброс или немаскируемое прерывание
IN dma, PA IN {ind}, PA [, next ARP]	V V	V V	V V	V V	Ввод данных из порта. Читает 16-битные данные одного из внешних портов ввода - вывода. Устройство M1867BM1: это двухцикловая инструкция. Во время первого цикла адрес порта посылается на линии A2/PA2–A0/PA0; DEN переходит в низкий уровень, стробируя данные, которые помещены на шине данных D15–D0. Устройства 1867BM2: линия IS переходит в низкий уровень для индикации доступа к адресам ввода - вывода, а синхросигналы STRB#, RD/WR# и READY есть то же самое, что и для чтения внешней памяти данных. Устройства 1867ВЦ10Т и 1867ВЦ2Т: линия IS переходит в низкий уровень для индикации доступа к адресам ввода - вывода, а синхросигналы STRB, RD и READY есть то же самое, что и для чтения внешней памяти данных
INTR K			V	V	Программное прерывание. Передает управление адресу памяти программ, указанному в K (целое от 0 до 31). Эта инструкция позволяет программе выполнить любую подпрограмму прерывания. Ячейки векторов прерываний занимают два адреса (0h, 2h, 4h, ... , 3Eh), разрешая разместить два слова инструкции перехода в каждой из позиций
LAC dma [, shift] LAC {ind} [, shift [, next ARP]]	V V	V V	V V	V V	Загрузка аккумулятора со сдвигом. Загрузка аккумулятора содержимым адресуемой ячейки памяти со сдвигом. Если сдвиг указан, то выполняется левый сдвиг содержимого памяти перед загрузкой. Во время сдвига самые младшие биты заполняются 0, а старшие биты знаково расширяются, если SXM = 1
LACB				V	Загрузка аккумулятора ACCB. Загрузка содержимого буфера аккумулятора в аккумулятор

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867BЦ10T	1867BЦ2T	Описание
LACC dma [, shift1] LACC {ind} [, shift1 [, next ARP]] LACC #lk [, shift2]		V V V	V V V	V V V	Загрузка аккумулятора со сдвигом. Загрузка аккумулятора содержимым адресуемой ячейки памяти или 16-битной константой со сдвигом. Если сдвиг указан, то выполняется левый сдвиг содержимого памяти перед загрузкой. Во время сдвига самые младшие биты заполняются 0, а старшие биты знаково расширяются, если SXM = 1
LACK 8-битная константа	V	V	V	V	Загрузка аккумулятора короткой константой. Загрузка 8-битной константы в младшие 16 разрядов аккумулятора. 24 старших разряда аккумулятора сбрасываются
LACL dma LACL {ind} [, next ARP] LACL #k			V V V	V V V	Загрузка младшего слова и очистка старшего слова аккумулятора. Загрузка содержимого адресуемой памяти данных или расширенной нулями 8-битной константы в 16 младших разрядов аккумулятора. Старшие биты аккумулятора обнуляются. Данные обрабатываются как 16-битное беззнаковое число. Устройство 1867ВЦ10Т: константа 0 сбрасывает аккумулятор в 0 без расширения знака
LACT dma LACT {ind} [, next ARP]		V V	V V	V V	Загрузка аккумулятора со сдвигом, указанным в T регистре. Левый сдвиг содержимого адресуемой ячейки памяти данных на величину, указанную в четырех младших битах T регистра, загрузка результата в аккумулятор. Если указан сдвиг, то левый сдвиг выполняется перед загрузкой в аккумулятор. Во время сдвига младшие биты аккумулятора заполняются 0, старшие по порядку биты расширяются, если SXM = 1
LALK #lk [, shift]		V	V	V	Загрузка аккумулятора длинным непосредственным значением. Загрузка аккумулятора 16-битным непосредственным значением в аккумулятор. Если указан сдвиг, то константа сдвигается перед загрузкой в аккумулятор. Во время сдвига младшие по порядку биты аккумулятора заполняются 0, старшие по порядку биты расширяются, если SXM = 1

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
LAMM dma LAMM {ind} [, next ARP]				V V	Загрузка аккумулятора регистром, картрируемым в памяти. Загружает содержимое адресуемого картируемого в памяти регистра в младшее слово аккумулятора. 9 старших разрядов адреса памяти данных очищаются, независимо от текущего значения DP или 9 младших бит AR (ARP)
LAR AR, dma LAR AR, {ind} [, next ARP] LAR AR, #k LAR AR, #lk	V V	V V	V V V V	V V V V	Загрузка вспомогательного регистра. Устройства M1867BM1 и 1867BM2: загрузка содержимого адресуемой ячейки памяти данных в указанный вспомогательный регистр
LARK AR, 8-bit constant	V	V	V	V	Загрузка вспомогательного регистра коротким значением. Загружает 8-битную положительную константу в указанный вспомогательный регистр
LARP 1-битная константа LARP 3- битная константа	V	V	V	V	Загрузка в указатель вспомогательного регистра. Устройство M1867BM1: загружает 1-битную константу в указатель вспомогательного регистра (указывает AR0-AR1). Устройства 1867BM2, 1867ВЦ10Т и 1867ВЦ2Т: загружает 3-битную константу в указатель вспомогательного регистра (указывает AR0-AR7)
LDP dma LDP {ind} [, next ARP] LDP #k	V V	V V	V V V	V V V	Загружает в указатель страницы памяти данных. Устройство M1867BM1: загружает младшие биты адресуемой ячейки памяти данных в регистр DP. Все старшие по порядку биты игнорируются. DP = 0 определяет страницу 0 (слова 0–127) и DP = 1 определяет страницу 1 (слова 128 - 143/255). Устройства 1867BM2, 1867ВЦ10Т и 1867ВЦ2Т: загружает младшие биты адресуемой ячейки памяти данных в регистр DP или 9-битное непосредственное значение в DP. DP и 7 бит адреса памяти данных инструкции формирует 16-битный адрес памяти данных

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
LDPK 1-bit constant LDPK 9-bit constant	V	V	V	V	Загрузка указателя страницы памяти данных непосредственным значением. Устройство M1867BM1: загружает 1-битное значение в регистр DP . DP = 0 определяет страницу 0 (слова 0–127) и DP = 1 определяет страницу 1 (слова 128–143/255). Устройства 1867BM2, 1867ВЦ10Т и 1867ВЦ2Т: загружает младшие биты адресуемой ячейки памяти данных в регистр DP или 9-битное непосредственное значение в DP регистр. DP и 7-бит адреса памяти данных инструкции формируют 16-битный адрес памяти данных DP=8 указывает на внешнюю память данных. DP с 4 по 7 указывает на блоки B0 и B1 внутрикристальной памяти данных. Блок B2 размещается выше 32 слов в странице 0
LMMR dma, #lk LMMR {ind}, #lk [, next ARP]				V V	Загрузка картируемого в памяти данных регистра. Загружает содержимое картируемого в памяти регистра, указанного 7-битным значением прямой/косвенной адресацией в ячейку памяти данных, указанную длинной непосредственной адресацией. 9 старших по порядку разрядов адреса сбрасываются в 0, независимо от значения DP или 9 старших разрядов AR (ARP)
LPH dma LPH {ind} [, next ARP]		V V	V V	V V	Загружает старшее слово P регистра. Загружает содержимое ячейки адресуемой памяти данных в 16 старших по порядку разрядов P регистра. Младшие разряды P регистра не затрагиваются
LRLK AR, lk		V	V	V	Загрузка вспомогательного регистра 16-битным немедленным значением. Загрузка вспомогательного регистра 16-битным немедленным значением в указанный вспомогательный регистр
LST dma LST {ind} [, next ARP]	V V	V V	V V	V V	Загрузка статусного регистра. Загружает содержимое адресуемой ячейки памяти данных в ST (M1867BM1) или ST0 для (1867BM2, 1867ВЦ10Т и 1867ВЦ2Т)
LST #n, dma LST #n, {ind} [, next ARP]		V V	V V	V V	Загружает статусный регистр с номером n. Загружает содержимое адресуемой ячейки памяти данных в STn
LST1 dma LST1 {ind} [, next ARP]		V V	V V	V V	Загрузка статусного регистра 1. Загружает содержимое адресуемой ячейки памяти данных в ST1

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
LT dma LT {ind} [, next ARP]	V V	V V	V V	V V	Загрузка Т регистра. Загрузка содержимого адресуемой ячейки памяти данных в Т регистр (M1867BM1, 1867BM2 и 1867ВЦ10Т) или TREG0 (1867ВЦ2Т)
LTA dma LTA {ind} [, next ARP]	V V	V V	V V	V V	Загрузка Т регистра и аккумулятора предыдущим произведением. Загрузка содержимым ячейки адресуемой памяти в Т регистр (M1867BM1, 1867BM2 и 1867ВЦ10Т) или TREG0 (1867ВЦ2Т) и сложение содержимого Р регистра с аккумулятором. Устройства 1867BM2, 1867ВЦ10Т и 1867ВЦ2Т: перед сложением сдвиг содержимого Р регистра как указано в статусных битах PM
LTD dma LTD {ind} [, next ARP]	V V	V V	V V	V V	Загрузка Т регистра; суммирует предыдущее значение произведения к аккумулятору и пересылает данные. Загружает содержимое ячейки адресуемой памяти в Т регистр (M1867BM1, 1868BM2 и 1867ВЦ10Т) или TREG0 (1867ВЦ2Т), а также складывает содержимое Р регистра с аккумулятором и копирует содержимое указанной ячейки в следующий старший адрес (обе ячейки должны находиться в внутрикристальной памяти данных). Устройства 1867BM2, 1867ВЦ10Т и 1867ВЦ2Т: перед сложением содержимое Р регистра сдвигается на величину, указанную в статусных битах PM
LTP dma LTP {ind} [, next ARP]		V V	V V	V V	Загрузка Т регистра и сохранение Р регистра в аккумуляторе. Загружает содержимое ячейки адресуемой памяти данных в Т регистр (M1867BM1, 1867BM2 и 1867ВЦ10Т) или TREG0 (1867ВЦ2Т).Сохраняет содержимое Р регистра в аккумуляторе

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867BЦ10T	1867BЦ2T	Описание
LTS dma LTS {ind} [, next ARP]		V V	V V	V V	Загрузка P регистра и вычитание предыдущего произведения. Загружает содержимое ячейки адресуемой памяти данных в T регистр (M1867BM1, 1867BM2 и 1867BЦ10T) или TREG0 (1867BЦ2T). Сдвигает содержимое P регистра на величину, указанную в статусных битах PM и вычитает результат из аккумулятора
MAC pma, dma MAC pma, {ind} [, next ARP]		V V	V V	V V	Умножает с накоплением. Умножает значение памяти данных на значение программной памяти и добавляет предыдущее произведение (сдвинутое как определено битами PM) к аккумулятору
MACD dma, pma MACD pma, {ind} [, next ARP]		V V	V V	V V	Умножает и накапливает с пересылкой данных. Умножает значение памяти данных на значение программной памяти и добавляет предыдущее произведение (сдвинутое как определено в битах PM) к аккумулятору. Если адрес памяти данных в блоках B0, B1 или B2 внутрикристальной памяти, то копирует содержимое адреса в следующий более старший адрес
MADD dma MADD {ind} [, next ARP]				V V	Умножает с накоплением и пересылкой данных и динамической адресацией. Умножает значение памяти данных и добавляет предыдущее произведение (сдвинутое как определено статусными битами PM) в аккумулятор. Адрес программной памяти, содержащейся в BMAR позволяет динамически адресовать коэффициенты таблицы. MADD функционирует аналогично MADS с дополнительным перемещением данных внутри блоков внутрикристальной памяти
MADS dma MADS {ind} [, next ARP]				V V	Умножает с накоплением с динамической адресацией. Умножает значение памяти данных и добавляет предыдущее произведение (сдвинутое как определено статусными битами PM) в аккумулятор. Адрес программной памяти, содержащейся в BMAR, позволяет динамически адресовать коэффициенты таблицы
MAR dma MAR {ind} [, next ARP]	V V	V V	V V	V V	Модификация вспомогательного регистра. Модифицирует текущий AR или ARP как определяется. MAR действует как NOP в при индексной адресации

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
MPY dma MPY {ind} [, next ARP] MPY #k MPY #lk	V V	V V	V V V V	V V V V	Умножение. Устройства M1867BM1 и 1867BM2: умножает Т регистр на содержимое адресуемой ячейки памяти и помещает результат в Р регистр. Устройства 1867ВЦ10Т и 1867ВЦ2Т: умножает содержимое Т регистра (1867ВЦ10Т) или TREG0 (1867ВЦ2Т) на содержимое адресуемой памяти данных или на 13 или на 16-битное непосредственное значение. Помещает результат в Р регистр
MPYA dma MPYA {ind} [, next ARP]		V V	V V	V V	Умножает и накапливает предыдущее произведение. Умножает содержимое Т регистра (1867BM2, 1867ВЦ10Т) или TREG0 (1867ВЦ2Т) на содержимое ячейки адресуемой памяти данных. Помещает результат в Р регистр. Добавляет предыдущее произведение (сдвинутое как определено статусными битами PM) к аккумулятору
MPYK 13-bit constant	V	V	V	V	Умножение на константу. Умножает содержимое Т регистра (1867BM2, 1867ВЦ10Т) или TREG0 (1867ВЦ2Т) на знаковую 13-битную константу и помещает результат в Р регистр
MPYS dma MPYS {ind} [, next ARP]		V V	V V	V V	Умножает и вычитает предыдущее произведение. Умножает содержимое Т регистра (1867BM2, 1867ВЦ10Т) или TREG0 (1867ВЦ2Т) на содержимое адресуемой ячейки памяти данных и помещает результат в Р регистр. Вычитает предыдущее произведение (сдвинутое как указано статусными битами PM) из аккумулятора
MPYU dma MPYU {ind} [, next ARP]		V V	V V	V V	Умножение без знака. Беззнаковое умножение содержимого Т регистра (1867BM2, 1867ВЦ10Т) или TREG0 (1867ВЦ2Т) на беззнаковое содержимое адресуемой ячейки памяти данных. Результат помещает в Р регистр
NEG		V	V	V	Дополнение до 2 аккумулятора. Дополнение до 2 содержимого аккумулятора
NMI			V	V	Немаскируемое прерывание. Устанавливает программный счетчик на вектор 24h немаскируемого прерывания. NMI оказывает тот же эффект, что и аппаратное немаскируемое прерывание
NOP	V	V	V	V	Нет операции. Не исполняет операцию
NORM NORM {ind}		V V	V V	V V	Нормализация содержимого аккумулятора. Нормализует знаковое число в аккумуляторе

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
OPL [#lk,] dma OPL [#lk,] {ind} [, next ARP]				V V	OR с DBMR или непосредственным значением. Если указано непосредственное значение, то оно OR со значением ячейки памяти данных, иначе вторым операндом операции OR является DBMR. Результат записывается обратно в ячейку памяти данных, хранящей первый операнд
OR dma OR {ind} [, next ARP] OR #lk [, shift]	V V	V V	V V V	V V V	OR с аккумулятором. Устройства M1867BM1 и 1867BM2: OR 16 младших бит аккумулятора с содержимым адресуемой памяти данных. 16 старших разрядов аккумулятора OR с 0. Устройства 1867ВЦ10Т и 1867ВЦ2Т: OR 16 младших бит аккумулятора или 16-битного непосредственного значения с содержимым адресуемой ячейки памяти данных. Если указан сдвиг, то выполняется левый сдвиг перед операцией OR. Самые левые биты и самые правые по порядку биты сдвинутого значения обрабатываются как 0
ORB				V	OR ACCB с аккумулятором. OR содержимого ACCB с содержимым аккумулятора. Размещает результат в аккумуляторе
ORK #lk [, shift]		V	V	V	OR непосредственного значения с аккумулятором со сдвигом. OR 16-битного непосредственного значения с содержимым аккумулятора. Если указывается сдвиг, то выполняется левый сдвиг перед операцией OR. Самые левые биты ниже и самые правые по порядку биты выше сдвинутого значения обрабатываются как 0
OUT dma, PA OUT {ind}, PA [, next ARP]	V V	V V	V V	V V	Вывод данных в порт. Записывает 16-битное значение из памяти данных в порт ввода - вывода. Устройство M1867BM1: на первом цикле инструкции помещается адрес порта на линии адреса A2/PA2–A0/PA0. Во время этого цикла WREN# переходит в низкий уровень и данные помещаются на шину данных D15–D0. Устройства 1867BM2, 1867ВЦ10Т и 1867ВЦ2Т: линия IS переходит в 0 для индикации доступа к пространству ввода - вывода. Сигналы на линиях STRB#, RD/WR# и READY синхронизируют аналогично обращению к внешней памяти данных

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
PAC	V	V	V	V	Загрузка аккумулятора P регистром. Загрузка содержимого P регистра в аккумулятор. Устройства 1867BM2, 1867ВЦ10Т и 1867ВЦ2Т: перед загрузкой сдвигает P регистр как специфицировано в статусных битах PM
POP	V	V	V	V	Чтение верхушки стека в младшее слово аккумулятора. Копирует содержимое верхушки стека в младшие 12 (M1867BM1) или 16 (1867BM2, 1867ВЦ10Т и 1867ВЦ2Т) разрядов аккумулятора и после этого выталкивает стек на один уровень. Старшие 16 бит аккумулятора обнуляются
POPD dma POPD {ind} [, next ARP]		V V	V V	V V	Чтение верхушки стека в память данных. Пересылает значение на верхушке стека в адресуемую ячейку памяти данных и после этого выталкивает стек на один уровень
PSHD dma PSHD {ind} [, next ARP]		V V	V V	V V	Записывает адресуемую ячейку памяти данных в верхушку стека. Копирует адресуемую ячейку памяти данных в стек. Стек опускается вниз на один уровень перед копированием этого значения.
PUSH	V	V	V	V	Запись младшего слова аккумулятора в стек. Копирует 12 (M1867BM1) или 16 (1867BM2, 1867ВЦ10Т и 1867ВЦ2Т) младших бит аккумулятора в верхушку аппаратного стека. Стек опускается вниз на один уровень перед тем как значение копируется
RC		V	V	V	Сброс бита переноса. Сбрасывает статусный бит C в 0
RET	V	V	V		Возврат из подпрограммы. Копирует содержимое верхушки стека в PC и поднимает стек на один уровень
RET[D]				V	Возврат из подпрограммы с задержкой. Копирует содержимое верхушки стека в PC и поднимает стек на один уровень. Если указана задержка перехода (RETD), то следующие два слова (две 1-словные или одна 2-словная инструкция) выбираются и выполняются перед возвратом
RETC cond1 [, cond2] [, ...]			V		Условный переход. Если указанные условия имеют место, то RETC выполняет стандартный возврат. Не все комбинации условий допустимы

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867BЦ10T	1867BЦ2T	Описание
RETC[D] cond1 [, cond2] [, ...]				V	Условный возврат с задержкой. Если указанные условия имеют место, то RETC выполняет стандартный возврат. Не все комбинации условий допустимы. Если указана задержка перехода (RETCD), то следующие два слова (две однословные или одна двухсловная инструкция) выбираются и выполняются перед возвратом
RETE				V	Разрешение прерывания и возврат из прерывания. Копирует содержимое верхушки стека в PC и поднимает стек на один уровень вверх. RETE автоматически очищает бит разрешения глобального прерывания и читает теневые регистры (сохраненные когда было прерывание) назад в их соответствующие стратегические регистры. Следующие регистры имеют «тень»: ACC, ACCB, PREG, ST0, ST1, PMST, ARCR, INDX, TREG0, TREG1, TREG2
RETI				V	Возврат из прерывания. Копирует содержимое верхушки стека в PC и поднимает стек на один уровень. RETI также читает значения в теневых регистрах (запомненные во время прерывания) назад в их соответствующие стратегические регистры. Следующие регистры имеют «тень»: ACC, ACCB, PREG, ST0, ST1, PMST, ARCR, INDX, TREG0, TREG1, TREG2
RFSM		V			Сброс режима фреймовой синхронизации последовательного порта. Сбрасывает статусный бит FSM в 0
RHM		V		V	Сброс режима удержания. Сброс статусного бита HM в 0
ROL		V	V	V	Вращение аккумулятора влево. Вращение аккумулятора на один бит влево
ROLB				V	Вращение ACCB и аккумулятора влево. Вращение ACCB и аккумулятора влево на один бит
ROR		V	V	V	Вращение аккумулятора вправо. Вращение аккумулятора на один бит вправо
RORB				V	Вращение ACCB и аккумулятора вправо. Вращение ACCB и аккумулятора вправо на один бит
ROVM	V	V	V	V	Сброс режима переполнения. Сбрасывает статусный бит OVM в 0; это запрещает режим переполнения

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
RPT dma RPT {ind} [, next ARP] RPT #k RPT #lk		V V	V V V V	V V V V	Повторение следующей инструкции. Устройство 1867ВМ2: загружает 8 младших бит адресуемого значения в RPTC. Инструкция, следующая за RPT, выполняет число раз, которое указано в RPTC+1. Устройства 1867ВЦ10Т и 1867ВЦ2Т: загружает 8 младших бит адресуемого значения или 16 бит непосредственного значения в RPTC. Инструкция, следующая за RPT, выполняет число раз, которое указано в RPTC+1
RPTB pma				V	Повторение блока. RPTB повторяет блок инструкций сколько раз указанное в картируемой в памяти. BRCR должен загружаться перед выполнением RPTB
RPTK #k		V	V	V	Повторение инструкции как указано непосредственным значением. Загружает 8-битное непосредственное значение в RPTC. Инструкция, следующая за RPTK, выполняется количество раз, которое указано в RPTC + 1
RPTZ #lk				V	Повторение с предварительной очисткой аккумулятора и P регистра. Очищает аккумулятор и регистр произведения и повторяет инструкцию, следующую за RPTZ n раз, где n = lk +1
RSXM		V	V	V	Сброс режим расширения знака. Сбрасывает статусный бит SXM в 0; это запрещает расширение знака при сдвиге значений данных для следующих инструкций: ADD, ADDT, ADLK, LAC, LACT, LALK, SBLK, SUB и SUBT
RTC		V	V	V	Сбрасывает Тест/Контрольный флаг. Сбрасывает статусный бит TC в 0
RTXM		V			Сброс режима передачи последовательного порта. Сбрасывает статусный бит TXM в 0; это конфигурирует секцию передачи в режим, когда передатчик управляется FSX
RXF		V	V	V	Сброс флага XF. Сброс вывода XF и статусного бита XF в 0
SACB				V	Сохранение аккумулятора в ACCB. Копирует содержимое аккумулятора в ACCB

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
SACH dma [, shift] SACH {ind} [, shift [, next ARP]]	V V	V V	V V	V V	Сохранение старшего слова аккумулятора со сдвигом. Копирует содержимое аккумулятора в сдвигатель. Сдвиг всего содержимого на 0, 1 или 4 бита (M1867BM1) или от 0 до 7 бит (1867BM2, 1867ВЦ10Т и 1867ВЦ2Т) и копирование 16 старших бит сдвинутого значения в ячейку адресуемой памяти данных. Аккумулятор не изменяется
SACL dma SACL dma [, shift] SACL {ind} [, shift [, next ARP]]	V V	 V V	 V V	 V V	Сохранение младшего слова аккумулятора со сдвигом. Устройство M1867BM1: сохраняет 16 младших бит аккумулятора в ячейке адресуемой памяти. Величина сдвига, равная 0, должна указываться, если ARP должен меняться. Устройства 1867BM2, 1867ВЦ10Т и 1867ВЦ2Т: Сохраняет 16 младших слов аккумулятора в ячейке адресуемой памяти данных. Если указан сдвиг, то сдвигает содержимое аккумулятора перед сохранением. Значение сдвига равно 0, 1 или 4 бита или от 0 до 7 бит (1867BM2, 1867ВЦ10Т и 1867ВЦ2Т)
SAMM dma SAMM {ind} [, next ARP]				V V	Сохранение аккумулятора в картируемом памяти данных регистре. Сохраняет младшее слово аккумулятора в адресуемом в памяти данных регистре. Верхние 9 бит адреса данных сбрасываются независимо от текущего значения DP или 9 старших бит текущего значения AR (ARP)
SAR AR, dma SAR AR, {ind} [, next ARP]	V V	V V	V V	V V	Сохранение вспомогательного регистра. Сохраняет содержимое вспомогательного регистра в ячейке адресуемой памяти данных
SATH				V	Сдвиг аккумулятора как указано в T регистре. Если 4 бит TREG1 равен 1, то выполняется сдвиг аккумулятора вправо на 16 бит, иначе аккумулятор не меняется
SATL				V	Сдвиг младшего слова аккумулятора как указано в T регистре. Сдвиг аккумулятора вправо на значение, указанное в 4 младших битах TREG1
SBB				V	Вычитание ACCB из аккумулятора. Вычитание содержимого ACCB из аккумулятора. Результат сохраняется в аккумуляторе, ACCB не изменяется

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
SBBB				V	Вычитание ACCB из аккумулятора с заемом. Вычитание содержимого ACCB и логической инверсии бита переноса C из аккумулятора. Результат сохраняется в аккумуляторе, ACCB не изменяется. Сбрасывает бит переноса C, если результат генерирует заем
SBLK #lk [, shift]		V	V	V	Вычитание непосредственного значения из аккумулятора. Вычитает непосредственное значение из аккумулятора. Если указан сдвиг, то сдвигает влево значение перед вычитанием. Во время сдвига левые по порядку биты заполняются 0, а правые биты знаково расширяются, если SXM = 1
SBRK #k		V	V	V	Вычитание из вспомогательного регистра короткого непосредственного значения. Вычитание 8 бит непосредственного значения из определенного вспомогательного регистра
SC		V	V	V	Установка бита переноса C. Установка статусного бита переноса C в 1
SETC control bit			V	V	Установка управляющего бита. Устанавливает указанный управляющий бит в 1. Маскируемые прерывания запрещаются немедленно после выполнения инструкции SETC
SFL		V	V	V	Сдвиг аккумулятора влево. Сдвиг содержимого аккумулятора влево на 1 бит
SFLB				V	Сдвиг ACCB и аккумулятора влево. Сдвиг аккумулятора и ACCB (с конкантенацией) влево на 1 бит. Младшие биты ACCB сбрасываются в 0, а старшие биты ACCB сдвигаются в бит переноса
SFR		V	V	V	Сдвиг аккумулятора вправо. Сдвиг содержимого аккумулятора вправо на 1 бит. Если SXM = 1, то инструкция SFR создает арифметический сдвиг вправо. Если SXM = 0, то инструкция формирует логический сдвиг вправо
SFRB				V	Сдвиг ACCB и аккумулятора вправо. Сдвигает конкантенацию аккумулятора и ACCB вправо на 1 бит. Младшие биты ACCB сдвигаются в бит переноса. Если SXM = 1, то инструкция выполняет арифметический сдвиг, если SXM = 1, то инструкция выполняет логический сдвиг
SFSM		V			Установка режима фреймовой синхронизации последовательного порта. Установка статусного бита FSM в 1
SHM		V		V	Установка Hold режима. Установка статусного бита HM в 1

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
SMMR dma, #lk SMMR {ind}, #lk [, next ARP]				V V	Сохранение значения, картируемого в памяти данных регистра. Сохранение значения картируемого в памяти данных регистра, указанного 7 младшими битами адреса памяти данных в ячейку, адресуемую длинной непосредственной константой. 9 старших бит адреса памяти данных сбрасываются независимо от текущего значения DP или старших 9 бит AR(ARP)
SOVM	V	V	V	V	Установка режима переполнения. Установка статусного бита OVM в 1. Это разрешает режим переполнения (инструкция ROVM очищает бит OVM)
SPAC	V	V	V	V	Вычитание P регистра из аккумулятора. Вычитает содержимое P регистра из аккумулятора. Устройства 1867BM2, 1867ВЦ10Т, и 1867ВЦ2Т: перед вычитанием выполняется сдвиг содержимого P регистра как указано в статусных битах PM
SPH dma SPH {ind} [, next ARP]		V V	V V	V V	Сохранение старшего слова P регистра. Сохраняет старшее слово P регистра (сдвинутого как указано в статусных битах PM) в ячейку адресуемой памяти данных
SPL dma SPL {ind} [, next ARP]		V V	V V	V V	Сохранение младшего слова P регистра. Сохраняет младшее слово P регистра (сдвинутого как указано в статусных битах PM) в ячейку адресуемой памяти данных
SPLK #lk, dma SPLK #lk, {ind} [, next ARP]			V	V V	Сохранение длинной непосредственной константы. Записывает целое 16-битное слово в ячейку памяти. Параллельное логическое устройство (PLU) поддерживает эту манипуляцию с битами независимо от АЛУ. Аккумулятор не изменяется
SPM 2-битная константа		V	V	V	Установка режима сдвига P регистра. Копирует 2-битное непосредственное значение в поле PM статусного регистра ST1. Это управляет сдвигом P регистра, как показано ниже: PM = 00 - выход умножителя не сдвигается. PM = 01 – выход умножителя сдвигается на 1 бит влево и заполняется 0. PM = 10 - PM = 01 – выход умножителя сдвигается на 4 бита влево и заполняется 0. PM = 11 - PM = 01 – выход умножителя сдвигается на 6 бит вправо и младшие биты теряются

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
SQRA dma SQRA {ind} [, next ARP]		V V	V V	V V	Вычисление квадрата и накопление предыдущего произведения. Добавляет содержимое P регистра (сдвинутое как указано статусными битами PM) к аккумулятору. После этого загружает содержимое ячейки адресуемой памяти данных в T регистр (1867BM2, 1867ВЦ10Т) или регистр, вычисляет значение квадрата и сохраняет результат в P регистре
SQRS dma SQRS {ind} [, next ARP]		V V	V V	V V	Вычисление квадрата и вычитание предыдущего произведения. Вычитает содержимое P регистра (сдвинутое как указано в статусных битах PM) из аккумулятора. Загружает содержимое ячейки адресуемой памяти данных в T регистр (1867BM2, 1867ВЦ10Т) или регистр TREG0 (1867ВЦ2Т), вычисляет квадрат и сохраняет результат в P регистре
SST dma SST {ind} [, next ARP	V V	V V	V V	V V	Сохранение статусного регистра. Сохраняет содержимое статусного регистра ST (M1867BM1) или ST0 (1867BM2, 1867ВЦ10Т и 1867ВЦ2Т) в адресуемой ячейке памяти данных
SST #n, dma SST #n, {ind} [, next ARP]			V V	V V	Сохранение статусного регистра n. Сохраняет STn в памяти данных
SST1dma SST1 {ind} [, next ARP]		V V	V V	V V	Сохранение статусного регистра ST1. Сохраняет ST1 в памяти данных
SSXM		V	V	V	Установка режима расширения знака. Устанавливает статусный бит SXM в 1; это разрешает режим расширения знака
STC		V	V	V	Установка флага тест/управление. Установка флага TC в 1
STXM		V			Установка режима передачи последовательного порта. Установка статусного бита TXM в 1

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
SUB dma [, shift] SUB {ind} [, shift [, next ARP]] SUB #k SUB #lk [, shift2	V V	V V	V V V V	V V V V	Вычитание из аккумулятора со сдвигом. Устройства M1867BM1 и 1867BM2: вычитает содержимое адресуемой ячейки памяти данных из аккумулятора. Если указан сдвиг, то значение сдвигается влево перед вычитанием. Во время сдвига младшие биты заполняются 0, а старшие биты знаково расширяются, если SXM = 1. Устройства 1867ВЦ10Т и 1867ВЦ2Т: вычитание содержимого адресуемой ячейки памяти данных или 8/16-битной константы из аккумулятора. Если указан сдвиг, то выполняется левый сдвиг перед вычитанием. Во время сдвига младшие биты заполняются 0, а старшие биты знаково расширяются, если SXM = 1
SUBB dma SUBB {ind} [, next ARP]		V V	V V	V V	Вычитание из аккумулятора с заемом. Вычитает содержимое адресуемой ячейки памяти данных и значения бита переноса C из аккумулятора. Бит переноса изменяется обычным образом
SUBC dma SUBC {ind} [, next ARP]	V V	V V	V V	V V	Условное вычитание. Выполняет вычитание с условием. Инструкция SUBC может использоваться для деления
SUBH dma SUBH {ind} [, next ARP]	V V	V V	V	V V	Вычитание из старшего слова аккумулятора. Вычитает содержимое ячейки адресуемой памяти данных из 16 старших разрядов аккумулятора. Младшие 16 бит аккумулятора не затрагиваются (меняются)
SUBK #k		V	V	V	Вычитание из аккумулятора короткого непосредственного значения. Вычитает 8-битное непосредственное значение из аккумулятора. Данные обрабатываются как 8-битное положительное число; расширение знака блокируется
SUBS dma SUBS {ind} [, next ARP]	V V	V V	V V	V V	Вычитание из аккумулятора с блокированием расширения знака. Вычитает содержимое ячейки адресуемой памяти данных из аккумулятора. Данные обрабатываются как 16-битное число без знака; расширение знака блокируется

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
SUBT dma SUBT {ind} [, next ARP]		V V	V V	V V	Вычитание из аккумулятора со сдвигом, указанным в Т регистре. Выполняет левый сдвиг значения ячейки памяти данных, как указано в младших разрядах Т регистра (1867BM2, 1867ВЦ10Т) или TREG1 (1867ВЦ2Т) и вычитает результат из аккумулятора. Если указан сдвиг, то сдвиг содержимого ячейки памяти данных выполняется перед вычитанием. Во время сдвига младшие разряды заполняются 0, а старшие разряды знаково расширяются, если SXM = 1
SXF		V	V	V	Установка внешнего флага XF. Устанавливает вывод XF и статусный бит XF в 1
TBLR dma TBLR {ind} [, next ARP]	V V	V V	V V	V V	Чтение таблицы. Передает слово из памяти программ в память данных. Адрес программной памяти определяется 12 (M1867BM1) или 16 (1867BM2, 1867ВЦ10Т и 1867ВЦ2Т) младшими битами аккумулятора
TBLW dma TBLW {ind} [, next ARP]	V V	V V	V V	V V	Запись таблицы. Передает слово из памяти данных в программную память. Адрес программной памяти определяется 12 (M1867BM1) или 16 (1867BM2, 1867ВЦ10Т и 1867ВЦ2Т) младшими битами аккумулятора
TRAP		V	V	V	Программное прерывание. Инструкция TRAP – это программное прерывание, которое передает управление в ячейку 30h (1867BM2) или 22h (1867ВЦ10Т, 1867ВЦ2Т) программной памяти и записывает PC + 1 в аппаратный стек. Инструкция по адресу 30h или 22h может содержать инструкцию перехода для передачи управления в подпрограмму обработки TRAP. Запись PC + 1 в стек разрешает инструкции RET читать значение PC возврата

Продолжение таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867BЦ10T	1867BЦ2T	Описание
XC n, cond1 [, cond2] [, ...]				V	Условное выполнение. Выполняет условно следующие n слов инструкции, где $1 \leq n \leq 2$. Не все комбинации условий значимы
XOR dma XOR {ind} [, next ARP] XOR #lk [, shift]	V V	V V	V V V	V V V	Исключающее ИЛИ с аккумулятором. Устройства M1867BM1 и 1867BM2: исключающее ИЛИ адресуемой ячейки памяти данных с 16 младшими битами аккумулятора. Старшие разряды не затрагиваются. Устройства 1867BЦ10T и 1867BЦ2T: исключающее ИЛИ содержимого адресуемой ячейки памяти данных или 16-битным непосредственным значением с аккумулятором. Если указан сдвиг, то выполняется левый сдвиг перед выполнением операции исключающего ИЛИ. Младшие биты и старшие биты после сдвига обрабатываются как 0
XORB				V	Исключающее ИЛИ ACCB с аккумулятором. Исключающее ИЛИ содержимого аккумулятора с содержимым ACCB. Результат помещается в аккумулятор
XORK #lk [, shift]		V	V	V	Исключающее ИЛИ непосредственного значения с аккумулятором со сдвигом. Исключающее ИЛИ 16-битного непосредственного значения. Если указывается сдвиг, то выполняется левый сдвиг перед выполнением операции исключающего ИЛИ. Младшие разряды и старшие разряды сдвинутого значения обрабатываются как 0
XPL [#lk,] dma XPL [#lk ,] {ind} [, next ARP]				V V	Исключающее ИЛИ длинного непосредственного значения или DBMR со значением адресуемой памяти данных. Если указано длинное непосредственное значение, то выполняет исключающее ИЛИ со значением адресуемой ячейки памяти данных; иначе исключающее ИЛИ содержимого DBMR со значением адресуемой ячейки памяти данных. Аккумулятор не затрагивается

Окончание таблицы А.4

Синтаксис	M1867BM1	1867BM2	1867ВЦ10Т	1867ВЦ2Т	Описание
ZAC	V	V	V	V	Обнуление аккумулятора. Очищает содержимое аккумулятора в 0
ZALH dma ZALH {ind} [, next ARP]	V V	V V	V V	V V	Обнуление младшего слова аккумулятора и загрузка старшего слова аккумулятора. Очищает 16 младших бит аккумулятора в 0 и загружает содержимое адресуемой ячейки памяти данных в 16 старших разрядов аккумулятора
ZALR dma ZALR {ind} [, next ARP]		V V	V V	V V	Обнуление младшего слова аккумулятора и загрузка старшего слова аккумулятора с округлением. Загружает содержимое адресуемой ячейки памяти данных в 16 старших слов аккумулятора. Значение округляется 1/2 младшего бита, что означает 15 бит аккумулятора (14-00 сбрасываются в 0, а 15 бит устанавливается в 1).
ZALS dma ZALS {ind} [, next ARP]	V V	V V	V V	V V	Обнуление аккумулятора, загрузка младшего слова аккумулятора с запрещением расширения знака. Загружает содержимое адресуемой ячейки памяти данных в 16 младших разрядов аккумулятора. 16 старших разрядов аккумулятора обнуляются. Данные обрабатываются как без знаковое число
ZAP				V	Обнуление аккумулятора и регистра произведения. Аккумулятор и регистр произведения обнуляются. Инструкция ZAP повышает скорость подготовки для повторения умножения/накопления
ZPR				V	Обнуление регистра произведения. Регистр произведения очищается

Приложение Б
(обязательное)

Использование эмулятора ВЦ10Т

Эмулятор-ВЦ10Т является программно-аппаратным комплексом для отладки микросхем 1867ВЦ10Т из интегрированной среды разработки Code Composer version: 4.10.36 (Texas Instruments) через тестовый порт JTAG (стандарт IEEE 1149.1) микросхемы.

Техническая часть эмулятора ВЦ10Т включает:

- Устройство JEM-16-JTAG.
- Исходные тексты управляющих программ микроконтроллера JEM-16-JTAG.
- Файл с программой управления микроконтроллера JEM-16-JTAG.
- Инструкция (паспорт) по эксплуатации устройства JEM-16-JTAG.
- Кабель для подключения через порт USB к персональному компьютеру.
- Конструкторскую документацию на устройство JEM-16-JTAG.
- Драйвер.
- Исходный текст драйвера.
- Описание программы.

Б.1 Конструкция 14-выводного разъема эмулятора ВЦ10Т

Целевые JTAG устройства поддерживают эмуляцию через соответствующий порт эмуляции. Этот порт доступен из эмулятора и обеспечивает функции эмуляции, которые являются расширением функций, специфицированных IEEE 1149.1.

Конструкция эмулятора ВЦ10Т состоит из аппаратной части (устройство JEM-16-JTAG) и программной части (драйвера).

Б.1.1 Аппаратная часть эмулятора ВЦ10Т

Отладочное устройство JEM-16-JTAG выполнено в пластиковом корпусе. Оно имеет с одной стороны корпуса разъем USB для подключения устройства JEM-16-JTAG к персональному компьютеру и, с другой стороны, разъем для подключения микросхемы 1867ВЦ10Т. Электропитание JEM-16-JTAG осуществляется от USB-порта персонального компьютера. Устройство JEM-16-JTAG имеет средства индикации в виде трех светодиодов, отображающих:

- наличие питания на плате JEM-16-JTAG;
- наличие сигнала сброса TRST# на плате JEM-16-JTAG;
- режим выполнения пользовательской программы: ход или останов.

Отладочное устройство JEM-16-JTAG обеспечивает:

- соответствие требованиям стандарта IEEE 1149.1;
- обмен данными с персональным компьютером через порт USB 2.0 в режимах Full Speed и High Speed;
- обмен данными с платой пользователя с помощью плоского кабеля и отладочного разъема типа IDC-14F на тактовой частоте сигнала TCK в пределах 10 – 12 МГц;

- обмен данными и управляющей информацией с блоком отладки On-Chip-Debug (OCD) микросхемы 1867ВЦ10Т через отладочный JTAG-разъем;

- доступ к ресурсам процессора и периферии при помощи OCD;

- выходное напряжение низкого уровня при токе нагрузки $\geq$ 2 мА, не более 0,4 В;

- выходное напряжение высокого уровня при токе нагрузки $\geq$ 2 мА, не менее 2,4 В;

- максимальный ток нагрузки для высокого и низкого уровней сигнала по входу TDO, не более 2 мА;

- следующий порядок передачи битов по входу TDI – младший разряд передается первым;

- подключение к плате пользователя с помощью плоского кабеля и отладочного разъема типа IDC-14F;

- стойкость к воздействию статического электричества с потенциалом не менее 2000 В.

Для соединения с эмулятором целевая система должна иметь 14-выводной разъем (две колонки по семь выводов) с сигналами эмуляции, которые подсоединены к разъему устройства JEM-16-JTAG в соответствии с таблицей Б.1 и показаны на рисунке Б.1.

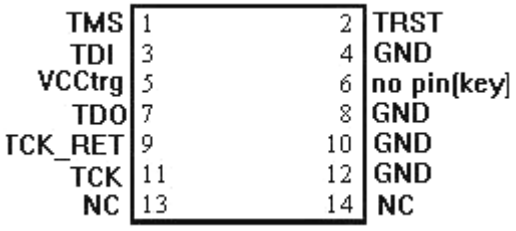


Рисунок Б.1 – Расположение сигналов в 14-выводном разъеме

Для предотвращения неправильного соединения вывод 6 разъема имеет ключ (обозначен - вывод 6).

Таблица Б.1 – Контакты отладочного разъема устройства JEM-16-JTAG

Сигнал	Описание	Тип вывода на эмуляторе	Тип вывода на целевой системе
GND	Общий на плате пользователя, соединяется с выводами #GND2 микросхемы (общий буферов ввода/вывода).	-	-
VCCtrg	Напряжение питания +3.3 В +/- 5 % на плате пользователя, соединяется с выводами #VCC2 (напряжение питания буферов ввода/вывода) микросхемы	I	O
TCK	Вывод тактовой частота JTAG (вход микросхемы, pull-up 75 кОм)	O	I
TCK_RET	Test clock return. Входной синхроимпульс для эмулятора	I	O

Окончание таблицы Б.1

Сигнал	Описание	Тип вывода на эмуляторе	Тип вывода на целевой системе
TDI	Вывод записи последовательных данных в TAP-контроллер (вход микросхемы, pull-up 75 кОм)	O	I
TDO	Вывод чтения последовательных данных из TAP-контроллера (выход микросхемы)	I	O
TMS	Сигнал изменения состояния TAP-контроллера (вход микросхемы, pull-up 75 кОм)	O	I
TRST#	Сигнал сброса тестового порта JTAG (вход микросхемы, pull-down 75 кОм)	O	I
NC	Не подключается		

Примечания

1 I – вход, O – выход, I/O – вход/выход.

2 Не используйте подтягивающий к высокому уровню резистор на TRST#. Этот вывод имеет внутрикристальный подтягивающий резистор к низкому уровню для уменьшения возможных наводок. Вывод TRST# может быть не подключен. В сильно шумящем окружении может потребоваться подключение дополнительного резистора к низкому уровню.

Б.1.2 Программная часть эмулятора ВЦ10Т

Драйвер обеспечивает поддержку следующих отладочных функций с микросхемой/и 1867ВЦ10Т из интегрированной среды разработки Code Composer version: 4.10.36 (Texas Instruments):

- загрузку программы в адресуемую память (File->Load Program);
- перезагрузку программы в адресуемую память (File->Reload Program);
- загрузку данных в страницу Data (File->Data->Load);
- сохранение данных из страницу Data (File->Data->Save);
- передавать или принимать данные в/из микросхем/ы 1867ВЦ10Т из/в файл хост-машины (File->File I/O);
- редактирование памяти (Edit->Edit Memory);
- редактирование регистров (Edit->Edit Registers);
- чтение содержимого регистров процессора (View->CPU Registers-> CPU Register);

чтение содержимого регистров процессора (View->CPU Registers-> Peripheral Reg);

- чтение содержимого адресуемой памяти (View->Memory);
- установку аппаратных Breakpoints (Debug->Breakpoints);
- установку программных Breakpoints;

- установку Watch Breakpoints;
- установку Probe Point (Debug->Probe Point);
- установку пошагового исполнения программы внутри функции Step Into (Debug->Step Into);
- установку пошагового исполнения программы на функциях Step Over (Debug->Step Over);
- установку пошагового исполнения программы при выходе из функции Step Out (Debug->Step Out);
- запуск процессора на выполнение программы с остановкой на курсоре в окне Dis-Assembler (Debug->Run to Cursor);
- продолжение выполнения программы после останова (Debug->Animate);
- запуск процессора на выполнение программы (Debug->Run);
- остановку выполнения программы процессором (Debug->Halt);
- установку окна Watch Window (Debug Toolbar ->View Watch Window);
- быструю установку окна Watch Window (Debug Toolbar->View Quick Watch Window);
- установку окна call stack (Debug Toolbar ->View Call Stack);
- (Debug Toolbar->View Memory Window);
- (Debug-> Toolbar->View CPU Register Window);
- запуск процессора на выполнение программы с блокировкой Breakpoints (Debug->Run Free);
- установку (Debug->Multiple Operations);
- сброс процессора (Debug->Reset DSP);
- перезапуск процессора (Debug->Restart);
- установку временной Breakpoint на символе Main (Debug->Go to Main);
- установку режима работы процессора в реальном времени (Debug->Real Time Mode);
- мультипроцессорный режим отладки.

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Подп.	Дата
	изме-ненных	заме-нен-ных	новых	анну-лиро-ванных				
-	-	-	все	-	868	КФДЛ.8992		25.12.15
1	1	-	-	-	-			28.12.15