МИКРОСХЕМА ИНТЕГРАЛЬНАЯ

1867ВМ7Т

Руководство пользователя

2013

Содержание

Введение

ИС 1867ВМ7Т представляет собой однокристальный высокоскоростной 16-разрядный процессор обработки сигналов с фиксированной запятой по типу 1867ВМ2 (TMS320C25).

Микросхема выполнена на основе модифицированной Гарвардской архитектуры, в которой память данных и память для хранения программного кода располагаются в отдельных адресных областях. Это позволяет совместить в одном такте выборку текущей и выполнение предыдущей инструкций. Предусмотрены механизмы для обеспечения обмена данными между двумя областями. С точки зрения пользователя, области памяти для хранения данных и программ используют одни и те же адресную и информационную шины, чтобы обеспечить доступ к обеим областям при минимальном числе выводов корпуса. С точки зрения внутренней организации, архитектура ИС 1867ВМ7Т увеличивает вычислительную мощность с помощью сохранения двух раздельных шинных структур для программ и данных, чтобы обеспечить выполнение команд с максимальной скоростью.

ИС 1867ВМ7Т осуществляет трехуровневую конвейерную обработку с раздельным уровнем для декодирования команд. Конвейер (первый уровень – выбор команды; второй уровень – декодирование команды; третий уровень – выполнение команды) по существу прозрачен для пользователя за исключением некоторых случаев, где конвейер должен быть разорван (например, при командах ветвления). Микросхема выполняет большинство команд, связанных с обращением к внешней памяти, в одном машинном цикле, если используется достаточно быстрая память. ИС также может работать с более медленными внешними ЗУ или с периферийными устройствами, используя сигнал READY (готовность). В этих случаях соответствующие команды становятся многоцикловыми.

В ИС 1867ВМ7Т реализована синхронная однофазная система синхронизации с тактированием по обоим фронтам синхросигнала. Один машинный цикл равен по длительности одному периоду внешнего тактового сигнала (X2/CLKIN).

1867ВМ7Т обеспечивает цифровую обработку сигналов в реальном масштабе времени, и может применяться в прикладных системах с большим объемом вычислений в таких областях как электросвязь, модемы, обработка речевой информации, обработка графических символов и изображений, спектральный анализ, обработка звука, цифровая фильтрация, высокоскоростное управление, измерительные средства и цифровая обработка.

1 Функциональные характеристики ИС 1867ВМ7Т

- Устройство управления последовательностью выполнения программы (сиквенсор):
  - машинный цикл, равный одному периоду тактового сигнала;
  - декодирование 16-разрядных инструкций;
  - трехуровневая конвейеризация выполнения инструкций;
  - функция повтора инструкций для увеличения быстродействия 1867ВМ7Т и оптимизации использования программной памяти;
  - восьмиуровневый аппаратный стек.
- Центральное арифметико-логическое устройство (CALU):
  - 16-разрядный параллельный сдвигатель;
  - 32-разрядные арифметические и логические операции;
  - 16 × 16 разрядов аппаратный умножитель с 32-разрядным произведением;
  - одноцикловые инструкции умножения с накоплением.
- Файл вспомогательных регистров:
  - восемь 16-разрядных регистров для реализации косвенной адресации и временного хранения данных;
  - выделенное 16-разрядное арифметическое устройство (ARAU), реализующее, в том числе, операции с реверсивным распространением переноса.
- Режимы адресации памяти:
  - прямая адресация, формирующая 16-разрядный адрес памяти путем конкатенации 9-разрядного указателя страниц (DP) с семью младшими разрядами слова инструкции;
  - гибкая система косвенной адресации с использованием файла вспомогательных регистров и ARAU;
  - непосредственная адресация, при которой короткий (до 13 разрядов) операнд содержится (вместе с опкодом) в теле текущей инструкции, или длинный операнд размещается в следующем за инструкцией 16-разрядном слове;
  - блочные перемещения массивов и таблиц, как внутри памяти данных, так и для обмена между памятью данных и программ.
- Контроллер прерываний:
  - шесть (три внешних и три внутренних) источников приоритезируемых маскируемых прерываний, аппаратный сброс и одно программно инициируемое прерывание.
- Синхронный последовательный интерфейс для работы с кодеками.
- Шестнадцатиразрядный перезагружаемый таймер.
- Организация программной памяти:
  - адресация до 64K внешней программной памяти;
  - возможность конфигурирования внутрикристального ОЗУ в качестве памяти программ.

- Организация памяти данных:
  - интерфейс к внутрикристальному оперативному запоминающему устройству (ОЗУ) с блочной организацией и реализацией поблочного реконфигурирования (память данных/память программ), с возможностью асинхронного и синхронного доступа;
  - интерфейс к внутрикристальным картированным в памяти регистрам;
  - адресация до 96K внешней памяти данных.
- Шестнадцать параллельных портов ввода/вывода.
- Состояния ожидания для обмена с низкоскоростными внешними устройствами.
- Конфигурируемый (синхронный/асинхронный) доступ к внешней памяти.
- Поддержка мультипроцессорных применений.
- Блок управления режимами пониженного энергопотребления.
- Конкурентный DMA с расширенными HOLD операциями.

Требования к электрическим параметрам и режимам эксплуатации

Электрические параметры микросхемы 1867ВМ7Т при приемке и поставке приведены в таблице 1.

Значения предельно допустимых электрических режимов эксплуатации в диапазоне рабочих температур приведены в таблице 2.

Таблица 1 – Электрические параметры микросхем при приемке и поставке

Наименование параметра, единица измерения, режим измерения		Буквенное обозначение параметра	Норма параметра		Температура среды, ºС
			не менее	не более	
1		2	3	4	5
1 Выходное напряжение низкого уровня, В, U_{CC1} = 3,0 В, U_{CC2} = 3,0 В, I_{OL} = 4,0 мА		U_{OL}	–	0,45	–60 ± 3 25 ± 10 125 ± 5
2 Выходное напряжение высокого уровня, В, U_{CC1} = 3,0 В, U_{CC2} = 3,0 В, I_{OH} = –1,0 мА		U_{OH}	U_{CC1}–0,3	–	
3 Входной ток низкого уровня, мкА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, U_{IL} = 0 В	Тактовый вход	I_{IL}	–55	55	
	Все остальные входы		–15	15	
4 Входной ток высокого уровня, мкА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, U_{IH} = U_{CC1}	Тактовый вход	I_{IH}	–55	55	
	Все остальные входы		–15	15	
5 Выходной ток низкого уровня буфера с третьим состоянием в состоянии «Выключено», мкА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, U_{OZL} = 0 В		I_{OZL}	–15	–	

Окончание таблицы 1

1	2	3	4	5
6 Выходной ток высокого уровня буфера с третьим состоянием в состоянии «Выключено», мкА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, U_{OZH} = U_{CC1}	I_{OZH}	–	15	–60 ± 3 25 ± 10 125 ± 5
7 Динамический ток потребления периферии, мА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, f_{CI} = 10 МГц	I_{OCC1}	–	25	
8 Динамический ток потребления ядра микросхемы, мА, U_{CC1} = 3,6 В, U_{CC2} = 3,6 В, f_{CI} = 10 МГц	I_{OCC2}	–	210	
9 Функциональный контроль, U_{CC1} = U_{CC2} = (3,0; 3,6) В, f_{CI} = (1 – 12,5) МГц	ФК	–	–	
Примечание – Параметры I_{IL}, I_{IH}, I_{OZL}, I_{OZH} при температуре минус 60 ºС не измеряются, а гарантируются нормами при температуре (25 ± 10) ºС.				

Таблица 2 – Предельно допустимые и предельные режимы эксплуатации микросхем в диапазоне рабочих температур от минус 60 до 125 ºС

Наименование параметра режима, единица измерения		Буквенное обозначение параметра	Предельно допустимый режим		Предельный режим	
			не менее	не более	не менее	не более
1 Напряжение питания периферии, В		U_{CC1}	3,0	3,6	–0,3	5,0
2 Напряжение питания ядра ИС, В		U_{CC2}	3,0	3,6	–0,2	5,0
3 Входное напряжение низкого уровня, В		U_{IL}	0	0,6	–0,3	–
4 Входное напряжение высокого уровня, В	X2/CLKIN	U_{IH}	2,6	U_{CC1}	–	U_{CC1} + 0,3
	Остальные входы		2,0	U_{CC1}	–	U_{CC1} + 0,3
5 Напряжение на выходе с третьим состоянием в состоянии «Выключено», В		U_{OZ}	0	U_{CC1}	–0,3	U_{CC1} + 0,3
6 Выходной ток низкого уровня, мА		I_{OL}	–	4,0	–	8,0
7 Выходной ток высокого уровня, мА		I_{OH}	–1,0	–	–2,0	–
8 Частота следования импульсов тактового сигнала, МГц		f_{CI}	1,0	12,5	–	–
9 Емкость нагрузки, пФ		C_L	–	40	–	50
Примечание – Время работы в одном из предельных режимов должно быть не более 5 с.						

2 Обзор архитектуры построения ИС 1867ВМ7т
2.1 Схема электрическая структурная ИС 1867ВМ7Т

На рисунке 1 приведена схема электрическая структурная ИС 1867ВМ7Т.

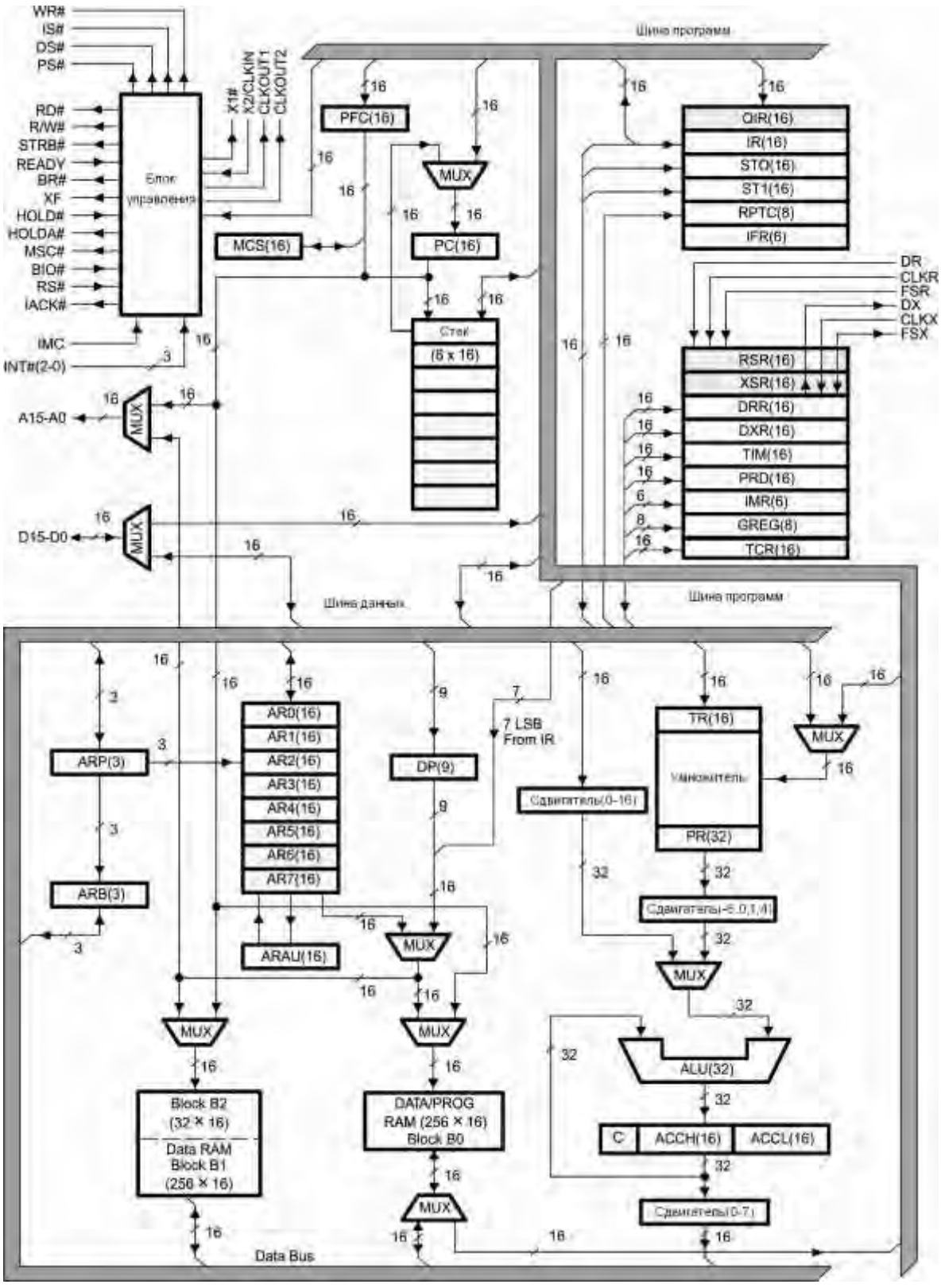


Рисунок 1, лист 1 – Схема электрическая структурная ИС 1867ВМ7Т

Обозначения на рисунке:

7 LSB From IR – семь младших разрядов регистра инструкций;
ACCH – старшая половина аккумулятора;
ACCL – младшая половина аккумулятора;
ALU – арифметико-логическое устройство;
ARx – вспомогательные регистры 0 – 7;
ARAU – арифметическое устройство вспомогательных регистров;
ARB – буфер указателя вспомогательных регистров;
ARP – указатель вспомогательных регистров;
Block Bx – блоки 0 – 2 внутрикристального ОЗУ;
C – бит переноса;
Data Bus – шина данных;
Data/Prog RAM – ОЗУ данных/программ;
Data RAM – ОЗУ данных;
DP – указатель страниц памяти данных;
DRR – регистр данных приемного последовательного порта;
DXR – регистр данных передающего последовательного порта;
GREG – регистр распределения глобальной памяти;
IFR – регистр флагов прерываний;
IMR – регистр маски прерываний;
IR – регистр инструкций;
MCS – стек микровызовов;
MUX – мультиплексор;
PC – счетчик команд;
PFC – счетчик предварительной выборки команд;
PR – регистр результата умножения;
PRD – регистр периода таймера;
QIR – регистр предварительно выбранной инструкций;
STx – регистры состояния (статусные) 0, 1;
Stack – 8-уровневый аппаратный стек;
TCR – регистр управления таймером;
TIM – таймер;
TR – регистр временного хранения множителя;
RPTC – счетчик повтора инструкции;
RSR – сдвиговый регистр приемного последовательного порта;
XSR – сдвиговый регистр передающего последовательного порта.

Рисунок 1, лист 2

2.2 Условное графическое изображение (УГО), функциональное назначение и описание выводов ИС 1867ВМ7Т

На рисунке 2 приведено условное графическое изображение ИС 1867ВМ7Т, в таблице 3 приведены функциональное назначение, описание и тип выводов микросхемы.

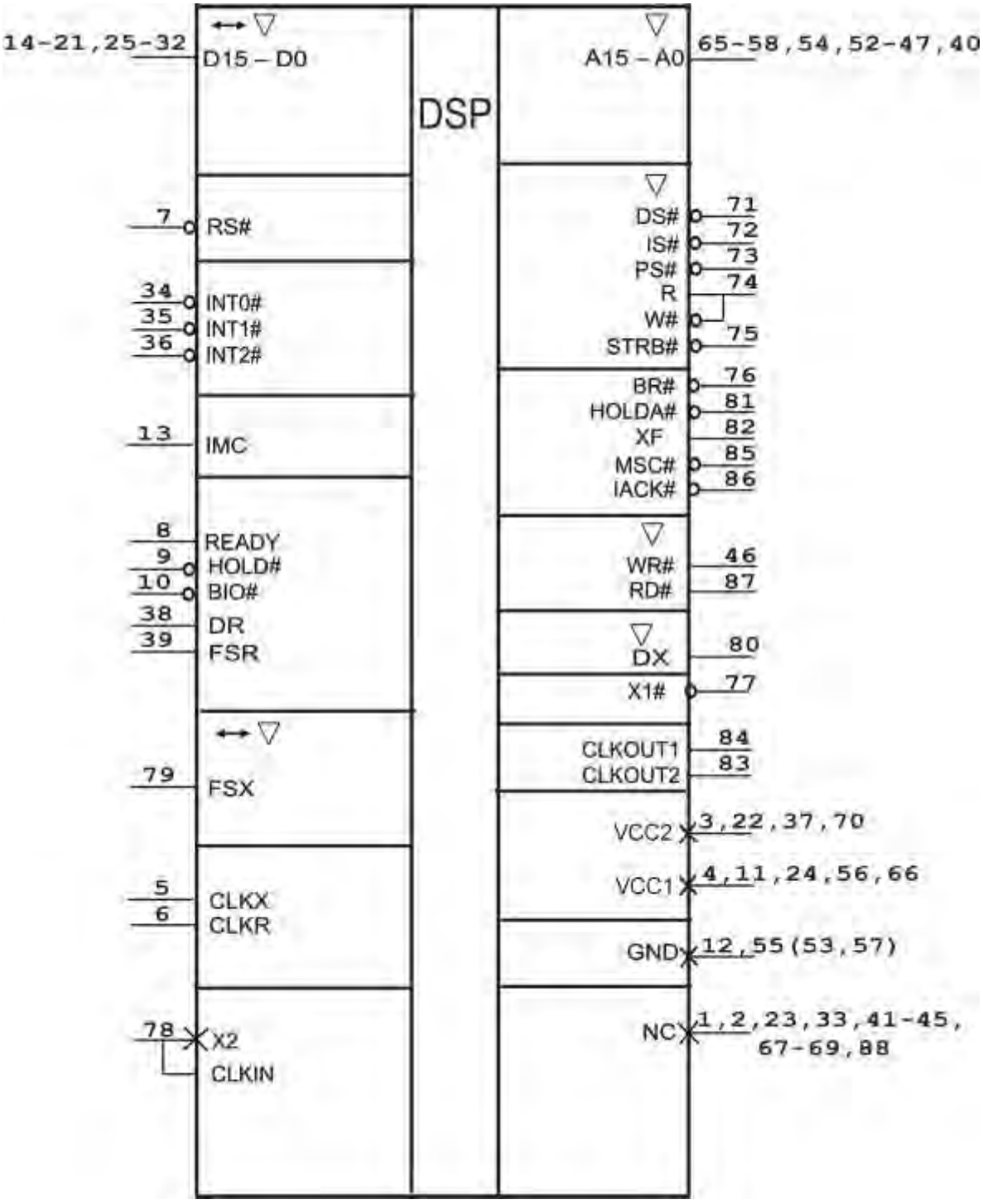


Рисунок 2 – УГО ИС 1867ВМ7Т в корпусе 4235.88-1

После разварки кристалла в корпус выводы 12 и 55 «perif_GND» электрически соединены с выводами 53, 57.

Таблица 3 – Функциональное назначение, описание и тип выводов ИС 1867ВМ7Т

Номер вывода	Тип вывода	Условное обозначение вывода	Функциональное назначение вывода
14 – 21, 25 – 32	I/O/Z	D15 –D0	Входы/выходы 16-разрядной шины данных
7	I	RS#	Вход сигнала сброса
34 35 36	I I I	INT0# INT1# INT2#	Входы сигналов внешних прерываний
13	I	IMC	Вход сигнала управления режимом тестирования памяти
9	I	HOLD#	Вход сигнала установки режима HOLD (захват)
8	I	READY	Вход сигнала готовности данных
10	I	BIO#	Вход сигнала управления ветвлением
38	I	DR	Вход приема последовательного порта
39	I	FSR	Вход сигнала кадровой синхронизации для приема данных
79	I/O/Z	FSX	Вход/выход сигнала кадровой синхронизации для передачи данных
6	I	CLKR	Тактовый вход приема данных
5	I	CLKX	Тактовый вход передачи данных
80	O/Z	DX	Выход для передачи последовательных данных
73	O/Z	PS#	Выход сигнала выбора программы
72	O/Z	IS#	Выход сигнала ввода/вывода
71	O/Z	DS#	Выход сигнала выбора данных
75	O/Z	STRB#	Выход сигнала стробирования
74	O/Z	R/W#	Выход сигнала считывания/записи
46	O/Z	WR#	Запись данных на внешнюю шину
87	O/Z	RD#	Чтение данных с внешней шины.

Окончание таблицы 3

Номер вывода	Тип вывода	Условное обозначение вывода	Функциональное назначение вывода
65 - 58 54 52 51 50 49 48 47 40	O/Z	A15 - A0	Выходы 16-разрядной шины адреса
86	O	IACK#	Выход сигнала подтверждения приема прерывания
85	O	MSC#	Выход сигнала окончания микросостояния
81	O	HOLDA#	Выход сигнала подтверждения режима HOLD (захват)
76	O	BR#	Выход сигнала запроса шины
82	O	XF	Выход программно устанавливаемого внешнего флага
78	I	X2/CLKIN	Вход внутреннего генератора для подключения кварцевого резонатора или внешнего тактового сигнала
77	O	X1#	Выход внутреннего генератора при подключении кварцевого резонатора
84	O	CLKOUT1	Выход первого тактового сигнала
83	O	CLKOUT2	Выход второго тактового сигнала
3, 22, 37, 70	-	VCC2	Выводы питания ядра ИС
4, 11, 24, 56, 66	-	VCC1	Выводы питания периферийных буферов
12, 55	-	GND	Выводы “земли” периферийных буферов
1, 2, 23, 33, 41-45, 67-69, 88	-	NC	Не используются

2.3 Система команд ИС 1867ВМ7Т

Таблица 4 – Набор инструкций ИС 1867ВМ7Т в алфавитном порядке

Мнемо-ника	Синтаксис	Выполнение
ABS	ABS	ACC ← (|ACC|), C ← 0
ADD	ADD dma [, shift] ADD * [, shift [, next ARP]]	C, ACC ← (ACC)+(dma) × 2^{shift}
ADDC	ADDC dma ADDC * [, next ARP]	C, ACC ← (ACC)+(dma)+(C)
ADDH	ADDH dma ADDH * [, next ARP]	C, ACC ← (ACC)+(dma) × 2^{16}
ADDK	ADDK constant	C, ACC ← (ACC)+8-бит положи-тельная константа
ADDS	ADDS dma ADDS * [, next ARP]	C, ACC ← (ACC)+(dma), где (dma) - 16-бит беззнаковое число
ADDT	ADDT dma ADDT * [, next ARP]	C, ACC ← (ACC)+(dma) × $2^{TR(3:0)}$
ADLK	ADLK constant [, shift]	C, ACC ← (ACC)+константа × 2^{shift}
ADRK	ADRK constant	AR(ARP) ← (AR(ARP))+8-бит по-ложительная константа
AND	AND dma AND * [, next ARP]	ACC(31:16) ← 0; ACC(15:0) ← (ACC(15:0)) Лог. «И» (dma)
ANDK	ANDK constant [, shift]	ACC(31) ← 0; ACC(30:0) ← (ACC(30:0)) Лог. «И» [константа × 2^{shift}]
APAC	APAC	C, ACC ← (ACC)+(сдвинутый PR)
B	B pma, [* [, next ARP]]	PC ← pma
BACC	BACC	PC ← (ACC(15:0))
BANZ	BANZ pma, [* [, next ARP]]	если AR(ARP)≠0, тогда PC ← pma
BBNZ	BBNZ pma, [* [, next ARP]]	если TC=1, тогда PC ← pma
BBZ	BBZ pma, [* [, next ARP]]	если TC=0, тогда PC ← pma
BC	BC pma, [* [, next ARP]]	если C=1, тогда PC ← pma
BGEZ	BGEZ pma, [* [, next ARP]]	если (ACC)≥0, тогда PC ← pma
BGZ	BGZ pma, [* [, next ARP]]	если (ACC)>0, тогда PC ← pma
BIOZ	BIOZ pma, [* [, next ARP]]	если BIO=0, тогда PC ← pma
BIT	BIT dma, bit code BIT *, bit code [, next ARP]	TC ← (dma бит(код бита))
BITT	BITT dma BITT * [, next ARP]	TC ← (dma бит(15 – TR(3:0)))
BLEZ	BLEZ pma, [* [, next ARP]]	если (ACC)≤0, тогда PC ← pma
BLKD	BLKD dma1, dma2 BLKD dma1, * [, next ARP]	dma2 ← (dma1, адресуемый PFC)

Мнемо-ника	Синтаксис	Выполнение
BLKP	BLKP pma, dma BLKP pma, * [, next ARP]	dma ← (pma, адресуемый PFC)
BLZ	BLZ pma, [* [, next ARP]]	если (ACC)<0, тогда PC ← pma
BNC	BNC pma, [* [, next ARP]]	если C=0, тогда PC ← pma
BNV	BNV pma, [* [, next ARP]]	если OV=0, тогда PC ← pma
BNZ	BNZ pma, [* [, next ARP]]	если (ACC)≠0, тогда PC ← pma
BV	BV pma, [* [, next ARP]]	если OV=1, тогда PC ← pma
BZ	BZ pma, [* [, next ARP]]	если (ACC)=0, тогда PC ← pma
CALA	CALA	TOS ← (PC); PC ← (ACC(15:0))
CALL	CALL pma, [* [, next ARP]]	TOS ← (PC); PC ← pma
CMPL	CMPL	ACC ← (инверсия ACC)
CMPR	CMPR constant	Сравнить AR(ARP) с AR0; TC ← результат
CNFD	CNFD	Бит CNF ← 0
CNFP	CNFP	Бит CNF ← 1
DINT	DINT	Бит INTM ← 1
DMOV	DMOV dma DMOV * [, next ARP]	dma+1 ← (dma)
EINT	EINT	Бит INTM ← 0
FORT	FORT constant	Бит FO ← 1-бит константа
IDLE	IDLE	Бит INTM ← 0; Останов исполне-ния программы до прерывания
IN	IN dma, PA IN * , PA, [, next ARP]	Шина адреса(3:0) ← Адрес порта Шина адреса (15:4) ← 0 dma ← Шина данных
LAC	LAC dma , shift LAC * , shift [, next ARP]	ACC ← (dma) × 2^{shift}
LACK	LACK constant	ACC ← 8-бит положит. константа
LACT	LACT dma LACT * [, next ARP]	ACC ← (dma) × $2^{TR(3:0)}$
LALK	LALK constant [, shift]	ACC ← 16-бит константа × 2^{shift}
LAR	LAR AR, dma LAR AR, * [, next ARP]	AR ← (dma)
LARK	LARK AR, constant	AR ← 8-бит положит. константа
LARP	LARP constant	ARB ← (ARP) ARP ← 3-бит константа
LDP	LDP dma LDP * [, next ARP]	DP ← (dma(8:0))
LDPK	LDPK constant	DP ← 9-бит положит. константа
LPH	LPH dma LPH * [, next ARP]	PR(31:16) ← (dma)
LRLK	LRLK AR, constant	AR ← 16-бит константа

Мнемо-ника	Синтаксис	Выполнение
LST	LST dma LST * [, next ARP]	ST0 ← (dma)
LST1	LST1 dma LST1 * [, next ARP]	ST1 ← (dma) ARP ← (dma(15:13))
LT	LT dma LT * [, next ARP]	TR ← (dma)
LTA	LTA dma LTA * [, next ARP]	TR ← (dma) C, ACC ← (ACC)+(сдвинутый PR)
LTD	LTD dma LTD * [, next ARP]	TR ← (dma) C, ACC ← (ACC)+(сдвинутый PR) dma+1 ← (dma)
LTP	LTP dma LTP * [, next ARP]	TR ← (dma) ACC ← (сдвинутый PR)
LTS	LTS dma LTS * [, next ARP]	TR ← (dma) C, ACC ← (ACC)–(сдвинутый PR)
MAC	MAC pma, dma MAC pma, * [, next ARP]	C, ACC ← (ACC)+(сдвинутый PR) TR ← (dma); PR ← (dma) × (pma, адресуемый PFC)
MACD	MACD pma, dma MACD pma, * [, next ARP]	C, ACC ← (ACC)+(сдвинутый PR); TR ← (dma); dma+1 ← (dma); PR ← (dma) × (pma, адресуемый PFC)
MAR	MAR dma MAR * [, next ARP]	Модификация AR(ARP) и ARP
MPY	MPY dma MPY * [, next ARP]	PR ← (dma) × (TR)
MPYA	MPYA dma MPYA * [, next ARP]	C, ACC ← (ACC(сдвинутый PR) PR ← (dma) × (TR)
MPYK	MPYK constant	PR register ← (TR) × 13бит кон-станта
MPYS	MPYS dma MPYS * [, next ARP]	C, ACC ← (ACC)– (сдвинутый PR) PR ← (dma) × (TR)
MPYU	MPYU dma MPYU * [, next ARP]	PR ← беззнаковый (dma) × без-знаковый (TR)
NEG	NEG	ACC ← (ACC) × –1; если (ACC)=0, тогда C ← 1, иначе C ← 0
NOP	NOP	Нет операции
NORM	NORM *	Если (ACC)=0, тогдаTC ← 1 иначе, если (ACC(31) «Исключ. ИЛИ» ACC(30))=1, тогда TC ← 0; ACC ← (ACC) × 2 иначе TC ← 1

Мнемо-ника	Синтаксис	Выполнение
OR	OR dma OR * [, next ARP]	ACC(15:0) ← (ACC(15:0)) Лог «ИЛИ» (dma)
ORK	ORK constant [, shift]	ACC(30:0) ← (ACC(30:0)) Лог «ИЛИ» [константа × 2^{shift}]
OUT	OUT dma, PA OUT * , PA, [, next ARP]	Шина адреса(3:0) ← Адрес порта Шина адреса (15:4) ← 0 Шина данных ← (dma)
PAC	PAC	ACC ← (сдвинутый PR)
POP	POP	ACC(15:0) ← (TOS) ACC(15:0) ← 0
POPD	POPD dma POPD * [, next ARP]	dma ← (TOS)
PSHD	PSHD dma PSHD * [, next ARP]	TOS ← (dma)
PUSH	PUSH	TOS ← (ACC(15:0))
RC	RC	Бит C ← 0
REMC	REMC	Бит EMC ← 0
RET	RET	PC ← (TOS)
RFSM	RFSM	Бит FSM ← 0
RHM	RHM	Бит HM ← 0
RIMC	RIMC	Бит IMC ← 0
ROL	ROL	C ← ACC(31) ACC(31:1) ← (ACC(30:0)) ACC(0) ← предыдущий C
ROR	ROR	C ← ACC(0) ACC(30:0) ← (ACC(31:1)) ACC(31) ← предыдущий C
ROVM	ROVM	Бит OVM ← 0
RPDM	RPDM	Бит PDM ← 0
RPT	RPT dma RPT * [, next ARP]	RPTC ← (dma(7:0)) Повтор следующей инструкции (dma(7:0))+1 раз
RPTK	RPTK constant	RPTC ← 8-бит константа; Повтор следующей инструкции константа+1 раз
RSXM	RSXM	Бит SXM ← 0
RTC	RTC	Бит TC ← 0
RTXM	RTXM	Бит TXM ← 0
RXF	RXF	Бит XF ← 0
SACH	SACH dma [, shift] SACH * [, shift [, next ARP]]	dma ← (ACC(31:16) × 2^{shift})

Мнемо-ника	Синтаксис	Выполнение
SACL	SACL dma [, shift] SACL * [, shift [, next ARP]]	dma ← (ACC(15:0) × 2^{shift})
SAR	SAR AR, dma SAR AR, * [, next ARP]	dma ← (AR)
SBLK	SBLK constant [, shift]	C, ACC ← (ACC)–константа × 2^{shift}
SBRK	SBRK constant	AR(ARP) ← (AR(ARP))–8-бит положительная константа
SC	SC	Бит C ← 1
SEMC	SEMC	Бит EMC ← 1
SFL	SFL	C ← ACC(31) ACC(31:1) ← (ACC(30:0)) ACC(0) ← 0
SFR	SFR	C ← ACC(0) ACC(30:0) ← (ACC(31:1)) если SXM=0, тогда ACC(31) ← 0, иначе ACC(31) ← (ACC(31))
SFSM	SFSM	Бит FSM ← 1
SHM	SHM	Бит HM ← 1
SIMC	SIMC	Бит IMC ← 1
SOVM	SOVM	Бит OVM ← 1
SPAC	SPAC	C, ACC ← (ACC)–(сдвинутый PR)
SPDM	SPDM	Бит PDM ← 1
SPH	SPH dma SPH * [, next ARP]	dma ← (сдвинутый PR(31:16))
SPL	SPL dma SPL * [, next ARP]	dma ← (сдвинутый PR(15:0))
SPM	SPM constant	Биты режима сдвига произведения PM ← 2-бит константа
SQRA	SQRA dma SQRA * [, next ARP]	C, ACC ← (ACC)+(сдвинутый PR) TR ← (dma) PR ← (dma) × (dma)
SQRS	SQRS dma SQRS * [, next ARP]	C, ACC ← (ACC)– (сдвинутый PR) TR ← (dma) PR ← (dma) × (dma)
SST	SST dma SST * [, next ARP]	dma ← (ST0)
SST1	SST1 dma SST1 * [, next ARP]	dma ← (ST1)
SSXM	SSXM	Бит SXM ← 1
STC	STC	Бит TC ← 1
STXM	STXM	Бит TXM ← 1

Мнемо-ника	Синтаксис	Выполнение
SUB	SUB dma [, shift] SUB * [, shift [, next ARP]]	C, ACC ← (ACC)–(dma) × 2^{shift}
SUBB	SUBB dma SUBB * [, next ARP]	C, ACC ← (ACC)–(dma)–(инверсия C)
SUBC	SUBC dma SUBC * [, next ARP]	Выход ALU ← (ACC)–(dma) × 2^{15} если выход ALU≥0, тогда ACC ← (выход ALU) × 2+1, иначе ACC ← (ACC) × 2
SUBH	SUBH dma SUBH * [, next ARP]	C, ACC ← (ACC)–(dma) × 2^{16}
SUBK	SUBK constant	C, ACC ← (ACC)–8-бит положительная константа
SUBS	SUBS dma SUBS * [, next ARP]	C, ACC ← (ACC)–(dma), где (dma) - 16-бит беззнаковое число
SUBT	SUBT dma SUBT * [, next ARP]	C, ACC ← (ACC)–(dma) × $2^{TR(3:0)}$
SXF	SXF	XF ← 1
TBLR	TBLR dma TBLR * [, next ARP]	dma ← (pma, адресуемый PFC)
TBLW	TBLW dma TBLW * [, next ARP]	pma, адресуемый PFC ← (dma)
TRAP	TRAP	TOS ← (PC) PC ← 001Eh
XOR	XOR dma XOR * [, next ARP]	ACC(15:0) ← (ACC(15:0)) «Исключ. ИЛИ» (dma)
XORK	XORK constant [, shift]	ACC(30:0) ← (ACC(30:0)) «Исключ. ИЛИ» [constant × 2^{shift}]
ZAC	ZAC	ACC ← 0
ZALH	ZALH dma ZALH * [, next ARP]	ACC(31:16) ← (dma) ACC(15:0) ← 0
ZALR	ZALR dma ZALR * [, next ARP]	ACC(31:16) ← (dma) ACC(15:0) ← 8000h
ZALS	ZALS dma ZALS * [, next ARP]	ACC(31:16) ← 0 ACC(15:0) ← (dma)

2.4 Организация памяти ИС 1867ВМ7Т

В процессорном ядре ИС 1867ВМ7Т реализован интерфейс к трем раздельным пространствам памяти:

- Программная память общим объемом 64K 16-разрядных слов:
  - один или несколько блоков конфигурируемого внутреннего ОЗУ;
  - внешняя программная память.
- Память данных общим объемом 64K 16-разрядных слов (с возможностью расширения до 96K слов при использовании механизма обращения к глобальной внешней памяти):
  - блоки внутреннего ОЗУ;
  - картированные в памяти внутренние регистры;
  - внешнее ОЗУ данных.
- Внешнее пространство ввода-вывода через 16-разрядные параллельные порты.

Доступ к внешним областям этих пространств осуществляется под управлением трех выходных сигналов 1867ВМ7Т: PS#, DS#, и IS#, отвечающих за выбор внешней программной памяти, внешней памяти данных и внешнего пространства ввода-вывода соответственно.

Внутренняя оперативная память ИС 1867ВМ7Т суммарным объемом 544 16-разрядных слова, разделена на три блока (два из которых имеют объем 256 слов, а третий – 32). При этом один из трех блоков (блок B0), объемом 256 слов, может быть программно сконфигурирован в пространство памяти программ, а оставшиеся два (блоки B1 и B2 – 288 слов) всегда являются памятью данных. Для конфигурирования блока B0 предусмотрены две инструкции: CNFD - картирует B0 во внутреннюю память данных, и CNFP - картирует B0 в старшие адреса памяти программ.

Вся область памяти данных 1867ВМ7Т разделена на 512 страниц объемом 128 шестнадцати-разрядных слов каждая. Страницы 7 - 0 относятся к пространству внутренней памяти данных, страницы 511 - 8 картированы в пространство внешней памяти. Карты памяти данных 1867ВМ7Т (в зависимости от варианта конфигурирования блока B0) приведены на рисунке 3.

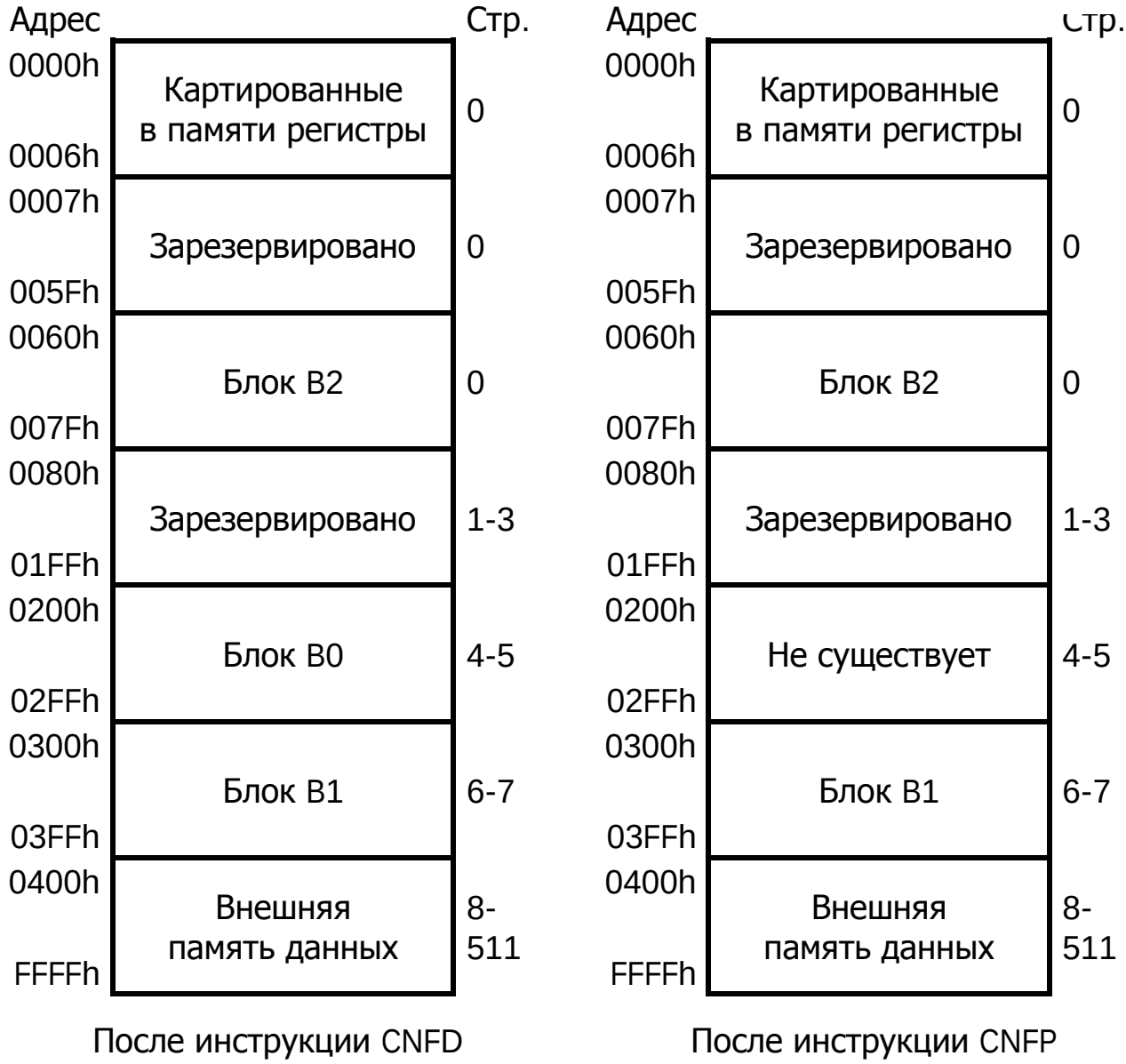


Рисунок 3 - Типовые карты памяти данных ИС 1867ВМ7Т

В нулевой странице памяти данных 1867ВМ7Т размещены 32-словный блок B2 внутреннего ОЗУ и картированные в памяти регистры.

Размер и расположение блока B2 определяются его назначением. Как правило, он используется для сохранения контекста (содержимого основных регистров ядра 1867ВМ7Т) при прерываниях и вызовах подпрограмм. Выбор нулевой страницы обусловлен тем, что при аппаратном сбросе указатель страниц памяти данных (DP) обнуляется, выбирая, таким образом, именно эту страницу, а 32 слов достаточно для сохранения основных регистров ядра 1867ВМ7Т. В остальном блок B2 ничем не отличается от других блоков внутреннего ОЗУ и, наряду с контекстом, также может хранить исходные операнды и результаты вычислений. Процессор поддерживает абсолютно все инструкции работы с памятью в отношении этого блока, включая табличный (TBLR, TBLW) и блочный (BLKD, BLKP) обмен данными, а также функцию сдвига данных во внутренней памяти (DMOV, LTD, MACD).

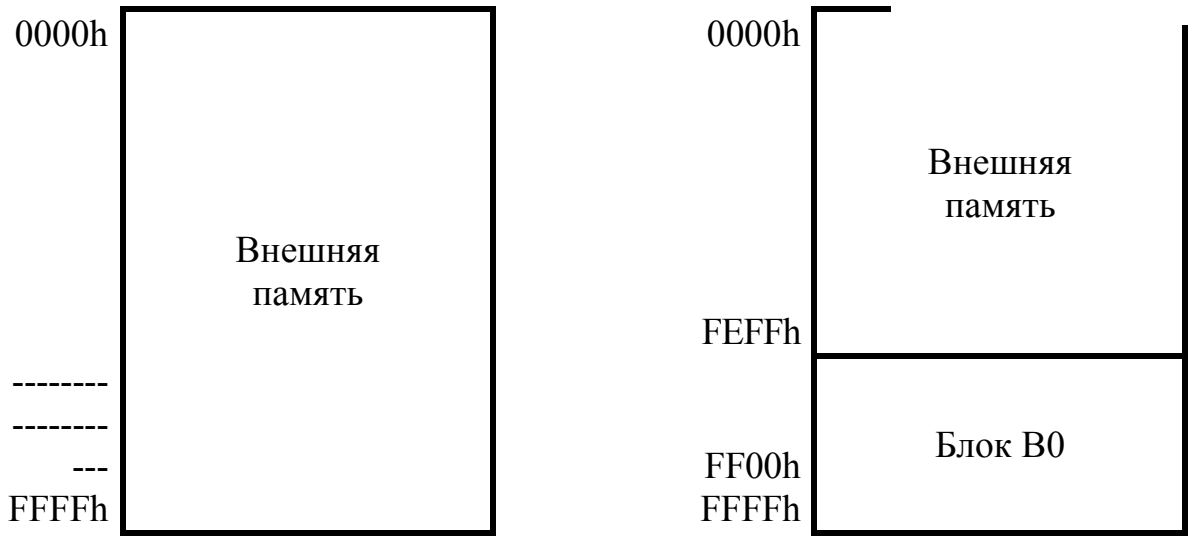
Ядро ИС 1867ВМ7Т имеет семь регистров, картированных во внутреннюю область памяти данных. Название, адреса и назначение этих регистров приведены в таблице 5.

Таблица 5 - Картированные в памяти регистры

Название	Адрес	Назначение
DRR	0000h	Регистр принимаемых данных последовательного порта
DXR	0001h	Регистр передаваемых данных последовательного порта
TIM	0002h	Регистр таймера
PRD	0003h	Регистр периода таймера
IMR	0004h	Регистр маски прерывания
GREG	0005h	Регистр распределения глобальной памяти
TCR	0006h	Регистр управления таймером

Картированные в памяти регистры позволяют осуществлять управление периферийными устройствами ИС 1867ВМ7Т без введения специализированных инструкций. Также как и обычные ячейки памяти, картированные в памяти регистры доступны для чтения и записи стандартными командами работы с памятью, за небольшим исключением: эти регистры не могут быть использованы в инструкциях блочного обмена данными (BLKD, BLKP) и сдвига данных во внутренней памяти (DMOV, LTD, MACD).

Конфигурация программной памяти зависит от состояния бита CNF статусного регистра ST1, который определяет картирование блока B0 внутреннего ОЗУ либо в область памяти данных, либо в старшие адреса программной памяти. По умолчанию (после сброса) CNF=0, а блок B0 доступен как страницы четвертая и пятая памяти данных. Если алгоритм не требует большого объема памяти, но критичен к скорости обработки, можно стандартными инструкциями перемещения данных записать программный код в блок B0, затем инструкцией CNFP установить бит CNF в 1, и передать управление программой на адрес FF00h. Именно с этого адреса блок B0 располагается в программной памяти, если CNF=1 (см. рисунок 4).



После инструкции CNFD

После инструкции CNFP

Рисунок 4 - Карты программной памяти ИС 1867ВМ7Т

Внешнее пространство ввода/вывода ИС 1867ВМ7Т содержит 16 входных и 16 выходных параллельных портов. При однократном выполнении операции ввода или вывода (инструкции IN и OUT, соответственно) обычно требуют двух машинных циклов. Если использовать эти операции в режиме повтора (с инструкциями RPT или RPTK), они становятся эффективно одноцикловыми. При обмене данными в пространстве ввода/вывода процессор использует внешние шины адреса и данных, при этом временные диаграммы работы этих шин ничем не отличаются от других операций с внешней памятью.

3 Описание аппаратуры ИС 1867ВМ7Т

ИС 1867ВМ7Т включает в себя следующие основные блоки:

- Устройство управления последовательностью выполнения программы.
- Центральное арифметико-логическое устройство (CALU).
- Файл вспомогательных регистров.
- Статусные регистры.
- Интерфейс памяти.
- Контроллер прерываний и периферийные устройства:
  - последовательный интерфейс;
  - таймер.
- Блок управления режимами пониженного энергопотребления.

3.1 Устройство управления последовательностью выполнения программы

Устройство управления последовательностью выполнения программы (сиквенсор) обеспечивает декодирование инструкций, корректную циклограмму их выполнения в трехуровневом конвейере (в том числе – в режиме повтора), работу механизмов ветвления программного кода и часть функций по конфигурированию памяти.

Сиквенсор состоит из:

- 16-разрядного программного счетчика (PC);
- счетчика предвыборки (PFC);
- 16-разрядного стека микровызовов (MCS) для временного сохранения содержимого PFC;
- 16-разрядного регистра инструкций (IR);
- 16-разрядного регистра предвыбранных инструкций (QIR);
- 8-разрядного счетчика повтора инструкций;
- 8-уровневого аппаратного (LIFO) стека.

3.2 Программный счетчик

16-разрядный программный счетчик (PC) содержит адрес инструкции, которая будет выполняться следующей. При сбросе в него загружается значение 0000h, таким образом, первая инструкция выбирается из ячейки с нулевым адресом внешней программной памяти. При последовательном выполнении программы, после декодирования инструкции, PC инкрементируется (на 1 или на 2, в зависимости от того была ли эта инструкция однословной или двухсловной). При прерываниях или вызовах подпрограмм содержимое PC помещается в верхний уровень (вершину) стека TOS. Инструкция возврата RET восстанавливает значение PC. При организации вложенных вызовов необходимо помнить, что при переполнении стека содержимое его низшего уровня теряется.

Счетчик предвыборки (PFC) строго следует за PC и фактически именно его содержимое (а не PC) выдается на шину программного адреса. При сбросе в PFC

также появляется значение 0000h, и затем его содержимое инкрементируется всякий раз, когда инициируется чтение или запись программной памяти. Некоторые инструкции (BLKP, BLKD, MAC, MACD, TBLR, TBLW) используют PFC для генерации адреса второго операнда, который может содержаться как в памяти данных, так и в программной памяти. Эти инструкции модифицируют только содержимое PFC и не оказывают влияния на PC. На время выполнения таких инструкций содержимое PFC сохраняется в одноуровневом стеке микровызовов, и затем, по завершении выполнения, сохраненное значение загружается обратно в PFC.

В случае ветвления программного кода (при переходах, вызовах и прерывания), в PC и в PFC загружается один и тот же программный адрес, по которому и осуществляется переход, вызов или обработка прерывания.

3.3 Счетчик повторений

Восьмиразрядный обратный (работающий на уменьшение) счетчик повторений (RPTC) предназначен для аппаратной организации многократного (до 256 раз) выполнения инструкций. Его использование позволяет уменьшить объем программного кода и увеличить быстродействие 1867ВМ7Т, поскольку большинство повторяемых инструкций в режиме повтора (если отсутствуют состояния ожидания, обусловленные применением медленной внешней памяти) становятся эффективно одноцикловыми. Циклограммы выполнения инструкций в однократном режиме и в режиме повтора приведены в приложении А. Однако не все инструкции являются повторяемыми. Поэтому необходимо рассмотреть механизм организации повторений подробнее.

Счетчик повторений может быть загружен инструкциями RPT или RPTK. После загрузки в RPTC ненулевого значения N, следующая инструкция будет повторена N+1 раз (при условии, что эта инструкция – повторяемая). При каждом повторе RPTC будет декрементироваться на 1, пока не достигнет нуля. Поскольку не предусмотрено механизма сохранения содержимого RPTC, операция повторения является непрерываемой, т.е. возможные запросы на прерывания будут отложены до ее завершения. Если после RPT или RPTK окажется неповторяемая инструкция, произойдет ее однократное выполнение, при этом RPTC не будет активирован, и его содержимое не изменится. Соответственно первая встреченная в программном коде повторяемая инструкция и будет выполнена N+1 раз. Организация «отложенных» повторений (когда инструкция, предназначенная для выполнения в режиме повтора, не следует непосредственно за RPT или RPTK, а отделена от них неповторяемыми инструкциями) может привести к некорректным результатам. Будучи однажды загруженным ненулевым значением RPTC активируется только, когда в программном коде встретится повторяемая инструкция (обнулить счетчик повторений без такой активации можно только с помощью загрузки в него инструкциями RPT или PRTK нулевого значения). Если в промежутке между загрузкой RPTC и его активацией произойдет прерывание, то, вместо повтора предназначенной для этого инструкции, произойдет повтор первой повторяемой инструкции в подпрограмме обработки прерывания.

3.4 Операции конвейера

Процессор 1867ВМ7Т осуществляет трехуровневую конвейерную обработку с раздельными уровнями для декодирования инструкций. Первый уровень конвейера – предварительная выборка инструкции; второй уровень – декодирование инструкции; третий уровень – выполнение инструкции. Стадия предварительной выборки связана с PFC и регистром предвыбранной инструкции (QIR). Выход PFC соединен с шиной программного адреса, содержимое ячейки программной памяти (инструкция), на которую указывает этот адрес, записывается в QIR, если предыдущая инструкция (содержавшаяся до этого в QIR) «ушла» на стадию декодирования. В свою очередь, на стадии декодирования можно непосредственно выбрать инструкцию из памяти, если декодер свободен или; в противном случае инструкция записывается в QIR. В последнем случае, когда декодер освобождается, следующая инструкция для декодирования берется не с программной шины, а из регистра предвыбранной инструкции (QIR). В зависимости от вариантов декодирования в конвейере все инструкции делятся на несколько классов (см. таблицу 5). Причем, для корректной организации возможных операций с памятью, распознавание класса инструкции выполняется до ее попадания в QIR или на стадию декодирования. По существу конвейер прозрачен для программиста, так как результат любой предыдущей инструкции всегда доступен для следующей. Единственным исключением является конфигурирование блока B0 внутреннего ОЗУ. Новая конфигурация памяти будет доступна после изменения значения бита CNF для первой же инструкции доступа к данным, но только для второй инструкции выборки команд.

1867ВМ7Т выполняет большинство инструкций в течение одного машинного цикла, если используется достаточно быстрая внешняя память. ИС также может работать с более медленными ЗУ вне кристалла или с периферийными устройствами, используя вход READY (готовность). В этих случаях, соответствующие инструкции становятся многоцикловыми.

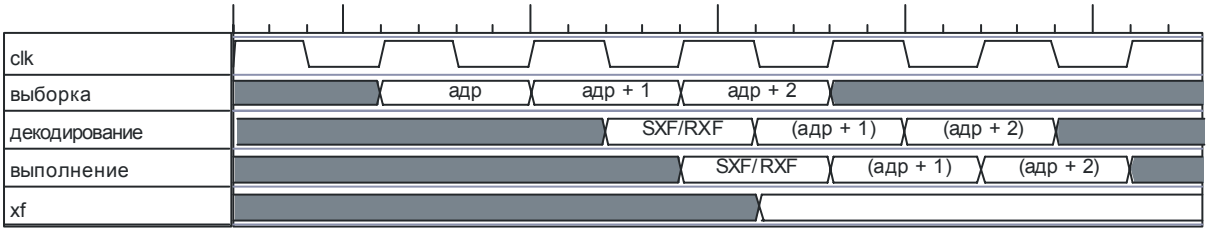
Таблица 6 - Классы инструкций 1867ВМ7Т

Класс	Инструкции	Описание
1	ADD ADDC ADDH ADDS ADDT AND BIT BITT LAC LACT LPH LT LTA LTP LTS MPY MPYA MPYS MPYU PSHD OR RPT SQRA SQRS SUB SUBB SUBC SUBH SUBS SUBT XOR ZALH ZALR ZALS	Чтение данных из памяти и выполнение заданных операций над ними
1M	DMOV LTD	Чтение данных из памяти, сдвиг данных и выполнение заданных операций над ними
2	LAR LDP LST LST1	Чтение данных из памяти и загрузка статусных регистров
3	POPD SACH SACL SAR SPH SPL SST SST1	Запись данных в память
4	ABS ADDK ADRK APAC CMPL CMPR CNFD CNFP DINT EINT FORT LACK LARK LARP LDPK MAR MPYK NEG NOP NORM PAC POP PUSH RC REMC RFSM RHM RIMC ROL ROR ROVM RPDM RPTK RSXM RTC RTXM RXF SBRK SC SEMC SFL SFR SFSM SHM SIMC SOVM SPAC SPDM SPM SSXM STC STXM SUBK SXF ZAC	Инструкции без операндов или с коротким непосредственным операндом
5	ADLK ANDK LALK LRLK ORK SBLK XORK	Инструкции с длинным непосредственным операндом
6	MAC	Инструкция с длинным непосредственным операндом и чтением из памяти
6M	MACD	Инструкция со сдвигом данных и с длинным непосредственным операндом и чтением из памяти
7	B BANZ BBNZ BBZ BC BGEZ BGZ BIOZ BLEZ BLZ BNC BNV BNZ BV BZ CALL	Двухсловные инструкции ветвления
8	BACC CALA RET TRAP	Однословные инструкции ветвления
9	IN	Чтение данных из порта
10	OUT	Запись данных в порт
11	TBLR	Копирование программной памяти в память данных (адрес из АСС)
12	TBLW	Копирование памяти данных в программную память (адрес из АСС)
13	BLKD	Копирование внутри памяти данных (двухсловная)
14	BLKP	Копирование программной памяти в память данных (двухсловная)
15	IDLE	Останов до прерывания

Циклограммы выполнения инструкций различных классов в зависимости от доступа в различные области памяти приведены в приложении А.

3.5 Выводы общего назначения ИС 1867ВМ7Т

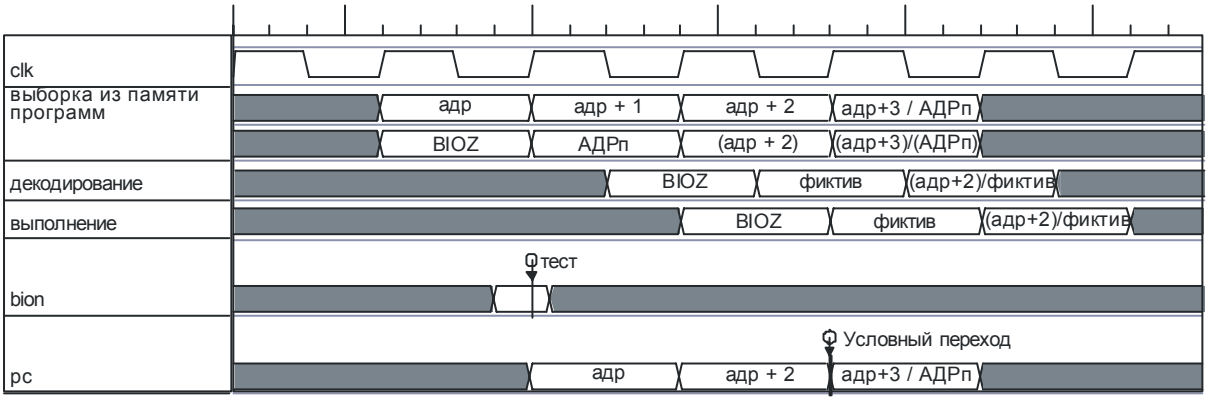
ИС 1867ВМ7Т имеет два вывода общего назначения. Вывод внешнего флага XF является программно управляемым выходом и отражает состояние одноимённого бита в статусном регистре ST1 (ST1[4]). Соответственно инструкции SXF, RXF и LST1, модифицируя бит статусного регистра, одновременно изменяют состояние выхода XF.



адр - адрес памяти программ
(адр) - данные по указанному адресу памяти программ

Рисунок 5 - Временная диаграмма выхода внешнего флага XF

Вход BIO# используется для управления ветвлением программного кода. Инструкция условного перехода BIOZ тестирует состояние этого входа и, если на BIO# низкий уровень, выполняет переход на указанный адрес. При высоком уровне BIO#, программа продолжает выполнение со следующей за BIOZ инструкции. Опрос входа BIO# осуществляется по переднему фронту clk, при этом для активизации ветвления по BIOZ, BIO# должен быть низким как минимум в течение одного машинного цикла. Таким образом, состояние BIO# актуально на стадии выборки инструкции BIOZ.



адр - адрес памяти программ;
(адр) - данные по указанному адресу памяти программ;
АДРп - адрес перехода;
тест - тестирование состояния сигнала BIO#

Рисунок 6 - Временная диаграмма входа BIO#

3.6 Вспомогательные регистры ИС 1867ВМ7Т

ИС 1867ВМ7Т имеет регистровый файл, содержащий восемь вспомогательных регистров (AR0 – AR7). Вспомогательные регистры (все или некоторые) могут быть использованы как для косвенной адресации операндов в памяти данных, так и для временного хранения собственно операндов или информации.

Косвенная адресация в процессоре 1867ВМ7Т одноопернадная, поэтому для текущей операции выбирается один из восьми регистров. Выбор регистра, содержащего актуальный адрес операнда, осуществляется загрузкой необходимого значения (от 0 до 7) в указатель вспомогательного регистра (ARP); при этом будет выбран регистр AR0 – AR7, соответственно (см. рисунок 7).

Вспомогательные регистры и ARP могут быть загружены как из памяти данных, так и с помощью непосредственного операнда, задаваемого в коде команды. Содержимое этих регистров может быть сохранено в памяти данных.

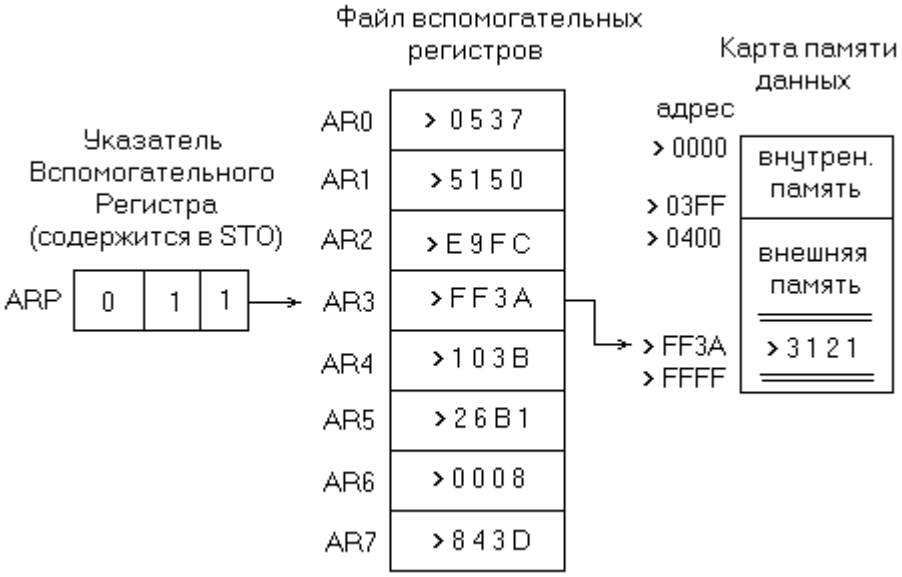


Рисунок 7 - Пример косвенной адресации с помощью вспомогательных регистров

С файлом вспомогательных регистров связано выделенное арифметическое устройство ARAU (см. рисунок 8). При косвенной адресации ARAU позволяет модифицировать содержимое текущего вспомогательного регистра для реализации необходимой последовательности выборки операндов из памяти данных.

Модификация происходит одновременно с адресацией ячейки памяти данных при помощи этого же регистра. Модификация может заключаться в приращении или уменьшении текущего регистра на 1 (инкремент/декремент) либо на значение, содержащееся в AR0 (индексная адресация). ARAU также может выполнять сравнение содержимого текущего регистра AR(ARP) с содержимым AR0, помещая результат в TC-бит. Выделенное арифметическое устройство позволяет освободить основной блок обработки данных (CALU), для выполнения собственно операций по цифровой обработке сигналов. Тем не менее, для сложных вариантов адресации к массивам данных, имеется возможность использования CALU.

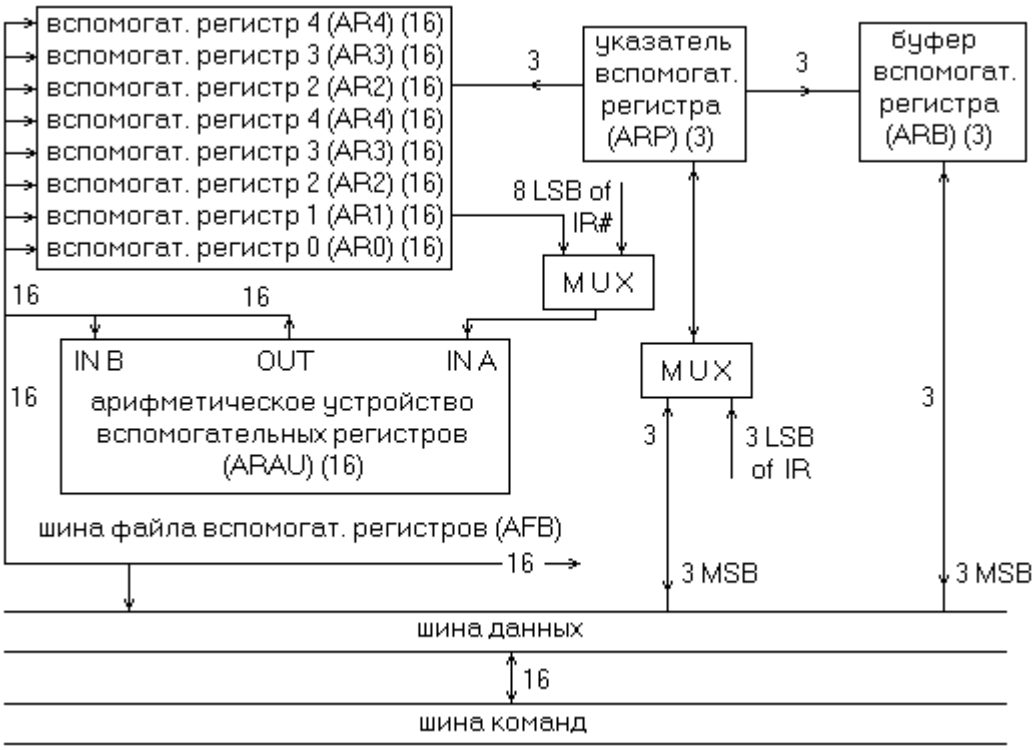


Рисунок 8 - Файл вспомогательных регистров

Вспомогательный регистр AR0 (см. рисунок 8) постоянно подключен к одному из входов ARAU. Другой вход соединен с текущим AR(ARP). ARAU способно выполнять следующие вычисления:

AR(ARP) + AR0	–> AR(ARP)	К содержимому текущего AR прибавляется содержимое AR0.
AR(ARP) – AR0	–> AR(ARP)	Из содержимого текущего AR вычитается содержимое AR0.
AR(ARP) + 1	–> AR(ARP)	Автоинкремент текущего AR на 1.
AR(ARP) – 1	–> AR(ARP)	Автодекремент текущего AR на 1.
AR(ARP)	–> AR(ARP)	AR(ARP) остается без изменений.
AR(ARP) + IR(7-0)	–> AR(ARP)	К содержимому текущего AR прибавляется непосредственная 8-разрядная константа.
AR(ARP) – IR(7-0)	–> AR(ARP)	Из содержимого текущего AR вычитается непосредственная 8-разрядная константа.
AR(ARP) + rcAR0	–> AR(ARP)	Индексация с инверсией битов: к текущему AR прибавляется AR0 с перестановкой разрядов (reverse-carry (rc)).
AR(ARP) – rcAR0	–> AR(ARP)	Индексация с инверсией битов: из текущего AR вычитается AR0 с перестановкой разрядов (reverse-carry (rc)).

Использование ARAU очень эффективно для управления адресацией, т.к. параллельно могут выполняться другие операции. Однако ARAU может применяться также, в качестве обычного арифметического устройства благодаря тому, что файл вспомогательных регистров имеет непосредственную связь с памятью данных. ARAU способно выполнять 16-разрядную беззнаковую арифметику, в то время как CALU выполняет 32-разрядную арифметику в дополнительном коде. Набор инструкций поддерживает переходы в зависимости от результата сравнения AR0 с текущим вспомогательным регистром, выбранным с помощью ARP.

На рисунке 8 также показан буфер указателя вспомогательного регистра (ARB), хранящий содержимое ARP во время программных вызовов и прерываний.

3.7 Статусные регистры ИС 1867ВМ7Т

В ИС 1867ВМ7Т предусмотрены два статусных регистра, содержащих различные биты состояния и управления. Организация статусных регистров приведена на рисунке 9, а описание их разрядов и полей – в таблице 7.

Предусмотрены специальные инструкции для записи их содержимого в память данных (SST и SST1) и для обратной операции – загрузки значений из памяти данных в эти регистры (LST и LST1). Эти инструкции позволяют сохранить состояние устройства (контекст) при обработке прерываний или вызовах подпрограмм.

Инструкции SST и SST1 сохраняют в памяти все без исключения биты статусных регистров ST0 и ST1, а инструкции восстановления статусных регистров из памяти не модифицируют состояние некоторых разрядов. Так, инструкция LST не изменяет состояние битов PDM и INTM, а LST1 не оказывает влияния на биты EMC и IMC. В тоже время инструкция LST1 загружает значения трех старших разрядов и в ARB (ST1[15:13]) и в ARP (ST0[15:13]).

Для всех разрядов и полей статусных регистров предусмотрены связанные с ними инструкции (например, LDP, LARP, SPM, CNFP, STXM, RPDM и т.д.), которые могут менять их состояние индивидуально.

При сбросе все без исключения поля и биты статусных регистров приводятся в известное состояние (см. раздел «Аппаратный сброс»).

Статусный регистр ST0

15 14 13	12	11	10	9	8 7 6 5 4 3 2 1 0
ARP	OV	OVM	PDM	INTM	DP

Статусный регистр ST1

15 14 13	12	11	10	9	8	7	6	5	4	3	2	1 0
ARB	CNF	TC	SXM	C	EMC	IMC	HM	FSM	XF	FO	TXM	PM

Рисунок 9 - Статусные регистры ИС 1867ВМ7Т

Таблица 7 - Описание статусных регистров ИС 1867Вм1т

Поле	Описание
ARP	Указатель вспомогательного регистра. Трехбитное поле выбирает вспомогательный регистр, который используется для косвенной адресации. Когда ARP загружается, его предыдущее значение копируется в ARB, за исключением выполнения инструкции LST. ARP может модифицироваться инструкциями обращения к памяти, использующими косвенную адресацию, а также инструкциями LARP, MAR и LST. Когда ARB загружается инструкцией LST1, это же значение копируется в ARP. При сбросе ARP обнуляется
OV	Бит флага переполнения. Этот бит устанавливается в 1, когда в CALU происходит переполнение. Если переполнение произошло, бит OV остается установленным до тех пор, пока он не будет очищен сбросом, условным переходом по переполнению BV или по его отсутствию BNV или инструкцией LST
OVM	Бит режима переполнения. Этот бит определяет, как будет обрабатываться переполнение CALU. Инструкции SOVM и ROVM устанавливают и сбрасывают этот бит, соответственно. Для модификации бита OVM также может быть использована инструкция LST. OVM=0, переполненный результат загружается в аккумулятор (с потерей старших разрядов). OVM=1, аккумулятор, при переполнении, загружается максимальным положительным или отрицательным значением (в зависимости от направления переполнения)
PDM	Бит управления режимом пониженного энергопотребления определяет режим выполнения инструкции IDLE. Если PDM=1, инструкция IDLE останавливает тактирование процессорного ядра (сигнал CLKOUT1 останавливается в низком уровне), а периферия и контроллер прерываний продолжают работать (сигнал CLKOUT2 повторяет внешний тактовый сигнал X2/CLKIN). Возврат к нормальному выполнению программы осуществляется по аппаратному прерыванию или сбросу. Если сбросить PDM в 0, инструкция IDLE полностью останавливает работу устройства (оба сигнала CLKOUT1 и CLKOUT2 останавливаются в низком уровне); содержимое внутреннего ОЗУ сохраняется, а выход из этого состояния возможен только по сбросу. Этот бит модифицируется только инструкциями RPDM и SPDM. Инструкция LST не изменяет состояние PDM. При сбросе PDM устанавливается в 1
INTM	Бит разрешения прерывания. Этот бит разрешает или запрещает маскируемые прерывания. Бит INTM устанавливается или сбрасывается инструкциями DINT и EINT соответственно. Он не может быть загружен инструкцией LST. Бит INTM не оказывает влияния на сброс или программно инициируемые прерывания (TRAP). Бит INTM устанавливается в 1, после подтверждения прерывания (за исключением инструкции TRAP) и при сбросе. INTM=0, все маскируемые прерывания разрешены; INTM=1, все маскируемые прерывания запрещены

Продолжение таблицы 7

Поле	Описание
DP	Указатель страницы памяти данных. В инструкциях с прямой адресацией полный 16-разрядный адрес данных формируется путем конкатенации 9-битного поля DP с семью младшими битами (LSB) слова инструкции. Запись в DP осуществляют инструкции LST, LDP и LDPK. При сбросе DP обнуляется.
ARB	Буфер указателя вспомогательного регистра. При загрузке ARP, его предыдущее значение копируется в ARB, за исключением выполнения инструкции LST. Когда ARB загружается инструкцией LST1, это же значение копируется в ARP. При сбросе ARB обнуляется
CNF	Бит конфигурации блока B0 внутреннего ОЗУ. Этот бит определяет картирование блока B0 в область данных (CNF=0) или область программ (CNF=1). CNF может быть модифицироваться инструкциями CNFD, CNFP и LST1. При сбросе устанавливается в 0
TC	Флаг тестирования/управления. Бит TC устанавливается в 1 инструкциями BIT или BITT, если проверяемый ими бит, установлен в 1; если выполнено тестируемое условие в инструкции CMPR; если в инструкции NORM выполняются условия ACC=0 или ACC(31)≠ACC(30). Состояние этого бита может использоваться для ветвления программного кода в инструкциях условных переходов BBZ и BBNZ. Бит TC может быть модифицирован (кроме упомянутых ранее) инструкциями STC, RTC и LST1. После сброса бит TC устанавливается в 0
SXM	Режим расширения знака. Если SXM=1, при выходе из масштабирующего сдвигателя над данными выполняется знаковое расширение. Также знаковое расширение происходит в ACC при правом сдвиге по инструкции SFR. Если SXM=0 – режим расширения знака запрещен. Этот бит устанавливается и сбрасывается командами SSXM и RSXM соответственно и загружается инструкцией LST1. При сбросе SXM=1
C	Бит переноса. Этот бит устанавливается в 1, если результат операции сложения генерирует перенос или сбрасывается в 0, если результат операции вычитания генерирует заем. В противном случае, он сбрасывается после операции сложения или устанавливается после операции вычитания, исключая инструкции ADDH и SUBH. Инструкция ADDH может только установить, а инструкция SUBH только очистить бит переноса. Состояние этого бита может использоваться для ветвления программного кода в инструкциях условных переходов BC и BNC. Бит переноса может быть модифицирован инструкциями SC, RC и LST1. После сброса бит C устанавливается в 1

Продолжение таблицы 7

Поле	Описание
EMC	Бит конфигурации внешней памяти. Когда EMC=1, доступ на запись по внешней шине данных сконфигурирован для асинхронного ОЗУ и выполняется, по меньшей мере, за два машинных цикла. Если EMC=0, доступ на запись сконфигурирован для синхронного внешнего ОЗУ и выполняется за один цикл, увеличивая скорость обмена данными. На чтение данных с внешней шины не влияет. Этот бит модифицируется только инструкциями REMC и SEMC. Инструкция LST1 не изменяет состояние EMC. При сбросе EMC=1
IMC	Бит конфигурации внутренней памяти. Когда IMC=1, запись во внутреннее ОЗУ осуществляется в асинхронном режиме и занимает два машинных цикла. Если IMC=0, запись во внутреннее ОЗУ осуществляется в синхронном режиме и выполняется за один цикл, увеличивая скорость работы устройства. На чтение данных из внутреннего ОЗУ не влияет. Этот бит модифицируется только инструкциями RIMC и SIMC. Инструкция LST1 не изменяет состояние IMC. При сбросе IMC=1
HM	Бит управления режимом HOLD. Если HM=1, процессор останавливает выполнение всех внутренних операций после подтверждения HOLD и переводит внешние шины и стробы в третье состояние. При HM=0, активизация HOLD-режима также приводит к переводу внешних шин и стробов в третье состояние, но при этом внутренние операции продолжают выполняться до тех, пока не потребуется доступ в какую-либо область внешней памяти (программ, данных или ввода/вывода). Бит HM может быть модифицирован инструкциями SHM, RHM и LST1. После сброса бит HM устанавливается в 1
FSM	Бит управления режимом кадровой синхронизации последовательного порта. Если FSM=1, каждая операция порта инициируется кадровым синхроимпульсом на FSX/FSR. При FSM=0, порт, после однократной подачи кадрового синхроимпульса, продолжает работать в непрерывном режиме. FSM бит может быть модифицирован инструкциями SFSM, RFSM и LST1. После сброса FSM=1
XF	Состояние вывода XF. Этот бит определяет состояние вывода XF, который используется как выход общего назначения. Этот бит устанавливается в 1 инструкцией SXF и сбрасывается в 0 инструкцией RXF. Бит XF модифицируется инструкцией LST1. При сбросе устанавливается в 1
FO	Бит формата данных последовательного порта. При FO=1, порт передает и принимает 8-ми разрядные слова. Если FO=0, порт работает с 16-ти разрядными данными. Этот бит может быть модифицирован инструкциями FORT или LST1. После сброса FO=0

Окончание таблицы 7

Поле	Описание
TXM	Бит управления режимом передачи последовательного порта. Определяет источник выработки кадрового синхроимпульса FSX. Когда TXM=1, кадровый синхроимпульс передачи данных генерируется самим портом и выдается на вывод FSX. Если необходимо передавать данные по внешнему кадровому синхроимпульсу, бит TXM необходимо сбросить в 0. Кадровый импульс будет приниматься по входу FSX. TXM бит может быть модифицирован инструкциями STXM, RTXM и LST1. После сброса TXM=0.
PM	Режим сдвига произведения. PM определяет коэффициент сдвига содержимого регистра произведения PR перед его выдачей в CALU. При этом само содержимое PR не меняется. Поле PM модифицируется инструкциями SPM и LST1. Биты PM очищаются при сбросе. PM=0, 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 заполняется 0). PM=11, 32-битное произведение передается в CALU или память данных с правым сдвигом на 6 бит (знаковый разряд расширяется).

3.8 Центральное арифметико-логическое устройство (CALU)

Центральное арифметико-логическое устройство (CALU) содержит 16-разрядный масштабирующий регистр сдвига, 16 × 16 аппаратный умножитель, 32-разрядное арифметико-логическое устройство (ALU), 32-разрядный аккумулятор и несколько дополнительных сдвиговых регистров, расположенных как на выходе умножителя, так и выходе из аккумулятора. Функциональная схема CALU представлена на рисунке 10.

Любая операция ALU выполняется в следующей последовательности:

- Данные считываются из ОЗУ на шину данных.
- Данные проходят через масштабирующий сдвиговый регистр и через ALU, в котором выполняются арифметические операции.
- Результат передается в аккумулятор.

Один вход в ALU всегда соединен с выходом аккумулятора, а второй может получать информацию либо из регистра произведения (PR) умножителя, либо загружаться из памяти через масштабирующий сдвиговый регистр.

3.9 Масштабирующий сдвиговый регистр

Масштабирующий сдвиговый регистр имеет 16-разрядный вход с шины данных и 32-разрядный выход в ALU (см. рисунок 11). Сдвиговый регистр выполняет левый сдвиг на 0-16 бит, как указано в команде.

При этом младшие разряды заполняются нулями, старшие разряды заполняются либо нулями, либо расширением знака, в зависимости от состояния бита SXM (режим расширения знака) в регистре состояния ST0.

ИС 1867ВМ7Т также имеет несколько других сдвиговых регистров, которые позволяют выполнить числовое масштабирование, выделение бита, расширенную арифметику и предотвращать переполнение. Эти сдвиговые регистры соединены с выходами умножителя и аккумулятора.

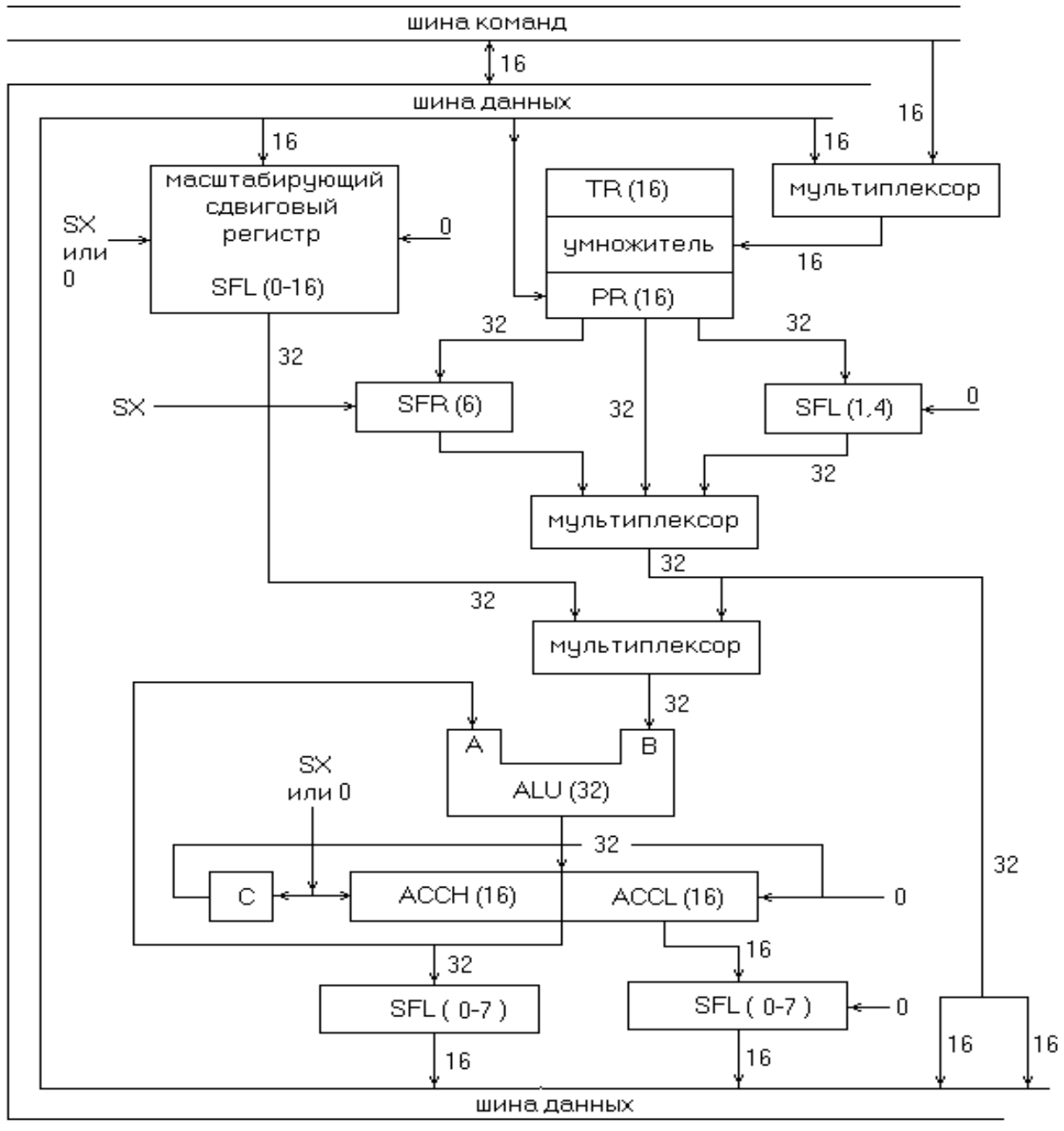


Рисунок 10 - Центральное арифметико-логическое устройство (CALU)

3.10 Арифметико-логическое устройство (ALU) и аккумулятор (ACC)

32-разрядные ALU и аккумулятор ИС 1867ВМ7Т реализуют широкий спектр арифметических и логических операций, большинство из которых исполняется в течение одного командного цикла. Результат любой операции, выполняемой в ALU, помещается в аккумулятор. Данные, поступающие на обработку в ALU, могут быть предварительно промасштабированы в масштабирующем сдвиговом регистре.

На один из входов ALU всегда подается содержимое аккумулятора, а другой вход может быть соединен либо с регистром произведения (PR) умножителя, либо с выходом масштабирующего сдвигового регистра.

32-разрядный аккумулятор (рисунок 10) разделен на два 16-разрядных слова для удобства при сохранении его в памяти данных: ACCH (старшее слово) и ACCL (младшее слово).

Дополнительный сдвиговый регистр, соединенный с выходом аккумулятора обеспечивает левый сдвиг от 0 до 7 разрядов. Этот сдвиг осуществляется перед, выдачей данных на шину для сохранения их в памяти. При этом содержимое аккумулятора не изменяется. При сдвиге влево ACCH, младшие разряды выбираются из ACCL , а старшие теряются. При сдвиге влево ACCL, младшие разряды заполняются нулями, а старшие теряются.

Процессор 1867ВМ7Т поддерживает операции с плавающей точкой для вычислений, требующих большого динамического диапазона. Команда нормализации (NORM) нормализует числа с фиксированной точкой, находящиеся в аккумуляторе, путем выполнения левого сдвига. Команда LACT (загрузить аккумулятор со сдвигом, определяемым Т-регистром) денормализует числа с плавающей точкой с помощью арифметического левого сдвига мантиссы через входной масштабирующий сдвиговый регистр. Коэффициент сдвига в этом случае – это значение экспоненты, определяемое четырьмя младшими разрядами Т-регистра (TR). Команды ADDT и SUBT (прибавить к/вычесть из аккумулятора со сдвигом, заданным в Т-регистре) также позволяют обрабатывать числа с 16-разрядной мантиссой и 4-разрядной экспонентой.

Режим насыщения при переполнении может быть запрограммирован при помощи команд SOVM/ROVM (установить/сбросить режим переполнения). Если аккумулятор находится в режиме насыщения, и произошло переполнение, устанавливается флаг переполнения, и в аккумулятор загружается наибольшее положительное (>7FFFFFFFh) или отрицательное (>80000000h) число, в зависимости от знака переполнения. Если режим насыщения не установлен (бит OVM сброшен), результат операции, приведшей к переполнению, загружается в аккумулятор без изменений (т.е. с потерей старших разрядов). Необходимо отметить, что логические операции не могут вызывать переполнения.

В ИС 1867ВМ7Т реализованы команды ветвления, в зависимости от состояния ACC (больше/меньше/равен нулю, переполнен, сгенерировал перенос). Кроме того, инструкции BACC (перейти по адресу, расположенному в аккумуляторе) и CALA (вызвать подпрограмму по адресу, расположенному в аккумуляторе) позволяют вычислять адрес ветвления программного кода. Команды BIT/BITT, не

изменяя содержимого аккумулятора, позволяют тестировать значение конкретных разрядов в любом слове памяти данных.

В аккумуляторе предусмотрен бит переноса, который устанавливается или сбрасывается в зависимости от операций выполняемых устройством. Бит переноса позволяет более эффективно выполнять умножение, сложение и вычитание. Также он удобен при работе с переполнениями. Бит переноса изменяется при выполнении большинства арифметических команд, а также команд сдвига и вращения аккумулятора. Но он не изменяется при загрузке аккумулятора, логических операциях, или при других неарифметических или управляющих операциях. Он также не изменяется при выполнении команд умножения MPY, MPYK и MPYU, но изменяется командами MAC и MACD. Примеры работы с битом переполнения представлены на рисунке 11.

```
C  MSB       LSB
X    FFFF FFFF     ACC
   +          1
1    0000 0000

X    7FFF FFFF     ACC
   +          1   (OVM=0)
0    8000 0000

1    0000 0000     ACC
   +          0   (команда
                   ADDC)
0    0000 0001

C  MSB       LSB
X    0000 0000     ACC
   -          1
0    FFFF FFFF

X    8000 0000     ACC
   -          1   (OVM=0)
1    7FFF FFFF

0    FFFF FFFF     ACC
   -          1   (команда
                   ADDC)
1    FFFF FFFE
```

Рисунок 11 - Примеры операций с битом переноса

Значение, прибавляемое или вычитаемое из аккумулятора, показанное в примере на рисунке 11, может быть предварительно пропущено через сдвиговый регистр. Бит переноса устанавливается, если при сложении в аккумуляторе генерируется перенос, или сбрасывается, если при вычитании возникает заем из старшего разряда. Во всех остальных случаях он сбрасывается при сложении и устанавливается при вычитании. Команды ADDC (прибавить с переносом) и SUBB (вычесть с заемом) позволяют воспользоваться значением переноса, полученным в предыдущих командах сложения/ вычитания.

Одно исключение при работе с битом переноса из примеров, показанных на рисунке 11, связано с использованием команд ADDH (прибавить к старшему слову аккумулятора) и SUBH (вычесть из старшего слова аккумулятора). ADDH может только устанавливать бит переноса, если возникло переполнение, а SUBH может только сбрасывать его, если сгенерирован заем; иначе эти команды не изменяют содержимого бита переноса.

Две команды условного переходов BC и BNC обеспечивают ветвление, в зависимости от состояния бита переноса. Команды SC, RC и LST1 также могут изменять содержимое бита переноса. После аппаратного сброса бит переноса всегда установлен в 1.

В аккумуляторе также реализована возможность внутреннего одноразрядного сдвига содержимого вправо или влево (команды SFR и SFL) или вращения содержимого через бит переноса (команды ROR и ROL). Бит SXM оказывает влияние на режим выполнения команды SFR (сдвинуть аккумулятор вправо). Когда SXM=1, SFR осуществляет арифметический правый сдвиг, сохраняющий знак аккумулятора. Когда SXM=0, SFR выполняет логический сдвиг, выдвигая младший разряд и заполняя нулем старший. Команда SFL (сдвинуть аккумулятор влево) выполняется одинаково, вне зависимости от значения SXM, выдвигая наружу старший разряд и заполняя нулем младший. Значение SXM не оказывает влияния на выполнение правого и левого кольцевых сдвигов (ROR и ROL). Команды RPT и RPTC могут быть использованы с командами сдвига для их многократного повторения.

3.11 Умножитель, T- и P-регистры

В ИС 1867ВМ7Т реализован аппаратный 16 × 16 умножитель, работающий с числами в дополнительном коде, который вычисляет 32-разрядное произведение за один командный цикл. С умножителем связаны следующие два регистра:

- 16-разрядный временный регистр (TR) хранит один из сомножителей;
- 32-разрядный регистр произведения (PR) хранит произведение.

В стандартном варианте для умножения используются две команды. Команда LT (загрузить Т-регистр), загружая TR с шины данных, обеспечивает один из сомножителей, а команда MPY (умножить) обеспечивает другой сомножитель, также считывая его с шины данных. Умножение может также выполняться с использованием непосредственного операнда. И в том и в другом случае произведение вычисляется за два командных цикла.

Две команды умножения с накоплением (MAC и MACD) полностью реализуют вычислительные возможности умножителя, позволяя подавать оба сомножителя одновременно через шины команд и данных. Использование команд MAC и MACD с командами повторения RPT и RPTK позволяет выполнять операции умножения с накоплением за один командный цикл. Команды MAC и MACD могут выбирать операнды, как из внутренней, так и из внешней памяти и использовать одну и ту же область внутреннего блока ОЗУ. Однако необходимо отметить, что функция перемещения данных (DMOV), входящая в команду MACD, не работает с внутренними картированными в памяти регистрами и в пространстве внешней памяти данных. В этом случае команда MACD выполняется по алгоритму команды MAC.

Команды SQRA (возвести в квадрат/сложить) и SQRS (возвести в квадрат/вычесть) подают одно и то же значение с шины данных на оба входа умножителя для вычисления квадрата.

Команда MPYU осуществляет беззнаковое умножение, необходимое при выполнении операций с повышенной точностью. Беззнаковое содержимое Т-регистра умножается на беззнаковое содержимое адресуемой ячейки памяти, результат помещается в Р-регистр. Это позволяет поместить операнды, имеющие

длину более 16-ти разрядов, в две ячейки памяти и, обрабатывая их раздельно, получать произведения длиной более 32-х разрядов.

После умножения двух 16-разрядных чисел в дополнительном коде получится 32-разрядное произведение, которое размещается в 32-разрядном регистре произведения (PR). Содержимое PR может затем быть передано в ALU без изменений, либо с предварительным сдвигом. Возможны четыре режима сдвига произведения (PM). Поле PM (статусный регистр ST1[1:0]) определяет режим сдвига, как показано в таблице 8:

Таблица 8 - Режимы сдвига произведения (PM)

Содержимое PM	Режим сдвига
00	Нет сдвига
01	Левый сдвиг на 1 разряд
10	Левый сдвиг на 4 разряда
11	Правый сдвиг на 6 разрядов

Левые сдвиги применяются в операциях с дробными числами и для выравнивания дробных произведений. Например, при умножении двух нормированных 16-разрядных чисел или двух чисел в формате Q15 произведение будет содержать два знаковых разряда, один из которых избыточный. Левый сдвиг на 1 бит удалит этот дополнительный бит, перед выдачей произведения в ALU. Таким образом, результат будет приведен к формату сомножителей.

Если для умножения использовалась инструкция MPYK (умножение на 13-разрядный непосредственный операнд), то произведение 16-разрядного нормированного числа в дополнительном коде или числа в формате Q15 с 13-разрядной константой в дополнительном коде будет иметь пять знаковых разрядов. Левый сдвиг на четыре разряда корректно выровняет произведение перед выдачей его в ALU.

Использование правого сдвига позволяет выполнить 128 последовательных команд умножения с накоплением в аккумуляторе без возникновения его переполнения. Необходимо отметить, что правый сдвиг всегда выполняется с расширением знака не зависимо от состояния бита SXM.

Нулевой коэффициент сдвига используется при работе с целыми числами или при операциях, требующих 32-разрядной точности.

Четыре младших значащих бита T-регистра (TR) могут использоваться для задания коэффициента сдвига масштабирующего сдвигового регистра в инструкциях LACT/ADDT/SUBT (загрузить/прибавить/вычесть к(из) аккумулятора со сдвигом, указанным в TR). Эти команды используются при операциях с плавающей точкой, в которых числа должны быть предварительно денормализованы, т.е. представлены в формате с фиксированной точкой. Команда тестирования бита (BITT) позволяет проверить один разряд в слове данных, номер которого указывается содержимым четырех младших разрядов TR.

4 Интерфейс памяти

Модуль управления памятью ИС 1867ВМ7Т обеспечивает интерфейс ко всем блокам внутренней и внешней памяти. Для определения физической области памяти, к которой будет осуществляться доступ (внутреннее ОЗУ: блоки B0, B1, B2, внешняя память), дешифрируются адресные шины программ и данных.

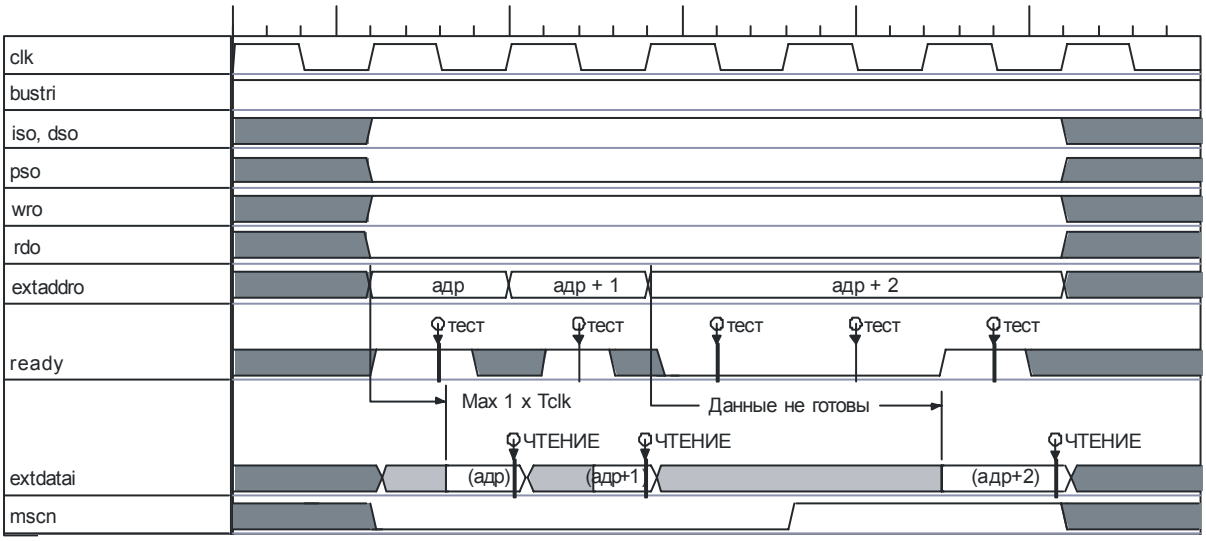
По запросу блока управления (сиквенсора), интерфейс памяти выполняет операции чтения/записи в/из областей памяти программ, данных или портов ввоы/вывода. При операции чтения данные из памяти передаются на внутреннюю шину. При операции записи, интерфейс памяти записывает информацию с внутренней шины данных в указанные области памяти программ, данных или портов вводы/вывода.

Если выполняется обращение к медленным внешним устройствам, интерфейс памяти также обеспечивает генерацию циклов ожидания. Активный уровень сигнала на выходе MSC# указывает на доступ к памяти и может использоваться внешней логикой для перевода ИС 1867ВМ7Т в состояние ожидания.

Флаг конфигурации памяти CNF управляет отображением блока B0 внутреннего ОЗУ в область памяти программ или памяти данных. При CNF=0 блок B0 отображен в память данных по адресу 0200h. При CNF=1, отображается в память программ по адресу FF00h (см. рисунок 4).

На рисунках 12 – 24 приведены временные диаграммы внутренних сигналов при работе с памятью. Показаны и циклы ожидания внешних устройств.

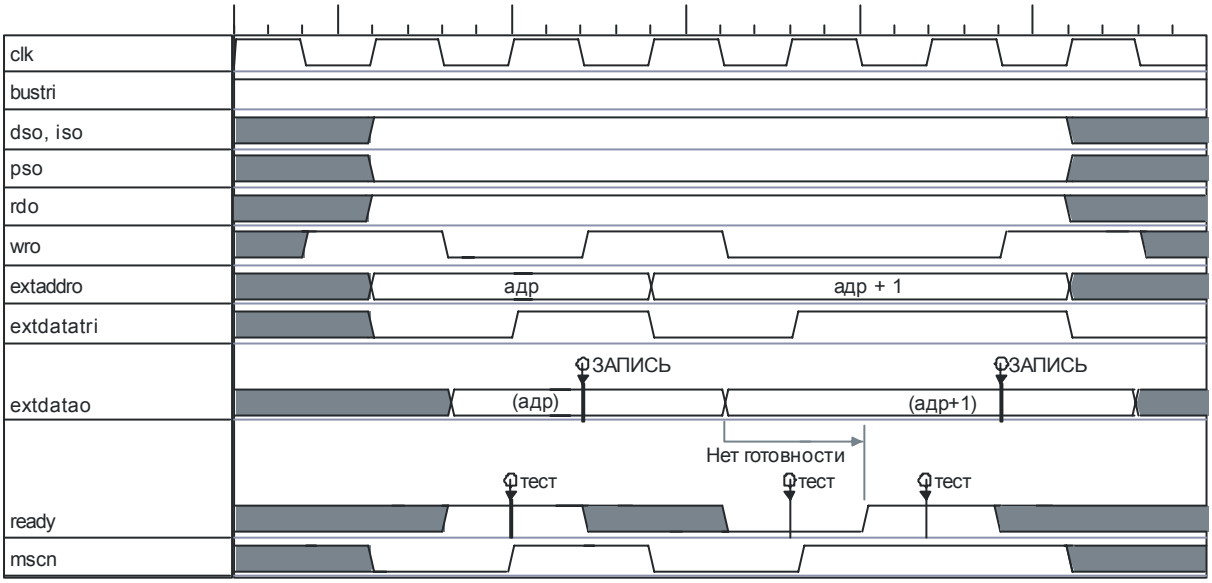
Память программ – операция чтения



адр - адрес памяти программ
(адр) - содержание памяти программ, по указанному адресу
тест - тест состояния сигнала ready - готовность внешней памяти программ
ЧТЕНИЕ - момент чтения данных
Tclk - период тактового сигнала

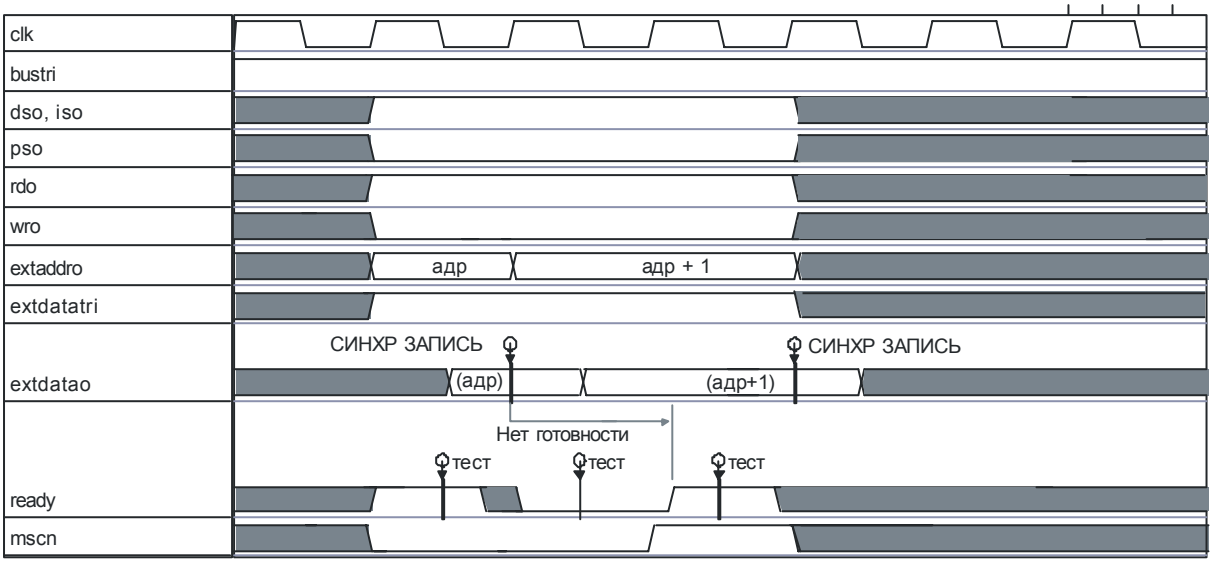
Рисунок 12 - Временная диаграмма чтения внешней памяти программ

Память программ – операция записи



адр - адрес памяти программ
(адр) - данные, записываемые по указанному адресу памяти программ
тест - тест состояния сигнала ready - готовность внешней памяти программ
ЗАПИСЬ - момент записи данных во внешнюю память

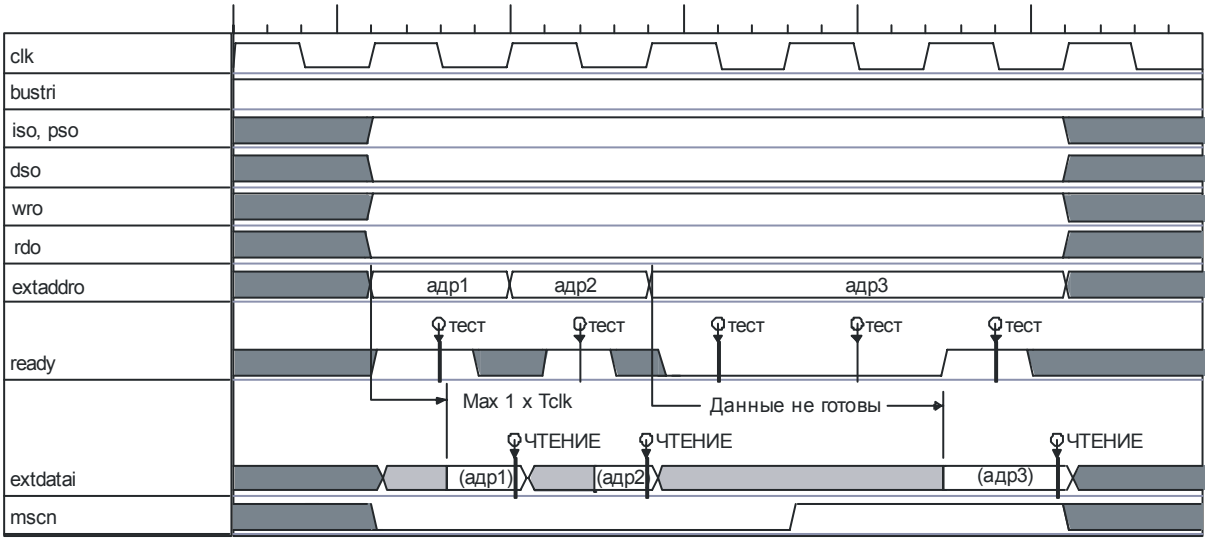
Рисунок 13 - Временная диаграмма записи во внешнюю память программ (после инструкции SEMC)



адр - адрес памяти программ
(адр) - данные, записываемые по указанному адресу памяти программ
тест - тест состояния сигнала ready - готовность внешней памяти программ
СИНХР ЗАПИСЬ - момент записи данных во внешнюю память

Рисунок 14 - Временная диаграмма записи во внешнюю память программ (после инструкции REMC)

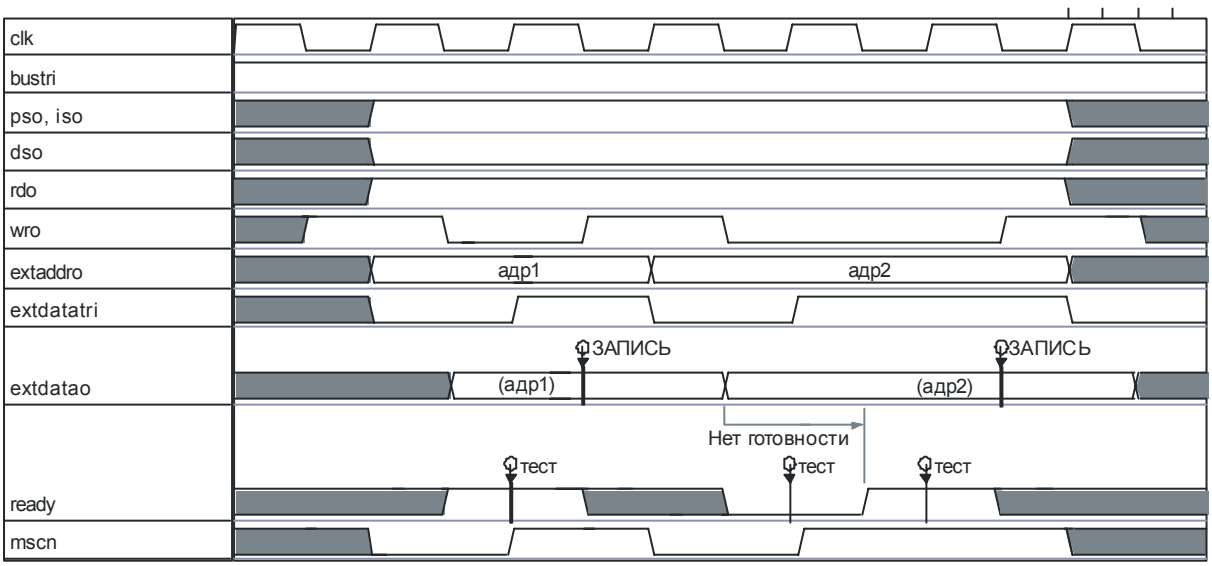
Память данных – операция чтения



адр1, адр2, адр3 - адреса памяти данных
(адр1), (адр2), (адр3) - содержание памяти данных, по указанному адресу
тест - тест состояния сигнала ready
ЧТЕНИЕ - момент чтения данных
Tclk - период тактового сигнала

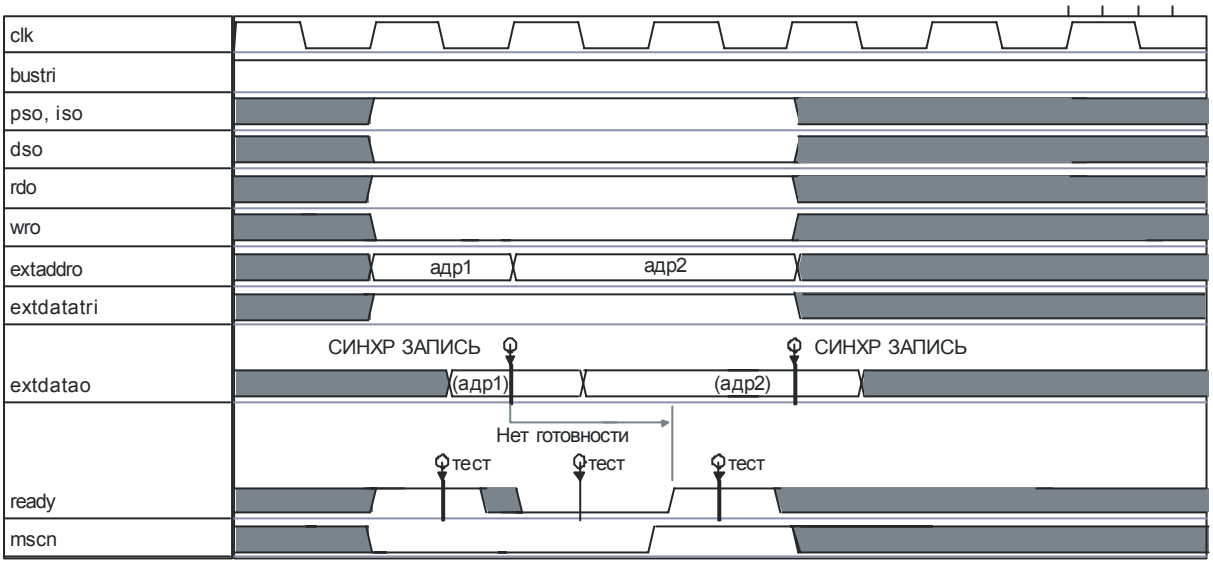
Рисунок 15 - Временная диаграмма чтения внешней памяти данных

Память данных – операция записи



адр1, адр2 - адреса памяти данных
(адр1), (адр2) - данные, записываемые по указанному адресу памяти данных
тест - тест состояния сигнала ready
ЗАПИСЬ - момент записи данных во внешнюю память

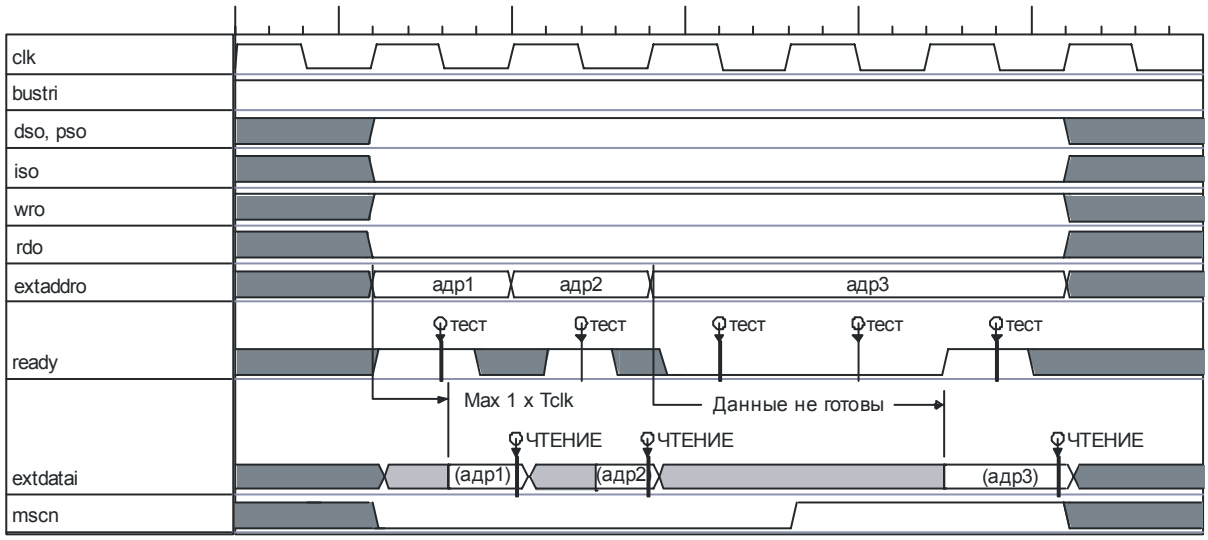
Рисунок 16 - Временная диаграмма записи во внешнюю память данных (после инструкция SEMC)



адр1, адр2 - адреса памяти данных
(адр1), (адр2) - данные, записываемые по указанному адресу памяти данных
тест - тест состояния сигнала
СИНХР ЗАПИСЬ - момент записи данных во внешнюю память

Рисунок 17 - Временная диаграмма записи во внешнюю память данных (после инструкции REMC)

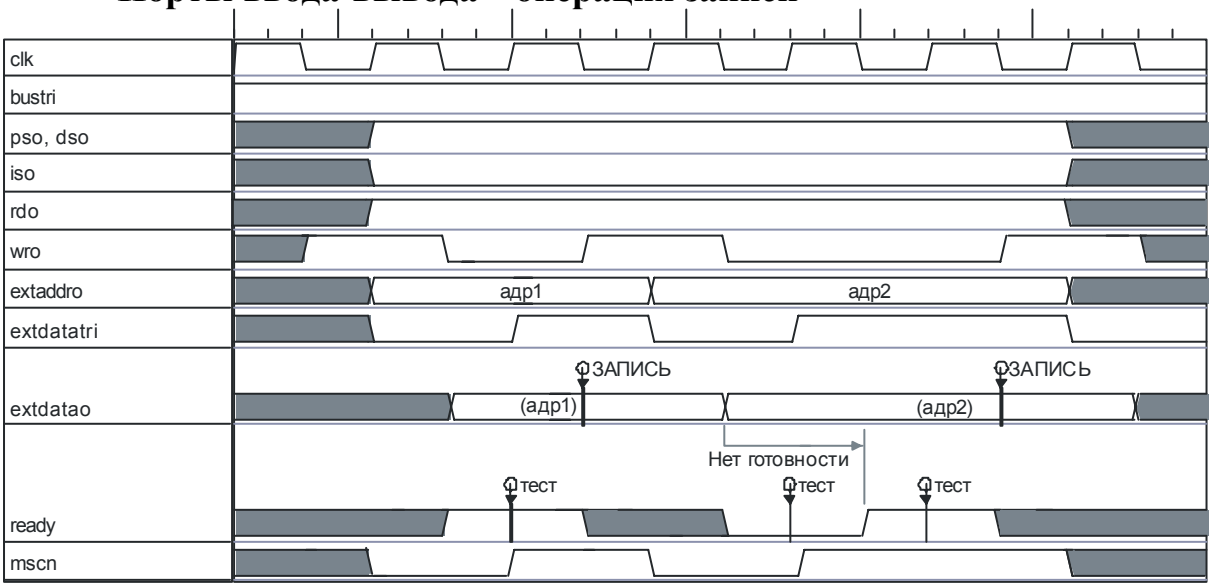
Порты ввода-вывода – операция чтения



адр1, адр2, адр3 - адреса портов ввода/вывода
(адр1), (адр2), (адр3) - данные, по указанному адресу
тест - тест состояния сигнала ready
ЧТЕНИЕ - момент чтения данных

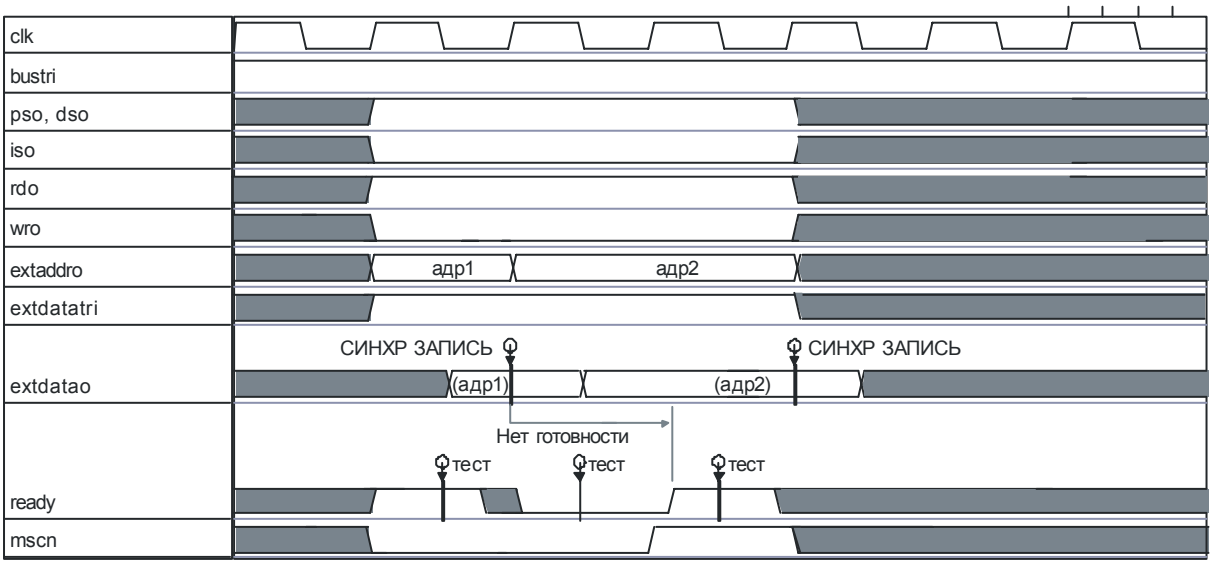
Рисунок 18 - Временная диаграмма чтения портов ввода-вывода

Порты ввода-вывода – операция записи



адр1, адр2 - адреса портов ввода/вывода
(адр1), (адр2) - данные, записываемые по указанному адресу
тест - тест состояния сигнала ready
ЗАПИСЬ - момент записи данных

Рисунок 19 - Временная диаграмма записи в порты ввода/вывода (после инструкции SEMC)



адр1, адр2 - адреса портов ввода/вывода
(адр1), (адр2) - данные, записываемые по указанному адресу
тест - тест состояния сигнала ready
СИНХР ЗАПИСЬ - момент записи данных

Рисунок 20 - Временная диаграмма записи в порты ввода-вывода (после инструкции REMC)

Режим прямого доступа к памяти (DMA)

ИС 1867ВМ7Т поддерживает режим прямого доступа к памяти (DMA) для внешних памяти программ, памяти данных и портов ввода-вывода.

Сигналы HOLD# и HOLDA# могут быть использованы другим устройством, чтобы взять на себя управление над внешними шинами 1867ВМ7Т.

Когда на вход HOLD# подается активный (низкий) уровень, процессор:

- выставляет на выходе HOLDA# активный (низкий) уровень сигнала (подтверждение прямого доступа к памяти);
- одновременно переводит в третье состояние выводы шины адреса, шины данных и всех управляющих сигналов.

Режим прямого доступа к памяти не влияет на работу последовательного порта и вывода XF.

Описание временной диаграммы режима прямого доступа к памяти

Состояние входа HOLD# тестируется на каждом положительном фронте тактового сигнала. После обнаружения активного (низкого) уровня сигнала на входе HOLD#, процессору требуется, по крайней мере, три цикла тактового сигнала, чтобы войти в режим DMA. Все управляющие сигналы и шины переводятся в третье состояние, на выходе HOLDA# выставляется активный (низкий) уровень сигнала.

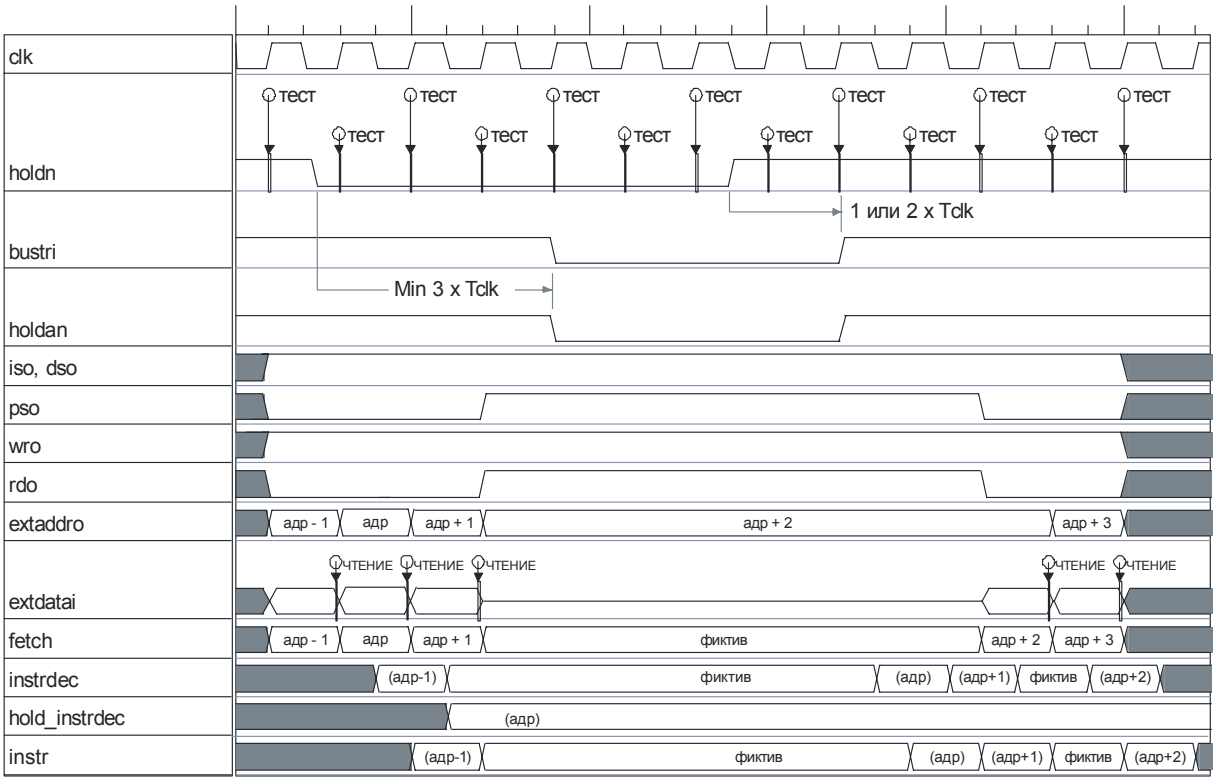
Если статусный бит HM установлен в 0, ИС 1867ВМ7Т продолжает выполнение программы, пока не потребуется доступ к любой внешней области памяти программ, памяти данных или портов ввода/вывода.

Если статусный бит HM установлен в 1, процессор останавливает выполнение программы независимо от обращения к любой области памяти после завершении выполнения текущей инструкции или цикла повтора инструкции.

ИС 1867ВМ7Т находится в режиме DMA до тех пор, пока сохраняется низкий уровень сигнала на входе HOLD#.

При завершении режима DMA процессор выставляет на выходе HOLDA# высокий уровень и берет на себя управление над внешними шинами.

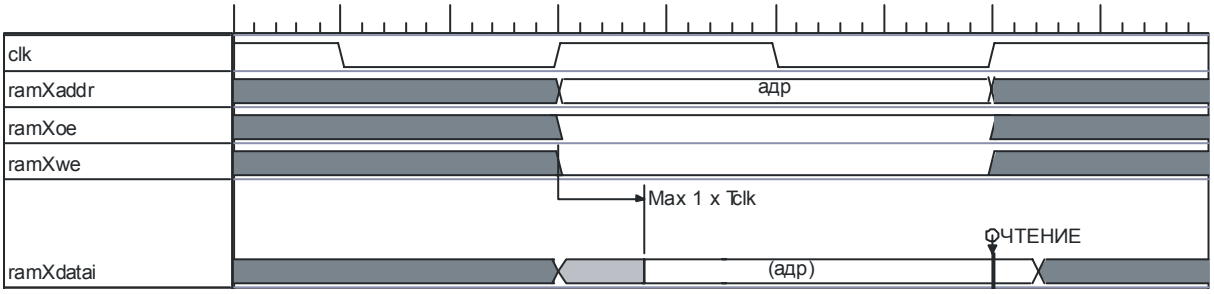
Следующее чтение памяти программ/данных начинается после еще двух циклов тактового сигнала.



адр - адреса памяти программ
(адр) - данные по указанному адресу памяти программ
тест - тест состояния сигнала holdn
ЗАПИСЬ - момент записи данных во внешней памяти
Tclk - период тактового сигнала
instrdec - регистр декодирования инструкций
hold_instrdec - инструкция, выполняемая при выходе из режима
instr - регистр инструкций

Рисунок 21 - Временная диаграмма режима прямого доступа к памяти

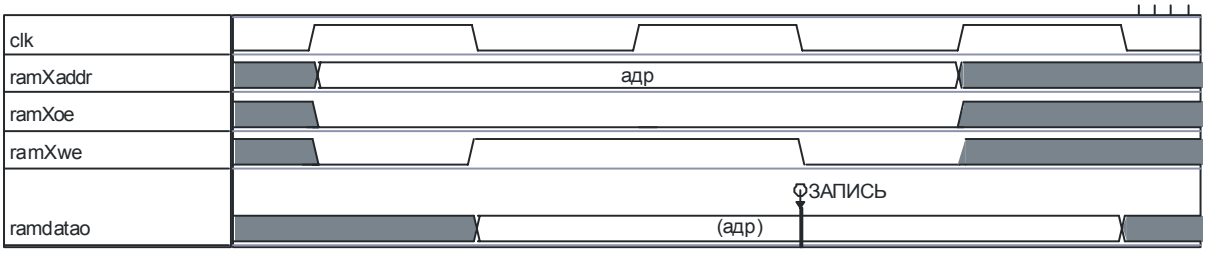
Внутреннее ОЗУ – чтение



адр - адреса внутреннего ОЗУ
(адр) - данные по указанному адресу ОЗУ
ЧТЕНИЕ - момент чтения данных
ramXoe - разрешение доступа к внутреннему ОЗУ
ramXwe - сигнал записи во внутреннее ОЗУ

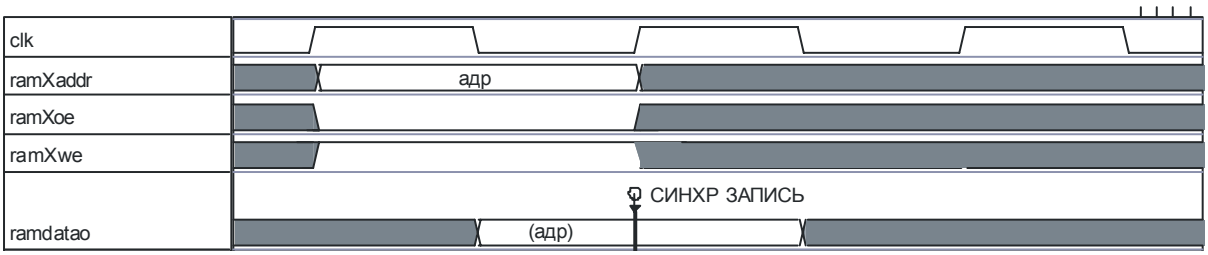
Рисунок 22 - Временная диаграмма чтения внутреннего ОЗУ

Внутреннее ОЗУ – запись



адр - адреса внутреннего ОЗУ
(адр) - данные по указанному адресу ОЗУ
ЗАПИСЬ - момент чтения данных
ramXoe - разрешение доступа к внутреннему ОЗУ
ramXwe - сигнал записи во внутреннее ОЗУ

Рисунок 23 - Временная диаграмма записи во внутреннее ОЗУ (после инструкции SIMC)



адр - адреса внутреннего ОЗУ
(адр) - данные по указанному адресу ОЗУ
СИНХР ЗАПИСЬ - момент чтения данных
ramXoe - разрешение доступа к внутреннему ОЗУ
ramXwe - сигнал записи во внутреннее ОЗУ

Рисунок 24 - Временная диаграмма записи во внутреннее ОЗУ (после инструкции RIMC)

4.1 Глобальная память

Сигнал запроса шины BR# и регистр глобального распределения памяти GREG также используются ИС 1867ВМ7Т для управления памятью.

Регистр GREG определяет размер области внешней памяти данных, которая будет рассматриваться как глобальная память (совместно используемая память с другими процессорами в мультипроцессорной системе). Это – восьмиразрядный картированный в память (по адресу 0005h) регистр. На основании значения регистра GREG при каждом доступе (чтение или запись) к глобальной памяти процессор вырабатывает сигнал BR#, который используется внешней логикой для арбитража доступа к совместно используемой памяти. Длительность цикла работы с памятью регулируется сигналом READY. Сигнал BR# остается активным (низкий уровень) во время всего цикла доступа, даже если доступ удлиняется при использовании внешним устройством состояния ожидания.

В таблице 9 приведены допустимые значения регистра GREG и соответствующее распределение памяти данных на локальные и глобальные области.

Запись в регистр GREG значений, отличных от приведённых в таблице 9, вызовет фрагментацию глобальной памяти.

Таблица 9 - Конфигурация глобальной памяти

Значение GREG	Локальная память		Глобальная память	
	Диапазон адресов	Объем, слов	Диапазон адресов	Объем, слов
000000-	0–FFFFh	64К	-	0
10000000	0-7FFFh	32К	8000h-FFFFh	32К
11000000	0–BFFFh	48К	C000h-FFFFh	16К
11100000	0–DFFFh	56К	E000h-FFFFh	8К
11110000	0–EFFFh	60К	F000h-FFFFh	4К
11111000	0-F7FFh	62К	F800h-FFFFh	2К
11111100	0–FBFFh	63К	FC00h-FFFFh	1К
11111110	0–FDFFh	63,5К	FE00h-FFFFh	0,5К
11111111	0–FEFFh	63,75К	FF00h-FFFFh	0,25К

При доступе к области памяти, которая определяется значением GREG регистра как глобальная, всегда формируется сигнал BR#. Это позволяет, путём программирования GREG и незначительной модификации внешней логики доступа к данным, увеличить общий объём адресуемой памяти данных в однопроцессорных конфигурациях (максимум до 96 К слов).

5 Аппаратный сброс

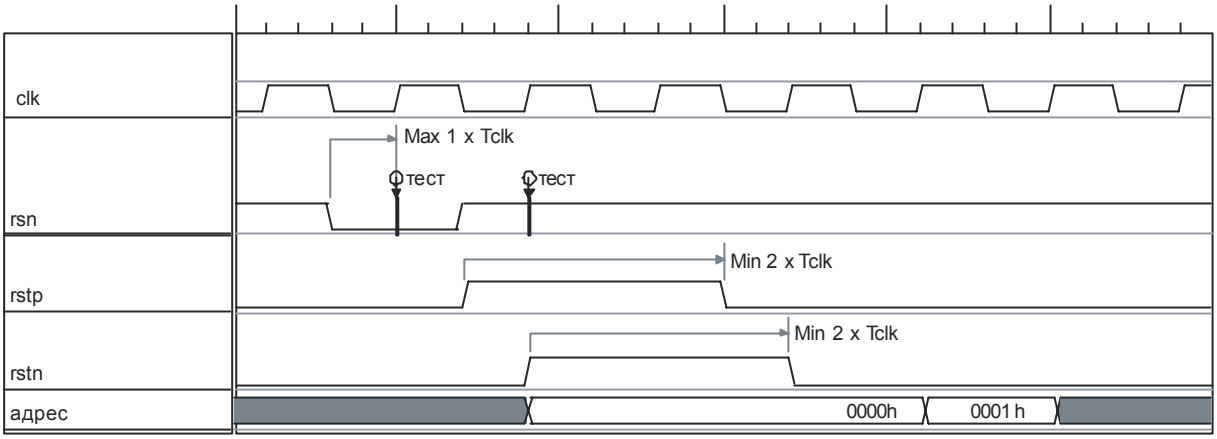
Вход RS# - вход аппаратного сброса. Аппаратный сброс, отрабатываемый как немаскируемое прерывание, переводит ИС 1867ВМ7Т в известное начальное состояние. Наличие активного (низкого) уровня сигнала на входе RS# проверяется на каждом положительном фронте тактового сигнала X2/CLKIN. После обнаружения активного уровня, через цикл тактового сигнала процессор вырабатывает два внутренних сигнала сброса «rstp» и «rstn» (активный уровень – высокий) для элементов, управляемых по положительному и отрицательному фронтам тактового сигнала. Эти сигналы, в свою очередь в течение одного цикла тактового сигнала X2/CLKIN, приводят всю внутреннюю синхронную логику в известное состояние. В таблице 10 перечислены значения регистров, устанавливаемые во время аппаратного сброса.

Таблица 10 - Инициализация регистров 1867ВМ7Т

Регистр	Значение
PC	0000h
PFC	0000h
RPTC	00h
ST0	0600h
ARP	000
OV	0
OVM	0
PDM	1
INTM	1
DP	000000000
ST1	07F0h
ARB	000
CNF	0
TC	0
SXM	1
C	1
EMC	1
IMC	1
HM	1
FSM	1
XF	1
FO	0
TXM	0
PM	00
ACC	00000000h
TR	0000h
PR	00000000h
AR0÷AR7	0000h
TCR	000h
TIM	FFFFh
PRD	FFFFh
GREG	00h
IMR	00000
DRR	0000h
DXR	0000h
IFR	000000
Все уровни стека	0000h

Временная диаграмма сброса

Для перевода 1867ВМ7Т в режим сброса необходимо присутствие активного уровня на входе RS# в течение одного цикла тактового сигнала.



тест - тест состояния сигнала rsn
rstn, rstp - внутренние сигналы сброса
Tclk - период тактового сигнала

Рисунок 25 - Временная диаграмма аппаратного сброса

6 Периферийные устройства

6.1 Таймер

Таймер – это внутрикристальный счетчик, который состоит из трех регистров и может использоваться для генерирования прерываний с заданной периодичностью.

Регистры таймера

В таблице 11 представлены три регистра таймера:

Таблица 11 - Регистры таймера

Адрес	Регистр	Описание
0002h	TIM	Таймерный регистр
0003h	PRD	Регистр периода таймера
0006h	TCR	Регистр управления таймером

- Регистр таймера (TIM). 16-разрядный регистр таймера загружается значением регистра периода PRD, уменьшается на единицу.
- Регистр периода таймера (PRD). 16-разрядный регистр периода таймера используется для перезагрузки регистра таймера (TIM).
- Регистр управления таймером (TCR). 16-разрядный регистр управления таймером содержит биты управления и состояния таймера. Формат TCR приведён на рисунке 26; функциональное назначение его битов и полей описано в таблице 12.

15-10	9-6	5	4	3-0
Резерв	**PSC**	**TRB**	**TSS**	**TDDR**

Рисунок 26 - Диаграмма регистра управления таймером TCR

Таблица 12 - Описание разрядов управляющего регистра таймера

Разряды	Название	Значение после сброса	Функция
15-10	**Резерв**	-	Неиспользуемые разряды. При чтении всегда 000000.
9-6	**PSC**	0000	Предварительный счетчик таймера. Определяет счет для основного счетчика таймера. Когда содержимое PSC уменьшается до нуля или после сброса таймера, PSC загружается содержимым TDDR, а содержимое TIM уменьшается на единицу. Разряды доступны только для чтения.
5	**TRB**	0	Перезагрузка таймера. Установка разряда TRB в 1 инициализирует таймер, приводя к загрузке: - регистра TIM значением из регистра PRD; - счетчика PSC значением из TDDR. При чтении регистра TCR значение разряда TRB всегда 0.
4	**TSS**	0	Останов таймера. Запрещает или разрешает работу таймера: - TSS = 0 запуск таймера; - TSS = 1 останов таймера. При сбросе TSS устанавливается в 0, и таймер начинает отсчёт.
3-0	**TDDR**	0000	Коэффициент для предварительного делителя частоты таймера. Когда значение счетчика PSC уменьшается до 0, значение TDDR загружается в PSC.

Операции таймера

Таймер – это работающий на уменьшение (декремент) счетчик, который используется для генерации периодических прерываний ИС 1867ВМ7Т. Каждый раз, когда содержимое таймера уменьшается до нуля, генерируется прерывание по таймеру (TINT) и счетчик перезагружается содержимым PRD.

На рисунке 27 представлена структурная схема таймера. Она состоит из двух блоков: блока основного таймера, содержащего PRD и TIM, блока предварительного счетчика, содержащего поля TDDR и PCS. Таймер тактируется синхронизацией CPU (X2/CLKIN).

Содержимое регистра PRD загружается в счетчик таймера TIM, когда последний уменьшается до нуля. Содержимое регистра PRD также загружается при сбросе устройства (SRESET#) или при индивидуальном сбросе таймера (установ-

кой TRB в 1). Каждый выходной тактовый импульс из блока предварительного счетчика уменьшает содержимое TIM на единицу. Выход блока основного счетчика – сигнал прерывания по таймеру (TINT), который пересылается в CPU или на внешний вывод TOUT. Длительность импульса TOUT эквивалентна периоду CLKOUT.

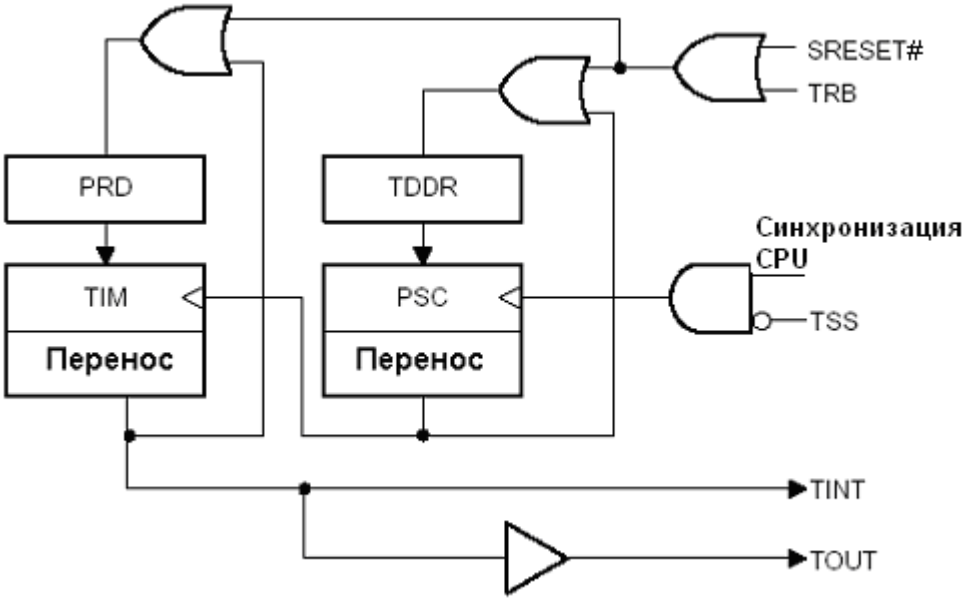


Рисунок 27 - Структурная схема таймера

Частоту прерываний таймера (TINT) можно выразить формулой:

$$TINT = \frac{1}{tc(C) \cdot U \cdot V} = \frac{1}{tc(C) \cdot (TDDR + 1) \cdot (PRD + 1)},$$

где: $tc(C)$ – период X2/CLKIN;

U – сумма содержимого TDDR+1;

V – сумма содержимого TDDR+1.

Счетчик PSC загружается значением из TDDR, когда PSC уменьшается до нуля. Содержимое TDDR также загружается в PSC при сбросе процессора или установкой бита TRB. Содержимое PSC можно считать, обратившись к регистру TCR, но нельзя загрузить непосредственно.

Таймер может быть остановлен, используя бит TSS для отключения входа синхронизации в таймер. Это позволяет сократить уровень потребляемой процессором мощности. Таймер обеспечивает удобный и эффективный способ генерации системного времени для аналогового интерфейса.

Инициализация таймера:

- Остановить таймер, записав 1 в бит TSS регистра TCR.
- Загрузить PRD.
- Установить бит TSS в 0 и бит TRB в 1, для перезагрузки периода таймера.

При сбросе TIM и PRD устанавливаются в максимальное значение FFFFh. Поле TDDR регистра TCR очищается в ноль и таймер запускается.

6.2 Последовательный порт

В состав 1867ВМ7Т входит полнодуплексный синхронный последовательный порт для коммуникаций с другими устройствами или процессорами в мультипроцессорных системах.

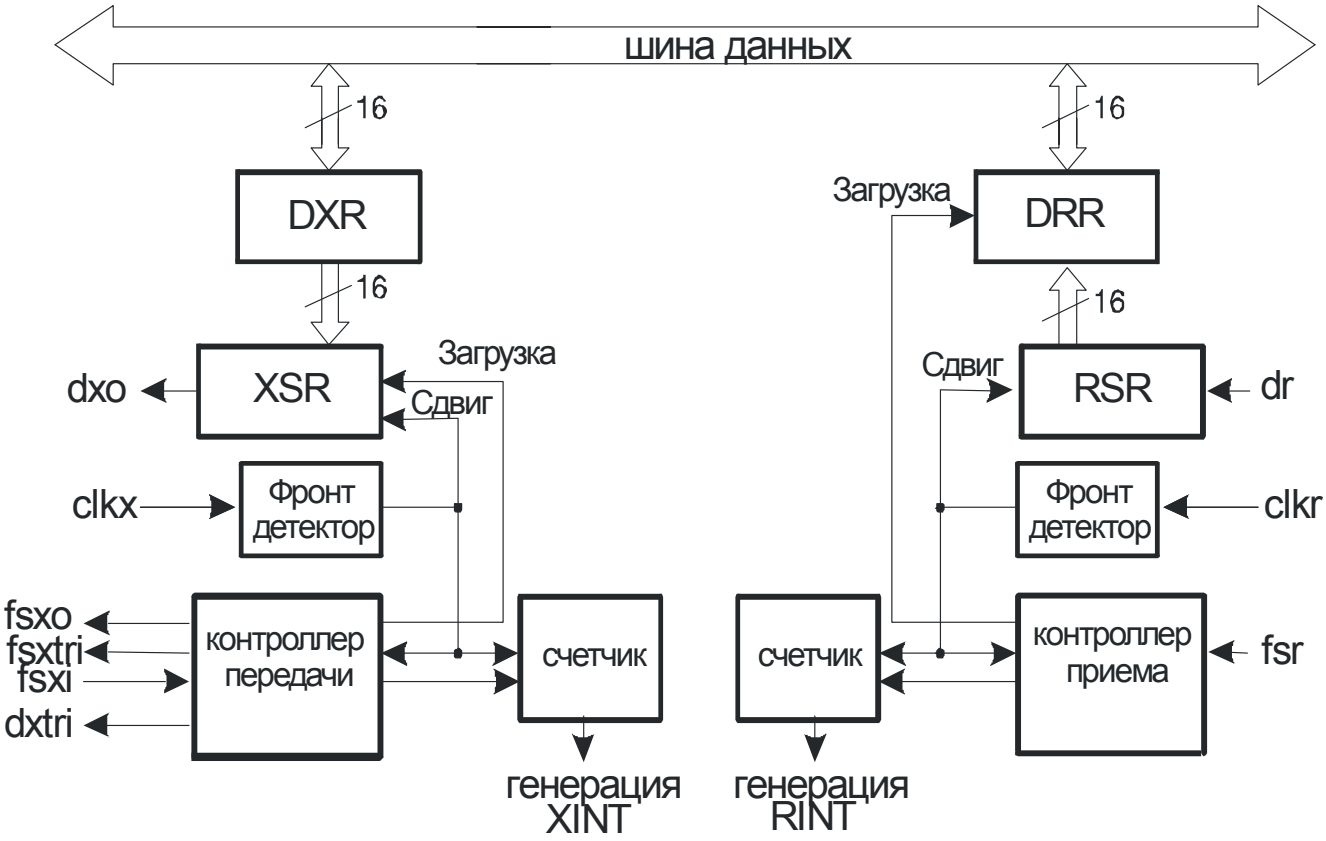


Рисунок 28 - Последовательный порт

Прием и передача осуществляются независимо друг от друга. И передатчик и приемник имеют собственные входы тактовых сигналов, что позволяет работать даже с различными тактовыми частотами. Прием и передача осуществляются с двойной буферизацией.

Прием и передача синхронизируются тактовыми сигналами, подающимися на входы CLKR и CLKX, соответственно. Максимальная частота синхроимпульсов последовательного порта составляет одну четвертую тактовой частоты

1867ВМ7Т, что определяется схемой детектирования изменения уровней сигналов на входах CLKR и CLKX.

Для осуществления приема/передачи используются картированные в памяти регистры:

- DRR - регистр принимаемых данных последовательного порта, адрес 0000h. Содержит принятые данные после каждой операции приема;
- DXR - регистр передаваемых данных последовательного порта, адрес 0001h. Содержимое регистра передается при инициализации передачи.

Разрядность слов приема/передачи (16 или 8 бит) определяется состоянием бита формата последовательного порта FO статусного регистра ST1. При FO=1 разрядность 8 бит, если FO=0 разрядность 16 бит. В 8-битовом режиме, прием дополнительно буферизован, потому что только младшие восемь разрядов регистра DRR, загружаются с полученными данными, а старшие 8 разрядов загружаются 8 битами предыдущего принятого слова. При передаче в 8-битовом режиме, передаются младшие восемь разрядов регистра DXR.

Режим использования кадрового импульса устанавливается значением бита FSM статусного регистра ST1. При FSM=1 кадровый импульс требуется, при FSM=0 нет – непрерывный режим. При наличии кадрового импульса и положительного фронта тактового сигнала последовательного порта инициализируется прием/передача. При FSM=1, после приема/передачи слова, будет ожидаться следующий кадровый импульс для передачи/приема новых данных. Если кадровый импульс подан до окончания приема/передачи слова, то прием/передача данных начинается заново и предыдущие данные теряются, прерывание не вырабатывается. При подаче кадрового импульса во время приема/передачи последнего бита слова данных, инициализируется прием/передача следующего слова данных и вырабатывается прерывание (непрерывный режим при FSM=1). При FSM=0 кадровый импульс требуется для начала приема/передачи только первого слова данных, последующие слова данных принимаются без кадрового импульса, до тех пор, пока значение FSM не установлено в 1.

Передача данных

Данные выдаются на вывод DX из сдвигового регистра XSR на положительном фронте сигнала CLKX. Передача начинается со старших разрядов. При отсутствии данных для передачи, вывод dx переводится в третье состояние. Тактовый сигнал на входе CLKX необходим только во время фактической передачи и может быть остановлен, когда отсутствуют данные для передачи.

При инициализации передачи содержимое регистра DXR копируется в сдвиговый регистр передаваемых данных последовательного порта XSR. Новые данные могут быть записаны в регистр DXR во время передачи из XSR – буферизация режима передачи.

Инициализации передачи обычно производится кадровым импульсом, как описано выше. Значение бита TXM статусного регистра ST1 определяет источник кадрового импульса для передачи данных:

- TXM=0 - вывод FSX конфигурируется как вход, используется внешний кадровый импульс;
- TXM=1 - вывод FSX конфигурируется как выход, генерируется внутренний кадровый импульс. В этом случае передача начинается при записи данных в регистр DXR. Кадровый импульс выдается на выход FSX, данные из регистра XSR сдвигаются на выход DX на каждом положительном фронте тактового сигнала CLKX. После передачи слова генерируется внутреннее прерывание XINT.

Непрерывный режим передачи последовательного порта может быть осуществлен:

- установкой бита FSM = 0 (кадровые импульсы не требуются);
- при FSM=1, подачей кадрового импульса во время передачи последнего бита слова данных;
- при FSM = 1 и TXM = 1, записью новых данных в регистр DXR во время передачи.

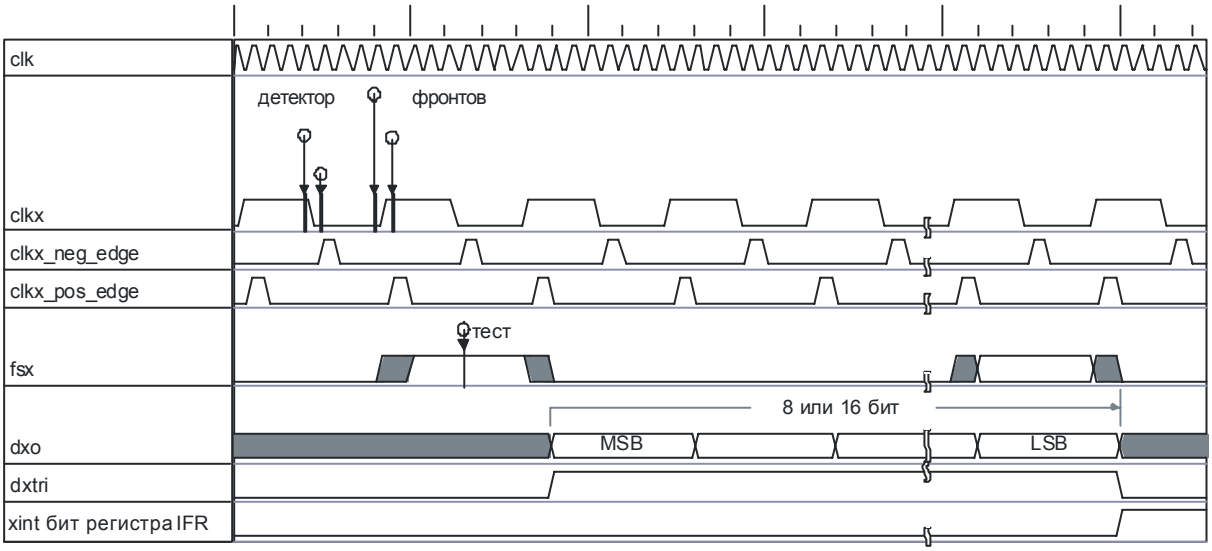


Рисунок 29 - Временная диаграмма передачи данных

Прием данных

Прием данных, как и передача, инициализируется кадровым импульсом, за исключением непрерывного режима, когда FSM=0. Кадровые импульсы для приема данных подаются на вход FSR. После обнаружения кадрового импульса, данные поступающие на вход DR сдвигаются в сдвиговый регистр принимаемых данных последовательного порта RSR на отрицательном фронте сигнала CLKR. Прием начинается со старших разрядов. Тактовый сигнал на входе CLKR необходим только во время фактической передачи и может быть остановлен, когда отсутствуют данные для передачи.

После приема последнего бита, принятое слово данных из регистра RSR копируется в регистр DRR, вырабатывается внутреннее прерывание RINT. Данные в регистре DRR доступны до окончания приема следующего слова – буферизация режима приема.

Непрерывный режим приема последовательного порта может быть осуществлен:

- установкой бита FSM = 0 (кадровые импульсы не требуются);
- при FSM=1, подачей кадрового импульса во время приема последнего бита слова данных.

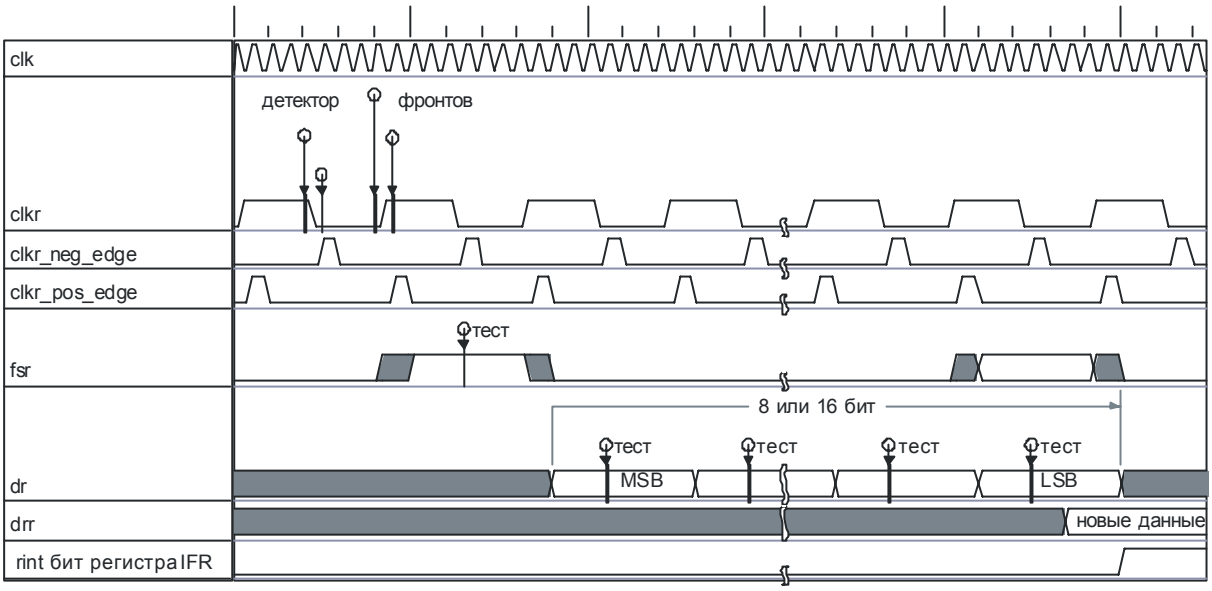


Рисунок 30 - Временная диаграмма приема данных

7 Прерывания

ИС 1867ВМ7Т поддерживает шесть маскируемых, одно немаскируемое программное прерывание и аппаратный сброс (см. таблицу 13):

- внешние маскируемые прерывания INT0#, INT1# и INT2#;
- внутренние маскируемые прерывания:
  TINT – прерывание таймера;
  XINT – завершение передачи слова данных последовательным портом;
  RINT – завершение приема слова данных последовательным портом;
- прерывание, инициализируемое выполнением инструкции TRAP (не имеет приоритета, и фактически работает как инструкция CALL);
- аппаратный сброс по входу RS# (рассматривается, как прерывание с самым высоким приоритетом).

Таблица 13 - Приоритет и вектора прерываний 1867ВМ71

Источник	Вектор	Приоритет	Описание
RS#	0000h	0 (самый высокий)	Аппаратный сброс
INT0#	0002h	1	Внешнее прерывание 0
INT1#	0004h	2	Внешнее прерывание 1
INT2#	0006h	3	Внешнее прерывание 2
TINT	0018h	4	Прерывание таймера
RINT	001Ah	5	Прерывание приемника последовательного порта
XINT	001Ch	6 (самый низкий)	Прерывание передатчика последовательного порта
TRAP	001Eh	—	Программное прерывание

Запрос на прерывание запоминается в регистре флагов прерываний IFR - соответствующий прерыванию бит устанавливается в 1 и остается установленным до момента подтверждения прерывания. Если разрешена обработка прерываний (INTM=0) и прерывание незамаскировано в регистре IMR, вырабатывается внутренний сигнал «irq» для блока управления, значение PC будет сохранено в стеке, в PC и PFC загружается значение вектора прерывания. Происходит выборка инструкций по адресу вектора прерывания, вырабатывается сигнал подтверждения прерывания IACK#. Подтверждение прерывания автоматически устанавливает бит INTM=1, что запрещает обработку других прерываний.

Подпрограмма обработки прерывания перед возвращением к главной программе должна разрешить обработку прерываний с помощью инструкции EINT.

Регистр маски прерывания

Регистр маски прерывания IMR, картирован в памяти по адресу 0004h. Старшие десять разрядов регистра доступны только для чтения и всегда читаются как 1. Младшие биты регистра IMR используется для маскирования внешних и внутренних прерываний. При записи 0 в разряд регистра IMR, соответствующие прерывание будет замаскировано, и запрос на прерывание будет только фиксироваться в регистре IFR, а обслуживание будет отложено до снятия маски – записи 0 в соответствующий бит регистра IMR. При аппаратном сбросе маскируются все прерывания.

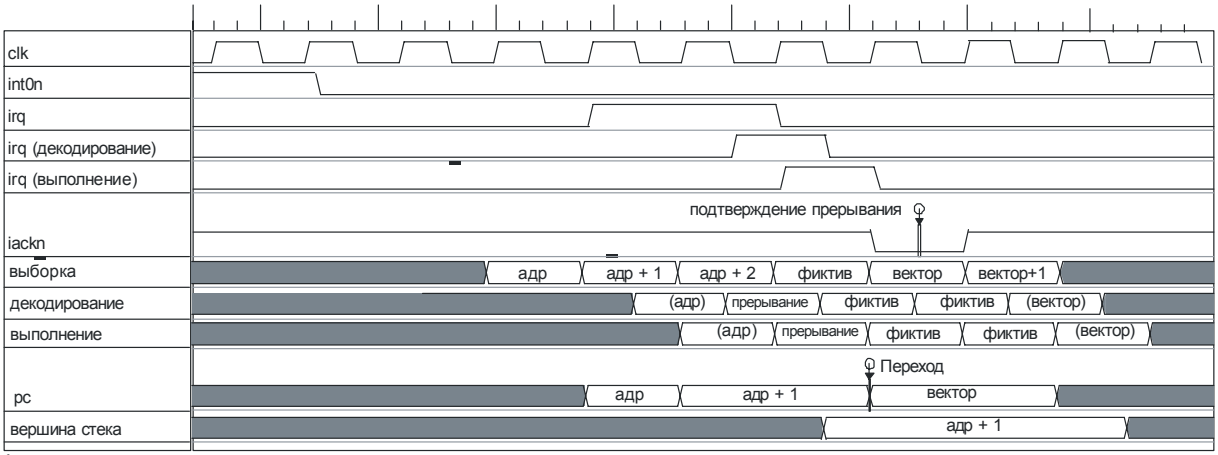
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Не используются										**XINT**	**RINT**	**TINT**	**INT2**	**INT1**	**INT0**

Рисунок 31 - Регистр маски прерываний IMR

Обработка прерываний

Если разрешена обработка прерываний (INTM=0) и прерывание немаскировано, в блок управления поступает внутренний сигнал запроса прерывания «irq»

(см. рисунки 32, 33). Блок управления завершает выборку и декодирование инструкции, сохраняет значение PC в стек, через внутреннюю шину данных значение вектора прерывания из блока управления прерываниями загружается в PC и PFC. Начинается выборка инструкций, вырабатывается подтверждение прерывания – сигнал IACK# (активный уровень низкий), очищается соответствующий бит в регистре IFR, устанавливается INTM=1 для запрета обработки других прерываний.



irq - внутренний сигнал запроса прерывания
адр - адрес памяти программ
(адр) - данные по указанному адресу памяти программ

Рисунок 32 - Временная диаграмма обработки прерывания

Внешние прерывания

Состояние входов внешних прерываний INT0#, INT1# и INT2# опрашивается в каждом цикле тактового сигнала X2/CLKIN. При обнаружении низкого уровня или отрицательного фронта на входах, соответствующий бит регистра IFR устанавливается в 1. Если в момент завершения подпрограммы обработки прерывания сигнал на внешнем входе запроса прерывания активен, то подпрограмма будет вызвана заново.

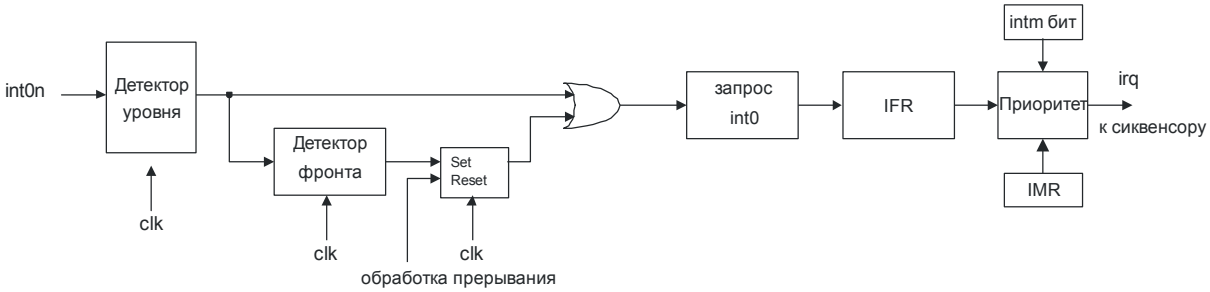


Рисунок 33 - Внешние прерывания

Внутренние прерывания

В отличие от внешних прерываний, детектирование внутренних запросов на прерывание происходит только по фронтам сигналов, а не по уровню.

Внутренние маскируемые прерывания 1867ВМ7Т:

- TINT – прерывание таймера;
  изменение сигнала детектора нулевого значения регистра TIM.
- XINT – завершение передачи слова данных последовательным портом;
  положительный фронт тактового сигнала CLKX и переполнение счетчика переданных бит слова данных.
- RINT – завершение приема слова данных последовательным портом;
  положительный фронт тактового сигнала CLKR и переполнение счетчика принятых бит слова данных.

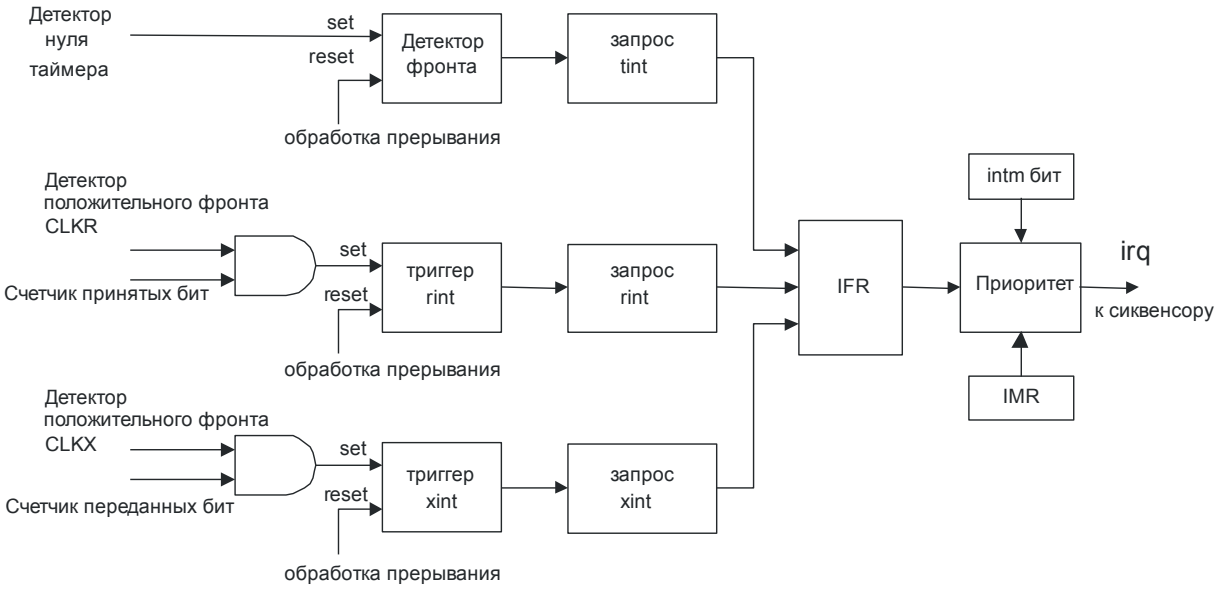


Рисунок 34 - Внутренние прерывания

8 Управление режимом пониженного энергопотребления

Перевод ИС 1867ВМ7Т в режим пониженного энергопотребления осуществляется инструкцией IDLE.

Возможны два варианта режима в зависимости от значения бита PDM:

- PDM=1. Прекращается подача тактовых сигналов на все модули 1867ВМ7Т, за исключением периферии и контроллера прерываний. Выход из режима осуществляется по прерыванию.
- PDM=0. Прекращается подача тактовых сигналов на все модули 1867ВМ7Т. Выход из режима осуществляется аппаратным сбросом.

При выполнении инструкции IDLE бит INTM очищается, разрешая прерывания.

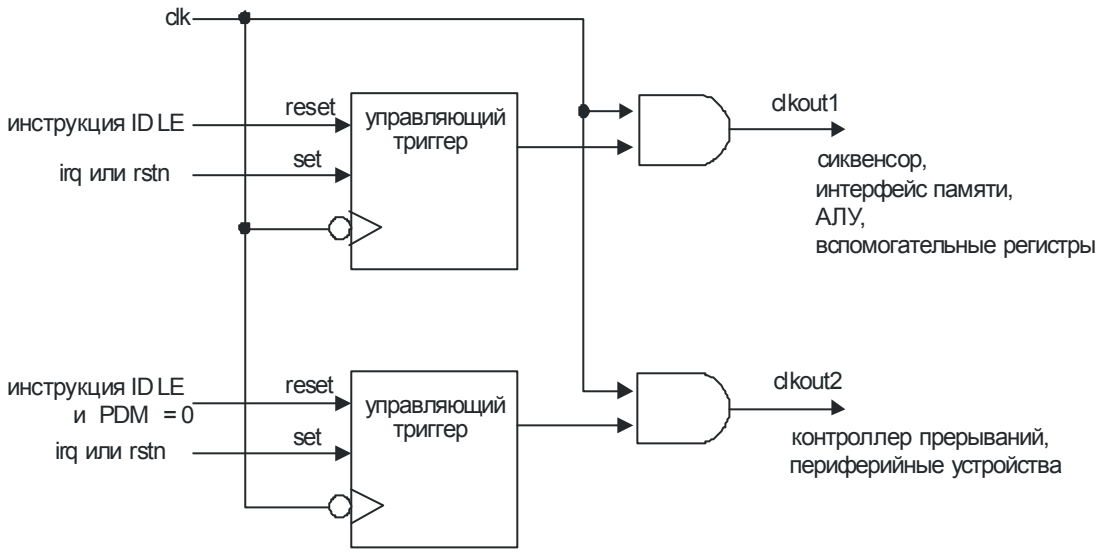


Рисунок 35 - Управление тактовыми сигналами

Заключение

В настоящем руководстве КФДЛ.431299.027 приведено подробное описание архитектуры, функционального построения, системы команд и особенностей применения ИС 1867ВМ7Т, которая представляет собой СБИС однокристального 16-разрядного процессора цифровой обработки сигналов с фик-сированной запятой с тактовой частотой 10 МГц, ОЗУ (544 × 16) бит; центральным арифметико-логическим устройством, обеспечивающим выполнение 32-разрядных арифметических, логических и сдвиговых операций; 16 × 16 разрядов аппаратным умножителем с 32-разрядным произведением; файлом из восьми 16-разрядных вспомогательных регистров для реализации косвенной адресации и временного хранения данных с выделенным 16-разрядным арифметическим устройством, реализующим, в том числе, операции с реверсивным распространением переноса.

Микропроцессор предназначен для применения в перспективных образцах специальной техники и аппаратуры связи, в бортовой аппаратуре обслуживаемых и необслуживаемых космических аппаратов, объектах атомной энергетики и бортовых системах управления, обработки и передачи данных, в телекоммуникационной и другой технике.

Все значения электрических параметров микросхемы приведены в технических условиях АЕЯР.431280.901; значения параметров, приведённые в настоящем техническом описании, являются справочными.

Руководство КФДЛ.431299.027 может служить практическим пособием по применению цифровых сигнальных микропроцессоров для раз-работчиков систем на основе ИС 1867ВМ7Т и программистов.

Приложение А

(обязательное)

Циклограммы выполнения инструкций

Циклограммы выполнения инструкций ИС ПЦОС представлены в таблице А.1.

Принятые условные обозначения:

d – количество состояний ожидания для внешней памяти данных;

p – количество состояний ожидания для внешней программной памяти;

i – количество состояний ожидания для пространства ввода-вывода;

n – количество раз выполнения инструкций в режиме повтора;

DI – инструкции используют внутреннюю память данных;

DE – инструкции используют внешнюю память данных;

PI – программная память во внутреннем ОЗУ (блок B0);

PE – внешняя программная память.

Примечание – Все значения в циклограммах приведены в количестве машинных циклов (тактов синхросигнала), требуемых для выполнения инструкции.

Таблица А.1 – Циклограммы выполнения инструкций

Однократное выполнение												
Класс инструкции	PI/DI		PI/DE		PE/DI		PE/DE		PR/DI		PR/DE	
	IMC=0	IMC=1	EMC=0	EMC=1	IMC=0	IMC=1	EMC=0	EMC=1	IMC=0	IMC=1	EMC=0	EMC=1
1	2		2+d		2+p		2+p+d		2		2+d	
1M	2	3	2+d		2+p	3+p	2+p+d		2	3	2+d	
2	2		2+d		2+p		2+p+d		2		2+d	
3	1	2	1+d	2+d	2+p	3+p	2+p+d	3+p+d	1	2	1+d	2+d
4	1		1		1+p		1+p		1		1	
5	2		2		2+2p		2+2p		2		2	
6	Таблица во внутреннем ОЗУ 4		4+d		4+2p		4+d+2p		4		4+d	
	Таблица во внутреннем ПЗУ 4		4+d		4+2p		4+d+2p		4		4+d	
	Таблица во внешней программной памяти 4+p		4+d+p		4+3p		4+d+3p		4+p		4+d+p	
6M	Таблица во внутреннем ОЗУ 4	5	4+d		4+2p	5+2p	4+d+2p		4	5	4+d	
	Таблица во внутреннем ПЗУ 4	5	4+d		4+2p	5+2p	4+d+2p		4	5	4+d	
	Таблица во внешней программной памяти 4p	5+p	4+d+p		4+3p	5+3p	4+d+3p		4+p	5+p	4+d+p	
7	Выполненный переход Приемник во внутреннем ОЗУ 3		3		3+2p		3+2p		3		3	
	Приемник во внутреннем ПЗУ 3		3		3+2p		3+2p		3		3	
	Приемник во внешней программной памяти 3+p		3+p		3+3p		3+3p		3+p		3+p	
	Невыполненный переход Любой приемник 2		2		2+2p		2+2p		2		2	
8	Приемник во внутреннем ОЗУ 3		3		3+p		3+p		3		3	
	Приемник во внутреннем ПЗУ 3		3		3+p		3+p		3		3	
	Приемник во внешней программной памяти 3+p		3+p		3+2p		3+2p		3+p		3+p	
9	2+i	3+i	2+d+i	3+d+i	2+p+i	3+p+i	3+d+p+i	4+d+p+i	2+i	3+i	2+d+i	3+d+i
10	EMC=0 2+i		2+d+i		2+p+i		3+d+p+i		2+i		2+d+i	
	EMC=1 3+i		3+d+i		3+p+i		4+d+p+i		3+i		3+d+i	

Продолжение таблицы А.1

Однократное выполнение													
Класс инструк-ции	PI/DI		PI/DE		PE/DI		PE/DE		PR/DI		PR/DE		
	IMC=0	IMC=1	EMC=0	EMC=1	IMC=0	IMC=1	EMC=0	EMC=1	IMC=0	IMC=1	EMC=0	EMC=1	
11	Таблица во внутреннем ОЗУ												
	4	5	4+d	5+d	4+p	5+p	4+d+p	5+d+p	4	5	4+d	5+d	
	Таблица во внутреннем ПЗУ												
	4	5	4+d	5+d	4+p	5+p	4+d+p	5+d+p	4	5	4+d	5+d	
	Таблица во внешней программной памяти												
	4+p	5+p	4+d+p	5+d+p	4+2p	5+2p	4+d+2p	5+d+2p	4+p	5+p	4+d+p	5+d+p	
12	Таблица во внутреннем ОЗУ												
	IMC=0												
	4		4+d		4+p		4+d+p		4		4+d		
	IMC=1												
	5		d=0: 5		5+p		d=0 : 5+p		5		d=0 : 5		
			d≠0: 4+d				d≠0 : 4+d+p				d≠0 : 4+d		
	Таблица во внутреннем ПЗУ												
	Не применимо												
	Таблица во внешней программной памяти												
	4+p		4+d+p	5+d+p	4+2p		4+d+2p	5+d+2p	4+p		4+d+p	5+d+p	
13	Источник во внутренней памяти данных												
	5	6	5+d	6+d	4+2p	5+2p	4+d+2p	5+d+2p	4	5	4+d	5+d	
	Источник во внешней памяти данных												
	5+d	6+d	5+2d	6+2d	4+d+2p	5+d+2p	5+2d+2p	6+2d+2p	4+d	5+d	5+2d	6+2d	
14	Таблица во внутреннем ОЗУ												
	5	6	5+d	6+d	4+2p	5+2p	4+d+2p	5+d+2p	4	5	4+d	5+d	
	Таблица во внутреннем ПЗУ												
	5	6	5+d	6+d	4+2p	5+2p	4+d+2p	5+d+2p	4	5	4+d	5+d	
	Таблица во внешней программной памяти												
	5+p	6+p	5+d+p	6+d+p	4+p+2p	5+p+2p	5+d+p+2p	6+d+p+2p	4+p	5+p	5+d+p	6+d+p	
15	Вектор прерывания во внутреннем ПЗУ												
	3		3		3		3		3		3		
	Вектор прерывания во внешней программной памяти												
	3+2p		3+2p		3+2p		3+2p		3+2p		3+2p		

Продолжение таблицы А.1

Выполнение в режиме повтора												
Класс инструк-ции	PI/DI		PI/DE		PE/DI		PE/DE		PR/DI		PR/DE	
	IMC=0	IMC=1	EMC=0	EMC=1	IMC=0	IMC=1	EMC=0	EMC=1	IMC=0	IMC=1	EMC=0	EMC=1
1	1+n		1+n+nd		1+n+p		1+n+p+nd		1+n		1+n+nd	
1M	1+n	3n	1+n+nd		1+n+p	3n+p	1+n+p+nd		1+n	3n	1+n+nd	
2	2n		2n+nd		2n+p		2n+p+nd		2n		2n+nd	
3	n	2n	n+nd	2n+d	n+p	2n+p	1+n+p+nd	1+2n+p+nd	n	2n	n+nd	2n+nd
4	n		n		n+p		n+p		n		n	
5	Не повторяемые											
6	Таблица во внутреннем ОЗУ											
	2+n		2+2n+nd		2+n+2p		2+2n+nd+2p		2+n		2+2n+nd	
	Таблица во внутреннем ПЗУ											
	2+n		2+2n+nd		2+n+2p		2+2n+nd+2p		2+n		2+2n+nd	
	Таблица во внешней программной памяти											
	2+n+np		2+2n+nd+np		2+n+np+2p		2+2n+nd+np+2p		2+n+np		2+2n+nd+np	
6M	Таблица во внутреннем ОЗУ											
	2+2n	2+3n	2+2n+nd		2+2n+2p	2+3n+2p	2+2n+nd+2p		2+2n	2+3n	2+2n+nd	
	Таблица во внутреннем ПЗУ											
	2+2n	2+3n	2+2n+nd		2+2n+2p	2+3n+2p	2+2n+nd+2p		2+2n	2+3n	2+2n+nd	
	Таблица во внешней программной памяти											
	2+2n+np	2+3n+np	2+2n+nd+np		2+2n+np+2p	2+3n+np+2p	2+2n+nd+np+2p		2+2n+np	2+3n+np	2+2n+nd+np	
7	Не повторяемые											
8	Не повторяемые											
9	1+n+ni	i=0 : 1+2n i≠0 : 2+n+ni	2n+nd+ni	3n+nd+ni	1+n+p+ni	i=0 : 1+2n+p i≠0 : 2+n+p+ni	1+2n+nd+p+ni	1+3n+nd+p+ni	1+n+ni	i=0 : 1+2n i≠0 : 2+n+ni	2n+nd+ni	3n+nd+ni
10	EMC=0											
	1+n+ni		2n+nd+ni		1+n+p+ni		1+2n+nd+p+ni		1+n+ni		2n+nd+ni	
	EMC=1											
	1+2n+ni		3n+nd+ni		1+2n+p+ni		1+3n+nd+p+ni		1+2n+ni		3n+nd+ni	
11	Таблица во внутреннем ОЗУ											
	3+n	3+2n	2+n+nd	2+2n+nd	3+n+p	3+2n+p	3+n+nd+p	3+2n+nd+p	3+n	3+2n	3+n+nd	3+2n+nd
	Таблица во внутреннем ПЗУ											
	3+n	3+2n	2+n+nd	2+2n+nd	3+n+p	3+2n+p	3+n+nd+p	3+2n+nd+p	3+n	3+2n	3+n+nd	3+2n+nd
	Таблица во внешней программной памяти											
	3+n+np	p=0 : 3+2n p≠0 : 4+n+np	2+2n+nd+np	2+3n+nd+np	3+n+np+p	p=0 : 3+2n p≠0 : 4+n+np+p	2+2n+nd+np+p	2+3n+nd+np+p	3+n+np	p=0 : 3+2n p≠0 : 4+n+np	2+2n+nd+np	2+3n+nd+np

Окончание таблицы А.1

<table>
<tr><td colspan="13">Выполнение в режиме повтора</td></tr>
<tr><td rowspan="2">Класс инструк-ции</td><td colspan="2">PI/DI</td><td colspan="2">PI/DE</td><td colspan="2">PE/DI</td><td colspan="2">PE/DE</td><td colspan="2">PR/DI</td><td colspan="2">PR/DE</td></tr>
<tr><td>IMC=0</td><td>IMC=1</td><td>EMC=0</td><td>EMC=1</td><td>IMC=0</td><td>IMC=1</td><td>EMC=0</td><td>EMC=1</td><td>IMC=0</td><td>IMC=1</td><td>EMC=0</td><td>EMC=1</td></tr>
<tr><td>12</td><td colspan="12">Таблица во внутреннем ОЗУ</td></tr>
<tr><td rowspan="6"></td><td colspan="12">IMC=0
3+n | 3+n+nd | 3+n+p | 3+n+nd+p | 3+n | 3+n+nd
IMC=1
3+2n | d=0 : 3+2n
d≠0 : 3+n+nd | 3+2n+p | d=0 : 3+2n+p
d≠0 : 3+n+nd+p | 3+2n | d=0 : 3+2n
d≠0 : 3+n+nd</td></tr>
<tr><td colspan="12">Таблица во внутреннем ПЗУ</td></tr>
<tr><td colspan="12">Не применимо</td></tr>
<tr><td colspan="12">Таблица во внешней программной памяти</td></tr>
<tr><td colspan="2">3+n+np</td><td>2+2n+nd+np</td><td>2+3n+nd+np</td><td colspan="2">3+n+np+p</td><td>2+2n+nd+np+p</td><td>2+3n+nd+np+p</td><td colspan="2">3+n+np</td><td>2+2n+nd+np</td><td>2+3n+nd+np</td></tr>
<tr><td>13</td><td colspan="12">Источник во внутренней памяти данных</td></tr>
<tr><td colspan="12">4+n | 4+2n | 4+n+nd | 4+2n+nd | 3+n+2p | 3+2n+2p | 3+n+nd+2p | 3+2n+nd+2p | 3+n | 3+2n | 3+n+nd | 3+2n+nd
Источник во внешней памяти данных
4+n+nd | d=0 : 4+2n
d≠0 : 5+n+nd | 3+2n+2nd | 3+3n+2nd | 3+n+nd+2p | d=0 : 3+2n
d≠0 : 4+n+nd+2p | 3+2n+2nd+2p | 3+3n+2nd+2p | 3+n+nd | d=0 : 3+2n
d≠0 : 4+n+nd | 3+2n+2nd | 3+3n+2nd</td></tr>
<tr><td>14</td><td colspan="12">Таблица во внутреннем ОЗУ
4+n | 4+2n | 4+n+nd | 4+2n+nd | 3+n+2p | 3+2n+2p | 3+n+nd+2p | 3+2n+nd+2p | 3+n | 3+2n | 3+n+nd | 3+2n+nd
Таблица во внутреннем ПЗУ
4+n | 4+2n | 4+n+nd | 4+2n+nd | 3+n+2p | 3+2n+2p | 3+n+nd+2p | 3+2n+nd+2p | 3+n | 3+2n | 3+n+nd | 3+2n+nd
Таблица во внешней программной памяти
4+n+np | p=0 : 4+2n
p≠0 : 5+n+np | 3+2n+nd+np | 3+3n+nd+np | 3+n+np.+2p | p=0 : 3+2n
p≠0 : 4+n+np+2p | 3+2n+nd+np+2p | 3+3n+nd+np+2p | 3+n+np | p=0 : 3+2n
p≠0 : 4+n+np | 3+2n+nd+np | 3+3n+nd+np</td></tr>
<tr><td>15</td><td colspan="12">Неповторяемая</td></tr>
</table>

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	-	67			09.12.13
1	1	-	-	-	-			24.12.13
2	-	5, 6	-	-	-			22.05.14