

МИКРОСХЕМЫ ИНТЕГРАЛЬНЫЕ
1887ВЕ1У

Техническое описание
КФДЛ.431295.025ТО

Содержание

Введение.....	5
1 Назначение и область применения.....	5
2 Краткое техническое описание ИМС 1887ВЕ1У.....	5
2.1 Функциональные возможности:	6
2.2 Особенности архитектуры:	7
2.3 Система команд.....	11
2.4 Электрические параметры микросхем	11
3 Функциональное описание ИМС 1887ВЕ1У.....	13
3.1 Ядро микроконтроллера	15
3.1.1 Анализ архитектуры	15
3.1.2 АЛУ – Арифметико-логическое устройство	17
3.1.3 Регистр файлов общего назначения	19
3.1.4 Указатель стека	20
3.1.5 Сброс и обработка прерываний	22
3.2 Устройства памяти микроконтроллера 1887ВЕ1У	24
3.2.1 Внутрисистемная перепрограммируемая флэш-память программ	24
3.2.2 Память данных ЭСППЗУ	26
3.2.3 Предотвращение ошибок в ЭСППЗУ	31
3.2.4 Память ввода-вывода (I/O Memory)	32
3.3 Система синхронизации	33
3.3.1 Тактовый генератор с внешним резонатором	35
3.3.2 Кварцевый генератор низкой частоты	37
3.3.3 Внешний RC генератор	37
3.3.4 Калируемый внутренний RC генератор	39
3.3.5 Внешний тактовый генератор	40
3.3.6 Генератор таймера/счетчика	41
3.4 Энергосберегающие режимы	42
3.4.1 Режим холостого хода	43
3.4.2 Режим снижения шума АЦП	43
3.4.3 Режим микропотребления	43
3.4.4 Режим энергосбережения	44
3.4.5 Дежурный режим	44
3.4.6 Длительный дежурный режим	44
3.5 Сброс и управление системой	47
3.5.1 Сброс по включению питания	49
3.5.2 Внешний сброс	49
3.5.3 Детектирование отключения питания	50
3.5.4 Сброс сторожевого таймера	51
3.6 Сторожевой таймер	53
3.7 Прерывания	57
3.7.1 Регистр управления общим прерыванием GICR	60
3.8 Порты ввода-вывода	62
3.8.1 Порты в качестве общих цифровых портов ввода-вывода	63
3.8.2 Конфигурация вывода	64
3.8.3 Считывание значения вывода	64
3.8.4 Режим разрешения цифрового входа и спящий режим	67
3.8.5 Неподсоединеные выводы	67
3.8.6 Альтернативные функции портов	67
3.8.7 Альтернативные функции порта А	70
3.8.8 Альтернативные функции порта В	71
3.8.9 Альтернативные функции порта С	73
3.8.10 Альтернативные функции порта D	75

3.9 Внешние прерывания	79
3.9.1 Регистр управления МПУ – MCUCR	79
3.9.2 Регистр управления общим прерыванием – GICR	81
3.9.3 Регистр флага общего прерывания – GIFR	81
3.10 8-разрядный таймер/счетчик 0 с ШИМ (PWM)	83
3.10.1 Обзор	83
3.10.2 Регистры	84
3.10.3 Определения	84
3.10.4 Источники тактового сигнала таймера/счетчика	84
3.10.5 Модуль счетчика	84
3.10.6 Модуль сравнения выхода	85
3.10.7 Принудительное сравнение по выходу	86
3.10.8 Блокировка совпадения/сравнения с помощью записи TCNT0	87
3.10.9 Использование блока сравнения выхода	87
3.10.10 Блок сравнения совпадений на выходе	87
3.10.11 Режим сравнения выхода и генерация формы сигнала	88
3.10.12 Режимы работы	89
3.10.13 Описание 8-разрядных регистров таймера/счетчика	95
3.11 Устройства предварительного деления частоты таймера/счетчика 0 и таймера/счетчика 1	100
3.11.1 Источник внутреннего синхроимпульса	100
3.11.2 Установка устройства предварительного деления частоты	100
3.11.3 Источник внешней синхронизации	100
3.12 16-разрядный таймер/счетчик 1	103
3.13 8-битный Таймер/Счетчик 2 с ШИМ и асинхронными операциями	134
3.14 Последовательный периферийный интерфейс (SPI)	153
3.14.1 Функционирование модуля	153
3.14.2 Режимы передачи данных	156
3.14.3 Использование вывода SS#	157
3.15 Последовательный синхронно-асинхронный приемопередатчик (USART)	159
3.15.1 Краткий обзор	159
3.15.2 Работа с удвоением скорости связи (U2X)	161
3.15.3 Внешняя синхронизация	162
3.15.4 Режим синхронной связи	162
3.15.5 Передача данных - Передатчик УСАПП	165
3.15.6 Флаги и прерывания передатчика	166
3.15.7 Генератор паритета	167
3.15.8 Отключение передатчика	167
3.15.9 Прием данных - Приемник УСАПП	167
3.15.10 Прием посылок с 5...8 битами данных	167
3.15.11 Флаг и прерывание по завершению приема	168
3.15.12 Флаги ошибок приемника	168
3.15.13 Устройство проверки паритета	169
3.15.14 Асинхронный прием данных	170
3.15.15 Многопроцессорный режим связи	173
3.15.16 Использование MPCM	175
3.16 Двухпроводной последовательный интерфейс TWI	184
3.16.1 Определение шины TWI	184
3.16.2 Терминология TWI	185
3.16.3 Формат посылки и передаваемых данных	186
3.16.4 Формат адресного пакета	187
3.16.5 Формат пакета данных	188
3.16.6 Сочетание пакетов адреса и данных во время сеанса связи	188

3.16.7 Системы многомастерных шин, арбитраж и синхронизация.....	189
3.16.8 Обзор модуля TWI	191
3.16.9 Описание регистров TWI	193
3.16.10 Рекомендации по использованию регистра TWI.....	196
3.17 Аналоговый компаратор	217
3.18 Аналого-цифровой преобразователь	220
3.18.1 Принцип действия АЦП.....	221
3.18.2 Предделитель АЦП и временная диаграмма преобразования	223
3.18.3 Каналы дифференциального усиления	225
3.18.4 Источник опорного напряжения АЦП.....	226
3.18.5 Подавитель шумов АЦП	226
3.18.6 Схема аналогового входа	227
3.18.7 Методы компенсации смещения	230
3.19 Поддержка загрузчика – самопрограммирование	233
3.19.1 Регистр управления SPMCR	234
3.19.2 Изменение памяти программ	236
3.19.3 Изменение ячеек защиты загрузчика	237
3.19.4 Чтение конфигурационных ячеек и ячеек защиты	237
3.20 Программирование памяти	238
3.20.1 Защита кода и данных	238
3.20.2 Конфигурационные ячейки.....	240
3.20.3 Идентификатор	241
3.20.4 Калибровочная ячейка.....	241
3.20.5 Параллельное программирование	241
3.20.6 Программирование по последовательному каналу	253
4 Адресация памяти	257
4.1 Прямая адресация	257
4.1.1 Прямая адресация одного РОН	257
4.1.2 Прямая адресация двух РОН	257
4.1.3 Прямая адресация РВВ.....	257
4.1.4 Прямая адресация ОЗУ	257
4.2 Косвенная адресация	258
4.2.1 Простая косвенная адресация	258
4.2.2 Относительная косвенная адресация	258
4.2.3 Косвенная адресация с преддекрементом	258
4.2.4 Косвенная адресация с постинкрементом	258
5 Введение в систему команд ИМС 1887ВЕ1У	259
5.1 Команды логических операций	259
5.2 Команды арифметических операций и команды сдвига.....	259
5.3 Команды операций с битами	259
5.4 Команды пересылки данных.....	260
5.5 Команды передачи управления	260
5.6 Команды управления системой	261
6 Отладочные средства.....	262
7 Заключение	266
Приложение А (обязательное) Описание системы команд	267
Лист регистрации изменений	Ошибка! Закладка не определена.

Введение

В настоящем техническом описании КФДЛ.431295.025ТО приведено описание архитектуры, функционального построения, системы команд и особенностей применения ИМС 1887ВЕ1У, которая представляет собой 8-разрядную микро-ЭВМ с AVR RISC архитектурой, тактовой частотой 8 МГц, содержащий 8К байт Flash ЗУ программ, ЭСППЗУ (512×8) бит, ОЗУ (512×8) бит, 8-канальный 10-разрядный АЦП, последовательный периферийный интерфейс SPI, двухпроводной последовательный интерфейс TWI, последовательный порт USART. Разработанные микросхемы будут служить основой для создания перспективных систем управления.

В настоящее время одним из основных факторов обеспечения конкурентоспособности отечественной радиоэлектронной аппаратуры и ее живучести является применение при ее разработке и производстве импортонезависимой элементной базы. Особенno это касается аппаратуры для вооружения, военной и специальной техники. Импортозамещение электронных компонентов наиболее эффективно в случае использования полных функциональных аналогов изделий микроэлектроники.

Настоящее ТО может служить практическим руководством по применению микроконтроллеров для разработчиков систем на основе ИМС 1887ВЕ1У.

1 Назначение и область применения

Микросхемы предназначены для применения в системах встроенного управления комплексами радиосвязи специального назначения.

В области промышленного производства микросхема 1887ВЕ1У может быть использована для управления робототехническими комплексами, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, оргтехнике, вычислительной технике, телекоммуникационной технике и т. п. Особенno перспективно применение микросхем 1887ВЕ1У в портативной носимой аппаратуре и приборах, имеющих жесткие ограничения по соотношению быстродействие/потребляемая мощность/стоимость.

2 Краткое техническое описание ИМС 1887ВЕ1У

Схема является КМОП 8-битным микроконтроллером, построенным на расширенной AVR RISC архитектуре. Используя команды, исполняемые за один машинный такт, контроллер достигает производительности в 1 MIPS на рабочей частоте 1 МГц, что позволяет разработчику эффективно оптимизировать потребление энергии за счёт выбора оптимальной производительности.

AVR ядро сочетает расширенный набор команд с 32 рабочими регистрами общего назначения. Все 32 регистра соединены с АЛУ, что обеспечивает доступ к двум независимым регистрам на время исполнения команды за один машинный такт. Благодаря выбранной архитектуре достигнута наивысшая скорость кода и, соответственно, высокая производительность в 10 раз превосходящая скорость соответствующего CISC микроконтроллера. Микроконтроллер содержит: 8К байт внутрисистемной программируемой флэш-памяти программ с возможностью чтения в процессе записи, 512 байтов ЭСППЗУ, 512 байтов СОЗУ, 32 входа-выхода общего назначения, 32 рабочих регистра, три гибких таймера/счётчика с режимом сравнения, внешние и внутренние прерывания, последовательный программируемый USART, 8-канальный 10-битный АЦП, программируемый сторожевой таймер с внутренним генератором, последовательный SPI порт и шесть, выбираемых программно, режимов сбережения энергии.

В ждущем режиме ЦПУ не функционирует, в то время как функционируют

СОЗУ, таймеры/счётчики, SPI порт и система прерываний. В микроконтроллере существует специальный режим подавления шума АЦП, при этом в целом в спящем режиме функционируют только АЦП и асинхронный таймер для исключения цифровых шумов в процессе преобразования АЦП. В режиме микропотребления процессор сохраняет содержимое всех регистров, останавливает генератор тактовых сигналов, приостанавливает все другие функции кристалла до прихода внешнего прерывания или поступления внешней команды RESET. В режиме ожидания работает генератор тактовых частот, в то время как остальные блоки находятся в спящем режиме. Благодаря этому переход в нормальный режим работы происходит гораздо быстрее. В расширенном режиме ожидания в рабочем состоянии находятся основной генератор и асинхронный таймер.

Микросхемы выпускаются при использовании технологии энергонезависимой памяти высокой плотности. Встроенная Flash позволяет перепрограммировать память программ внутрисистемно через последовательный SPI интерфейс стандартным программатором энергонезависимой памяти или встроенной загрузочной программой, работающей в ядре ЦПУ. Загрузочная программа может использовать любой интерфейс для экспорта рабочей программы во флэш-память.

Комбинация расширенной 8-битной RISC архитектуры ЦПУ и внутрисистемной флэш-памяти обеспечивают микроконтроллеру высокую гибкость и экономическую эффективность во встраиваемых системах управления.

2.1 Функциональные возможности:

тактовая частота, МГц	8
объём встроенного ОЗУ, бит	512 × 8
память программ (Flash типа), бит	8K × 8
ЭСППЗУ, бит	512 × 8
количество источников прерываний	20
количество параллельных 8-разрядных портов	4
число каналов аналого-цифрового преобразователя	8
число разрядов аналого-цифрового преобразователя	10
16-разрядных таймеров	1
8-разрядных таймеров	2
число последовательных портов (USART, SPI)	2
аналоговый компаратор	1
сторожевой таймер (WDT)	1
число режимов пониженного потребления мощности	6

Микросхемы разработаны в металлокерамическом корпусе 5133.48-3.

Номинальное значение напряжения питания микросхем плюс 5 В. Допустимое отклонение напряжения питания от номинального ±10 %. Амплитуда пульсаций напряжения питания не более 50 мВ.

Напряжение источника опорного напряжения от 4,0 до 5,5 В. Допустимое отклонение напряжения питания от крайних значений минус 1 % для напряжения 4,0 В и плюс 1 % для напряжения 5,5 В.

2.2 Особенности архитектуры:

- быстродействующая архитектура типа "регистр-регистр";
- регистровое ОЗУ емкостью до 512 байт;
- последовательный периферийный интерфейс (SPI);
- последовательный синхронно-асинхронный приемопередатчик (USART);
- двухпроводной последовательный интерфейс (TWI);
- 16-разрядный таймер/счетчик;
- два 8-разрядных таймера/счетчика;
- 8-разрядный сторожевой таймер;
- до 8 каналов аналого-цифрового преобразователя с разрешением в 10 бит;
- аналоговый компаратор;
- 4-канальный ШИМ;
- четыре 8-разрядных порта ввода-вывода;
- режимы холостого хода (IDLE) и пониженного энергопотребления (POWERDOWN).

Структурная схема ИМС приведена на рисунке 2.1.

Условное графическое обозначение микросхемы приведено на рисунке 2.2.

Функциональное назначение выводов микросхемы и их альтернативные функции представлены в таблице 2.1.

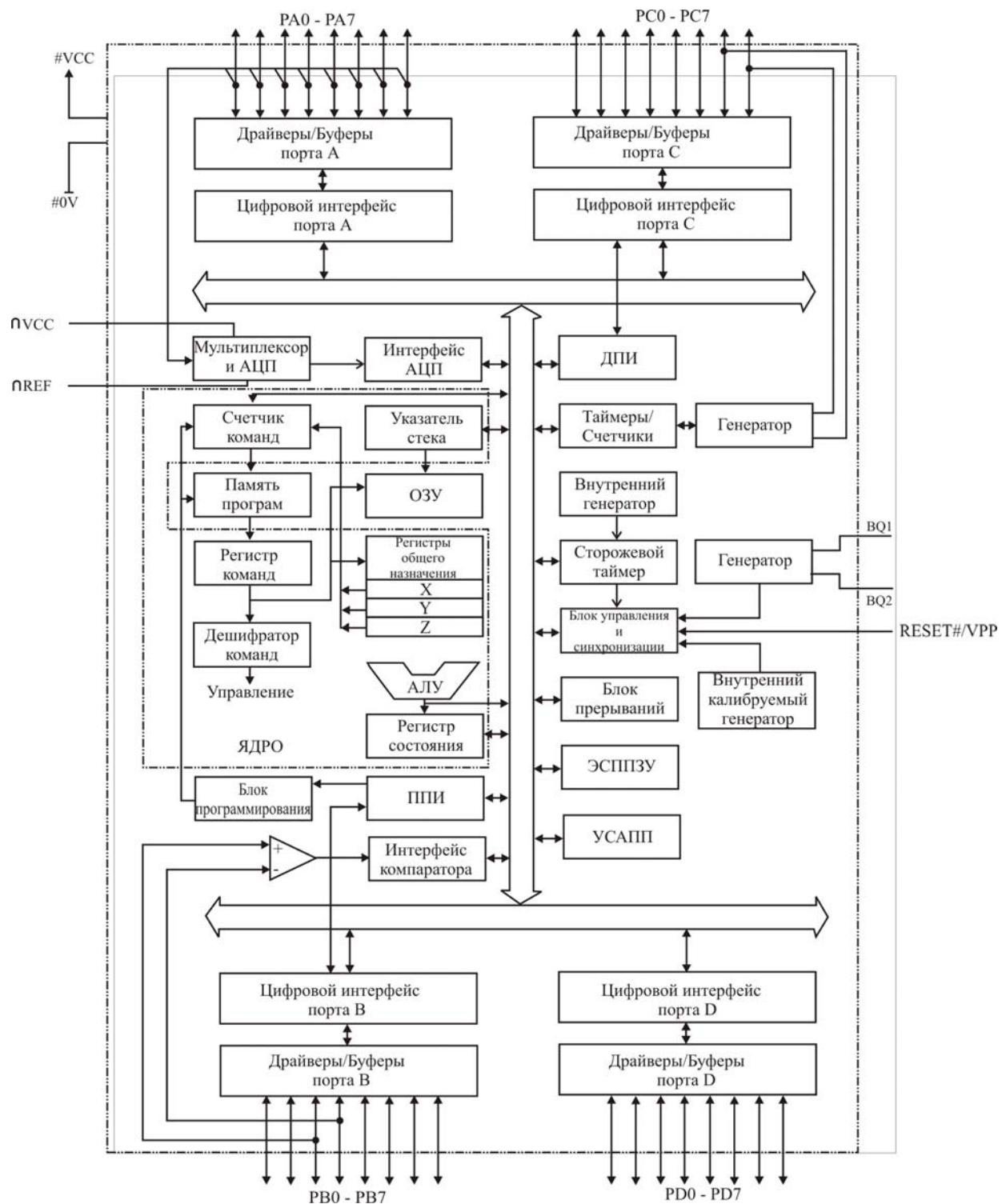


Рисунок 2.1 – Структурная схема ИМС 1887ВЕ1У

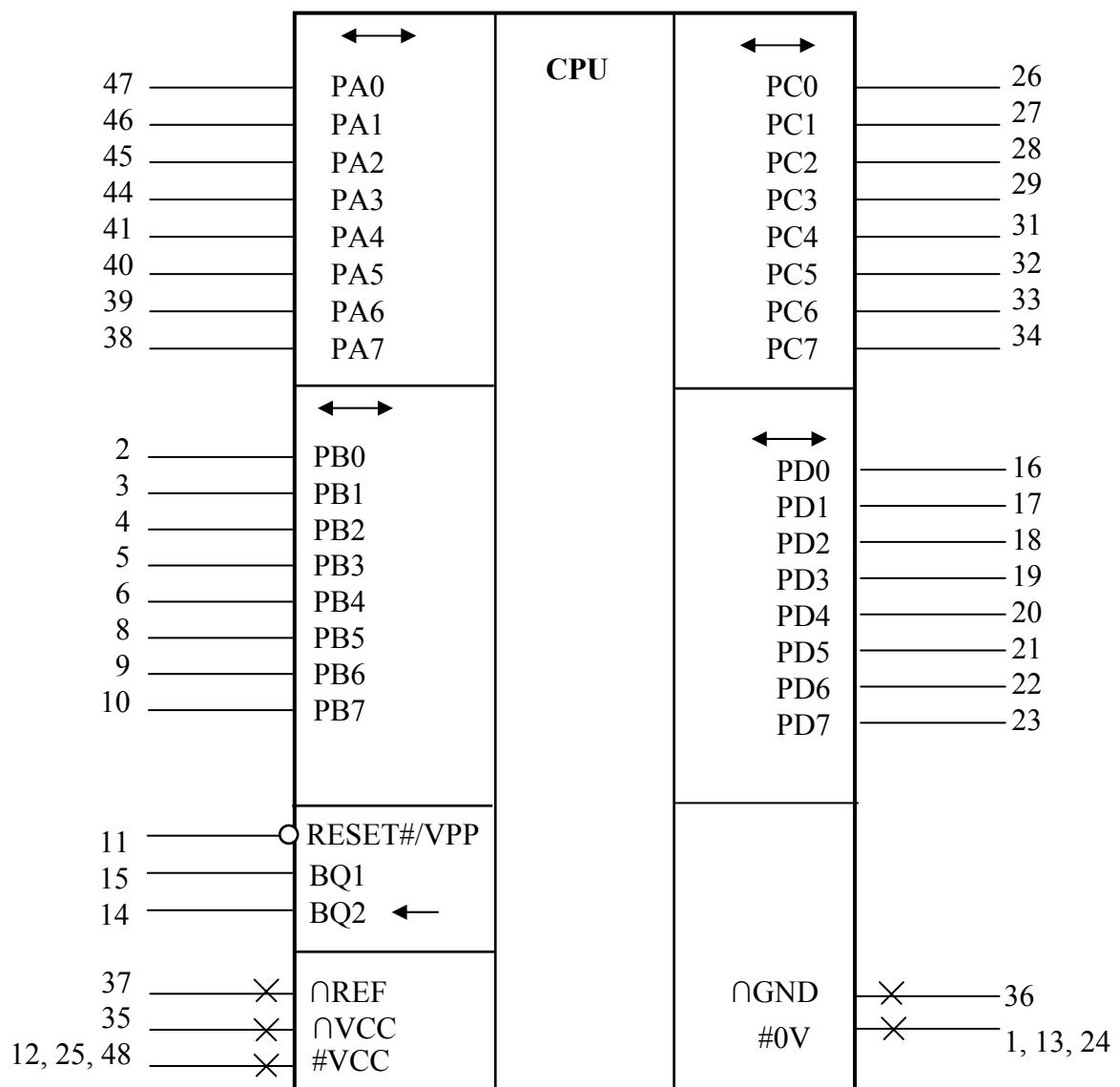


Рисунок 2.2 – Условно-графическое обозначение ИМС 1887ВЕ1У

Таблица 2.1 – Назначение выводов микросхемы

Номер вывода	Обозна-чение вывода	Функциональное назначение вывода	Дополнительная функция вывода	Тип выво-да
47 46 45 44 41 40 39 38	PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7	8-разрядный двунаправленный порт А вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход	вход АЦП, канал 0 вход АЦП, канал 1 вход АЦП, канал 2 вход АЦП, канал 3 вход АЦП, канал 4 вход АЦП, канал 5 вход АЦП, канал 6 вход АЦП, канал 7	I/O
2 3 4 5 6 8 9 10	PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7	8-разрядный двунаправленный порт В вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход	вход Т/C0: T0 вход Т/C1: T1 прямой аналоговый вход компаратора AIN0 инверсный аналоговый вход компаратора AIN1 вход выбора последовательного периферийного интерфейса ППИ: SS# вход/выход данных ППИ: MOSI вход/выход данных ППИ: MISO вход/выход тактового сигнала ППИ: SCK	I/O
26 27 28 29 31 32 33 34	PC0 PC1 PC2 PC3 PC4 PC5 PC6 PC7	8-разрядный двунаправленный порт С вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход	вход/выход тактового сигнала ДПИ: SCL вход/выход данных ДПИ: SDA вывод подключения часовому кварцевого резонатора вывод подключения часовому кварцевого резонатора	I/O
16 17 18 19 20 21 22 23	PD0 PD1 PD2 PD3 PD4 PD5 PD6 PD7	8-разрядный двунаправленный порт D вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход вход/выход	вход последовательных данных приемника RXD выход последовательных данных передатчика TXD вход внешнего прерывания 0: INT0 вход внешнего прерывания 1: INT1 компараторный выход В Т/C1: OC1B компараторный выход А Т/C1: OC1A вход захвата Т/C1: ICP компараторный выход Т/C2: OC2	I/O
37	REF	Вывод опорного напряжения		I
35	VCC	Вывод питания аналоговой части микросхемы		-
36	0V	Общий вывод аналоговой части микросхемы		-
14 15	BQ2 BQ1	Выводы для подключения кварцевого резонатора		O I
11	RESET# / VPP	Вывод сигнала общего сброса / вывод программирования		I
1, 13, 24	#0V	Общий вывод 0 В		-
12, 25, 48	#VCC	Выводы питания цифровой части микросхемы		-
Примечания				
1 Выводы 7, 30, 42, 43 не задействованы.				
2 Условные обозначения: I – вход, O – выход.				

2.3 Система команд

Система команд включает в себя полный набор арифметических, логических команд, а также операций над битами, инструкций управления и перехода с различными способами адресации. Общее число команд – 130 (см. приложение А).

2.4 Электрические параметры микросхем

Электрические параметры микросхем при приемке и поставке приведены в таблице 2.2

Значения предельно-допустимых электрических режимов эксплуатации в диапазоне рабочих температур приведены в таблице 2.3.

Таблица 2.2 - Электрические параметры микросхем при приемке и поставке

Наименование параметра, единица измерения, режим измерения	Бук- венное об- значе- ние	Норма		Темпе- ратура, °C
		не менее	не более	
1	2	3	4	5
1 Выходное напряжение низкого уровня по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, В $I_{OL} = 20 \text{ мА}$, $U_{\#VCC} = U_{\eta VCC} = 5 \text{ В}$	U_{OL}	-	0,6	-60 ± 3 25 ± 10 85 ± 3
2 Выходное напряжение высокого уровня по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, В $I_{OH} = -3 \text{ мА}$, $U_{\#VCC} = U_{\eta VCC} = 5 \text{ В}$	U_{OH}	4,2	-	
3 Токи утечки на входе по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, мкА $0,45 \text{ В} \leq U_I \leq 5,5 \text{ В}$, $U_{\#VCC} = U_{\eta VCC} = 5,5 \text{ В}$	I_{ILL} I_{ILH}	-8	8	25 ± 10 85 ± 3
4 Динамический ток потребления по выводам #VCC, $\cap VCC$ в активном режиме, мА $U_{\#VCC} = U_{\eta VCC} = 5,5 \text{ В}$, $f_{CI} = 4 \text{ МГц}$	I_{OCC1}	-	30	
5 Динамический ток потребления по выводам #VCC, $\cap VCC$ в режиме пониженного потребления, мА $U_{\#VCC} = U_{\eta VCC} = 5,5 \text{ В}$, $f_{CI} = 4 \text{ МГц}$	I_{OCC2}	-	8	-60 ± 3 25 ± 10 85 ± 3
6 Ток потребления по выводам #VCC, $\cap VCC$ в режиме хранения, мкА $U_{\#VCC} = U_{\eta VCC} = 5,5 \text{ В}$	I_{CCS}	-	200	
7 Напряжение смещения компаратора, мВ $U_I = 2,5 \text{ В}$, $U_{\#VCC} = U_{\eta VCC} = 5 \text{ В}$	U_{IO}	-	40	
8 Функциональный контроль $U_{\#VCC} = U_{\eta VCC} = (4,5 \div 5,5) \text{ В}$, $f_{CI} = 8 \text{ МГц}$	ФК	-	-	
9 Время переключения на выходе PB0, нс	время нарастания	t_r	14	
	время спада	t_f		

Таблица 2.3 – Значения предельно-допустимых электрических режимов эксплуатации в диапазоне рабочих температур

Наименование параметра режима, единица измерения	Буквенное обозначение	Предельно - допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
1 Напряжение питания цифровой части ИМС, В	U _{#VCC}	4,5	5,5	-0,3	7
2 Напряжение питания аналоговой части ИМС, В	U _{nvcc} *	4,5	5,5	-0,3	7
3 Входное напряжение низкого уровня, В	U _{IL}	-0,5	0,2U _{#VCC}	-0,6	-
4 Входное напряжение высокого уровня по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, В	U _{IHI}	0,6U _{#VCC}	U _{#VCC} +0,5	-	U _{#VCC} +0,6
5 Входное напряжение высокого уровня по выводу BQ1, В	U _{IH2}	0,8U _{#VCC}	U _{#VCC} +0,5	-	U _{#VCC} +0,6
6 Входное напряжение высокого уровня по выводу RESET#/VPP, В	U _{IHZ}	0,9U _{#VCC}	U _{#VCC} +0,5	-	U _{#VCC} +0,6
7 Напряжение программирования на выводе RESET#/VPP, В	U _{PP}	11,5	12,5	-	13
8 Выходной ток низкого уровня по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, мА	I _{OL} **	-	20	-	25
9 Выходной ток высокого уровня по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, мА	I _{OH} ***	-3	-	-5	-
10 Длительность фронтов входных сигналов, нс - для входа BQ1 - для остальных входов	t _{LH} , t _{HL}	-	10 20	-	-
11 Тактовая частота, МГц	f _{Cl}	0	8	-	-
12 Емкость нагрузки по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, пФ	C _L	-	80	-	150

* Между напряжением питания аналоговой части и цифровой части микросхемы должно сохраняться соотношение $|U_{#VCC} - U_{nvcc}| \leq 0,3$.

** Сумма всех I_{OL} по всем выводам не должна превышать 400 мА.

Сумма всех I_{OL} по выводам A0 – A7 не должна превышать 100 мА.

Сумма всех I_{OL} по выводам B0 – B3 не должна превышать 100 мА.

Сумма всех I_{OL} по выводам B4 – B7 не должна превышать 100 мА.

Сумма всех I_{OL} по выводам C0 – C3 не должна превышать 100 мА.

Сумма всех I_{OL} по выводам C4 – C7 не должна превышать 100 мА.

Сумма всех I_{OL} по выводам D0 – D3 и XTAL2 не должна превышать 100 мА.

Сумма всех I_{OL} по выводам D4 – D7 не должна превышать 100 мА.

Если I_{OL} превышает условия тестирования, V_{OL} может превышать заявленные значения. Выводы не гарантируют нагрузочный ток, превышающий условия тестирования.

*** Сумма всех I_{OH} по всем выводам не должна превышать 400 мА.

Сумма всех I_{OH} по выводам A0 – A7 не должна превышать 100 мА.

Сумма всех I_{OH} по выводам B0 – B3 не должна превышать 100 мА.

Сумма всех I_{OH} по выводам B4 – B7 не должна превышать 100 мА.

Сумма всех I_{OH} по выводам C0 – C3 не должна превышать 100 мА.

Сумма всех I_{OH} по выводам C4 – C7 не должна превышать 100 мА.

Сумма всех I_{OH} по выводам D0 – D3 и XTAL2 не должна превышать 100 мА.

Сумма всех I_{OH} по выводам D4 – D7 не должна превышать 100 мА.

Если I_{OH} превышает условия тестирования, V_{OH} может превышать заявленные значения. Выводы не гарантируют нагрузочный ток, превышающий условия тестирования.

3 Функциональное описание ИМС 1887ВЕ1У

1887ВЕ1У представляет собой маломощный КМОП 8-разрядный микроконтроллер, построенный на расширенной архитектуре RISC. Благодаря выполнению команд в течение одного тактового цикла, 1887ВЕ1У достигает производительности в 1 миллион операций в секунду, что позволяет разработчику систем оптимизировать потребляемую мощность с учетом скорости обработки данных.

Ядро микропроцессора объединяет в себе богатый набор команд с 32 рабочими регистрами общего назначения. Все 32 регистра связаны непосредственно с арифметическим логическим устройством (АЛУ), что позволяет обеспечивать выборку двух независимых регистров, которые напрямую связаны с АЛУ. Это позволяет одной простой командой в течение одного тактового цикла обеспечить доступ к двум независимым регистрам. С точки зрения кодирования такая архитектура более эффективна, при этом, по сравнению с обычными CISC микроконтроллерами достигается в десять раз большая производительность.

Микроконтроллер 1887ВЕ1У содержит: 8К байт внутрисистемной программируемой флэш-памяти с возможностью считывания во время записи, ЭСППЗУ емкостью 512 байт, 512 байт СОЗУ, 32 линии ввода-вывода общего назначения, 32 рабочих регистра общего назначения, три гибких таймера/счетчика с режимами сравнения, внутреннее и внешнее прерывания, последовательный программируемый универсальный синхронно-асинхронный последовательный порт (УСАПП), побайтно ориентированный двухпроводный последовательный интерфейс, 8-канальный 10-разрядный АЦП с дополнительным дифференциальным входным каскадом с программируемым усилением. Микроконтроллер имеет также программируемый сторожевой таймер с внутренним генератором, последовательный SPI порт и шесть энергосберегающих режимов (sleep), выбираемых программно. Режим холостого хода останавливает ЦПУ, в то время как СОЗУ, таймеры/счетчики, SPI порт и система прерывания продолжают работать. Режим пониженной мощности сохраняет содержимое регистров, но приостанавливает генератор, отключая все остальные функции кристалла до наступления следующего прерывания или общего сброса. В энергосберегающем режиме продолжает работать асинхронный таймер, что позволяет пользователю поддерживать базу таймера, в то время как остальная часть устройства находится в спящем режиме. Режим пониженного шума АЦП останавливает ЦПУ и все модули входа/выхода для того, чтобы свести к минимуму шум при переключении во время АЦП преобразований. В режиме ожидания работает кварцевый генератор, в то время как остальная часть устройства находится в спящем режиме. Это позволяет обеспечить очень быстрый запуск при малой потребляемой мощности. В длительном режиме ожидания продолжают работать основной генератор и асинхронный таймер.

Прибор изготавливается с использованием технологии по производству нестираемой памяти с высокой плотностью упаковки. Встроенная в кристалл флэш-память с внутрисистемным программированием позволяет перепрограммировать память с помощью последовательного SPI интерфейса или же с помощью также встроенной в кристалл памяти загрузки. Программа загрузки может использовать любой интерфейс для загрузки прикладных программ в флэш-память. Программное обеспечение (ПО) в секции загрузки флэш-памяти будет продолжать работать во время обновления прикладной флэш-памяти, благодаря чему обеспечивается реальное считывание при записи. Благодаря объединению 8-разрядного RISC ЦПУ со встроенной самопрограммируемой флэш-памятью на одном кристалле, контроллер 1887ВЕ1У является маломощным микроконтроллером, обеспечивающим чрезвычайно гибкое и экономичное решение для многих видов применения систем встроенного управления.

1887ВЕ1У поддерживает полный набор программ и средств системной разработки, включая: С компиляторы, макроассемблеры, средства отладки и моделирования, внутрисхемные эмуляторы и оценочные комплекты.

Обозначение вывода	Описание вывода
1	2
#VCC	Напряжение питания
#GND	Земля
Port A (PA7...PA0)	<p>Аналоговые входы АЦП.</p> <p>Port A является также 8-разрядным двунаправленным портом ввода-вывода, если не используется АЦП.</p> <p>Выходы порта могут служить также в качестве внутренних нагрузочных резисторов (выбираются для каждого разряда). Выходные буферы Port A имеют симметричные модуляционные характеристики с высокой нагрузочной способностью и емкостью источника.</p> <p>Когда выводы PA0...PA7 используются в качестве входов и извне переводятся в низкое состояние, они становятся источниками тока, если активированы внутренние нагрузочные резисторы.</p> <p>Выходы Port A имеют три состояния в случае активации сброса, даже если счетчик не работает.</p>
Port B (PB7...PB0)	<p>Port B является 8-разрядным двунаправленным портом ввода-вывода с внутренними нагрузочными резисторами (выбираемыми для каждого разряда). Буферы Port B имеют симметричные модуляционные характеристики с высокой нагрузочной способностью и емкостью источника.</p> <p>Если выводы Port B используются в качестве входов, то с помощью внешнего воздействия они переводятся в низкое состояние и будут генерировать ток при активации нагрузочных резисторов.</p> <p>Выходы Port B имеют три состояния в случае активации сброса, даже если счетчик не работает.</p> <p>Port B выполняет также различные специальные функции.</p>
Port C (PC7...PC0)	<p>Port C является 8-разрядным двунаправленным портом ввода-вывода с внутренними нагрузочными резисторами (выбираемыми для каждого разряда). Буферы Port C имеют симметричные модуляционные характеристики с высокой нагрузочной способностью и емкостью источника. Если выводы Port C используются в качестве входов, то с помощью внешнего воздействия они переводятся в низкое состояние и будут генерировать ток при активации нагрузочных резисторов.</p> <p>Выходы Port C имеют три состояния в случае активации сброса, даже если счетчик не работает.</p>
Port D (PD7...PD0)	<p>Port D является 8-разрядным двунаправленным портом ввода-вывода с внутренними нагрузочными резисторами (выбираемыми для каждого разряда). Буферы Port D имеют симметричные модуляционные характеристики с высокой нагрузочной способностью и емкостью источника. Если выводы Port D используются в качестве входов, то с помощью внешнего воздействия они переводятся в низкое состояние и будут генерировать ток при активации нагрузочных резисторов.</p> <p>Выходы Port D имеют три состояния в случае активации сброса, даже если счетчик не работает.</p> <p>Port D выполняет также различные специальные функции.</p>
RESET	<p>Выход для сигнала общего сброса. Состояние низкого уровня на этом выводе продолжительнее, чем минимальная длина импульса, генерирующая сигнал сброса, даже если счетчик не работает. Не гарантируется, что более короткие импульсы осуществлят сброс.</p>

Обозначение вывода	Описание вывода
1	2
BQ1	Вход инвертирующего усилителя тактового генератора и вход рабочей схемы внутреннего тактового генератора.
BQ2	Выход инвертирующего усилителя тактового генератора.
□VCC	□VCC представляет собой вывод напряжения питания для Port A и АЦП. Он должен быть внешне подсоединен к #VCC, даже если АЦП не используется. Если АЦП используется, то он должен быть подсоединен к #VCC через фильтр низких частот.
□REF	□REF является аналоговым контрольным выводом для АЦП.

3.1 Ядро микроконтроллера

Данный раздел описывает архитектуру ядра микропроцессора в целом. Основной функцией ядра ЦПУ является обеспечение правильного исполнения программы. С учетом этого ЦПУ должен иметь возможность доступа к устройствам памяти, выполнения вычислений, управления периферийными устройствами и обращения с прерываниями.

3.1.1 Анализ архитектуры

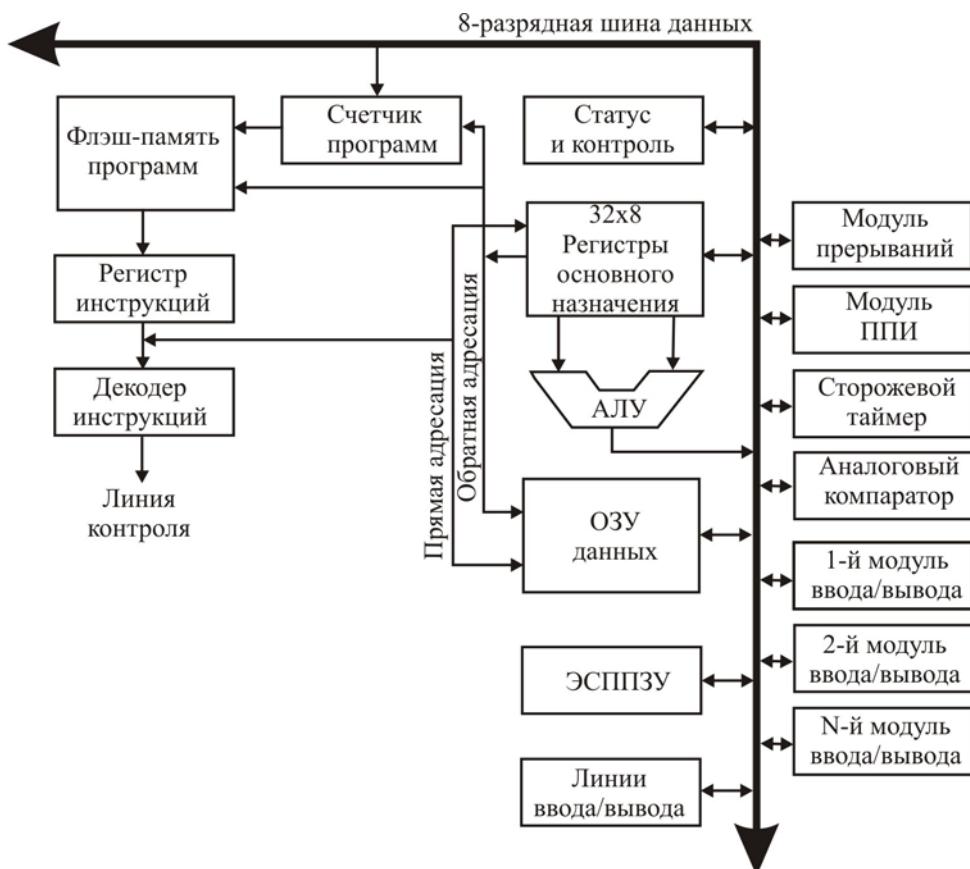


Рисунок 3.1 – Блок-схема архитектуры ядра микропрограммного управления

Для того чтобы максимально улучшить характеристики и параллелизм, в микропроцессоре используется Гарвардская архитектура с отдельными устройствами памяти и шинами для программ и данных. Команды в памяти программ выполняются на едином

уровне конвейеризации. Во время исполнения одной команды следующая команда предварительно извлекается из памяти программ. Такой принцип позволяет исполнять команды за каждый тактовый цикл. Память программ представляет собой внутрипроцессорную перепрограммируемую флэш-память.

Регистр файлов с ускоренным доступом содержит 32 8-разрядных рабочих регистра общего назначения с единственным временем доступа за тактовый цикл. Это позволяет обеспечить работу АЛУ за один цикл. В типичном режиме работы АЛУ из файла регистра берутся два операнда, выполняется операция, и результат опять сохраняется в файле регистра; все это выполняется за один тактовый цикл.

Шесть из 32 регистров могут использоваться как три 16-разрядных указателя адреса регистра для адресации пространства данных, позволяя обеспечить эффективные вычисления адресов. Один из этих указателей адреса может быть также использован как указатель адреса для таблиц просмотра (look up tables) во флэш-памяти программ. Эти дополнительные регистры функций являются 16-разрядными X-, Y- и Z-регистрами, описанными далее в этом разделе.

АЛУ поддерживает арифметические и логические операции между регистрами или между константой и регистром. АЛУ может выполнить одну операцию с регистром. После арифметической операции регистр состояния обновляется для вывода информации о результатах операции.

Поток программы обеспечивается условным и безусловным переходом и командами вызова, которые способны обеспечить непосредственную адресацию ко всему адресному пространству. Большинство команд имеют единый 16-разрядный формат слова. Каждый адрес программы памяти содержит 16-и или 32-разрядную команду.

Пространство флэш-памяти программы разделено на две секции: секцию программы загрузки и секцию прикладной программы. Обе секции имеют специальные блокирующие разряды для защиты записи и считывания/записи. Команда SPM, которая осуществляет запись в секцию флэш-памяти прикладных программ, должна находиться в секции программы загрузки.

Во время запросов на прерывание и исполнение подпрограмм, обратный адрес счетчика программ (PC) хранится в стеке. Стек расположен в СОЗУ общих данных, его размер ограничен только общим объемом СОЗУ и характером использования СОЗУ. Все программы пользователя должны инициализировать SP во время процедуры сброса (до исполнения подпрограмм или прерываний). Указатель стека SP доступен для считывания/записи в пространстве ввода-вывода. СОЗУ данных легко доступно с помощью пяти различных режимов адресации, поддерживаемых архитектурой микроконтроллера.

Все пространства памяти в архитектуре процессора являются линейными и регулярными.

Гибкий модуль прерывания имеет собственные регистры управления в пространстве ввода-вывода с дополнительным разрядом разрешения общего прерывания в регистре состояния. Все прерывания имеют отдельный вектор прерывания в таблице вектора прерывания. Прерывания имеют приоритет в зависимости от положения их вектора. Чем ниже адрес вектора прерывания, тем выше приоритет.

Пространство памяти ввода-вывода содержит 64 адреса для таких периферийных функций ЦПУ, как Регистры Управления, SPI и другие функции ввода-вывода. Доступ к памяти ввода-вывода может осуществляться напрямую или адресацией пространства данных, следующих за аналогичными адресами файла регистров, 0x20 - 0x5F.

3.1.2 АЛУ – Арифметико-логическое устройство

Высокоэффективное АЛУ работает непосредственно со всеми 32-мя рабочими регистрами общего назначения. В течение одного тактового цикла выполняются арифметические операции между регистрами общего назначения или между регистром и соседним регистром. Операции АЛУ подразделяются на три категории – арифметические, логические и функции разрядов.

Регистр состояний содержит информацию о результатах самых последних выполненных арифметических команд. Данная информация может быть использована для изменения потока программ при выполнении условных операций. Необходимо обратить внимание на то, что регистр состояний обновляется после выполнения всех операций АЛУ. В подобной ситуации, во многих случаях, это устраняет необходимость в использовании специальных команд сравнения, что обеспечивает ускоренный и более компактный код. Регистр состояния не сохраняется автоматически при вводе процедуры прерывания и сохраняется при возвращении из прерывания. Подобная ситуация должна решаться использованием ПО.

Определение регистра состояния SREG – выглядит следующим образом:

Бит	7	6	5	4	3	2	1	0	SREG
	I	T	H	S	V	N	Z	C	
Чтение/запись	R/W								
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7: I - Разрешение общего прерывания

Разряд разрешения общего прерывания должен быть установлен для разрешения прерываний. После этого в отдельных регистрах прерывания выполняется управление индивидуальным прерыванием. Если очищается Регистр Разрешения Общего Прерывания, то не разрешается ни одно из прерываний, несмотря на индивидуальные установки разрешения прерывания. Разряд I очищается с помощью аппаратного обеспечения после ввода прерывания и затем устанавливается с помощью команды RETI для разрешения последующих прерываний. Разряд I может быть также установлен и затем сброшен с помощью команд SEI и CLI, как это указано в описании команд.

Разряд 6: T - Хранение копии разряда

Команды копирования разряда BLD (Bit Load) и BST (Bit STore) используют разряд T в качестве источника или места для разряда, с которым работают. Разряд из регистра в файле регистра может быть скопирован в T с помощью команды BST, а разряд из T может быть скопирован в разряд в регистре, находящемся в файле регистра, с помощью команды BLD.

Разряд 5: H - Флаг половинного переноса

Флаг половинного переноса H обозначает половинный перенос в некоторых арифметических операциях. Половинный перенос используется в двоично-десятичных арифметических преобразованиях.

Разряд 4: S - Знаковый разряд, S = N^V

Разряд S всегда означает “исключающее или” между отрицательным флагом N и флагом V двойного дополнительного переполнения.

Разряд 3: V - Флаг двойного дополнительного переполнения

Флаг V двойного дополнительного переполнения поддерживает арифметику двойного дополнения. Для подробного рассмотрения см. приложение А « Описание системы команд».

Разряд 2: N - Отрицательный флаг

Отрицательный флаг N означает отрицательный результат при арифметической или логической операции. Для подробного рассмотрения см. приложение А “Описание системы команд”.

Разряд 1: Z - Нулевой флаг

Нулевой флаг Z означает нулевой результат арифметической или логической операции. Для подробного рассмотрения см.приложение А “Описание системы команд”.

Разряд 0: C - Флаг переноса

Флаг переноса С обозначает перенос в логической или арифметической операции.

3.1.3 Регистр файлов общего назначения

Регистр файлов оптимизирован для расширенного набора команд. Для достижения требуемых характеристик и гибкости регистром файлов поддерживаются следующие варианты ввода-вывода:

- один 8-разрядный выходной операнд и один 8-разрядный ввод результата;
- два 8-разрядных выходных операнда и один 8-разрядный ввод результата;
- два 8-разрядных выходных операнда и один 16-разрядный ввод результата;
- один 16-разрядный выходной операнд и один 16-разрядный ввод результата.

На рисунке 3.2 показана структура 32 рабочих регистров общего назначения в ЦПУ.

7	0	Адрес	
	R0	0x00	
	R1	0x01	
	...		
	R12	0x0C	
	R13	0x0D	
	R14	0x0E	
	R15	0x0F	
	R16	0x10	
	...		
	R25	0x19	
	R26	0x1A	X-регистр мл. байт
	R27	0x1B	X-регистр ст. байт
	R28	0x1C	Y-регистр мл. байт
	R29	0x1D	Y-регистр ст. байт
	R30	0x1E	Z-регистр мл. байт
	R31	0x1F	Z-регистр ст. байт

мл.-младший;
ст.-старший.

Рисунок 3.2 - Рабочие регистры общего назначения ЦПУ

Большинство команд, работающих с регистром файлов, имеют прямой доступ ко всем регистрам и большинство из них представляют собой одноцикловые команды.

Как показано на рисунке 3.2, каждому регистру приписывается также адрес памяти данных. При этом регистры непосредственно размещаются в первые 32 ячейки пространства данных пользователя. Хотя физически они не выполнены в виде адресов СОЗУ, подобная организация памяти обеспечивает большую гибкость с точки зрения доступа к регистрам, поскольку указатели X-, Y- и Z-регистров могут быть установлены для указания любого регистра в файле.

Регистр X, регистр Y и регистр Z

Регистры R26...R31 имеют некоторые функции, добавленные к их общему назначению. Эти регистры являются 16-разрядными указателями адреса для непрямой адресации пространства данных. Три косвенных регистра адреса X, Y и Z определяются, как показано на рисунке 3.3.

X-register	15 7	XH R27(0×1B)	XL R26(0×1A) 0	0
Y-register	15 7	YH R29(0×1D)	YL R28(0×1C) 0	0
Z-register	15 7	ZH R31(0×1F)	ZL R30(0×1E) 0	0

Рисунок 3.3 - X-, Y-, Z- регистры

В различных режимах адресации эти регистры адресов имеют такие функции, как фиксированные перемещения, автоматическое приращение и автоматическое уменьшение.

3.1.4 Указатель стека

Стек используется в основном для хранения временных данных, для хранения локальных переменных и для хранения возвратных адресов после прерываний и вызовов подпрограмм. Регистр указателя стека всегда указывает на верхнюю часть стека. Необходимо обратить внимание на то, что стек выполнен по возрастанию от более высоких адресов памяти к более низким адресам. Это предполагает, что команда стека PUSH уменьшает указатель стека.

Указатель стека указывает на ту часть стека данных СОЗУ, в которой расположены стеки подпрограмм и прерываний. Это пространство стека в данных СОЗУ должно определяться программой до того, как исполняются любые вызовы подпрограмм, или разрешаются прерывания. Указатель стека должен быть установлен на точку выше 0×60. Указатель стека уменьшается на один по мере того, как данные перемещаются в стек с помощью команды PUSH, и он уменьшается на два, когда обратный адрес перемещается в стек с помощью вызова подпрограммы или прерывания. Указатель стека увеличивается на один, когда данные выходят из стека с помощью команды POP, и увеличивается на два, когда данные выходят из стека, возвращаясь из подпрограммы RET или возвращаясь из прерывания RETI.

Указатель стека выполняется в виде двух 8-разрядных регистров в пространстве ввода-вывода. Количество используемых разрядов, фактически, зависит от реализации. Необходимо обратить внимание на то, что пространство данных при реализации данной архитектуры настолько мало, что требуется только SPL. В этом случае регистр SPH не будет присутствовать.

Бит	15	14	13	12	11	10	9	8	SPH
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
Read/Write	7 R/W	6 R/W	5 R/W	4 R/W	3 R/W	2 R/W	1 R/W	0 R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Временные диаграммы исполнения команд

Ниже описываются общие концепции временных диаграмм доступа для выполнения команд. Центральное процессорное устройство запускается с помощью тактового генератора ЦПУ, который возбуждается непосредственно от выбранного для кристалла источника тактовых импульсов. При этом не используется внутреннее деление тактовой частоты.

На рисунке 3.4 представлена параллельная выборка команд и их исполнение, позволяемое Гарвардской архитектурой и концепцией регистра файлов с быстрым доступом. Это представляет собой основную концепцию работы в режиме конвейеризации для получения производительности вплоть до 1 миллиона команд в секунду, что, в свою очередь, обеспечивает уникальные результаты по сочетанию производительности со стоимостью, количеству функций на тактовую частоту и функций на единицу мощности.

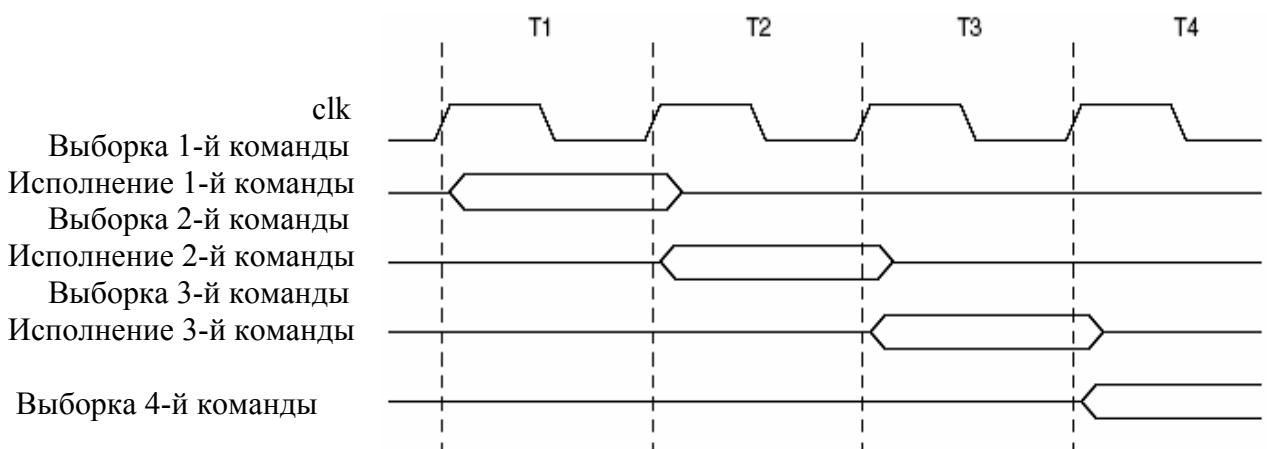


Рисунок 3.4 - Выборки параллельной команды и исполнения команд

На рисунке 3.5 представлена временная концепция для регистра файла. В одном тактовом цикле выполняется операция АЛУ с использованием двух operandов регистра, и результат сохраняется обратно в регистре назначения.

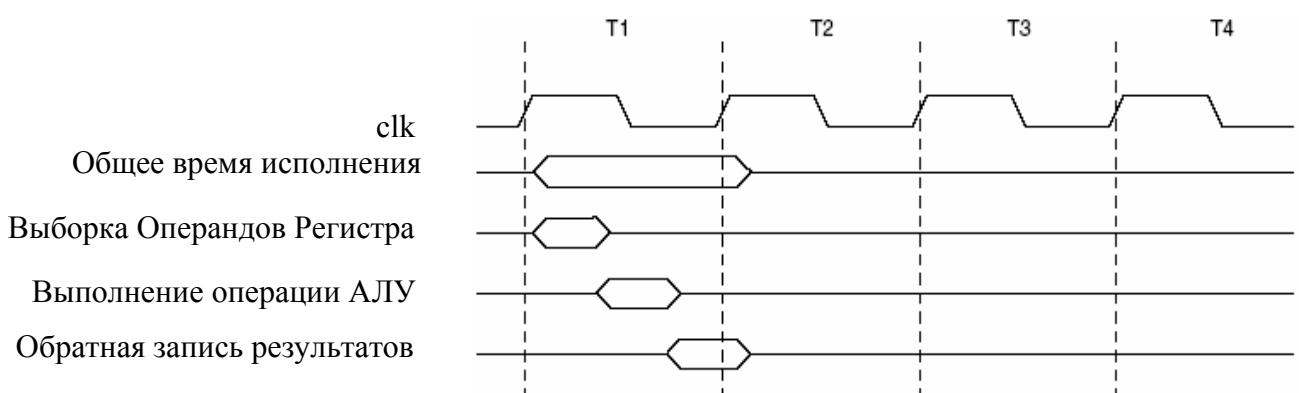


Рисунок 3.5 - Работа АЛУ в одиночном цикле

3.1.5 Сброс и обработка прерываний

1887ВЕ1У содержит несколько различных источников прерывания. Каждое из этих прерываний и отдельный вектор сброса имеют отдельный программный вектор в пространстве памяти программ. Все прерывания имеют свои отдельные разряды разрешения, которые должны быть записаны логической единицей вместе с разрядом разрешения глобального прерывания в регистре состояний для того, чтобы разрешить прерывание. В зависимости от значения счетчика программ, прерывания могут быть автоматически заблокированы, когда программируются разряды BLB02 или BLB12 блокировкой загрузки. Подобная особенность повышает безопасность программного обеспечения. Для подробного рассмотрения см. в подразделе 3.20 «Программирование памяти».

Младшие адреса в пространстве памяти программ, по умолчанию, определены как векторы сброса и прерывания. Перечень устанавливает также уровень приоритетов для различных прерываний. Чем ниже адрес, тем выше уровень приоритета. Сброс имеет наивысший приоритет, а следующим является INT0 – Запрос 0 внешнего прерывания. Векторы прерывания могут быть перемещены в начало секции загрузки флэш-памяти путем установки разряда IVSEL в регистре управления общим прерыванием (GICR). Для подробного рассмотрения вопроса по “Прерываниям” см. соответствующий подраздел 3.7. Вектор сброса может быть также перемещен в начало секции загрузки флэш-памяти с помощью программирования перемычки BOOTRST, см. подраздел 3.19 “Поддержка загрузчика – самопрограммирование”.

При возникновении прерывания I-разряд разрешения глобального прерывания очищается, и все прерывания блокируются. Пользовательское ПО может записывать логическую единицу к I-разряду для разрешения вложенных прерываний. Все разрешенные прерывания могут прерывать текущий процесс прерывания. Разряд I устанавливается автоматически при исполнении возврата из команды прерывания RETI.

Существует два основных типа прерываний. Первый тип запускается событием, которое устанавливает флаг прерывания. Для этих прерываний счетчик программ выбирает действующий вектор прерываний для исполнения программы обращения с прерываниями, при этом с помощью аппаратного обеспечения очищается соответствующий флаг прерывания. Флаги прерывания могут также очищаться путем записи логической единицы в позицию разряда флага очистки. Если возникает условие прерывания пока очищен соответствующий разряд прерывания, флаг прерывания будет установлен и запомнен до разрешения прерывания или флаг будет очищен программным способом. Подобным образом, если возникает одно или более условий прерывания при сброшенном разряде Разрешения глобального прерывания, будет установлен и внесен в память соответствующий флаг(и) до тех пор, пока установится, а затем будет выполнен в порядке приоритета разряд разрешения глобального прерывания.

Второй тип прерываний будет запущен как только возникнет условие прерывания. Эти прерывания не обязательно должны иметь флаги прерывания. Если условие прерывания исчезает до разрешения прерывания, то прерывание не будет запущено.

После выхода из прерывания микропроцессор всегда возвращается к основной программе и исполняет одну основную команду до обслуживания любого ждущего прерывания.

Обратите внимание на то, что регистр статуса не сохраняется автоматически при возврате из процедуры прерывания. Это должно выполняться программным способом.

При использовании команды CLI для блокировки прерывания, прерывания будут немедленно заблокированы. Ни одно прерывание не будет выполняться после команды CLI, даже если оно происходит одновременно с командой CLI. Следующий пример показывает, как это может быть использовано для того, чтобы избежать прерываний во время запланированной последовательности записи ЭСППЗУ.

Пример ассемблерного кода
in r16, SREG ; запись значения в SREG
cli ; блокировка прерываний в течение времени цикла
sbi EECR, EEMWE ; запуск записи в EEPROM
sbi EECR, EEWE
out SREG, r16 ; восстановление значения SREG (I-bit)
Пример С-кода
char cSREG; cSREG = SREG; /* запись значения в SREG */ /* блокировка прерываний в течение времени цикла */ _CLI(); EECR = (1<<EEMWE); /* запуск записи в EEPROM */ EECR = (1<<EEWE); SREG = cSREG; /* восстановление значения SREG (I-bit)*/

При использовании команды SEI для разрешения прерываний команда, следующая за SEI, будет выполнена до любого ожидаемого прерывания, как это показано в данном примере.

Пример ассемблерного кода
sei ; установка разрешения глобального прерывания
sleep ; включение режима sleep, ожидание прерываний
; заметка: режим sleep включен до возникновения любого
; прерывания(ий)
Пример С-кода
_SEI(); /* установка разрешения глобального прерывания */ _SLEEP(); /* включение режима sleep, ожидание прерываний */ /* заметка: режим sleep включен до возникновения любого прерывания(ий)*/

Время отклика прерывания

Отклик исполнения прерывания для всех разрешенных прерываний составляет минимум четыре тактовых цикла. После четырех тактовых циклов выбирается адрес вектора программ для фактической процедуры обращения к прерываниям. Во время этого периода четырех тактовых циклов счетчик программ переводится в стек. Вектор обычно скачком переходит в процедуру прерывания, и этот переход занимает три тактовых цикла. Если прерывание происходит во время исполнения команды, состоящей из множества циклов, то команда завершается до того, как обслуживается прерывание. Если прерывание происходит, когда модуль микроконтроллера находится в спящем режиме, то время отклика для запуска из выбранного спящего режима, возвращение из процедуры обращения с прерыванием занимает четыре тактовых цикла. Это позволяет быстрее восстанавливаться при активированном режиме Sleep.

Возвращение из режима прерывания обычно занимает четыре тактовых цикла. Во время этих четырех циклов, Счетчик программ (два байта) быстро возвращается из стека, указатель стека возрастает на два, и устанавливается разряд I в SREG.

3.2 Устройства памяти микроконтроллера 1887ВЕ1У

Этот подраздел описывает различные устройства памяти в 1887ВЕ1У. Архитектура процессора имеет два основных пространства памяти: память данных и память программ. Кроме того, 1887ВЕ1У отличается наличием ЭСППЗУ для хранения данных. Все три пространства памяти являются линейными и регулярными.

3.2.1 Внутрисистемная перепрограммируемая флэш-память программ

1887ВЕ1У содержит 8К байт встроенной в кристалл внутрисистемной перепрограммируемой флэш-памяти для хранения программ. Поскольку все команды процессора имеют ширину 16 или 32 разряда, флэш-память организована в виде $4K \times 16$. Для повышения безопасности ПО пространство флэш-памяти разделено на две секции: секцию загрузки программ и секцию прикладных программ.

Флэш-память имеет срок службы, по меньшей мере, 10000 циклов записи и стирания. Счетчик программ (PC) для 1887ВЕ1У имеет разрядность, равную 12, что позволяет адресацию к 4К адресов памяти программ. Работа области программы загрузки и связанных разрядов блокировки загрузки подробно описаны в подразделе 3.19 “Поддержка загрузчика – самопрограммирование”.

Таблицы постоянных величин могут располагаться внутри адресного пространства всей памяти программ.

На рисунке 3.6 показана организация СОЗУ для 1887ВЕ1У.

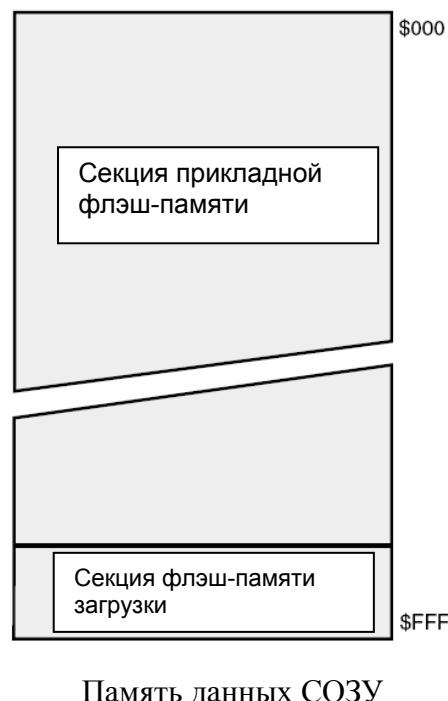


Рисунок 3.6 - Карта памяти программ

608 адресов памяти данных выделены для файла регистра, памяти ввода-вывода и СОЗУ внутренних данных. Первые 96 позиций предназначены для размещения адресов файла регистра и ввода-вывода памяти, а следующие 512 позиций - для СОЗУ внутренних данных.

Пять различных режимов адресации для памяти данных включают: прямой, косвенный с перемещением, косвенный, косвенный с преддекрементом и косвенный с по-

стинкрементом. В файле регистров регистры R26 - R31 отличаются наличием регистров указателей с косвенной адресацией.

Непосредственная адресация охватывает все пространство данных.

Режим косвенный с перемещением охватывает 63 места адресов из основного адреса, выданного регистром Y или Z.

При использовании режимов с косвенной адресацией, регистров с автоматическим преддекрементом и постинкрементом, регистры адресов X, Y и Z уменьшаются или возрастают.

32 рабочих регистра общего назначения, 64 регистра ввода-вывода и 512 байт СОЗУ внутренних данных в 1887ВЕ1У доступны с помощью всех упомянутых режимов адресации.

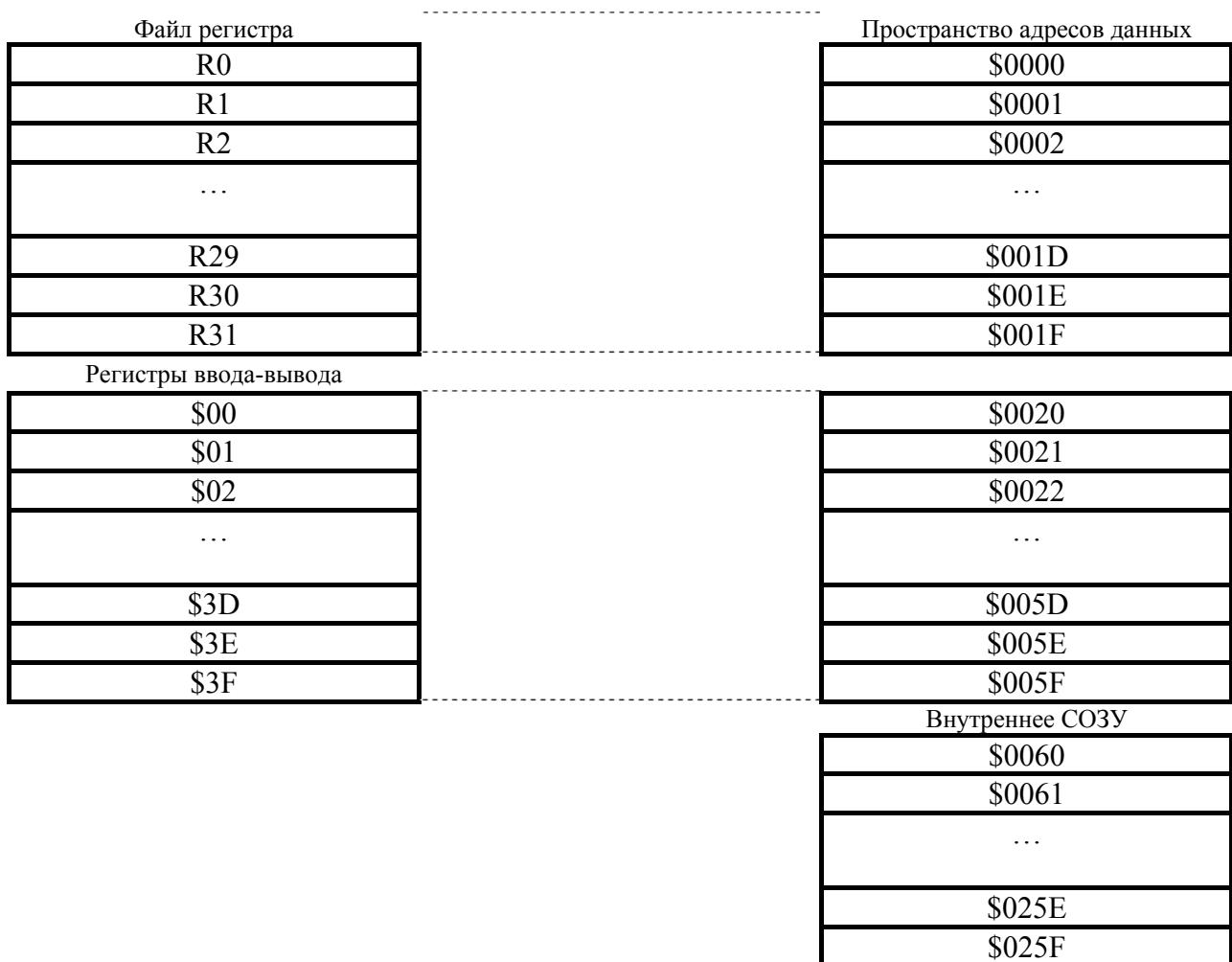


Рисунок 3.7 - Карта памяти данных

Времена доступа к памяти данных

Ниже описываются общие понятия, касающиеся времени для доступа к внутренней памяти. Доступ к внутренним данным СОЗУ выполняется в течение двух тактовых циклов clkCPU, как это показано на рисунке 3.8.

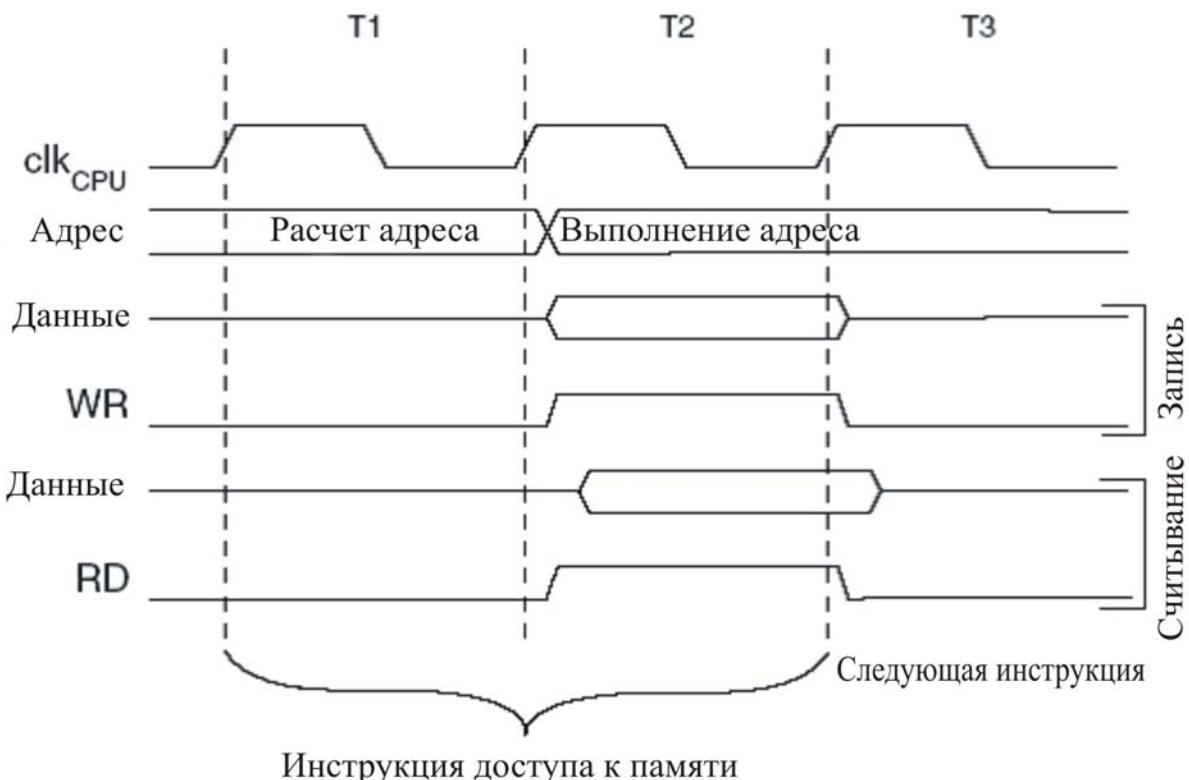


Рисунок 3.8 - Циклы доступа к данным СОЗУ на кристалле

3.2.2 Память данных ЭСППЗУ

1887ВЕ1У содержит 512 байт памяти данных ЭСППЗУ. Она организована в виде отдельного пространства данных, в котором могут считывааться и записываться отдельные байты. ЭСППЗУ имеет срок службы не менее 100000 циклов записи/считывания. Доступ между ЭСППЗУ и ЦПУ описан далее с указанием регистров адреса, регистров данных и регистра управления ЭСППЗУ.

Доступ считывания/записи ЭСППЗУ

Регистры доступа к ЭСППЗУ доступны в пространстве ввода-вывода.

Время доступа при записи для ЭСППЗУ приведено в таблице 3.1. При этом функция синхронизации позволяет пользователю ПО определять время, когда может быть записан следующий байт. Если пользовательский код содержит инструкции для записи ЭСППЗУ, то необходимо предпринять определенные меры предосторожности. Для источников питания с мощными фильтрами существует вероятность, что VCC при включении/выключении будет возрастать или спадать медленно. При этом устройство в течение некоторого периода времени будет работать при более низких напряжениях, чем те, которые указаны в ТУ как минимальные для используемой тактовой частоты.

Для предотвращения случайной записи ЭСППЗУ необходимо обеспечить выполнение определенной процедуры записи.

При считывании ЭСППЗУ, ЦПУ приостанавливается на четыре тактовых цикла перед исполнением следующей команды. При записи ЭСППЗУ, ЦПУ приостанавливается на два тактовых цикла перед исполнением следующей команды.

Адрес ЭСППЗУ

Регистр – EEARH и EEARL

Бит	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	-	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
Чтение/запись	R R/W	R/W R/W							
Начальное значение	0 X	X X							

Разряды 15...9: Res - Резервные разряды

Эти разряды в 1887ВЕ1У являются резервными и всегдачитываются как ноль.

Разряды 8...0: EEAR8...0 - Адрес СППЗУ

Регистры адреса ЭСППЗУ – EEARH и EEARL - указывают адрес ЭСППЗУ в 512 байтах пространства ЭСППЗУ. Байты данных ЭСППЗУ адресуются линейным образом в пространстве от 0 до 511. Начальное значение EEAR не определено. Надлежащее значение должно быть записано до того, как будет получен доступ к ЭСППЗУ.

Регистр данных ЭСППЗУ – EEDR

Бит	7	6	5	4	3	2	1	0	
								LSB	EEDR
Чтение/запись	R/W 0								
Начальное значение	0 0								

Разряды 7...0: EEDR7...0 - Данные ЭСППЗУ

Для операции записи ЭСППЗУ регистр EEDR содержит данные, которые должны быть записаны в ЭСППЗУ в адрес, предложенный регистром EEAR. Для операции считывания ЭСППЗУ EEDR содержит данные, считываемые из ЭСППЗУ в адресе, предложенным EEAR.

Управление ЭСППЗУ

Регистр – EECR

Бит	7	6	5	4	3	2	1	0	
					EERIE	EEMWE	EEWE	EERE	EECR
Чтение/запись	R 0	R 0	R 0	R 0	R/W 0	R/W 0	R/W 0	R/W X	
Начальное значение	0 0	0 0	0 0	0 0	0 0	0 0	0 X	0 0	

Разряды 7...4: Res - Сохраненные разряды

Для 1887ВЕ1У эти разряды являются резервными и всегда будут считываться как ноль.

Разряд 3: EERIE - Разрешение готовности к прерыванию ЭСППЗУ

Запись EERIE в единицу разрешает готовность прерывания ЭСППЗУ, если установлен разряд I в SREG. Запись EERIE в состояние нуля блокирует прерывание. Готовность прерывания ЭСППЗУ формирует постоянное прерывание, когда EEWE очищено.

Разряд 2: EEMWE - Разрешение записи основного ЭСППЗУ

Разряд EEMWE определяет, будет ли установка EEWE на единицу вызывать запись в ЭСППЗУ. При установке EEMWE установка EEWE в течение четырех тактовых циклов будет инициировать запись данных в ЭСППЗУ в выбранные адреса. Если EEMWE равно нулю, то установка EEWE не оказывает воздействия. Если EEMWE была записана в состояние единицы программным способом, то устройство сбрасывает разряд на ноль после четырех тактовых циклов.

Разряд 1: EEWE - Разрешение записи ЭСППЗУ

EEWE - сигнал разрешения записи ЭСППЗУ представляет собой стробирующий сигнал записи в ЭСППЗУ. Если адрес и данные указаны правильно, то разряд EEWE должен быть записан в единицу до того, как логическая единица записывается в EEWE, иначе не произойдет запись в ЭСППЗУ. При записи в ЭСППЗУ необходимо соблюдать следующую последовательность (порядок действий 3 и 4 не является существенным):

- 1 Дождаться, пока EEWE не станет равным нулю.
- 2 Дождаться, пока SPMEN в SPMCR не станет равной нулю.
- 3 Записать новый адрес ЭСППЗУ в EEAR (опция).
- 4 Записать новые данные ЭСППЗУ в EEDR (опция).
- 5 Записать логическую единицу в разряд EEMWE при выполнении записи в EEWE в EECR.

6 В течение четырех тактовых циклов после настройки EEMWE записать логическую единицу в EEWE.

ЭСППЗУ нельзя программировать во время записи ЦПУ во флэш-память. С помощью программного обеспечения необходимо убедиться в том, что программирование флэш-памяти завершено до начала нового процесса записи ЭСППЗУ. Шаг 2 целесообразен только в том случае, если ПО содержит загрузчик операционной системы, позволяющий ЦПУ программировать флэш-память. Если флэш-память никогда не обновляется с помощью ЦПУ, то шаг 2 можно пропустить.

Внимание: Прерывание между шагом 5 и шагом 6 вызовет отказ цикла записи, поскольку истечет время мастера разрешения записи ЭСППЗУ. Если процедура прерывания при доступе к ЭСППЗУ прерывает другой доступ к ЭСППЗУ, то содержимое регистра EEAR или EEDR будет изменено, что вызовет отказ прерванного доступа к ЭСППЗУ. Рекомендуется выполнить очистку флага глобального прерывания в течение всех этапов, чтобы избежать этих проблем. По истечении времени доступа к записи, разряд EEWE очищается аппаратным способом. Пользовательское ПО может опросить этот разряд и дождаться нуля перед записью следующего байта. После того как EEWE установилось, ЦПУ останавливается на два цикла перед тем, как исполнится следующая команда.

Разряд 0: EERE - Разрешение считывания ЭСППЗУ

EERE – сигнал разрешения считывания ЭСППЗУ является стробирующим сигналом считывания для ЭСППЗУ. При установке правильного адреса в регистре EEAR разряд EERE должен быть записан в логическую единицу для запуска считывания ЭСППЗУ. Доступ считывания ЭСППЗУ исполняет одну команду, и запрашиваемые данные сразу же становятся доступными. При считывании ЭСППЗУ само ЦПУ приостанавливается на четыре цикла перед тем, как исполняется следующая команда.

Перед началом операции считывания пользователь должен опросить разряд EEWE. Если операция записи уже происходит, то невозможно ни считывание ЭСППЗУ, ни внесение изменений в регистр EEAR.

Для синхронизации доступов к ЭСППЗУ используется калибранный генератор. В таблице 3.1 приведено типичное время программирования для доступа к ЭСППЗУ из ЦПУ.

Таблица 3.1 - Время программирования ЭСППЗУ из ЦПУ

Символ	Количество циклов калиброванного RC-осциллятора	Типовое время программирования, мс
Запись в EEPROM	8448	8,4
Примечание - Используется тактовая частота 1 МГц независимо от установок конфигурационного бита CKSEL.		

Нижеприведенный пример 3.1 показывает одну функцию ассемблера и одну функцию С для записи в ЭСППЗУ. Этот пример означает, что прерывания контролируются (с помощью глобальной блокировки прерываний) таким образом, что во время исполнения этих функций не происходит прерываний; что в ПО не присутствует загрузчик флэш-памяти. Если подобный код присутствует, то функция записи ЭСППЗУ должна также дождаться завершения любой текущей команды SPM.

Пример 3.1

Assembly Code Example

```

EEPROM_write:
; Wait for completion of previous write
sblc EECR,EEWE
rjmp EEPROM_write
; Set up address (r18:r17) in address register
out EEARH, r18
out EEARL, r17
; Write data (r16) to Data Register
out EEDR,r16
; Write logical one to EEMWE
sbl EECR,EEMWE
; Start eeprom write by setting EEWE
sbl EECR,EEWE
ret

```

C Code Example

```

void EEPROM_write(unsigned Int uiAddress, unsigned char ucData)
{
/* Wait for completion of previous write */
while(EECR & (1<<EEWE))
;

/* Set up Address and Data Registers */
EEAR = uiAddress;
EEDR = ucData;
/* Write logical one to EEMWE */
EECR |= (1<<EEMWE);
/* Start eeprom write by setting EEWE */
EECR |= (1<<EEWE);
}

```

Следующий пример 3.2 кодов показывает функции ассемблера и С функции для считывания ЭСППЗУ. В этом примере предполагается, что прерывания контролируются таким образом, что во время исполнения данных не происходит прерываний.

Пример 3.2

Assembly Code Example

```

EEPROM_read:
; Wait for completion of previous write
sblc EECR,EEWE
rjmp EEPROM_read
; Set up address (r18:r17) in Address Register
out EEARH, r18
out EEARL, r17
; Start eeprom read by writing EERE
sbl EECR,EERE
; Read data from Data Register
In r16,EEDR
ret

```

C Code Example

```

unsigned char EEPROM_read(unsigned int uiAddress)
{
/* Wait for completion of previous write */
while(EECR & (1<<EEWE))
;

/* Set up Address Register */
EEAR = uiAddress;
/* Start eeprom read by writing EERE */
EECR |= (1<<EERE);
/* Return data from Data Register */
return EEDR;
}

```

3.2.3 Предотвращение ошибок в ЭСППЗУ

В периоды падения U_{CC} данные в ЭСППЗУ могутискажаться, поскольку напряжение питания слишком низко для того, чтобы обеспечить правильную работу ЦПУ и ЭСППЗУ. Подобные проблемы аналогичны тем, с которыми сталкиваются на уровне системных плат, и при этом должны использоваться аналогичные конструкторские решения.

Сбой данных ЭСППЗУ может происходить в двух случаях при слишком малом напряжении. Во-первых, нормальная последовательность записи в ЭСППЗУ требует, чтобы минимальное напряжение поддерживалось нормально. Во-вторых, само ЦПУ может неправильно исполнять команды при слишком малом напряжении питания. В то же время, можно избежать сбоя данных при работе ЭСППЗУ, если следовать следующей рекомендации:

Необходимо поддерживать общий сброс в активном (низком) состоянии во время низкого напряжения питания. Это можно выполнить путем включения внутреннего устройства обнаружения условий пониженного питания (Brown-out Detector) BOD. Если уровень обнаруженного внутреннего BOD не соответствует требуемому уровню обнаружения, то можно использовать внешнюю схему сброса для защиты от низкого $U_{#VCC}$. Если сброс происходит в период, когда идет процесс записи, то процесс записи будет завершен, когда восстановится достаточный уровень напряжения питания.

3.2.4 Память ввода-вывода (I/O Memory)

В пространстве ввода-вывода 1887ВЕ1У размещены все устройства ввода-вывода и периферийные устройства. Доступ к адресам ввода-вывода осуществляется с помощью команд IN и OUT путем передачи данных между 32 рабочими регистрами общего назначения и пространством ввода-вывода. Доступ к регистрам ввода-вывода внутри диапазона адресов 0x00 – 0x1F осуществляется путем прямого доступа к разрядам, используя команды SBI и CBI. В указанных регистрах значение отдельных разрядов можно проверить, используя команды SBIS и SBIC. При использовании специфических команд ввода-вывода IN и OUT, необходимо использовать адреса 0x00 – 0x3F ввода-вывода. При адресации к регистрам ввода-вывода как к пространству данных, используя команды LD и ST, к этим адресам необходимо добавлять 0x20.

Для обеспечения совместимости с будущими устройствами резервные разряды должны быть записаны в ноль, если к ним осуществляется доступ. В зарезервированные адреса ввода-вывода памяти никогда не должна проводиться запись.

Некоторые из флагов статуса очищаются путем записи в них логической единицы. Обратите внимание на то, что инструкции CBI и SBI должны работать на всех разрядах в Регистре ввода-вывода, записывая единицу в любой флаг, считанный как установленный, очищая флаг таким образом. Команды CBI и SBI работают только с регистрами с 0x00 по 0x1F.

Пояснения по регистрам ввода-вывода и регистрам управления периферийными устройствами даны в последующих разделах.

3.3 Система синхронизации

На рисунке 3.9 представлены основные системы синхронизации и их распределение. Не обязательно, чтобы в данный промежуток времени все тактовые импульсы были активными. Для снижения потребляемой мощности, подача тактовых импульсов к неиспользуемым модулям может быть приостановлена с помощью использования различных спящих режимов. Подробности в отношении систем синхронизации приведены ниже.

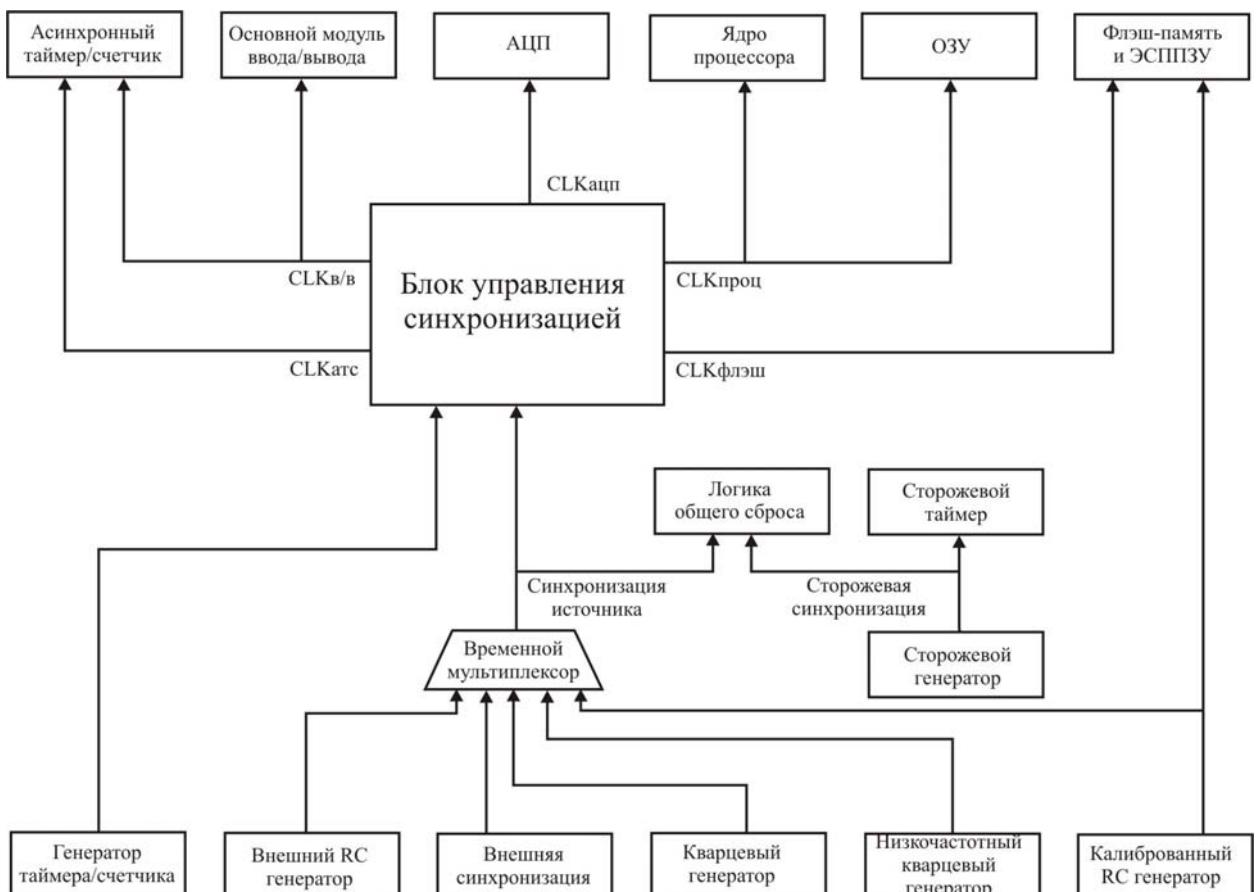


Рисунок 3.9 - Распределение тактовых импульсов

Тактовый сигнал ЦПУ – clk_{CPU}

Тактовый генератор ЦПУ подсоединен к узлам системы, связанным с работой ядра. Примерами таких модулей являются регистровый файл общего назначения, регистр состояний и память данных, имеющая указатель стека. Удержание тактового генератора ЦПУ блокирует выполнение ядром операций общего назначения и вычислений.

Тактовый сигнал ввода-вывода – $\text{clk}_{\text{I/O}}$

Тактовый генератор ввода-вывода используется основной частью модулей ввода-вывода, такими, как таймеры/счетчики, последовательно/параллельным интерфейсом и USART. Тактовый генератор ввода-вывода используется также модулем внешнего прерывания, при этом необходимо обратить внимание на то, что некоторые внешние прерывания обнаруживаются асинхронной логикой, позволяя обнаруживать подобные прерывания даже в случае остановки тактового генератора ввода-вывода. Необходимо обратить также внимание на то, что распознавание адреса в модуле TWI выполняется асинхронно при остановке $\text{clk}_{\text{I/O}}$, что позволяет обеспечить прием адреса TWI во всех спящих режимах.

Тактовый сигнал флэш-памяти – $\text{clk}_{\text{FLASH}}$

Тактовый генератор флэш-памяти $\text{clk}_{\text{FLASH}}$ управляет работой интерфейса. Обычно он активируется одновременно с тактовым генератором ЦПУ.

Тактовый сигнал асинхронного таймера - clk_{ASY}

Тактовый генератор асинхронного таймера clk_{ASY} обеспечивает непосредственную синхронизацию асинхронного счетчика/таймера с помощью внешнего кварцевого тактового генератора 32 кГц. Специально выделенный домен позволяет использовать этот таймер/счетчик в качестве счетчика реального времени, даже если устройство находится в спящем режиме.

Тактовый сигнал АЦП – clk_{ADC}

АЦП имеет специальный домен для тактового генератора, что позволяет выполнять остановку ЦПУ тактовых генераторов ввода-вывода для снижения шума, вызываемого цифровыми схемами. Это дает более точные результаты АЦП преобразований.

Источники тактового сигнала

Микроконтроллер имеет ряд опций источников тактового сигнала, выбираемых с помощью битов конфигурации согласно таблице 3.2. Тактовый импульс от выбранного источника вводится в тактовый генератор и перенаправляется в соответствующие модули.

Таблица 3.2 - Опции выбора синхронизирующего устройства

Опция выбора синхронизирующего устройства	CKSEL3...0
Внешний керамический/кварцевый резонатор	1111 - 1010
Внешний кварцевый генератор с низкой частотой	1001
Внешний RC генератор	1000 - 0101
Калибранный внутренний RC генератор	0100 - 0001
Внешний тактовый генератор	0000

Примечание - Для всех битов конфигурации “1” означает отсутствие программирования, в то время как “0” означает программирование.

Ниже представлены различные варианты выбора синхронизации для каждой опции. Когда ЦПУ начинает активироваться из режима пониженной мощности или режима энергосбережения, выбранный тип синхронизации используется для синхронизации запуска, обеспечивая при этом стабильную работу генератора до начала исполнения команд. Когда ЦПУ запускается с помощью команды RESET, возникает дополнительная задержка, позволяющая довести мощность до стабильного уровня к началу нормальной работы. Следующий генератор используется для синхронизации этого отрезка времени запуска. Количество циклов генератора WDT, используемых для каждого периода, показано в таблице 3.3. Частота работы следующего генератора зависит от напряжения.

Таблица 3.3 - Количество циклов следующего генератора WDT

Типовое время сброса, мс ($U_{\#VCC} = 5,0$ В)	Типовое время сброса, мс ($U_{\#VCC} = 3,0$ В)	Количество циклов
4,1	4,3	4K (4,096)
65	69	64K (65,536)

Источник тактовых импульсов по умолчанию

Значения по умолчанию бит CKSEL = “0001”, бит SUT = “10”. Вследствие этого начальная установка источника тактовых импульсов соответствует внутреннему RC генератору с самым длинным временем запуска. Эта установка гарантирует, что все пользователи могут выбирать желаемый источник тактового сигнала, используя внутрисистемное или параллельное программирование.

3.3.1 Тактовый генератор с внешним резонатором

Резонатор подключается к выводам BQ1 и BQ2 микроконтроллера, как показано на рисунке 3.10. Эти выводы являются соответственно входом и выходом инвертирующего усилителя тактового генератора.

Усилитель тактового генератора может работать в одном из двух режимов, определяемом состоянием конфигурационной ячейки СКОРТ . Если эта ячейка запрограммирована, размах колебаний на выходе усилителя (вывод BQ2) практически равен напряжению питания. Данный режим полезен при работе устройства в условиях сильных электромагнитных помех, а также при использовании сигнала тактового генератора для управления внешними устройствами.

Если ячейка СКОРТ не запрограммирована, размах колебаний на выходе усилителя будет значительно меньше. Соответственно ток потребления микроконтроллера уменьшается, однако при этом сужается и диапазон возможных частот тактового сигнала. В этом режиме сигнал тактового генератора микроконтроллера нельзя использовать для управления внешними устройствами.

Оптимальный номинал конденсаторов зависит от используемого кварца или резонатора, величины паразитной емкости и окружающих электромагнитных шумов. Определенные начальные рекомендации по выбору конденсаторов для использования совместно с кварцевыми резонаторами представлены в таблице 3.4. Для керамических резонаторов следует использовать номиналы конденсаторов согласно рекомендациям производителей резонаторов.

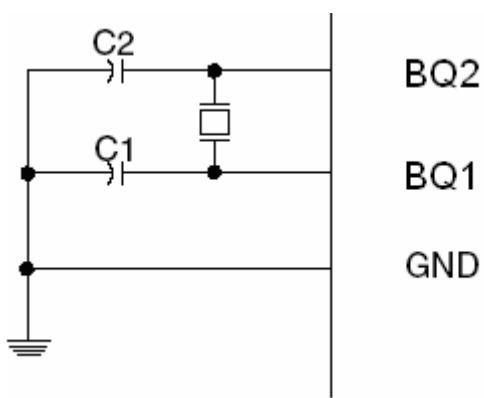


Рисунок 3.10 - Подключение кварцевого или керамического резонатора

Генератор может работать в трех различных режимах, каждый из которых оптимизирован для конкретного диапазона частот. Рабочий режим выбирается с помощью ячеек конфигурации CKSEL3...1, как показано в таблице 3.4.

Таблица 3.4 – Выбор режимов работы генератора

СКОРТ	CKSEL3...1	Частотный диапазон ¹⁾ , МГц	Рекомендуемые значения для конденсаторов C1 и C2 для использования с кварцевыми резонаторами, пФ
1	101 ²⁾	0,4 – 0,9	–
1	110	0,9 – 3,0	12 – 22
1	111	3,0 – 8,0	12 – 22
0	101, 110, 111	1,0 – 16,0	12 – 22

¹⁾ Значения для диапазонов частот являются предварительными.

²⁾ Данная опция не должна использоваться с кварцевыми, а только с керамическими резонаторами.

Таблица 3.5 - Времена запуска для выбора частоты кварцевого генератора

CKSEL0	SUT1...0	Время запуска от режима пониженного питания и режима энергосбережения	Период дополнительной задержки от переустановки, мс ($U_{CC} = 5,0$ В)	Рекомендуемое применение
0	00	258 CK ¹⁾	4,1	керамический резонатор, быстро увеличивающаяся мощность
0	01	258 CK ¹⁾	65	керамический резонатор, медленно увеличивающаяся мощность
0	10	1K CK ²⁾	–	керамический резонатор, BOD разрешен
0	11	1K CK ²⁾	4,1	керамический резонатор, быстро увеличивающаяся мощность
1	00	1K CK ²⁾	65	керамический резонатор, медленно увеличивающаяся мощность
1	01	16K CK	–	керамический резонатор, BOD разрешен
1	10	16K CK	4,1	керамический резонатор, быстро увеличивающаяся мощность
1	11	16K CK	65	керамический резонатор, медленно увеличивающаяся мощность

¹⁾ Эти опции должны использоваться только в том случае, когда работа выполняется не на границе предельной частоты работы прибора, и если стабильность по частоте при запуске не играет роли для данного применения. Эти опции не пригодны для использования с кварцевыми резонаторами.

²⁾ Эти опции предназначены для использования с керамическими резонаторами и обеспечивают стабильность частоты при запуске. Их также можно использовать с кварцами, если не применять при работе на частоте, близкой к предельной, и если стабильность по частоте при запуске не важна для данного вида применения.

Перемычка CKSEL0 вместе с перемычками SUT1...0 выбирает временные интервалы для запуска, как это показано в таблице 3.5.

3.3.2 Кварцевый генератор низкой частоты

Для использования следящего кристалла с частотой 32,768 кГц в качестве основного тактового генератора необходимо выбирать низкочастотный кварцевый генератор с помощью установки плавких перемычек CKSEL в состояние - frequency “1001”. Кварцевый генератор следует подсоединять в соответствии с рисунком 3.12. С помощью программирования перемычки СКОРТ пользователь может подключить внутренние емкости к BQ1 и BQ2, устранив тем самым необходимость использования внешних конденсаторов. Внутренние конденсаторы имеют номинал 36 пФ.

При выборе такого генератора величина времени запуска определяется выбором перемычек SUT, как это показано в таблице 3.6.

Таблица 3.6 - Времена запуска для выбора низкочастотного кварцевого тактового генератора

SUT1...0	Время запуска из состояния режима пониженной мощности и энергосберегающего режима	Дополнительная задержка из состояния переустановки, мс ($U_{CC} = 5,0$ В)	Рекомендуемый вид применения
00	1K CK*	4,1	Быстро возрастающая мощность или разрешено BOD
01	1K CK*	65	Медленно возрастающая мощность
10	32K CK	65	Стабильная частота при запуске
11		Зарезервировано	
* Эти опции следует использовать, только если стабильность по частоте при запуске не важна для данного вида применения.			

3.3.3 Внешний RC генератор

Внешняя конфигурация RC - генератора, показанная на рисунке 3.11, предназначена для применений, не чувствительных к частоте. Частоту можно приблизительно оценить по формуле $f = 1/(3RC)$. Величина С должна быть, по меньшей мере, 22 пФ. Выполняя программирование с помощью перемычки СКОРТ, пользователь может подключить внутреннюю емкость в 36 пФ между BQ1 и GND, устранивая тем самым потребность во внешней емкости.

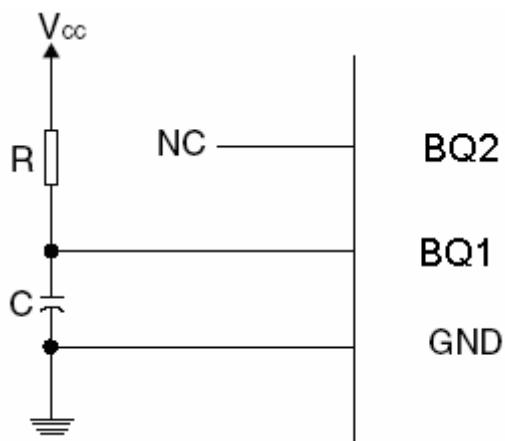


Рисунок 3.11 - Электрическая схема подключения внешнего RC генератора

Генератор может работать в четырех различных режимах, каждый из которых оптимизирован для определенного диапазона частот. Рабочий режим выбирается с помощью перемычек CKSEL3...0, как это показано в таблице 3.7.

Таблица 3.7 - Рабочие режимы внешнего RC генератора

CKSEL 3...0	Частотный диапазон, МГц
0101	0,4 ÷ 0,9
0110	0,9 ÷ 3,0
0111	3,0 ÷ 8,0
1000	8,0 ÷ 12,0

При выборе этого генератора времена запуска определяются перемычками SUT, как это показано в таблице 3.8.

Таблица 3.8 - Времена запуска для выбора внешнего RC генератора

SUT1...0	Время запуска из состояния режима пониженной мощности и энергосберегающего режима	Дополнительная задержка из состояния переустановки, мс ($U_{\#VCC} = 5,0$ В)	Рекомендуемый вид применения
00	18CK	—	BOD разрешено
01	18CK	4,1	Быстро возрастающая мощность
10	18CK	65	Медленно возрастающая мощность
11	6CK ¹⁾	4,1	Быстро возрастающая мощность или разрешено BOD

¹⁾ Эта опция не должна использоваться при работе, близкой к максимальной частоте прибора.

3.3.4 Калируемый внутренний RC генератор

Калируемый внутренний RC генератор задает фиксированную тактовую частоту 1,0; 2,0; 4,0 или 8,0 МГц. Все частоты являются номинальными значениями при 5 вольтах и 25 °C. Этот генератор может быть выбран в качестве системного с помощью программирования бит CKSEL, как это показано в таблице 3.9. Если он выбран, то он может работать без внешних компонентов. При использовании этой опции перемычка СКОРТ всегда должна быть не запрограммирована. Во время сброса, аппаратное обеспечение загружает байт для калибровки в регистр OSCCAL и таким образом автоматически калибрует RC генератор. При 5 вольтах, 25 °C и выбранной частоте генератора 1,0 МГц такая калибровка обеспечивает отклонение частоты в пределах ± 1 % от номинальной. При использовании этого генератора в качестве генератора для кристалла следящий генератор будет по-прежнему использоваться для следящего таймера и для переустановки времени истечения ожидания.

Таблица 3.9 - Рабочие режимы RC генератора с внутренней калибровкой

CKSEL3...0	Номинальная частота, МГц
0001*	1,0
0010	2,0
0011	4,0
0100	8,0

* Прибор поставляется с уже активированной данной опцией.

Если выбирается этот генератор, то времена запуска определяются перемычками SUT, как это показано в таблице 3.10. BQ1 и BQ2 следует оставить не подсоединенными к (NC).

Таблица 3.10 - Времена запуска для выбора RC генератора с внутренней калибровкой

SUT1...0	Время запуска из состояния режима пониженной мощности и энергосберегающего режима.	Дополнительная задержка из состояния переустановки, мс (U _{#VCC} = 5,0 В)	Рекомендуемый вид применения
00	6CK	–	BOD разрешено
01	6CK	4,1	быстро возрастающая мощность
10*	6CK	65	медленно возрастающая мощность
11	Зарезервировано		

* Прибор поставляется с уже выбранной данной опцией.

Регистр калибровки генератора – OSCCAL

Бит	7	6	5	4	3	2	1	0	OSCCAL
Чтение/Запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	Значение калибровки для данного прибора								

Разряды 7...0: CAL7...0 - Значение калибровки генератора

Запись байта калибровки в этот адрес скорректирует внутренний генератор для устранения изменений в процессе в зависимости от частоты. Во время переустановки значение калибровки 1 МГц, расположенное в верхнем байте строки сигнатур (адрес 0x00), автоматически загружается в регистр OSCCAL. Если на других частотах используется внутренний RC генератор, то значения калибровки должны загружаться вручную. Это можно реализовать, сначала прочитав строку сигнатур с помощью программирующего устройства, а затем сохранить значения калибровки во флэш-памяти или ЭСППЗУ. Затем данное значение можно считать программным способом и загрузить в регистр OSCCAL.

Если OSCCAL находится в состоянии ноль, то выбирается самая низкая из частот, имеющихся в наличии. Запись не нулевых значений в этот регистр увеличит частоту внутреннего генератора. Запись 0xFF в регистр даст самую высокую из имеющихся в наличии частот. Для синхронизации доступа в ЭСППЗУ и флэш-память используется калиброванный генератор. Если выполняется запись в ЭСППЗУ и флэш-память, то не следует выполнять калибровку при значениях частоты, более 10 % от номинальной. В противном случае может не произойти запись в ЭСППЗУ и флэш-память. Необходимо обратить внимание на то, что генератор предназначен для калибровки при 1,0; 2,0; 4,0 или 8,0 МГц. Как видно из таблицы 3.11 настройка для других значений не гарантируется.

Таблица 3.11 - Диапазон частот для внутреннего RC генератора.

Значение OSCCAL	Минимальная частота в процентах от номинальной частоты, %	Максимальная частота в процентах от номинальной частоты, %
0x00	50	100
0x7F	75	150
0xFF	100	200

3.3.5 Внешний тактовый генератор

Для возбуждения прибора от источника внешнего генератора BQ1 должен запускаться, как это показано на рисунке 3.12. Для работы прибора на внешний тактовый генератор необходимо запрограммировать плавкие перемычки CKSEL на "0000". С помощью приемов программирования перемычки СКОРТ пользователь может подключить внутреннюю емкость 36 пФ между BQ1 и GND.

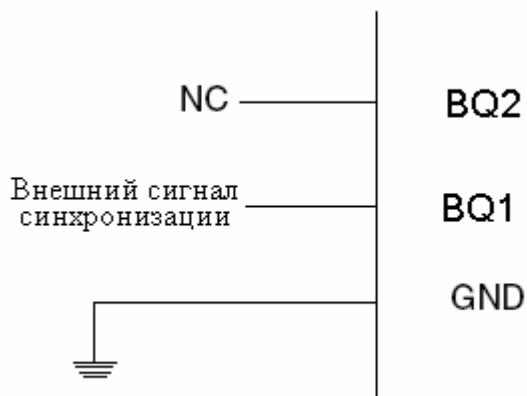


Рисунок 3.12 - Конфигурация при возбуждении от внешнего генератора

Таблица 3.12 - Времена запуска для выбора внешнего синхронизирующего генератора

SUT1...0	Время запуска из состояния режима пониженной мощности и энергосберегающего режима	Дополнительная задержка из состояния переустановки, мс ($U_{\#VCC} = 5,0 \text{ В}$)	Рекомендуемый вид применения
00	6СК	—	BOD разрешено
01	6СК	4,1	быстро возрастающая мощность
10	6СК	65	медленно возрастающая мощность
11	Зарезервировано		

При выборе этого источника синхронизации времена запуска определяются переключками SUT, как это показано в таблице 3.12. При приложении внешнего тактового генератора необходимо избегать внезапных изменений в прилагаемой тактовой частоте для обеспечения стабильной работы МПУ (микропроцессорное устройство). Изменения в частоте более чем на 2 % от одного тактового цикла до другого могут вызвать непредсказуемое поведение. Необходимо обеспечить, чтобы во время подобных изменений тактовой частоты МПУ находилось в состоянии сброса.

3.3.6 Генератор таймера/счетчика

Для микроконтроллеров, имеющих выводы генератора таймера/счетчика (TOSC1 и TOSC2), кварцевый генератор подсоединяется непосредственно между выводами, при этом не требуется никаких внешних конденсаторов. Генератор оптимизирован для работы с кварцевым резонатором для часов с частотой 32,768 кГц. Использование внешнего синхронизирующего источника для TOSC1 не рекомендуется.

3.4 Энергосберегающие режимы

Использование спящих режимов позволяет отключать неиспользуемые модули в микроконтроллере, благодаря чему экономится мощность. 1887ВЕ1У обеспечивает различные спящие режимы, которые позволяют пользователю приспособливать потребление мощности для различных видов применения.

Для введения любого из шести спящих режимов, в MCUCR необходимо записать разряд SE в логическую единицу и выполнить команду SLEEP. С помощью разрядов SM2, SM1 и SM0 в регистре MCUCR выбирается, какой из спящих режимов (Idle (режим холостого хода), ADC Noise Reduction (снижение шума АЦП), Power-down (режим пониженной мощности), Power-save (режим энергосбережения), Standby (дежурный режим) или Extended Standby (длительный дежурный режим)) будет активирован с помощью команды SLEEP. Сводные сведения вы можете увидеть в таблице 3.13. Если произойдет разрешенное прерывание, когда МПУ находится в спящем режиме, то МПУ просыпается. Затем МПУ приостанавливается на четыре цикла в дополнение ко времени запуска, выполняет процедуру прерывания и возобновляет исполнение с команды, которая следует за командой SLEEP. Содержимое файла регистра и СОЗУ не изменяется, когда устройство пробуждается ото сна. Если сброс происходит во время спящего режима, то МПУ пробуждается и начинает исполнение от вектора сброса.

Регистр управления МПУ – MCUCR

Регистр управления МПУ содержит разряды управления для регулировки мощности.

Бит	7	6	5	4	3	2	1	0	MCUCR
Чтение/запись	R/W								
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7, 5, 4: SM2...0 - Спящий режим выбирает Разряды 2, 1 и 0

С помощью этих разрядов выбирается один из шести спящих режимов, как это показано в таблице 3.13.

Таблица - 3.13. Выбор спящего режима

SM2	SM1	SM0	Спящий Режим
0	0	0	Режим холостого хода
0	0	1	Снижение шума АЦП
0	1	0	Пониженная мощность
0	1	1	Режим энергосбережения
1	0	0	Зарезервировано
1	0	1	Зарезервировано
1	1	0	Дежурный режим*
1	1	1	Длительный дежурный режим*

* Дежурный режим и длительный дежурный режим могут быть организованы только с использованием внешних кварцевых генераторов или резонаторов.

Разряд 6: SE - Разрешение спящего режима

Разряд SE должен быть записан в логическую единицу для того, чтобы МПУ вошло в спящий режим во время исполнения команды SLEEP. Чтобы избежать перехода МПУ в спящий режим, если только это не предусмотрено программой, рекомендуется записать разряд Sleep Enable (SE) – разрешение на спящий режим в единицу непосредственно перед исполнением команды SLEEP и сразу же очистить ее после пробуждения.

3.4.1 Режим холостого хода

Если разряды SM2...0 записываются в 000, то команда SLEEP заставляет МК переходить в режим холостого хода, останавливая ЦПУ, но разрешая при этом, чтобы SPI, USART, аналоговый компаратор, АЦП, двухпроводной последовательный интерфейс, таймеры/счетчики, сторожевой таймер и система прерывания продолжали работать. Подобный спящий режим в основном останавливает clk_{CPU} и $\text{clk}_{\text{FLASH}}$, позволяя при этом работать другим тактовым генераторам.

Режим холостого хода разрешает МК пробудиться с помощью запущенных извне прерываний, а также внутренних прерываний, таких как переполнение таймера и завершение передачи USART. Если пробуждения от прерывания аналогового компаратора не требуется, то мощность аналогового компаратора можно понизить путем установки разряда АЦП в управлении аналоговым компаратором и регистре состояний – ACSR. Это позволит снизить потребляемую мощность в режиме холостого хода. Если АЦП активирован, то преобразование начнется автоматически при запуске этого режима.

3.4.2 Режим снижения шума АЦП

Если разряды SM2...0 записываются в 001, то команда SLEEP переводит МК в режим снижения шума АЦП, останавливая при этом ЦПУ, но разрешая продолжение работы АЦП, внешних прерываний, двухпроводного последовательного интерфейса слежения за адресом, таймера/счетчика 2 и устройства слежения (при разрешении). Этот спящий режим в основном останавливает $\text{clk}_{\text{I/O}}$, clk_{CPU} и $\text{clk}_{\text{FLASH}}$, позволяя при этом работать другим тактовым генераторам.

Это уменьшает внешние шумы АЦП, разрешая выполнять измерения с повышенной разрешающей способностью. Если АЦП включен, то преобразование начинается автоматически при вводе данного режима. Кроме прерывания полного преобразования АЦП, только внешний сброс, сброс устройства слежения, сброс понижения мощности (Brown-out Reset) и прерывание соответствия адреса двухпроводного последовательного интерфейса, прерывание таймера/счетчика 2, прерывание готовности SPM/EEPROM, прерывание внешнего уровня на INT0 или INT1, или внешнее прерывание на INT2 могут вывести МПУ из режима понижения шума АЦП.

3.4.3 Режим микропотребления

Если разряды SM2...0 записываются в “010”, то команда SLEEP переводит МК в режим микропотребления. В этом режиме внешний генератор останавливается, в то время как внешние прерывания, слежение за соответствием адреса двухпроводного последовательного интерфейса и устройство слежения продолжают работать (если это разрешено). Только внешний сброс, сброс следящего устройства, сброс в режим понижения мощности (Brown-out Reset), прерывание соответствия адреса двухпроводного последовательного интерфейса, прерывание внешнего уровня для INT0 или INT1 или внешнее прерывание для INT2 могут активировать МК. Этот спящий режим в основном блокирует

ет все генерируемые синхронизирующие сигналы, разрешая работать только асинхронным модулям.

Обратите внимание на то, что если прерывание, срабатывающее от уровня, используется для активации из режима пониженной мощности, то измененный уровень должен удерживаться в течение некоторого времени для того, чтобы вызвать активацию МК.

При активации из режима пониженной мощности имеет место задержка, пока активация становится эффективной. Это позволяет перезапустить тактовый генератор и стабилизировать его работу после остановки. Период пробуждения определяется теми же плавкими перемычками CKSEL, которые определяют период переустановки времени перерыва в работе.

3.4.4 Режим энергосбережения

Если “011” записывается в разряды SM2...0, то команда SLEEP заставляет МК перейти в режим энергосбережения. Этот режим аналогичен режиму пониженной мощности, за одним исключением: если таймер/счетчик 2 тактируются асинхронно, т. е. устанавливается разряд AS2 в ASSR, то таймер/счетчик 2 будет продолжать работать во время «сна». Прибор может пробудиться либо от события переполнения таймера или сравнения выхода от таймера/счетчика 2, если в TIMSK установлены соответствующие разряды разрешения прерывания, а в SREG установлен разряд разрешения глобального прерывания.

Если асинхронный таймер не тактируется асинхронно, то вместо режима энергосбережения рекомендуется режим пониженной мощности, поскольку содержимое регистров в асинхронном таймере должно рассматриваться как неопределенное после активации в режиме энергосбережения, если AS2 равно 0.

Этот режим ожидания в основном приостанавливает все тактовые импульсы, за исключением clk_{ASY} , разрешая работу только асинхронных модулей, включая таймер/счетчик 2, если тактирование выполняется асинхронно.

3.4.5 Дежурный режим

Если разрядами SM2...0 являются “110” и выбирается опция внешнего кварцевого генератора/резонатора, то команда SLEEP переводит МК в дежурный режим. Этот режим аналогичен режиму пониженного питания, за исключением того, что генератор продолжает работать. Устройство активируется из режима Standby в течение шести тактовых циклов.

3.4.6 Длительный дежурный режим

Если разрядами SM2...0 являются “111” и выбирается опция внешнего кварцевого генератора/резонатора, то команда SLEEP переводит МК в длительный дежурный режим. Этот режим аналогичен энергосберегающему режиму, за исключением того, что генератор продолжает работать. Устройство активируется из режима Standby (Дежурный Режим) в течение шести тактовых циклов.

Таблица 3.14 - Активные домены тактового генератора и источники активации в различных режимах спящего состояния

Режим ожидания	Активные временные домены					Излучатели		Источники активации						
	clk _{CPU}	clk _{FLASH}	clk _{IO}	clk _{ADC}	clk _{ASY}	Основной источник синхронизации активен	Генератор таймера активен	INT2	INT1	INT0	TWI	TIMER	SPM	ADC
Холостой ход			X	X	X	X	X ²⁾	X	X	X	X	X	X	X
Снижение шума АЦП				X	X	X	X ²⁾	X ³⁾	X	X	X	X	X	
Отключение питания								X ³⁾	X					
Энерго-сбережение						X ²⁾		X ²⁾	X ³⁾	X	X ²⁾			
Резервный ¹⁾							X		X ³⁾	X				
Расширенный резервный ¹⁾					X ²⁾	X	X ²⁾	X ³⁾	X	X ²⁾				

¹⁾ В качестве источника тактовых сигналов выбирается внешний кварцевый генератор или резонатор.

²⁾ Если установлен разряд AS2 в ASSR.

³⁾ Только INT2 или прерывание уровня INT1 и INT0.

Минимизация потребления мощности

При попытке свести к минимуму потребляемую мощность в системе, контролируемой процессором, следует рассмотреть несколько проблем. В целом целесообразно как можно чаще использовать спящие режимы. Спящий режим необходимо выбирать таким образом, чтобы в работе участвовало как можно меньше функций прибора. Все функции, в которых нет необходимости, должны быть отключены. Особое внимание следует уделить следующим модулям для достижения минимального потребления мощности.

Аналогово - цифровой преобразователь

При активации, АЦП будет задействован во всех спящих режимах. Для сбережения мощности АЦП должен быть отключен перед входом в любой спящий режим. Если АЦП выключается и затем снова включается, то следующее преобразование будет представлять собой расширенное преобразование.

Аналоговый компаратор

Если аналоговый компаратор не используется, то при переходе в режим холостого хода он должен быть отключен. Если АЦП вводится в режим пониженного шума, то Аналоговый компаратор должен быть отключен. В других спящих режимах аналоговый компаратор отключается автоматически. Однако если аналоговый компаратор настраивается для использования в качестве входа внутреннего опорного напряжения, то аналоговый компаратор должен быть отключен во всех спящих режимах. В других случаях, внутреннее опорное напряжение будет включено независимо от спящего режима.

Детектор отключения питания

Если детектор отключения питания не нужен для работы, то этот модуль должен быть отключен. Если детектор включается с помощью конфигурационного бита BODEN, то он будет задействован во всех спящих режимах и, следовательно, всегда потребляет мощность. В более глубоких спящих режимах это внесет существенный вклад в общий потребляемый ток.

Внутреннее опорное напряжение

Внутреннее опорное напряжение будет включено, когда это необходимо для детектора отключения питания, аналогового компаратора, или АЦП. Если эти модули отключаются, как это описано в разделах выше, то внутреннее опорное напряжение будет отключено и мощность не будет потребляться. При повторном включении пользователь должен разрешить запуск опорного напряжения перед тем, как будет использован выход. Если опорное напряжение поддерживается в спящем режиме, то выход можно использовать немедленно.

Сторожевой таймер

Если для данного вида применения сторожевой таймер не нужен, то этот модуль следует отключить. Если сторожевой таймер включен, то он будет включен во всех спящих режимах и, следовательно, будет всегда потреблять мощность. В более глубоких спящих режимах это будет вносить существенный вклад в суммарное потребление мощности.

Выходы портов

При переходе в спящий режим все выводы портов должны получить такую конфигурацию, для которой потребление мощности минимально. В этом случае самым важным является то, что выводы не возбуждают резистивные нагрузки. В спящих режимах, когда оба тактовых импульса вход/выход ($\text{clk}_{\text{I/O}}$) и АЦП (clk_{ADC}) останавливаются, входные буферные устройства прибора отключаются. Тем самым гарантируется отсутствие потребления мощности логическими входами, когда в этом нет необходимости. В некоторых случаях логика для входов необходима для обнаружения условий пробуждения и в таком случае она будет активирована. Если активируется буфер входа и входной сигнал остается в плавающем состоянии или если уровень аналогового сигнала близок к $U_{\text{CC}}/2$, то входной буфер будет потреблять избыточную мощность.

3.5 Сброс и управление системой

Во время сброса все РВВ устанавливаются на свои первоначальные значения, и программа начинает исполнение от вектора сброса. Команда, размещаемая в векторе сброса, должна являться командой RJMP для сброса рабочей процедуры. Если программа никогда не разрешает источник прерывания, то векторы прерывания не используются, и в этих адресах может быть размещен обычный программный код. То же самое происходит, если вектор сброса находится в секции приложения, в то время как векторы прерывания находятся в секции загрузки или наоборот. На схемной диаграмме рисунка 3.13 показана логика для сброса. В таблице 3.15 определены электрические параметры схемы сброса. Порты ввода-вывода контроллера немедленно переустанавливаются в свое исходное состояние при активации источника сброса. При этом не требуется, чтобы работал какой-либо из источников тактового сигнала. После того как все источники сброса стали неактивными, запускается счетчик задержки, растягивая внутренний сброс. Это позволяет достичь стабильного уровня мощности до начала нормальной работы. Период выключения счетчика задержки определяется пользователем с помощью перемычек CKSEL.

1887ВЕ1У имеет четыре источника сброса:

- Сброс включения питания. МК сбрасывается, если напряжение питания ниже порога сброса включения питания (V_{POT}).
- Внешний сброс. МК сбрасывается, если присутствует низкий уровень на выводе RESET# в течение времени, превышающего минимальную длину импульса.
- Сброс сторожевого таймера. МК сбрасывается по истечению периода сторожевого таймера и когда сторожевой таймер активируется.
- Сброс отключения питания. МК сбрасывается, если напряжение питания U_{CC} ниже, чем порог сброса внезапного отключения (V_{BOT}), и устройство обнаружения отключения активировано.

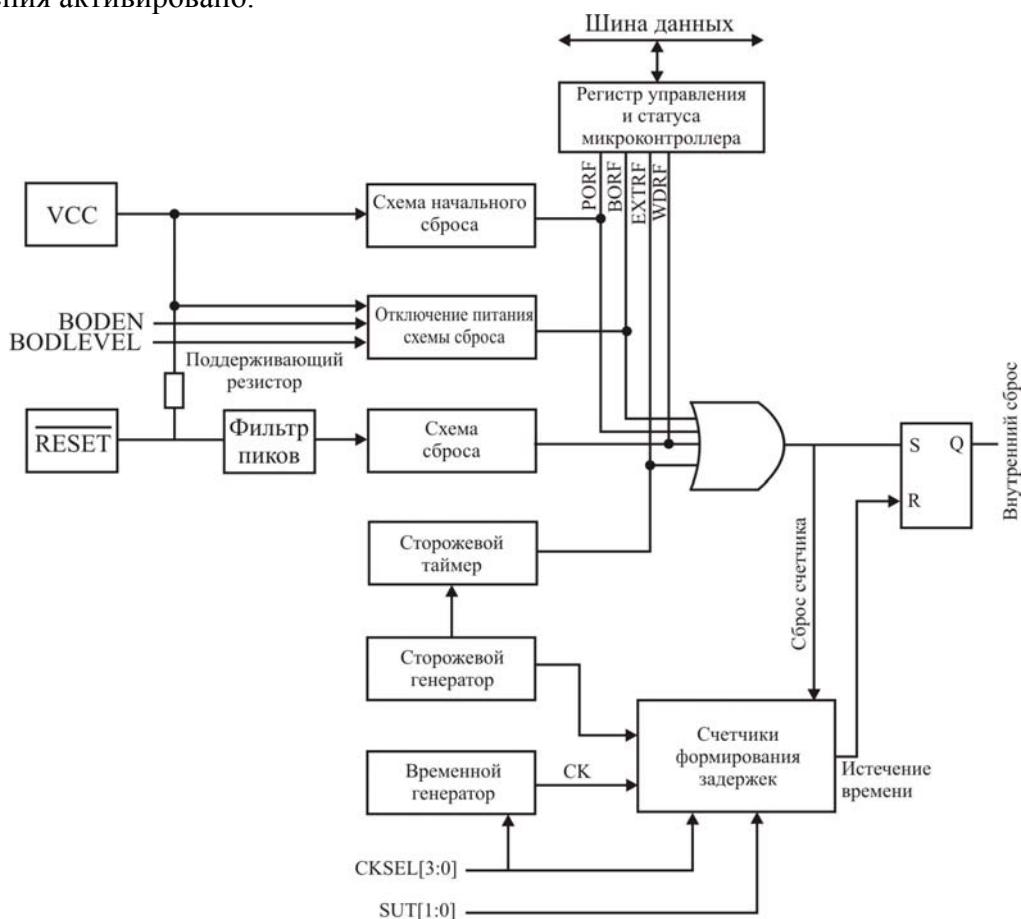


Рисунок 3.13 - Логика сброса

Таблица 3.15 - Характеристики переустановки*

Символ	Параметр	Условие	Мин	Тип	Макс	Единица измерения
U_{POT}	Начальный сброс, пороговый уровень (передний фронт)			1,4	2,3	V
	Начальный сброс, пороговый уровень (задний фронт)			1,3	2,3	V
U_{RST}	Пороговое напряжение на выводе общего сброса		0,1		0,9	V
t_{RST}	Минимальная длина импульса сброса			50		нс
U_{BOT}	Пороговое напряжение выключения сброса	BODLEVEL=1	2,5	2,7	3,2	V
		BODLEVEL=0	3,7	4,0	4,2	
t_{BOD}	Минимальная длительность подачи низкого напряжения для отключения	BODLEVEL=1		2		мкс
		BODLEVEL=0		2		мкс
U_{HYST}	Задержка отключения			130		мВ
Примечания						
1 Сброс включения мощности не будет срабатывать, пока напряжение питания не снизится до величины ниже U_{POT} (спадающее).						
2 Для некоторых приборов U_{BOT} может оказаться ниже минимального рабочего напряжения. Для тех приборов, для которых это имеет место, во время технологического контроля измерения производятся вплоть до $U_{CC} = U_{BOT}$. Тем самым гарантируется, что сброс при внезапном отключении питания произойдет до того, как U_{CC} упадет до уровня напряжения, при котором правильная работа микроконтроллера не может быть больше гарантирована.						
* Данные значения являются справочными.						

3.5.1 Сброс по включению питания

Импульс для сброса включения питания (POR) генерируется схемой детектирования, встроенной в кристалл. Уровень обнаружения определяется по таблице 3.15. Импульс POR активируется в любом случае, когда $U_{\#VCC}$ будет располагаться ниже уровня обнаружения. Схема POR может быть использована для запуска сброса, а также для обнаружения проблем с напряжением питания. Схема сброса по включению питания (POR) обеспечивает сброс прибора при включении питания. Когда напряжение сброса по включению питания достигает порогового значения, то при этом запускается счетчик задержки, который определяет, сколько времени прибор будет находиться в состоянии RESET после повышения значения $U_{\#VCC}$. Сигнал RESET активируется вновь, причем без задержки, если $U_{\#VCC}$ снижается ниже уровня обнаружения.

На рисунках 3.14 – 3.18 $U_{\#VCC} = U_{CC}$.

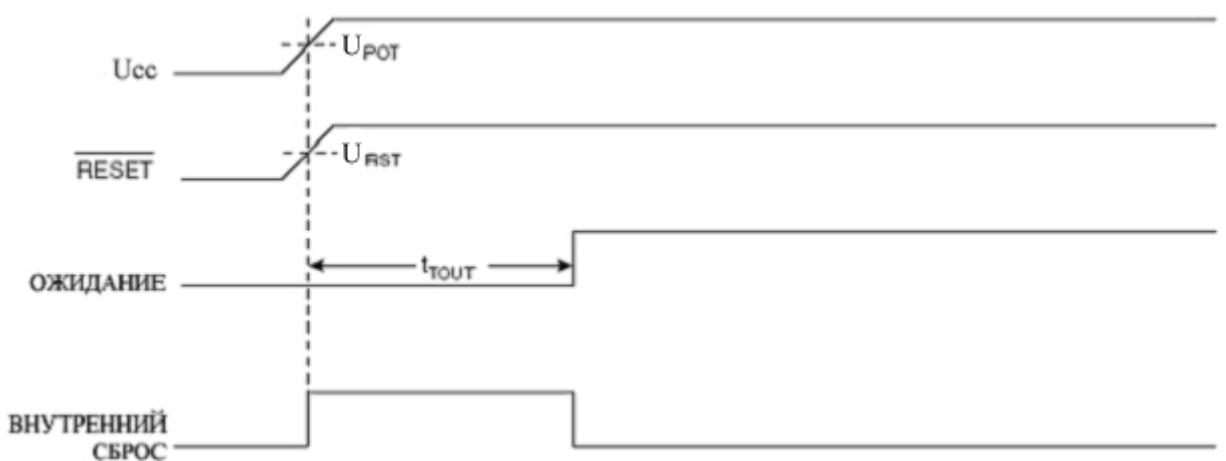


Рисунок 3.14 - Запуск МК, команда RESET связана с уровнем $U_{\#VCC}$

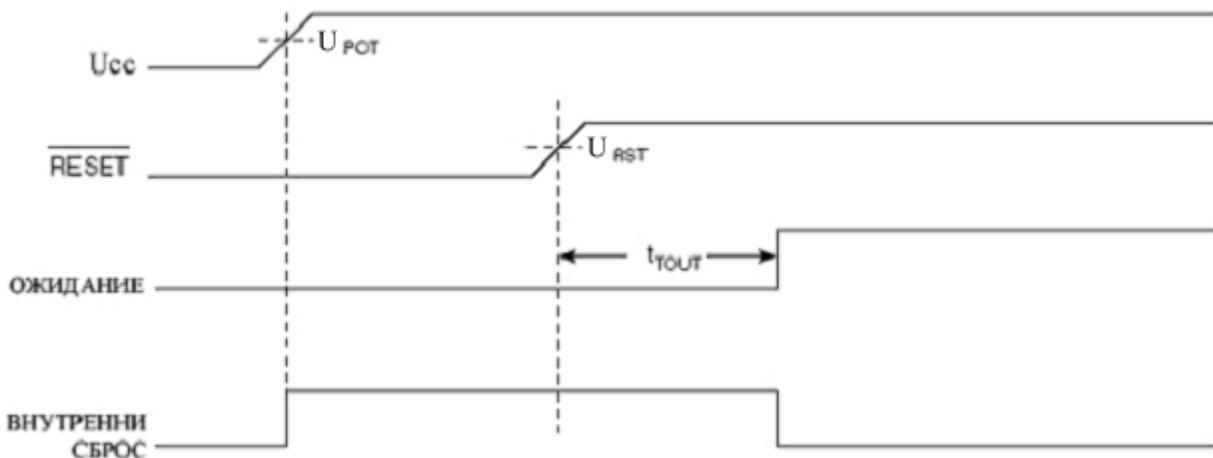


Рисунок 3.15 - Запуск МКУ, команда RESET продлена с помощью внешнего воздействия

3.5.2 Внешний сброс

Внешний сброс запускается при низком уровне на выводе RESET#. Импульсы сброса, превышающие минимальную ширину импульса (см. таблицу 3.15), активируют сброс даже, если тактовый генератор не работает. При более коротких импульсах не гарантируется активация сброса. Когда приложенный сигнал достигает уровня порогового напряжения сброса, – U_{RST} на своем положительном фронте, то счетчик задержки запускает МКУ после окончания периода прерывания t_{WAIT} .

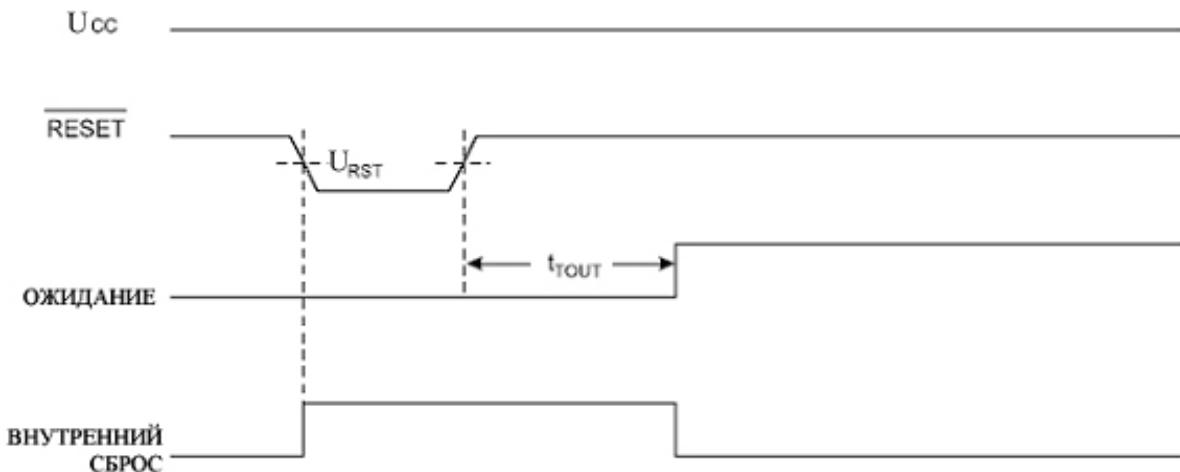


Рисунок 3.16 - Внешняя переустановка во время работы

3.5.3 Детектирование отключения питания

1887ВЕ1У имеет встроенную схему детектирования отключения питания (BOD) для контроля уровня $U_{\#VCC}$ во время работы, что реализуется путем его сравнения с фиксированным уровнем триггера. Уровень триггера для BOD можно выбрать с помощью перемычки BODLEVEL, чтобы он был равен 2,7 вольта (BODLEVEL непрограммируемый (unprogrammed) или 4,0 вольта (BODLEVEL программируемый (programmed)). Уровень триггера имеет гистерезис для обнаружения отключения питания без наличия выбросов напряжения. Гистерезис для уровня обнаружения должен интерпретироваться как

$$U_{BOT+} = U_{BOT} + U_{HYST}/2 \text{ и } U_{BOT-} = U_{BOT} - U_{HYST}/2.$$

Схему BOD можно подключить/отключить с помощью перемычки BODEN. Когда BOD включена, (BODEN программируемый), и $U_{\#VCC}$ понижается до значения ниже уровня триггера, сразу же включается сброс отключения питания. При возрастании U_{CC} выше уровня триггера, (U_{BOT+} на рисунке 3.17), счетчик задержки запускает МК после истечения периода отключения t_{TOUT} . Схема BOD обнаружит падение напряжения $U_{\#VCC}$ только в том случае, если напряжение остается ниже уровня триггера не более чем в течение периода t_{BOD} , указанного в таблице 3.15.

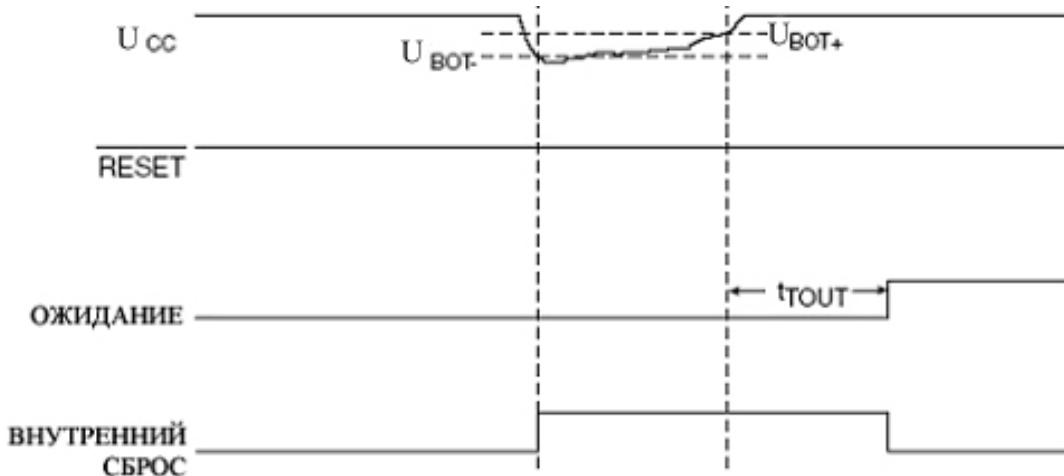


Рисунок 3.17 - Переустановка внезапного отключения питания во время работы

3.5.4 Сброс сторожевого таймера

Если переполняется счетчик сторожевого таймера, схема будет генерировать короткий импульс сброса длительностью в один цикл. На заднем фронте этого импульса счетчик задержки начинает отсчитывать период временного отключения t_{TOUT} .

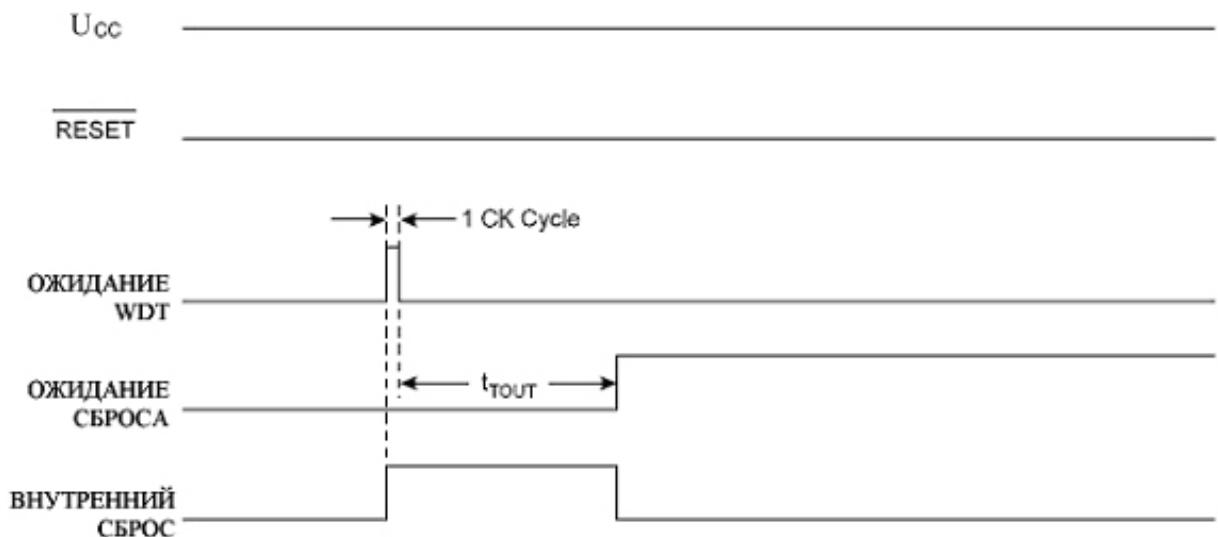


Рисунок 3.18 - Переустановка следящего устройства во время работы

Контроль и состояние МК

Регистр – MCUCSR

Регистр управления и состояния МК выдает информацию о том, какой из источников переустановки вызвал переустановку МК.

Бит	7	6	5	4	3	2	1	0	
	-	SE	-	-	WDRF	BORF	EXTRF	PORF	MCUCSR
Чтение/запись	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 3: WDRF - Флаг сброса сторожевого таймера

Этот разряд устанавливается в том случае, если происходит сброс сторожевого таймера. Разряд сбрасывается с помощью сброса по включению питания или путем записи логического нуля во флаг.

Разряд 2: BORF - Флаг сброса отключения питания

Этот разряд устанавливается в том случае, если происходит сброс при отключении питания. Разряд сбрасывается с помощью сброса по включению питания или путем записи логического нуля во флаг.

Разряд 1: EXTRF - Флаг внешнего сброса

Этот разряд устанавливается в том случае, если происходит внешний сброс. Разряд сбрасывается с помощью сброса по включению питания или путем записи логического нуля во флаг.

Разряд 0: PORF - Флаг сброса по включению питания

Этот разряд устанавливается в том случае, если происходит сброс по включении питания. Разряд сбрасывается только с помощью записи логического нуля во флаг.

Для того, чтобы использовать флаги сброса для идентификации условия сброса, пользователь должен сосчитать, а затем сбросить MCUCSR в программе как можно раньше. Если регистр очищается до того, как происходит другой сброс, источник сброса можно найти путем анализа флагов сброса.

Внутреннее опорное напряжение

ИМС 1887ВЕ1У имеет отличительную особенность в виде внутреннего эталонного (опорного) генератора. Этот опорный уровень используется для обнаружения отключения питания, и он может быть использован в качестве входа для аналогового компаратора или АЦП. Опорный уровень с величиной 2,56 В для АЦП генерируется внутренним эталонным генератором.

Разрешающие сигналы опорного напряжения и время запуска

Опорное напряжение имеет время установки, которое может повлиять на способ его использования. Время запуска приведено в таблице 3.16. С целью экономии энергии опорное напряжение не всегда включается. Опорное напряжение включается в следующих случаях:

- 1 Когда запускается BOD (путем программирования перемычки BODEN).
- 2 Когда полоса опорного напряжения подсоединенена к аналоговому компаратору (с помощью установки разряда ACBG в ACSR).

3 Когда работает АЦП.

Так, если BOD не активирован, то после установки разряда ACBG или активации АЦП, пользователь всегда должен разрешить запуск опорного напряжения до того, как будет использован выход аналогового компаратора или АЦП. Для снижения потребления мощности в режиме Пониженной мощности, пользователь может избежать трех вышеуказанных условий, чтобы отключить опорное напряжение до того, как запустится режим пониженной мощности.

Таблица 3.16 - Характеристики внутреннего опорного напряжения

Символ	Параметр	Мин	Тип	Макс	Ед. изм.
U_{BG}	Опорное напряжение	1,15	1,23	1,35	В
t_{BG}	Время установки опорного напряжения		40	70	мкс
I_{BG}	Ток потребления опорного напряжения		10		мкА

3.6 Сторожевой таймер

Сторожевой таймер синхронизируется от отдельного генератора, встроенного в кристалл и работающего на частоте 1 МГц. Это типовое значение при $U_{CC} = 5$ В. Благодаря использованию устройства предварительного масштабирования сторожевого таймера, интервал его переустановки может быть отрегулирован. Сторожевой таймер сбрасывается с помощью команды WDR – сброс сторожевого таймера. Сторожевой таймер сбрасывается также в случае отключения питания и когда происходит общий сброс. Для определения периода сброса можно выбрать восемь различных периодов тактового цикла. Если период сброса истечет без повторного сброса сторожевого таймера, то 1887BE1У выполняет сброс и исполнение из вектора сброса.

Для предотвращения непреднамеренного отключения сторожевого таймера или непреднамеренного изменения периода прерывания с помощью бит конфигурации S8535C и WDTON выбираются три различных уровня безопасности, как это показано в таблице 3.17. Уровень безопасности 0 соответствует установке для 1887BE1У. WDT можно установить без ограничений на любой уровень безопасности.

Таблица 3.17 - Конфигурация команды WDT в зависимости от установок для перемычек M103C и WDTON

S8535C	WDTON	Уровень безопасности	Начальное значение WDT	Как запретить команду WDT	Как изменить период срабатывания
Незапрограммировано	Незапрограммировано	1	Запрещено	Временная последовательность	Временная последовательность
Незапрограммировано	Запрограммировано	2	Разрешено	Всегда разрешен	Временная последовательность
Запрограммировано	Незапрограммировано	0	Запрещено	Временная послед.	Нет ограничений
Запрограммировано	Запрограммировано	2	Разрешено	Всегда разрешен	Временная последовательность

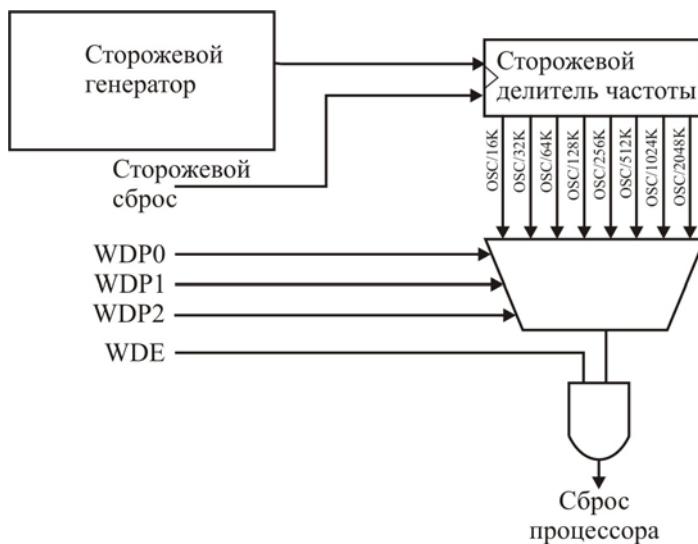


Рисунок 3.19 - Сторожевой таймер

Бит	7	6	5	4	3	2	1	0	MCUCR
Чтение/запись	-	-	-	WDCE	WDE	WDP2	WDP1	WDP0	
Начальное значение	R	R	R	R/W	R/W	R/W	R/W	R/W	

Разряды 7...5: Res - Резервные Разряды Reserved Bits

Эти разряды являются резервными разрядами для 1887ВЕ1У, они всегда должны считываться как ноль.

Разряд 4: WDCE - Разрешение на смену сторожевого таймера

Этот разряд должен быть установлен, когда разряд WDE записывается в логический ноль. В противном случае, сторожевой таймер не будет отключен. Как только разряд будет записан в единицу, аппаратное обеспечение очистит этот разряд после четырех тактовых циклов. Для уровней безопасности 1 и 2 этот разряд должен также устанавливаться при смене разрядов устройства предварительного масштабирования.

Разряд 3: WDE - Разрешение для сторожевого таймера

Если разряд WDE записывается в логическую единицу, то сторожевой таймер активируется, а если WDE записывается в логический ноль, то функция сторожевого таймера отключается. Разряд WDE может быть очищено только в том случае, если разряд WDCE имеет уровень логической единицы. Для отключения работающего сторожевого таймера необходимо выполнить следующую последовательность действий:

1 Во время той же операции, необходимо записать логическую единицу в WDCE и WDE. Логическая единица должна быть записана в WDE, даже если она установлена на единицу до начала операции отключения.

2 В течение следующих четырех циклов необходимо записать логический 0 в WDE. Благодаря этому следящее устройство отключится.

При уровне безопасности 2 невозможно отключить сторожевой таймер даже с помощью вышеописанного алгоритма.

Разряды 2...0: WDP2, WDP1, WDP0 - Биты предделителя сторожевого таймера 2, 1 и 0

Разряды WDP2, WDP1 и WDP0 определяют коэффициент деления частоты при включении сторожевого таймера. Различные значения предварительного масштабирования и их соответствующие периоды прерывания показаны в таблице 3.18.

Таблица 3.18 – Выбор предделителя сторожевого таймера

WDP2	WDP1	WDP0	Количество циклов осциллятора WDT	Типичный период WDT*
0	0	0	16K(16384)	16,3 мс
0	0	1	32K(32786)	32,5 мс
0	1	0	64K(65536)	65 мс
0	1	1	128K(131072)	0,13 с
1	0	0	256K(262144)	0,26 с
1	0	1	512K(524288)	0,52 с
1	1	0	1024K(1048576)	1 с
1	1	1	2048K(2097152)	2,1 с

* Данные значения являются лишь ориентировочными.

Ниже приведены одна функция ассемблера и одна функция С для отключения WDT. Данный пример предполагает, что прерывания контролируются (путем глобального отключения прерываний) таким образом, чтобы не происходило прерываний во время исполнения этих функций.

Assembly Code Example
<pre>WDT_off: ; Write logical one to WDCE and WDE ldi r16, (1<<WDCE) (1<<WDE) out WDTCR, r16 ; Turn off WDT ldi r16, (0<<WDE) out WDTCR, r16 ret</pre>
C Code Example
<pre>void WDT_off(void) { /* Write logical one to WDCE and WDE */ WDTCR = (1<<WDCE) (1<<WDE); /* Turn off WDT */ WDTCR = 0x00; }</pre>

Временные последовательности для изменения конфигурации сторожевоготаймера

Последовательность для изменения конфигурации сторожевого таймера имеет небольшое отличие для трех уровней безопасности. Для каждого уровня описываются различные процедуры.

Уровень безопасности 0

Этот режим совместим с работой сторожевого таймера, имеющегося в 1887ВЕ1У. Таймер устройства первоначально не активирован, но его можно запустить, записав разряд WDE в единицу без всяких ограничений. Для отключения работающего таймера и/или изменения времени прерывания, необходимо соблюдать следующую последовательность:

1 Во время той же операции, необходимо записать логическую единицу в разряд WDCE и WDE. Логическая единица должна быть записана в разряд WDE независимо от первоначального значения разряда WDE.

2 В пределах следующих четырех тактовых циклов во время той же операции, необходимо записать разряды WDE и WDP по усмотрению пользователя, но при этом разряд WDCE должен быть очищен.

Уровень безопасности 1

В этом режиме сторожевой таймер сначала отключен, но можно его включить, записав разряд WDE на 1 без всяких ограничений. При изменении периода перерыва в работе или отключении работающего таймера необходима соответствующая временная последовательность. Для отключения работающего таймера и/или изменения периода перерыва в работе, необходимо выполнить следующие процедуры:

1 Во время той же операции запишите логическую единицу в разряд WDCE и WDE. Логическая единица должна быть записана в WDE независимо от первоначального значения разряда WDE.

2 В пределах следующих четырех тактовых циклов во время той же операции, записываются разряды WDE и WDP по усмотрению пользователя, но при этом разряд WDCE должен быть очищен.

Уровень безопасности 2

В этом режиме таймер всегда включен, но разряд WDE будет всегда считываться, как единица. При изменении периода перерыва в работе необходима временная последовательность. Для изменения периода перерыва в работе таймера необходимо выполнить следующие процедуры:

1 Во время той же операции записать логическую единицу в разряд WDCE и WDE. Логическая единица должна быть записана в разряд WDE независимо от первоначального значения разряда WDE. Хотя WDE всегда считается установленным, разряд WDE необходимо записать как единицу для запуска временной последовательности.

2 В пределах следующих четырех тактовых циклов во время той же операции записать разряды WDP по усмотрению пользователя, но при этом разряд WDCE должен быть очищен. Величина, записанная в разряд WDE, значения не имеет.

3.7 Прерывания

Данный подраздел описывает особенности обращения с прерываниями, выполняемыми в микросхеме 1887ВЕ1У.

Векторы прерывания в 1887ВЕ1У

Таблица 3.19 - Векторы Сброса и Прерывания

Номер вектора	Адрес	Источник	Описание прерывания
1	0x000	RESET	Внешний пин, POR, BOR и сброс WDT
2	0x001	INT0	Запрос внешнего прерывания 0
3	0x002	INT1	Запрос внешнего прерывания 1
4	0x003	TIMER2 COMP	Сравнение-совпадение таймер/счетчика 2
5	0x004	TIMER2 OVF	Переполнение таймер/счетчика 2
6	0x005	TIMER1 CAPT	Событие захвата таймер/счетчика 1
7	0x006	TIMER1 COMPA	Сравнение-совпадение А таймер/счетчика 1
8	0x007	TIMER1 COMPB	Сравнение-совпадение В таймер/счетчика 1
9	0x008	TIMER1 OVF	Переполнение таймер/счетчика 1
10	0x009	TIMER0 OVF	Переполнение таймер/счетчика 0
11	0x00A	SPI, STC	Последовательная передача завершена
12	0x00B	USART, RXC	Прием завершен
13	0x00C	USART, UDRE	Регистр данных пуст
14	0x00D	USART, TXC	Передача завершена
15	0x00E	ADC	ADC преобразование завершено
16	0x00F	EE_RDY	Готовность EEPROM
17	0x010	ANA_COMP	Аналоговый компаратор
18	0x011	TWI	Двухпроводной последовательный интерфейс
19	0x012	INT2	Запрос внешнего прерывания 2
20	0x013	TIMER0 COMP	Сравнение-совпадение таймер/счетчика 0
21	0x014	SPM_RDY	Готовность SPM

Примечания

1 При программировании перемычки BOOTRST прибор при сбросе переходит скачкообразно в адрес Boot Loader.

2 При установке разряда IVSEL в GICR векторы сброса будут перемещаться к началу области загрузчика флэш-памяти (Boot Flash). Адрес каждого вектора прерывания станет затем адресом в данной таблице, добавленным к исходному адресу раздела Boot Flash.

В таблице 3.20 показано размещение векторов прерывания для различных комбинаций установок BOOTRST и IVSEL settings. Если программа никогда не запускает источник прерывания, то векторы прерывания не используются, и в эти места можно поместить обычный программный код. То же самое происходит, когда вектор сброса находится в секции приложения, в то время как векторы прерывания находятся в секции загрузчика и наоборот.

Таблица 3.20 - Размещение векторов сброса и прерываний

BOOTRST ¹⁾	IVSEL	Адрес сброса	Стартовый адрес векторов прерывания
1	0	0×0000	0×0000
1	1	0×0000	Адрес сброса загрузчика + 0×0001
0	0	Адрес сброса загрузчика	0×0000
0	1	Адрес сброса загрузчика	Адрес сброса загрузчика + 0×0001

¹⁾ Для бита BOOTRST значение “1” означает: не запрограммировано, а значение “0” означает запрограммировано.

Ниже представлена наиболее типичная и общая настройка программы для адресов векторов сброса и прерываний в 1887BE1У:

Адрес	Код	Источник	Комментарий
0×000	rjmp	RESET	Обработчик переустановки
0×001	rjmp	EXT_INT0	IRQ0 обработчик
0×002	rjmp	EXT_INT1	IRQ1 обработчик
0×003	rjmp	TIM2_COMP	Обработчик сравнения таймера 2
0×004	rjmp	TIM2_OVF	Обработчик переполнения таймера 2
0×005	rjmp	TIM1_CAPT	Обработчик захвата таймера 1
0×006	rjmp	TIM1_COMPA	Обработчик сравнения А таймера 1
0×007	rjmp	TIM1_COMPB	Обработчик сравнения В таймера 1
0×008	rjmp	TIM1_OVF	Обработчик переполнения таймера 2
0×009	rjmp	TIM0_OVF	Обработчик переполнения таймера 0
0×00A	rjmp	SPI_STC	Обработчик завершения переноса SPI
0×00B	rjmp	USART_RXC	Обработчик завершения USART RX
0×00C	rjmp	USART_UDRE	Обработчик освобождения UDR
0×00D	rjmp	USART_TXC	Обработчик завершения USART TX
0×00E	rjmp	ADC	Обработчик завершения преобразования АЦП
0×00F	rjmp	EE_RDY	Обработчик готовности ЭСППЗУ
0×010	rjmp	ANA_COMP	Обработчик аналогового компаратора
0×011	rjmp	TWSI	Обработчик двухпроводного последовательного интерфейса
0×012	rjmp	EXT_INT2	Обработчик IRQ 2
0×013	rjmp	TIM0_COMP	Обработчик таймер 0
0×014	rjmp	SPM_RDY	Обработчик готовности SPM
;			
0×015 RESET:	ldi	r16,high(RAMEND)	Запуск основной программы
0×016	out	SPH,r16	Установка указателя стека в старшую область памяти
0×017	ldi	r16,low(RAMEND)	
0×018	out	SPL,r16	
0×019	sei		Разрешение прерываний
0×020	<instr>	xxx	
...	

Если бит BOOTRST не запрограммирован, то размер секции загрузчика устанавливается на 2К байта и разряд IVSEL в регистре GICR устанавливается до того, как будут разрешены любые прерывания, при этом наиболее типичная и общая настройка программы для адресов векторов прерываний и сброса выглядит следующим образом:

Адрес	Код	Комментарий
0×000 RESET:	ldi r16,high(RAMEND)	Запуск основной программы
0×001	out SPH,r16	Установка указателя стека в старшую область памяти
0×002	ldi r16,low(RAMEND)	
0×003	out SPL,r16	
0×004	sei TIM2_OVF	Разрешение прерываний
0×005	<instr> xxx	
;		
.org 0xC01		
0×C01	rjmp EXT_INT0	Обработчик IRQ0
0×C01	rjmp EXT_INT1	Обработчик IRQ1
...
0×C14	rjmp SPM_RDY	Обработчик готовности SPM

Если бит BOOTRST программируется и размер секции загрузчика установлен на 2К байта, то наиболее типичная и общая настройка программы для адресов векторов прерываний и сброса выглядит следующим образом:

Адрес	Код	Комментарий
.org 0x001		
0×001	rjmp EXT_INT0	Обработчик IRQ0
0×002	rjmp EXT_INT1	Обработчик IRQ1
...
0×014	rjmp SPM_RDY	Обработчик готовности SPM
;		
.org 0xC00		
.org 0xC01		
0×C00 RESET:	ldi r16,high(RAMEND)	Обработчик готовности SPM
0×C01	out SPH,r16	Установка указателя стека в старшую область памяти
0×C02	ldi r16,low(RAMEND)	
0×C03	out SPL,r16	
0×C04	sei Разрешение прерываний	
0×C05	<instr> xxx	

Если бит BOOTRST программируется и размер секции загрузчика установлен на 2К байта и разряд IVSEL в регистре GICR устанавливается до того, как будут разрешены любые прерывания, то наиболее типичная и общая настройка программы для адресов векторов прерываний и сброса выглядит следующим образом:

Адрес	Код	Комментарий
.org 0xC00		
0×C00	rjmp RESET	Обработчик Reset
0×C01	rjmp EXT_INT0	Обработчик IRQ0
0×C02	rjmp EXT_INT1	Обработчик IRQ1
...
0×C14	rjmp SPM_RDY	Обработчик готовности SPM
;		
0×C15 RESET:	ldi r16,high(RAMEND)	Запуск основной программы
0×C16	out SPH,r16	Установка указателя стека в старшую область памяти
0×C17	ldi r16,high(RAMEND)	
0×C18	out SPH,r16	
0×C19	ldi Разрешение прерываний	
0×C20	<instr> xxx	

Перемещение прерываний между секциями приложения и загрузчика

Регистр управления управляет общим прерыванием размещением в таблице вектора прерывания.

3.7.1 Регистр управления общим прерыванием GICR

Разряд 1: IVSEL - Выбор вектора прерывания

При очистке разряда IVSEL векторы прерывания помещаются в начало флэш-памяти. При установке этого разряда в «единицу», векторы прерывания перемещаются в начало секции загрузчика флэш-памяти. Фактический адрес запуска секции загрузчика флэш-памяти определяется перемычками BOOTSZ.

Для того чтобы избежать непреднамеренных изменений в таблицах вектора прерывания, необходимо соблюдать специальную процедуру записи для изменения разряда IVSEL:

- 1 Записать разряд разрешения изменения вектора прерывания (IVCE) в «единицу».

- 2 В течение четырех циклов записать желаемое значение в IVSEL, выполняя запись нуля в IVCE.

При выполнении этой процедуры прерывания будут автоматически отключены. Прерывания отключаются при установке цикла IVCE, и они остаются в отключенном состоянии до поступления команды, следующей за записью в IVSEL. Если IVSEL не записывается, то прерывания остаются отключенными в течение четырех циклов. Автоматическое отключение не влияет на разряд I в регистре состояний.

Примечание - Если векторы прерывания размещаются в секции автоматического загрузчика программ и программируется разряд блокировки загрузки BLB02, то прерывания отключаются при выполнении из секции приложений. Если векторы прерывания размещаются в секции приложений и программируется разряд блокировки загрузки BLB12, то прерывания отключаются при исполнении из секции автоматического загрузчика программ.

Разряд 0: IVCE - Разрешение изменения вектора прерывания

Разряд IVCE должен быть записан в логическую единицу для того, чтобы разрешить изменение в разряде IVSEL. IVCE очищается аппаратным способом за четыре цикла после того, как он был записан или же был записан IVSEL. Установка разряда IVCE отключит прерывания, как это пояснено выше в описании относительно IVSEL. См. ниже пример кода.

Assembly Code Example

```
Move_interrupts:  
    ; Enable change of interrupt vectors  
    ldi r16, (1<<IVCE)  
    out GICR, r16  
    ; Move interrupts to boot Flash section  
    ldi r16, (1<<IVSEL)  
    out GICR, r16  
    ret
```

C Code Example

```
void Move_interrupts(void)  
{  
    /* Enable change of interrupt vectors */  
    GICR = (1<<IVCE);  
    /* Move interrupts to boot Flash section */  
    GICR = (1<<IVSEL);  
}
```

3.8 Порты ввода-вывода

Все порты ИМС имеют истинную функциональность чтение-модификация-запись, если они используются в качестве цифровых портов ввода-вывода общего назначения. Это означает, что направление одного вывода порта может быть изменено без не-преднамеренного изменения направления любого другого вывода с помощью инструкций SBI и CBI. То же самое имеет место при изменении величины возбуждения (drive value) (если вывод порта конфигурируется в качестве выхода) или подключения/отключения нагрузочных резисторов (если вывод порта используется в качестве входа). Каждый выходной буфер имеет симметричные характеристики возбуждения с высокими способностями для работы, как в качестве истока, так и стока. Имеющийся на выводе драйвер имеет достаточную мощность для прямого запуска светодиодов и индикаторов. Все выводы портов имеют выбираемые отдельно нагрузочные резисторы с со-противлением, не зависящим от напряжения питания. Все выводы порта имеют защитные диоды как для питания $U_{\#VCC}$, так и для земли, как это показано на рисунке 3.20.

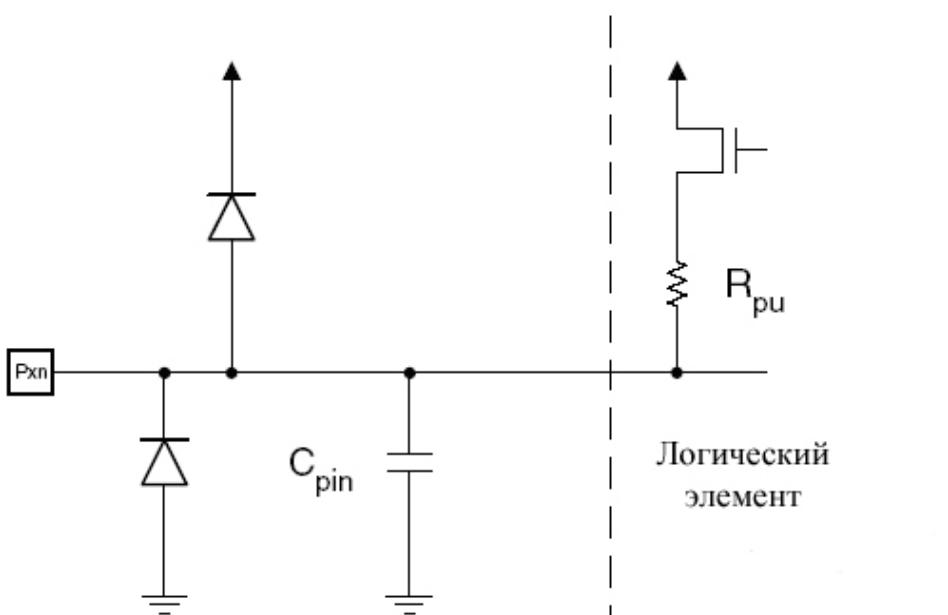


Рисунок 3.20 - Эквивалентная схема порта входа/выхода

Все данные по регистрам и разрядам в данном подразделе написаны в общем виде. Буква “x” в нижнем регистре представляет собой цифровое обозначение порта, а буква “n” в нижнем регистре представляет собой номер разряда. Однако, при использовании определений регистра или разряда в программе необходимо использовать точную форму. Например, PORTB3 для разряда номер 3 порта В в данном случае документально указан как PORTxn.

Для каждого порта имеется три адресуемых регистра ввода-вывода: регистр данных – PORTx, регистр направления данных – DDRx, и выводы порта – PINx. Значения выводов порта ввода-вывода только считываются, в то время как регистр данных и регистр направления данных считываются/записываются. Кроме того, отключение разряда PUD в SFIOR, а именно, отключение нагрузки (pull-up disable) при установке отключает функцию нагрузки (pull-up function) для всех выводов во всех портах.

Большинство выводов портов уплотнены (мультиплексированы) альтернативными функциями для периферийных функций устройства. В отношении полного описания альтернативных функций необходимо обратиться к описаниям отдельных модулей и обратить внимание на то, что активация альтернативной функции какого-либо вывода пор-

та не влияет на использование других выводов порта в качестве общего цифрового порта ввода-вывода.

3.8.1 Порты в качестве общих цифровых портов ввода-вывода

Порты являются двунаправленными портами ввода-вывода с опционными внутренними нагрузками. На рисунке 3.21 показано функциональное описание одного вывода порта ввода-вывода, который в общем виде назван Pxн.

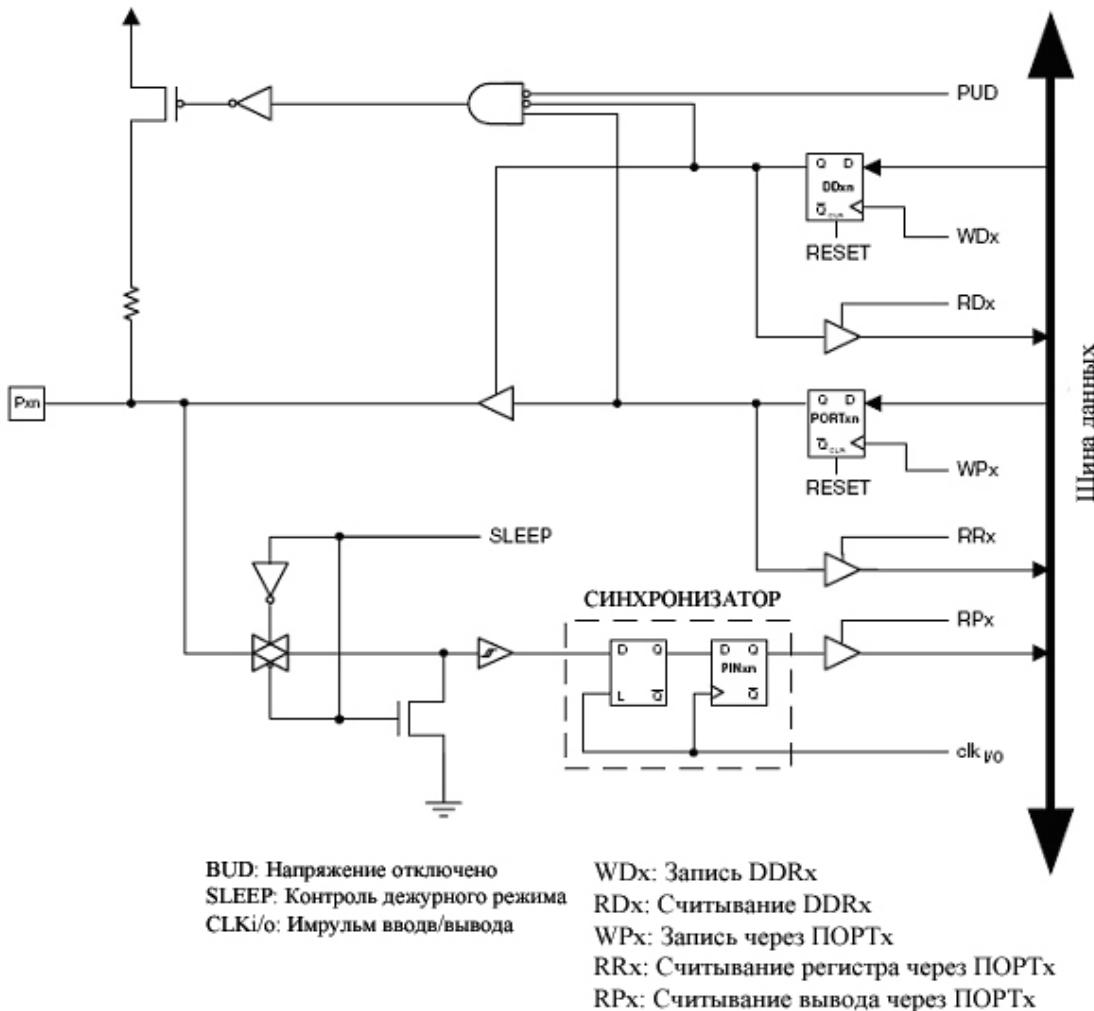


Рисунок 3.21 - Общий цифровой порт ввода-вывода

3.8.2 Конфигурация вывода

Каждый вывод порта состоит из трех регистров: DDxn, PORTxn и PINxn.

Разряд DDxn в регистре DDRx выбирает направление для данного вывода. Если в DDxn записывается логическая единица, то Pxн конфигурируется как выход. Если в DDxn записывается логический нуль, то Pxн конфигурируется как вход.

Если в PORTxn записывается логическая единица при конфигурировании порта в качестве входа, то нагрузочный резистор активируется. Для отключения нагрузочного резистора PORTxn должен быть записан логическим нулем или же порт должен быть сконфигурирован как выход. Выводы порта переходят в состояние высокого импеданса, когда сброс становится активным, даже если не работают тактовые генераторы.

Если в PORTxn записывается логическая единица при конфигурировании порта в качестве выхода, то вывод порта переводится в высокий уровень (единицу). Если в PORTxn записывается логический нуль при конфигурировании порта в качестве выхода, то вывод порта переводится в низкий уровень (нуль).

При переключении между состоянием высокого импеданса ($\{DDxn, PORTxn\} = 0b00$) и выходным высоким логическим уровнем ($\{DDxn, PORTxn\} = 0b11$) промежуточное состояние должно установиться либо благодаря активированным нагрузкам ($\{DDxn, PORTxn\} = 0b01$), либо выходу, находящемуся в низком логическом состоянии ($\{DDxn, PORTxn\} = 0b10$). В обычных условиях разрешенное состояние для нагрузки является полностью приемлемым, поскольку окружающая среда с высоким импедансом не обнаружит разницу между мощным драйвером и нагрузочным сопротивлением. В случае если это не так, то разряд PUD в регистре SFIOR может быть установлен на отключение всех нагрузок во всех портах.

Переключение между входом, имеющим нагрузочные сопротивления, и выходом в низком логическом уровне создает аналогичную проблему. Пользователь должен либо использовать высокоимпедансное состояние ($\{DDxn, PORTxn\} = 0b00$), либо состояние высокого уровня на выходе в качестве промежуточного шага ($\{DDxn, PORTxn\} = 0b10$). В таблице 3.21 приводятся сводные данные о сигналах управления для данного значения вывода.

3.8.3 Считывание значения вывода

Независимо от разряда направления данных DDxn, вывод порта может считываться с помощью бита регистра PINxn. Как показано на рисунке 3.23, разряд регистра PINxn и предшествующая ему триггер-защелка образуют синхронизирующее устройство. Оно необходимо для того, чтобы избежать метастабильности, если физический вывод изменяет значение недалеко от фронта внутреннего синхронизирующего импульса, однако, это в свою очередь, вносит задержку. Рисунок 3.22 изображает временную диаграмму синхронизации при считывании значения вывода, прикладываемого извне. Максимальное и минимальное время задержки распространения обозначены, соответственно, $t_{pd,max}$ и $t_{pd,min}$.

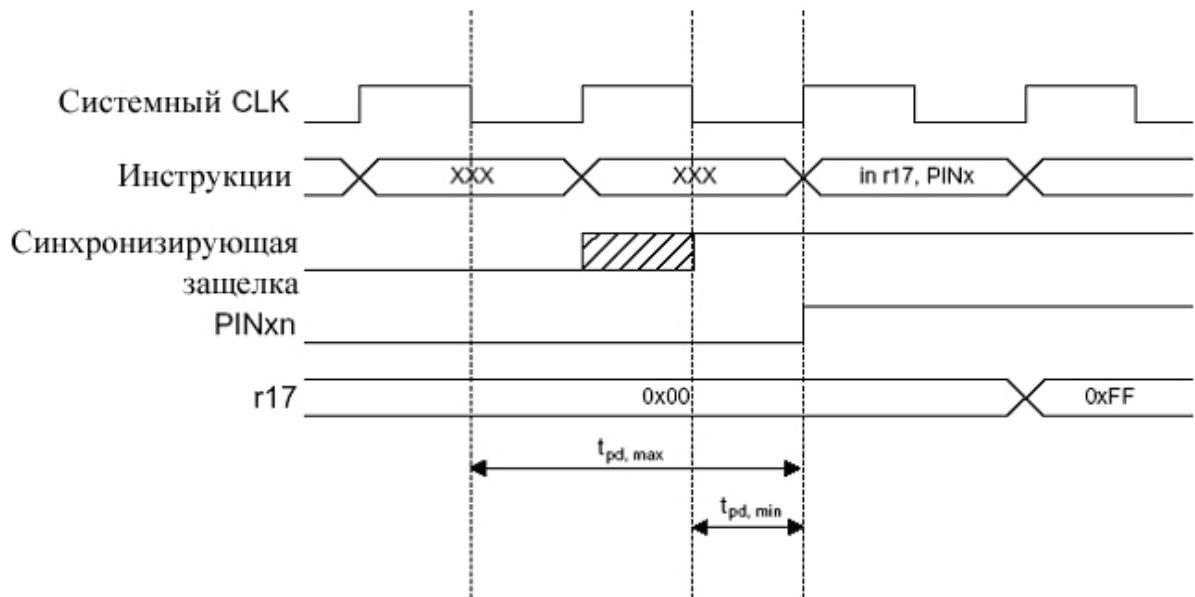


Рисунок 3.22 - Синхронизация при считывании значения вывода, определяемого извне

Рассмотрим тактовый период, начинающийся сразу же после заднего фронта импульса синхронизации системы. Защелка закрывается, когда тактовый сигнал находится в низком состоянии, и становится прозрачной, когда тактовый сигнал находится в высоком состоянии, как это показано в заштрихованной области сигнала “SYNC LATCH” рисунка 3.22. Величина сигнала фиксируется, когда системный тактовый импульс переходит в низкое состояние. Он синхронизируется в регистр PINxn при последующем положительном фронте сигнала синхронизации. Как указано двумя стрелками $t_{pd,max}$ и $t_{pd,min}$, переход одиночного сигнала на выводе будет задержан на величину от $\frac{1}{2}$ до $1\frac{1}{2}$ тактового периода в зависимости от времени установления уровня сигнала. При считывании величины для вывода, который был определен программным способом, необходимо вставить команду «пор», как указано на рисунке 3.23. Команда «out» устанавливает сигнал “SYNC LATCH” к положительному фронту сигнала синхронизации. В этом случае задержка t_{pd} при прохождении через синхронизирующее устройство составляет один тактовый период системы.

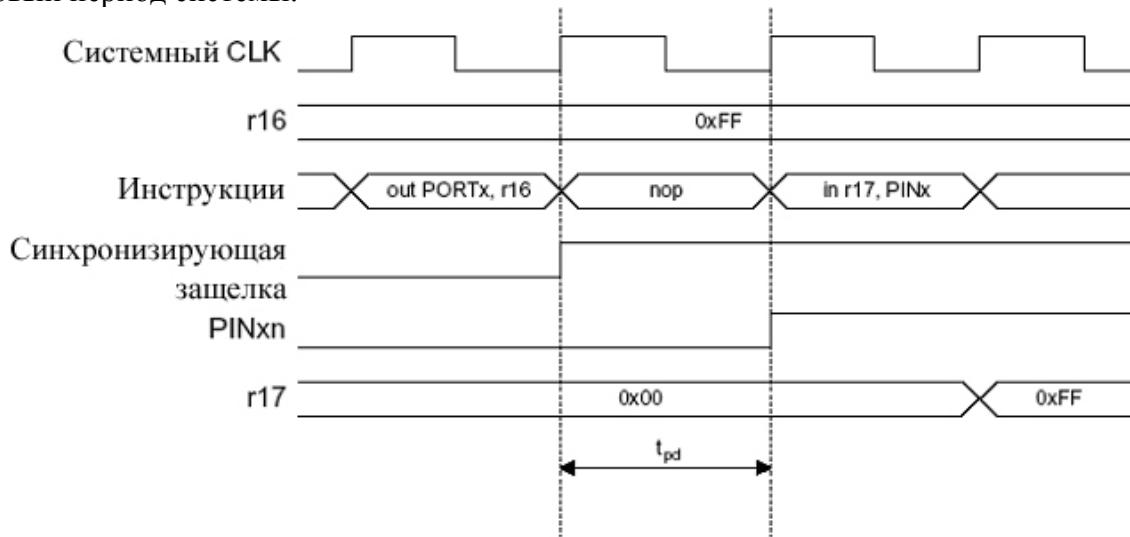


Рисунок 3.23 - Синхронизация при считывании значения вывода, определенного программно

Следующий пример кодирования показывает, как устанавливать выводы 0 и 1 порта В в высокий уровень, выводы 2 и 3 на низкий уровень, и определяет выводы порта В от 4 по 7 как входы с нагрузочными резисторами, приписанными к выводам 6 и 7 порта. Полученные значения выводов считываются снова, но как уже упоминалось ранее, сюда включена команда «пор», чтобы иметь возможность снова считывать значение, недавно приписанное к некоторым выводам.

Assembly Code Example⁽¹⁾

```

...
; Define pull-ups and set outputs high
; Define directions for port pins
ldi    r16,(1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi    r17,(1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out   PORTB,r16
out   DDRB,r17
; Insert nop for synchronization
nop
; Read port pins
in    r16,PINB
...

```

C Code Example

```

unsigned char i;
...
/* Define pull-ups and set outputs high */
/* Define directions for port pins */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Insert nop for synchronization*/
_NOP();
/* Read port pins */
i = PINB;
...

```

Примечание - Для программы ассемблера используются два временных регистра с целью свести к минимуму время от момента установки нагрузочных резисторов на выводах 0, 1, 6 и 7 до тех пор, пока будут правильно установлены биты направлений, при этом биты 2 и 3 определяются как низкий уровень, а биты 0 и 1 переустанавливаются как мощные драйверы.

3.8.4 Режим разрешения цифрового входа и спящий режим

Как показано на рисунке 3.23, входной цифровой сигнал может быть подсоединен к земле на входе триггера Шмитта. Сигнал, далее обозначенный SLEEP на рисунке, устанавливается с помощью контроллера спящего режима МК в режим пониженной мощности, энергосберегающий режим, дежурный режим и длительный дежурный режим для того, чтобы избежать высокого потребления мощности, если некоторые входные сигналы оставлены как плавающие или имеют уровень аналогового сигнала близкий к $U_{\#VCC}/2$.

SLEEP отменяется (корректируется) для выводов портов, активированных как выводы внешнего прерывания. Если запрос на внешнее прерывание не разрешается, то SLEEP является активным и для этих выводов. SLEEP может также отменяться другими различными альтернативными функциями.

Если на выводе асинхронное внешнее прерывание, получившем конфигурацию “прерывание по переднему фронту, заднему фронту или любом логическом изменении на выводе”, присутствует высокий логический уровень (“единица”) в то время, когда внешнее прерывание не разрешено, то соответствующий флаг внешнего прерывания будет установлен при возобновлении работы после возвращения из вышеупомянутых спящих режимов, поскольку запирание в этих спящих режимах вызывает запрашиваемое логическое изменение.

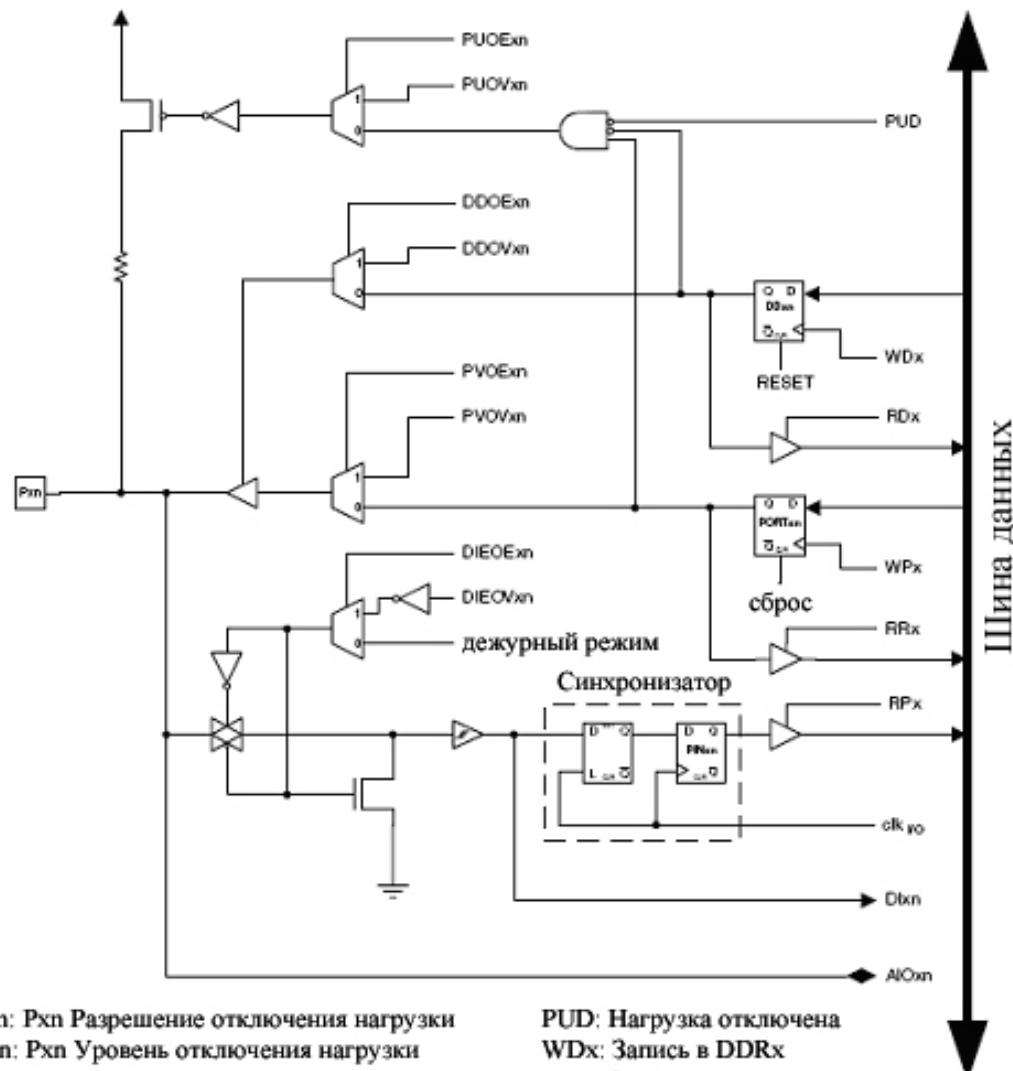
3.8.5 Неподсоединеные выводы

Если какие-то выводы не используются, то рекомендуется, чтобы эти выводы имели определенный логический уровень. Даже если большинство цифровых входов в спящих режимах отключено как это описано выше, следует избегать плавающих входов для снижения потребления во всех других режимах, для которых цифровые входы разрешены (режимы Reset, Active и Idle).

Самый простой метод обеспечения определенного уровня на неиспользуемых выводах заключается в подключении внутренней нагрузки. В этом случае повышение напряжения будет отключаться во время сброса. Если существенным фактором является малая потребляемая мощность во время сброса, то рекомендуется использовать внешнее повышение или понижение напряжения. Не рекомендуется непосредственно подключать неиспользуемые входы к #VCC или #GND, поскольку это может вызвать повышенные токи, если случайно этот вывод будет сконфигурирован как выход.

3.8.6 Альтернативные функции портов

Большинство портов имеют альтернативные функции в дополнение к основным цифровым входам/выходам. На рисунке 3.24 показано, каким образом сигналы управления выводом порта из упрощенной схемы могут быть скорректированы альтернативными функциями. Сигналы коррекции могут не присутствовать на выводах всех портов, этот рисунок представляет собой общее описание, применимое к выводам всех портов в семействе микроконтроллеров AVR.

PUOE_{xn}: Px_n Разрешение отключения нагрузкиPUOV_{xn}: Px_n Уровень отключения нагрузкиDDOE_{xn}: Px_n Разрешение отключения адреса данных

PUD: Нагрузка отключена

WD_x: Запись в DDR_xRD_x: Считывание DDR_x

Примечание - WP_x, WD_x, RR_x, RP_x и RD_x являются общими для всех выводов внутри того же порта. clk_{I/O}, SLEEP и PUD являются общими для всех портов. Все остальные сигналы являются единственными для каждого вывода.

Рисунок 3.24 - Альтернативные функции портов

В таблице 3.21 представлена сводная информация о сигналах корректировки. Индексы выводов и портов из рисунка 3.24 не приведены в нижеследующих таблицах. Сигналы корректировки генерируются внутри модулей, имеющих альтернативную функцию.

Таблица 3.21 - Общее описание сигналов корректировки для альтернативных функций

Наименование сигнала	Полное наименование	Описание
PUOE	Разрешение корректировки повышения напряжения	Если установлен этот сигнал, то разрешение на повышение напряжения управляется с помощью сигнала PUOV. Если этот сигнал сбрасывается, то повышение напряжения разрешается в том случае, когда {DDxn, PORTxn, PUD} = 0b101.
PUOV	Значение корректировки повышения напряжения	Если устанавливается PUOE, то повышение напряжения разрешается/отменяется когда PUOV устанавливается /очищается независимо от установки DDxn, PORTxn, и разрядов Регистра PUD.
DDOE	Разрешение корректировки направления данных	Если установлен этот сигнал, то Разрешение Выходного Драйвера управляется сигналом DDOV. Если этот сигнал сбрасывается, то Выходной драйвер активируется с помощью разряда Регистра DDxn.
DDOV	Значение корректировки направления данных	Если установлено значение DDOE, то выходной драйвер разрешается/отключается, если устанавливается/сбрасывается DDOV независимо от установки разряда регистра DDxn.
PVOE	Разрешение корректировки значения порта	Если установлен этот сигнал, и разрешен драйвер выхода, то значение порта управляется сигналом PVOV. Если PVOE сбрасывается, и разрешается выходной драйвер, то значение управляется разрядом выходного драйвера PORTxn.
PVOV	Величина корректировки значения порта	Если установлено PVOE, то значение порта устанавливается на PVOV, независимо от установки разряда регистра PORTxn.
DIEOE	Цифровой вход разрешение корректировки разрешения	Если установлен этот разряд, то разрешение цифрового входа управляется сигналом DIEOV. Если этот сигнал сбрасывается, то разрешение цифрового входа определяется состоянием МК (нормальный режим, спящие режимы).
DIEOV	Величина корректировки значения цифрового входа	Если установлено DIEOE, то цифровой вход разрешается/отключается, если DIEOV устанавливается/сбрасывается независимо от состояния МК (нормальный режим, спящие режимы).
DI	Цифровой вход	Это цифровой вход для изменения функций. На схеме сигнал подводится к выходу триггера Шмитта, но перед синхронизирующими устройством. Если цифровой вход не используется в качестве цифрового входа, то модуль с альтернативной функцией будет использовать свое собственное синхронизирующее устройство.
AI/O	Аналоговый вход/выход	Это аналоговый вход/выход к альтернативным функциям и от них. Сигнал подсоединяется непосредственно к выводу и может быть использован двунаправленно.

В нижеследующих подразделах кратко описаны альтернативные функции для каждого порта и связь сигналов корректировки с альтернативными функциями. Подробности смотрите в описании альтернативных функций.

Регистр специальных функций – SFIOR

Бит	7	6	5	4	3	2	1	0	SFIOR
	ADTS2	ADTS1	ADTS0	ADHSM	ACME	PUD	PSR2	PSR10	
Чтение/запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 2: PUD - Отключение нагрузочных резисторов

Если этот разряд записывается в единицу, то нагрузочные сопротивления в портах входа/выхода отключаются, даже если регистры DD_{xn} и PORT_{xn} имеют конфигурацию, активирующую нагрузочные сопротивления ($\{DD_{xn}, PORT_{xn}\} = 0b01$).

3.8.7 Альтернативные функции порта А

Порт А имеет альтернативную функцию в качестве аналогового входа для АЦП, как это показано в таблице 3.22. Если какие-либо выводы порта А конфигурируются как выходы, то очень важно, чтобы они не переключались в процессе перехода. Это может исказить результаты преобразования.

Таблица 3.22 – Альтернативные функции порта А

Разряды порта	Альтернативная функция
PA7	АЦП входной канал 7
PA6	АЦП входной канал 6
PA5	АЦП входной канал 5
PA4	АЦП входной канал 4
PA3	АЦП входной канал 3
PA2	АЦП входной канал 2
PA1	АЦП входной канал 1
PA0	АЦП входной канал 0

Таблицы 3.23 и 3.24 показывают связь альтернативных функций порта А с сигналами корректировки.

Таблица 3.23 - Сигналы корректировки для альтернативных функций в PA7...PA4

Обозначение сигнала	PA7/ADC7	PA6/ADC6	PA5/ADC5	PA4/ADC4
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
PI/O	ADC7 вход	ADC6 вход	ADC5 вход	ADC4 вход

Таблица 3.24 - Сигналы корректировки для альтернативных функций в PA3...PA0

Обозначение сигнала	PA3/ADC3	PA2/ADC2	PA1/ADC1	PA0/ADC0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	0	0
PVOV	0	0	0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	ADC3 вход	ADC2 вход	ADC1 вход	ADC0 вход

3.8.8 Альтернативные функции порта В

Таблица 3.25 - Альтернативные функции выводов порта В

Разряды порта	Альтернативная функция
PB7	SCK (SPI тактовый импульс)
PB6	MISO (SPI ведущий-вход, ведомый-выход)
PB5	MOSI (SPI ведущий-выход, ведомый-вход)
PB4	SS# (SPI выбор ведомого)
PB3	AIN1 (Отрицательный вход аналогового компаратора) OC0 (Выход сравнение/совпадение таймер/счетчика 0)
PB2	AIN0 (Положительный вход аналогового компаратора) INT2 (Вход внешнего прерывания 2)
PB1	T1 (Внешний счетный вход таймер/счетчика 1)
PB0	T0 (Внешний счетный вход таймер/счетчика 0) XCK (USART внешний тактовый вход/выход)

Конфигурация альтернативных выводов выглядит следующим образом:

SCK – Порт В, Бит 7

SCK: выход ведущего тактового генератора, вход ведомого тактового генератора для канала SPI. Когда SPI запускается в качестве вспомогательного устройства, этот вывод получает конфигурацию входа независимо от установок для DDB7. Когда SPI запускается в качестве основного устройства, то направление данных для этого вывода контролируется с помощью DDB7. Если вывод переводится принудительно для выполнения функций входа, нагружочное сопротивление по-прежнему можно контролировать с помощью бита PORTB7.

MISO – Порт В, Бит 6

MISO: вход данных ведущего, выход данных ведомого для канала SPI. При запуске SPI в качестве ведущего, этот вывод получает конфигурацию входа независимо от установок DDB6. При запуске SPI в качестве ведомого, направление данных для этого вы-

вода контролируется DDB6. Если SPI переводит вывод на функции входа, то нагрузочное сопротивление может по-прежнему управляться битом PORTB6.

MOSI – Порт В, Бит 5

MOSI: выход данных ведущего, вход данных ведомого для канала SPI. При запуске SPI в качестве ведомого, этот вывод получает конфигурацию входа независимо от установок DDB5. При запуске SPI в качестве ведущего, направление данных для этого вывода контролируется DB5. Если SPI переводит вывод на функции входа, то нагрузочное сопротивление может по-прежнему управляться битом PORTB5.

SS# – Порт В, Бит 4

SS# : вывод выбора ведомого устройства (Slave). При запуске SPI в качестве ведомого, этот вывод получает конфигурацию входа независимо от установок DDB4. В качестве ведомого устройства (Slave), SPI активируется, когда этот вывод переходит в низкое состояние. При запуске SPI в качестве ведущего, направление данных для этого вывода контролируется DDB4. Если SPI переводит вывод на функции входа, то нагрузочное сопротивление может по-прежнему управляться битом PORTB4.

AIN1/OC0 – Порт В, Бит 3

AIN1: Отрицательный вход аналогового компаратора. Конфигурируйте этот вывод порта при отключенном внутреннем нагрузочном резисторе для того, чтобы функция цифрового порта избегала взаимных помех с функцией аналогового компаратора.

OC0: выход сравнения соответствия выхода: вывод PB3 может служить внешним выходом для сравнения соответствия таймера/счетчика 0. Вывод PB3 необходимо конфигурировать в качестве выхода (DDB3 set (one)) для выполнения этой функции. Вывод OC0 является также выводом выхода для функций таймера режима PWM.

AIN0/INT2 – Порт В, бит 2

AIN0: положительный аналоговый вход компаратора. Конфигурируйте этот вывод порта В при отключенном внутреннем нагрузочном резисторе для того, чтобы функция цифрового порта избегала взаимных помех с функцией аналогового компаратора.

INT2: источник внешнего прерывания 2: PB2 может быть использован как источник внешнего прерывания для МК.

T1 – Порт В, бит 1

T1: источник счетных импульсов таймера/счетчика1.

T0/XCK – Порт В, бит 0

T0: источник счетчика таймер/счетчика0.

XCK: внешний тактовый генератор УСАПП. Регистр направления данных (DDB0) контролирует: является ли тактовый генератор выходом (DDB0 установлен) или входом (DDB0 очищен). Вывод XCK является активным только, если УСАПП работает в синхронном режиме.

SPI ведущий вход и SPI ведомый выход объединяют сигнал MISO, а SPI ведущий выход и SPI ведомый вход – сигнал MOSI.

Таблица 3.26 - Сигналы корректировки для альтернативных функций в PB7...PB4

Сигнал	PB7/SCK	PB6/MISO	PB5/MOSI	PB4/SS#
PUOE	SPE x MSTR#	SPE x MSTR	SPE x MSTR#	SPE x MSTR#
PUOV	PORTB7 x PUD#	PORTB6 x PUD#	PORTB5 x PUD#	PORTB4xPUD#
DDOE	SPE x MSTR#	SPE x MSTR	SPE x MSTR#	SPE x MSTR#
DDOV	0	0	0	0
PVOE	SPE x MSTR	SPE x MSTR#	SPE x MSTR	0
PVOV	SCK OUTPUT	SPI SLAVE OUT	SPI MSTR OUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SCK INPUT	SPI MSTR IN	SPI SLAVE IN	SPI SS#
AI0	-	-	-	-

Таблица 3.27 - Сигналы корректировки для альтернативных функций в PB3...PB0

Сигнал	PB3/OC0/AIN1	PB2/INT2/AIN0	PB1/T1	PB0/T0/XCK
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC0 ENABLE	0	0	UMSEL
PVOV	OC0	0	0	XCK OUTPUT
DIEOE	0	INT2 ENABLE	0	0
DIEOV	0	1	0	0
DI	-	INT2 INPUT	T1 INPUT	XCK IN / T0 IN
AI0	AIN1 INPUT	AIN0 INPUT	-	-

3.8.9 Альтернативные функции порта С

Выводы порта С с альтернативными функциями показаны в таблице 3.28.

Таблица 3.28 - Выводы альтернативных функций порта С

Выводы порта	Альтернативная функция
PC7	TOSC2 (Вход 2 осциллятора таймера)
PC6	TOSC1 (Вход 1 осциллятора таймера)
PC1	SDA (TWI шина данных)
PC0	SCL (TWI шина тактовых импульсов)

Альтернативная конфигурация выводов выглядит следующим образом:

TOSC2 – Порт С, бит 7

TOSC2, вывод 2 генератора таймера: при установке бита AS2 в ASSR для запуска асинхронного тактирования таймера/счетчика 2, вывод PC7 отсоединяется от порта и становится выходом инвертирующего усилителя генератора. В этом режиме кварцевый генератор подсоединен к этому выводу, и он не может использоваться в качестве порта входа/выхода.

TOSC1 – Порт С, бит 6

TOSC1, вывод 1 генератора таймера: при установке бита AS2 в ASSR для запуска асинхронного тактирования таймера/счетчика 2, вывод PC6 отсоединяется от порта и становится входом инвертирующего усилителя генератора. В этом режиме кварцевый генератор подсоединяется к этому выводу и он не может использоваться в качестве порта входа/выхода.

SDA – Порт С, бит 1

SDA, шина данных двухпроводного последовательного интерфейса: когда в TWCR для запуска двухпроводного последовательного интерфейса устанавливается бит TWEN (единица), то вывод PC1 отсоединяется от порта и становится выводом последовательного ввода-вывода данных для двухпроводного последовательного интерфейса. В этом режиме на выводе присутствует фильтр пиковых напряжений для подавления выбросов короче

50 нс для входного сигнала, и вывод возбуждается с помощью драйвера с открытым стоком с ограничением скорости нарастания. Когда этот вывод используется двухпроводным последовательным интерфейсом, нагрузочное сопротивление может контролироваться с помощью разряда PORTC1.

SCL – Порт С, бит 0

SCL - тактовая шина двухпроводного последовательного интерфейса: когда в TWCR для запуска двухпроводного последовательного интерфейса устанавливается бит TWEN (единица), то вывод PC0 отсоединяется от порта и становится выводом последовательного ввода-вывода данных для двухпроводного последовательного интерфейса. В этом режиме на выводе присутствует фильтр пиковых напряжений для подавления выбросов входного сигнала короче 50 нс, и вывод возбуждается с помощью драйвера с открытым стоком с ограничением скорости нарастания. Когда этот вывод используется двухпроводным последовательным интерфейсом, то нагрузочное сопротивление все еще может контролироваться с помощью разряда PORTC0.

Таблица 3.29 - Корректирующие сигналы для альтернативных функций в PC7, PC6

Сигнал	PC7/TOSC2	PC6/TOSC1	PC1/SDA	PC0/SCL
PUOE	AS2	AS2	TWEN	TWEN
PUOV	0	0	PORTC1 x PUD#	PORTC0 x PUD#
DDOE	AS2	AS2	TWEN	TWEN
DDOV	0	0	SDA_OUT	SCL_OUT
PVOE	0	0	TWEN	TWEN
PVOV	0	0	0	0
DIEOE	AS2	AS2	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	T/C2 OSC OUT	T/C2 OSC IN	SDA INPUT	SCL INPUT
Примечание - При активировании двухпроводной последовательный интерфейс запускает управление скоростью нарастания на выходных выводах PC0 и PC1. В дополнение к этому, фильтры выбросов подсоединенны между выходами AIO, показанными на схеме порта, и цифровой логикой модуля TWI.				

3.8.10 Альтернативные функции порта D

Выводы порта D с альтернативными функциями показаны в таблице 3.30.

Таблица 3.30 - Альтернативные функции выводов порта D

Разряды порта	Альтернативная функция
PD7	OC2 (T/C2 выход сравнения/совпадения)
PD6	ICP (T/C1 вход захвата)
PD5	OC1A (T/C1 выход сравнения/совпадения)
PD4	OC1B (T/C1 выход сравнения/совпадения)
PD3	INT1 (Вход внешнего прерывания 1)
PD2	INT0 (Вход внешнего прерывания 0)
PD1	TXD (USART выход данных)
PD0	RXD (USART вход данных)

Конфигурация альтернативного вывода выглядит следующим образом:

OC2 – Порт D, Бит 7

OC2 - выход сравнения/совпадения счетчика/таймера2. Вывод OC2 является также выводом выхода для функции таймера режима PWM.

ICP – Порт D, Бит 6

ICP – вывод входного захвата: вывод PD6 может действовать в качестве захвата входа для таймера/счетчика 1.

OC1A – Порт D, Бит 5

OC1A - выход сравнения/совпадения A T/C1. Для выполнения этой функции вывод должен иметь конфигурацию выхода (DDD5 установлен). Вывод OC1A является также выходом для функции таймера режима PWM.

OC1B – Порт D, Бит 4

OC1B - выход сравнения/совпадения B T/C1. Для выполнения этой функции вывод должен иметь конфигурацию выхода (DDD4 установлен). Вывод OC1B является также выходом для функции таймера режима PWM.

INT1 – Порт D, Бит 3

INT1 - источник 1 внешнего прерывания, вывод PD3 может служить источником внешнего прерывания.

INT0 – Порт D, Бит 2

INT0 - источник 0 внешнего прерывания, вывод PD2 может служить источником внешнего прерывания.

TXD – Порт D, Бит 1

TXD - передача данных (выход данных для УСАПП). При активации передатчика УСАПП этот вывод конфигурируется в качестве выхода независимо от значения DDD1.

RXD – Порт D, Бит 0

RXD - прием данных (вход данных для УСАПП). При активации приемника УСАПП, этот вывод конфигурируется в качестве входа независимо от значения DDD0. Когда УСАПП переводит этот вывод во вход, то нагрузочное сопротивление по-прежнему можно контролировать с помощью разряда PORTD0.

Таблицы 3.31 и 3.32 связывают альтернативные функции порта D с сигналами коррекции.

Таблица 3.31 - Сигналы коррекции для альтернативных функций PD7...PD4

Сигнал	PD7/OC2	PD6/ICP	PD5/OC1A	PD4/OC1B
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	OC2 ENABLE	0	OC1A ENABLE	OC1B ENABLE
PVOV	OC2	0	OC1A	OC1B
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	ICP INPUT	-	-
AIO	-	-	-	-

Таблица 3.32 - Сигналы коррекции для альтернативных функций PD3...PD0

Сигнал	PD3/	PD2/	PD1/	PD0/
PUOE	0	0	TXEN	RXEN
PUOV	0	0	0	PORTD0 × PUD#
DDOE	0	0	TXEN	RXEN
DDOV	0	0	1	0
PVOE	0	0	TXEN	0
PVOV	0	0	TXD	0
DIEOE	INT1 ENABLE	INT0 ENABLE	0	0
DIEOV	1	1	0	0
DI	INT1 INPUT	INT0 INPUT	-	RXD
AIO	-	-	-	-

Описание регистров для портов входа-выхода**Регистр данных порта А – PORTA**

Бит	7	6	5	4	3	2	1	0	PORTA
Чт./Зап.	R/W								
Начальное значение	0	0	0	0	0	0	0	0	

Регистр направления данных порта А – DDRA

Бит	7	6	5	4	3	2	1	0	DDRA
Чт./Зап.	R/W								
Начальное значение	0	0	0	0	0	0	0	0	

Адрес выводов входа порта А – PINA

Бит	7	6	5	4	3	2	1	0	PINA
Чт./Зап.	R/W								
Начальное значение	0	0	0	0	0	0	0	0	

Регистр данных порта В – PORTB

Бит	7	6	5	4	3	2	1	0	PORTB
Чт./Зап.	R/W								
Начальное значение	0	0	0	0	0	0	0	0	

Регистр направления данных порта В - DDRB

Бит	7	6	5	4	3	2	1	0	DDRB
Чт./Зап.	R/W								
Начальное значение	0	0	0	0	0	0	0	0	

Адрес выводов входа порта В – PINB

Бит	7	6	5	4	3	2	1	0	PINB
Чт./Зап.	R/W								
Начальное значение	0	0	0	0	0	0	0	0	

Регистр данных порта С – PORTC

Бит	7	6	5	4	3	2	1	0	PORTC
Чт./Зап.	R/W								
Начальное значение	0	0	0	0	0	0	0	0	

Регистр направления данных порта С - DDRC

Бит	7	6	5	4	3	2	1	0	DDRC
Чт./Зап.	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0	
Начальное значение	R/W								

Адрес выводов входа порта С – PINC

Бит	7	6	5	4	3	2	1	0	PINC
Чт./Зап.	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	
Начальное значение	R/W								

Регистр данных порта D – PORTD

Бит	7	6	5	4	3	2	1	0	PORTD
Чт./Зап.	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	
Начальное значение	R/W								

Регистр направления данных порта D – DDRD

Бит	7	6	5	4	3	2	1	0	DDRD
Чт./Зап.	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0	
Начальное значение	R/W								

Адрес выводов входа порта D – PIND

Бит	7	6	5	4	3	2	1	0	PIND
Чт./Зап.	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	
Начальное значение	R/W								

3.9 Внешние прерывания

Внешние прерывания запускаются с помощью выводов INT0 - INT2. Следует обратить внимание на то, что если они активируются, то прерывания запустятся даже в случае, если выводы INT0 - INT2 сконфигурированы как выходы. Подобная особенность обеспечивает генерацию программного прерывания. Внешние прерывания могут быть запущены фронтом нарастания или спада или низким уровнем (INT2 является прерыванием, запускаемым только фронтом). Они настраиваются, как указано в спецификации для регистра управления МПУ – MCUCR и регистром состояния и управления МПУ – MCUCSR. Если активируется внешнее прерывание, и оно конфигурируется для управления величиной уровня (только INT0/INT1), то прерывание будет запускаться, как только вывод будет удерживаться в низком состоянии.

Распознавание прерываний фронта спада или нарастания на INT0 и INT1 требует наличия тактового сигнала входа/выхода. Прерывания низкого уровня на INT0/INT1 и прерывание фронта на INT2 обнаруживаются асинхронно. Это означает, что эти прерывания могут использоваться для активации прибора также из спящих режимов в отличие от режима холостого хода. Тактовый генератор входа/выхода приостанавливается во всех спящих режимах, за исключением режима холостого хода.

Если уровень запускаемого прерывания используется для пробуждения из режима пониженной мощности, то измененный уровень должен некоторое время удерживаться для активации МПУ. Это делает МПУ менее чувствительным к шуму. Измененный уровень выбирается дважды с помощью тактового импульса генератора сторожевого таймера. Период генератора равен 1 мкс (номинальное напряжение) при 5,0 В и 25 °C. Частота генератора зависит от напряжения. МПУ будет запущено, если на входе присутствует требуемый уровень во время этой выборки, или если этот уровень сохраняется до окончания времени запуска. Время запуска определяется битами SUT. Если уровень дважды анализируется тактовым генератором следящего устройства, но он исчезает до окончания времени запуска, то МПУ все еще будет способно к активации, но прерывание при этом генерироваться не будет. Требуемый уровень должен удерживаться достаточно длительное время, чтобы МПУ завершило пробуждение для запуска прерывания уровня.

3.9.1 Регистр управления МПУ – MCUCR

Регистр управления МПУ содержит управляющие разряды для управления уточненным состоянием прерывания и общими функциями МПУ.

Бит	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Чт./Зап.	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 3, 2: ISC11, ISC10 - Управление обнаружением прерываний 1 разряд 1 и разряд 0

Внешнее прерывание 1 активируется с помощью внешнего вывода INT1, если установлены разряд SREG I и соответствующая маска прерывания в GICR. Уровень и фронты сигналов на внешнем выводе INT1, активирующем прерывание, приведены в таблице 3.33. Значение на выводе INT1 снимается (определяется) до обнаружения фронтов. Если выбирается прерывание с использованием фронта или переключения, то импульсы, длиющиеся более одного тактового периода, будут создавать прерывание. Более короткие импульсы не могут гарантированно создавать прерывания. Если выбирается низкий уровень прерывания, то он должен поддерживаться до завершения исполняемой в данный момент команды для того, чтобы создать прерывание.

Таблица 3.33 - Управление обнаружением прерываний 1

ISC11	ISC10	Описание
0	0	Низкий уровень INT1 генерирует запрос на прерывание
0	1	Любое изменение лог. уровня на INT1 генерирует запрос на прерывание
1	0	Задний фронт INT1 генерирует запрос на прерывание
1	1	Передний фронт INT1 генерирует запрос на прерывание

Разряды 1, 0: ISC01, ISC00 - Управление обнаружением прерываний 0 разряд 1 и разряд 0

Внешнее прерывание 0 активируется с помощью внешнего вывода INT0, если установлены флаг SREG I и соответствующая маска прерывания. Величина уровня и фронты на внешнем выводе INT0, которые активируют прерывание, приведены в таблице 3.34. Значение на выводе INT0 анализируется до обнаружения фронтов. Если выбирается прерывание с помощью фронтов или переключения, то импульсы, превышающие по длительности один тактовый период, будут создавать прерывание. При этом не гарантируется, что более короткие импульсы будут создавать прерывание. Если выбирается низкий уровень прерывания, то он должен сохраняться до тех пор, пока завершение исполнения в данный момент команды не вызовет прерывания.

Таблица 3.34 - Управление обнаружением прерывания 0

ISC01	ISC00	Описание
0	0	Низкий уровень INT0 генерирует запрос на прерывание.
0	1	Любое логическое изменение на INT0 генерирует запрос на прерывание.
1	0	Задний фронт INT0 генерирует запрос на прерывание.
1	1	Передний фронт INT0 генерирует запрос на прерывание.

Регистр состояния и управления МПУ – MCUCSR

Бит	7	6	5	4	3	2	1	0	MCUCSR
Чт./Зап.	-	ISC2	-	-	WDRF	BORF	EXTRF	PORF	
Начальное значение	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	

Разряд 6: ISC2 - Управление обнаружением прерывания 2

Асинхронное внешнее прерывание 2 активируется внешним выводом INT2, если устанавливаются разряд SREG I и соответствующая маска прерывания в GICR. Если ISC2 записывается в ноль, то задний фронт на INT2 активирует прерывание. Если ISC2 записывается в единицу, то передний фронт на INT2 активирует прерывание. Фронты на INT2 регистрируются асинхронно. Импульсы на INT2 с шириной больше, чем ширина минимального импульса, приведенного в таблице 3.35, будут генерировать прерывание. Более короткие импульсы не гарантируют генерацию прерывания. При изменении разряда ISC2 может произойти прерывание. С учетом этого, рекомендуется сначала дезактивировать INT2 путем очистки его разряда разрешения прерывания в регистре GICR, затем можно изменить разряд ISC2. После всего этого необходимо очистить флаг прерывания INT2 с помощью записи логической единицы в один из разрядов его флага прерывания (INTF2) в регистр GIFR до того, как будет вновь разрешено прерывание.

Таблица 3.35

Символ	Параметр	Условие	Мин	Тип	Макс	Единицы
t_{INT}	Минимальная ширина импульса для асинхронного внешнего прерывания		TBD	50	TBD	нс

3.9.2 Регистр управления общим прерыванием – GICR

Бит	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	-	-	-	IVSEL	IVCE	GICR
Чт./Зап.	R/W	R/W	R/W	R	R	R	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7: INT1 - Разрешение запроса 1 на внешнее прерывание

Когда устанавливается разряд INT1 (единица) и устанавливается разряд I в регистре состояний (SREG), то активируется прерывание для внешнего вывода. Разряд управления 1 обнаружения прерывания (Interrupt Sense Control1 bits) 1/0 (ISC11 и ISC10) в регистре общего управления МПУ (MCUCR) устанавливает, будет ли внешнее прерывание активироваться передним и/или задним фронтом вывода INT1 или будет обнаруживаться по уровню. Выполнение операций на выводе вызовет запрос на прерывание даже в том случае, когда INT1 конфигурируется как выход. Соответствующее прерывание запроса 1 на внешнее прерывание исполняется из вектора прерывания INT1.

Разряд 6: INT0 - Разрешение запроса 0 на внешнее прерывание

Когда устанавливается разряд INT0 (единица) и устанавливается разряд I в регистре состояний (SREG) на (единицу), то активируется прерывание для внешнего вывода. Разряд управления 0 обнаружения прерывания 1/0 (ISC01 и ISC00) в регистре общего управления МПУ (MCUCR) определяет, будет ли внешнее прерывание активироваться передним и/или задним фронтом вывода INT1 или будет обнаруживаться по уровню. Выполнение операций на выводе вызовет запрос на прерывание даже в том случае, когда INT0 конфигурируется как выход. Соответствующее прерывание запроса 0 на внешнее прерывание исполняется из вектора прерывания INT0.

Разряд 5: INT2 - Разрешение запроса 2 на внешнее прерывание

Когда устанавливается разряд INT2 и устанавливается разряд I в регистре состояний (SREG) , на (единицу), то активируется прерывание для внешнего вывода. Разряд управления 2 обнаружения прерывания (ISC2) в регистре общего управления МПУ (MCUCR), определяет, будет ли внешнее прерывание активироваться передним и/или задним фронтом вывода INT2. Выполнение операций на выводе вызовет запрос на прерывание даже в том случае, когда INT2 конфигурируется как выход. Соответствующее прерывание запроса 2 на внешнее прерывание исполняется из вектора прерывания INT2.

3.9.3 Регистр флага общего прерывания – GIFR

Бит	7	6	5	4	3	2	1	0	
	INTF1	INTF0	INTF2	-	-	-	-	-	GIFR
Чт./Зап.	R/W	R/W	R/W	R	R	R	R	R	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7: INTF1 - Флаг 1 внешнего прерывания

Когда фронт или логическое изменение на выводе INT1 запускает запрос на прерывание, то INTF1 устанавливается (единица). Если разряд I в SREG и разряд INT1 в GICR установятся (единица), то МПУ скачком перейдет на соответствующий вектор прерывания. Флаг очистится после выполнения процедуры прерывания. В качестве альтернативы флаг может быть очищен путем записи в него логической единицы. Этот флаг всегда очищается, когда NT1 получает конфигурацию прерывания уровня.

Разряд 6: INTF0 - Флаг 0 внешнего прерывания

Когда фронт или логическое изменение на выводе INT0 запускает запрос на прерывание, то INTF0 устанавливается (единица). Если разряд I в SREG и разряд INT0 в GICR установятся (единица), то МПУ скачком перейдет на соответствующий вектор прерывания. Флаг очистится после выполнения процедуры прерывания. В качестве альтернативы флаг может быть очищен путем записи в него логической единицы. Этот флаг всегда очищается, когда NT0 получает конфигурацию прерывания уровня.

Разряд 5: INTF2 - Флаг 2 внешнего прерывания

Когда событие на выводе INT2 запускает запрос на прерывание, то INTF2 устанавливается (единица). Если разряд I в SREG и разряд INT2 в GICR установятся (единица), то МПУ скачком перейдет на соответствующий вектор прерывания. Флаг очистится после выполнения процедуры прерывания. В качестве альтернативы флаг может быть очищен путем записи в него логической единицы. При переходе в некоторые спящие режимы при отключенном прерывании INT2 буфер входа на этом выводе будет отключен. Это может вызвать логическое изменение во внутренних сигналах, которые будут устанавливать флаг INTF2.

3.10 8-разрядный таймер/счетчик 0 с ШИМ (PWM)

Таймер/счетчик 0 (T/C0) является одноканальным модулем 8-разрядного Т/С общего назначения.

Отличительные особенности:

- одноканальный счетчик;
- сброс таймера при совпадении сравнения (автоматическая перезагрузка);
- ШИМ с фазовой коррекцией, свободный от сбоев;
- генератор частоты;
- счетчик внешних событий;
- 10-разрядный предварительный делитель частоты;
- источники прерывания, при совпадении при сравнении и при переполнении (TOV0 и OCF0).

3.10.1 Обзор

Упрощенная блок-схема 8-разрядного таймера/счетчика показана на рисунке 3.25.

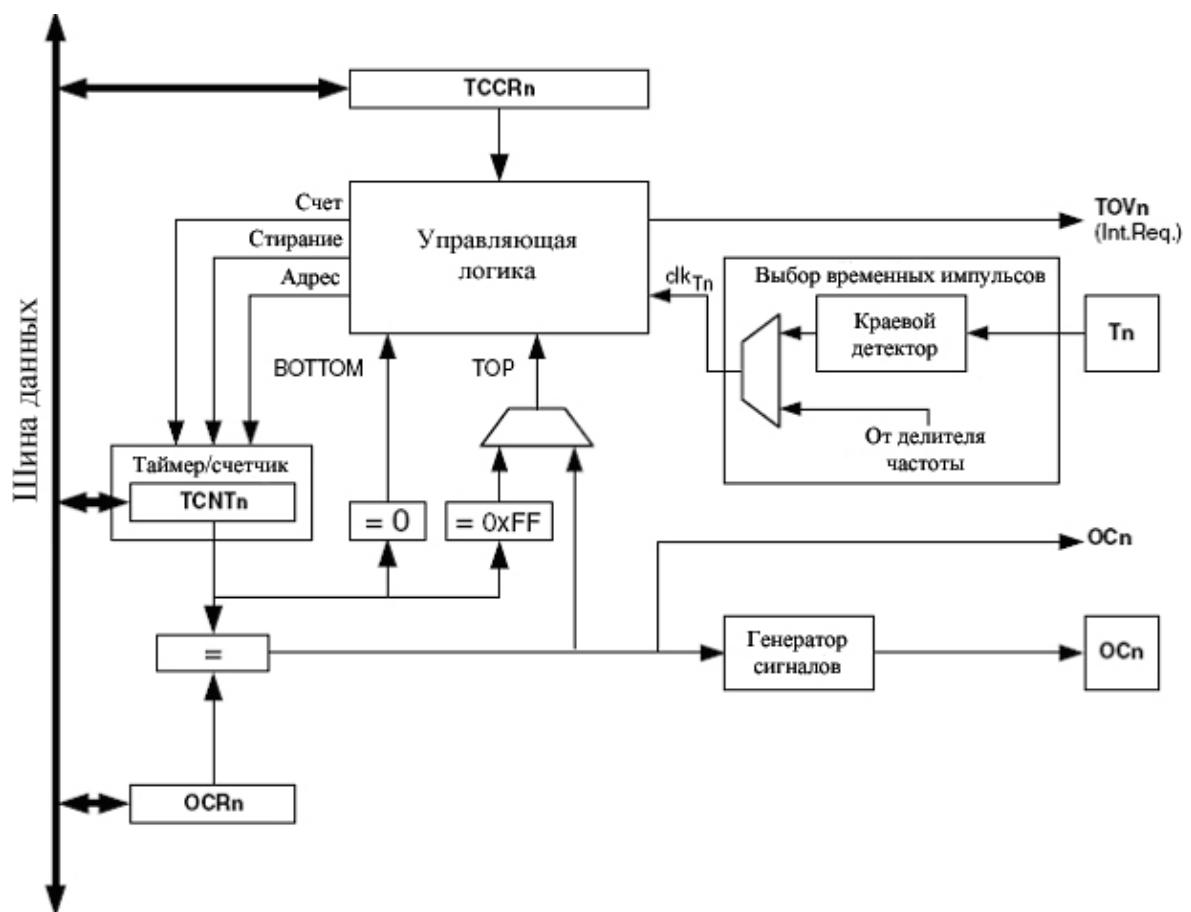


Рисунок 3.25 - Блок-схема 8-разрядного таймера/счетчика и его окружения

3.10.2 Регистры

Таймер/счетчик (TCNT0) и регистр сравнения выхода (OCR0) являются 8-разрядными регистрами. Все сигналы запросов на прерывание (на рисунке 3.25 сокращенно Int.Req.) видны в регистре флага прерывания таймера (TIFR). Все прерывания индивидуально маскируются с помощью регистра маски прерывания таймера (TIMSK). Эти регистры TIFR и TIMSK не показаны на рисунке, поскольку эти регистры совместно используются другими временными устройствами.

Таймер/счетчик может синхронизироваться внутрисхемно, через предварительный делитель частоты, или с помощью источника внешнего тактового генератора на выводе T0. Логический блок управления синхронизацией выбирает, какой источник сигнала синхронизации и фронта использует Таймер/Счетчик для увеличения или уменьшения его значения. Таймер/счетчик не активен, если не выбран источник сигнала синхронизации. Выход логики выбора тактового сигнала обозначается как генератор таймера ($\text{clk}_{\text{T}0}$). Регистр сравнения выхода с двойным буфером (OCR0) постоянно сравнивается со значением таймер/счетчик. Результат сравнения может быть использован генератором формы сигнала для запуска ШИМ или выхода переменной частоты на выводе сравнения выходов (OC0). Событие совпадения будет также устанавливать флаг сравнения (OCF0), который может использоваться для запроса прерывания сравнения выхода.

3.10.3 Определения

Многие справочные сведения относительно регистров и разрядов записаны в КФДЛ.431295.025ТО в общем виде. Буква “n” в нижнем регистре обозначения заменяет номер таймера/счетчика, в данном случае 0. Использование регистра или разряда определяется программой, необходимо использовать точную форму, например, TCNT0 для оценки значения таймер/счетчик 0 и т. д.

Определения в таблице 3.36 также используются в данном документе.

Таблица 3.36 – Определения

BOTTOM	Счетчик достигает уровня BOTTOM, 0x00.
MAX	Счетчик достигает уровня MAXimum, когда он становится равным 0xFF (декадальное 55).
TOP	Счетчик достигает уровня TOP, когда он становится равным наивысшему значению в последовательности подсчета. Значение TOP может быть приписано фиксированному значению 0xFF (MAX) или значению, сохраненному в регистре OCR0. Само присваивание зависит от режима работы.

3.10.4 Источники тактового сигнала таймера/счетчика

Таймер/счетчик может синхронизироваться как внутренним, так и внешним источником синхронизации. Источник синхронизации выбирается с помощью логики выбора тактового генератора, который управляется с помощью разрядов выбора сигнала синхронизации (CS02:0), расположенных в регистре управления таймера/счетчика (TCCR0).

3.10.5 Модуль счетчика

Основной частью 8-разрядного таймера/счетчика является программируемый двухнаправленный модуль счетчика. На рисунке 3.26 показана блок-схема модуля счетчика.

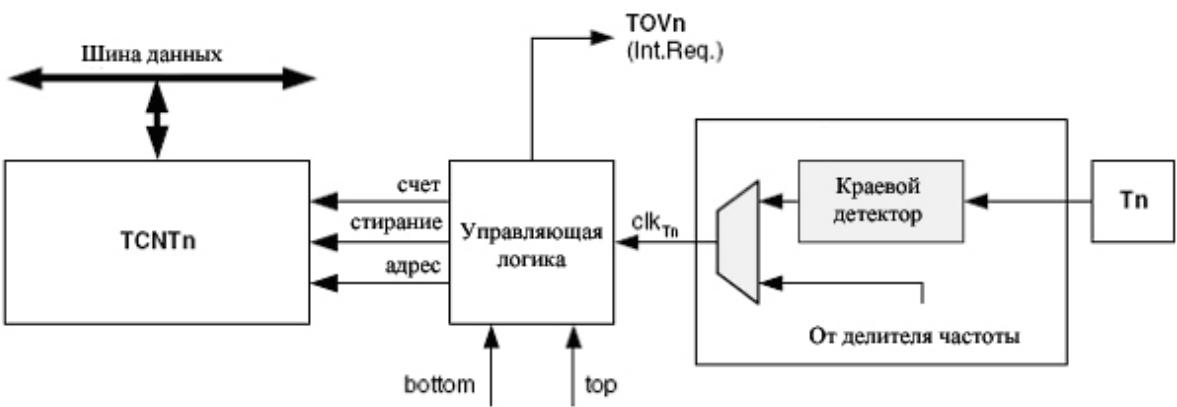


Рисунок 3.26 - Блок схема модуля счетчика

Описание сигналов (внутренние сигналы):

count	- увеличение или уменьшение TCNT0 на 1;
direction	- выбор между увеличением или уменьшением
clear	- очистка TCNT0 (все разряды устанавливаются в ноль)
clkTn	- тактовый генератор таймера/счетчика, далее именуемый как clkT0
top	- сигнализирует, что TCNT0 достигло максимального значения
bottom	- сигнализирует, что TCNT0 достигло минимального значения (нуля).

В зависимости от используемого режима работы счетчик сбрасывается, увеличивается или уменьшается при каждом тактовом сигнале таймера (clkT0). clkT0 сможет генерироваться от внешнего или внутреннего источника тактового сигнала, выборка при этом производится с помощью разрядов Выборка тактовых сигналов (CS02:0). Если источник тактовых сигналов не выбирается, т. е. (CS02:0 = 0), то таймер останавливается. В то же время величина TCNT0 может быть выбрана с помощью ЦПУ независимо от того, присутствует clkT0 или нет. Операция записи в ЦПУ отменяет (обладает большим приоритетом) все операции очистки счетчика или операции счета.

Последовательность счета определяется путем установки разрядов WGM01 и WGM00, расположенных в регистре управления таймером/счетчиком (TCCR0). Существует непосредственная связь между тем, как ведет себя счетчик (т. е. как он считает) и каким образом создаются формы сигнала на выходе в сравнении с выходом OC0.

Флаг переполнения таймера/счетчика (TOV0) устанавливается в соответствии с режимом работы, который выбирается с помощью разрядов WGM01:0. Для генерации прерывания ЦПУ можно использовать TOV0.

3.10.6 Модуль сравнения выхода

8-разрядный компаратор непрерывно сравнивает TCNT0 с регистром сравнения выхода (OCR0). Как только TCNT0 становится равным OCR0, компаратор сигнализирует о совпадении. Совпадение установит флаг сравнения выхода (OCF0) при следующем цикле таймера тактового генератора. При активации (OCIE0 = 1 при установке флага общего прерывания в SREG) флаг сравнения выхода создает прерывание сравнения выхода. Флаг OCF0 очищается автоматически при исполнении прерывания. В качестве альтернативы, флаг OCF0 можно очищать программно путем записи логической единицы в адрес разряда входа/выхода. Генератор формы сигнала использует сигнал сравнения для генерации выходного сигнала согласно режиму работы, установленному разрядами WGM01:0 и режиму сравнения выхода, установленному разрядами (COM01:0). Максимальное и нижнее значение сигнала используется генератором формы сигнала для решения специальных случаев с экстремальными значениями в некоторых режимах работы.

На рисунке 3.27 показана схема блока управления устройства сравнения выхода.

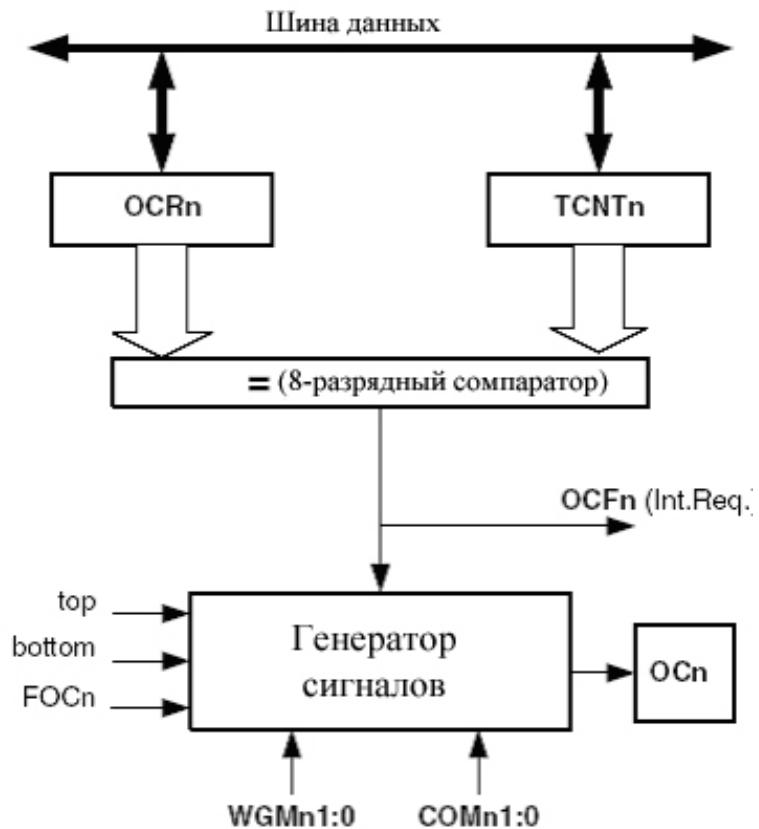


Рисунок 3.27 – Блок - схема модуля сравнения выхода

При работе в любом режиме с использованием ШИМ (широтно-импульсная модуляция), регистр OCR0 имеет двойное буферирование. При работе в обычном режиме и режиме очистки таймера при сравнении, СТС двойное буферирование отключается. Двойное буферирование синхронизирует обновление регистра сравнения либо вверх, либо вниз относительно последовательности подсчета. Синхронизация предотвращает возникновение импульсов избыточной длины, несимметричных импульсов ШИМ, обеспечивая таким образом бесперебойную работу выхода. Доступ к регистру может показаться достаточно сложным, но это не так. При активации двойного буферирования ЦПУ имеет доступ к регистру буфера OCR0, а если двойное буферирование отключено, то ЦПУ будет осуществлять прямую выборку OCR0.

3.10.7 Принудительное сравнение по выходу

В режимах генерации без использования ШИМ - принципа, выход сравнения для компаратора может быть установлен на запись единицы в разряд принудительного сравнения выхода (FOC0). Принудительное сравнение/совпадение не будет устанавливать флаг OCF0 или перезагружать/сбрасывать таймер, однако вывод OC0 будет обновляться, как будто произошло реальное сравнение совпадений (установки разрядов COM01:0 определяют, что произойдет с выводом OC0: будет ли он установлен, сброшен или переключен).

3.10.8 Блокировка совпадения/сравнения с помощью записи TCNT0

Все операции записи в регистр TCNT0 будут блокировать любое совпадение сравнений, происходящее в следующем цикле синхронизации таймера, даже в случае, когда таймер остановлен. Эта особенность позволяет инициализировать OCR0 на то же значение, что и TCNT0 без запуска прерывания, когда активируется тактовый генератор таймера/счетчика.

3.10.9 Использование блока сравнения выхода

Поскольку запись TCNT0 в любом рабочем режиме будет блокировать все совпадения для одного тактового цикла работы таймера, то существуют связанные с этим риски при смене TCNT0, когда используется канал сравнения выхода, независимо от того, будет ли работать таймер/счетчик или нет. Если значение, записанное в TCNT0, равно величине OCR0, то совпадение сравнений будет утрачено, что дает в результате неправильную генерацию формы сигнала. Аналогично, не следует записывать значение TCNT0, равное BOTTOM, когда счетчик отсчитывает счет вниз.

Настройка OC0 должна выполняться до установки регистра направления данных для вывода порта для выхода. Наиболее простым способом установки величины OC0 является использование разрядов стробирования принудительного сравнения выхода (FOC0) в нормальном режиме. Регистр OC0 сохраняет свое значение, даже если происходит изменение между режимами генерации формы сигнала.

Необходимо убедиться в том, что разряды COM01:0 не имеют двойного буферирования вместе с величиной сравнения. Изменение разрядов COM01:0 сразу же даст эффект.

3.10.10 Блок сравнения совпадений на выходе

Разряды режима сравнения на выходе (COM01:0) несут две функции. Генератор формы сигнала использует разряды COM01:0 для оценки состояния сравнения на выходе при следующем совпадении сравнений. Кроме этого, разряды COM01:0 управляют источником выхода для вывода OC0. На рисунке 3.28 представлена упрощенная блок - схема логики, используемой при установке разряда COM01:0. Регистры входа/выхода, разряды входа/выхода и выводы входа/выхода на данном рисунке показаны жирным шрифтом. Показаны только части общих регистров управления портом входа/выхода (DDR и PORT) на которые влияют разряды COM01:0. При ссылках на состояние OC0, речь идет о об внутреннем регистре OC0, а не о выводе OC0. Если происходит переустановка системы, то регистр OC0 переустанавливается на “0”.

Общая функция порта входа/выхода корректируется с помощью сравнения выхода (OC0) из генератора формы сигнала, если устанавливается какой-либо из разрядов COM01:0. Направление вывода OC0 (вход или выход) все еще контролируется регистром направления данных (DDR) для вывода порта. Разряд регистра направления данных для вывода OC0 (DDR_OC0) должен быть установлен в качестве выхода до того, как величина OC0 станет видимой на выводе. Функция коррекции порта не зависит от режима генерации формы сигнала.

Логическая схема сравнения выхода позволяет инициализацию состояния OC0 до того, как активируется выход. Следует обратить внимание на то, что некоторые установки разряда COM01:0 сохраняются для определенных режимов работы.

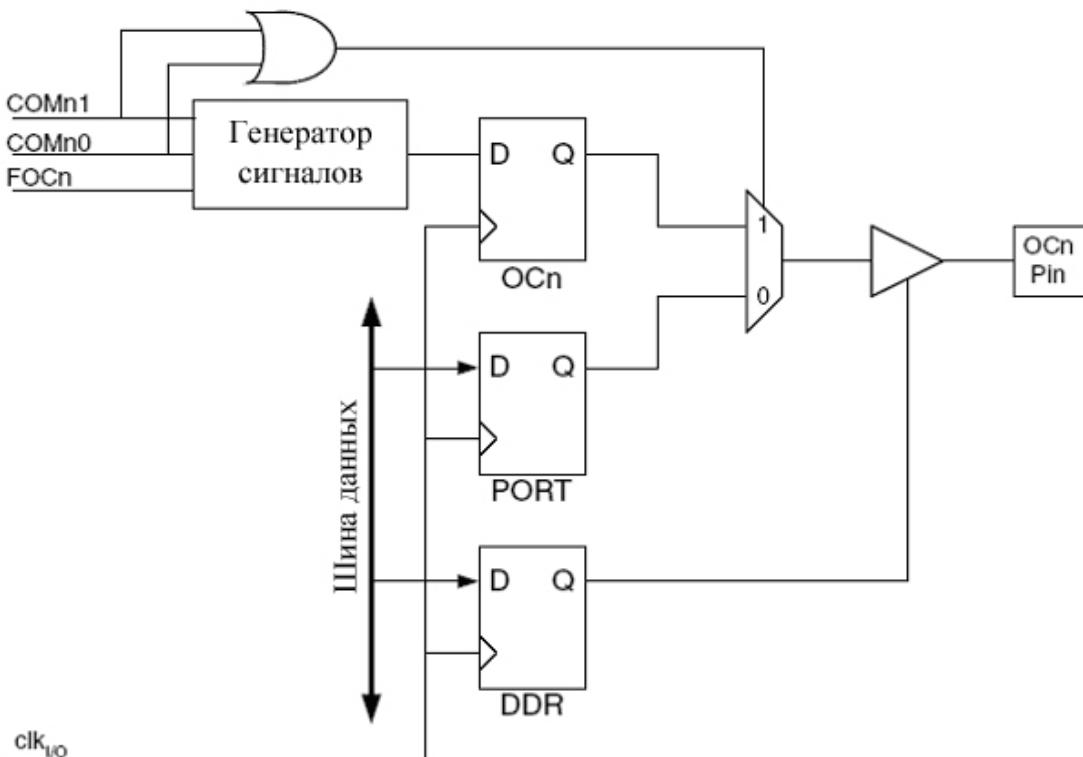


Рисунок 3.28 - Блок сравнения/совпадения на выходе

Общая функция порта входа/выхода корректируется с помощью сравнения выхода (OC0) из генератора формы волны, если устанавливается какой-либо из разрядов COM01:0. Однако, направление вывода OC0 (вход или выход) все еще контролируется регистром направления данных (DDR) для вывода порта. Разряд регистра направления данных для вывода OC0 (DDR_OC0) должен быть установлен в качестве выхода до того, как величина OC0 станет видимой на выводе. Функция коррекции порта не зависит от режима генерации формы сигнала.

Логическая схема вывода сравнения выхода позволяет инициализацию состояния OC0 до того, как активируется выход. Следует обратить внимание на то, что некоторые установки разряда COM01:0 сохраняются для определенных режимов работы.

3.10.11 Режим сравнения выхода и генерация формы сигнала

Генератор формы сигнала использует разряды COM01:0 различным образом для обычного режима, режима условных переходов (СТС) и режима широтно-импульсной модуляции. Для всех типов режимов установка COM01:0 = 0 сообщает генератору формы сигнала, что в регистре OC0 не должно выполняться никаких действий по следующему сравнению/совпадению.

Изменение состояния разрядов в COM01:0 повлияет на первое совпадение сравнений после записи разрядов. Для режимов, отличающихся от ШИМ, может быть использовано действие, дающее немедленный эффект благодаря использованию разрядов стробирования FOC0.

3.10.12 Режимы работы

Режим работы, т. е. поведение таймера/счетчика и выводов сравнения выхода, определяется комбинацией разрядов для режима генерации формы сигнала (WGM01:0) и режима сравнения выхода (COM01:0). Разряды режима сравнения выхода не влияют на последовательность подсчета, в то время как разряды режима генерации формы сигнала влияют. Разряды COM01:0 управляют тем, будет ли инвертирован или нет сгенерированный выход ШИМ (инвертированный или неинвертированный ШИМ). Для режимов, не использующих ШИМ, разряды COM01:0 контролируют должен ли выход быть установлен, очищен или переключен при совпадении сравнений.

Нормальный режим

Самым простым режимом работы является нормальный режим ($WGM01:0 = 0$). В этом режиме направление подсчета всегда следует вверх (возрастает) и сброс счетчика не выполняется. Счетчик просто переполняется, когда он проходит свое максимальное 8-разрядное значение ($TOP = 0xFF$) и затем перезапускается ($0x00$). В нормальном режиме работы флаг переполнения таймера/счетчика (TOV0) будет установлен во время того же цикла синхронизации таймера, когда TCNT0 становится равным нулю. Флаг TOV0 в этом случае ведет себя как 9-й разряд, за исключением того, что он только устанавливается, а не сбрасывается. Однако, при объединении с прерыванием переполнения таймера, которое автоматически сбрасывает флаг TOV0, разрешение таймера можно повысить программным способом. В нормальном режиме нет необходимости рассматривать специальные случаи, при этом новое значение таймера может быть записано в любое время.

Блок сравнения выхода может использоваться для генерации прерываний в заданное время. Использование сравнения выхода для генерации формы сигнала в нормальном режиме не рекомендуется, поскольку это займет слишком много времени у ЦПУ.

Сброс таймера в режиме сравнения/совпадения (CTC)

В режиме сброса таймера при сравнении или режиме CTC, ($WGM01:0 = 2$), регистр OCR0 используется для управления разрешением счетчика. В режиме CTC счетчик сбрасывается до нуля, когда значение счетчика (TCNT0) совпадает с OCR0. OCR0 определяет верхнее значение для счетчика, а следовательно, его разрешение. Данный режим обеспечивает большую степень управления выходной частотой сравнения совпадения. Это упрощает также работу для подсчета внешних событий.

Временная диаграмма для режима CTC показана на рисунке 3.29. Значение счетчика (TCNT0) возрастает до тех пор, пока не произойдет совпадение между TCNT0 и OCR0, а затем счетчик (TCNT0) сбрасывается.

Прерывание может генерироваться каждый раз, когда значение счетчика достигает величины TOP путем использования флага OCF0. Если прерывание разрешается, то процедуру прерывания программы обработчика можно использовать для обновления величины TOP. В то же время, изменение TOP на значение, близкое к BOTTOM при работе счетчика, когда не имеется никакой величины или существует малое значение предварительного делителя частоты, должно выполняться с осторожностью, поскольку режим CTC не имеет двойного буферирования. Если новое значение, записанное в OCR0, ниже, чем текущее значение TCNT0, то счетчик упустит совпадение при сравнении. Затем счетчику придется считать до своего максимального значения ($0xFF$) и возвращаться (wrap around) снова, начиная с $0x00$, прежде, чем может произойти совпадение сравнений.

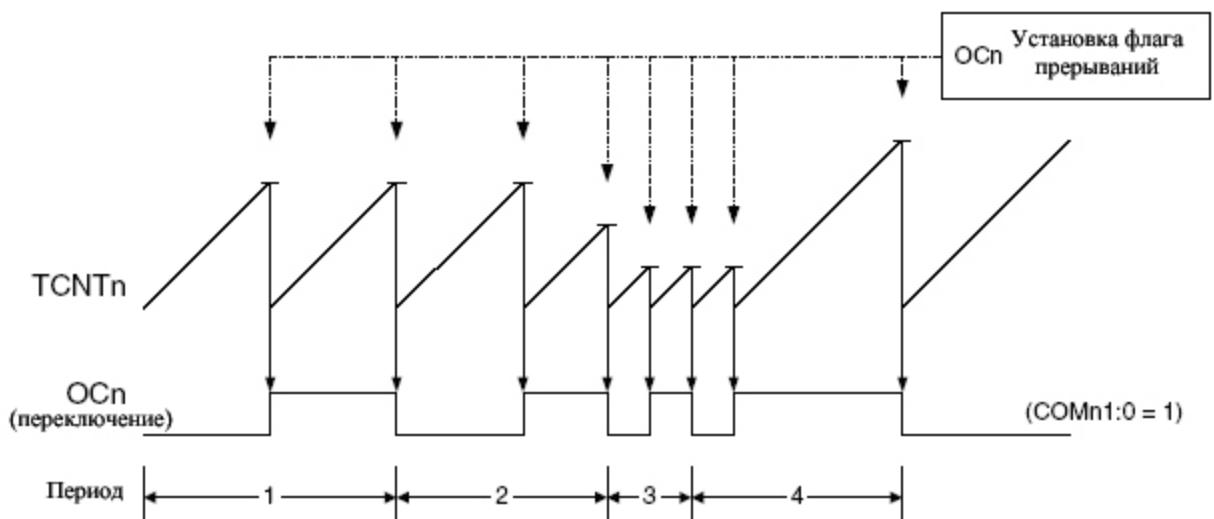


Рисунок 3.29 - Режим СТС, временная диаграмма

Для генерации выхода формы сигнала в режиме СТС выход **OC0** может быть установлен на переключение своего логического уровня при каждом совпадении сравнения путем установки разрядов режима сравнения выхода на режим переключения (**COM01:0=1**). Значение **OC0** не будет видно на выводе порта до тех пор, пока направление данных не установится для вывода. Генерируемая форма сигнала будет иметь максимальную частоту $f_{OC0} = f_{clk_I/O}/2$ при установке **OCR0** на нуль (0x00). Частота для формы сигнала определяется следующим выражением:

$$f_{OCn} = \frac{f_{clk_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

Переменная “N” представляет собой коэффициент предварительного деления частоты (1, 8, 64, 256 или 1024).

Что касается нормального режима работы, то флаг **TOV0** устанавливается во время того же цикла синхронизации таймера, при котором счетчик считает от **MAX** до 0x00.

Быстрый ШИМ режим

Быстрая широтно-импульсная модуляция или быстрый ШИМ режим (**WGM01:0=3**) предоставляет опцию генерации формы сигнала ШИМ с высокой частотой. Быстродействующая ШИМ отличается от другой опции ШИМ благодаря ее работе с использованием одиночных фронтов. Счетчик выполняет счет из положения **BOTTOM** до **MAX**, затем вновь начинает работу с **BOTTOM**. В неинвертируемом режиме сравнения выхода, сравнение по выходу (**OC0**) очищается при совпадении сравнения между **TCNT0** и **OCR0**, и устанавливается как **BOTTOM**. В инвертируемом режиме сравнения выхода, выход устанавливается при совпадении сравнения и сбрасывается при **BOTTOM**. Благодаря работе с использованием одиночных фронтов, рабочая частота для быстрого ШИМ режима может в два раза превышать частоту ШИМ режима с фазовой коррекцией частоты, которая использует принцип работы с применением двух фронтов. Благодаря столь высокой частоте быстрый ШИМ режим очень хорошо подходит для регулировки мощности, выпрямления, и применения в ЦАП. Высокая частота обуславливает малые физические размеры внешних компонентов (катушек, конденсаторов), благодаря чему снижается общая стоимость системы.

В быстром ШИМ режиме счетчик возрастает до тех пор, пока его значение не достигнет значения MAX. Затем счетчик сбрасывается при следующем цикле синхронизации таймера. Временная диаграмма для быстрого ШИМ режима показана на рисунке 3.30. Значение величины TCNT0 на временной диаграмме показано в виде гистограммы для иллюстрации работы в режиме с использованием одиночных фронтов. На схеме показаны как инвертированные, так и не инвертированные ШИМ выходы. Небольшие горизонтальные метки на фронтах TCNT0 представляют совпадения между OCR0 и TCNT0.

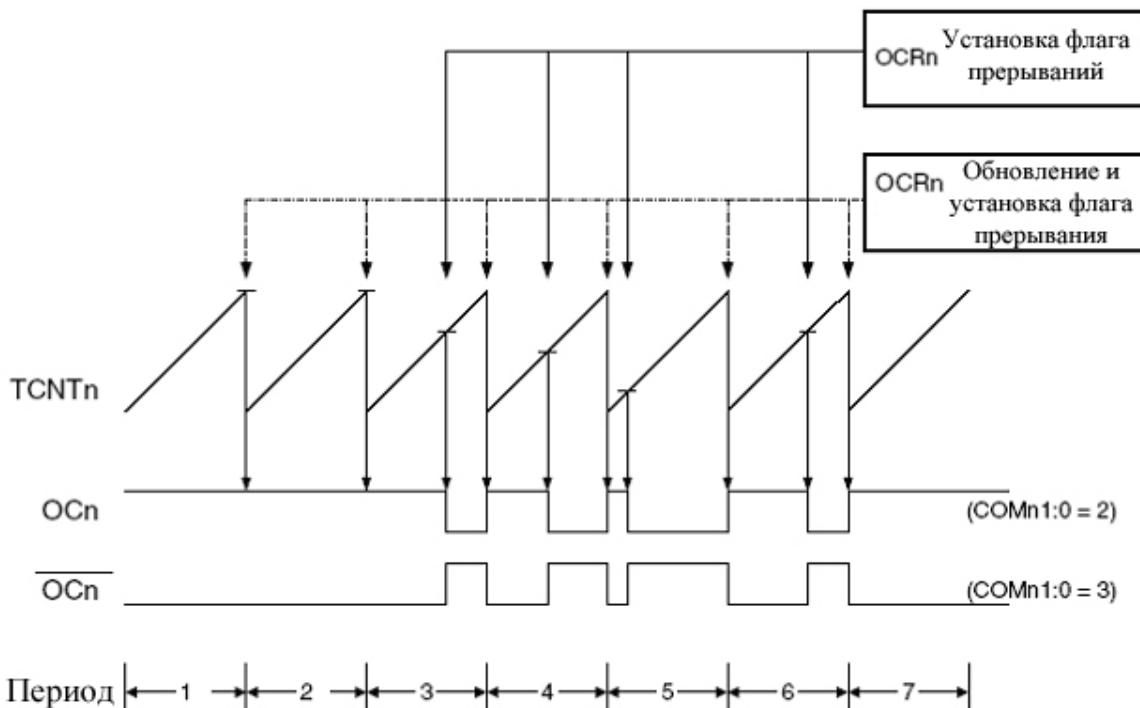


Рисунок 3.30 - Временная диаграмма для быстрого ШИМ режима

Флаг переполнения таймера/счетчика (TOV0) устанавливается каждый раз, когда счетчик достигает MAX. Если разрешено прерывание, то программа обработчика прерывания может использоваться для обновления величины сравнения.

В быстром ШИМ режиме, модуль сравнения разрешает генерацию ШИМ сигналов на выводе OC0. Установка разрядов COM01:0 на два дает не инвертированный ШИМ, а инвертированный ШИМ можно сформировать, установив COM01:0 на три. Фактическое значение OC0 будет видно на выводе выхода, если направление данных для вывода выхода установлено как выход. Форма сигнала ШИМ генерируется с помощью установки (или очистки) регистра OC0 при сравнении совпадений между OCR0 и TCNT0 и очистки (или установки) регистра OC0, во время тактового периода счетчик очищается (изменяется с MAX на BOTTOM).

Частоту ШИМ для выхода можно вычислить, используя следующее уравнение:

$$f_{OCnPWM} = \frac{f_{clk_I/O}}{N \cdot 256}$$

Переменная "N" представляет собой коэффициент предварительного масштабирования (1, 8, 64, 256 или 1024).

Крайние значения для регистра OCR0 представляют особые случаи при генерации формы сигнала ШИМ на выходе в быстром ШИМ режиме. Если OCR0 устанавливается с величиной, равной BOTTOM, то выходной сигнал будет представлять собой острый импульс для каждого тактового цикла таймера MAX+1. Если OCR0 устанавливается равным MAX, то это приводит к постоянному высокому или низкому выходу (в зависимости от полярности на выходе, которая установлена разрядами COM01:0).

Частота формы сигнала на выходе (при 50 % рабочем режиме) в быстром ШИМ режиме может быть получена путем установки OC0 на переключение своего логического уровня при каждом совпадении сравнений (COM01:0 = 1). Форма сигнала, генерируемая при этом, будет иметь максимальную частоту $f_{OC0} = f_{clk_I/O}/2$ при установке OCR0 на нуль. Эта особенность аналогична переключению OC0 в режиме СТС, за исключением того, что двойное буферирование в модуле сравнения выхода в быстром ШИМ режиме включено.

Режим фазовой коррекции ШИМ

Режим фазовой коррекции ШИМ (WGM01:0 = 1) обеспечивает опцию генерации сигнала ШИМ с фазовой коррекцией и высокой разрешающей способностью. Режим фазовой коррекции ШИМ основан на работе с использованием двух фронтов. Счетчик ведет повторяемый подсчет от BOTTOM к MAX, а затем от MAX к BOTTOM. В неинвертированном режиме сравнения OC0 очищается при совпадении между TCNT0 и OCR0 при инкременте и устанавливается при совпадении при декременте. В инвертированном режиме сравнения по выходу операция инвертируется. Принцип работы с двойным фронтом имеет пониженную максимальную рабочую частоту по сравнению с одинарным фронтом. В то же время, благодаря симметричной особенности ШИМ режимов с двойным фронтом, подобные режимы предпочтительны для использования при управлении двигателями.

Разрешение ШИМ для режима с фазовой коррекцией устанавливается на уровне 8 бит. В ШИМ режиме с фазовой коррекцией счетчик возрастает до тех пор, пока его значение не достигнет MAX. Когда счетчик достигает MAX, он меняет направление счета. Значение TCNT0 будет равно MAX для одного тактового цикла таймера. Временная диаграмма для ШИМ режима с фазовой коррекцией показана на рисунке 3.31. Значение TCNT0 во временной диаграмме показано в виде гистограммы, иллюстрирующей принцип работы с двойным наклоном. Диаграмма включает неинвертированные и инвертированные ШИМ выходы. Небольшие горизонтальные отметки на наклонах TCNT0 представляют совпадения сравнений между OCR0 и TCNT0.

Флаг переполнения таймера/счетчика (TOV0) устанавливается каждый раз, когда счетчик достигает значения BOTTOM. Флаг прерывания может использоваться для прерывания каждый раз, когда счетчик достигает значения BOTTOM.

В режиме ШИМ с фазовой коррекцией модуль сравнения обеспечивает генерацию импульсов ШИМ на выводе OC0. Установка разрядов COM01:0 на значение два вызовет неинвертированный ШИМ режим. Инвертированный выход ШИМ может генерироваться путем установки COM01:0 на три.

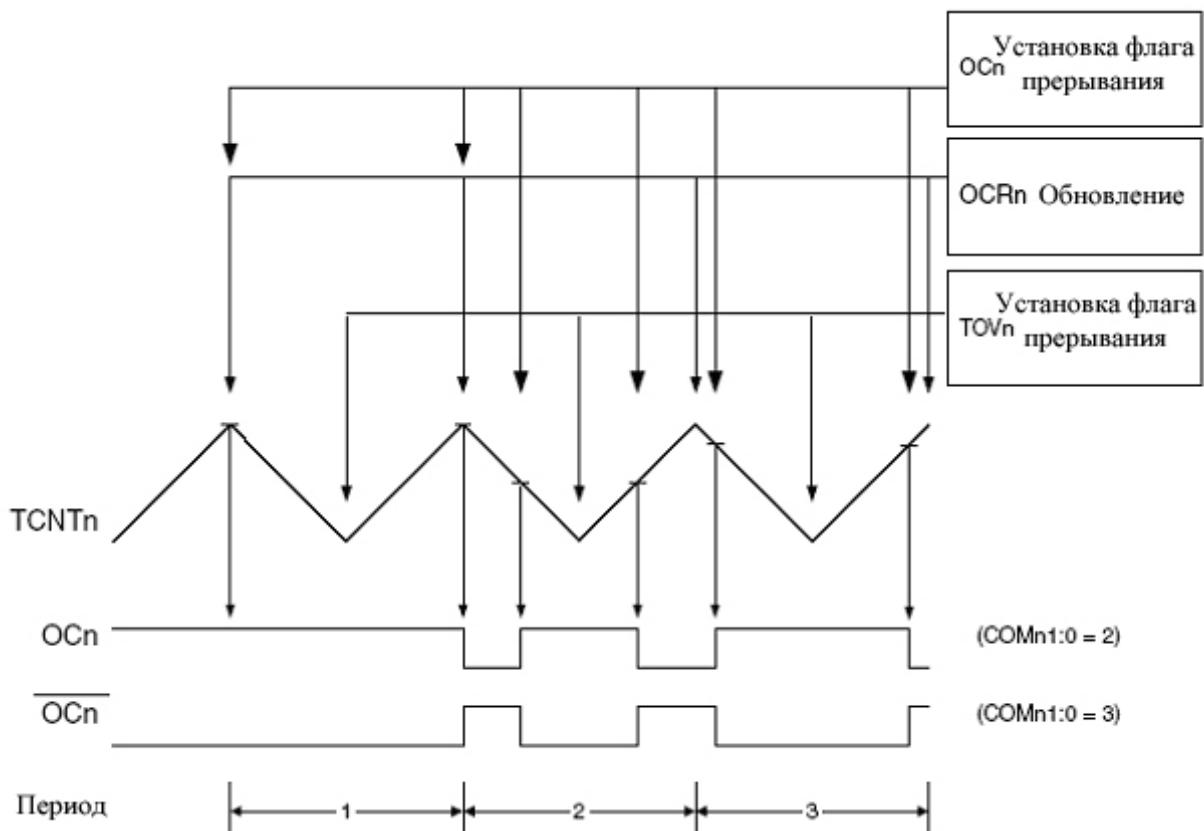


Рисунок 3.31 - Временная диаграмма, режим ШИМ с фазовой коррекцией

Фактическое значение OC0 будет видно на выводе порта только в случае, когда направление данных для вывода порта будет установлено как выход. Форма ШИМ сигнала генерируется путем сброса (или установки) регистра OC0 при совпадении между OCR0 и TCNT0, когда значения счетчика возрастают, и при установке (или сбросе) регистра OC0 при совпадении между OCR0 и TCNT0, когда значения счетчика уменьшаются. Частота ШИМ для выхода может быть вычислена с помощью следующего уравнения:

$$f_{OCnPCPWM} = \frac{f_{clk_I/O}}{N \cdot 510}$$

Переменная "N" представляет собой коэффициент предварительного деления (1, 8, 64, 256 или 1024).

Крайние значения для регистра OCR0 представляют особые случаи, когда происходит генерация выходного ШИМ сигнала в ШИМ режиме с фазовой коррекцией. Если OCR0 устанавливается равным BOTTOM, то выход будет постоянно находиться в низком состоянии, а если устанавливается равным MAX, то выход для неинвертированного ШИМ режима будет постоянно находиться в высоком состоянии. Для инвертированного ШИМ режима выход будет иметь противоположные логические значения.

Временные диаграммы таймера/счетчика

Таймер/счетчик имеет синхронный принцип работы и тактовый импульс таймера (clk_{T0}) с учетом этого показан на следующих рисунках как сигнал синхрогенератора. Рисунки включают информацию о том, когда устанавливаются флаги прерывания. На рисунке 3.32 приведены временные параметры по основному режиму работы таймера/счетчика. На рисунке приведена последовательность счета, близкая к значению MAX во всех режимах, отличающихся от режима ШИМ с фазовой коррекцией.

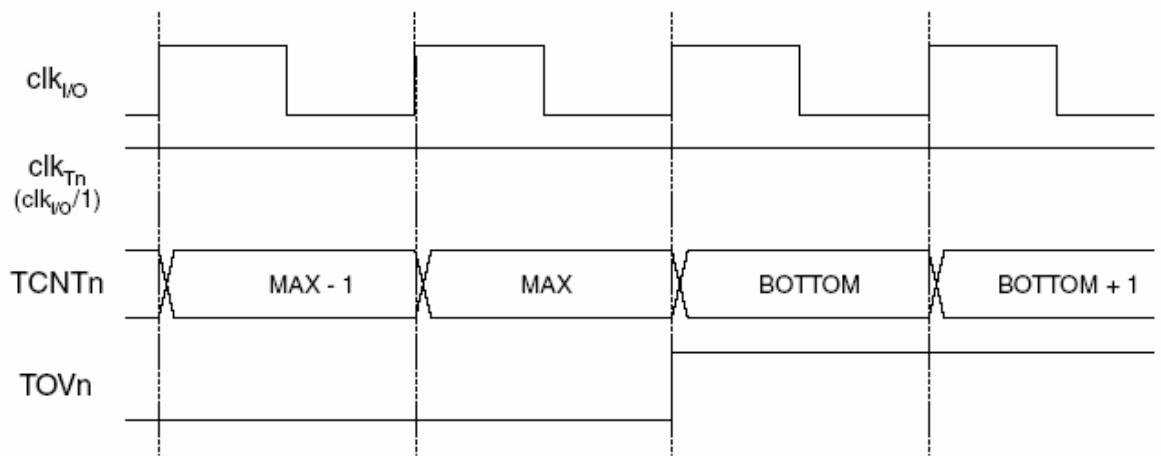


Рисунок 3.32 - Временная диаграмма таймера/счетчика без предварительного масштабирования

На рисунке 3.33 показаны те же временные параметры, но для включенного предварительного масштабирования.

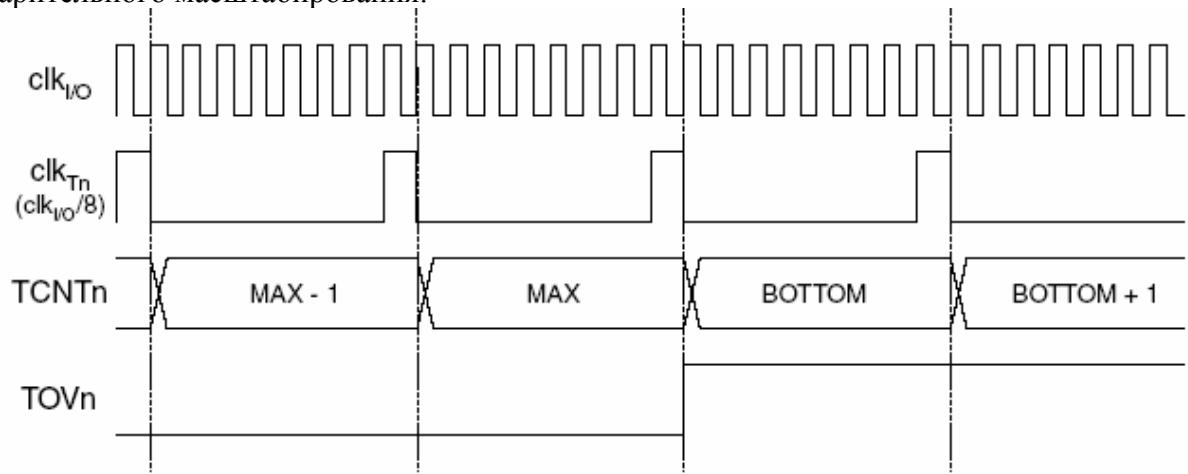


Рисунок 3.33 - Временная диаграмма таймера/счетчика с устройством предварительного масштабирования ($f_{clk_I/O}/8$)

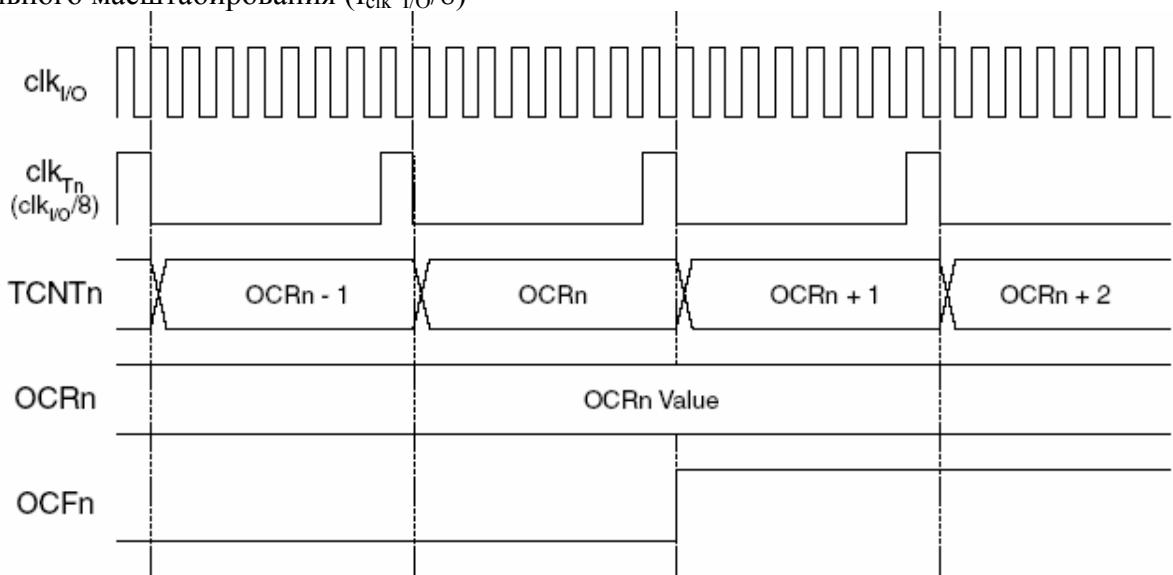


Рисунок 3.34 - Временная диаграмма таймера/счетчика с установкой OCF0 и устройством предварительного масштабирования ($f_{clk_I/O}/8$)

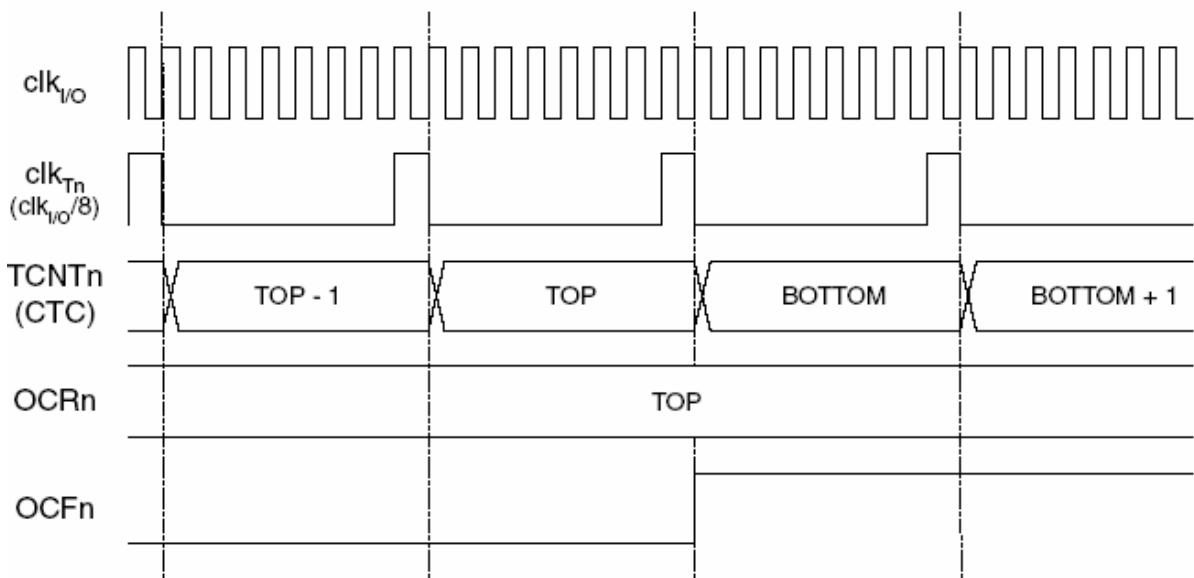


Рисунок 3.35 - Временная диаграмма таймера/счетчика, сброс таймера в режиме сравнения/совпадения, устройством предварительного масштабирования ($f_{\text{clk_I/O}}/8$)

3.10.13 Описание 8-разрядных регистров таймера/счетчика

Регистр управления таймера/счетчика – TCCR0

Бит	7	6	5	4	3	2	1	0	TCCR0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	
Чт./Зап.	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Начальное значение

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

Разряд 7: FOC0 - Принудительное сравнение выхода

Разряд FOC0 активируется только тогда, когда разряд WGM00 задает режим без использования ШИМ. В то же время, для совместимости с будущими устройствами, этот разряд должен быть установлен в ноль, когда записывается TCCR0 при работе в режиме ШИМ. При записи логической единицы в разряд FOC0, на модуль генерации формы сигнала подается команда на немедленное сравнение. Выход OC0 изменяется в соответствии с установкой его разрядов COM01:0. Следует обратить внимание на то, что разряд FOC0 реализуется в виде сигнала стробирования. С учетом этого, значение, присутствующее в разрядах COM01:0, определяет результат принудительного сравнения.

Стробирующий сигнал FOC0 не будет генерировать никакого прерывания, а также не будет очищать таймер в режиме СТС, используя OCR0 как TOP.

Разряд FOC0 всегда считывается как ноль.

Разряды 6, 3 – Режим генерации формы сигнала WGM01:0

Разряды 6, 3 управляют последовательностью счета для счетчика, источником максимального значения счетчика (TOP), а также используемым типом генерации формы сигнала. Модуль счетчика/таймера поддерживает следующие режимы работы: нормальный, режим сброса счетчика при совпадении (СТС), а также два типа режимов ШИМ.

Таблица 3.37 - Описание разряда режима работы генерации формы сигнала *

Режим	WGM01 (CTC0)	WGM00 (PWM0)	Режим работы таймера/счетчика	TOP	Обновление OCR0	Установка флага TOV0
0	0	0	Нормальный	0xFF	Немедленное	MAX
1	0	1	PWM, Коррекция фазы	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Немедленное	MAX
3	1	1	Быстрый ШИМ	0xFF	TOP	MAX

* Наименования разрядов CTC0 и PWM0 в настоящее время считаются устаревшими. Используйте определения WGM01:0, в то же время функциональность и расположение этих разрядов совместимы с прежними версиями таймера.

Разряды 5,4: COM01:0 - Режим сравнения/совпадения выхода

Эти разряды управляют поведением вывода сравнение выхода (OC0). Если устанавливается один или оба разряда COM01:0, то выход меняет нормальную функциональность порта входа/выхода. В то же время следует обратить внимание на то, что разряд регистра направления (DDR), соответствующий выводу OC0, должен быть установлен так, чтобы активировать выходной драйвер.

Когда к выводу подсоединен OC0, то функция разрядов COM01:0 зависит от установки разряда WGM01:0. В таблице 3.38 показана функциональность разряда COM01:0, когда разряды WGM01:0 устанавливаются в нормальный режим работы или в режим СТС (не режим ШИМ).

Таблица 3.38 - Режим сравнения выхода, режим без ШИМ

COM01	COM00	Описание
0	0	Нормальная работа порта, OC0 отсоединен.
0	1	Переключение OC0 на совпадение сравнений
1	0	Сброс OC0 на совпадение сравнений
1	1	Установка OC0 на совпадение сравнений

Таблица 3.39 показывает функциональность разряда COM01:0, когда разряды WGM01:0 устанавливаются на быстрый режим ШИМ.

Таблица 3.39 - Режим сравнения выхода, быстрый режим ШИМ *

COM01	COM00	Описание
0	0	Нормальная работа порта, OC0 отсоединен.
0	1	Зарезервировано
1	0	Сброс OC0 при совпадении сравнений, установка OC0 при TOP
1	1	Установка OC0 при совпадении сравнений, сброс OC0 при TOP

* Особый случай имеет место, когда OCR0 равно TOP и установлен COM01. В этом случае совпадение при сравнении игнорируется, однако установка или сброс выполняются при TOP.

Таблица 3.40 показывает функциональность разряда COM01:0, когда разряды WGM01:0 установлены на режим фазовой коррекции ШИМ.

Таблица 3.40 - Режим сравнения по выходу, режим фазовой коррекции ШИМ

COM01	COM00	Описание
0	0	Нормальная работа порта, OC0 отсоединен
0	1	Зарезервировано
1	0	Сброс OC0 при совпадении сравнений, при счете вверх Установка OC0 при совпадении сравнений при подсчете вниз
1	1	Установка OC0 при совпадении сравнений, сброс OC0 при совпадении сравнений при счете вниз
Примечание - Особый случай имеет место, когда OCR0 равно TOP и установлен COM01. В этом случае совпадение при сравнении игнорируется, однако установка или сброс выполняются при TOP.		

Разряды [2:0]: CS02:0 - Выбор синхронизации

Три разряда выбора синхронизации выбирают источник синхронизации для использования таймером/счетчиком.

Таблица 3.41 - Описание разряда выбора синхронизации

CS02	CS01	CS00	Описание
0	0	0	Нет источника синхронизации (Таймер/счетчик остановлен)
0	0	1	clk _{I/O} (без деления частоты)
0	1	0	clk _{I/O} /8 (от устройства предварительного деления частоты)
0	1	1	clk _{I/O} /64 (от устройства предварительного деления частоты)
1	0	0	clk _{I/O} /256 (от устройства предварительного деления частоты)
1	0	1	clk _{I/O} /1024 (от устройства предварительного деления частоты)
1	1	0	Внешний источник тактового сигнала на выводе T0. Импульс синхронизации на заднем фронте
1	1	1	Внешний источник тактового сигнала на выводе T0. Импульс синхронизации на переднем фронте

Если режимы подачи на внешний вывод используются для таймера/счетчика 0, то переходы на выводе T0 будут синхронизировать счетчик, даже если вывод конфигурируется как выход. Эта особенность позволяет обеспечить программное управление счетом.

Регистр таймера/счетчика – TCNT0

Бит	7	6	5	4	3	2	1	0	TCNT0
	TCNT0[7:0]								
Чт./Зап.	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр таймера/счетчика обеспечивает прямой доступ, как для операции считывания, так и записи на 8-ми разрядный счетчик блока таймера/счетчика. Запись в блоки регистра TCNT0 (удаляет) совпадение при сравнении для следующего сигнала синхронизации таймера. Изменение счетчика (TCNT0) во время работы счетчика создает риск отсутствия совпадения при сравнении между TCNT0 и регистром OCR0.

Регистр сравнения выхода – OCR0

Бит	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
Чт./Зап.	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр сравнения выхода содержит 8-разрядную величину, которая непрерывно сравнивается со значением счетчика (TCNT0). Совпадение может использоваться для генерации прерывания при сравнении по выходу или для генерации выходной формы сигнала на выводе OC0.

Регистр маски прерывания счетчика/таймера – TIMSK

Бит	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE1	TOIE0	TIMSK
Чт./Зап.	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 1: OCIE0 - Разрешение прерывания при совпадении сравнения на выходе таймера/счетчика

Если разряд OCIE0 записывается на единицу, а разряд I в регистре состояния установлен (единица), то запускается прерывание совпадения при сравнении таймера/счетчика 0. Соответствующее прерывание выполняется, если происходит совпадение при сравнении в таймере/счетчике 0 (т. е. когда разряд OCF0 установлен в регистре флага прерывания таймера/счетчика – TIFR).

Разряд 0: TOIE0 - Разрешение прерывания переполнения таймера/счетчика 0

Если разряд OCIE0 записывается на единицу, а разряд I в регистре состояния установлен (единица), то запускается прерывание переполнения таймера/счетчика 0. Соответствующее прерывание выполняется при появлении переполнения в таймере/счетчике 0 (т. е. соответствующее прерывание выполняется, если происходит переполнение в таймере/счетчике 0 (т. е. когда разряд TOV0 устанавливается в регистре флага прерывания таймера/счетчика – TIFR))

Регистр флага прерывания таймера/счетчика – TIFR

Бит	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Чт./Зап.	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 1: OCF0 - Флаг 0 сравнения выхода

Разряд OCF0 устанавливается (единица), когда происходит совпадение при сравнении между таймером/счетчиком 0 и данными с OCR0 – регистре 0 сравнения по выходу. OCF0 очищается аппаратным способом при исполнении соответствующего вектора обращения с прерываниями. В качестве альтернативы, OCF0 очищается путем записи логической единицы во флаг. Когда устанавливается разряд I в SREG, OCIE0 (разреше-

ние прерывания совпадения при сравнении таймера/счетчика 0), и OCF0 на (единицу), то исполняется прерывание при сравнении совпадения таймера/счетчика 0.

Разряд 0: TOV0 - Флаг переполнения таймера/счетчика 0

Разряд TOV0 устанавливается (единица), когда происходит переполнение таймера/счетчика 0. TOV0 очищается аппаратным способом при исполнении соответствующего вектора обращения с прерываниями. В качестве альтернативы, TOV0 очищается путем записи логической единицы во флаг. Когда устанавливаются разряды SREG I, TOIE0 (разрешение прерывания переполнения таймера/счетчика 0) и разряд TOV0 на (единицу), то исполняется прерывание при переполнении таймера/счетчика 0. В режиме фазовой коррекции ШИМ, этот разряд устанавливается когда таймер/счетчик 0 изменяет направление счета при 0x00.

3.11 Устройства предварительного деления частоты таймера/счетчика 0 и таймера/счетчика 1

Таймер/счетчик 1 и таймер/счетчик 0 используют совместно тот же самый модуль предварительного деления частоты, однако, таймеры/счетчики при этом могут иметь различные установки предварительного деления. Нижеприведенное описание применимо к обоим счетчикам – таймеру/счетчику 1 и таймеру/счетчику 0.

3.11.1 Источник внутреннего синхроимпульса

Таймер/счетчик может синхронизироваться непосредственно системным тактовым генератором (с помощью установки ($CSn2:0 = 1$). Это обеспечивает более быструю работу с максимальной тактовой частотой таймера/счетчика, равной тактовой частоте системы ($f_{clk_I/O}$). Альтернативно в качестве тактового генератора можно использовать один из четырех выводов от предварительного делителя частоты. Деление частоты позволяет получить одну из частот $f_{clk_I/O}/8$, $f_{clk_I/O}/64$, $f_{clk_I/O}/256$ или $f_{clk_I/O}/1024$.

3.11.2 Установка устройства предварительного деления частоты

Устройство предварительного деления частоты работает свободно (т.е. независимо от логики выбора частоты таймера/счетчика) и используется совместно таймером/счетчиком 1 и таймером/счетчиком 0. Поскольку на устройство предварительного деления частоты не влияет выбор тактовой частоты таймера/счетчика, то состояние устройства предварительного деления частоты будет влиять на ситуации, когда используется тактовый генератор с предварительным делением частоты. Один из примеров случая предварительного деления происходит, когда таймер запускается и синхронизируется с помощью устройства предварительного деления частоты ($6 > CSn2:0 > 1$). Число системных импульсов синхронизации с того момента, когда таймер запускается до момента, когда происходит первый подсчет, может составлять от 1 до $N+1$ системных импульсов синхронизации, где N равно коэффициенту деления предварительного делителя частоты (8, 64, 256 или 1024).

Переустановку предварительного делителя частоты можно использовать при синхронизации таймера/счетчика для исполнения программы. В то же время необходимо соблюдать осторожность, если другой таймер/счетчик, который применяет то же самое устройство предварительного деления частоты, также использует предварительное деление. Переустановка устройства предварительного деления повлияет на период устройства для всех счетчиков, которые связаны с ним.

3.11.3 Источник внешней синхронизации

Источник внешней синхронизации, прикладываемый к выводу T1/T0, может использоваться как устройство синхронизации таймера/счетчика (clk_{T1}/clk_{T0}). Вывод T1/T0 опрашивается однократно при каждом импульсе синхронизации системы с помощью логики синхронизации вывода. Синхронизированный (выбранный) сигнал затем проходит через устройство обнаружения фронта. На рисунке 3.37 показана функциональная эквивалентная блок - диаграмма логики синхронизации и обнаружения фронта T1/T0. Регистры синхронизируются при положительном фронте внутренней синхронизации системы ($clk_{I/O}$). Защелка прозрачна в период нахождения внутренней синхронизации системы в высоком состоянии. Устройство обнаружения фронта генерирует один импульс clk_{T1}/clk_{T0} для каждого положительного ($CSn2:0 = 7$) или отрицательного ($CSn2:0 = 6$) фронта, который оно обнаруживает.

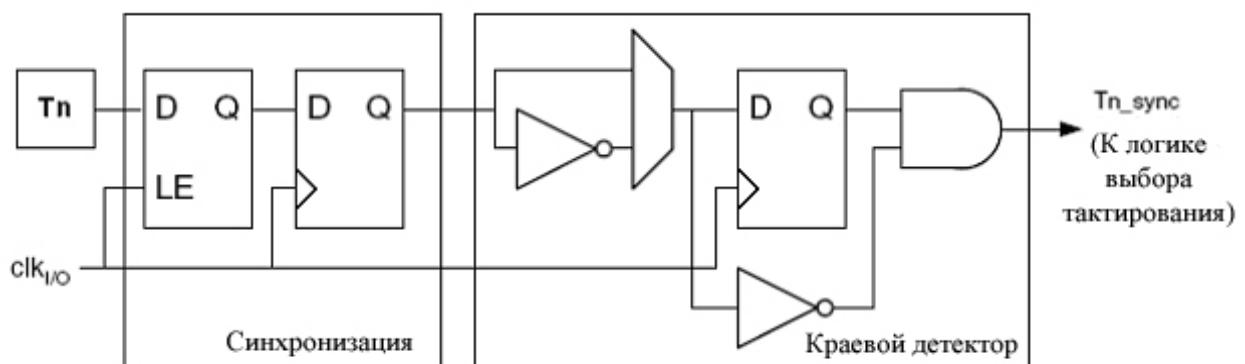


Рисунок 3.36 - Стробирование вывода T1/T0

Логика синхронизации и обнаружения фронтов вводит задержку, составляющую от 2,5 до 3,5 тактовых импульса системы от фронта, приложенного к выводу T1/T0 до счетчика, который обновляется.

Запуск и отключение входа синхронизации должны выполняться, когда T1/T0 находится в стабильном состоянии, по меньшей мере, в течение одного цикла синхронизации системы, иначе возникает риск генерации ложного импульса синхронизации таймера/счетчика.

Каждая половина периода приложенного внешнего импульса синхронизации должна превышать один один цикл синхронизации системы для обеспечения правильного стробирования. Необходимо гарантировать, что внешняя синхронизация будет иметь меньшую частоту, чем 1/2 системной частоты синхронизации ($f_{ExtClk} < f_{clk_I/O}/2$) при условии рабочего цикла (50/50) %. Поскольку устройство обнаружения фронтов использует стробирование, то максимальная частота внешней синхронизации, которую он может обнаруживать, равна половине частоты стробирования (теорема дискретизации Найквиста). В то же время из-за изменения тактовой частоты синхронизации системы и рабочего цикла, вызванных допустимыми отклонениями источника генерируемых сигналов (кварцевый резонатор и емкости), рекомендуется, чтобы максимальная частота внешнего источника синхронизации была меньше, чем $f_{clk_I/O}/2,5$.

Для источника внешнего сигнала синхронизации нельзя использовать предварительное деление частоты.

Регистр специальных функций входа/выхода – SFIOR

Бит	7	6	5	4	3	2	1	0	SFIOR
Чт./Зап.	ADTS2	ADTS1	ADTS0	ADHSM	ACME	PUD	PSR2	PSR10	
Начальное значение	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Разряд 0: PSR10 - Переустановка устройства предварительного деления частоты таймера/счетчика 1 и таймера/счетчика 0

Когда этот разряд записывается в единицу, устройство предварительного деления частоты таймера/счетчика 1 и таймера/счетчика 0 будут переустановлены. Разряд будет очищен аппаратным способом после завершения операции. Запись в этот разряд нуля не будет давать эффект. Следует обратить внимание на то, что таймер/счетчик 1 и таймер/счетчик 0 одновременно используют тот же предварительный делитель частоты и переустановка этого устройства повлияет на оба таймера. Этот разряд будет всегда считываться как нуль.

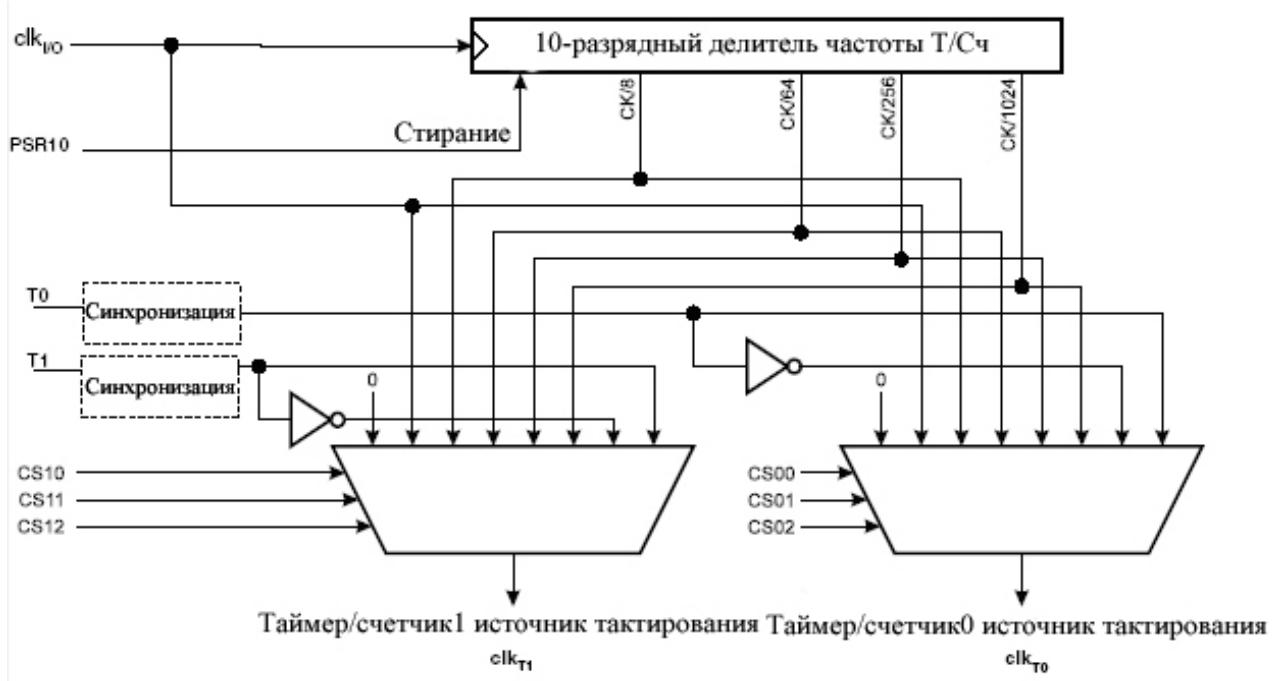


Рисунок 3.37 - Устройство предварительного деления частоты для таймера/счетчика 0 и таймера/счетчика 1

3.12 16-разрядный таймер/счетчик 1

Модуль 16-разрядного таймера/счетчика обеспечивает точное исполнение программы по времени, (управление событиями), генерацию формы сигнала и измерение временных параметров сигналов. Устройство имеет следующие основные отличительные особенности:

- Истинную 16-разрядную компоновку (т. е. допускает использование 16-разрядного ШИМ).
- Два независимых блока сравнения по выходу.
- Регистры сравнения по выходу с двойным буфером.
- Один блок захвата по входу.
- Устройство подавления шума захвата по выходу.
- Сброс таймера при совпадении сравнений (автоматическая перезагрузка).
- Надежный ШИМ модулятор с фазовой коррекцией.
- Регулируемый ШИМ период.
- Генератор частоты.
- Счетчик внешних событий.
- Четыре независимых источника прерывания (TOV1, OCF1A, OCF1B и ICF1).

Обзор

Основные ссылки на регистры и разряды в данном разделе представлены в общем виде. Буква “n” в нижнем регистре заменяет номер таймера/счетчика, а буква “x” в нижнем регистре заменяет номер канала сравнения по выходу. В то же время когда использование регистра или разряда определено в программе, необходимо использовать точную форму, например, TCNT1 для значения доступа к таймеру/счетчику 1 и т. д.

Регистры

Таймер/счетчик (TCNT1), регистры сравнения выводов (OCR1A/B) и регистр захвата на входе (ICR1) - все являются 16-разрядными регистрами. При доступе к 16-разрядным регистрам необходимо соблюдать специальные процедуры. Регистры команд таймера/счетчика (TCCR1A/B) являются 8-разрядными регистрами и не имеют никаких ограничений при доступе из ЦПУ. Запросы на прерывание (на рисунке 3.38 сокращенно Int. Req.) все видны в регистре флага прерывания таймера (TIFR). Все прерывания замаскированы индивидуально с использованием регистра маскирования прерывания таймера (TIMSK). TIFR и TIMSK не указываются в цифре, поскольку эти регистры используются одновременно другими модулями таймера.

Таймер/счетчик может быть синхронизирован внутри схемы через делитель частоты или с помощью источника внешнего таймера на выводе T1. Логическая схема выбора тактового генератора определяет, какой источник синхронизации и фронт используют таймер/счетчик, чтобы увеличить (или уменьшить) его значение. Таймер/счетчик не активен, если не выбран никакой источник синхронизации. Выход от логической схемы выбора тактового генератора называется тактовым генератором таймера (clk_{T1}).

Регистры сравнения выхода с двойным буфером (OCR1A/B) постоянно сравниваются со значением таймера/счетчика. Результат сравнения может использоваться генератором формы сигнала для создания режима ШИМ или сигнала с переменной частотой на выводе сравнения выхода (OC1A/B). Событие совпадения при сравнении также установит флаг соответствия при сравнении (OCF1A/B), который может использоваться для создания запроса на прерывание при сравнении по выходу.

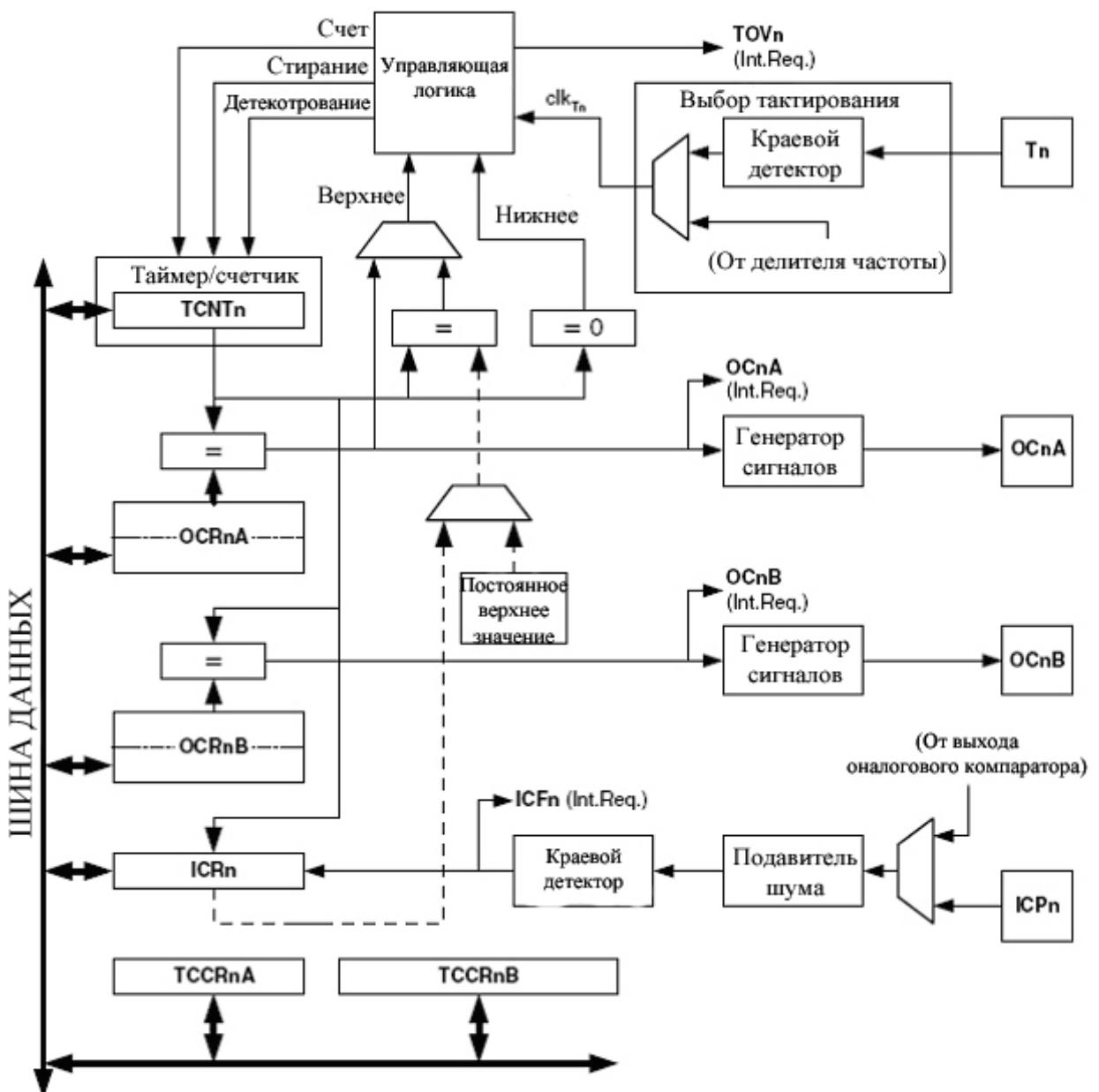


Рисунок 3.38 – Блок-схема 16-разрядного таймера/счетчика

Регистр захвата по входу может фиксировать значение таймера/счетчика для данного внешнего события (запускаемого фронтом) либо на выводе Захвата по входу (ICP1), либо на выводе аналогового компаратора (см. "Аналоговый компаратор" подраздел 3.17). Модуль захвата по входу включает в себя блок цифровой фильтрации (подавитель шума), уменьшающий возможность захвата шумовых выбросов.

Значение TOP или максимальное значение таймера/счетчика может в некоторых режимах работы определяться Регистром OCR1A или Регистром ICR1, либо набором установленных значений. При использовании OCR1A как TOP - значение в режиме ШИМ регистр OCR1A не может быть использован для генерации выхода ШИМ. В то же время, значение TOP в данном случае будет дважды буферировано, что позволяет изменять величину TOP в ходе работы. Если потребуется фиксированное значение TOP, то в качестве альтернативы можно использовать регистр ICR1, освобождая OCR1A для использования в качестве выхода ШИМ.

Определения

В данном документе достаточно часто используются следующие определения:

Таблица 3.42 - Определения

BOTTOM	Счетчик достигает значения BOTTOM, 0x0000.
MAX	Счетчик достигает своего значения MAXimum когда оно становится равным 0xFFFF (десятичное значение 65535).
TOP	Счетчик достигает значения TOP, когда оно становится равным наивысшему значению в последовательности подсчета. Значение TOP может быть приписано одному из фиксированных значений: 0x00FF, 0x01FF, или 0x03FF, или же значению, сохраненному в регистре OCR1A или ICR1. Сама операция присвоения зависит от режима работы.

Доступ к 16-разрядным регистрам

TCNT1, OCR1A/B и ICR1 - 16-разрядные регистры, к которым может обращаться ЦПУ через 8-битовую шину данных. Доступ к 16-разрядному регистру должен осуществляться по байтам, используя две операции считывания или записи. У каждого 16-разрядного таймера есть один 8-битовый регистр для временного хранения старшего байта 16-разрядного доступа. Тот же самый временный регистр используется совместно всеми 16-разрядными регистрами в пределах каждого 16-разрядного таймера. Доступ к младшему байту запускает операцию 16-разрядного считывания или записи. Когда младший байт 16-разрядного регистра записывается ЦПУ, то старший байт, сохраненный во временном регистре, и записанный младший байт оба копируются в 16-разрядный регистр в том же самом тактовом цикле. Когда младший байт 16-разрядного регистра считывается ЦПУ, то старший байт 16-разрядного регистра копируется во временный регистр в тот же самом тактовом цикле, пока считывается младший байт.

Не все 16-разрядные доступы используют временный регистр для старшего байта. При считывании OCR1A/B 16-разрядные регистры не используют временный регистр.

Чтобы выполнить 16-разрядную запись, старший байт должен быть записан перед младшим байтом. При 16-разрядном считывании младший байт должен читаться перед старшим байтом.

Следующие примеры кода показывают как обратиться к 16-разрядным Регистрам Таймера, полагая при этом, что никакие прерывания не обновляют временный регистр. Тот же самый принцип может использоваться непосредственно для доступа к регистрам OCR1A/B и ICR1. Следует обратить внимание на то, что при использовании “C”, компилятор обрабатывает 16-разрядный доступ.

Assembly Code Examples ⁽¹⁾

```

...
; Set TCNT1 to 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNT1H,r17
out TCNT1L,r16
; Read TCNT1 into r17:r16
in r16,TCNT1L
in r17,TCNT1H
...

```

C Code Examples ⁽¹⁾

```

unsigned int i;
...
/* Set TCNT1 to 0x01FF */
TCNT1 = 0x1FF;
/* Read TCNT1 into i */
i = TCNT1;
...

```

Примечание - Данный пример кода предполагает, что включен характерный для устройства заголовочный файл.

Пример кода ассемблера возвращает значение TCNT1 в пару регистра r17:r16.

Важно также отметить, что доступ к 16-разрядным регистрам являются элементарными операциями. Если происходит прерывание между двумя командами при доступе к 16-разрядному регистру и код прерывания обновляет временный регистр путем доступа к тому же самому или любому другому из 16-разрядных регистров таймера, то результат доступа за пределами прерывания будет искажен. Следовательно, когда оба кода – основной код и код прерывания обновляют регистр временного хранения, то основной код должен отключать прерывания во время 16-битного доступа.

Ниже приводятся примеры кодов, показывающие, как выполнять элементарное считывание содержимого регистра TCNT1. Считывание любого из регистров OCR1A/B или ICR1 может выполняться по тому же принципу.

Assembly Code Example ⁽¹⁾

```

TIM16_ReadTCNT1:
    ; Save Global Interrupt Flag
    in r18,SREG
    ; Disable interrupts
    cli
    ; Read TCNT1 into r17:r16
    in r16,TCNT1L
    in r17,TCNT1H
    ; Restore Global Interrupt Flag
    out SREG,r18
    ret

```

C Code Example ⁽¹⁾

```

unsigned int TIM16_ReadTCNT1( void )
{
    unsigned char sreg;
    unsigned int i;
    /* Save Global Interrupt Flag */
    sreg = SREG;
    /* Disable interrupts */
    _CLI();
    /* Read TCNT1 into i */
    i = TCNT1;
    /* Restore Global Interrupt Flag */
    SREG = sreg;
    return i;
}

```

Примечание - Данный пример кода предполагает, что включен характерный для устройства заголовочный файл.

Пример кода ассемблера возвращает значение TCNT1 в пару регистра r17:r16.

Следующие примеры кодов показывают, как выполнять поэлементную запись в содержимое регистров TCNT1. Запись любого из регистров OCR1A/B или ICR1 может быть выполнена с использованием того же принципа.

Assembly Code Example ⁽¹⁾

```

TIM16_WriteTCNT1:
    ; Save Global Interrupt Flag
    in r18,SREG
    ; Disable interrupts
    cli
    ; Set TCNT1 to r17:r16
    out TCNT1H,r17
    out TCNT1L,r16
    ; Restore Global Interrupt Flag
    out SREG,r18
    ret

```

C Code Example ⁽¹⁾

```

void TIM16_WriteTCNT1( unsigned int i )
{
    unsigned char sreg;
    /* Save Global Interrupt Flag */
    sreg = SREG;
    /* Disable interrupts */
    _CLI();
    /* Set TCNT1 to i */
    TCNT1 = i;
    /* Restore Global Interrupt Flag */
    SREG = sreg;
}

```

Примечание - Данный пример кода предполагает, что включен характерный для устройства заголовочный файл.

Пример кода ассемблера требует, чтобы пара регистров r17:r16 содержала значение, которое должно быть записано в TCNT1.

Повторное использование временного регистра старшего байта

При записи более, чем в один 16-разрядный регистр, в котором одинаков для всех записываемых регистров, старший разряд необходимо записать только один раз. В то же время, необходимо обратить внимание на то, что ранее описанный принцип поэлементной работы применим и в этом случае.

Источники синхронизации таймера/счетчика

Таймер/счетчик можно синхронизировать с помощью внутреннего или внешнего источника синхронизации. Источник синхронизации выбирается с помощью логики выбора синхронизации, которая управляется разрядами выбора синхронизации (CS12:0), расположенными в Регистре В управления таймером/счетчиком (TCCR1B).

Модуль счетчика

Основной частью 16-битного таймера/счетчика является программируемый модуль двунаправленного 16-битного счетчика.

На рисунке 3.39 показана блок-схема - счетчика и его устройств.

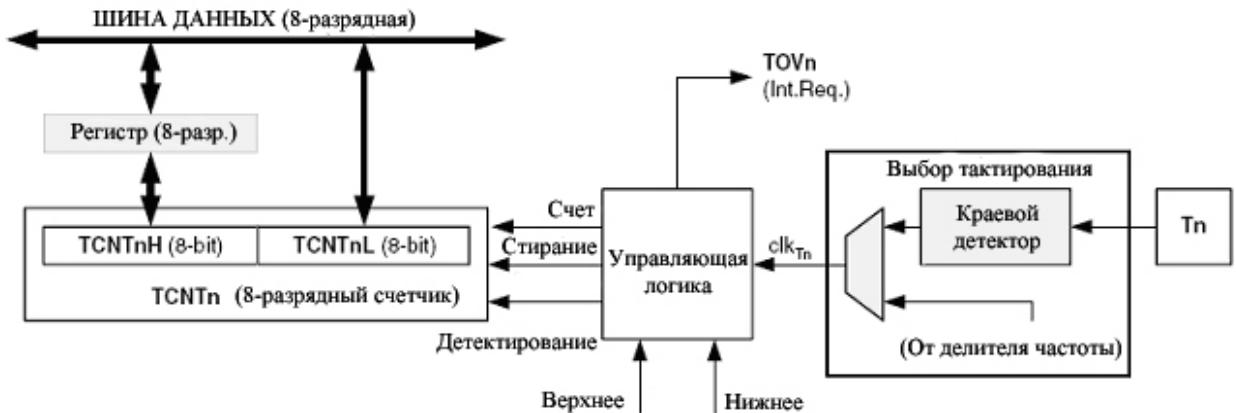


Рисунок 3.39 – Блок-схема модуля счетчика

Описание сигналов (внутренние сигналы):

- | | |
|-------------------|---|
| Count | - возрастание или уменьшение TCNT1 на ; |
| Direction | - выбор между возрастанием или убыванием; |
| Clear | - сброс TCNT1 (установка всех разрядов на ноль); |
| clk _{T1} | - тактовый генератор таймера/счетчика; |
| TOP | - указывает на то, что TCNT1 достиг максимального значения; |
| BOTTOM | - указывает на то, что TCNT1 достиг минимального значения (ноль). |

16-разрядный счетчик отображается в двух 8-битовых адресах памяти ввода - вывода: счетчик высокого уровня (TCNT1H), содержащий верхние восемь битов счетчика, и Счетчика Низкого уровня (TCNT1L), содержащего более низкие восемь битов. К регистру TCNT1H можно обратиться с помощью ЦПУ только косвенно. Когда центральный процессор осуществляет доступ к адресам ввода - вывода TCNT1H, центральный процессор обращается к (ВРЕМЕННОМУ) регистру временного хранения старшего байта. Регистр временного хранения обновляется на значение TCNT1H при считывании TCNT1L и TCNT1H обновляется на значение регистра временного хранения, когда TCNT1L записывается. Это позволяет центральному процессору читать или записывать все значения 16-разрядного счетчика в пределах одного тактового цикла через 8-битовую шину данных. Важно обратить внимание на то, что есть особые случаи записи Регистра TCNT1, когда счетчик подсчитывает непредсказуемые результаты. Особые случаи описаны в разделах, для которых они важны.

В зависимости от используемого режима работы, счетчик сбрасывается, увеличивает или уменьшает свои значения при каждом такте синхронизации таймера (clk_{T1}). Такт clk_{T1} может быть сформирован либо от внешнего, либо от внутреннего источника тактового генератора и выбирается битами выбора тактового генератора (CS12:0). Если не выбран никакой из источников (CS12:0 = 0), таймер останавливается. Однако величина TCNT1 может быть оценена ЦПУ независимо от того, присутствует clk_{T1} или нет. Запись ЦПУ отменяет (имеет приоритет над ними) все операции сброса или подсчета. Последовательность подсчета определяется установкой разрядов режима генерации формы сигнала (WGM13:0), расположенных в регистрах А и В таймера/счетчика команд (TCCR1A и TCCR1B). Существует выраженная взаимосвязь между тем, как счетчик ве-

дет себя (подсчитывает) и тем, какие формы волн генерируются на выводах сравнения по выходу OC1x. Флаг переполнения таймера/счетчика (TOV1) устанавливается согласно режиму работы, который выбирается разрядами WGM13:0. TOV1 может использоваться для генерации прерывания ЦПУ.

Модуль сбора данных на входе

Таймер/счетчик включает модуль сбора данных на входе, который может фиксировать внешние события и присваивать им временную метку, указывающую время поступления. Внешний сигнал, указывающий на событие или множественные события, может быть подан через вывод ICP1 или, в качестве альтернативы, через модуль аналогового компаратора. Временные метки могут тогда использоваться для вычисления частоты, длительности рабочего цикла и других особенностей поступающего сигнала. Альтернативно временные метки могут быть использованы для создания файла регистрации событий.

Модуль сбора данных на входе проиллюстрирован блок-схемой, показанной на рисунке 3.40. Элементы на блок-схеме, которые не являются непосредственно частью модуля кадра, затушеваны полутенью. Маленькая буква “n” в названиях регистров и разрядов указывает номер таймера/счетчика.

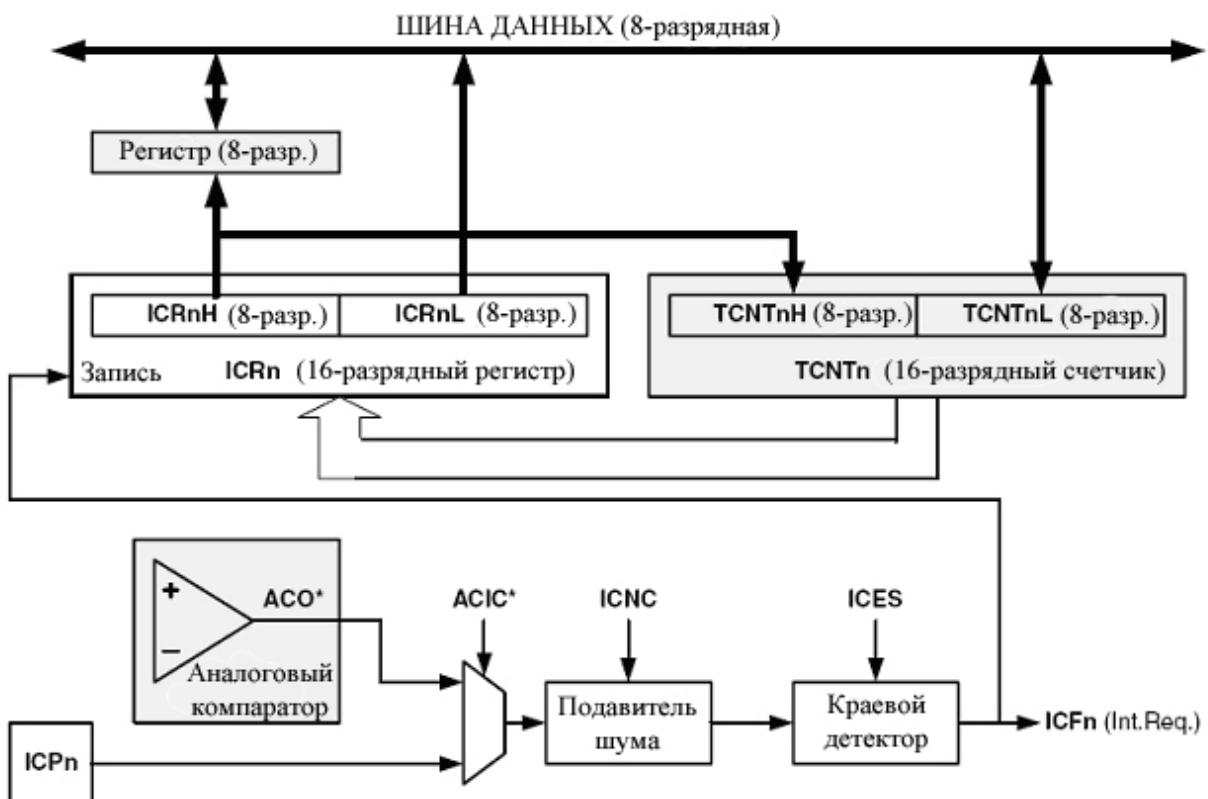


Рисунок 3.40 - Блок-схема модуля сбора данных на входе

Когда изменение логического уровня (событие) происходит на выводе сбора данных на входе (ICP1), или, альтернативно, на выводе аналогового компаратора(ACO), и если это изменение подтверждает установку устройства обнаружения фронта, то захват информации будет запущен. Когда захват запущен, то 16-разрядное значение счетчика (TCNT1) записывается в регистр захвата информации на входе (ICR1). Флаг сбора данных на входе (ICF1) устанавливается во время того же системного импульса синхронизации, поскольку значение TCNT1 копируется в регистр ICR1. Если получено разрешение, (TICIE1 = 1), то флаг сбора данных на входе генерирует прерывание захвата. Флаг

ICF1 автоматически сбрасывается при исполнении прерывания. В качестве варианта флаг ICF1 может быть очищен программным способом путем записи логической единицы в адрес бита ввода – вывода.

Считывание 16-разрядного значения во входном регистре кадра (ICR1) выполняется путем считывания сначала младшего байта (ICR1L), а затем старшего байта (ICR1H). Когда младший байт считывается, старший байт копируется во временный регистр (TEMP) старшего байта. Когда ЦПУ считывает адрес ввода - вывода ICR1H, то оно обращается к Регистру TEMP.

Регистр ICR1 может записываться только, применяя режим генерации формы сигнала, который использует регистр ICR1 для определения значения счетчика TOP. В этих случаях разряды для режима генерации формы сигнала (WGM13:0) должны быть установлены до того, как значение TOP может быть записано в регистр ICR1. При записи регистра ICR1 Register, старший байт должен быть записан в адрес ICR1H I/O до того, как младший байт записывается в ICR1L.

Источник запуска сбора данных на входе

Основным источником запуска для модуля сбора данных на входе является вывод входа для сбора данных (ICP1). Таймер/счетчик 1 может в качестве альтернативы использовать выход аналогового компаратора в качестве источника запуска для модуля сбора данных на входе. Выбор аналогового компаратора в качестве источника запуска производится путем установки бита сбора данных на входе аналогового компаратора (ACIC) в регистре анализа состояния и управления аналоговым компаратором (ACSR). Следует убедиться в том, что изменение источника запуска действительно может запускать сбор данных. Флаг сбора данных на входе должен с учетом этого быть сброшен.

Оба вывода – сбора данных на входе (ICP1) и выводы выхода аналогового компаратора (ACO) опрашиваются с использованием той же методики, что и для вывода T1. Устройство обнаружения фронтов также идентично. В то же время, когда устройство шумоподавления запускается, перед устройством обнаружения фронта встраивается дополнительная логика, которая увеличивает задержку на четыре цикла системной синхронизации. Следует обратить внимание на то, что вход устройства шумоподавления и устройство обнаружения фронта всегда разрешены до тех пор, пока таймер/счетчик устанавливается в режим генератора формы сигнала, который использует ICR1 для определения состояния TOP.

Сбор данных на входе может быть запущен программным способом путем управления портом вывода ICP1.

Устройство шумоподавления

Устройство шумоподавления повышает помехоустойчивость благодаря применению простой схемы цифровой фильтрации. Сложение за входом устройства шумоподавления ведется в течение четырех выборок, при этом все четыре выборки должны быть одинаковыми для смены выхода, что в свою очередь используется устройством обнаружения фронта.

Устройство шумоподавления запускается путем установки разряда устройства шумоподавления для входа сбора данных (ICNC1) в регистре В управления таймером/счетчиком (TCCR1B). При запуске устройство шумоподавления вводит четыре дополнительных системных тактовых сигнала задержки с помощью изменений, прикладываемых к выходу, для обновления регистра ICR1. Устройство шумоподавления использует системный сигнал синхронизации и поэтому на него не влияет предварительный делитель частоты.

Использование модуля сбора информации на входе

Основной проблемой при использовании модуля сбора информации на входе является обеспечение достаточной производительности процессора для обработки поступающих результатов. Время между наступлением двух событий является критически важным. Если процессор не успел считать значение в регистре ICR1 до наступления следующего события, то значение ICR1 будет перезаписано на новое. В этом случае результат сбора данных будет некорректным.

При использовании прерывания при сборе данных, регистр ICR1 должен считываться как можно раньше во время процедуры обработки прерываний. Даже если прерывание при сборе данных на входе будет иметь относительно высокий приоритет, то максимальное время отклика при прерывании зависит от максимального числа тактовых импульсов, требуемых для обработки любого из прочих запросов на прерывание.

Использование модуля сбора информации на входе в любом режиме работы, если значение TOP (разрешающая способность) активно изменяется во время работы, не рекомендуется.

Измерение рабочего цикла внешнего сигнала требует, чтобы фронт триггера изменился после каждого сбора информации. Изменение обнаружения фронта должно выполняться как можно раньше после считывания регистра ICR1. После изменения фронта флаг сбора информации на входе (ICF1) должен очищаться программным способом (с помощью записи логической единицы в адрес разряда ввода-вывода). Если выполняется только измерение частоты, то сброс флагжа ICF1 не требуется (при условии использования устройства обращения с прерываниями).

Модули сравнения входа

16-разрядный компаратор постоянно сравнивает TCNT1 с регистром сравнения выхода (OCR1x). Если TCNT равно OCR1x, то компаратор выдает сигнал о совпадении. Это совпадение установит флаг сравнения по выходу (OCF1x) во время следующего цикла синхронизации таймера. При получении разрешения (OCIE1x = 1) флаг сравнения выхода генерирует прерывание сравнения на выходе. Флаг OCF1x автоматически сбрасывается, при выполнении прерывания. В качестве альтернативы флаг OCF1x может быть сброшен программно путем записи логической единицы в адрес его разряда Входа/Выхода. Генератор формы сигнала использует сигнал совпадения для генерации выхода в соответствии с режимом работы, который установился соответствующими разрядами режима генерации формы сигнала (WGM13:0) и разрядами режима сравнения по выходу (COM1x1:0). Сигналы TOP и BOTTOM используются генератором формы сигнала для обработки особых случаев экстремальных значений для некоторых режимов работы.

Специальная функция модуля А сравнения по выходу позволяет ему определять величину TOP таймера/счетчика (т. е. разрешающую способность). В дополнение к разрешающей способности величина TOP определяет период времени для форм сигнала, генерируемых генератором формы сигнала.

На рисунке 3.41 показана блок-схема модуля сравнения по выходу. Маленькая буква “n” в названии регистра и бита указывает на номер устройства ($n = 1$ для таймера/счетчика 1), а “x” указывает на модуль сравнения по выходу (A/B). Элементы блок-схемы, которые не входят непосредственно в модуль сравнения по выходу, залиты серым фоном.

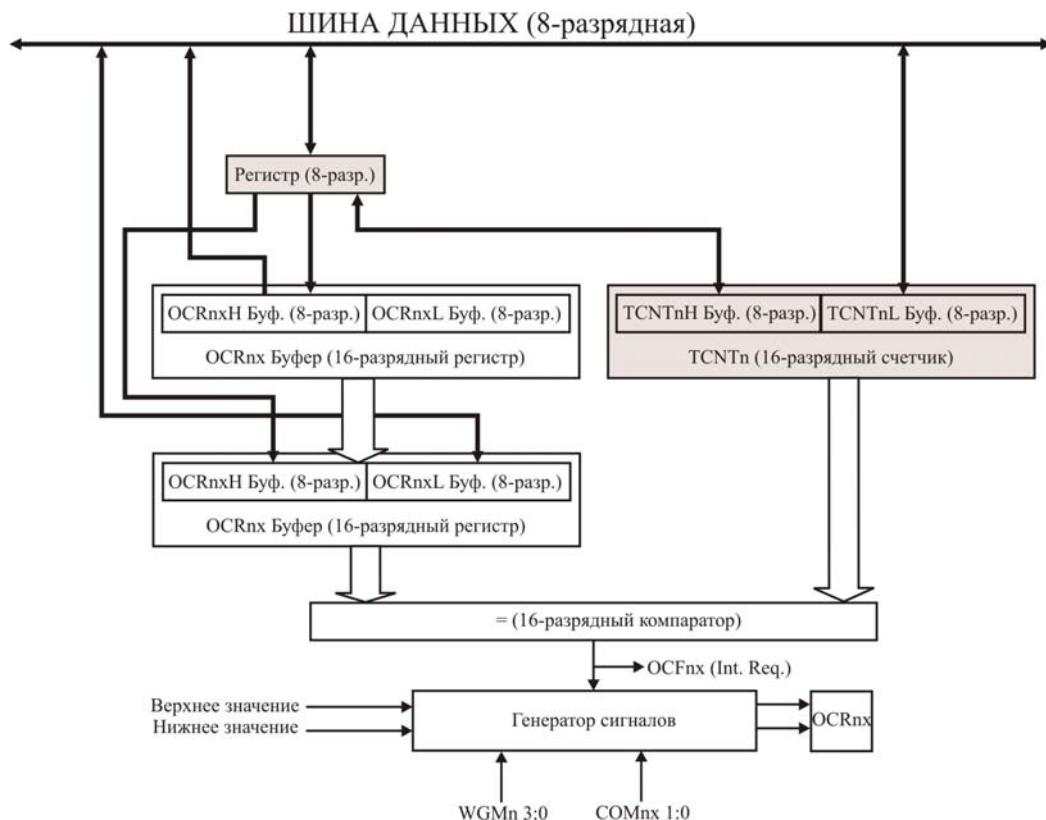


Рисунок 3.41 – Блок-схема модуля сравнения по выходу

Регистр OCR1x имеет двойное буфериование при использовании любого из двенадцати режимов ШИМ. В нормальном режиме и режиме очистки таймера при сравнении (режим СТС) двойное буфериование отключается. Двойное буфериование синхронизирует обновление регистра сравнения до ТОР или BOTTOM в последовательности считывания. Синхронизация предотвращает появление несимметричных импульсов избыточной длины, обеспечивая на выходе сигналы, свободные от помех.

Доступ к регистру OCR1x может показаться сложным, тем не менее, это не так. При запуске двойного буфериования ЦПУ получает доступ к регистру буфера OCR1x, а если двойное буфериование будет отключено, то ЦПУ получит прямой доступ к OCR1x. Содержимое регистра OCR1x (буфер или сравнение) изменяется только при операции записи (таймер/счетчик не обновляет этот счетчик автоматически, так как это выполняет TCNT1 – и регистр ICR1 Register). Вследствие этого OCR1x не считывается через регистр временного хранения старшего байта (TEMP). Однако, исходя из нормальной практики, следует сосчитать сначала младший байт, также как при доступе к 16-разрядным регистрам. Запись в регистры OCR1x должна выполняться через регистр TEMP, поскольку сравнение всех 16 разрядов выполняется непрерывно. Сначала должен записываться старший байт (OCR1xH). Когда ЦПУ записывает адрес входа/выхода старшего байта, содержимое регистра TEMP будет обновлено на записанное значение. Затем, когда в нижние восемь разрядов записывается младший байт (OCR1xL), старший байт будет скопирован в верхние восемь разрядов либо буфера OCR1x, либо регистра сравнения OCR1x в течение того же цикла синхронизации.

Принудительное сравнение по выходу

При работе, не относящихся к ШИМ режимов сравнение выхода компаратора может быть принудительным при записи единицы в разряд принудительного сравнения по выходу (FOC1x). Принудительное совпадение не установит флаг OCF1x и не перегрузит/очистит таймер, но вывод OC1x будет обновлен и, если произошло реальное совпадение при сравнении (установке) разрядов COM11:0, будут определять: установлен ли вывод OC1x, очищен или переключен.

Блокировка совпадения при сравнении с помощью записи TCNT1

Все записи ЦПУ в регистр TCNT1 будут блокировать любое совпадение, которое происходит в следующем цикле синхронизации таймера, даже при условии, что таймер остановлен. Эта особенность позволяет инициализировать OCR1x до того же самого значения, что и TCNT1 без запуска прерывания, когда активирован тактовый генератор таймера/счетчика.

Использование модуля сравнения по выходу

Поскольку запись TCNT1 в любом режиме работы блокирует все совпадения при сравнении для одного тактового цикла таймера, то существуют риски, связанные с изменениями TCNT1 при использовании любого из каналов сравнения по выходу, и это не зависит от того, работает ли таймер/счетчик или нет. Если значение, записанное в TCNT1, будет равняться значению OCR1x, то совпадение при сравнении будет пропущено, что приведет к генерации сигнала неправильной формы. Не следует записывать величину TCNT1, равную TOP, в режимах ШИМ с переменными значениями TOP. Совпадение при сравнении для TOP будет проигнорировано и подсчет продолжится до 0xFFFF. Точно так же не следует записывать значение TCNT1, равное BOTTOM, когда счетчик считает вниз.

Установка OC1x должна быть выполнена до настройки регистра направления данных для вывода порта для вывода данных. Самый простой способ установки значения OC1x состоит в том, чтобы использовать биты стробирования принудительного сравнения по выходу (FOC1x) в нормальном режиме. Регистр OC1x сохраняет свое значение, даже если он переходит в различные режимы генерации формы сигнала.

Необходимо убедиться в том, что биты COM1x1:0 не имеют двойной буферизации совместно со значением сравнения. Изменение битов COM1x1:0 немедленно вступит в силу.

Модуль сравнения/совпадения по выходу

Разряды режима сравнения по выходу (COM1x1:0) имеют две функции. Генератор формы сигнала использует разряды COM1x1:0 для определения состояния сравнения по выходу (OC1x) при последующем совпадении при сравнении. Вторая функция состоит в том, что разряды COM1x1:0 управляют источником выхода для вывода OC1x. На рисунке 3.42 показана упрощенная логическая схема, запускаемая с помощью установки разряда COM1x1:0. Полужирным шрифтом на рисунке 3.42 показаны регистры, разряды и выводы входа/выхода. В данном случае показаны только те части общих регистров управления порта входа/выхода (DDR и PORT), на которые воздействуют разряды COM1x1:0. При ссылке на состояние OC1x, эта ссылка относится к внутреннему регистру OC1x, а не к выводу OC1x. Если происходит переустановка системы, то регистр OC1x переустанавливается на “0”.

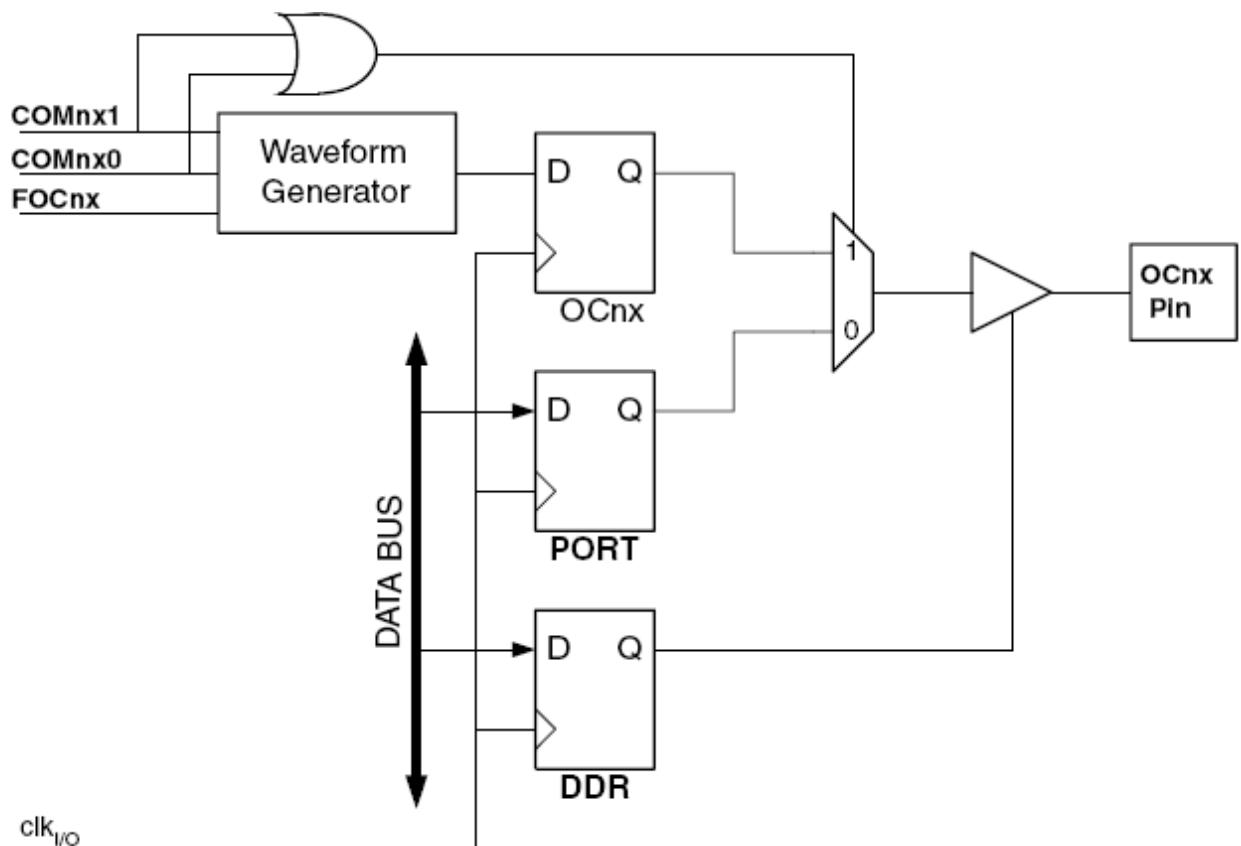


Рисунок 3.42 - Схема модуля сравнения/совпадения по выходу

Общая функция порта входа/выхода корректируется с помощью сравнения по выходу (OC1x), используя генератор формы сигнала, если установлен любой из разрядов COM1x1:0. Однако для вывода порта направление вывода OC1x (вход или выход) по-прежнему контролируется регистром направления данных (DDR). Бит регистра направления данных для вывода OC1x (DDR_OC1x) должен быть установлен как выход до того, как значение OC1x станет присутствовать на выводе. Функция отмены порта в общем виде не зависит от режима генерации формы сигнала, но при этом есть некоторые исключения.

Конструкция логики для вывода сравнения позволяет выполнять инициализацию состояния OC1x до получения разрешения для вывода. Следует обратить внимание на то, что некоторые установки разрядов COM1x1:0 резервируются для определенных режимов работы.

Биты COM1x1:0 никак не влияют на модуль сбора данных на входе.

Режим сравнения по выходу и режим генерации формы сигнала

Генератор формы сигнала использует биты COM1x1:0 по-разному в режимах: нормальный, условного перехода (CTC) и режиме широтно-импульсной модуляции. Для всех режимов установка значений COM1x1:0 = 0 сообщает генератору формы сигнала о том, что в отношении регистра OC1x не следует выполнять никаких действий при следующем совпадении при сравнении. Изменение состояния разрядов COM1x1:0 повлияет на первое совпадение сравнений после того, как биты записаны. Для режимов, не относящихся к ШИМ, действие может быть усилено для осуществления немедленного воздействия путем использования разрядов стробирования FOC1x.

Режимы работы

Режим работы, то есть поведение на выводах таймера/счетчика и выводе сравнения по выходу, определяется комбинацией разрядов для режима (WGM13:0) генерации формы сигнала и режима сравнения по выходу (COM1x1:0). Разряды режима сравнения по выходу не затрагивают последовательность подсчета, в то время как разряды режима генерации формы сигнала влияют на него. Разряды COM1x1:0 определяют, должен ли быть инвертирован или нет сгенерированный выходной ШИМ сигнал. Для режимов без ШИМ биты COM1x1:0 управляют действиями в отношении того, должен ли созданный выход ШИМ быть установлен, очищен или переключен на совпадение при сравнении.

Обычный режим работы

Самым простым режимом работы является обычный режим (WGM13:0 = 0). В этом режиме направление счета всегда выполняется по возрастанию, и при этом не выполняется очистка счетчика. Счетчик просто переполняется при прохождении своего максимального 16-битного значения (MAX = 0xFFFF), а затем перезапускается из ВОТ-ТОМ (0x0000). В нормальном режиме работы флаг переполнения таймера/счетчика (TOV1) будет установлен в течение того же цикла синхронизации таймера, когда TCNT1 становится равным нулю.

Флаг TOV1 в этом случае ведет себя как 17-й разряд, за исключением того, что он только устанавливается, но не сбрасывается. Однако, в сочетании с прерыванием переполнения таймера, которое автоматически сбрасывает флаг TOV1, разрешающая способность таймера может быть повышенена программным способом. В нормальном режиме нет особых случаев, которые необходимо рассматривать, при этом новое значение счетчика может быть записано в любое время.

Модуль сбора информации на входе можно легко использовать в нормальном режиме. При этом необходимо обратить внимание на то, что максимальный интервал между внешними событиями не должен превышать разрешающую способность счетчика. Если интервал между событиями слишком велик, то прерывание переполнения счетчика или устройства предварительного деления частоты необходимо использовать для повышения разрешающей способности модуля сбора информации.

Модули сравнения по выходу могут использоваться для генерации прерываний в определенный момент времени. Использование сравнения по выходу для генерации прерываний в нормальном режиме не рекомендуется, поскольку это будет занимать слишком значительную часть времени работы ЦПУ.

Очистка таймера в режиме совпадения при сравнении (CTC)

В режиме очистки таймера при сравнении или режиме CTC (WGM13:0 = 4 или 12), регистры OCR1A или ICR1 используются для управления разрешающей способностью счетчика. В режиме CTC счетчик сбрасывается до нуля, когда значение счетчика (TCNT1) совпадает либо с OCR1A (WGM13:0 = 4), либо с ICR1 (WGM13:0 = 12). OCR1A или ICR1 определяют верхнее значение счетчика, а следовательно, его разрешающую способность. Этот режим обеспечивает лучшее управление выходной частоты совпадения при сравнении. Он также упрощает операцию подсчета внешних событий.

Временная диаграмма для режима CTC показана на рисунке 3.43. Значение счетчика (TCNT1) возрастает до тех пор, пока не наступает совпадение при сравнении либо с OCR1A или ICR1, а затем счетчик (TCNT1) сбрасывается.

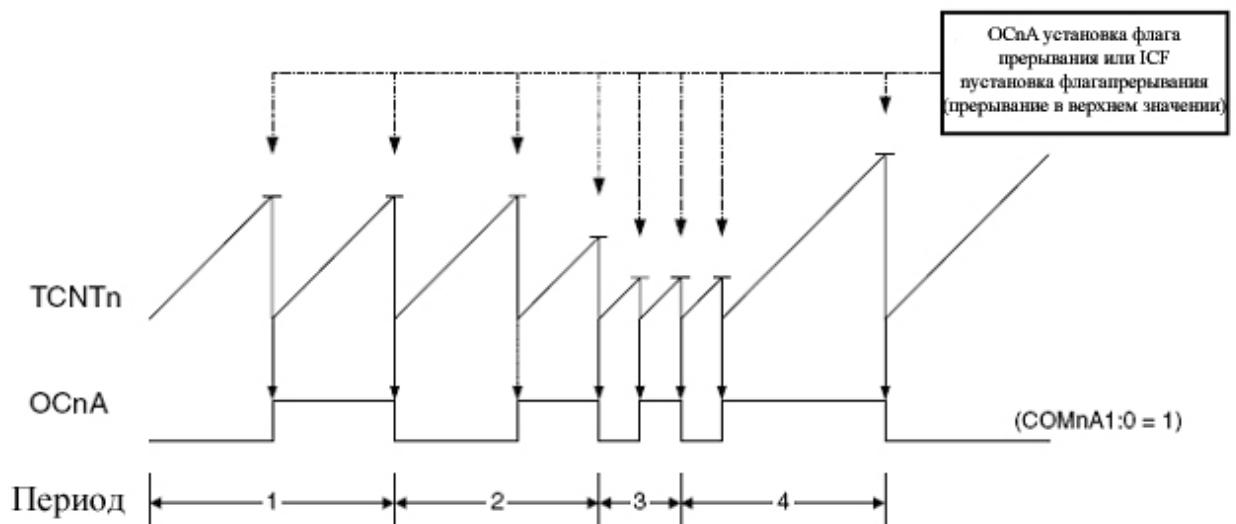


Рисунок 3.43 - Режим СТС, временная диаграмма

Прерывание может быть сгенерировано каждый раз, когда значение счетчика достигает ТОР благодаря использованию флагжа ОCF1A или ICF1 в соответствии с используемым регистром для определения значения ТОР. Если прерывание разрешено, то подпрограмма обработчика прерываний может использоваться для того, чтобы обновить значение ТОР. При этом изменение значения величины ТОР на величину, близкую к BOTTOM, когда счетчик работает с низким значением или без значения масштабирования частоты, должно выполняться очень осторожно, поскольку режим СТС не имеет возможностей для двойной буферизации. Если новое значение, записанное в OCR1A или ICR1, ниже, чем текущее значение TCNT1, то счетчик пропустит совпадение при сравнении. Счетчик должен будет тогда выполнить подсчет до его максимального значения (0xFFFF) и сделать сброс, начав со значения 0x0000 прежде, чем сможет произойти совпадение при сравнении. Во многих случаях эта особенность не желательна. В качестве альтернативы тогда необходимо будет использовать быстродействующий режим ШИМ, используя OCR1A для того, чтобы определить ТОР (WGM13:0 = 15), поскольку OCR1A тогда будет иметь двойное буферирование.

Для генерации выхода формы сигнала в режиме СТС, выход OC1A можно установить для переключения его логического уровня при каждом совпадении путем установки разрядов режима сравнения по выходу для режима переключения (COM1A1:0=1). Величина OC1A не будет видна на выводе порта до тех пор, пока направление данных для вывода не будет установлено как выход (DDR_OC1A = 1). Генерируемая форма сигнала будет иметь максимальную частоту $f_{OC1A} = f_{clk_I/O}/2$, когда OCR1A установится в ноль (0x0000). Частота формы сигнала определяется следующим уравнением:

$$f_{OCnA} = \frac{f_{clk_I/O}}{2 \cdot N \cdot (1 + OCRnA)}$$

В данном случае переменная N представляет собой коэффициент предварительно-го деления частоты (1, 8, 64, 256 или 1024).

Для нормального режима работы флаг TOV1 устанавливается в том же самом цикле синхронизации таймера, когда счетчик ведет подсчет от MAX до 0x0000.

Быстрый режим ШИМ

Быстрый режим ШИМ (WGM13:0 = 5, 6, 7, 14 или 15) предоставляет опцию генерации формы сигнала ШИМ с высокой частотой. Быстрый режим ШИМ отличается от других вариантов режимом работы с одинарным наклоном (single-slope operation). Счетчик выполняет подсчет от BOTTOM до TOP, затем вновь запускается с BOTTOM. В неинвертированном режиме сравнения по выходу (OC1x), вывод устанавливается на совпадение при сравнении между TCNT1 и OCR1x и затем сбрасывается при TOP. В режиме инвертированного сравнения по выходу вывод сбрасывается при совпадении сравнений и затем устанавливается при TOP. Благодаря режиму работы с одинарным наклоном, рабочая частота в быстродействующем режиме ШИМ может быть в два раза выше, чем для режимов ШИМ с фазовой или фазово-частотной коррекцией, которые используют режим работы с двойным наклоном. Такая высокая частота делает быстрый ШИМ режим очень приемлемым для регулировки мощности, выпрямления и применений для ЦАП. Высокая частота позволяет использовать внешние компоненты с малыми физическими размерами (катушки, конденсаторы), благодаря чему снижается общая стоимость системы.

Разрешение для быстрого режима ШИМ можно фиксировать до 8, 9 или 10 разрядов или определять разрешение с помощью ICR1 или OCR1A. Минимально допустимое разрешение составляет 2 разряда (ICR1 или OCR1A устанавливаются на 0x0003), а максимальное разрешение равно 16 разрядам (ICR1 или OCR1A установлены на MAX). Разрешение в ШИМ режиме можно вычислить по следующей формуле:

$$R_{FPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

В быстром ШИМ режиме значения счетчика возрастают до тех пор, пока значение счетчика не совпадет с одним из фиксированных значений 0x00FF, 0x01FF или 0x03FF (WGM13:0 = 5, 6 или 7), значением в ICR1 (WGM13:0 = 14) или значением в OCR1A (WGM13:0 = 15). Затем при следующем цикле синхронизации таймера счетчик сбрасывается. Временная диаграмма для быстрого режима ШИМ показана на рисунке 3.44. На рисунке показан быстрый режим ШИМ для случая, когда OCR1A или ICR1 используются для определения TOP. Величина TCNT1 на временной диаграмме показана как гистограмма для иллюстрации режима работы с одиночным наклоном. На схеме показаны как неинвертированные, так и инвертированные выходы ШИМ. Небольшие горизонтальные линейные метки на наклонных участках TCNT1 представляют совпадение между OCR1x и TCNT1. Флаг прерывания OC1x устанавливается, когда произойдет совпадение при сравнении.

Флаг переполнения таймера/счетчика (TOV1) устанавливается каждый раз, когда счетчик достигает значения TOP. В дополнение флаг OC1A или ICF1 флаг устанавливается в течение того же цикла синхронизации, когда устанавливается TOV1, и если OCR1A или ICR1 используется для определения значения TOP. Если разрешается одно из прерываний, то процедуру отладчика прерывания можно использовать для обновления TOP и величин сравнения.

При изменении величины TOP программа должна обеспечить, чтобы новое значение TOP было выше или равно значению всех регистров сравнения. Если значение TOP ниже, чем у любого из регистров сравнения, то совпадение при сравнении никогда не происходит между TCNT1 и OCR1x. Необходимо обратить внимание на то, что при использовании фиксированных значений TOP, неиспользуемые разряды маскируются нулем при записи любых значений регистров OCR1x.

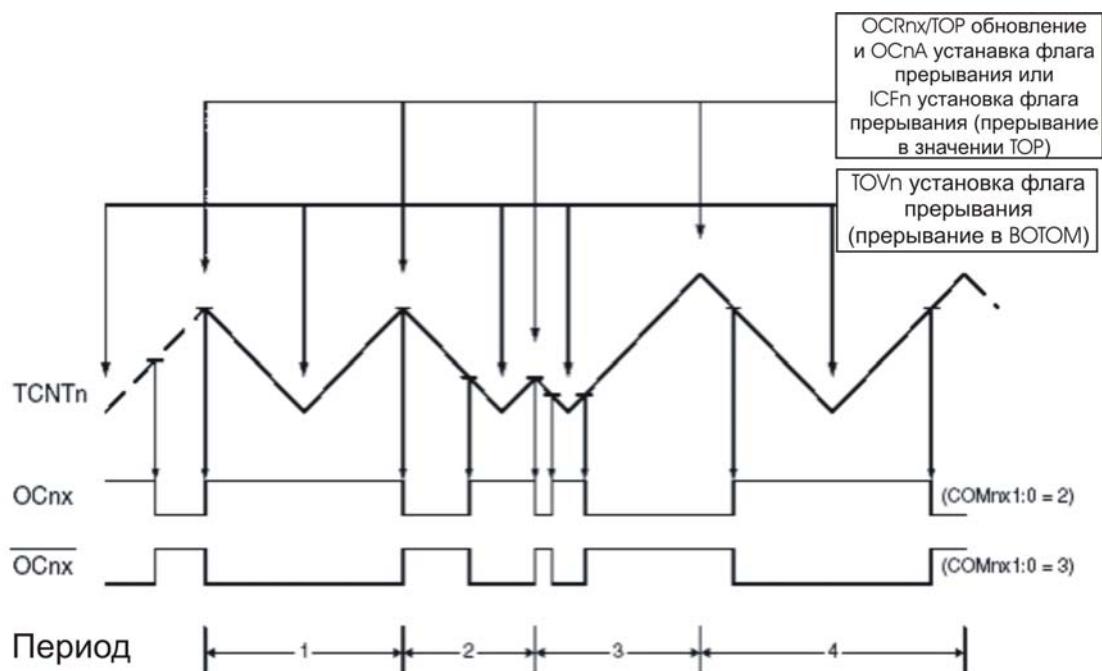


Рисунок 3.44 - Временная диаграмма, быстрый режим ШИМ

Процедура обновления ICR1 отличается от обновления OCR1A, когда она используется для определения величины TOP. Регистр ICR1 не имеет двойного буферирования. Это означает, что если ICR1 изменяется на низкое значение, когда счетчик работает при низком или отсутствующем значении устройства предварительного деления частоты, то существует риск, что новое записанное значение ICR1 будет ниже, чем текущее значение TCNT1. В результате счетчик пропустит совпадение сравнения при значении TOP. Затем счетчику придется считать до значения MAX (0xFFFF) и сделать циклический переход, начиная с 0x0000 до того, как может произойти совпадение при сравнении. При этом регистр OCR1A имеет двойное буферирование. Эта особенность позволяет записывать адрес входа/выхода OCR1A в любое время. При записи адреса входа/выхода OCR1A записываемое значение будет помещено в буферный регистр OCR1A. Затем регистр сравнения OCR1A будет обновлен на значение в буферном регистре, при следующем цикле синхронизации таймера TCNT1 совпадет с TOP. Обновление выполняется в том же цикле синхронизации таймера, поскольку TCNT1 сбрасывается и устанавливается флаг TOV1.

Использование регистра ICR1 для определения TOP хорошо срабатывает при использовании фиксированных значений TOP. Благодаря использованию ICR1, регистр OCR1A свободен для использования при генерации выхода ШИМ на OC1A. В то же время при активном изменении частоты ШИМ (за счет изменения величины TOP), использование OCR1A в качестве TOP является явно лучшим выбором благодаря наличию двойного буферирования.

В быстром режиме ШИМ модули сравнения позволяют обеспечить генерацию формы сигнала ШИМ на выводах OC1x. Установка разрядов COM1x1:0 в значение два создаст неинвертированный режим ШИМ и инвертированный выход ШИМ может генерироваться путем установки COM1x1:0 на три. Фактическое значение OC1x будет видимым на выводе порта, если направление данных для вывода порта установлено как выход (DDR_OC1x). Форма сигнала ШИМ генерируется путем установки (или очистки) регистра OC1x при совпадении OCR1x и TCNT1, и очистки (или установки) регистра OC1x каждый раз, когда счетчик обнуляется (изменения с TOP на BOTTOM).

Частоту ШИМ для выхода можно вычислить с помощью следующего уравнения:

$$f_{OCnxPWM} = \frac{f_{clk_I/O}}{N \cdot (1 + TOP)}$$

Переменная N представляет собой значение предделителя (1, 8, 64, 256 или 1024). Крайние значения OCR1x регистра представляют специальные случаи, когда генерируется ШИМ сигнал в быстром режиме. Если OCR1x установлен равным BOTTOM (0x0000), выходной сигнал будет представлять узкий импульс для каждого TOP+1 тактового импульса таймера. Установка OCR1x равным TOP приведет к постоянному высокому или низкому логическому уровню на выходе (в зависимости от полярности выхода, установленного битами COM1x1:0). Частота (с 50 % скважностью) выходного сигнала в быстром ШИМ режиме может быть достигнута установкой бита OC1A, чтобы переключить его логический уровень при каждом совпадении (COM1A1:0 = 1). Генерированный сигнал будет иметь максимальную частоту $f_{OC1A} = f_{clk_I/O}/2$, когда OCR1A будет равно нулю (0x0000). Эта особенность напоминает ситуацию, когда OC1A переключается в СТС режим, кроме двойной буферизации выхода модуля сравнения, разрешенного в быстром ШИМ режиме.

Режим фазовой коррекции

Режим фазовой коррекции ШИМ (WGM13:0 = 1, 2, 3, 10 или 11) обеспечивает высокое разрешение фазовой коррекции ШИМ сигнала. Фазовая коррекция ШИМ, подобно фазовой и частотной коррекции ШИМ режима, основывается на операции двойного наклона. В неинвертирующем режиме сравнения, выход результата сравнения (OC1x) очищается при обнаружении совпадения между TCNT1 и OCR1x при инкременте и устанавливается, когда обнаруживается совпадение при декременте. В инвертирующем режиме сравнения производится операция инверсии. Операция двойного наклона имеет максимальную частоту ниже, чем у операции одинарного наклона. Однако, благодаря симметричным особенностям режима ШИМ с двойным наклоном, эти режимы предпочтительней для приложений управления электродвигателями. Разрешение ШИМ для режима фазовой коррекции может быть зафиксировано 8, 9 или 10 битным, или определено либо ICR1, либо OCR1A. Минимальное разрешение – 2 бита (ICR1 или OCR1A set to 0x0003), максимальное разрешение – 16 бит (ICR1 или OCR1A устанавливаются MAX). Разрешение ШИМ может быть вычислено, используя следующее выражение:

$$R_{PCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

В режиме фазовой коррекции ШИМ счетчик инкрементируется до тех пор, пока его значение не станет равно 0x00FF, 0x01FF или 0x03FF (WGM13:0 = 1, 2 или 3), значению в ICR1 (WGM13:0 = 10) или значению в OCR1A (WGM13:0 = 11). Счетчик достигает значения TOP и изменяет направление счета. Значение TCNT1 будет равно значению TOP в течение одного периода тактового сигнала таймера. Временная диаграмма режима фазовой коррекции представлена на рисунке 3.45. Рисунок представлен в виде гистограммы для иллюстрации операции двойного наклона. Диаграмма содержит неинвертированный и инвертированный выходы ШИМ. Небольшая горизонтальная линия отмечает на TCNT1 наклоны, которые представляют сравнение совпадений между OCR1x и TCNT1. Флаг прерывания OC1x будет установлен, когда произойдет совпадение.

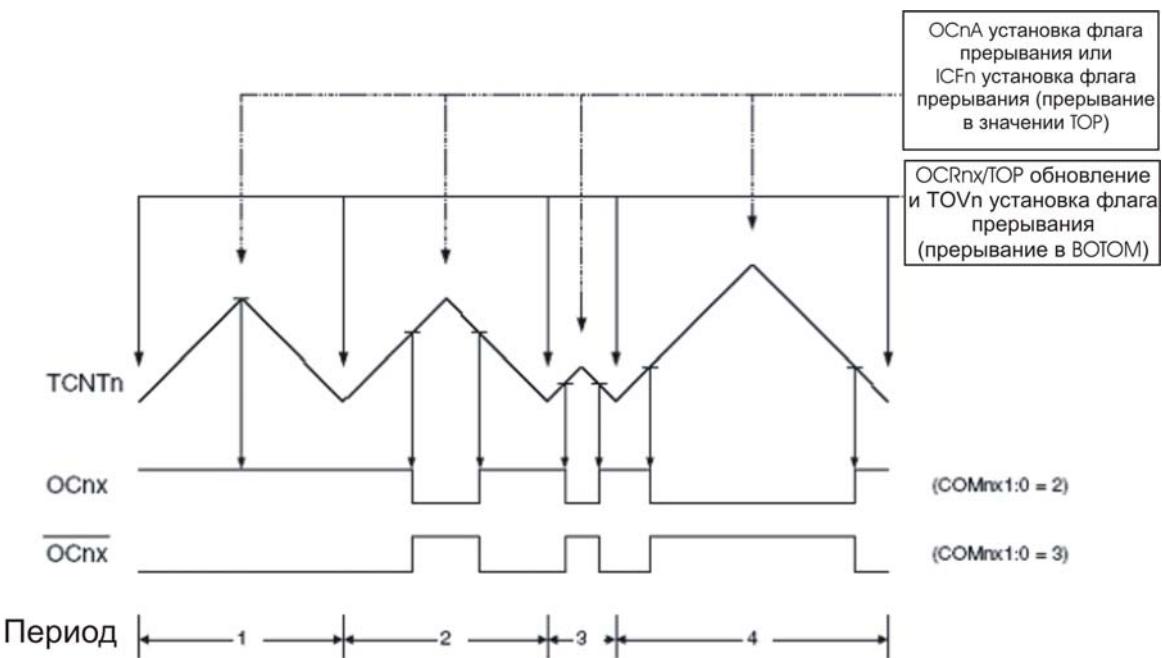


Рисунок 3.45 – Временная диаграмма режима фазовой коррекции ШИМ

Флаг переполнения таймера/счетчика (TOV1) устанавливается во время того же цикла синхронизации таймера по мере обновления регистров OCR1x на двойное буферированное значение (при значении BOTTOM). Если для определения значения TOP используется OCR1A или ICR1, то флаг OC1A или ICF1 устанавливается, когда TCNT1 достигает значения TOP. Флажки прерывания затем могут использоваться для генерации прерывания каждый раз, когда счетчик получает значение TOP или BOTTOM.

При смене значения TOP программа должна обеспечить, чтобы новое значение TOP было выше или равно значению всех регистров сравнения. Если значение TOP ниже, чем любое значение регистров сравнения, то совпадение при сравнении никогда не произойдет между значениями TCNT1 и OCR1x. Как показано на рисунке 3.46, генерируемый выходной сигнал, в отличие от режима фазовой коррекции, является симметричным в течение всех временных периодов. Поскольку обновление регистров OCR1x происходит при BOTTOM, то величина фронта нарастания и спада будет всегда одинаковой. В результате на выходе получаются симметричные импульсы и, соответственно, правильная частота. Использование регистра ICR1 для определения TOP хорошо срабатывает при использовании фиксированных значений TOP. При использовании ICR1, регистр OCR1A свободен и может использоваться для генерации выхода ШИМ на OC1A. В то же время, если базовая частота ШИМ активно изменяется при изменении значения TOP, использование OCR1A в качестве TOP совершенно очевидно, является лучшим выбором благодаря наличию функции двойного буферирования.

В режиме ШИМ с фазовой коррекцией модули сравнения позволяют обеспечить генерацию форм сигнала ШИМ на выводах OC1x. Установка разрядов COM1x1:0 в значение два создаст не инвертированный режим ШИМ и будет сгенерирован инвертированный режим ШИМ с помощью установки COM1x1:0 в значение три (таблица 3.47). Фактическое значение OC1x будет видно на выводе порта, если направление данных для вывода порта устанавливается как выход (DDR_OC1x). Форма сигнала ШИМ сигнала генерируется путем установки (или сброса) регистра OC1x при совпадении между OCR1x и TCNT1, когда значение счетчика возрастает и сбрасывается (или установки) регистра OC1x при совпадении между OCR1x и TCNT1, когда значение счетчика уменьшается. Частоту ШИМ для выхода при использовании ШИМ режима с фазовой и частотной коррекцией можно вычислить с помощью следующего уравнения:

$$f_{OCnxPFCPWM} = \frac{f_{clk_IO}}{2 \cdot N \cdot TOP}$$

Переменная “N” представляет предварительный делитель частоты (1, 8, 64, 256 или 1024). Крайние значения для регистра OCR1x представляют собой особые случаи, когда генерируется выходной сигнал ШИМ в режиме с фазовой коррекцией. Если OCR1x устанавливается равной величине BOTTOM, то выход постоянно будет находиться в низком состоянии, а если оно устанавливается на TOP, то выход будет установлен в высокое состояние для не инвертированного режима ШИМ. Для инвертированного режима ШИМ выход будет иметь противоположные логические значения.

Режимы коррекции фазы и частоты ШИМ

Фазовая и частотная коррекция ШИМ (WGM13:0 = 8 или 9) обеспечивает опцию генерации сигнала ШИМ с фазовой и частотной коррекцией, обладающей высокой разрешающей способностью. Режим фазовой и частотной коррекции подобен режиму фазовой коррекции ШИМ, который базируется на операции двойного наклона. Счетчик считает сначала от значения BOTTOM до значения TOP, после чего меняет направление счета и движется от значения TOP к значению BOTTOM. В неинвертирующем режиме сравнения выход компаратора сбрасывается при обнаружении совпадений между TCNT1 и OCR1x при декременте и устанавливается при обнаружении совпадения при декрементировании. В инвертирующем режиме сравнения результат будет представлен в инвертированном виде. Операция двойного наклона обеспечивает более низкую максимальную частоту операции по сравнению с операцией одного наклона. Однако, благодаря симметричным особенностям режима ШИМ с двойным наклоном, эти режимы более предпочтительны для приложений управления электродвигателями.

Основное отличие между режимами фазовой коррекции и фазочастотной коррекции состоит в том, что время OCR1x регистра обновляется OCR1x буферным регистром.

Разрешение ШИМ для фазочастотного режима может быть определено либо в ICR1 либо в OCR1A. Минимальное разрешение составляет 2 бита (ICR1 или OCR1A установлено в 0x0003), максимальное разрешение составляет 16 бит (ICR1 или OCR1A установлено в MAX). Разрешение ШИМ может быть вычислено, используя следующее выражение:

$$R_{PFCPWM} = \frac{\log(TOP + 1)}{\log(2)}$$

В фазочастотном режиме коррекции ШИМ счетчик инкрементируется до тех пор, пока его значение не станет равным значению в ICR1 (WGM13:0 = 8) или значению в OCR1A (WGM13:0 = 9). Счетчик достигает значения TOP и изменяет направление счета. Значение TCNT1 будет равным TOP в течение одного периода тактового сигнала таймера. Временная диаграмма фазовой коррекции и частотной коррекции представлена на рисунке 3.46.

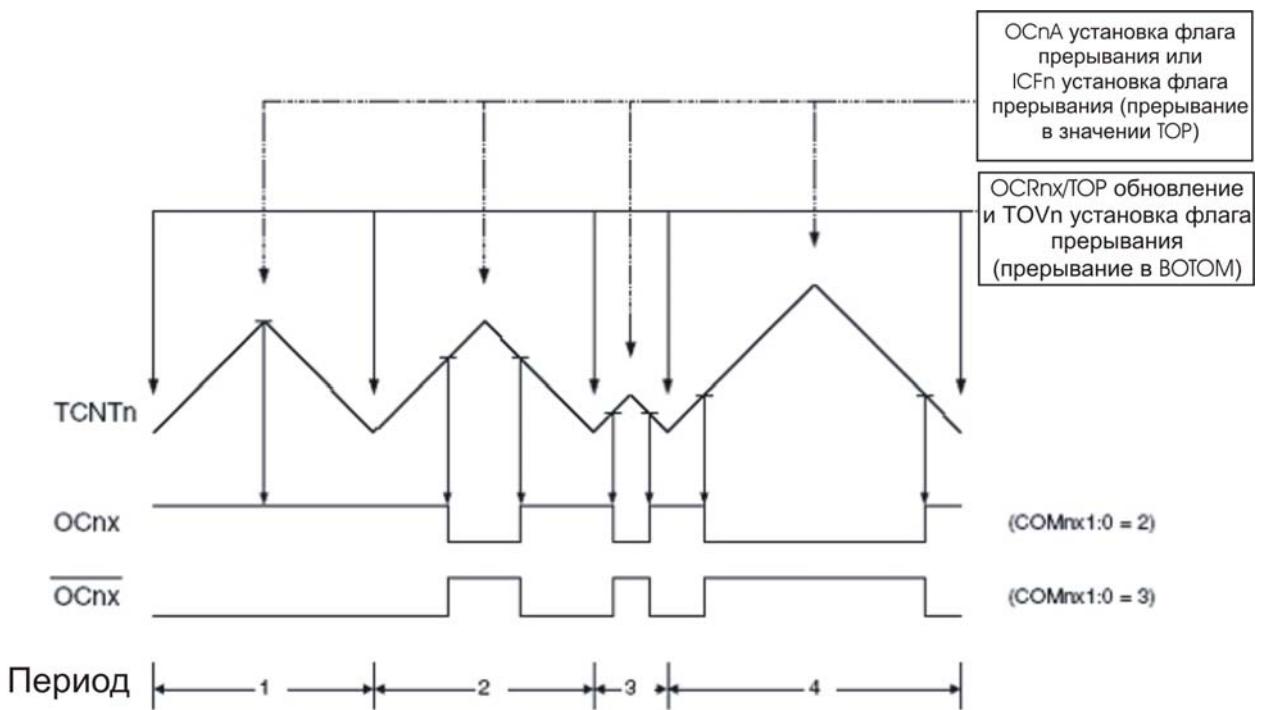


Рисунок 3.46 – Временная диаграмма режима фазовой и частотной коррекции

На рисунке показан режим фазовой и частотной коррекции ШИМ, когда OCR1A или ICR1 используются для определения TOP. Значение TCNT1 на временной диаграмме показано в виде гистограммы для иллюстрации операции двойного наклона. Небольшая горизонтальная линия отмечает на TCNT1 наклоны, которые представляют операцию сравнения между OCR1x и TCNT1. Флаг прерывания OC1x будет установлен, когда произойдет совпадение.

Флаг переполнения Т/C (TOV1) устанавливается в том же цикле тактового сигнала таймера, когда OCR1x регистры обновляются значением двойного буфера (BOTTOM). Когда либо OCR1A, либо ICR1 используются для определения значения TOP, OC1A или ICF1, флаг устанавливается, когда TCNT1 достигнет TOP. Флаги прерывания могут быть использованы для формирования прерываний каждый раз, когда счетчик достигнет значения TOP или BOTTOM. Когда изменяется значение TOP, программа должна гарантировать, что новое значение TOP будет выше или равным значениям всех регистров сравнения. Если значение TOP ниже, чем значение любого регистра сравнения, совпадение никогда не произойдет между TCNT1 и OCR1x. На рисунке 3.46 показан сгенерированный выходной сигнал, который в контрасте с режимом фазовой коррекции симметричен весь период. С момента, когда содержимое OCR1x регистра обновляется значением BOTTOM, длина нарастающего и спадающего наклонов всегда будет одинакова. Это приводит к симметричности выходных импульсов с корректной частотой. Использование ICR1 регистра для определения TOP обеспечит наилучшую работу при использовании фиксированных значений TOP. Использованием регистра ICR1 обеспечивается освобождение регистра OCR1A для генерации ШИМ выходного сигнала на OC1A. Однако, если базовая частота ШИМ активно изменяется изменением значения TOP, использование OCR1A как TOP является несомненно лучшим выбором благодаря особенности двойной буферизации. В фазочастотном режиме коррекции ШИМ модули сравнения позволяют генерировать ШИМ сигнал на OC1x выводах. Установка COM1x1:0 битов в значение два будет давать на выходе неинвертированный ШИМ сигнал, установка же COM1x1:0 битов в значение три приведет к генерации инверсного ШИМ сигнала. Действительное значение OC1x будет доступно только на выводах порта, если порт будет сконфигурирован на вывод данных. ШИМ сигнал генерируется установкой (или очист-

кой) OC1x регистра при обнаружении совпадения между регистрами OCR1x и TCNT1 при инкрементировании счетчика, и очисткой (или установкой) OC1x регистра при обнаружении совпадения между регистрами OCR1x и TCNT1 при декрементировании счетчика. Частота сигнала ШИМ при использовании режима фазочастотной коррекции может быть вычислена, используя следующее выражение:

$$f_{OCnxPFCPWM} = \frac{f_{clk_I/O}}{2 \cdot N \cdot TOP}$$

Переменная “N” представляет предварительный делитель частоты (1, 8, 64, 256 или 1024). Крайние значения для регистра OCR1x представляют собой особые случаи, когда генерируется выходной сигнал ШИМ в режиме с фазовой коррекцией. Если OCR1x устанавливается равной величине BOTTOM, то выход постоянно будет находиться в низком состоянии, а если оно устанавливается на TOP, то выход будет установлен в высокое состояние для не инвертированного режима ШИМ. Для инвертированного режима ШИМ выход будет иметь противоположные логические значения.

Временные диаграммы таймера/счетчика

Таймер/счетчик построен по синхронному принципу и сигнал синхронизации таймера (clk_{T1}) с учетом этого показан на следующих иллюстрациях как сигнал разрешения синхронизации. На рисунках показано когда устанавливаются флагги прерывания, а также, когда регистр OCR1x обновляется с помощью значения буфера OCR1x (только для режимов, использующих двойное буферирование). На рисунке 3.47 показана временная диаграмма для установки OCF1x.

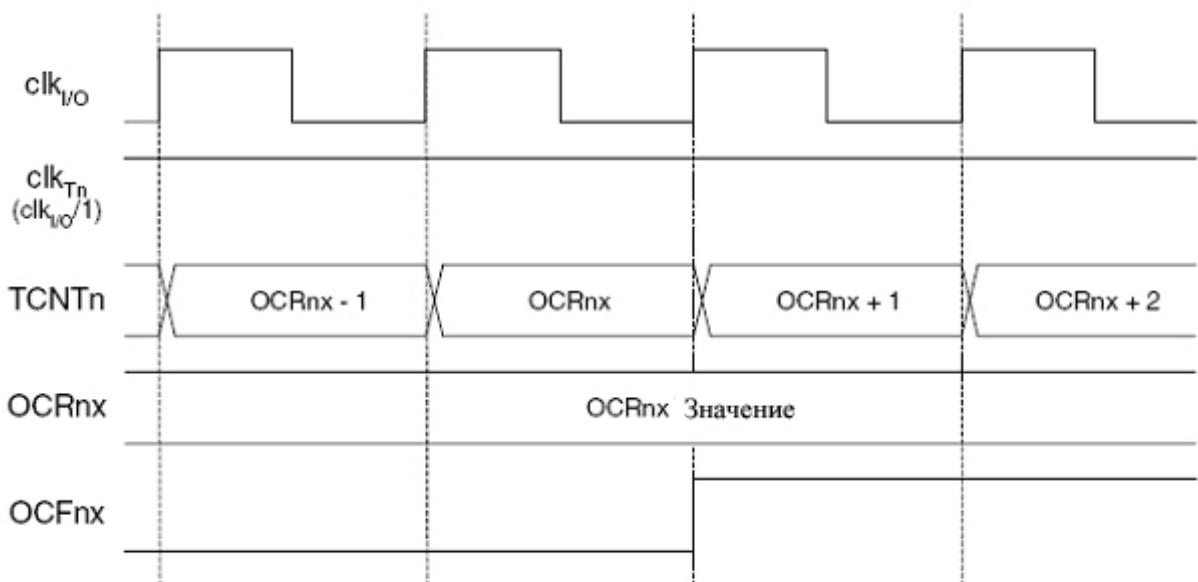


Рисунок 3.47 - Временная диаграмма таймера/счетчика, установка OCF1x без предварительного деления частоты

На рисунке 3.48 показана та же временная диаграмма, но при этом предварительный делитель частоты включен.

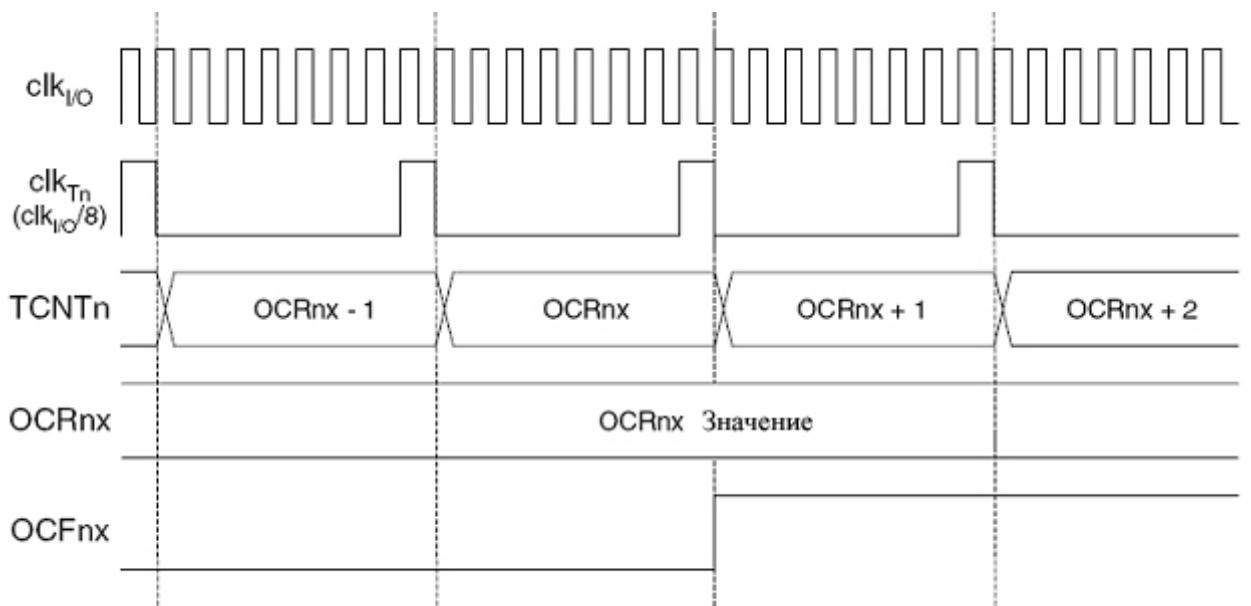


Рисунок 3.48 - Временная диаграмма таймера/счетчика, установка OCF1x с предварительным делением частоты

На рисунке 3.49 показана последовательность подсчета, близкая к TOP в различных режимах. При использовании ШИМ режима с фазовой и частотной коррекцией, регистр OCR1x обновляется при BOTTOM. Временные диаграммы будут такими же, но TOP следует заменить на BOTTOM, TOP-1 на BOTTOM+1 и т. д. Аналогичное переименование применимо для режимов, которые устанавливают флаг TOV1 при BOTTOM.

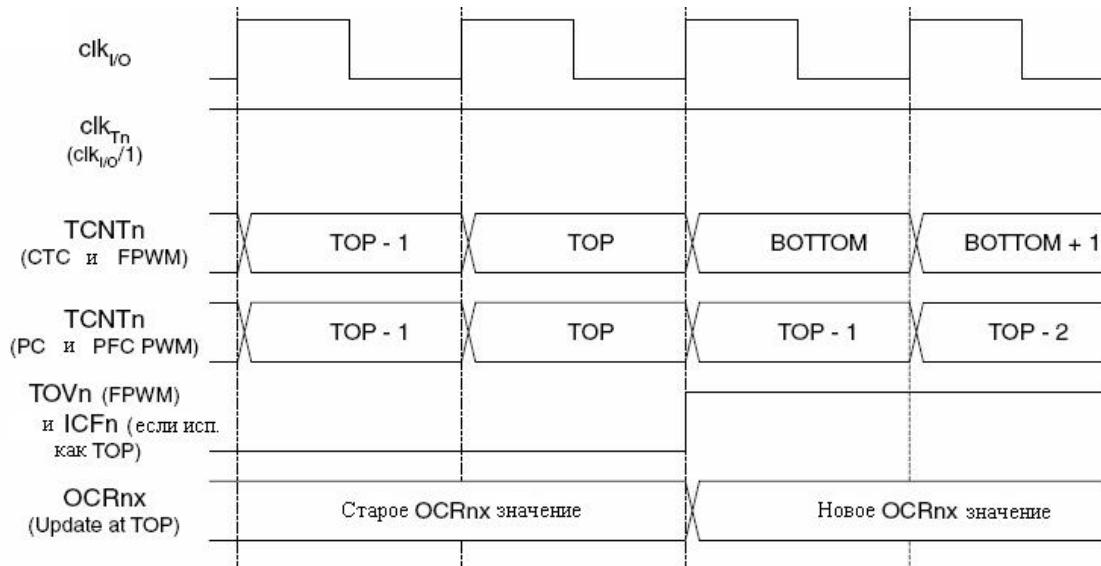


Рисунок 3.49 - Временная диаграмма таймера/счетчика без устройства предварительного деления частоты

На рисунке 3.50 показаны те же временные параметры, но при включенном устройстве предварительного деления частоты.

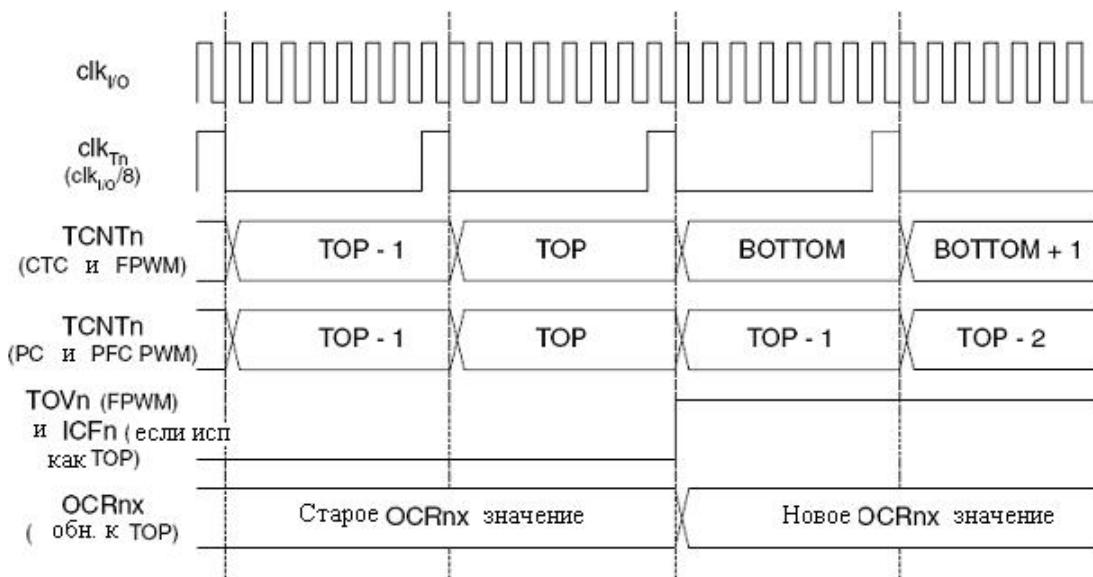


Рисунок 3.50 - Временная диаграмма таймера/счетчика с устройством предварительного деления частоты ($f_{\text{clk_I/O}}/8$)

Описание 16-разрядного регистра таймера/счетчика

Регистр А управления таймера/счетчика 1 – TCCR1A

Бит	7	6	5	4	3	2	1	0	TCCR1A
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	
Чтение/Запись	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
нач. значение	0	0	0	0	0	0	0	0	

Разряды 7, 6 : COM1A1:0 - режим сравнения по выходу для канала А

Разряды 5, 4: COM1B1:0 - режим сравнения по выходу для канала В

COM1A1:0 и COM1B1:0 управляют поведением выводов сравнения по выходу (соответственно, OC1A и OC1B). Если один или оба бита COM1A1:0 записываются в единицу, то выход OC1A отменяет нормальное функциональное назначение порта для вывода входа/выхода, к которому он подсоединен. При этом следует обратить внимание на то, что разряд регистра направления данных (DDR), соответствующий выводу OC1A или OC1B, должен быть установлен таким образом, чтобы запустить драйвер выхода. Если OC1A или OC1B подсоединенены к выводу, то функция разрядов COM1x1:0 зависит от установки разрядов WGM13:0. Таблица 3.42 показывает функции разряда COM1x1:0, когда разряды WGM13:0 установлены на нормальный режим или режим CTC (не режим ШИМ).

Таблица 3.42 - Режим сравнения по выходу, не режим ШИМ

COM1A1/ COM1B1	COM1A0/ COM1B0	Описание
0	0	Нормальная работа порта, OC1A/OC1B отсоединено.
0	1	Переключение OC1A/OC1B при совпадении сравнений.
1	0	Сброс OC1A/OC1B при совпадении сравнений (устанавливает выход на низкий уровень).
1	1	Сброс OC1A/OC1B при совпадении сравнений (устанавливает выход на низкий уровень).

Таблица 3.43 показывает функциональность разряда COM1x1:0 при установке разрядов WGM13:0 на быстрый режим ШИМ.

Таблица 3.43 - Режим сравнения по выходу, быстрый режим ШИМ¹⁾

COM1A1/ COM1B1	COM1A0/ COM1B0	Описание
0	0	Нормальная работа порта, OC1A/OC1B отсоединенны.
0	1	WGM13=0: Нормальный режим работы порта, OC1A/OC1B отсоединенны. WGM13=1: Переключение OC1A на совпадение сравнений, OC1B зарезервировано.
1	0	Очистка OC1A/OC1B при совпадении сравнений, установка OC1A/OC1B на TOP.
1	1	Установка OC1A/OC1B при совпадении сравнений, очистка OC1A/OC1B на TOP.

¹⁾ Обычный случай имеет место, когда OCR1A/OCR1B равняется TOP и установлено COM1A1/COM1B1. В этом случае совпадение сравнений игнорируется, однако установка или сброс выполняются на TOP.

В таблице 3.44 показаны функции разряда COM1x1:0, когда разряды WGM13:0 установлены на режим ШИМ с фазовой коррекцией или же с фазовой и частотной коррекцией.

Таблица 3.44 – Режим сравнения по выходу, фазовая и фазочастотная коррекция ШИМ.

COM1A1/ COM1B1	COM1A0/ COM1B0	Описание
0	0	Общая функция порта, OC1A/OC1B отключены
0	1	WGM13=0: Общая функция порта, OC1A/OC1B отключены WGM13=1: Включение OC1A в режим сравнения, OC1B зарезервирован
1	0	Сброс OC1A/OC1B при совпадении (инкремент счетчика) Установка OC1A/OC1B при совпадении (декремент счетчика)
1	1	Установка OC1A/OC1B при совпадении (инкремент счетчика) Сброс OC1A/OC1B при совпадении (декремент счетчика)

Примечание - Особый случай имеет место, когда OCR1A/OCR1B равняется TOP и установлено COM1A1/COM1B1.

Разряд 3: FOC1A - Принудительное сравнение по выходу для канала А

Разряд 2: FOC1B - Принудительное сравнение по выходу для канала В

Биты FOC1A/FOC1B активны только тогда, когда биты WGM13:0 определяют режим без ШИМ. В то же время для обеспечения совместимости с перспективными устройствами, эти биты должны быть установлены в ноль, если TCCR1A записывается при работе в режиме ШИМ. При записи логической единицы в разряд FOC1A/FOC1B, в блок генерации формы сигнала принудительно вводится совпадение сравнений. Выход OC1A/OC1B изменяется согласно своим установкам разряда COM1x1:0. Следует обратить также внимание на то, что биты FOC1A/FOC1B реализуются как сигналы стробирования. С учетом этого именно значение, присутствующее в разрядах COM1x1:0, определяет влияние принудительного сравнения. Стробирующий сигнал FOC1A/FOC1B не будет создавать какого-либо прерывания и не будет очищать таймер в режиме сброса тай-

мера при совпадении сравнений при использовании OCR1A как ТОР. Биты FOC1A/FOC1B всегда считаются как ноль.

Разряды 1, 0: WGM11:0 - Режим генерации формы сигнала

Вместе с битами WGM13:2, обнаруженными в регистре TCCR1B, данные разряды могут управлять последовательностью счета счетчика. В отношении источника для максимального значения счетчика (TOP), а также относительно того, какая форма сигнала должна использоваться, см. таблицу 3.46. К режимам работы, поддерживаемым блоком таймера/счетчика, относятся: нормальный режим (счетчик), сброс таймера при совпадении (CTC), а также три типа режимов ШИМ.

Таблица 3.45 - Режим ШИМ (1) сравнения по выходу, фазовой коррекции и фазовой и частотной коррекции

COM1A1/ COM1B1	COM1A0/ COM1B0	Описание
0	0	Нормальная работа порта, OC1A/OC1B отключены.
0	1	WGM13=0: Нормальная работа порта, OC1A/OC1B отключены. WGM13=1: Переключение OC1A на совпадение сравнений, OC1B зарезервирован.
1	0	Сброс OC1A/OC1B при совпадении сравнений, установка OC1A/OC1B на TOP.
1	1	Установка OC1A/OC1B на совпадение сравнений, сброс OC1A/OC1B на TOP.
Примечание - Особая ситуация создается, когда OCR1A/OCR1B равняется TOP и устанавливается COM1A1/COM1B1.		

Разряд 3: FOC1A - Принудительное сравнение по выходу для канала А

Разряд 2: FOC1B - Принудительное сравнение по выходу для канала В

Биты FOC1A/FOC1B являются активными только тогда, когда биты WGM13:0 определяют режим без использования ШИМ. Однако для обеспечения совместимости с будущими приборами, эти биты должны быть установлены на ноль при записи TCCR1A во время работы в режиме ШИМ. При записи логической единицы в бит FOC1A/FOC1B на вход генератора формы сигнала принудительно подается немедленный результат совпадения при сравнении. Выход OC1A/OC1B меняется согласно установке битов COM1x1:0.

Обратите внимание на то, что биты FOC1A/FOC1B реализуются в виде сигналов стробирования. С учетом этого значение, присутствующее в битах COM1x1:0 определяет эффект принудительного сравнения. Сигнал стробирования FOC1A/FOC1B не будет генерировать никакого сигнала прерывания, а также не будет очищать таймер в режиме сброса таймера в режиме совпадения сравнений (CTC) при использовании OCR1A как ТОР. Биты FOC1A/FOC1B всегда считаются как ноль.

Разряды 1, 0: WGM11:0 - Режим генерации формы сигнала

Совместно с битами WGM13:2, обнаруженными в регистре TCCR1B, эти разряды управляют последовательностью подсчета для счетчика. В отношении источника для максимального значения счетчика (TOP), а также информации о том, какой тип генерации формы сигнала необходимо использовать, см. таблицу 3.46. Режимы работы, поддерживаемые блоком таймера/счетчика, следующие: нормальный режим (счетчик), режим сброса таймера при совпадении сравнений (CTC) и три режима ШИМ.

Таблица 3.46 - Описание битов для режимов генератора формы сигнала (1)

Режим	WGM3	WGM12 (CTC1)	WGM11 (RWM11)	WGM10 (PWM10)	Режим операции T/C	TOP	Обновление OCR1x в:	TOV1 флаг ус- тановлен в:
0	0	0	0	0	Нормальный	0xFFFF	Immediate	MAX
1	0	0	0	1	ШИМ, фазовая коррекция, 8 бит	0x00FF	TOP	BOTTOM
2	0	0	1	0	ШИМ, фазовая коррекция, 9 бит	0x01FF	TOP	BOTTOM
3	0	0	1	1	ШИМ, фазовая коррекция, 10 бит	0x03FF	TOP	BOTTOM
4	0	1	0	0	СТС	OCR1A	Immediate	MAX
5	0	1	0	1	Быстрый ШИМ, 8 бит	0x00FF	TOP	TOP
6	0	1	1	0	Быстрый ШИМ, 9 бит	0x01FF	TOP	TOP
7	0	1	1	1	Быстрый ШИМ, 10 бит	0x03FF	TOP	TOP
8	1	0	0	0	ШИМ, фазовая и частотная корр.	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	ШИМ, фазовая и частотная корр.	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	ШИМ, фазовая коррекция	ICR1	TOP	BOTTOM
11	1	0	1	1	ШИМ, фазовая коррекция	OCR1A	TOP	BOTTOM
12	1	1	0	0	СТС	ICR1	Immediate	MAX
13	1	1	0	1	Резервирован	-	-	-
14	1	1	1	0	Быстрый ШИМ	ICR1	TOP	TOP
15	1	1	1	1	Быстрый ШИМ	OCR1A	TOP	TOP
Примечание - Наименования битов CTC1 и PWM11:0 являются устаревшими. Необходимо использовать определения WGM12:0, в то же время функциональное назначение и расположение этих битов совместимо с предыдущими версиями таймера.								

Управление таймером/счетчиком 1

Регистр В – TCCR1B

Бит	7	6	5	4	3	2	1	0	TCCR1B
Чт./Зап.	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	
	W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Бит 7: ICNC1 - Подавитель шума при захвате сигналов на входе

При установке этого разряда (на единицу) включается устройство для подавления шума на входе, при этом фильтруется шум на выводе захвата входа (ICP1). Выполнение функций фильтра требует четыре последовательных выборки с одинаковым значением на выводе ICP1 для изменения значения на его выходе. С учетом этого захват на входе задерживается на четыре тактовых цикла, если включается устройство для подавления шума.

Бит 6: ICES1 - Выбор переднего фронта кадра

Этот бит выбирает, какой из фронтов на выводе (ICP1) используется для того, чтобы запустить захват события. Когда бит ICES1 записан в ноль, задний (отрицательный) фронт используется как запускающий, а когда разряд ICES1 записан в единицу, то передний (положительный) фронт запустит захват. Когда захват запущен согласно установке для ICES1, то значение счетчика копируется во входной регистр кадра (ICR1). Данное событие также установит входной флаг кадра (ICF1), и это может использоваться для того, чтобы вызвать прерывание входа кадра, если это прерывание разрешено.

Когда ICR1 используется как значение TOP (см. описание разрядов WGM13:0, расположенных в Регистре TCCR1A и TCCR1B), то ICP1 отсоединяется и затем, отключается функция захвата кадра на входе.

Бит 5 - Зарезервированный бит

Этот бит зарезервирован для будущего использования. Для того чтобы гарантировать совместимость с будущими приборами, этот бит должен быть записан в ноль, когда записывается TCCR1B.

Биты 4, 3: WGM13:2 - Режим генерации формы сигнала

См. описание регистра TCCR1A.

Биты 2...0: CS12:0 - Выбор тактового генератора

Три бита выбора тактового генератора выбирают источник тактового генератора, который будет использоваться таймером/счетчиком.

Таблица 3.47 - Описание бита выбора тактовой частоты

CS12	CS11	CS10	Описание
0	0	0	Без источника тактовой частоты (Таймер/Счетчик остановлен)
0	0	1	clk _{I/O} /1 (без предварительного масштабирования)
0	1	0	clk _{I/O} /8 (от устройства предварительного масштабирования)
0	1	1	clk _{I/O} /64 (от устройства предварительного масштабирования)
1	0	0	clk _{I/O} /256 (от устройства предварительного масштабирования)
1	0	1	clk _{I/O} /1024 (от устройства предварительного масштабирования)
1	1	0	Источник внешней синхронизации на выводе T1. Тактовый импульс на заднем фронте.
1	1	1	Источник внешней синхронизации на выводе T1. Тактовый импульс на переднем фронте.

Если режимы на внешних выводах используются для таймера/счетчика1, то переходы на выводе T1 будут синхронизировать счетчик даже в том случае, если вывод имеет конфигурацию в качестве выхода. Эта особенность позволяет обеспечить программное управление функцией подсчета.

Таймер/Счетчик 1 – TCNT1H и TCNT1L

Бит	7	6	5	4	3	2	1	0	TCNT1H	TCNT1L
	TCNT1[15:8]									
	TCNT1[7:0]									
Чт./Зап.	W	R/W	R	R/W	R/W	R/W	R/W	R/W		
Начальное значение	0	0	0	0	0	0	0	0		

Два адреса входа/выхода таймера/счетчика (TCNT1H и TCNT1L, объединенного TCNT1) обеспечивают прямой доступ как для операций считывания, так и записи для блока таймера/счетчика 16-разрядного счетчика. Чтобы обеспечить одновременное считывание и записи как байтов с низким, так и с высоким уровнем при обращении ЦПУ к этим регистрам, доступ осуществляется с использованием 8-разрядного временного регистра старшего байта (TEMP). Этот временный регистр одновременно используется всеми остальными 16-разрядными регистрами. Если в счетчик вносятся изменения (TCNT1), когда он работает, то это вызывает риск пропуска совпадения при сравнении между TCNT1 и одним из регистров OCR1x.

Запись в регистр TCNT1 блокирует (удаляет) совпадение при сравнении для следующего тактового сигнала таймера для всех блоков сравнения.

Регистр 1 А сравнения по выходу – OCR1AH и OCR1AL

Бит	7	6	5	4	3	2	1	0	OCR1AH
	OCR1A[15:8]								OCR1AL
	OCR1A [7:0]								
Чт./Зап.	W	R/W							
Начальное значение	0	0	0	0	0	0	0	0	

Регистр 1 В сравнения по выходу – OCR1BH и OCR1BL

Бит	7	6	5	4	3	2	1	0	OCR1BH
	OCR1B[15:8]								OCR1BL
	OCR1B [7:0]								
Чт./Зап.	W	R/W							
Начальное значение	0	0	0	0	0	0	0	0	

Регистры сравнения по выходу содержат 16-разрядное значение, которое постоянно сравнивается со значением счетчика (TCNT1). Сравнение может быть использовано для генерации прерывания сравнения по выходу или для генерации формы сигнала выходного сигнала на выводе OC1x.

Регистры сравнения по выходу имеют размер 16 бит. Для обеспечения одновременной записи байтов как высокого, так и низкого разряда в случае, если ЦПУ выполняет запись в эти регистры, доступ осуществляется с использованием 8-разрядного временного регистра старшего байта (TEMP). Этот временный регистр одновременно используется всеми другими 16-разрядными регистрами.

Бит	7	6	5	4	3	2	1	0	ICR1H
	ICR1[15:8]								ICR1L
	ICR1 [7:0]								
Чт./Зап.	W	R/W							
Начальное значение	0	0	0	0	0	0	0	0	

Регистр захвата входа обновляется со счетчиком значением каждый раз, когда происходит событие на выводе ICP1 (или дополнительно на выходе аналогового компаратора для T/C1). Захват входа может быть использован для определения значения ТОР.

Регистр захвата входа имеет размер 16 бит. Для уверенности, что оба старший и младший байты читаются одновременно, когда ЦПУ обращается к этим регистрам, доступ обусловлен использованием временного 8-битного регистра (TEMP). Этот регистр совместно используется всеми другими 16-битными регистрами.

Регистр маски прерываний Т/С – TIMSK¹⁾

Бит	7	6	5	4	3	2	1	0	TIMSK
Чт./Зап.	OCIE2 R/W	TOIE2 R/W	TICIE1 R/W	OCIE1A R/W	OCIE1B R/W	TOIE1 R/W	OCIE0 R/W	TOIE0 R/W	
Начальное значение	0	0	0	0	0	0	0	0	

¹⁾ Этот регистр содержит биты управления прерываниями для различных Т/С, но только биты таймера 1 описаны в этом разделе. Остальные биты описаны в соответствующих разделах таймеров.

Бит 5: TICIE1 - Разрешение прерывания по захвату данных

Когда в этот бит записывается единица и I-флаг в статусном регистре установлен, прерывание Т/C1 по захвату входа разрешено. Переход на соответствующий вектор прерывания выполняется, когда ICF1 флаг, размещенный в TIFR, устанавливается.

Бит 4: OCIE1A - Разрешение прерывания по сравнению/совпадению А Т/C1

Когда в этот бит записывается единица и I-флаг в статусном регистре установлен, прерывание Т/C1 по сравнению/совпадению А Т/C1 разрешено. Переход на соответствующий вектор прерывания выполняется, когда OCF1A флаг, размещенный в TIFR, устанавливается.

Бит 3: OCIE1B - Разрешение прерывания по сравнению/совпадению В Т/C1

Когда в этот бит записывается единица и I-флаг в статусном регистре установлен, прерывание Т/C1 по сравнению/совпадению В Т/C1 разрешено. Переход на соответствующий вектор прерывания выполняется, когда OCF1B флаг, размещенный в TIFR, устанавливается.

Бит 2 – TOIE1: Разрешение прерывания по переполнению Т/C1

Когда в этот бит записывается единица и I-флаг в статусном регистре установлен, прерывание Т/C1 по переполнению Т/C1 разрешено. Переход на соответствующий вектор прерывания выполняется, когда TOV1 флаг, размещенный в TIFR, устанавливается.

Регистр флагов прерываний Т/С – TIFR¹⁾

Бит	7	6	5	4	3	2	1	0	TIFR
Чт./Зап.	OCF2 R/W	TOV2 R/W	ICF1 R/W	OCF1A R/W	OCF1B R/W	TOV1 R/W	OCF0 R/W	TOV0 R/W	
Начальное значение	0	0	0	0	0	0	0	0	

¹⁾ Этот регистр содержит биты управления прерываниями для различных Т/С, но только биты таймера 1 описаны в этом разделе. Остальные биты описаны в соответствующих разделах таймеров.

Бит 5: ICF1 - Флаг захвата данных

Этот флаг устанавливается, когда происходит событие захвата на ICP выводе. Когда регистр захвата входа ICR1 устанавливается WGM13:0 для использования его значения как TOP, флаг ICF1 устанавливается, когда счетчик достигает значения TOP.

ICF1 автоматически очищается, когда происходит переход на вектор прерывания захвата входа, альтернативно ICF1 может быть сброшен записью логической единицы в его битовую позицию.

Бит 4: OCF1A - Флаг сравнения/совпадения А Т/C1

Флаг устанавливается в следующем цикле тактового импульса таймера после того, как произойдет совпадение регистра совпадения по выходу А (OCR1A). При этом строб вынужденного сравнения по выходу не будет устанавливать OCF1A флаг.

OCF1A флаг автоматически очищается, когда происходит переход на вектор прерывания сравнения/совпадения по выходу А, альтернативно OCF1A может быть сброшен записью логической единицы в его битовую позицию.

Бит 3: OCF1B - Флаг сравнения/совпадения В Т/C1

Флаг устанавливается в следующем цикле тактового импульса таймера, после того как произойдет совпадение регистра совпадения по выходу В (OCR1B).

Заметим, что строб вынужденного сравнения по выходу не будет устанавливать OCF1B флаг.

OCF1B-флаг автоматически очищается, когда происходит переход на вектор прерывания сравнения/совпадения по выходу В, альтернативно OCF1B может быть сброшен записью логической единицы в его битовую позицию.

Бит 2: TOV1 - Флаг переполнения Т/C1

Этот флаг устанавливается в зависимости от установки битов WGM13. В нормальном и СТС режимах, TOV1 флаг устанавливается, когда таймер переполняется.

TOV1 флаг автоматически очищается, когда происходит переход на вектор прерывания по переполнению таймера, альтернативно TOV1 может быть сброшен записью логической единицы в его битовую позицию.

3.13 8-битный Таймер/Счетчик 2 с ШИМ и асинхронными операциями

Таймер/Счетчик 2 (T/C2) является одноканальным модулем 8-разрядного Т/С общего назначения.

Отличительные особенности:

- Одноканальный счетчик.
- Сброс таймера при совпадении сравнения (автоматическая перезагрузка).
- ШИМ с фазовой коррекцией, свободной от сбоев.
- Генератор частоты.
- Счетчик внешних событий.
- 10-разрядный предварительный делитель частоты.
- Источники прерывания при совпадении, при сравнении и при переполнении (TOV2 и OCF2).

Обзор

Упрощенная блок-схема 8-разрядного таймера/счетчика показана на рисунке 3.51

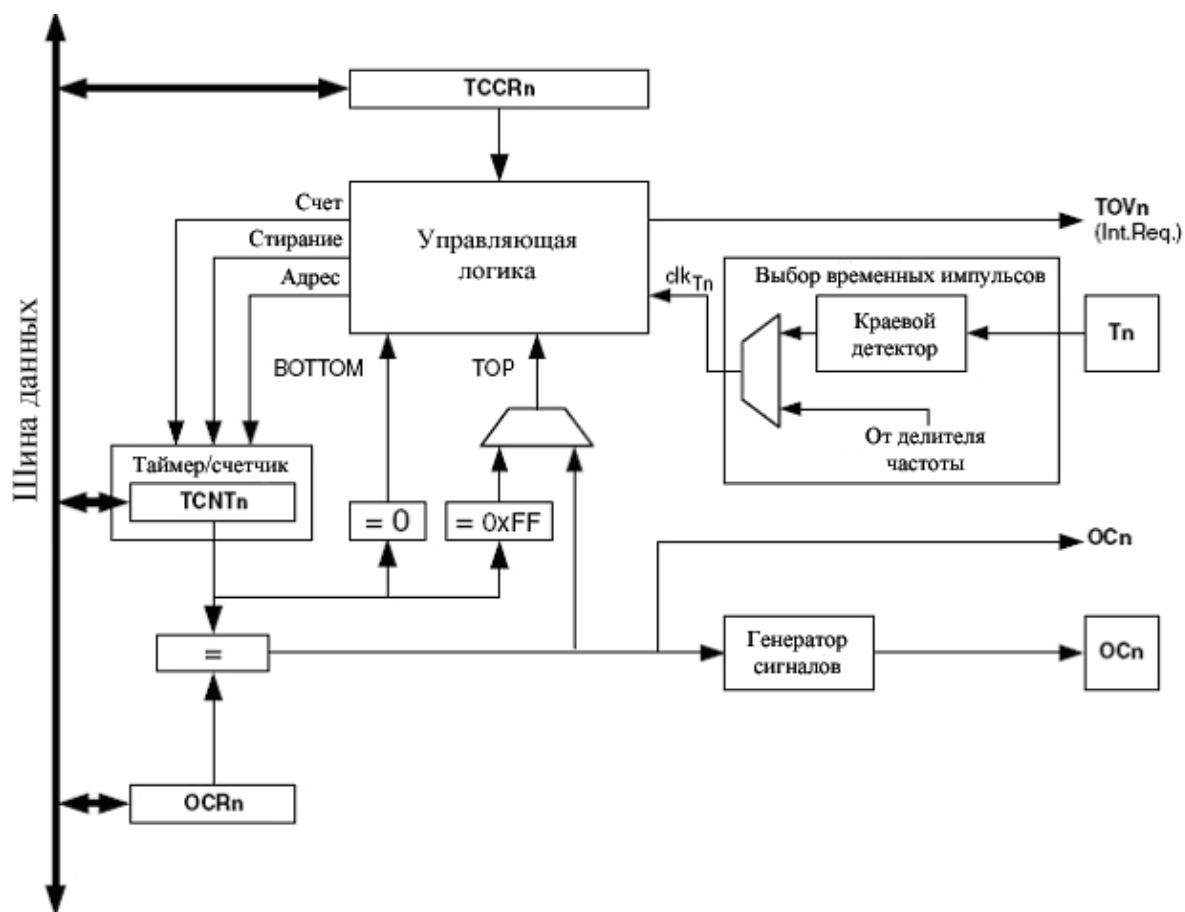


Рисунок 3.51 - Блок-схема 8-разрядного таймера/счетчика

Регистры

Таймер/счетчик (TCNT2) и регистр сравнения выхода OCR2 являются 8-разрядными регистрами. Все сигналы запросов на прерывание (на рисунке 3.51 сокращенно Int.Req.) видны в регистре флага прерывания таймера (TIFR). Все прерывания индивидуально маскируются с помощью регистра маски прерывания таймера (TIMSK). Эти регистры TIFR и TIMSK не показаны на рисунке, поскольку эти регистры совместно используются другими временными устройствами.

Таймер/Счетчик может синхронизироваться внутрисхемно через предварительный делитель частоты или с помощью источника внешнего тактового генератора на выводе T2. Логический блок управления синхронизацией выбирает, какой источник сигнала синхронизации и фронта использует Таймер/Счетчик для увеличения или уменьшения его значения. Таймер/Счетчик не активен, если не выбран источник сигнала синхронизации. Выход логики выбора тактового сигнала обозначается как генератор таймера (clk_{T2}). Регистр сравнения выхода с двойным буфером OCR2 постоянно сравнивается со значением Таймер/Счетчик. Результат сравнения может быть использован генератором формы сигнала для запуска ШИМ или выхода переменной частоты на выводе сравнения выходов (OC2). Событие совпадения будет также устанавливать флаг сравнения (OCF2), который может использоваться для запроса прерывания сравнения выхода.

Определения

Многие справочные сведения относительно регистров и разрядов записаны в этом документе в общем виде. Буква “n” в нижнем регистре обозначения заменяет номер Таймера/Счетчика, в данном случае 2. Однако использование регистра или разряда определено программой, поэтому необходимо использовать точную форму, например, TCNT2 для оценки значения счетчика Таймер/Счетчик 2 и т. д.

Определения в таблице 3.48 также интенсивно используются повсюду в данном документе.

Таблица 3.48 - Определения

BOTTOM	Счетчик достигает уровня BOTTOM, 0x00.
MAX	Счетчик достигает уровня MAX, когда он становится равным 0xFF.
TOP	Счетчик достигает уровня TOP, когда он становится равным наивысшему значению в последовательности подсчета. Значение TOP может быть приписано фиксированному значению 0xFF (MAX) или значению, сохраненному в регистре OCR2. Само присваивание зависит от режима работы.

Источники тактового сигнала таймера/счетчика

Таймер/счетчик может синхронизироваться как внутренним, так и внешним источником синхронизации. Источник синхронизации выбирается с помощью логики выбора тактового генератора, который управляется с помощью разрядов выбора сигнала синхронизации (CS22:0), расположенных в регистре управления таймера/счетчика (TCCR2).

Модуль счетчика

Основной частью 8-разрядного таймера/счетчика является программируемый двухнаправленный модуль счетчика. На рисунке 3.52 показана блок-схема счетчика и его окружения.

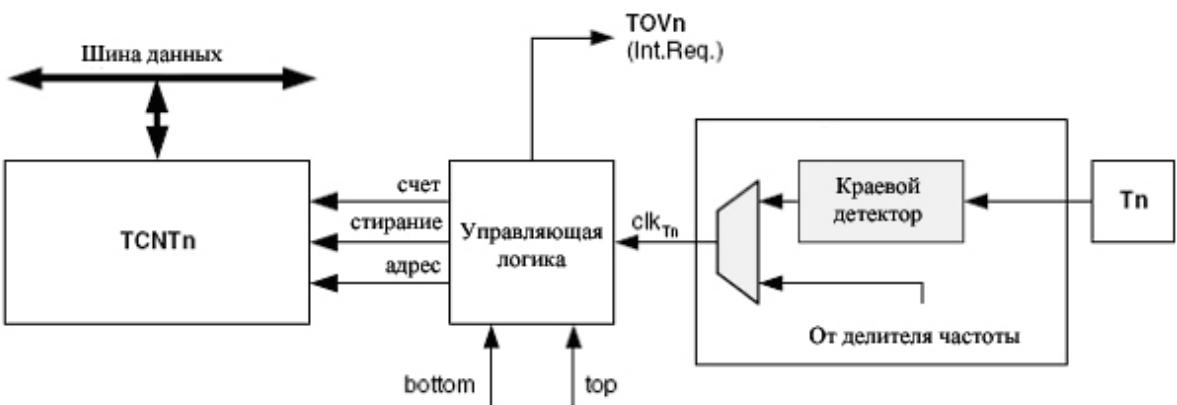


Рисунок 3.52 – Блок-схема модуля счетчика

Описание сигналов (внутренние сигналы):

count	Увеличение или уменьшение TCNT2 на 1.
direction	Выбор между увеличением или уменьшением.
clear	Очистка TCNT2 (все разряды устанавливаются в ноль).
clk_Tn	Тактовый генератор таймера/счетчика, далее именуемый как clk_T2.
top	Сигнализирует, что TCNT2 достигло максимального значения.
bottom	Сигнализирует, что TCNT2 достигло минимального значения (нуля).

В зависимости от используемого режима работы, счетчик сбрасывается, увеличивается или уменьшается при каждом тактовом сигнале таймера (clk_{T2}). clk_{T2} сможет генерироваться от внешнего или внутреннего источника тактового сигнала, выборка при этом производится с помощью разрядов Выборка тактовых сигналов (CS22:0). Если источник тактовых сигналов не выбирается, т. е. CS22:0 = 0, то таймер останавливается. В то же время, величина TCNT2 может быть выбрана с помощью ЦПУ независимо от того, присутствует clk_{T2} или нет. Операция записи в ЦПУ отменяет (обладает большим приоритетом) все операции очистки счетчика или операции счета.

Последовательность счета определяется путем установки разрядов WGM21 и WGM20, расположенных в регистре управления таймером/счетчиком TCCR2. Существует непосредственная связь между тем, как ведет себя счетчик (т. е. как он считает) и каким образом создаются формы сигнала на выходе в сравнении с выходом OC2.

Флаг переполнения таймера/счетчика TOV2 устанавливается в соответствии с режимом работы, который выбирается с помощью разрядов WGM21:0. Для генерации прерывания ЦПУ можно использовать TOV2.

Модуль сравнения выхода

8-разрядный компаратор непрерывно сравнивает TCNT2 с регистром сравнения выхода (OCR2). Как только TCNT2 становится равным OCR2, компаратор сигнализирует о совпадении. Совпадение установит флаг сравнения выхода OCF2 при следующем цикле таймера тактового генератора. При активации (OCIE0 = 1 при установке флага общего прерывания в SREG), флаг сравнения выхода создает прерывание сравнения выхода. Флаг OCF2 очищается автоматически при исполнении прерывания. В качестве альтернативы флаг OCF2 можно очищать программно путем записи логической единицы в адрес разряда входа/выхода. Генератор формы сигнала использует сигнал сравнения для генерации выходного сигнала согласно режиму работы, установленному разрядами WGM21:0 и режиму сравнения выхода, установленному разрядами (COM21:0). Максимальное и нижнее значения сигнала используются генератором формы сигнала для решения специальных случаев с экстремальными значениями в некоторых режимах работы.

На рисунке 3.53 показана схема блока управления устройством сравнения выхода.

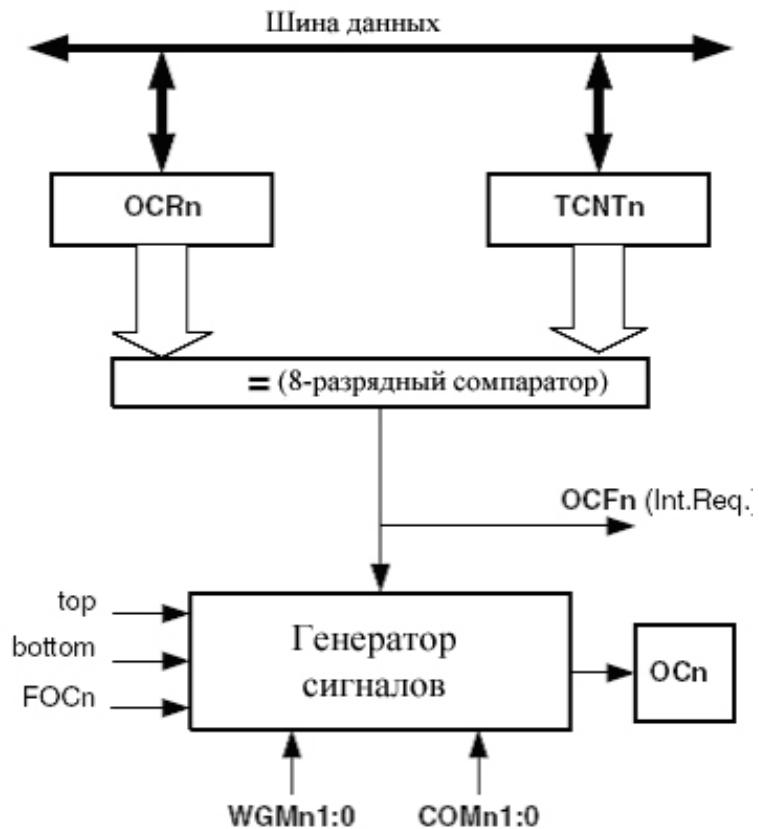


Рисунок 3.53 - Блок схема модуля сравнения выхода

При работе в любом режиме с использованием ШИМ (широкото-импульсная модуляция), регистр OCR2 имеет двойное буферирование. При работе в обычном режиме и режиме очистки таймера при сравнении (CTC), двойное буферирование отключается. Двойное буферирование синхронизирует обновление регистра сравнения либо вверх, либо вниз относительно последовательности подсчета. Синхронизация предотвращает возможновение импульсов избыточной длины, несимметричных импульсов ШИМ, обеспечивая таким образом бесперебойную работу выхода. Доступ к регистру может показаться достаточно сложным, но это не так. При активации двойного буферирования ЦПУ имеет доступ к регистру буфера OCR2, а если двойное буферирование отключено, то ЦПУ будет осуществлять прямую выборку OCR2.

Принудительное сравнение по выходу

В режимах генерации без использования ШИМ принципа, выход сравнения для компаратора может быть установлен на запись единицы в разряд принудительного сравнения выхода (FOC2). Принудительное сравнение/совпадение не будет устанавливать флаг OCF2 или перезагружать/сбрасывать таймер, однако вывод OC2 будет обновляться, как будто произошло реальное сравнение совпадений (установки разрядов COM2:0 определяют, что произойдет с выводом OC2, будет ли он установлен, сброшен или переключен).

Блокировка совпадения сравнения с помощью записи TCNT2

Все операции записи в регистр TCNT2 будут блокировать любое совпадение сравнений, происходящее в следующем цикле синхронизации таймера, даже в случае, когда таймер остановлен. Эта особенность позволяет инициализировать OCR2 на то же значение, что и TCNT2 без запуска прерывания, когда активируется тактовый генератор таймера/счетчика.

Использование блока сравнения выхода

Поскольку запись TCNT2 в любом рабочем режиме будет блокировать все совпадения для одного тактового цикла работы таймера, то существуют связанные с этим риски при смене TCNT2, когда используется канал сравнения выхода независимо от того, будет работать таймер/счетчик, или нет. Если значение, записанное в TCNT2, равно величине OCR2, то совпадение сравнений будет утрачено, что дает в результате неправильную генерацию формы сигнала, аналогично не следует записывать значение TCNT2, равное BOTTOM, когда счетчик отсчитывает счет вниз.

Настройка OC2 должна выполняться до установки регистра направления данных для вывода порта для выхода. Наиболее простым способом установки величины OC2 является использование разрядов стробирования принудительного сравнения выхода (FOC2) в нормальном режиме. Регистр OC2 сохраняет свое значение, даже если происходит изменение между режимами генерации формы сигнала.

Следует убедиться в том, что разряды COM21:0 не имеют двойного буферирования вместе с величиной сравнения. Изменение разрядов COM21:0 сразу же даст эффект.

Блок сравнения совпадений на выходе

Разряды режима сравнения на выходе (COM21:0) несут две функции. Генератор формы сигнала использует разряды COM21:0 для оценки состояния сравнения на выходе при следующем совпадении сравнений. Кроме этого, разряды COM21:0 управляют источником выхода для вывода OC2. На рисунке 3.54 представлена упрощенная блок-схема логики, используемой при установке разряда COM21:0. Показаны только части общих регистров управления портом входа/выхода DDR и PORT на которые влияют разряды COM21:0. При ссылках на состояние OC2 речь идет о внутреннем регистре OC2, а не о выводе OC2. Если происходит переустановка системы, то регистр OC2 переустанавливается на “0”.

Общая функция порта входа/выхода корректируется с помощью сравнения выхода (OC2) из генератора формы сигнала, если устанавливается какой-либо из разрядов COM21:0. Однако, направление вывода OC2 (вход или выход) все еще контролируется регистром направления данных DDR для вывода порта. Разряд регистра направления данных для вывода OC2 (DDR_OC2) должен быть установлен в качестве выхода до того, как величина OC2 станет видимой на выводе. Функция коррекции порта не зависит от режима генерации формы сигнала.

Логическая схема сравнения выхода позволяет инициализацию состояния OC2 до того, как активируется выход. Обратите внимание на то, что некоторые установки разряда COM21:0 сохраняются для определенных режимов работы.

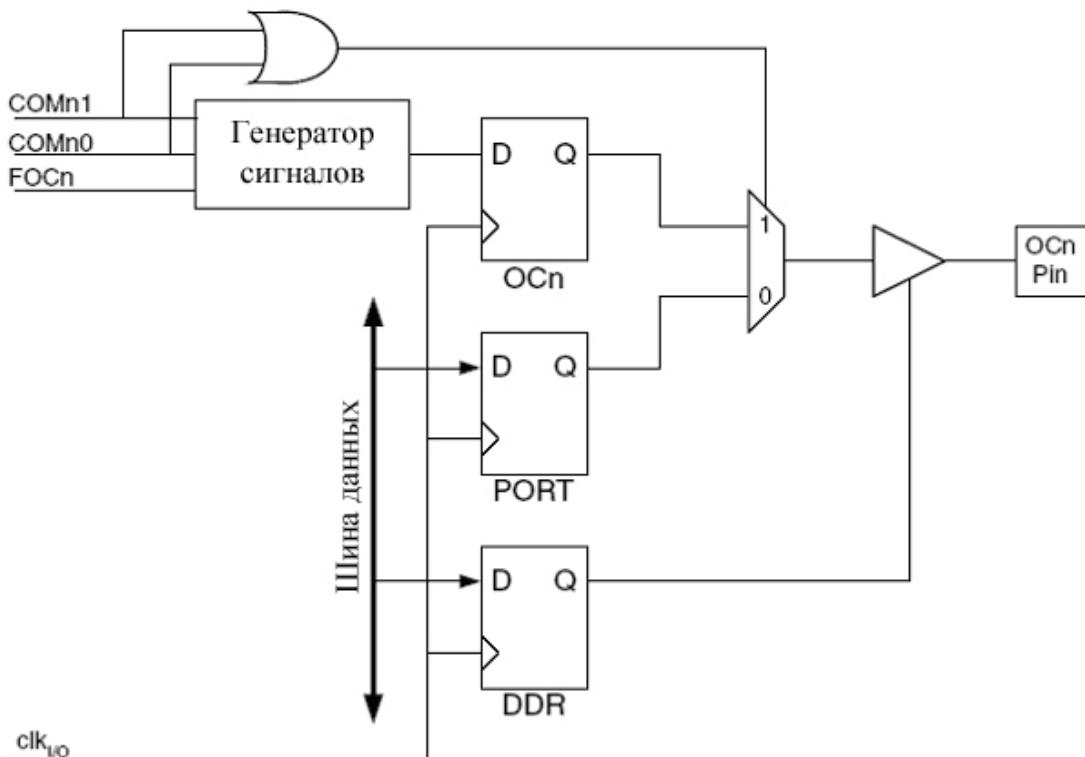


Рисунок 3.54 - Блок сравнения/совпадения на выходе

Логическая схема вывода сравнения выхода позволяет инициализацию состояния OC2 до того, как активируется выход. Обратите внимание на то, что некоторые установки разряда COM21:0 сохраняются для определенных режимов работы.

Режим сравнения выхода и генерация формы сигнала

Генератор формы сигнала использует разряды COM21:0 различным образом для обычного режима, режима условных переходов (СТС) и режима широтно-импульсной модуляции. Для всех типов режимов установка COM21:0 = 0 сообщает генератору формы сигнала, что в регистре OC2 не должно выполняться никаких действий по следующему сравнению/совпадению.

Изменение состояния разрядов в COM21:0 повлияет на первое совпадение сравнений после записи разрядов. Для режимов, отличающихся от ШИМ, может быть использовано действие, дающее немедленный эффект благодаря использованию разрядов стробирования FOC2.

Режимы работы

Режим работы, т. е., поведение таймера/счетчика и выводов сравнения выхода, определяется комбинацией разрядов для режима генерации формы сигнала (WGM21:0) и режима сравнения выхода (COM21:0). Разряды режима сравнения выхода не влияют на последовательность подсчета, в то время как разряды режима генерации формы сигнала влияют. Разряды COM21:0 управляют тем, будет инвертирован или нет сгенерированный выход ШИМ (инвертированный или неинвертированный ШИМ). Для режимов, не использующих ШИМ, разряды COM21:0 контролируют должен ли выход быть установлен, очищен или переключен при совпадении сравнений.

Нормальный режим

Самым простым режимом работы является нормальный режим ($WGM21:0 = 0$). В этом режиме направление подсчета всегда следует вверх (возрастает) и сброс счетчика не выполняется. Счетчик просто переполняется, когда он проходит свое максимальное 8-разрядное значение ($TOP = 0xFF$) и затем перезапускается ($0x00$). В нормальном режиме работы флаг переполнения счетчика/таймера (TOV2) будет установлен во время того же цикла синхронизации таймера, когда TCNT2 становится равным нулю. Флаг TOV2 в этом случае ведет себя как 9-й разряд, за исключением того, что он только устанавливается, а не сбрасывается. Однако, при объединении с прерыванием переполнения таймера, которое автоматически сбрасывает флаг TOV2, разрешение таймера можно повысить программным способом. В нормальном режиме нет необходимости рассматривать специальные случаи, при этом новое значение таймера может быть записано в любое время.

Блок сравнения выхода может использоваться для генерации прерываний в заданное время. Использование сравнения выхода для генерации формы сигнала в нормальном режиме не рекомендуется, поскольку это займет слишком много времени у ЦПУ.

Сброс таймера в режиме сравнения совпадения (CTC)

В режиме сброса таймера при сравнении или режиме CTC, $WGM21:0 = 2$, регистр OCR2 используется для управления разрешением счетчика. В режиме CTC счетчик сбрасывается до нуля, когда значение счетчика (TCNT2) совпадает с OCR2. OCR2 определяет верхнее значение для счетчика, а следовательно, его разрешение. Данный режим обеспечивает большую степень управления выходной частотой сравнения совпадения. Это упрощает также работу для подсчета внешних событий.

Временная диаграмма для режима CTC показана на рисунке 3.55. Значение счетчика (TCNT2) возрастает до тех пор, пока не произойдет совпадение между TCNT2 и OCR2, а затем счетчик (TCNT2) сбрасывается.

Прерывание может генерироваться каждый раз, когда значение счетчика достигает величины TOP, путем использования флага OCF2. Если прерывание разрешается, то процедуру прерывания программы обработчика можно использовать для обновления величины TOP. В то же время изменение TOP на значение, близкое к BOTTOM при работе счетчика, когда не имеется никакой величины или существует малое значение предварительного делителя частоты, должно выполняться с осторожностью, поскольку режим CTC не имеет двойного буферирования. Если новое значение, записанное в OCR2, ниже, чем текущее значение TCNT2, то счетчик упустит совпадение при сравнении. Затем счетчику придется считать до своего максимального значения ($0xFF$) и возвращаться (wrap around) снова, начиная с $0x00$, прежде, чем может произойти совпадение сравнений.

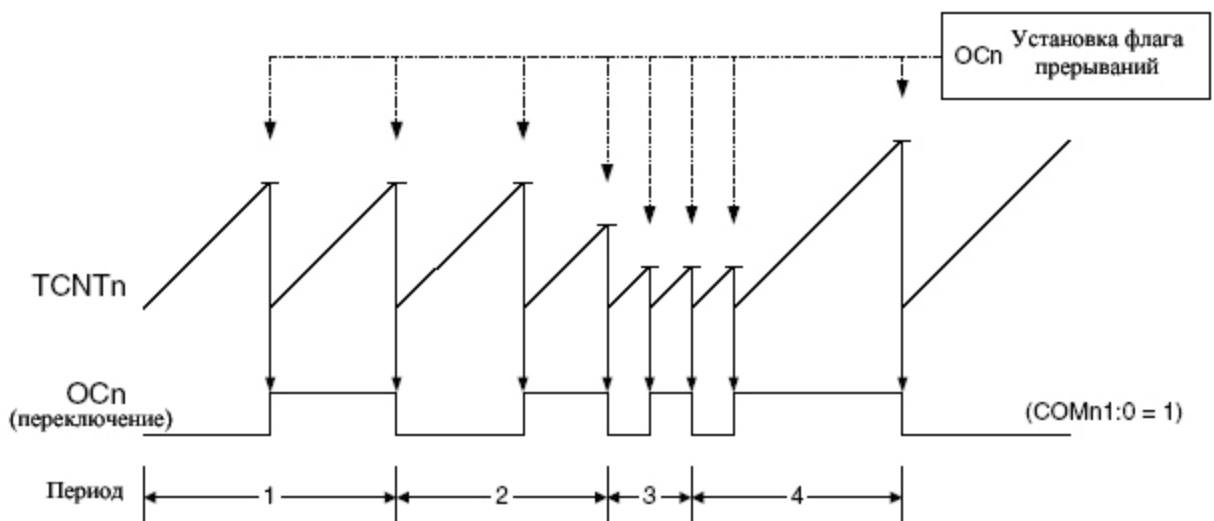


Рисунок 3.55 - Режим СТС, временная диаграмма

Для генерации выхода формы сигнала в режиме СТС выход OC2 может быть установлен на переключение своего логического уровня при каждом совпадении сравнения путем установки разрядов режима сравнения выхода на режим переключения (COM21:0 = 1). Значение OC2 не будет видно на выводе порта до тех пор, пока направление данных не установится для вывода. Генерируемая форма сигнала будет иметь максимальную частоту $f_{OC2} = f_{clk_I/O}/2$ при установке OCR2 на нуль (0x00). Частота для формы сигнала определяется следующим выражением:

$$f_{OCn} = \frac{f_{clk_I/O}}{2 \cdot N \cdot (1 + OCRn)}$$

Переменная “N” представляет собой коэффициент предварительного деления частоты (1, 8, 64, 256 или 1024).

Что касается нормального режима работы, то флаг TOV2 устанавливается во время того же цикла синхронизации таймера, при котором счетчик считает от MAX до 0x00.

Быстрый ШИМ режим

Быстрая широтно-импульсная модуляция или быстрый ШИМ режим (WGM21:0 = 3) предоставляет опцию генерации формы сигнала ШИМ с высокой частотой. Быстродействующая ШИМ отличается от другой опции ШИМ благодаря ее работе с использованием одиночных фронтов. Счетчик выполняет счет из положения BOTTOM до MAX, затем вновь начинает работу с BOTTOM. В неинвертируемом режиме сравнения выхода, сравнение по выходу (OC2) очищается при совпадении сравнения между TCNT2 и OCR2 и устанавливается как BOTTOM. В инвертируемом режиме сравнения выход устанавливается при совпадении сравнения и сбрасывается при BOTTOM. Благодаря работе с использованием одиночных фронтов, рабочая частота для быстрого ШИМ режима может в два раза превышать частоту ШИМ режима с фазовой коррекцией частоты, которая использует принцип работы с применением двух фронтов. Благодаря столь высокой частоте, быстрый ШИМ режим очень хорошо подходит для регулировки мощности, выпрямления, и применения в ЦАП. Высокая частота обуславливает малые физические размеры внешних компонентов (катушек, конденсаторов), благодаря чему снижается общая стоимость системы.

В быстром ШИМ режиме счетчик возрастает до тех пор, пока его значение не достигнет значения MAX. Затем счетчик сбрасывается при следующем цикле синхронизации таймера. Временная диаграмма для быстрого ШИМ режима показана на рисунке 3.56. Значение величины TCNT2 на временной диаграмме показано в виде гистограммы для иллюстрации работы в режиме с использованием одиночных фронтов. На схеме показаны как инвертированные, так и неинвертированные ШИМ выходы. Небольшие горизонтальные метки на фронтах TCNT2 представляют совпадения между OCR2 и TCNT2.

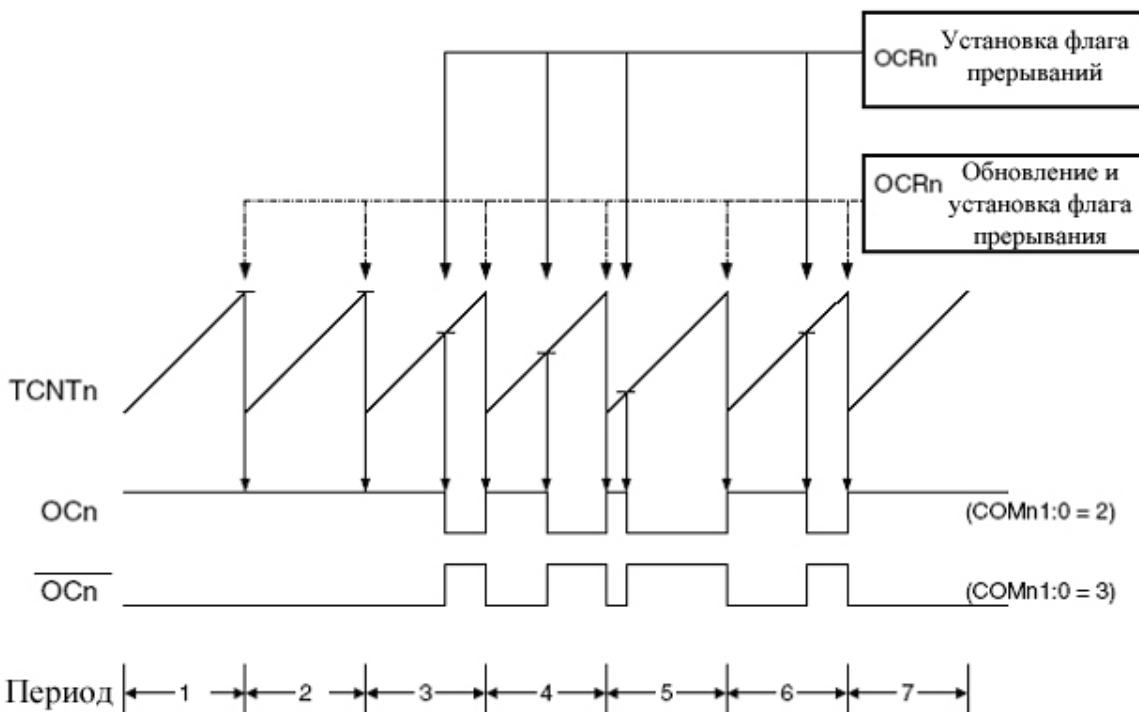


Рисунок 3.56 - Временная диаграмма для быстрого ШИМ режима

Флаг переполнения таймера/счетчика (TOV2) устанавливается каждый раз, когда счетчик достигает MAX. Если разрешено прерывание, то программа обработчика прерывания может использоваться для обновления величины сравнения.

В быстром ШИМ режиме, модуль сравнения разрешает генерацию ШИМ сигналов на выводе OC2. Установка разрядов COM21:0 на два дает неинвертированный ШИМ, а инвертированный ШИМ можно сформировать, установив COM21:0 на три. Фактическое значение OC2 будет видно на выводе входа, если направление данных для вывода выхода установлено как выход. Форма сигнала ШИМ генерируется с помощью установки (или очистки) регистра OC2 при сравнении совпадений между OCR2 и TCNT2 и очистки (или установки) регистра OC2, во время тактового периода счетчик очищается (изменяется с MAX на BOTTOM).

Частоту ШИМ для выхода можно вычислить, используя следующее уравнение:

$$f_{OCnPWM} = \frac{f_{clk_I/O}}{N \cdot 256}$$

Переменная “N” представляет собой коэффициент предварительного масштабирования (1, 8, 64, 256 или 1024).

Крайние значения для регистра OCR2 представляют особые случаи при генерации формы сигнала ШИМ на выходе в быстром ШИМ режиме. Если OCR2 устанавливается с

величиной, равной BOTTOM, то выходной сигнал будет представлять собой острый импульс для каждого тактового цикла таймера MAX+1. Если OCR2 устанавливается равным MAX, то это приводит к постоянному высокому или низкому выходу (в зависимости от полярности на выходе, которая установлена разрядами COM21:0).

Частота формы сигнала на выходе (при 50 % рабочем режиме) в быстром ШИМ режиме может быть получена путем установки OC2 на переключение своего логического уровня при каждом совпадении сравнений (COM21:0 = 1). Форма сигнала, генерируемая при этом, будет иметь максимальную частоту $f_{OC2} = f_{clk_I/O}/2$ при установке OCR2 на нуль. Эта особенность аналогична переключению OC2 в режиме СТС, за исключением того, что двойное буферирование в модуле сравнения выхода в быстром ШИМ режиме включено.

Режим фазовой коррекции ШИМ

Режим фазовой коррекции ШИМ (WGM21:0 = 1) обеспечивает опцию генерации сигнала ШИМ с фазовой коррекцией и высокой разрешающей способностью. Режим фазовой коррекции ШИМ основан на работе с использованием двух фронтов. Счетчик ведет повторяемый подсчет от BOTTOM к MAX, а затем от MAX к BOTTOM. В неинвертированном режиме сравнения OC2 очищается при совпадении между TCNT2 и OCR2 при инкременте и устанавливается при совпадении при декременте. В инвертированном режиме сравнения по выходу операция инвертируется. Принцип работы с двойным фронтом имеет пониженную максимальную рабочую частоту по сравнению одинарным фронтом. В то же время благодаря симметричной особенности ШИМ режимов с двойным фронтом, подобные режимы предпочтительны для использования при управлении двигателями.

Разрешение ШИМ для режима с фазовой коррекцией устанавливается на уровне 8бит. В ШИМ режиме с фазовой коррекцией счетчик возрастает до тех пор, пока его значение не достигнет MAX. Когда счетчик достигает MAX, он меняет направление счета. Значение TCNT2 будет равно MAX для одного тактового цикла таймера. Временная диаграмма для ШИМ режима с фазовой коррекцией показана на рисунке 3.57. Значение TCNT2 во временной диаграмме показано в виде гистограммы, иллюстрирующей принцип работы с двойным наклоном. Диаграмма включает неинвертированные и инвертированные ШИМ выходы. Небольшие горизонтальные отметки на наклонах TCNT2 представляют совпадения сравнений между OCR2 и TCNT2.

Флаг переполнения таймера/счетчика (TOV2) устанавливается каждый раз, когда счетчик достигает значения BOTTOM. Флаг прерывания может использоваться для прерывания каждый раз, когда счетчик достигает значения BOTTOM.

В режиме ШИМ с фазовой коррекцией модуль сравнения обеспечивает генерацию импульсов ШИМ на выводе OC2. Установка разрядов COM21:0 на значение два вызовет неинвертированный ШИМ режим. Инвертированный выход ШИМ может генерироваться путем установки COM21:0 на три.

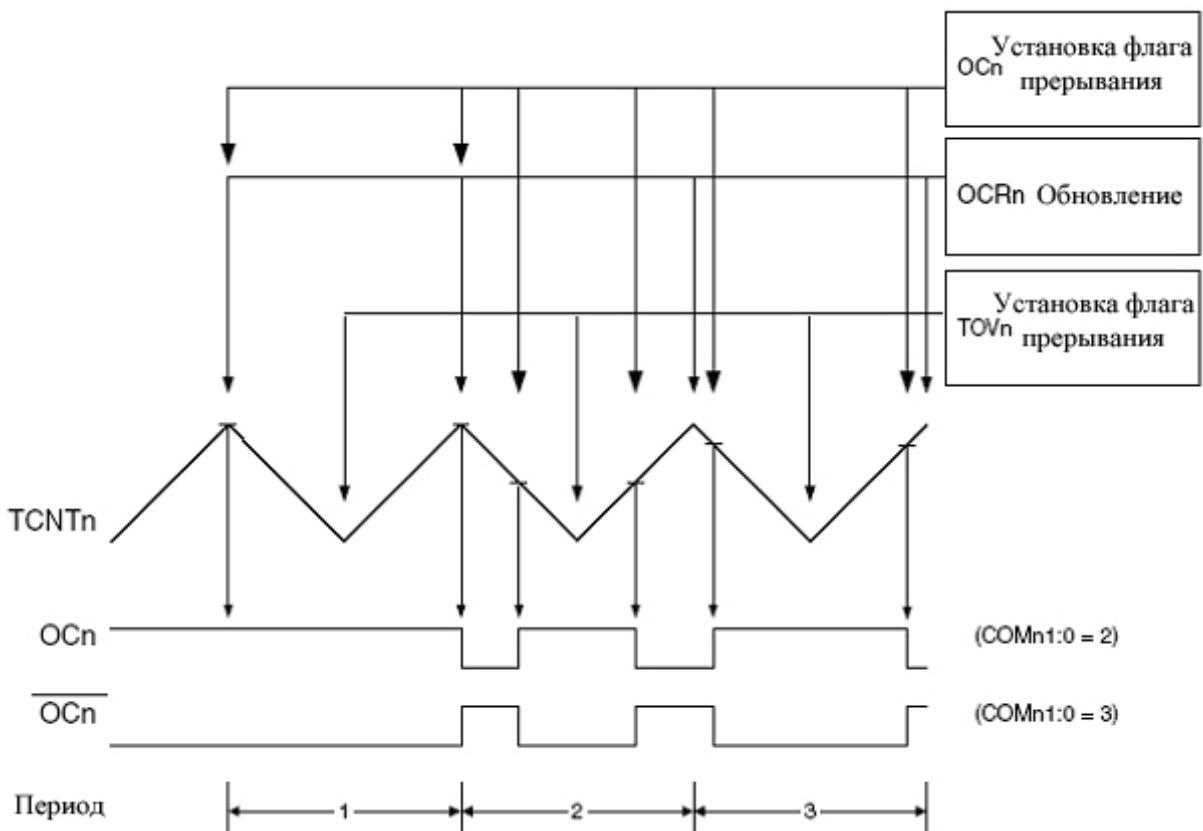


Рисунок 3.57 - Временная диаграмма, режим ШИМ с фазовой коррекцией

Фактическое значение OC2 будет видно на выводе порта только в случае, когда направление данных для вывода порта будет установлено как выход. Форма ШИМ сигнала генерируется путем сброса (или установки) регистра OC2 при совпадении между OCR2 и TCNT2, когда значения счетчика возрастают, и при установке (или сбросе) регистра OC2 при совпадении между OCR2 и TCNT2, когда значения счетчика уменьшаются. Частота ШИМ для выхода может быть вычислена с помощью следующего уравнения:

$$f_{OCnPCPWM} = \frac{f_{clk_I/O}}{N \cdot 510}$$

Переменная “ N” представляет собой коэффициент предварительного деления (1, 8, 64, 256 или 1024).

Крайние значения для регистра OCR2 представляют особые случаи, когда происходит генерация выходного ШИМ сигнала в ШИМ режиме с фазовой коррекцией. Если OCR2 устанавливается равным BOTTOM, то выход будет постоянно находиться в низком состоянии, а если устанавливается равным MAX, то выход для неинвертированного ШИМ режима будет постоянно находиться в высоком состоянии. Для инвертированного ШИМ режима выход будет иметь противоположные логические значения.

Временные диаграммы таймера/счетчика

Таймер/счетчик имеет синхронный принцип работы и тактовый импульс таймера (clk_{T2}) с учетом этого показан на следующих рисунках как сигнал синхрогенератора. Рисунки включают информацию о том, когда устанавливаются флаги прерывания. На рисунке 3.58 приведены временные параметры по основному режиму работы таймера/счетчика. На рисунке приведена последовательность счета, близкая к значению MAX во всех режимах, отличающихся от режима ШИМ с фазовой коррекцией.

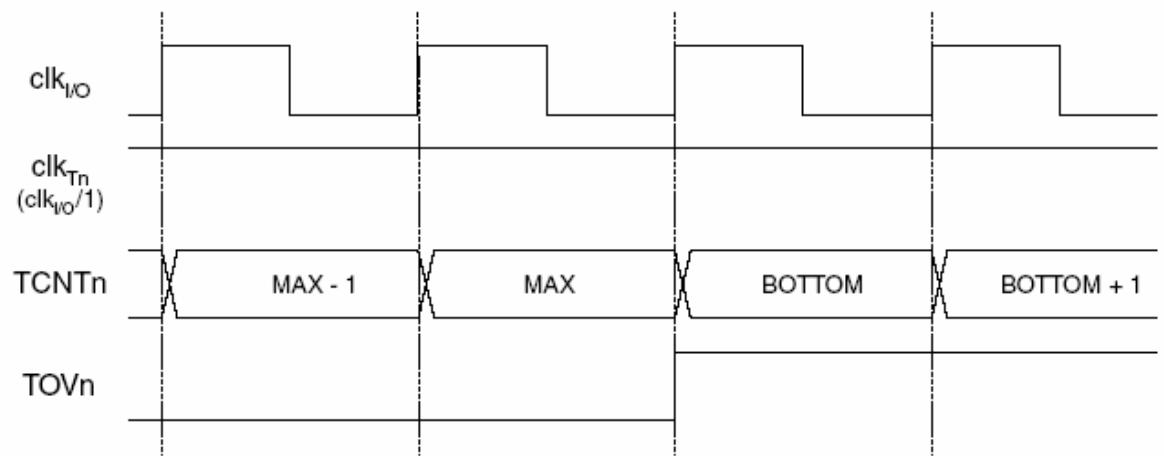


Рисунок 3.58 - Временная диаграмма таймера/счетчика без предварительного масштабирования

На рисунке 3.59 показаны те же временные параметры, но для включенного предварительного масштабирования.

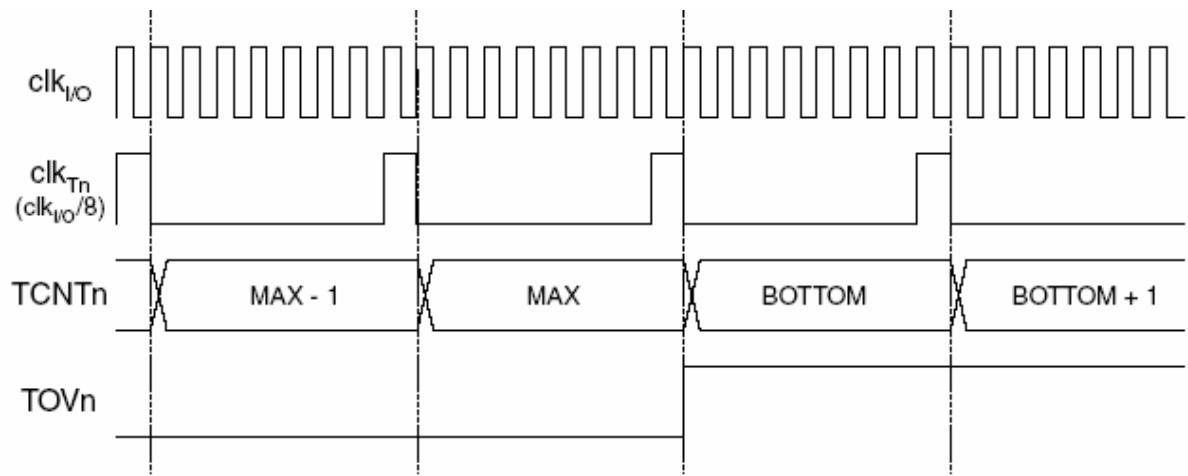


Рисунок 3.59 - Временная диаграмма таймера/счетчика с устройством предварительного масштабирования ($f_{\text{clk}_{\text{IO}}}/8$)

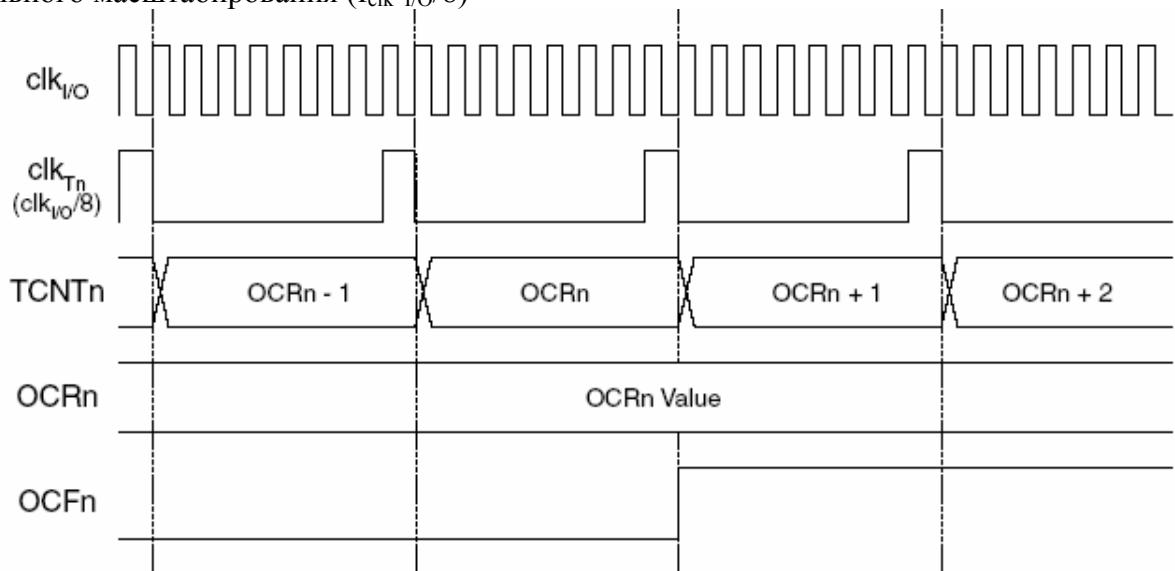


Рисунок 3.60 - Временная диаграмма таймера/счетчика с установкой OCF2 и устройством предварительного масштабирования ($f_{\text{clk}_{\text{IO}}}/8$)

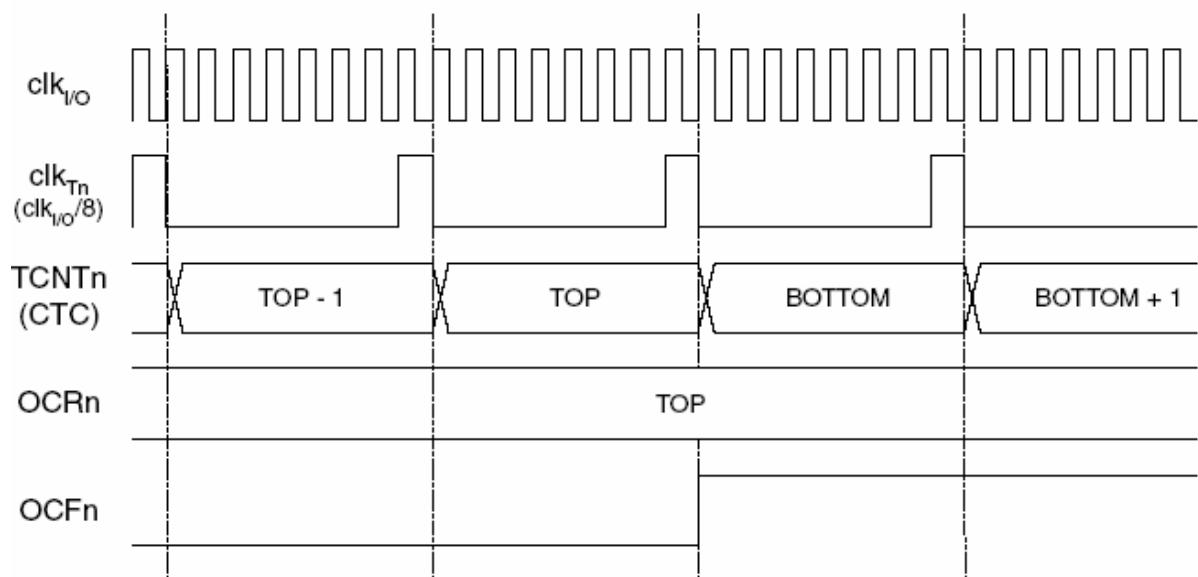


Рисунок 3.61 - Временная диаграмма таймера/счетчика, сброс таймера в режиме сравнения совпадения, устройством предварительного масштабирования ($f_{\text{clk_I/O}}/8$)

Описание 8-разрядного регистра таймера/счетчика

Регистр управления таймера/счетчика – TCCR2

Бит	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Чт./Зап.	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7: FOC2 - Принудительное сравнение выхода

Разряд FOC2 активируется только тогда, когда разряд WGM20 задает режим без использования ШИМ. В то же время для совместимости с будущими устройствами этот разряд должен быть установлен в ноль, когда записывается TCCR2 при работе в режиме ШИМ. При записи логической единицы в разряд FOC2 на модуль генерации формы сигнала подается команда на немедленное сравнение. Выход OC2 изменяется в соответствии с установкой его разрядов COM21:0. Следует обратить внимание на то, что разряд FOC2 реализуется в виде сигнала стробирования. С учетом этого, значение, присутствующее в разрядах COM21:0, определяет результат принудительного сравнения.

Стробирующий сигнал FOC2 не будет генерировать никакого прерывания, а также не будет очищать таймер в режиме CTC, используя OCR2 как TOP.

Разряд FOC2 всегда считывается как ноль.

Разряды 6, 3 - Режим генерации формы сигнала WGM21:0

Эти разряды управляют последовательностью счета для счетчика, источником максимального значения счетчика (TOP), а также используемым типом генерации формы сигнала. Модуль счетчика/таймера поддерживает следующие режимы работы: нормальный, режим сброса счетчика при совпадении (CTC), а также два типа режимов ШИМ.

Таблица 3.49 - Описание разряда режима работы генерации формы сигнала¹⁾

Режим	WGM21 (CTC0)	WGM20 (PWM0)	Режим работы таймера/счетчика	TOP	Обновление OCR2	Установка флага TOV2
0	0	0	Нормальный	0xFF	Немедленное	MAX
1	0	1	PWM, Коррекция фазы	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR2	Немедленное	MAX
3	1	1	Быстрый ШИМ	0xFF	TOP	MAX

¹⁾ Наименования разрядов CTC0 и PWM0 в настоящее время считаются устаревшими. Используйте определения WGM21:0. В то же время, функциональность и расположение этих разрядов совместимы с прежними версиями таймера.

Разряды 5, 4 :COM21:0 - Режим сравнения совпадения выхода

Эти разряды управляют поведением вывода сравнение выхода (OC2). Если устанавливается один или оба разряда COM21:0, то выход меняет нормальную функциональность порта входа/выхода. В то же время необходимо обратить внимание на то, что разряд регистра направления (DDR), соответствующий выводу OC2, должен быть установлен так, чтобы активировать выходной драйвер.

Когда к выводу подсоединен OC2, то функция разрядов COM21:0 зависит от установки разряда WGM21:0. В таблице 3.51 показана функциональность разряда COM21:0, когда разряды WGM21:0 устанавливаются в нормальный режим работы или в режим CTC (не режим ШИМ).

Таблица 3.51 - Режим сравнения выхода, режим без ШИМ

COM21	COM20	Описание
0	0	Нормальная работа порта, OC2 отсоединен.
0	1	Переключение OC2 на совпадение сравнений
1	0	Сброс OC2 на совпадение сравнений
1	1	Установка OC2 на совпадение сравнений

Таблица 3.52 показывает функциональность разряда COM21:0, когда разряды WGM21:0 устанавливаются на быстрый режим ШИМ.

Таблица 3.52 - Режим сравнения выхода, быстрый режим ШИМ¹⁾

COM21	COM20	Описание
0	0	Нормальная работа порта, OC2 отсоединен.
0	1	Зарезервировано
1	0	Сброс OC0 при совпадении сравнений, установка OC2 при TOP
1	1	Установка OC0 при совпадении сравнений, сброс OC0 при TOP

¹⁾ Особый случай имеет место, когда OCR2 равно TOP и установлен COM21. В этом случае совпадение при сравнении игнорируется, однако установка или сброс выполняются при TOP.

Таблица 3.53 показывает функциональность разряда COM21:0, когда разряды WGM21:0 установлены на режим фазовой коррекции ШИМ.

Таблица 3.53 - Режим сравнения по выходу, режим фазовой коррекции ШИМ

COM21	COM20	Описание
0	0	Нормальная работа порта, OC2 отсоединен.
0	1	Зарезервировано
1	0	Сброс OC0 при совпадении сравнений, при счете вверх. Установка OC2 при совпадении сравнений при подсчете вниз
1	1	Установка OC0 при совпадении сравнений, сброс OC0 при совпадении сравнений при счете вниз

Примечание - Особый случай имеет место, когда OCR2 равно TOP и установлен COM21. В этом случае совпадение при сравнении игнорируется, однако установка или сброс выполняются при TOP.

Разряды 2:0 : CS22:0 - Выбор синхронизации

Три разряда выбора синхронизации выбирают источник синхронизации для использования таймером/счетчиком.

Таблица 3.54 - Описание разряда выбора синхронизации

CS22	CS21	CS20	Описание
0	0	0	Нет источника синхронизации (Таймер/счетчик остановлен)
0	0	1	clk _{I/O} / (Без деления частоты)
0	1	0	clk _{I/O} /8 (От устройства предварительного деления частоты)
0	1	1	clk _{I/O} /64 (От устройства предварительного деления частоты)
1	0	0	clk _{I/O} /256 (От устройства предварительного деления частоты)
1	0	1	clk _{I/O} /1024 (От устройства предварительного деления частоты)
1	1	0	Внешний источник тактового сигнала на выводе T0. Импульс синхронизации на заднем фронте
1	1	1	Внешний источник тактового сигнала на выводе T0. Импульс синхронизации на переднем фронте

Если режимы подачи на внешний вывод используются для таймера/счетчика 0, то переходы на выводе T0 будут синхронизировать счетчик, даже если вывод конфигурируется как выход. Эта особенность позволяет обеспечить программное управление счетом.

Регистр таймера/счетчика – TCNT2

Бит	7	6	5	4	3	2	1	0	
	TCNT2[7:0]								TCNT2
Чт./Зап.	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр таймера/счетчика обеспечивает прямой доступ как для операции считывания, так и записи на 8-ми разрядный счетчик блока таймера/счетчика. Запись в блоки регистра TCNT2 удаляет совпадение при сравнении для следующего сигнала синхронизации таймера. Изменение счетчика (TCNT2) во время работы счетчика создает риск отсутствия совпадения при сравнении между TCNT2 и регистром OCR2.

Регистр сравнения выхода – OCR2

Бит	7	6	5	4	3	2	1	0	
	OCR2[7:0]								OCR2
Чт./Зап.	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр сравнения выхода содержит 8-разрядную величину, которая непрерывно сравнивается со значением счетчика (TCNT2). Совпадение может использоваться для генерации прерывания при сравнении по выходу или для генерации выходной формы сигнала на выводе OC2.

Регистр маски прерывания таймера/счетчика – TIMSK

Бит	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE1	TOIE0	TIMSK
Чт./Зап.	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 1: OCIE0 - Разрешение прерывания при совпадении сравнения на выходе таймера/счетчика

Если разряд OCIE0 записывается на единицу, а разряд I в регистре состояния установлен (единица), то запускается прерывание совпадения при сравнении таймера/счетчика 0. Соответствующее прерывание выполняется, если происходит совпадение при сравнении в таймере/счетчике 0 (т. е. когда разряд OCF2 установлен в регистре флага прерывания таймера/счетчика – TIFR).

Разряд 0 : TOIE0 - Разрешение прерывания переполнения таймера/счетчика0

Если разряд OCIE0 записывается на единицу, а разряд I в регистре состояния установлен (единица), то запускается прерывание переполнения таймера/счетчика 0. Соответствующее прерывание выполняется при появлении переполнения в таймере/счетчике 0 (т. е. соответствующее прерывание выполняется, если происходит переполнение в таймере/счетчике 0 (т. е. когда разряд TOV2 устанавливается в регистре флага прерывания таймера/счетчика – TIFR)

Регистр флага прерывания таймера/счетчика – TIFR

Бит	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF2	TOV2	TIFR
Чт./Зап.	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 1: OCF2 - Флаг 0 сравнения выхода

Разряд OCF2 устанавливается (единица), когда происходит совпадение при сравнении между таймером/счетчиком 0 и данными с OCR2 – регистре 0 сравнения по выходу. OCF2 очищается аппаратным способом при исполнении соответствующего вектора обращения с прерываниями. В качестве альтернативы OCF2 очищается путем записи логической единицы во флаг. Когда устанавливается разряд I в SREG, OCIE0 (разрешение прерывания совпадения при сравнении таймера/счетчика 0) и OCF2 на (единицу), то исполняется прерывание при сравнении совпадения таймера/счетчика 0.

Разряд 0: TOV2 - Флаг переполнения таймера/счетчика 0

Разряд TOV2 устанавливается (единица), когда происходит переполнение таймера/счетчика 0. TOV2 очищается аппаратным способом при исполнении соответствующего вектора обращения с прерываниями. В качестве альтернативы TOV2 очищается путем записи логической единицы во флаг. Когда устанавливаются разряды SREG I, TOIE0 (разрешение прерывания переполнения таймера/счетчика 0) и разряд TOV2 на (единицу), то исполняются прерывания при переполнении таймера/счетчика 0. В режиме фазовой коррекции ШИМ этот разряд устанавливается, когда таймер/счетчик 0 изменяет направление счета при 0x00.

Асинхронный режим

Отличительной особенностью таймера/счетчика T2 является его возможность работать в асинхронном режиме. В этом режиме на вход предделителя поступает сигнал с вывода TOSC1, что позволяет использовать таймер/счетчик в качестве часов реального времени. Источником сигнала может быть как кварцевый резонатор частотой 32768 Гц, подключаемый к выводам TOSC1 и TOSC2 микроконтроллера, так и внешняя схема. Несмотря на то, что тактовый генератор таймера/счетчика настроен на частоту 32768 Гц, частота сигнала от внешней схемы может лежать в пределах 0...256 кГц. При этом частота внешнего сигнала должна быть в четыре раза меньше частоты тактового сигнала микроконтроллера.

Непосредственная запись в регистры TCNT2, OCR2 и TCCR2 в асинхронном режиме синхронизируется с тактовым сигналом таймера/счетчика. При записи числа в любой из указанных регистров оно сохраняется в специальном временном регистре: своем для каждого регистра таймера/счетчика. А пересылка содержимого временного регистра в рабочий регистр таймера/счетчика осуществляется по третьему после записи положительному фронту сигнала на выводе TOSC1. Соответственно запись нового значения можно производить только после пересылки содержимого временного регистра в регистр таймера/счетчика.

Для определения момента действительного изменения регистров TCNT2, OCR2 и TCCR2, а также для переключения таймера/счетчика в асинхронный режим предназначен регистр ASSR, расположенный по адресу \$22 (\$42). Формат этого регистра приведен на рисунке 3.62.

	7	6	5	4	3	2	1	0
Чтение(R)/Запись(W)	—	—	—	—	AS2	TCN2UB	OCR2UB	TCR2UB
Начальное значение	R	R	R	R	R/W	R	R	R

Рисунок 3.62 – Регистр ASSR

Таблица 3.55 – Регистр состояния асинхронного режима ASSR

Разряд	Название	Описание
7...4	–	Зарезервированы, читаются как «0»
3	AS2	<p>Переключение режима работы Если разряд установлен в «1», на вход предделителя таймера/счетчика T2 поступают импульсы с вывода TOSC1 микроконтроллера (асинхронный режим). В этом режиме выводы TOSC1 и TOSC2 используются для подключения кварцевого резонатора и, соответственно, не могут использоваться как контакты ввода/вывода общего назначения.</p> <p>Если разряд сброшен в «0», на вход предделителя поступает внутренний тактовый сигнал микроконтроллера. В этом случае выводы TOSC1 и TOSC2 являются контактами ввода/вывода общего назначения.</p> <p>При изменении состояния этого разряда содержимое регистров TCNT2, OCR2 и TCCR2 может быть повреждено</p>
2	TCN2UB	<p>Состояние обновления регистра TCNT2 При записи в регистр TCNT2 этот флаг устанавливается в «1», а после пересылки записываемого значения в этот регистр флаг аппаратно сбрасывается в «0». Таким образом, сброшенный флаг TCN2UB означает, что регистр TCNT2 готов для записи в него нового значения.</p> <p>Запись в регистр TCNT2 при установленном флаге TCN2UB может привести к повреждению прежнего содержимого регистра и к генерации прерывания</p>
1	OCR2UB	<p>Состояние обновления регистра OCR2 При записи в регистр OCR2 этот флаг устанавливается в «1», а после пересылки записываемого значения в этот регистр флаг аппаратно сбрасывается в «0». Таким образом, сброшенный флаг OCR2UB означает, что регистр OCR2 готов для записи в него нового значения.</p> <p>Запись в регистр OCR2 при установленном флаге OCR2UB может привести к повреждению прежнего содержимого регистра и к генерации прерывания</p>
0	TCR2UB	<p>Состояние обновления регистра TCCR2 При записи в регистр TCCR2 этот флаг устанавливается в «1», а после пересылки записываемого значения в этот регистр флаг аппаратно сбрасывается в «0». Таким образом, сброшенный флаг TCR2UB означает, что регистр TCCR2 готов для записи в него нового значения.</p> <p>Запись в регистр TCCR2 при установленном флаге TCR2UB может привести к повреждению прежнего содержимого регистра и к генерации прерывания</p>

Необходимо отметить, что при переключении между синхронным и асинхронным режимами содержимое регистров таймера/счетчика может быть повреждено. Чтобы этого избежать, рекомендуется придерживаться следующей последовательности действий:

- 1 Запретить прерывания от таймера/счетчика T2 (разряды TOIE2 и OCIE2 регистра TIMSK).
- 2 Переключить таймер/счетчик в требуемый режим.
- 3 Записать новые значения в регистры TCNT2, OCR2 и TCCR2.
- 4 В случае переключения в асинхронный режим — ждать, пока флаги TCN2UB, OCR2UB и TCR2UB не будут сброшены.
- 5 Разрешить прерывания (если требуется).

При работе таймера/счетчика T2 в асинхронном режиме установка флагов прерываний от него производится синхронно с тактовым сигналом микроконтроллера. Для синхронизации требуется 3 машинных цикла плюс один период тактового сигнала таймера/счетчика. Поэтому к моменту, когда микроконтроллер сможет прочитать состояние счетчика, вызвавшее установку флага прерывания, оно изменится по меньшей мере на единицу. Изменение состояния вывода OC2 производится по тактовому сигналу таймера/счетчика и не синхронизируется с тактовым сигналом микроконтроллера.

Отдельно следует сказать о «взаимодействии» асинхронного режима таймера/счетчика T2 с режимами пониженного энергопотребления микроконтроллера Power Down и Power Save. Первое замечание касается использования прерываний от таймера/счетчика T2 для «пробуждения» микроконтроллера. В этом случае при переводе микроконтроллера в режим Power Save после записи в регистры таймера/счетчика необходимо

димо убедиться, что операция записи завершена (флаги TCN2UB, OCR2UB и TCR2UB сброшены). Наиболее важно это в случае, когда для «пробуждения» микроконтроллера используется прерывание от схемы сравнения. Дело в том, что во время записи в регистры TCNT2 и OCR2, функция сравнения выключена. Соответственно, если переход в режим Power Save произойдет до окончания операции записи в указанные регистры, прерывание от схемы сравнения никогда не произойдет и микроконтроллер не сможет выйти из спящего режима.

Вторая особенность связана с синхронизацией установки флагов прерываний от таймера/счетчика. При выходе микроконтроллера из режима Power Save по прерыванию от таймера/счетчика флаг соответствующего прерывания устанавливается только спустя 3 машинных цикла после запуска тактового генератора микроконтроллера. Во время этих циклов процессор выполняет команды, следующие за командой SLEEP, и только потом переходит к обработке прерывания (если оно разрешено). Кроме того, необходимо быть осторожным при повторном переходе в режим Power Save после выхода из него по прерыванию от таймера/счетчика T2. Дело в том, что в этом случае для запуска схемы прерываний требуется промежуток времени, равный одному периоду сигнала на выводе TOSC1. Если же промежуток времени между «пробуждением» и повторным переходом в режим Power Save будет меньше указанного, генерации прерывания и, соответственно, «пробуждения» микроконтроллера не произойдет. Для формирования задержки требуемой длительности рекомендуется после «пробуждения» микроконтроллера выполнить запись в какой-либо из регистров таймера/счетчика и дождаться завершения этой операции.

После подачи напряжения питания, а также после «пробуждения» микроконтроллера из режима Power Down таймер/счетчик рекомендуется использовать только спустя секунду после указанных событий. Эта задержка необходима для запуска тактового генератора таймера/счетчика. Соответственно при выходе из режима Power Down содержимое всех регистров таймера/счетчика T2 можно считать потерянным (из-за нестабильности тактового сигнала во время запуска генератора).

3.14 Последовательный периферийный интерфейс (SPI)

Последовательный периферийный интерфейс SPI (Serial Peripheral Interface), реализованный в микроконтроллерах семейства, имеет два назначения. Прежде всего через него может быть осуществлено программирование микроконтроллера (так называемый режим последовательного программирования). Использование интерфейса SPI в этом качестве будет описано в главе программирования ИМС.

Вторым назначением интерфейса является организация высокоскоростного обмена данными между микроконтроллером и различными периферийными устройствами, такими как цифровые потенциометры ЦАП/АЦП, Flash_ПЗУ и др. Посредством этого интерфейса также может производиться обмен данными между несколькими микроконтроллерами. Использование интерфейса SPI в качестве высокоскоростного канала связи и рассматривается в данном подразделе.

При обмене данными по интерфейсу SPI микроконтроллер AVR может работать как в режиме Master, так и в режиме Slave. При этом пользователь может задать следующие параметры:

- скорость передачи (четыре программируемых значения);
- формат передачи (от младшего разряда к старшему или наоборот).

Дополнительной возможностью подсистемы SPI является «пробуждение» микроконтроллера из режима Idle при поступлении данных.

3.14.1 Функционирование модуля

Структурная схема модуля SPI приведена на рисунке 3.63.

Модуль SPI использует четыре вывода микроконтроллера. Как и для большинства прочих периферийных устройств эти выводы являются линиями порта ввода-вывода общего назначения.

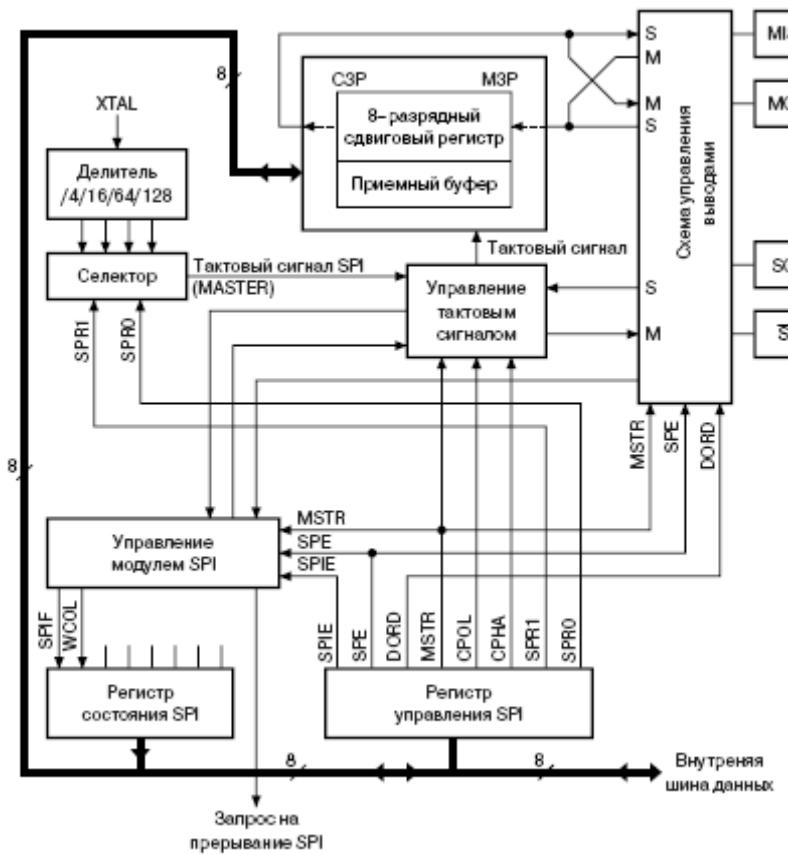


Рисунок 3.63 – Структурная схема модуля SPI

Таблица 3.56 – Выводы, используемые модулем SPI

Вывод	Режим Master	Режим Slave
MOSI	Определяется пользователем*	Вход
MISO	Вход	Определяется пользователем*
SCK	Определяется пользователем*	Вход
SS	Определяется пользователем*	Вход

* Смотрите подраздел 3.8.8 “Альтернативные функции порта B” для детального описания, как определить направление выводов SPI.

Как видно из таблицы 3.56, в некоторых случаях пользователь должен самостоятельно задать режим работы вывода, используемого модулем SPI, в соответствии с его назначением (см. далее по тексту). Причем возможность управления внутренними подтягивающими резисторами выводов, работающих как входы, сохраняется независимо от способа управления их режимом работы.

Для управления модулем SPI предназначен регистр управления SPCR, расположенный по адресу \$0D (\$2D). Формат этого регистра приведен на рисунке 3.64, а краткое описание функций разрядов регистра приведено в таблице 3.57. Подробно использование различных разрядов регистра будет описано далее.

	7	6	5	4	3	2	1	0
Чтение(R)/Запись(W)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Начальное значение	R/W 0							

Рисунок 3.64 – Формат регистра SPCR

Таблица 3.57 – Регистр SPCR

Разряд	Название	Описание
7	SPIE	Разрешение прерывания от SPI
6	SPE	Включение/выключение SPI
5	DORD	Порядок передачи данных
4	MSTR	Выбор режима работы (Master/Slave)
3	CPOL	Полярность тактового сигнала
2	CPHA	Фаза тактового сигнала
1, 0	SPR1:SPR0	Скорость передачи

Контроль состояния модуля осуществляется с помощью регистра состояния SPSR (доступен только для чтения), расположенного по адресу \$0E (\$2E). Формат этого регистра приведен на рисунке 3.65, а назначение его разрядов описано в таблице 3.58.

	7	6	5	4	3	2	1	0
Чтение(R)/Запись(W)	SPIF	WCOL	–	–	–	–	–	–
Начальное значение	R 0							

Рисунок 3.65 – Формат регистра SPSR

Таблица 3.58 – Описание регистра SPSR

Разряд	Название	Описание
7	SPIF	Флаг прерывания от SPI Данный флаг устанавливается в «1» по окончании передачи очередного байта. Если флаг SPIE регистра SPCR установлен в «1» и прерывания разрешены, одновременно с установкой флага генерируется прерывание от SPI. Также флаг SPIF устанавливается в «1» при переводе микроконтроллера из режима Master в режим Slave посредством вывода SS (см. раздел 10.3). Флаг сбрасывается аппаратно либо при старте подпрограммы обработки прерывания, либо после чтения регистра состояния SPI с последующим обращением к регистру данных SPI (SPDR)
6	WCOL	Флаг конфликта записи Данный флаг устанавливается в «1» при попытке записи в регистр данных (SPDR) во время передачи очередного байта. Флаг сбрасывается аппаратно после чтения регистра состояния SPI с последующим обращением к регистру данных SPI
5..0	–	Зарезервированы, читаются как «0»

Передаваемые данные записываются, а принимаемые - считаются из регистра данных SPDR, расположенного по адресу \$0F (\$2F). Запись в этот регистр инициирует начало передачи, а при его чтении считывается содержимое приемного буфера сдвигового регистра. Поэтому этот регистр можно назвать буфером между регистровым файлом микроконтроллера и сдвиговым регистром модуля SPI.

Соединение двух микроконтроллеров (ведущий - ведомый) по интерфейсу SPI показано на рисунке 3.66. Вывод SCK ведущего микроконтроллера является выходом тактового сигнала, а ведомого микроконтроллера - входом.

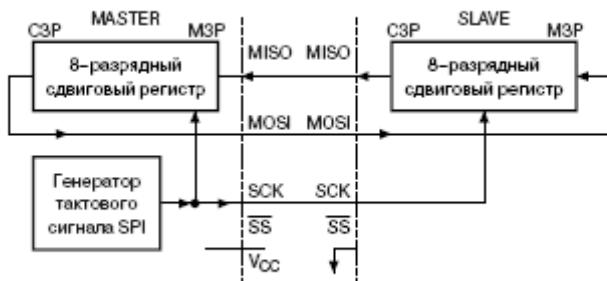


Рисунок 3.66 – Соединение микроконтроллеров по интерфейсу SPI

Перед выполнением обмена необходимо прежде всего разрешить работу модуля SPI. Для этого следует установить в «1» разряд SPE регистра SPCR. Режим работы определяется состоянием разряда MSTR этого регистра: если разряд установлен в «1», микроконтроллер работает в режиме Master (ведущий), если сброшен в «0» — в режиме Slave (ведомый).

Передача данных осуществляется следующим образом. При записи в регистр данных SPI ведущего микроконтроллера запускается генератор тактового сигнала модуля SPI, и данные начинают поразрядно выдаваться на вывод MOSI и, соответственно, поступать на вывод MOSI ведомого микроконтроллера. Порядок передачи разрядов данных определяется состоянием разряда DORD регистра SPCR. Если разряд установлен в «1», первым передается младший разряд байта, если же сброшен в «0» — старший разряд. После выдачи последнего разряда текущего байта генератор тактового сигнала останавливается с одновременной установкой в «1» флага «Конец передачи» (SPIF). Если прерывания от модуля SPI разрешены (флаг SPIE регистра SPCR установлен в «1»), генерируется запрос на прерывание. При подключении к ведущему устройству нескольких ве-

домых, что разрешено спецификацией SPI, выбор конкретного ведомого устройства осуществляется подачей на его вход SS# сигнала НИЗКОГО уровня.

Образно говоря, два сдвиговых регистра ведомого и ведущего устройств можно считать одним распределенным 16-разрядным циклическим сдвиговым регистром как показано на рисунке 3.66. Одновременно с передачей данных от ведущего к ведомому происходит передача и в обратном направлении. Таким образом, в каждом цикле сдвига происходит обмен данными между устройствами.

В модуле используется одинарная буферизация при передаче и двойная — при приеме. Это означает, что готовый для передачи байт данных не может быть записан в регистр данных SPI до окончания предыдущего цикла обмена. При попытке изменить содержимое регистра данных во время передачи устанавливается в «1» флаг WCOL регистра SPSR. Сбрасывается этот флаг после чтения регистра SPSR с последующим обращением к регистру данных SPI. Соответственно во время приема принятый байт должен быть прочитан из регистра данных SPI до того, как в сдвиговый регистр поступит последний разряд следующего байта. В противном случае первый байт будет потерян.

3.14.2 Режимы передачи данных

Спецификация интерфейса SPI предусматривает 4 режима передачи данных. Эти режимы различаются соответствием между фазой (момент считывания сигнала) тактового сигнала SCK, его полярностью и передаваемыми данными. Всего существует 4 такие комбинации, определяемые состоянием разрядов CPOL и CPHA регистра SPCR (см. таблицу 3.59).

Таблица 3.59 – Задание режима передачи данных

Разряд	Описание
CPOL	Полярность тактового сигнала 0 — генерируются импульсы положительной полярности, при отсутствии импульсов на выводе присутствует НИЗКИЙ уровень; 1 — генерируются импульсы отрицательной полярности, при отсутствии импульсов на выводе присутствует ВЫСОКИЙ уровень
CPHA	Фаза тактового сигнала 0 — обработка данных производится по переднему фронту импульсов сигнала SCK (для CPOL = 0 — по нарастающему, а для CPOL = 1 — по спадающему фронту). 1 — обработка производится по заднему фронту импульсов сигнала SCK

Соответствующие этим режимам форматы обмена данными через SPI приведены на рисунках 3.67 и 3.68 (передача ведется от старшего разряда к младшему). Частота тактового сигнала SCK и, соответственно, скорость передачи данных по интерфейсу определяются состоянием разрядов SPR1:SPR0 регистра SPCR (см. таблицу 3.60). Речь идет о микроконтроллере, работающем в режиме Master, т. к. именно он является источником тактового сигнала. Для устройства, находящегося в режиме Slave, состояние этих разрядов безразлично.

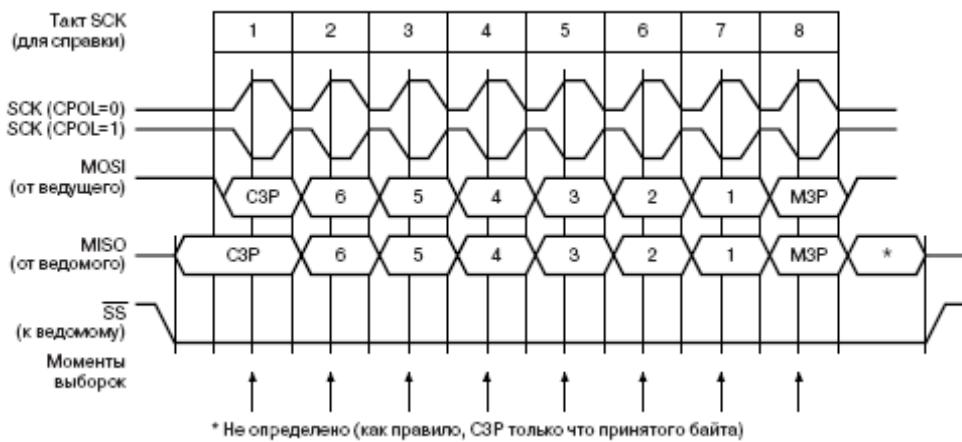
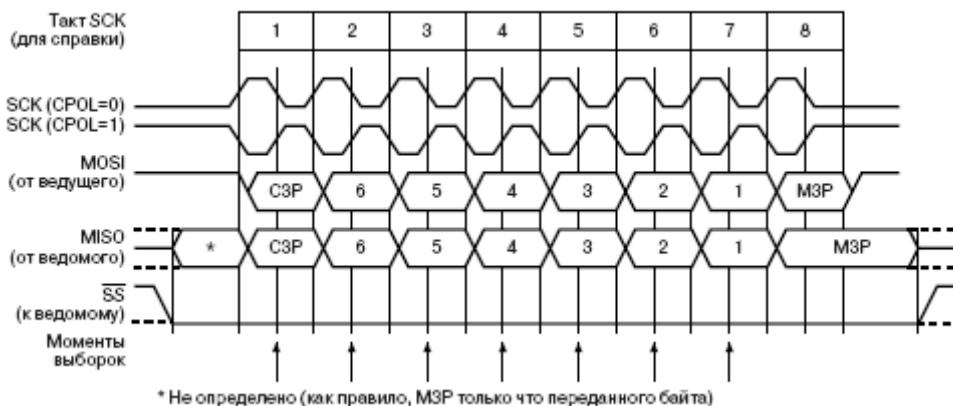
Рисунок 3.67 – Передача данных при $\text{CPHA} = 0$ и $\text{DORD} = 0$ Рисунок 3.68 – Передача данных при $\text{CPHA} = 1$ и $\text{DORD} = 0$

Таблица 3.60 – Задание частоты тактового сигнала

SPR1	SPR0	Частота сигнала SCK
0	0	$F_{\text{CLK}}/4^*$
0	1	$F_{\text{CLK}}/16$
1	0	$F_{\text{CLK}}/64$
1	1	$F_{\text{CLK}}/128$

3.14.3 Использование вывода SS#

Этот вывод предназначен для выбора активного ведомого устройства и в режиме Slave всегда является входом. При подаче на него напряжения НИЗКОГО уровня модуль SPI активируется и вывод MOSI переключается в режим вывода данных (если это задано пользователем). Остальные выводы модуля SPI являются в этом режиме входами. А при подаче на вывод SS# напряжения ВЫСОКОГО уровня все выводы модуля SPI переключаются в режим ввода данных. При этом модуль переходит в неактивное состояние и прием данных не производится.

Следует помнить, что каждый раз, когда на вывод SS# подается напряжение ВЫСОКОГО уровня, происходит сброс модуля SPI. Соответственно, если изменение состояния этого вывода произойдет во время передачи данных, то прием и передача немедленно прекратятся, а передаваемый и принимаемый байты будут потеряны.

Если же микроконтроллер находится в режиме Master (разряд MSTR регистра SPCR установлен в «1»), направление передачи данных через вывод SS# определяется пользователем. Если вывод сконфигурирован как выход, он работает как линия вывода общего назначения и не влияет на работу модуля SPI. Если же он сконфигурирован как вход, то для обеспечения нормальной работы модуля SPI на него должен быть подан сигнал ВЫСОКОГО уровня. Дело в том, что подача на этот вход сигнала НИЗКОГО уровня от какой-либо внешней схемы будет воспринята модулем SPI как выбор данного микроконтроллера в качестве ведомого и, соответственно, начало передачи ему данных. Во избежание конфликта на шине система SPI в таких случаях выполняет следующие действия:

1 Флаг MSTR регистра SPCR сбрасывается, и микроконтроллер переключается в режим Slave. Как следствие, выводы MOSI и SCK начинают функционировать как входы.

2 Устанавливается флаг SPIF регистра SPSR, генерируя запрос на прерывание от SPI. Если прерывания от SPI разрешены и флаг I регистра SREG установлен в «1», происходит запуск подпрограммы обработки прерывания.

Таким образом, если ведущий микроконтроллер использует передачу данных, управляемую прерыванием, и существует вероятность подачи на вход SS# сигнала НИЗКОГО уровня, в подпрограмме обработки прерывания от SPI обязательно должна осуществляться проверка состояния флага MSTR.

При обнаружении сброса этого флага он должен быть программно установлен обратно в «1» для обратного перевода микроконтроллера в режим Master.

3.15 Последовательный синхронно-асинхронный приемопередатчик (USART)

Универсальный синхронный и асинхронный последовательный приемопередатчик (УСАПП) предназначен для организации гибкой последовательной связи.

Отличительные особенности:

- полнодуплексная работа (раздельные регистры последовательного приема и передачи);
- асинхронная или синхронная работа;
- ведущее или подчиненное тактирование связи в синхронном режиме работы;
- высокая разрешающая способность генератора скорости связи;
- поддержка формата передаваемых данных с 5, 6, 7, 8 или 9 битами данных и 1 или 2 стоп-битами;
- аппаратная генерация и проверка бита паритета (четность/нечетность);
- определение переполнения данных;
- определение ошибки в структуре посылки;
- фильтрация шума с детекцией ложного старт-бита и цифровым ФНЧ;
- три раздельных прерывания по завершении передачи, освобождении регистра передаваемых данных и завершении приема;
- режим многопроцессорной связи;
- режим удвоения скорости связи в асинхронном режиме.

3.15.1 Краткий обзор

В состав УСАПП входят три основных блока: тактовый генератор, передатчик и приемник. Регистры управления используются всеми блоками. Логика тактового генератора состоит из логики синхронизации, связанной с внешним тактовым входом (используется в подчиненном режиме), и генератора скорости связи. Вывод XCK (синхронизация передачи) используется только в режиме синхронной передачи. Передатчик состоит из одного буфера записи, последовательного сдвигового регистра, генератора паритета и управляющей логики, которая поддерживает различные форматы последовательной посылки. Буфер записи позволяет непрерывно передавать данные без каких-либо задержек между передачей посылок. Приемник является более сложным блоком УСАПП, т. к. в его состав входят модули обнаружения данных и синхронизации. Модули обнаружения необходимы для асинхронного приема данных. Помимо модулей обнаружения в приемник входят: устройство проверки паритета, сдвиговый регистр и двухуровневый приемный буфер (UDR). Приемник поддерживает те же последовательные форматы, что и передатчик, и может определить ошибку в посылке (кадре), переполнение данных и ошибку паритета.

Логика генерации тактовых импульсов формирует основную синхронизацию приемника и передатчика. УСАПП поддерживает четыре режима работы синхронизации: нормальная асинхронная, асинхронная с удвоением скорости, ведущая синхронная и подчиненная синхронная. Бит UMSEL в регистре С управления и статуса (UCSRC) позволяют выбрать асинхронную или синхронную работу. Удвоение скорости (только в асинхронном режиме) управляется битом U2X в регистре UCSRA. При использовании синхронного режима (UMSEL = 1) соответствующий бит в регистре направления данных для вывода XCK (DDR_XCK) задает: будет синхронизация внутренней (ведущий режим) или внешней (подчиненный режим). Вывод XCK активен только при использовании синхронного режима.

На рисунке 3.69 показана функциональная схема логики синхронизации.

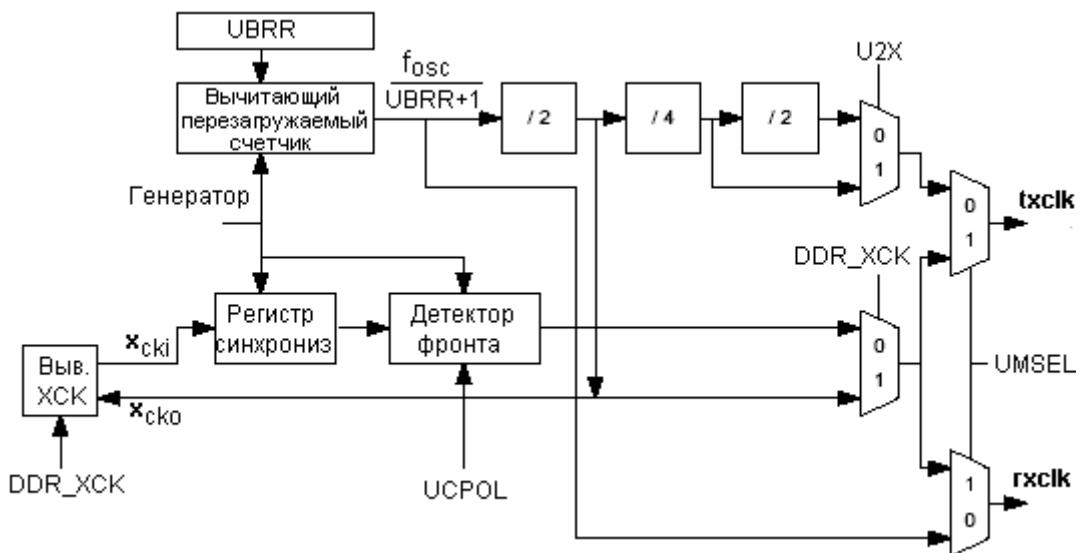


Рисунок 3.69 - Функциональная схема логики синхронизации УСАПП

Описание сигналов:

- txclk - синхронизация передатчика (внутренний сигнал);
- rxclk - основная синхронизация приемника (внутренний сигнал);

- x_{cki} - вход от вывода XCK (внутренний сигнал). Используется для синхронной подчиненной работы;

- x_{cko} - выход синхронизации к выводу XCK (внутренний сигнал). Используется в ведущем синхронном режиме;

- f_{osc} - вывод частоты XTAL (системная синхронизация).

Генерация внутренней синхронизации - генератор скорости связи.

Внутренняя синхронизация используется для асинхронного и ведущего синхронного режимов работы.

Регистр генератора скорости связи (UBRR) и связанный с ним вычитающий счетчик функционируют как программируемый предделитель или генератор скорости связи. Вычитающий счетчик тактируется системной синхронизацией (f_{osc}) и перезагружается значением из регистра UBRRR всякий раз при достижении нулевого значения или после записи регистра UBRRRL. Тактовый сигнал генерируется всякий раз при достижении счетчиком нулевого значения. Данный тактовый сигнал является тактовым выходом генератора скорости связи ($= f_{osc}/(UBRR+1)$). Передатчик делит частоту генератора скорости связи на 2, 8 или 16 в зависимости от режима работы. Модули обнаружения синхронизации и данных приемника подключены непосредственно к тактовому выходу генератора скорости связи. Цифровой автомат модулей обнаружения использует 2, 8 или 16 состояний в зависимости от режима, задаваемого битами UMSEL, U2X и DDR_XCK.

Таблица 3.61 содержит выражения для вычисления скорости связи (в битах в секунду) и вычисления значений UBRR для каждого из рабочих режимов при использовании внутренне генерируемого тактового источника.

Таблица 3.61 - Выражения для вычисления установок регистра скорости связи

Режим работы	Выражение для вычисления (1) скорости связи	Выражение для вычисления значения UBRR
Нормальный асинхронный режим (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Асинхронный режим с удвоением скорости (U2X=1)	$BAUD = \frac{f_{OSC}}{8(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Синхронный ведущим режим	$BAUD = \frac{f_{OSC}}{2(UBRR + 1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$
Примечания 1 BAUD - скорость связи (в битах в секунду, бод). 2 fosc - частота синхронизации системного генератора. 3 UBRR - Содержимое регистров UBRRH и UBRL (0 ... 4095).		
¹⁾ Скорость связи представлена в битах в секунду (бод).		

3.15.2 Работа с удвоением скорости связи (U2X)

Скорость передачи данных может быть удвоена, если установить бит U2X в регистре UCSRA. Установка данного бита оказывает действие только в асинхронном режиме. При использовании синхронного режима необходимо установить нулевое значение данного бита.

Установка данного бита приводит к уменьшению коэффициента деления частоты генератора скорости связи с 16 до 8, тем самым удваивая скорость асинхронной связи. Следует обратить внимание, что в этом случае приемник сокращает количество выборок с 16 до 8 при обнаружении синхронизации и данных, поэтому, при использовании данного режима необходимо использовать более точные установки скорости связи и более стабильный тактовый источник. Для передатчика удвоение скорости не связано с какими-либо ограничениями.

3.15.3 Внешняя синхронизация

Внешняя синхронизация используется в синхронном подчиненном режиме работы (см. рисунок 3.70).

Во избежание возможности возникновения метастабильности вход внешней синхронизации с вывода XCK связан с регистром синхронизации. Выход регистра синхронизации проходит через детектор фронтов, а только затем используется приемником и передатчиком. На данный процесс затрачивается два такта синхронизации ЦПУ и поэтому максимальная частота внешней синхронизации на выводе XCK ограничивается следующим выражением:

$$f_{XCK} < f_{OSC} / 4$$

Частота f_{osc} зависит от стабильности системного источника синхронизации. В связи с этим рекомендуется учесть некоторый запас для предотвращения возможности потери данных из-за колебаний частоты.

3.15.4 Режим синхронной связи

Если используется режим синхронной связи ($UMSEL = 1$), то вывод XCK используется или как вход синхронизации (подчиненный режим) или как выход синхронизации (ведущий режим). Зависимость между тактовыми фронтами и выборкой данных или изменением данных одна и та же. Основной принцип работы заключается в том, что выборка вводимых данных (на RXD) осуществляется фронтом XCK, который противоположен фронту, по которому происходит изменение выходных данных (на TXD).

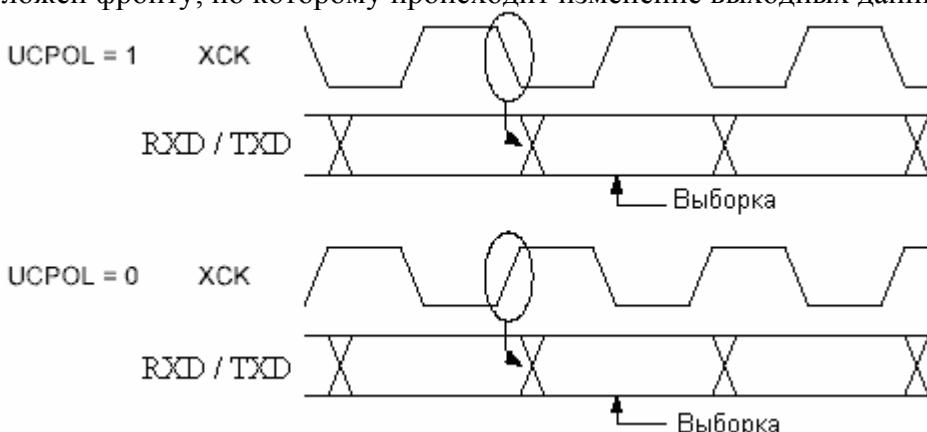


Рисунок 3.70 - Временная диаграмма для синхронного режима XCK

Бит UCPOL регистра UCRSC выбирает, какой фронт XCK используется для выборки данных, а какой для изменения данных. На рисунке 3.70 показано, что при $UCPOL=0$ изменение данных происходит по нарастающему фронту XCK, а выборка по падающему фронту XCK. Если установлен бит $UCPOL=1$, то изменение данных происходит по падающему фронту XCK, а выборка по нарастающему фронту XCK.

Последовательная посылка состоит из бит данных, бит синхронизации (старт и стоп-биты), а также опционального бита паритета для поиска ошибок. УСАПП поддерживает все 30 комбинаций следующих форматов посылок:

- 1 старт-бит;
- 5, 6, 7, 8 или 9 бит данных;
- без паритета, с битом четности, с битом нечетности;
- 1 или 2 стоп-бита.

Посылка начинается со старт-бита, а за ним следует передача бит данных, начиная с самого младшего разряда. Затем следует передача остальных бит данных (максимальное число бит данных 9), которая заканчивается передачей старшего разряда дан-

ных. Если разрешена функция контроля паритета, то сразу после бит данных передается бит паритета, а затем стоп-биты. После завершения передачи посылки имеется возможность либо передавать следующую посылку, либо перевести линию связи в состояние ожидания (высокий уровень). Рисунок 3.71 иллюстрирует возможность сочетания форматов посылки. Наличие прямоугольной скобки указывает наoptionalность данного формата посылки.



- St - старт-бит имеет всегда низкий уровень;
- 0...8 - номера бита данных;
- P - бит паритета: четность или нечетность;
- Sp1, Sp2 - стоп-бит имеет всегда высокий уровень;
- IDLE - состояние ожидания, в котором приостановлена передача на RXD или TXD. В состоянии ожидания на линии должен быть высокий уровень.

Рисунок 3.71 – Сочетение форматов посылки

Формат посылки, который используется УСАПП, задается битами UCSZ2:0, UPM1:0 и USBS в регистрах UCSRB и UCSRC. Приемник и передатчик используют одни и те же установки форматов. Следует обратить внимание, что изменение установок любого из этих бит может привести к повреждению текущего сеанса связи как для приемника, так и для передатчика.

Биты выбора длины передаваемых данных (UCSZ2:0) определяют из скольких бит данных состоит посылка. Биты режима паритета (UPM1:0) разрешают передачу/контроль бита паритета и устанавливают тип паритета: четность, нечетность. Выбрать один или два стоп-бита позволяет бит выбора стоп-бита УСАПП (USBS). Приемник игнорирует второй стоп-бит. Флаг ошибки посылки FE позволяет выявить ситуацию, когда первый стоп-бит равен 0.

Вычисление бита паритета

Бит паритета вычисляется путем выполнения логической операции исключающего ИЛИ над всеми битами данных. Если используется нечетность, то результат этой операции инвертируется. Сказанное отражено в следующих выражениях:

$$P_{\text{ЧЕТН}} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$

$$P_{\text{НЕЧЕТН}} = d_{n-1} \oplus \dots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

где $P_{\text{ЧЕТН}}$ - бит четного паритета;

$P_{\text{НЕЧЕТН}}$ - бит нечетного паритета;

d_n - n-ый бит данных в посылке.

После разрешения, бит паритета передается между последним битом данных и первым стоп-битом.

Перед началом сеанса связи необходимо выполнить инициализацию УСАПП. Процесс инициализации обычно состоит из установки скорости связи, задания формата посылки и разрешения работы передатчика и приемника. Если используется управление

связью по прерываниям, то во время инициализации необходимо, чтобы был сброшен флаг общего разрешения прерываний (т. е. необходимо запретить все прерывания).

Если необходимо выполнить повторную инициализацию USART, например, для изменения скорости связи или формата посылки, то необходимо убедиться, что во время инициализации передача приостановлена. Флаг TXC может использоваться для проверки завершения работы передатчика, а флаг RXC - для проверки отсутствия в приемном буфере несчитанных данных. Следует обратить внимание, что при использовании флага TXC, он должен сбрасываться программно перед началом каждой передачи (перед записью в UDR).

В следующих примерах показаны функции для простой инициализации USART на Ассемблере и Си. В примерах предполагается, что используется управление связью по опросу флагов состояния (не по прерываниям) и фиксированный формат посылки. Скорость связи выступает как параметр функции. Для примера на ассемблере предполагается, что параметр скорости связи записан перед вызовом функции в регистры r17:r16.

Пример кода на Ассемблере¹⁾

```
USART_Init:
; Установка скорости связи
out UBRRH, r17
out UBRRL, r16
; Разрешение работы приемника и передатчика
ldi r16, (1<<RXEN) | (1<<TXEN)
out UCSRB,r16
; Установка формата посылки: 8 бит данных, 2стоп-бита
ldi r16, (1<<USBS) | (3<<UCSZ0)
out UCSRC,r16
ret
```

Пример кода на Си¹⁾

```
void USART_Init( unsigned int baud )
{
/* Установка скорости связи */
UBRRH = (unsigned char)(baud>>8);
UBRRL = (unsigned char)baud;
/* Разрешение работы передатчика и приемника */
UCSRB = (1<<RXEN) | (1<<TXEN);
/* Установка формата посылки: 8 бит данных, 2 стоп-бита */
UCSRC = (1<<USBS) | (3<<UCSZ0);
}
```

¹⁾ В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода, необходимо заменить инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

Более совершенные процедуры инициализации могут использовать расширенный интерфейс функций, где в качестве параметров выступают, например, формат посылки, отключение прерываний и т. д. Однако, в большинстве приложений используются фиксированные установки скорости связи и управляющих регистров, поэтому для них представленные примеры могут быть непосредственно включены в основную программу или к процедурам инициализации других модулей ввода-вывода.

3.15.5 Передача данных - Передатчик УСАПП

Работа передатчика УСАПП разрешается путем установки бита разрешения передачи (TXEN) в регистре UCSRB. После разрешения функция вывода TXD, как обычного порта заменяется на функцию выхода последовательной передачи данных. Скорость связи, режим работы и формат посылки должны быть установлены однократно перед началом какой-либо передачи. Если используется синхронная работа, то функция вывода XCK также заменяется на альтернативную - синхронизация передачи.

Передача посылок с 5...8 битами данных

Начало передачи инициируется записью передаваемых данных в буфер передатчика. ЦПУ может загрузить буфер передатчика путем записи в регистр UDR, расположенный в памяти ввода-вывода. Буферизованные данные в буфере передатчика будут перемещены в сдвиговый регистр после того, как он будет готов к отправке новой посылки. Запись в сдвиговый регистр новых данных происходит в состоянии ожидания (когда передача завершена) или сразу после завершения передачи последнего стоп-бита предыдущей посылки. Если в сдвиговый регистр записаны новые данные, то начинается передача одной посылки на скорости, определенной в регистре скорости связи, битом U2X или XCK в зависимости от выбранного режима работы.

В следующих примерах представлены простые функции передачи через УСАПП, использующие опрос флага освобождения регистра данных (UDRE). Если используется посылка с менее 8 бит данных, то старшие биты, записанные в UDR, игнорируются. Перед вызовом данной функции должна быть выполнена инициализация УСАПП. Для кода на Ассемблере предполагается, что передаваемые данные записаны в регистр R16 перед вызовом процедуры.

Пример кода на Ассемблере¹⁾

```
USART_Transmit:
; Ожидание освобождения буфера передатчика
sbis UCSRA, UDRE
rjmp USART_Transmit
; Помещение данных (r16) в буфер, отправка данных
out UDR, r16
ret
```

C Code Example¹⁾

```
void USART_Transmit( unsigned char data )
{
/* Ожидание освобождения буфера передатчика */
while ( !( UCSRA & (1<<UDRE)) );
/* Помещение данных в буфер, отправка данных */
UDR = data;
}
```

¹⁾ В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода, необходимо заменить инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

Перед загрузкой новых данных для передачи в данной функции осуществляется ожидание освобождения буфера передатчика путем опроса флага UDRE.

Отправка посылок с 9 битами данных

Если необходимо передавать 9 бит данных (`UCSZ = 7`), то 9-ый бит данных должен быть записан в бит `TXB8` регистра `UCSRB` перед тем, как младший байт будет записан в `UDR`. В следующих примерах показаны функции для передачи 9 бит данных. Для кода на Ассемблере предполагается, что отправляемые данные предварительно записаны в регистры `r17:r16`.

Пример кода на Ассемблере¹⁾

```
USART_Transmit:
; Ожидание освобождения буфера передатчика
sbis UCSRA, UDRE
rjmp USART_Transmit
; Копирование 9-го бита из r17 в TXB8
cbi UCSRB, TXB8
sbrc r17, 0
sbi UCSRB, TXB8
; Помещение мл. байта данных (r16) в буфер, отправка данных
out UDR, r16
ret
```

Пример кода на Си¹⁾

```
void USART_Transmit( unsigned int data )
{
/* Ожидание освобождения буфера передатчика */
while ( !( UCSRA & (1<<UDRE) ) );
/* Копирование 9-го бита в TXB8 */
UCSRB &= ~(1<<TXB8);
if ( data & 0x0100 ) UCSR_B |= (1<<TXB8);
/* Помещение данных в буфер, отправка данных */
UDR = data;
```

¹⁾ Данные функции записаны как функции общего назначения. Они оптимизированы под статическое содержимое `UCSRB`, т. е. когда после инициализации в регистре `UCSRB` изменяется только бит `TXB8`. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода, необходимо заменить инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

9-ый бит данных может использоваться для индикации адреса посылки в много-процессорном режиме связи или в других протоколах.

3.15.6 Флаги и прерывания передатчика

Передатчик УСАПП имеет два флага, которые индицируют его состояние: флаг освобождения регистра данных УСАПП (`UDRE`) и флаг завершения передачи (`TXC`). Оба флага могут использоваться для генерации прерываний. Флаг освобождения регистра данных (`UDRE`) индицирует готовность буфера передатчика принять новые данные. Данный бит устанавливается при освобождении передающего буфера и сбрасывается, когда буфер передатчика содержит данные для передачи, но которые еще не были переданы в сдвиговый регистр. Для совместимости с будущими микроконтроллерами в данный бит необходимо записывать ноль во время записи в регистр `UCSRA`.

Если записать логическую 1 в бит разрешения прерывания по освобождению регистра данных в регистре `UCSRB`, то прерывание по освобождению регистра данных УСАПП будет выполняться всякий раз, когда устанавливается бит `UDRE` (с учетом того,

что общие прерывания разрешены). UDRE сбрасывается при записи в UDR. Если используется управление связью по прерываниям, то в процедуре обработки прерывания по освобождению регистра данных необходимо или записать новые данные в регистр UDR для сброса флага UDRE, или выключить прерывание по освобождению регистра данных. В противном случае при выходе из процедуры обработки прерывания сразу возникнет новое прерывание.

Флаг завершения передачи (TXC) принимает единичное значение, если вся посылка в сдвиговом регистре передатчика была полностью сдвинута и в буфере передатчика отсутствуют новые данные для передачи. Флаг TXC автоматически сбрасывается при переходе на вектор обработки прерывания по завершению передачи или программно сбрасывается путем записи в него логическую 1. Флаг TXC полезно использовать при организации полу duplexной связи, где имеется необходимость перевода передающей стороны в режим приема после завершения передачи, тем самым достигая освобождения шины.

Если разрешено прерывание по завершению передачи (TXCIE=1) в регистре UCSRB, то прерывание по завершению передачи УСАПП выполняется всякий раз, когда флаг TXC принимает единичное состояние (с учетом, что активно общее разрешение прерываний). При переходе на вектор обработки прерывания по завершении передачи УСАПП нет необходимости сбрасывать флаг TXC, т. к. это происходит автоматически.

3.15.7 Генератор паритета

Генератор паритета вычисляет бит паритета для включения в состав последовательной посылки. Если бит паритета установлен (UPM1 = 1), то управляющая логика передатчика вставляет бит паритета между последним битом данных и первым стоп-битом в отправляемой посылке.

3.15.8 Отключение передатчика

Отключение передатчика после сброса TXEN наступит только тогда, когда завершится текущая и ожидаемая передача, т. е. когда в сдвиговом регистре передатчика и буфере передатчика не будет данных для передачи. После отключения передатчика изменяется альтернативное назначение вывода TXD.

3.15.9 Прием данных - Приемник УСАПП

Работа приемника УСАПП разрешается, если записать логическую 1 в бит разрешения работы приемника (RXEN) в регистре UCSRB. После разрешения работы приемника обычное назначение вывода RXD заменяется на альтернативное: вход последовательного ввода данных приемника УСАПП. Скорость связи, режим работы и формат посылки должны быть установлены однократно перед началом выполнения приема данных. Если используется синхронная работа, то вывод XCK будет использоваться для синхронизации связи.

3.15.10 Прием посылок с 5...8 битами данных

Приемник начинает прием данных только после определения действительного старт-бита. Выборка следующих за старт-битом бит данных происходит с частотой, равной скорости связи или частотой сигнала XCK, и размещается в сдвиговом регистре приемника. Второй стоп-бит приемником игнорируется. После получения первого стоп-бита, т. е. когда последовательная посылка полностью принята и находится в сдвиговом регистре приемника, содержимое сдвигового регистра перемещается в приемный буфер. Приемный буфер считывается при чтении регистра ввода-вывода UDR.

В следующих примерах приведены простые функции для организации приема данных, которые основаны на опросе состояния флага завершения приема (RXC). Если используется формат посылки с числом бит менее 8, то после считывания содержимого UDR старшие неиспользуемые разряды будут обнулены. Перед вызовом данных функций должна быть выполнена инициализация УСАПП.

Пример кода на Си¹⁾

```
unsigned int USART_Receive( void )
{
    unsigned char status, resh, resl;
    /* Ожидание окончания приема данных */
    while ( !(UCSRA & (1<<RXC)) );
    /* Опрос статусных бит и 9-го бита данных перед чтением данных из буфера */
    status = UCSRA;
    resh = UCSRB;
    resl = UDR;
    /* Если ошибка, то возврат -1 */
    if ( status & (1<<FE) | (1<<DOR) | (1<<UPE) )
        return -1;
    /* Выделение 9-го бита данных перед выходом */
    resh = (resh >> 1) & 0x01;
    return ((resh << 8) | resl);
}
```

¹⁾ В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода, необходимо заменить инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

В данных функциях считаются все регистры ввода-вывода в файл регистров перед выполнением каких-либо вычислений. Такой подход позволяет наиболее оптимально использовать заполняемость буфера, т. к. буфер становится свободным для приема новых данных, как только это станет возможным.

3.15.11 Флаг и прерывание по завершению приема

Приемник УСАПП имеет один флаг, который индицирует состояние приемника.

Флаг завершения приема (RXC) сигнализирует о наличии несчитанных данных в приемном буфере. Данный флаг равен 1, если имеются несчитанные данные, и равен 0, если буфер приемника свободен (т. е. не содержит каких-либо несчитанных данных). Если приемник отключается (RXEN = 0), то приемный буфер будет сброшен и флаг RXC примет нулевое значение.

Если установлен бит разрешения прерывания по завершению приема (RXCIE) в регистре UCSRB, то при установке флага RXC программа переходит на вектор обработки данного прерывания (при условии, что активно общее разрешение прерываний). Если используется организация связи с управлением по прерываниям, то при выполнении процедуры обработки запроса на прерывание по завершению приема необходимо считать данные из UDR, чтобы сбросить флаг RXC. В противном случае новое прерывание возникнет сразу после выхода из текущего.

3.15.12 Флаги ошибок приемника

Приемник УСАПП имеет три флага ошибок: ошибка посылки (кадра) FE, переполнение данных DOR и ошибка паритета UPE. Данные флаги входят в состав регистра UCSRA. Общим свойством данных флагов является то, что они хранятся в приемном буфере вместе с той посылкой данных, для которой они отражают состояние ошибок. С

учетом этого, необходимо следить, чтобы флаги ошибок считывались из регистра UCSRA перед чтением данных из приемного буфера (UDR), т. к. после чтения из UDR изменяется состояние буфера. Другим сходством флагов ошибок является невозможность программно повлиять на их состояние. Однако в целях совместимости с УСАПП последующих микроконтроллеров, во время записи регистра UCSRA в позициях флагов ошибок необходимо указывать нулевые значения. Ни один из флагов ошибок не может вызвать прерывание.

Флаг ошибки посылки (кадра) FE индицирует состояние первого стоп-бита сохраненной в приемном буфере посылки. Флаг FE равен 0, если стоп-бит имел корректное значение (логическая 1), и равен 1, если некорректное, т. е. 0. Данный флаг может использоваться для выявления условия разсинхронизации, обрыва связи и манипуляции над протоколом связи. Флаг FE не изменяется при установке бита USBS в регистре UCSRC, т. к. приемник игнорирует все стоп-биты за исключением первого. Для совместимости с последующими микроконтроллерами в позиции данного бита необходимо указывать 0 во время записи в регистр UCSRA.

Флаг переполнения данных (DOR) сигнализирует о потере данных из-за переполнения приемного буфера. Переполнение данных возникает, если приемный буфер заполнен (две посылки), в сдвиговом регистре ожидает считывания только что принятая посылка и обнаружен новый старт-бит. Если флаг DOR установлен, то значит одна или более последовательных посылок потеряны между последним и следующим считанными значениями из UDR. Для совместимости с будущими микроконтроллерами в позицию данного бита необходимо всегда записывать логический 0 во время записи в регистр UCSRA. Флаг DOR сбрасывается, если принятая посылка была успешно перемещена из сдвигового регистра в приемный буфер.

Флаг ошибки паритета (UPE) сигнализирует, что во время приема посылки была обнаружена ошибка паритета. Если контроль паритета отключен, то данный флаг всегда имеет нулевое значение. Для совместимости с новыми разработками микроконтроллеров в позицию данного бита необходимо всегда записывать 0 во время записи в регистр UCSRA.

3.15.13 Устройство проверки паритета

Устройство проверки паритета становится активным после установки бита режима паритета УСАПП UPM1. Тип контроля паритета: четность или нечетность задается битом UPM0. После активизации устройство проверки паритета вычисляет паритет принятых данных и сравнивает полученное значение с принятым вместе с этими данными в одной посылке битом паритета. Результат сравнения запоминается в приемном буфере вместе с принятыми данными и стоп-битом. Флаг ошибки паритета UPE может быть считан программно тогда, когда в посылке имеется ошибка паритета. Бит UPE устанавливается, если в посылке, которая может быть считана из приемного буфера, имеется ошибка паритета и во время приема этой посылки был разрешен контроль паритета (UPM1 = 1). Данный бит должен быть опрошен до считывания буфера приемника (UDR).

Отключение приемника

В отличие от передатчика отключение приемника происходит незамедлительно. При этом принимаемые данные будут потеряны. После отключения (т. е. когда RXEN = 0) приемник далее не поддерживает альтернативные настройки вывода порта RXD. Приемный буфер FIFO сбрасывается после отключения приемника, следовательно, оставшиеся в нем данные будут потеряны.

Сброс приемного буфера

Приемный буфер FIFO сбрасывается после отключения приемника, т. е. полностью освобождается от своего содержимого. Несчитанные данные будут потеряны. Если буфер необходимо очистить в процессе нормальной работы, например, при возникновении ошибок, то необходимо считывать содержимое UDR пока не очиститься флаг RXC. В следующем примере показано как очистить буфер приемника.

Пример кода на Ассемблере¹⁾

```
USART_Flush:
    sbis UCSRA, RXC
    ret
    in r16, UDR
    rjmp USART_Flush
```

Пример кода на Си¹⁾

```
void USART_Flush( void )
{
    unsigned char dummy;
    while ( UCSRA & (1<<RXC) ) dummy = UDR;
}
```

¹⁾ В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода, необходимо заменить инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".

3.15.14 Асинхронный прием данных

УСАПП содержит блоки обнаружения данных и синхронизации для управления асинхронным приемом данных. Логика обнаружения синхронизации используется для синхронизации с внутренним генератором скорости связи для обеспечения возможности ввода последовательной посылки с выводов RXD. Логика обнаружения данных осуществляет выборку и фильтрацию (ФНЧ) каждого входящего бита данных, тем самым увеличивая помехоустойчивость приемника. Рабочий диапазон асинхронного приема определяется точностью встроенного генератора скорости связи, точностью скорости входящей посылки и размером посылки (количество бит).

Асинхронный поиск синхронизации

Логика обнаружения синхронизации стабилизирует во времени работу приемника с входящей последовательной посылкой. На рисунке 3.72 иллюстрируется процесс поиска старт-бита во входящей посылке. Частота выборок в 16 раз выше скорости связи для нормального режима и 8 раз выше для режима удвоения скорости. Горизонтальные стрелки иллюстрируют возможный уход синхронизации в процессе выборки. Следует обратите внимание на более высокую разсинхронизацию во времени при использовании режима удвоения скорости ($U2X = 1$). Выборки, обозначенные номером 0, соответствуют состоянию ожидания на линии RXD (т. е. при неактивной связи).

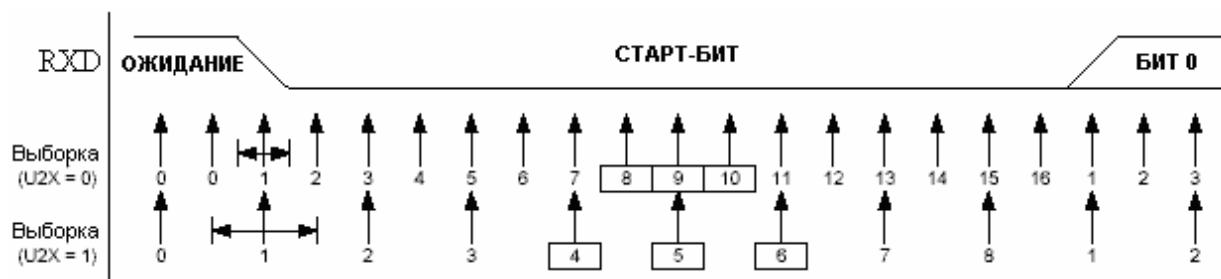


Рисунок 3.72 – Процесс поиска старт-бита во входящей посылке

Если логика обнаружения синхронизации определяет переход из высокого (состояние ожидания) к низкому (старт) состоянию на линии RXD, то инициируется последовательность действий по обнаружению старт-бита. Примем, что выборка 1 означает первую выборку с нулевым значением. Тогда по выборкам 8, 9, 10 в нормальном режиме и выборкам 4, 5, 6 в режиме удвоения скорости определяется действительность старт-бита (на рисунке эти выборки помещены в рамку). Если две или более из этих выборок имеют единичное состояние (принцип мажоритарного голосования), то старт-бит отключается как ложный, а приемник продолжит поиск следующего перехода из 1 в 0. Однако, если определен действительный старт-бит, то логика обнаружения синхронизации оказывается засинхронизированной, после чего вступит в силу логика обнаружения данных. Процесс синхронизации повторяется для каждого старт-бита.

Асинхронный поиск данных

После обнаружения старт-бита начинает работу логика обнаружения данных. Блок обнаружения данных использует цифровой автомат с 16 состояниями в нормальном режиме работы и с 8 состояниями в режиме удвоения скорости. На рисунке 3.73 показана выборка бит данных и бита четности. Для каждой выборки указан номер, который соответствует номеру состояния цифрового автомата блока обнаружения данных.

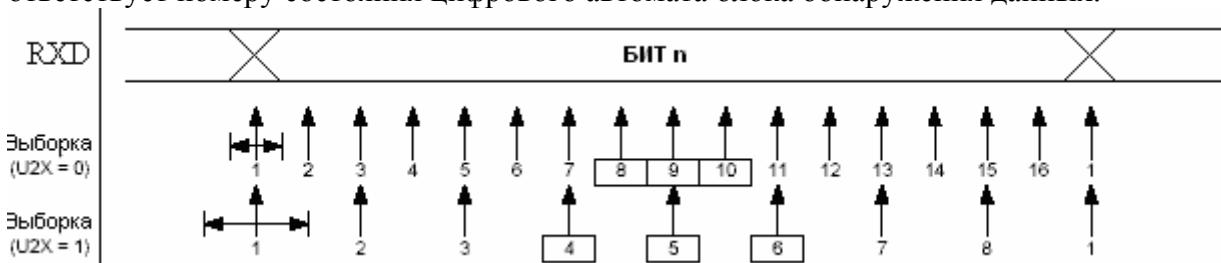


Рисунок 3.73 – Выборка бит данных и бита четности

Определение логического уровня принимаемого бита данных происходит с помощью мажоритарного голосования по трем выборкам, расположенным по центру принятого бита. Центральные выборки выделены на рисунке 3.73 путем размещения их в рамке. Процесс мажоритарного голосования состоит в следующем: если две или все три выборки имеют высокие уровни, то принятый бит фиксируется как логическая 1. Если две или три выборки имеют низкие уровни, то принятый бит фиксируется как логический 0. Процесс мажоритарного голосования, по сути, представляет собой фильтр низких частот для входящего сигнала с вывода RXD. Процесс обнаружения повторяется до полного завершения приема/посылки, в т. ч. первого стоп-бита. Следует обратить внимание, что приемник определяет только первый стоп-бит посылки, а второй игнорируется. На рисунке 3.74 отображен процесс выборки стоп-бита и начальный момент возможности обнаружения старт-бита следующей посылки.

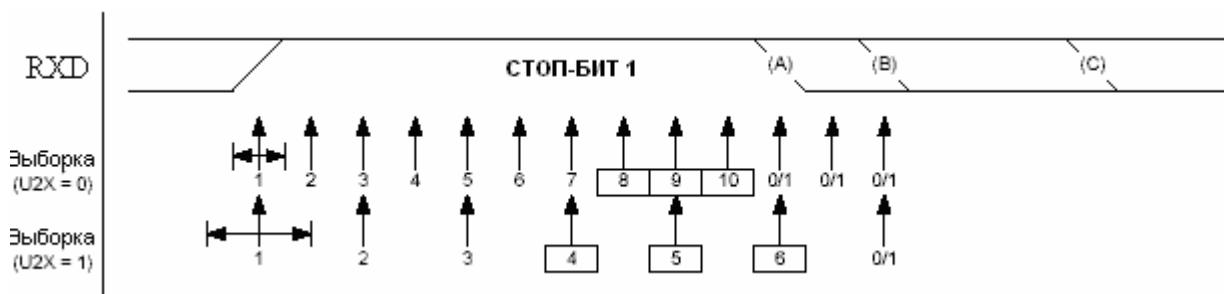


Рисунок 3.74 – Процесс выборки стоп-бита

Принцип мажоритарного голосования, рассмотренный на примере стоп-бита, аналогично распространяется и на другие биты в посылке. Если обнаруженный стоп-бит имеет нулевое значение, то устанавливается флаг ошибки посылки FE.

Новое изменение из 1 в 0 будет воспринято как стоп-бит новой посылки, если это изменение произошло после выборки последнего бита, используемого при мажоритарном голосовании. Для режима с нормальной скоростью первая выборка с низким уровнем может находиться в позиции, обозначенной А на рисунке 3.74. Для режима удвоения скорости появление низкого уровня допускается позже (точка В). Точка С соответствует полному завершению передачи стоп-бита. Использование раннего обнаружения стоп-бита влияет на рабочий диапазон приемника (допустимое расхождение частот при фиксированном формате посылки).

Рабочий диапазон асинхронной связи

Рабочий диапазон приемника зависит от расхождения между внутренне-генерируемой скоростью связи и скоростью принимаемых бит. Если передатчик отправляет посылки на более высокой или более низкой скорости или внутренне-генерируемая скорость связи приемника не соответствует основной частоте, то приемник окажется неспособным засинхронизировать посылку по отношению к старт-биту.

Следующие выражения могут использоваться для вычисления отношения скорости принимаемых данных и внутренней скорости приемника:

$$R_{MIN} = \frac{(D + 1)S}{S - 1 + D \cdot S + S_F}, R_{MAX} = \frac{(D + 2)S}{(D + 1) \cdot S + S_M},$$

где D - сумма количества передаваемых бит данных и бит паритета, D = 5...10;

S - количество выборок в секунду. S = 16/8 в режиме нормальной/удвоенной скорости;

S_F - Номер первой выборки используемой для мажоритарного голосования. SF = 8/4 в режиме нормальной/удвоенной скорости;

S_M - Номер центральной выборки используемой при мажоритарном голосовании. S_M = 9/5 в режиме нормальной/удвоенной скорости;

R_{мин} - отношение наименьшей скорости принимаемых данных к скорости приемника;

R_{макс} - отношение наибольшей скорости принимаемых данных к скорости приемника.

В таблицах 3.62 и 3.63 приведен список максимальн-допустимых погрешностей при генерации скорости приемника. Следует обратить внимание, что режим нормальной скорости устойчив к более широким изменениям скорости связи.

Таблица 3.62 - Рекомендуемая максимальная погрешность генерации скорости связи приемника в режиме нормальной скорости ($U2X = 0$)

D (количество бит данных и паритета)	R_{\min} , %	R_{\max} , %	Общая максимальная погрешность, %	Рекомендуемая максимальная погрешность приемника, %
5	93,20	106,67	+6,67/-6,8	$\pm 3,0$
6	94,12	105,79	+5,79/-5,88	$\pm 2,5$
7	94,81	105,11	+5,11/-5,19	$\pm 2,0$
8	95,36	104,58	+4,58/-4,54	$\pm 2,0$
9	95,81	104,14	+4,14/-4,19	$\pm 1,5$
10	96,17	103,78	+3,78/-3,83	$\pm 1,5$

Таблица 3.63 - Рекомендуемая максимальная погрешность генерации скорости связи приемника в режиме удвоенной скорости ($U2X = 1$)

D (количество бит данных и паритета)	R_{\min} , %	R_{\max} , %	Общая максимальная погрешность, %	Рекомендуемая максимальная погрешность приемника, %
5	94,12	105,66	+5,66/-5,88	$\pm 3,0$
6	94,92	104,92	+4,92/-5,08	$\pm 2,5$
7	95,52	104,35	+4,35/-4,48	$\pm 2,0$
8	96,00	103,90	+3,90/-4,00	$\pm 2,0$
9	96,39	103,53	+3,53/-3,61	$\pm 1,5$
10	96,70	103,23	+3,23/-3,30	$\pm 1,5$

Рекомендуемая погрешность генератора скорости связи приемника была выбрана исходя из того, что передатчик и приемник в совокупности определяют общую максимальную погрешность. Имеется два возможных источника влияния на погрешность скорости связи приемника. Системная синхронизация (XTAL) приемника всегда имеет некоторую нестабильность в зависимости от напряжения питания и температуры. При использовании кварцевого резонатора для генерации системной синхронизации как правило не возникает проблем, но при использовании керамических резонаторов частота синхронизации может изменяться более чем на 2 % в зависимости характеристик выбранного резонатора. Второй источник влияния на погрешность является более управляемым. Требуемую скорость связи не всегда удается получить путем деления частоты синхронизации на целое число. В этом случае необходимо выбрать такое значение UBRR, которое обеспечивает минимально возможную погрешность результирующей частоты.

3.15.15 Многопроцессорный режим связи

Установка бита многопроцессорного режима связи MPCM в регистре UCSRA активизирует функцию фильтрации входящих посылок приемником УСАПП. Посылки, которые не содержат информации об адресе, игнорируются и не помещаются в приемный буфер. Это позволяет существенно уменьшить количество входящих посылок, подлежащих обработке ЦПУ в многопроцессорных системах, связь между процессорами в которых организована через одну последовательную шину. Значение бита MPCM не оказывает никакого влияния на работу передатчика, но при этом передатчик должен быть использован иначе, если используется режим многопроцессорной связи.

Если приемник настраивается на прием посылок с 5...8 битами данных, то первый стоп-бит позволяет отличить назначение принятых данных: адрес или данные. Если приемник настроен на прием 9 бит данных, то значение 9-го бита (RXB8) используется для

идентификации адреса или данных. Если идентификатор типа посылки (первый стоп-бит или 9-ый бит данных) равен 1, то в посылке содержится адрес. В противном случае в посылке переданы данные.

Режим многопроцессорной связи позволяет нескольким подчиненным микроконтроллерам принимать данные от одного ведущего. При этом подчиненные микроконтроллеры по первой адресной посылке определяют к какому микроконтроллеру адресуется ведущий. Если один из подчиненных микроконтроллеров обнаруживает свой адрес, то следующие посылки данных он будет принимать в нормальном режиме, а остальные подчиненные микроконтроллеры эти данные игнорируют до тех пор, пока не будет обнаружена следующая адресная посылка.

3.15.16 Использование МРСМ

Если микроконтроллер действует как ведущий, то он может использовать 9-битный формат данных в посылке ($UCSZ = 7$). 9-ый бит данных ($TXB8$) устанавливается при передаче адресной посылки ($TXB8 = 1$) и сбрасывается при передаче посылки данных ($TXB = 0$). В этом случае подчиненные микроконтроллеры также должны устанавливать 9-битный формат.

Для обмена данными в многопроцессорном режиме связи необходимо использовать процедуры:

1 Все подчиненные микроконтроллеры переводятся в многопроцессорный режим связи ($MPCM = 1$ в $UCSRA$).

2 Ведущий МК отправляет адресную посылку, а все подчиненные принимают и считывают эту посылку. В подчиненных МК флаг RXC в регистре $UCSRA$ устанавливается как обычно.

3 Каждый подчиненный МК считывает регистр UDR и определяет к кому адресуется ведущий МК. Адресуемый МК должен очистить бит $MPCM$ в $UCSRA$, в противном случае он ожидает следующего адресного байта и сохраняет установки $MPCM$.

4 Адресуемый МК принимает все данные до следующей адресной посылки. Другие подчиненные МК, у которых бит $MPCM$ остался установленным, будут игнорировать посылки данных.

5 После приема адресуемым МК последней посылки данных устанавливается бит $MPCM$ и ожидается прием новой адресной посылки от ведущего МК. Далее процесс повторяется с пункта 2.

6 Использование 5...8-разрядных форматов данных возможно, но не удобно, т. к. приемник должен переключаться между n и $n+1$ форматами посылки. Это делает затруднительной полнодуплексную связь, т. к. передатчик и приемник используют общие установки формата. При использовании 5...8-разрядных данных в посылке передатчик должен использовать два стоп-бита, т. к. первый стоп-бит будет задействован для индикации типа посылки.

Не следует использовать инструкции "чтение-модификация-запись" (SBI и CBI) для установки или сброса бита $MPCM$. Бит $MPCM$ находится в одной ячейке с флагом TXC , поэтому последний может быть случайно сброшен при выполнении инструкций SBI или CBI .

Описание регистров УСАПП

Регистр данных УСАПП - $UDRn$

Разряд	7	6	5	4	3	2	1	0	
	RXBn[7:0]								
	TXBn[7:0]								
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Буферные регистры данных передатчика и приемника УСАППn расположены по одному и тому же адресу в области ввода-вывода, обозначенной как регистр данных УСАППn или $UDRn$. Если выполнять запись по адресу регистра $UDRn$, то записываемые данные помещаются в буферный регистр данных передатчика $TXBn$. По аналогии, при чтении регистра $UDRn$ извлекается содержимое буферного регистра данных приемника $RXBn$.

При использовании 5-, 6- или 7-битных форматов данных передатчик игнорирует, а приемник устанавливает нулевые значения неиспользуемых разрядов.

Запись в буфер передатчика можно выполнять, если установлен флаг UDREn в регистре UCSRA_n. Данные, записанные в UDR_n при сброшенном флаге UDREn, будут игнорированы передатчиком УСАППn. Если выполнена запись в приемный буфер и при этом работа передатчика была разрешена, то после освобождения сдвигового регистра передатчик загрузит в него значение из буферного регистра. После этого выполняется передача данных на выводе TxD_n. Приемный буфер организован как двухуровневый буфер FIFO (первый пришел - последний вышел). Буфер FIFO изменяет свое состояние, если выполнено чтение из приемного буфера. Вследствие такой организации буфера необходимо следить, чтобы по данному адресу не использовались инструкции "чтение-модификация-запись" (SBI и CBI). Также нужно быть внимательным при использовании инструкций тестирования бита (SBIC и SBIS), т. к. их выполнение может также изменить состояние буфера FIFO.

Регистр А управления и статуса УСАПП - UCSRnA

Разряд	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
	Чт/Зап								
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7: RXCn - Флаг завершения приема УСАПП

Данный флаг устанавливается, если в приемном буфере содержатся несчитанные данные и сбрасывается, когда приемный буфер свободен (т. е. не содержит несчитанных данных). Если приемник отключается, то приемный буфер сбрасывается и, следовательно, флаг RXCn принимает нулевое значение. Флаг RXCn может использоваться для генерации прерывания по завершению приема (смотрите описание бита RXCIEn).

Разряд 6: TXCn - Флаг завершения передачи УСАПП

Данный флаг устанавливается, если вся посылка из сдвигового регистра передатчика полностью передана и в передающем буфере UDR_n нет новых данных для передачи. Флаг TXCn автоматически сбрасывается при переходе на вектор прерывания по завершению передачи или сбрасывается программно путем записи логической 1 в позицию данного бита. Флаг TXCn может служить источником для генерации прерывания по завершению передачи (смотрите также описание бита TXCIEn).

Разряд 5: UDREn - Флаг освобождения регистра данных УСАПП

Флаг UDREn индицирует о готовности приемного буфера UDR_n к приему новых данных. Если UDREn=1, то буфер свободен и, следовательно, готов к записи. Флаг UDREn может служить источником для генерации прерывания по освобождению регистра данных (смотрите описание бит UDRIEn). UDREn устанавливается после сброса, индицируя о готовности передатчика.

Разряд 4: FEn - Ошибка посылки

Данный бит устанавливается, если при приеме посылки, находящейся на выходе из приемного буфера, была определена ошибка в структуре посылки. Под ошибкой структуры в данном случае понимается нулевое значение первого стоп-бита в этой посылке. Значение данного бита действительно до чтения содержимого приемного буфера

(UDRn). Флаг FEn принимает нулевое значение, если принятый стоп-бит имел правильное единичное значение. При записи в регистр UCSRnA в позиции данного бита необходимо указывать логический 0.

Разряд 3: DORn - Флаг переполнения данных

Данный бит устанавливается, если выявлено условие переполнения. Переполнение данных возникает, если заполнен приемный буфер (две посылки), новая посылка полностью принята в приемный сдвиговый регистр, а также обнаружен новый старт-бит. Значение данного бита действительно до чтения содержимого приемного буфера (UDRn). При записи в регистр UCSRnA в позиции данного бита необходимо указывать логический 0.

Разряд 2: UPEn - Ошибка паритета

Данный бит устанавливается, если следующая посылка в приемном буфере характеризуется ошибкой паритета, если во время приема этой посылки был разрешен контроль паритета ($UPMn1 = 1$). Данный бит имеет действительное значение до чтения приемного буфера (UDRn). При записи в регистр UCSRnA в позиции данного бита необходимо указывать логический 0.

Разряд 1: U2Xn - Удвоение скорости связи УСАПП

Данный бит оказывает влияние только в асинхронном режиме связи. В синхронном режиме в данный бит необходимо записать логический 0. Запись в данный бит логический 1 уменьшает в два раза значение коэффициента деления скорости связи с 16 до 8, тем самым удваивая скорость передачи данных в асинхронном режиме.

Разряд 0: MPCMn - Режим многопроцессорной связи

Данный бит разрешает режим многопроцессорной связи. Если в бит MPCMn записать логическую 1, то все входящие посылки принимаемые приемником УСАПП будут игнорироваться, если они не содержат адресной информации. Установка бита MPCMn не влияет на работу передатчика.

Регистр В управления и статуса УСАППn - UCSRnB

Разряд	7	6	5	4	3	2	1	0	
	RXCIE n	TXCIE n	UDRIE n	RXENn	TXENn	UCSZn 2	RXB8n	TXB8n	UCSRn B
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7: RXCIEn - Разрешение прерывания по завершению приема

Запись в данный бит логическую 1 разрешает прерывание по флагу RXCn. Прерывание по завершению приема УСАППn генерируется, если RXCIEn=1, флаг общего разрешения прерываний I=1 (в регистре SREG), а также установлен бит RXCn в регистре UCSRnA.

Разряд 6: TXCIE - Разрешение прерывания по завершению передачи

Запись в данный бит логическую 1 разрешает прерывание по флагу TXCn. Прерывание по завершению передачи УСАППн генерируется, если TXCIEn=1, флаг общего разрешения прерываний I=1 (в регистре SREG), а также установлен бит TXCn в регистре UCSRnA.

Разряд 5: UDRIEn - Разрешение прерывания по освобождению регистра данных УСАПП

Установка данного флага разрешает прерывание по флагу UDREn. Прерывание по освобождению регистра данных генерируется, если бит UDRIEn=1, флаг общего разрешения прерываний I=1 (в регистре SREG) и установлен бит UDREn в регистре UCSRnA.

Разряд 4: RXENn - Разрешение работы приемника

Запись в данный бит логическую 1 приводит к разрешению работы приемника УСАППн. При этом, приемник формирует отключающий сигнал, который разрешает альтернативную функцию вывода RXDn. Отключение приемника приводит к сбросу приемного буфера, теряя при этом значения флагов FEn, DORn и UPEn.

Разряд 3: TXENn - Разрешение работы передатчика

Запись в данный бит логическую 1 разрешает работу передатчика УСАППн. После этого передатчик генерирует отключающий сигнал, который активизирует альтернативную функцию вывода TXDn. Отключение передатчика (запись логического 0 в TXENn) вступит в силу только по завершении генерации посылки, т. е. когда освободятся и сдвиговый регистр и буфер передатчика. После отключения вывод TXDn возвращается к выполнению функции обычной линии ввода-вывода.

Разряд 2: UCSZn - Формат данных

Бит UCSZn2 вместе с битами UCSZn1:0 в регистре UCSRnC задают количество бит данных в посылке как для приемника, так и для передатчика.

Разряд 1: RXB8n - Значение 8-ого разряда принятых данных

RXB8n содержит значение 9-го бита принятой посылки с 9-битным форматом. Данный бит необходимо считать прежде, чем будут считаны младшие 8 бит из регистра UDRn.

Разряд 0: TXB8n - 8-ой разряд передаваемых данных

TXB8n содержит значение 9-ого бита данных для передачи посылки с 9-битным форматом. Данный бит необходимо записать перед тем, как будут записаны младшие разряды данных в UDRn.

Регистр С управления и статуса УСАПП - UCSRnC

Разряд	7	6	5	4	3	2	1	0	
	-	IM- SELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
На- чаль- ное значе- ние	0	0	0	0	0	0	0	0	

Разряд 7 - Резервный бит

Данный бит зарезервирован для будущего использования. Однако для совместимости с будущими микроконтроллерами при записи в регистр UCSRnC в позицию данного бита необходимо записывать логический 0.

Разряд 6: UMSELn - Выбор режима УСАПП

Данный бит позволяет переключаться между синхронным и асинхронными режимами последовательной связи.

Таблица 3.64 - Установки бита UMSELn

UMSELn	Режим связи
0	Асинхронный
1	Синхронный

Разряды 5, 4: UPMn1:0 - Режим паритета

Данные биты разрешают и устанавливают тип генерируемого и контролируемого паритета. После разрешения паритета передатчик автоматически генерирует и передает бит паритета в каждой посылке. Приемник генерирует бит паритета для принятых данных и сравнивает его со значением принятого в этой посылке бита паритета, а по результату сравнения устанавливает флаг ошибки паритета UPEn в регистре UCSRnA.

Таблица 3.65 - Установки бит UPMn

UPMn1	UPMn0	Режим паритета
0	0	Отключен
0	1	Резерв
1	0	Четность
1	1	Нечетность

Разряд 3: USBSn - Выбор числа стоп-бит

Данный бит определяет сколько стоповых бит вставляет передатчик при генерации посылки. Приемник игнорирует эту настройку.

Таблица 3.66 - Установки бита USBSn

USBSn	Число стоп-бит
0	1 бит
1	2 биты

Разряды 2, 1: UCSZn1:0 - Формат данных

Биты UCSZn1:0 вместе с UCSZn2 в регистре UCSRnB задают количество бит данных в посылке как для приемника, так и для передатчика.

Таблица 3.67 - Установки бит UCSZn

UCSZn2	UCSZn1	UCSZn0	Формат данных
0	0	0	5 бит
0	0	1	6 бит
0	1	0	7 бит
0	1	1	8 бит
1	0	0	Резерв
1	0	1	Резерв
1	1	0	Резерв
1	1	1	9 бит

Разряд 0: UCPOLn - Полярность синхронизации

Данный бит используется только в синхронном режиме. Если используется асинхронный режим, то в данный бит необходимо записать логический 0. В синхронном режиме бит UCPOLn определяет соотношение между выборкой входящих данных и обновлением передаваемых данных и сигналом тактирования синхронной связи (XCKn).

Таблица 3.68 - Установки бит UCPOLn

UCPOLn	Изменение передаваемых данных на выходе TXDn	Выборка принимаемых данных на входе RXDn
0	Нарастающий фронт XCKn	Падающий фронт XCKn
1	Падающий фронт XCKn	Нарастающий фронт XCKn

Регистры скорости связи УСАПП - UBRRnL и UBRRnH

Разряд	7	6	5	4	3	2	1	0	
	-	-	-	-	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	Чт.	Чт.	Чт.	Чт.	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Примечание - Регистр UBRRnH не доступен в режиме совместимости с mega103.

Разряды 15...12 - Зарезервированные разряды

Данные разряды зарезервированы для будущего использования. Для совместимости с последующими разработками необходимо записать лог. 0 в эти разряды во время записи в регистр UBRRnH.

Разряды 11...0: UBRRn11:0 - Регистр скорости связи УСАПП

UBRR - 12-разрядный регистр, который задает значение скорости связи УСАПП. Регистр UBRRnH содержит 4 старших разряда, а UBRRnL 8 младших разрядов значения скорости УСАППн. Если во время передачи или приема изменить скорость связи, то сеанс связи будет нарушен. Запись в регистр UBRRnL инициирует обновление предделителя скорости связи.

В таблице 3.69 приведены примеры установок UBRR для генерации стандартных скоростей связи при типичных тактовых частотах микроконтроллера. Значения UBRR, которые дают результирующую скорость связи, отличающуюся не более, чем на 0,5 % от искомого значения. Более высокие погрешности также приемлемы, но приемник будет обладать меньшей помехоустойчивостью, особенно при передаче длинных посылок. Значения погрешностей вычислены по следующему выражению:

$$\delta = \left(\frac{f_{\text{ген}}}{16 \cdot (UBRR + 1) \cdot f_{\text{связи}}} - 1 \right) \cdot 100, \% .$$

где $f_{\text{ген}}$ - частота тактового генератора, Гц;

$f_{\text{связи}}$ - скорость связи, бит/с;

UBRR - значение регистра UBRR.

Таблица 3.69 - Примеры установок UBRR для типичных частот тактового генератора

Скорость связи, бит/с	$f_{\text{ген}} = 1,0000 \text{ МГц}$				$f_{\text{ген}} = 1,8432 \text{ МГц}$				$f_{\text{ген}} = 2,0000 \text{ МГц}$			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	$\delta, \%$	UBRR	$\delta, \%$	UBRR	$\delta, \%$	UBRR	$\delta, \%$	UBRR	$\delta, \%$	UBRR	$\delta, \%$
2400	25	0,2	51	0,2	47	0,0	95	0,0	51	0,2	103	0,2
4800	12	0,2	25	0,2	23	0,0	47	0,0	25	0,2	51	0,2
9600	6	-7,0	12	0,2	11	0,0	23	0,0	12	0,2	25	2,1
14,4K	3	8,5	8	-3,5	7	0,0	15	0,0	8	-3,5	16	0,2
19,2K	2	8,5	6	-7,0	5	0,0	11	0,0	6	-7,0	12	-
28,8K	1	-	3	8,5	3	0,0	7	0,0	3	8,5	8	3,5
38,4K	1	18,6	2	8,5	2	0,0	5	0,0	2	8,5	6	-7,0
valign="top" 57,6K	0	8,5	1	8,5	1	0,0	3	0,0	1	8,5	3	8,5
76,8K	-	-	1	-	1	-	2	0,0	1	-	2	8,5
115,2K	-	-	0	18,6	0	25,0	1	0,0	0	18,6	1	8,5
230,4K	-	-	-	8,5	-	0,0	0	0,0	-	-	-	-
250K	-	-	-	-	-	-	-	-	-	-	0	0,0
Макс. ⁽¹⁾	62,5K бит/с		125K бит/с		115,2K бит/с		230,4K бит/с		125K бит/с		250K бит/с	

⁽¹⁾ UBRR = 0, погрешность = 0,0 %.

Таблица 3.70 - Примеры установок UBRR для типичных частот тактового генератора

Скорость связи, бит/с	f _{ген} = 3,6864 МГц				f _{ген} = 4,0000 МГц				f _{ген} = 7,3728 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %
2400	95	0,0	191	0,0	103	0,2	207	0,2	191	0,0	383	0,0
4800	47	0,0	95	0,0	51	0,2	103	0,2	95	0,0	191	0,0
9600	23	0,0	47	0,0	25	0,2	51	0,2	47	0,0	95	0,0
14,4К	15	0,0	31	0,0	16	2,1	34	-	31	0,0	63	0,0
19,2К	11	0,0	23	0,0	12	0,2	25	0,8	23	0,0	47	0,0
28,8К	7	0,0	15	0,0	8	-	16	0,2	15	0,0	31	0,0
38,4К	5	0,0	11	0,0	6	3,5	12	2,1	11	0,0	23	0,0
57,6К	3	0,0	7	0,0	3	-	8	0,2	7	0,0	15	0,0
76,8К	2	0,0	5	0,0	2	7,0	6	-	5	0,0	11	0,0
115,2К	1	0,0	3	0,0	1	8,5	3	3,5	3	0,0	7	0,0
230,4К	0	0,0	1	0,0	0	8,5	1	-	1	0,0	3	0,0
250К	0	-	1	-	0	8,5	1	7,0	1	-	3	-
0,5М	-	7,8	0	7,8	-	8,5	0	8,5	0	7,8	1	7,8
1М	-	-	-	-	0,0	-	8,5	-	7,8	-	0	-
		-		7,8	-	-	0,0	-	7,8	-	7,8	-
		-		-	-	-	0,0	-	7,8	-	7,8	-
Макс. ⁽¹⁾	230,4К бит/с		460,8К бит/с		250К бит/с		0,5М бит/с		460,8К бит/с		921,6К бит/с	

⁽¹⁾ UBRR = 0, погрешность = 0,0 %.

Таблица 3.71 - Примеры установок UBRR для типичных частот тактового генератора

Скорость связи, бит/с	f _{ген} = 8,0000 МГц				f _{ген} = 11,592 МГц				f _{ген} = 14,456 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %
2400	207	0,2	416	-	287	0,0	575	0,0	383	0,0	767	0,0
4800	103	0,2	207	0,1	143	0,0	287	0,0	191	0,0	383	0,0
9600	51	0,2	103	0,2	71	0,0	143	0,0	95	0,0	191	0,0
14,4К	34	-	68	0,2	47	0,0	95	0,0	63	0,0	127	0,0
19,2К	25	0,8	51	0,6	35	0,0	71	0,0	47	0,0	95	0,0
28,8К	16	0,2	34	0,2	23	0,0	47	0,0	31	0,0	63	0,0
38,4К	12	2,1	25	-	17	0,0	35	0,0	23	0,0	47	0,0
57,6К	8	0,2	16	0,8	11	0,0	23	0,0	15	0,0	31	0,0
76,8К	6	-	12	0,2	8	0,0	17	0,0	11	0,0	23	0,0
115,2К	3	3,5	8	2,1	5	0,0	11	0,0	7	0,0	15	0,0
230,4К	1	-	3	0,2	2	0,0	5	0,0	3	0,0	7	0,0
250К	1	7,0	3	-	2	-	5	-	3	-	6	5,3
0,5М	0	8,5	1	3,5	-	7,8	2	7,8	1	7,8	3	-
1М	-	8,5	0	8,5	-	-	-	-	0	-	1	7,8
		0,0		0,0		-		7,8		7,8	-	7,8
		0,0		0,0		-		-		-		7,8
Макс. ⁽¹⁾	0,5 Мбит/с		1 Мбит/с		691,2К бит/с		1,3824М бит/с		921,6К бит/с		1,8432 Мбит/с	

⁽¹⁾ UBRR = 0, погрешность = 0,0 %.

Таблица 3.72 - Примеры установок UBRR для типичных частот тактового генератора

Скорость связи, бит/с	f _{гех} = 16,0000 МГц				f _{гех} = 18,4320 МГц				f _{гех} = 20,0000 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %
2400	416	-	832	0,0	479	0,0	959	0,0	520	0,0	1041	0,0
4800	207	0,1	416	-	239	0,0	479	0,0	259	0,2	520	0,0
9600	103	0,2	207	0,1	119	0,0	239	0,0	129	0,2	259	0,2
14,4К	68	0,2	138	0,2	79	0,0	159	0,0	86	-	173	-
19,2К	51	0,6	103	-	59	0,0	119	0,0	64	0,2	129	0,2
28,8К	34	0,2	68	0,1	39	0,0	79	0,0	42	0,2	86	0,2
38,4К	25	-	51	0,2	29	0,0	59	0,0	32	0,9	64	-
57,6К	16	0,8	34	0,6	19	0,0	39	0,0	21	-	42	0,2
76,8К	12	0,2	25	0,2	14	0,0	29	0,0	15	1,4	32	0,2
115,2К	8	2,1	16	-	9	0,0	19	0,0	10	-	21	0,9
230,4К	3	0,2	8	0,8	4	0,0	9	0,0	4	1,4	10	-
250К	3	-	7	0,2	4	-	8	2,4	4	1,7	9	1,4
		3,5		2,1		7,8				-		-
		8,5		-						1,4		1,4
		0,0		3,5						8,5		-
				0,0						0,0		1,4
												0,0
0,5М	1	0,0	3	0,0	-	-	4	-	-	-	4	0,0
1М	0	0,0	1	0,0	-	-	-	7,8	-	-	-	-
Макс. ⁽¹⁾	1М бит/с		2М бит/с		1,152М бит/с		2,304М бит/с		1,25М бит/с		2,5М бит/с	

⁽¹⁾ UBRR = 0, погрешность = 0,0 %.

3.16 Двухпроводной последовательный интерфейс TWI

Отличительные особенности:

- гибкий, простой, при этом эффективный последовательный коммуникационный интерфейс, требующий только две линии связи;
- поддержка как ведущей, так и подчиненной работы;
- возможность работы, как приемника, так и как передатчика;
- 7-разрядное адресное пространство позволяет подключить к шине до 128 подчиненных устройств;
- поддержка многомастерного арбитрирования;
- скорость передачи данных до 400 кГц;
- выходы драйверов с ограниченной скоростью изменения сигналов;
- схема шумоподавления повышает стойкость к выбросам на линиях шины;
- программируемый адрес для подчиненного режима с поддержкой общего вызова;
- пробуждение микроконтроллера из режима сна при определении заданного адреса на шине.

3.16.1 Определение шины TWI

Двухпроводной последовательный интерфейс TWI идеально подходит для типичных применений микроконтроллера. Протокол TWI позволяет проектировщику системы внешне связать до 128 различных устройств через одну двухпроводную двунаправленную шину, где одна линия - линия синхронизации SCL, а вторая - линия данных SDA. В качестве внешних аппаратных компонентов, которые требуются для реализации шины, необходимы только подтягивающий к плюсу питания резистор на каждой линии шины. Все устройства, которые подключены к шине, имеют свой индивидуальный адрес, а механизм определения содержимого шины поддерживается протоколом TWI (см. рисунок 3.74).

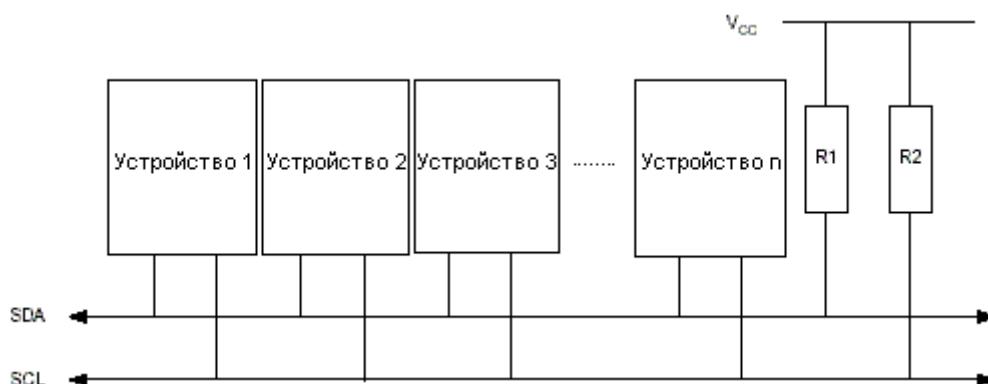


Рисунок 3.74 – Коммутация устройств на шинах SDA и SCL

3.16.2 Терминология TWI

Таблица 3.73 - Терминология TWI

Термин	Описание
Ведущий	Устройство, которое инициирует и прекращает сеанс связи. На стороне ведущего также генерируется сигнал синхронизации SCL.
Подчиненный	Устройство, которое адресуется ведущим устройством.
Передатчик	Устройство, размещающее данные на шине.
Приемник	Устройство,читывающее данные с шины.

Как показано на рисунке 3.74, обе линии шины подключены к положительной шине питания через подтягивающие резисторы. Среди всех совместимых с TWI устройств в качестве драйверов шины используются транзистор или с открытым стоком, или с открытым коллектором. Этим реализована функция монтажного И, которая очень важна для двунаправленной работы интерфейса. Низкий логический уровень на линии шины TWI генерируется, если одно или более из TWI-устройств выводит логический 0. Высокий уровень на линии присутствует, если все TWI-устройства перешли в третье высокоимпедансное состояние, позволяя подтягивающим резисторам задать уровень логической 1. Следует обратить внимание, что при подключении к шине TWI нескольких AVR-микроконтроллеров для работы шины должны быть питаны все из этих микроконтроллеров.

Количество устройств, которое может быть подключено к одной шине, ограничивается предельно-допустимой емкостью шины (400 пФ) и 7-разрядным адресным пространством. Поддерживаются два различных набора технических требований, где один набор для шин со скоростью передачи данных ниже 100 кГц и один действителен для скоростей выше 400 кГц.

3.16.3 Формат посылки и передаваемых данных

Передаваемые биты

Каждый передаваемый бит данных по шине TWI сопровождается импульсом на линии синхронизации. Уровень данных должен быть стабильным, когда на линии синхронизации присутствует логическая 1. Исключением для этого правила является генерация условий СТАРТА и ОСТАНОВа сеанса связи.

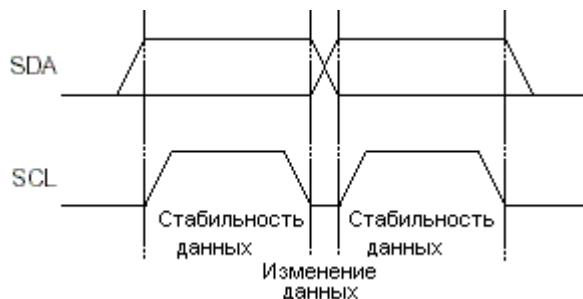


Рисунок 3.75 - Действительность данных

Условия СТАРТА и ОСТАНОВа

Ведущее устройство инициирует и заканчивает передачу данных. Передача инициируется, когда ведущий формирует условие СТАРТА на шине, и прекращается, когда ведущий формирует на шине условие ОСТАНОВа. Между условиями СТАРТА и ОСТАНОВа шина считается занятой и в этом случае никакой другой мастер не может осуществлять управляющие воздействия на шине. Существуют особые случаи, когда новое условие СТАРТА возникает между условиями СТАРТА и ОСТАНОВа. Данного случай имеется как условие "Повторного старта" и используется при необходимости инициировать мастером новый сеанс связи, не теряя при этом управление шиной. После "Повторного старта" шина считается занятой до следующего ОСТАНОВа. Это идентично поведению после СТАРТА, следовательно, при описании ссылка на условие СТАРТА распространяется и на "Повторный старта", если, конечно же, нет специального примечания. Как показано ниже, условия СТАРТА и ОСТАНОВа являются изменением логического уровня на линии SDA, когда на линии SCL присутствует логическая 1.

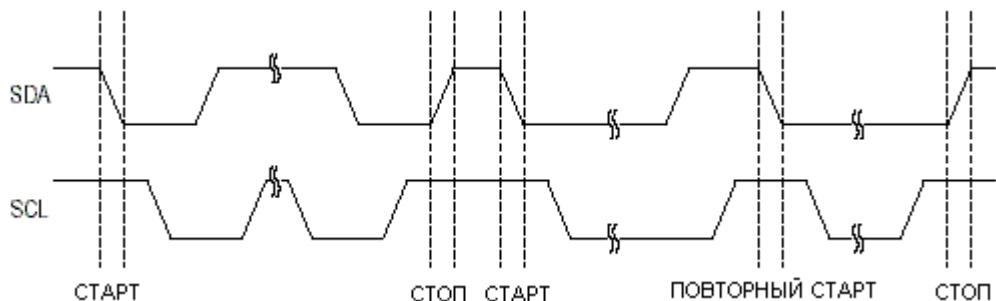


Рисунок 3.76 - Условия СТАРТА, ПОВТОРНОГО СТАРТА и ОСТАНОВа

3.16.4 Формат адресного пакета

Все передаваемые адресные пакеты по шине TWI состоят из 9 бит, в т. ч. 7 бит адреса, один бит управления для задания типа операции ЧТЕНИЕ/ЗАПИСЬ и один бит подтверждения. Если бит ЧТЕНИЕ/ЗАПИСЬ = 1, то будет выполнена операция чтения, иначе - запись. Если подчиненный распознает, что к нему происходит адресация, то он должен сформировать низкий уровень на линии SDA на 9-ом цикле SCL (формирование бита подтверждения). Если адресуемое подчиненное устройство занято или по каким-либо другим причинам не может обслужить ведущее устройство, то на линии SDA необходимо оставить высокий уровень во время цикла подтверждения. Ведущий после этого может передать условие ОСТАНОВА или "Повторного старта" для инициации новой передачи. Адресный пакет, состоящий из адреса подчиненного устройства и бита ЧТЕНИЕ или ЗАПИСЬ, обозначим как ПОДЧИН_АДР+ЧТЕНИЕ или ПОДЧИН_АДР+ЗАПИСЬ, соответственно.

Старший разряд адресного байта передается первым. Нет никаких ограничений на выбор адреса подчиненного устройства, за исключением адреса 0000 000, который зарезервирован для общего вызова.

При определении общего вызова все подчиненные устройства должны ответить низким уровнем на линии SDA во время цикла подтверждения (ACK). Общий вызов необходимо использовать, если одно и тоже сообщение необходимо передать от ведущего к нескольким подчиненным устройствам. Если вслед за битом ЗАПИСИ передан адрес общего вызова, то все подчиненные устройства устанавливают низкий уровень на линии SDA для подтверждения общего вызова во время цикла подтверждения. Следующие пакеты данных будут приниматься всеми подчиненными устройствами, которые подтвердили общий вызов. Следует обратить внимание, что передача адреса общего вызова вслед за битом ЧТЕНИЕ бессмысленна, т. к. одновременное чтение нескольких подчиненных устройств одним ведущим не возможно.

Все адреса с форматом 1111xx необходимо зарезервировать для будущего использования.

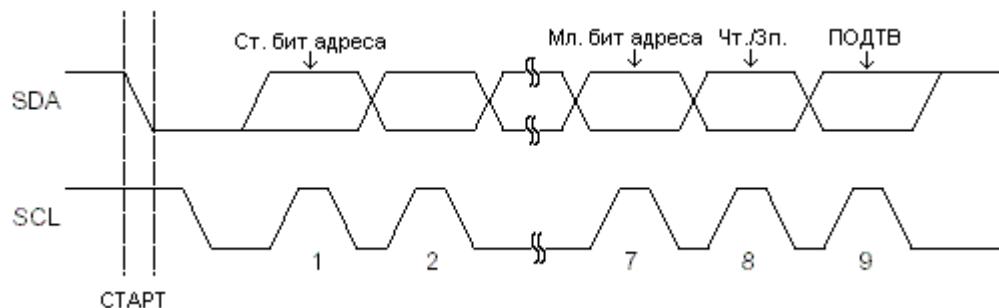


Рисунок 3.77 - Формат адресного пакета

3.16.5 Формат пакета данных

Все пакеты данных, передаваемые по шине TWI, состоят из 9 бит, в т. ч. 1 байт данных и бит подтверждения. Во время передачи данных ведущее устройство генерирует синхронизацию, а также условия СТАРТА и ОСТАНОВА, при этом на приемник возлагается подтверждение приема. Подтверждение (ПОДТВ) сигнализируется приемником выводом низкого уровня на линию SDA во время 9-го такта сигнала SCL. Если приемник оставляет линию SDA в высоком состоянии, то это сигнализирует о том, что подтверждения не было (НЕТ ПОДТВ). После получения приемником последнего байта или если по каким-либо причинам нет возможности далее принимать данные, он должен информировать передатчик отправкой бита НЕТ ПОДТВ (нет подтверждения) после последнего байта. Старший бит данных передается первым.

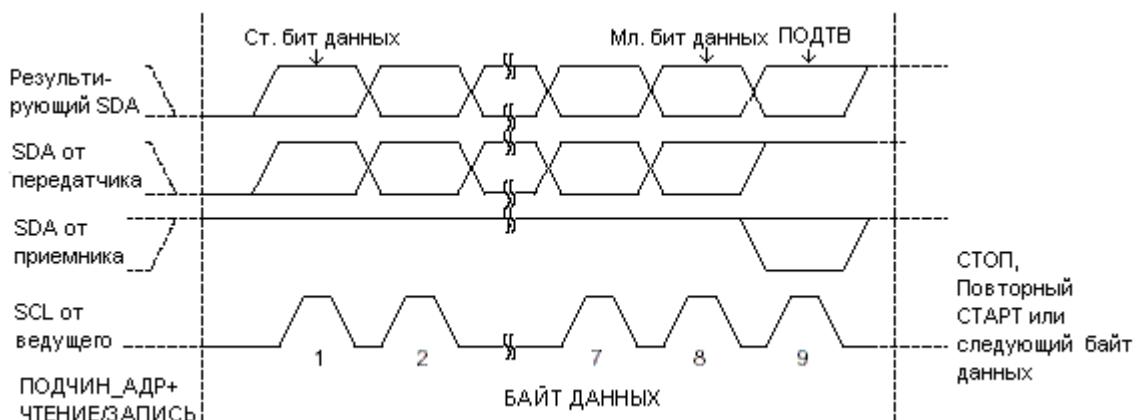


Рисунок 3.78 - Формат пакета данных

3.16.6 Сочетание пакетов адреса и данных во время сеанса связи

Сеанс связи обычно состоит из условия СТАРТА, ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ, одного или более пакетов данных и условия ОСТАНОВА. Передача пустого сообщения, которое состоит из условия СТАРТА, переданного вслед за условием ОСТАНОВА, является недопустимым. Следует братить внимание, что монтажное "И" на линии SCL может использоваться для реализации подтверждения связи между ведущим и подчиненным. Подчиненный может продлить низкое состояние на линии SCL путем установки логического 0 на выводе SCL. Данный способ полезно использовать, если установленная скорость связи мастером является повышенной по отношению к подчиненному или если подчиненному требуется дополнительное время на обработку между приемами данных. Почкиненный, продлеваящий низкое состояние на линии SCL, не будет оказывать влияние на длительность высокого состояния SCL, которая определяется мастером. Как следствие, подчиненный может снизить скорость передачи данных, продлевая рабочий цикл SCL.

На рисунке 3.79 показана типичная передача данных. Следует обратить внимание, что несколько байт данных могут быть переданы между условиями ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ и СТОП в зависимости от программного протокола, реализованного в прикладной программе.

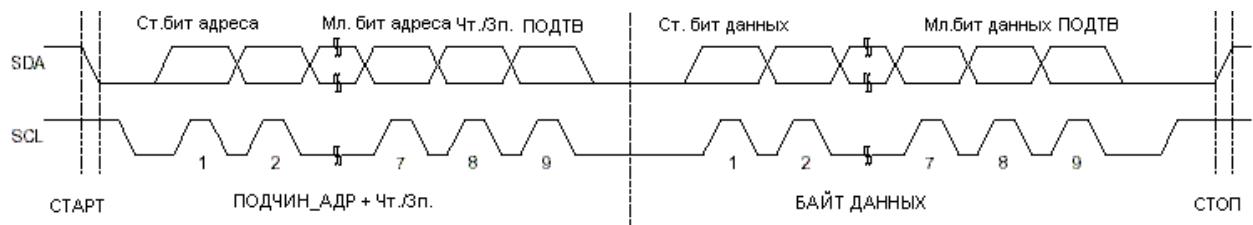


Рисунок 3.79 - Типичная передача данных

3.16.7 Системы многомастерных шин, арбитраж и синхронизация

Протокол TWI допускает размещение нескольких ведущих устройств на однойшине. Чтобы гарантировать нормальность процесса передачи, даже если два и более ведущих устройств инициируют передачу в одно и то же время, следует предпринять меры. Две проблемы, которые необходимо решить в многомастерных системах:

Алгоритм необходимо реализовать так, чтобы только одно ведущее устройство могло завершить передачу. Все остальные ведущие устройства должны прекратить передачу после обнаружения потери процесса выбора. Данный процесс выбора носит название арбитрирование. Если ведущее устройство обнаруживает потерю процесса арбитрирования, то он должен сразу перейти в подчиненный режим, чтобы проверить не к нему ли обращается ведущее устройство, которое выиграло процесс арбитрирования. Факт начала одновременной передачи несколькими ведущими устройствами не должен обнаружиться подчиненными, т. е. передаваемые данные должны быть повреждены.

Разные ведущие устройства могут использовать разные частоты сигнала SCL. Необходимо придумать схему, которая позволяла бы засинхронизировать тактовые сигналы всех ведущих устройств.

Использование принципа монтажного "И" позволяет решить обе эти проблемы. Соединение тактовых сигналов всех ведущих устройств по схеме монтажного "И" означает, что длительность результирующего единичного импульса будет равна длительности самого короткого единичного импульса в этом соединении. Длительность низкого уровня результирующего тактового сигнала равна длительности низкого уровня тактового сигнала того ведущего устройства, у которого она максимальная. Следует обратить внимание, что все ведущие устройства, подключенные к линии SCL, начинают счет времени окончания периодов с высоким и низким состояниями, когда результирующий сигнал SCL переходит в высокое или низкое состояние, соответственно.

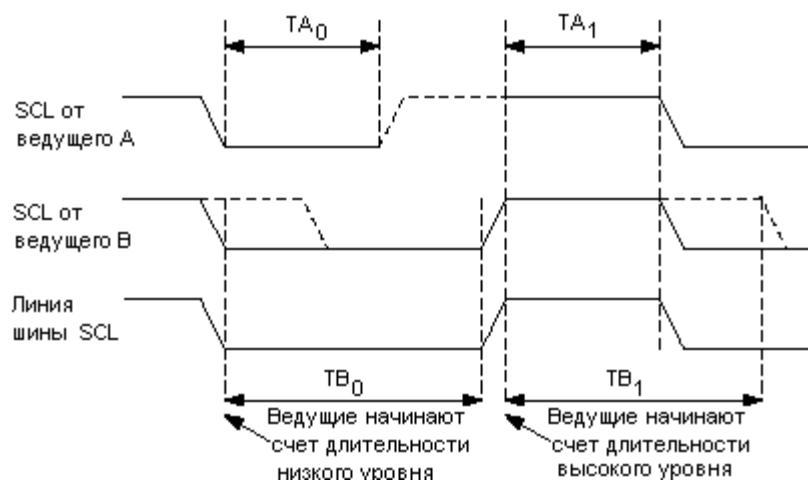


Рисунок 3.80 - Синхронизация на линии SCL между несколькими ведущими устройствами

Арбитрация реализована путем контроля состояния линии SDA всеми ведущими, выводящими данные. Если уровень, присутствующий на линии SDA не совпадает со значением, который передал ведущий, то данный ведущий теряет арбитрацию. Арбитрация теряется только в том случае, если ведущий передал логическую 1, а фактически на линии SDA присутствовал логический 0. После потери ведущим арбитрации, он немедленно переходит в подчиненный режим для определения: не к нему ли адресуется выигравший арбитрацию ведущий? Ведущие, которые проиграли процесс арбитрации, могут продолжать генерировать таймовый сигнал до окончания передачи текущего пакета адреса или данных, но при этом они должны выводить высокий уровень на линию SDA. Арбитрация выполняется до тех пор, пока останется активным только один ведущий и для этого в ряде случаев может быть передано много бит. Если несколько ведущих пытаются адресоваться к одному и тому же подчиненному, то процесс арбитрации переносится на пакет данных.

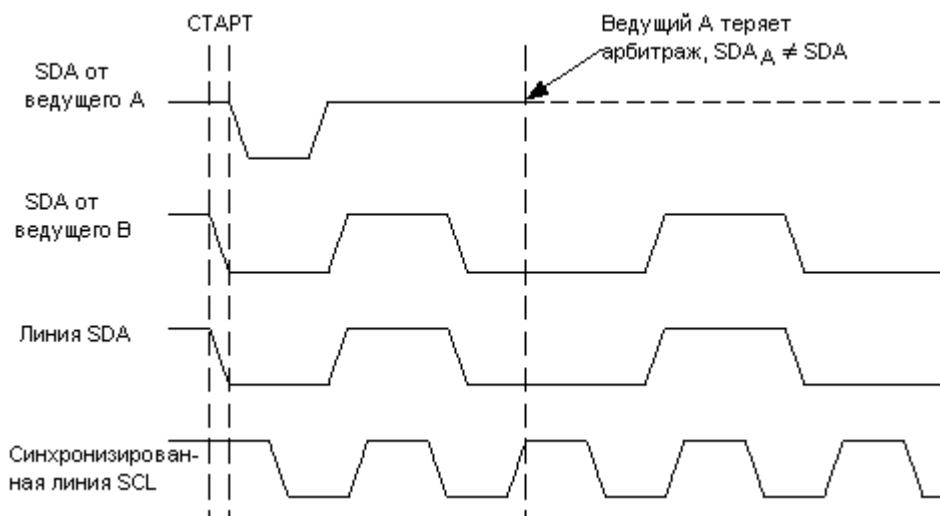


Рисунок 3.81 - Арбитрирование двух мастеров

Арбитрация не выполняется между передачей:

- условия ПОВТОРНЫЙ СТАРТ и бита данных
- условия СТОП и бита данных
- условия ПОВТОРНЫЙ СТАРТ и СТОП

Гарантируется невозможность возникновения данных условий, возлагается на программное обеспечение пользователя. Этим подразумевается, что во многомастерных системах должны использоваться одинаковое сочетание пакетов данных и ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ. Или иначе: любой сеанс связи должен состоять из одинакового числа пакетов данных, в противном случае результат арбитрации будет неопределенным.

3.16.8 Обзор модуля TWI

Модуль TWI состоит из нескольких подмодулей (смотрите рисунок 3.82). Все регистры, выделенные жирной линией на рисунке 3.82, доступны через шину данных микроконтроллера.

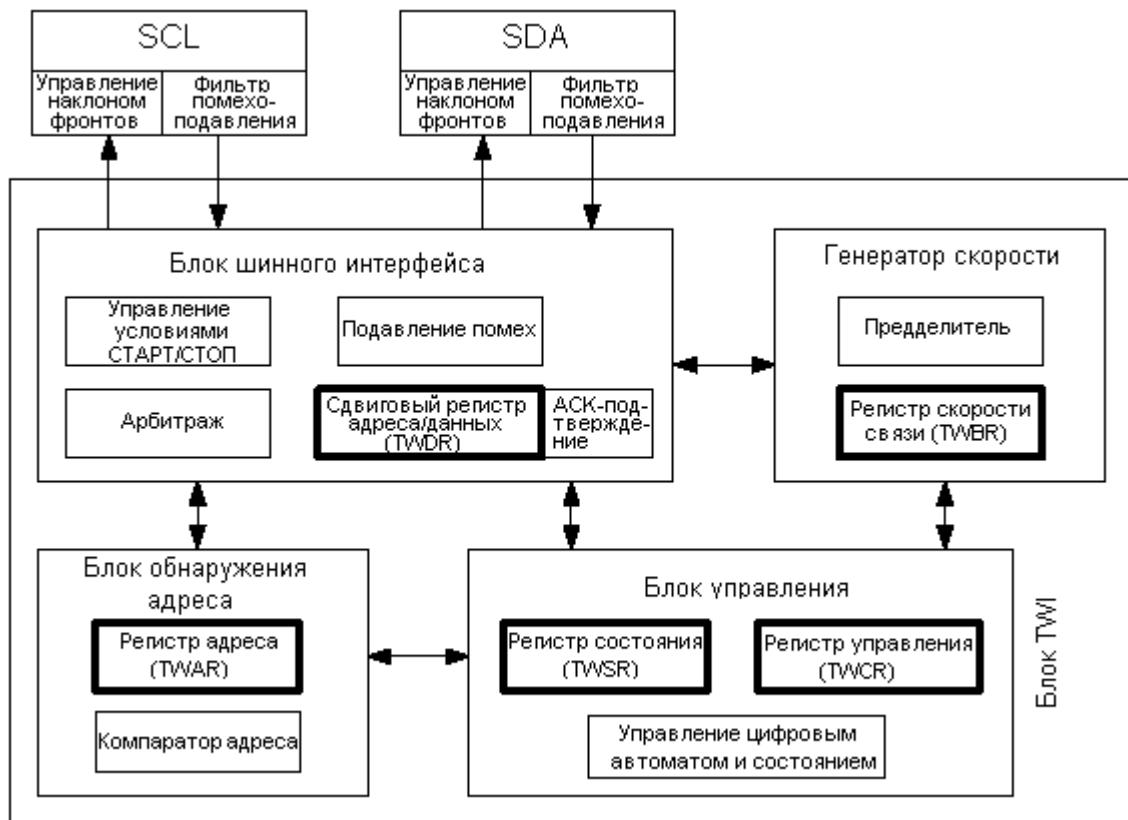


Рисунок 3.82 - Функциональная схема модуля TWI

Выводы SCL и SDA

Данные выводы связывают двухпроводной интерфейс микроконтроллера с остальными микроконтроллерами в системе. Драйверы выходов содержат ограничитель скорости изменения фронтов для выполнения требований к TWI. Входные каскады содержат блок подавления помех, задача которого состоит в игнорировании импульсов длительностью менее 50 нс. К каждой из этих линий можно подключить внутренний подтягивающий резистор путем установки разрядов PORTD.0 (SCL), PORTD.1 (SDA). Использование встроенных подтягивающих резисторов в ряде случаев позволяет отказаться от применения внешних.

Блок генератора скорости связи

Данный блок управляет периодом импульсов SCL в режиме ведущего устройства. Период SCL задается регистром скорости TWI (TWBR) и значением бит управления предделителем в регистре состояния TWI (TWSR). В подчиненном режиме значения скорости или установки предделителя не оказывают влияния на работу, но частота синхронизации ЦПУ подчиненного устройства должна быть минимум в 16 раз выше частоты SCL. Обратите внимание, что подчиненные могут продлевать длительность низкого уровня на линии SCL, тем самым уменьшая среднюю частоту синхронизации шины TWI. Частота SCL генерируется в соответствии со следующим выражением:

$$f_{SCL} = \frac{f_{\text{ЦПУ}}}{16 + 2 \cdot (TWBR) \cdot 4^{\text{TWPS}}},$$

где TWBR - значение регистра скорости TWI;

TWPS - значение бит предделителя в регистре состояния TWI.

Примечание - TWBR должен быть равен не менее 10, если TWI работает в ведущем режиме. Если TWBR меньше 10, то ведущий может генерировать некорректное состояние на линиях SDA и SCL. Проблема возникает при работе в ведущем режиме при передаче условий СТАРТ+ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ подчиненному.

Блок шинного интерфейса

Данный блок содержит сдвиговый регистр адреса и данных (TWDR), контроллер СТАРТА/СТОПа и схему арбитрации. TWDR содержит передаваемый байт адреса или данных или принятый байт адреса или данных. Помимо 8-разрядного регистра TWDR в состав блока шинного интерфейса также входит регистр, хранящий значение передаваемого или принятого бита НЕТ ПОДТВ. К данному регистру нет прямого доступа со стороны программного обеспечения. Однако во время приема он может устанавливаться или сбрасываться путем манипуляций с регистром управления TWI (TWCR). В режиме передатчика значение принятого бита НЕТ ПОДТВ можно определить по значению регистра TWSR.

Контроллер СТАРТА/СТОПа отвечает за генерацию и детекцию условий СТАРТ, ПОВТОРНЫЙ СТАРТ и СТОП. Контроллер СТАРТА/СТОПа позволяет обнаружить условия СТАРТ и СТОП, даже если микроконтроллер находится в одном из режимов сна. Этим обеспечивается возможность пробуждения микроконтроллера по запросу ведущего шины.

Если TWI инициировал передачу в качестве ведущего, то схема арбитрации непрерывно контролирует передачу, определяя возможность дальнейшей передачи. Если TWI теряет арбитрацию, то блок формирует соответствующий сигнал блоку управления, который выполняет адекватные действия и генерирует соответствующий код состояния.

Блок обнаружения адреса

Блок обнаружения адреса проверяет равен ли принятый адрес значению 7-разрядного адреса из регистра TWAR. Если установлен бит разрешения обнаружения общего вызова TWGCE в регистре TWAR, то все входящие адресные биты будут дополнительно сравниваться с адресом общего вызова. При адресном совпадении подается сигнал блоку управления, что позволяет выполнить ему необходимые действия. В зависимости от установки регистра TWCR подтверждение адреса TWI может происходить, а может и нет. Блок обнаружения адреса способен функционировать даже когда микроконтроллер переведен в режим сна, тем самым позволяя возобновить нормальную работу микроконтроллера по запросу мастера шины. Если при адресном совпадении TWI в экономичном режиме микроконтроллера, т. е. когда инициируется возобновление работы микроконтроллера, возникает другое прерывание (например, INT0), то TWI прекращает работу и возвращается к состоянию холостого хода (Idle). Если возникновение данного эффекта нежелательно, то необходимо следить, чтобы во время обнаружения адресования, когда микроконтроллер находится в режиме выключения (Power-down), было разрешено только одно прерывание.

Блок управления

Блок управления наблюдает за шиной TWI и генерирует отклики в соответствии с установками регистра управления TWI (TWCR). Если на шине TWI возникает событие, которое требует внимания со стороны программы, то устанавливается флаг прерывания TWINT. Следующим тактом обновляется содержимое регистра статуса TWI - TWSR, в котором будет записан код, идентифицирующий возникшее событие. Даная информация хранится в TWSR только тогда, когда установлен флаг прерывания TWI. Остальное время в регистре TWSR содержится специальный код состояния, который информирует о том, что нет информации о состоянии TWI. До тех пор пока установлен флаг TWINT, линия SCL остается в низком состоянии. Этим обеспечивается возможность завершить программе все задачи перед продолжением сеанса связи.

Флаг TWINT устанавливается в следующих ситуациях:

- после передачи условия СТАРТ/ПОВТОРНЫЙ_СТАРТ;
- после передачи ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ;
- после передачи адресного байта;
- после потери арбитрирования;
- после того как TWI адресован собственным подчиненным адресом или общим вызовом;
- после приема байта данных;
- после приема условия СТОП или ПОВТОРНЫЙ_СТАРТ в режиме подчиненной адресации;
- после возникновения ошибки по причине некорректного условия СТАРТ или СТОП.

3.16.9 Описание регистров TWI

Регистр скорости связи шины TWI - TWBR

Разряд	7	6	5	4	3	2	1	0	
	TWBR 7	TWBR 6	TWBR 5	TWBR 4	TWBR 3	TWBR 2	TWBR 1	TWBR 0	TWBR
	Чт/Зап								
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7...0 - Биты регистра скорости связи шины TWI

TWBR задает коэффициент деления частоты генератора скорости связи. Генератор частоты скорости связи - делитель частоты, который формирует сигнал синхронизации SCL в режимах "Ведущий". В разделе "Блок генератора скорости связи" показана методика вычисления скоростей связи.

Регистр управления шиной TWI - TWCR

Разряд	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	TWCR
	Чт/Зап								
Начальное значение	0	0	0	0	0	0	0	0	

Регистр TWCR предназначен для управления работой TWI. Он используется для разрешения работы TWI, для инициации сеанса связи ведущего путем генерации условия СТАРТ на шине, для генерации подтверждения приема, для генерации условия СТОП и для ОСТАНОВа шины во время записи в регистр TWDR. Он также сигнализирует о попытке ошибочной записи в регистр TWDR, когда доступ к нему был запрещен.

Разряд 7: TWINT - Флаг прерывания TWI

Данный бит устанавливается аппаратно, если TWI завершает текущее задание и ожидает реакции программы. Если бит I в SREG и бит TWIE в TWCR установлены, то микроконтроллер переходит на вектор прерывания TWI. Линия SCL остается в низком состоянии, пока установлен флаг TWINT. Флаг TWINT сбрасывается программно путем записи в него логической 1. Следует обратить внимание, что данный флаг сбрасывается не автоматически при переходе на вектор прерывания. Также нужно учесть, что очистка данного флага приводит к возобновлению работы TWI. Из этого следует, что программный сброс данного флага необходимо выполнить после завершения опроса регистров TWAR, TWSR и TWDR.

Разряд 6: TWEA - Бит разрешения подтверждения

Бит TWEA управляет генерацией импульса подтверждения. Если в бит TWEA записана логическая 1, то импульс ПОДТВ генерируется на шине TWI, если выполняется одно из следующих условий:

- принят собственный подчиненный адрес;
- принят общий вызов, когда установлен бит TWGCE в регистре TWAR;
- принят байт данных в режиме ведущего приемника или подчиненного приемника;
- запись логического 0 в бит TWEA позволяет временно отключиться от двухпроводной последовательной шины. Для возобновления распознавания адреса необходимо записать в данный бит логическую 1.

Разряд 5: TWSTA - Бит условия СТАРТ

Программист должен установить данный бит при необходимости стать ведущим на двухпроводной последовательнойшине. TWI аппаратно проверяет доступность шины и генерирует условие СТАРТ, если шина свободна. Однако если шина занята, то TWI ожидает появления условия СТОП, а затем генерирует новое условие СТАРТ для перевода состояния ведущего шины. TWSTA необходимо сбрасывать программно после передачи условия СТАРТ.

Разряд 4: TWSTO - Бит условия СТОП

Установка бита TWSTO в режиме ведущего приводит к генерации условия СТОП на двухпроводной последовательнойшине. Если на шине выполняется условие СТОП, то бит TWSTO сбрасывается автоматически. В подчиненном режиме установка бита TWSTO может использоваться для выхода из условия ошибки. В этом случае условие СТОП не генерируется, но интерфейс TWI возвращается к хорошо сконфигурированному безадресному подчиненному режиму и переводит линии SCL и SDA в высокоимпедансное состояние.

Разряд 3: TWWC - Флаг ошибочной записи

Бит TWWC устанавливается при попытке записи в регистр данных TWDR, когда TWINT имеет низкий уровень. Флаг сбрасывается при записи регистра TWDR, когда TWINT = 1.

Разряд 2: TWEN - Бит разрешения работы TWI

Бит TWEN разрешает работу TWI и активизирует интерфейс TWI. Если бит TWEN установлен, то TWI берет на себя функции управления линиями ввода-вывода SCL и SDA. При этом разрешается работа ограничителей скорости изменения фронтов и помехоподавляющих фильтров. Если данный бит равен нулю, то TWI отключается и все передачи прекращаются независимо от состояния работы.

Разряд 1 - Резервный бит

Данный бит является резервным и считывается как 0.

Разряд 0: TWIE - Разрешение прерывания TWI

Если в данный бит записана логическая 1 и установлен бит I в регистре SREG, то запрос на прерывание TWI будет генерироваться до тех пор, пока установлен флаг TWINT.

Регистр состояния TWI - TWSR

Регистр состояния TWI - TWSR

Разряд	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWS1	TWS0	TWSR
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7...3: TWS - Состояние TWI

Данные 5 бит отражают состояние логики блока TWI и двухпроводной последовательной шины. Различия в кодах состояния будут представлены далее в этом разделе. Следует обратить внимание, что считываемое значение из регистра TWSR содержит и 5-разр. код состояния и 2-разрядное значение, управляющее предделителем. Программист должен маскировать к 0 биты предделителя во время проверки бит состояния. В этом случае проверка состояния не будет зависеть от настройки предделителя.

Разряд 2 - Резервный бит

Данный бит является резервным и считывается как 0.

Разряды 1...0: TWPS - Биты предделителя TWI

Данные биты отличаются полным доступом (чтение/запись) и позволяют управлять предделителем скорости связи.

Таблица 3.73 - Предделитель скорости связи TWI

TWPS1	TWPS0	Значение предделения
0	0	1
0	1	4
1	0	16
1	1	64

Формула для вычисления скорости связи представлена в подразделе "Блок генератора скорости связи". Значение бит TWPS1...0 используется в ней.

Регистр данных шины TWI - TWDR

Разряд	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
	Чт/Зап								
Начальное значение	0	0	0	0	0	0	0	0	

Если TWI настроен на режим подчиненного передатчика или приемника, то будет реагировать только на адрес, записанный в этот регистр (в 7 старших разрядах TWAR). В остальных режимах ведущего данный регистр не используется. В многомастерных системах регистр TWAR устанавливается в том ведущем, к которому адресуются как к подчиненному другие ведущие шины.

Младший разряд регистра TWAR используется для разрешения обнаружения адреса общего вызова (\$00). Специальный компаратор выполняет сравнение подчиненного адреса (или адреса общего вызова) с принятым адресом. Если обнаруживается совпадение, то генерируется запрос на прерывание.

Разряды 7...1: TWA - Регистр подчиненного адреса TWI

Данные семь бит составляют подчиненный адрес блока TWI.

Разряд 0: TWGCE - Бит разрешения обнаружения общего вызова по шине TWI

После установки данного бита разрешается работа схемы обнаружения общего вызова, передаваемого по двухпроводной последовательнойшине.

3.16.10 Рекомендации по использованию регистра TWI

TWI ориентирован на передачу данных в байтном формате с управлением по прерываниям. Прерывания возникают после обнаружения одного из событий нашине, например, прием байта или передача условия СТАРТ. Управление TWI по прерываниям позволяет освободить программное обеспечение на выполнение других задач во время передачи байта данных. Следует обратить внимание, что установка флага TWINT приводит к генерации запроса на прерывание только в том случае, когда установлен бит разрешения прерывания TWIE в регистре TWCR, а также разрешена работа прерываний установкой бита в регистре SREG. Если бит TWIE сброшен, то состояние TWINT должно отслеживаться программно для оценки ситуации нашине TWI.

После установки флага TWINT интерфейс TWI приостанавливает работу и ожидает реакции программы. В этом случае регистр статуса TWI (TWSR) содержит значение, которое индицирует текущее состояние шины TWI. Исходя из этого, программа задает дальнейшее поведение шины TWI, манипулируя регистрами TWCR и TWDR.

На рисунке 3.83 показан простой пример подключения кшине TWI. Здесь предполагается, что мастер желает передать один байт данных подчиненному. Данное описание весьма общее, а более подробно объяснение приводится далее в этом подразделе.

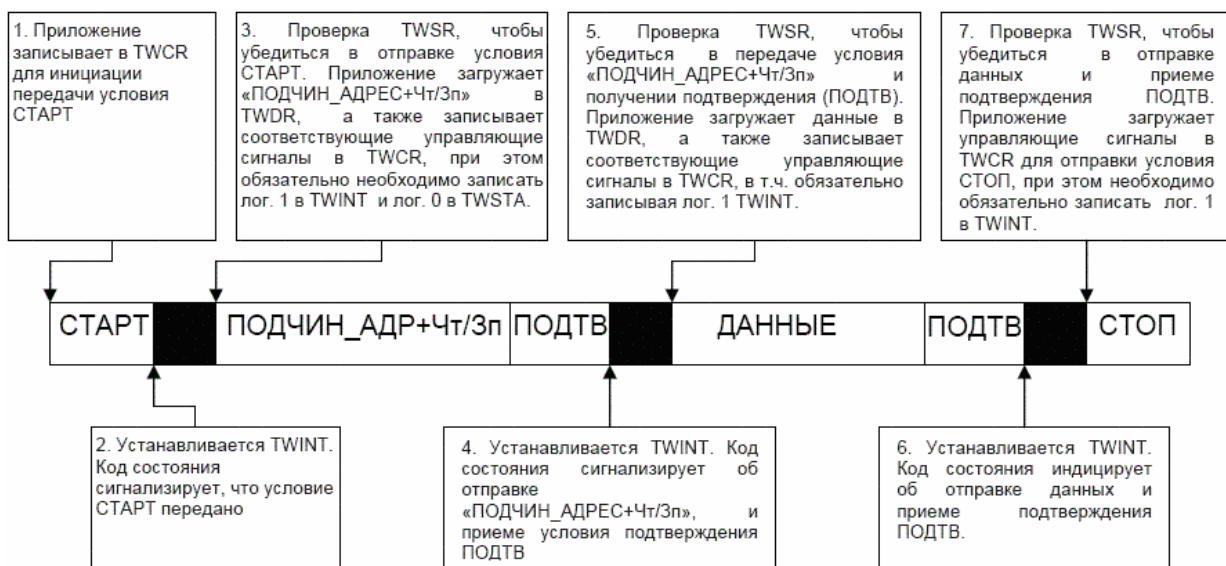


Рисунок 3.83 - Последовательность обслуживания TWI при типичной передаче

Первым шагом работы регистра TWI является передача условия СТАРТ. Это инициируется путем записи специфического значения в TWCR. О значении, которое необходимо записать, будет сказано позже. Однако необходимо следить, чтобы в записываемом в регистр значении был установлен бит TWINT. Запись логической 1 в TWINT сбрасывает этот флаг. TWI не начнет работу до тех пор, пока будет установлен флаг TWINT в регистре TWCR. Сразу после сброса TWINT начинается передача условия СТАРТ.

После передачи условия СТАРТ устанавливается флаг TWINT в регистре TWCR, а содержимое TWSR обновляется значением кода состояния, индицирующего об успешной передаче условия СТАРТ.

В программе необходимо выполнить проверку значения TWSR, чтобы убедиться в том, что условие СТАРТ было успешно передано. Если TWSR индицирует прочую ситуацию, то программа выполняет особые действия, например, вызывает процедуру обработки ошибочных ситуаций. Если код состояния имеет ожидаемое значение, то выполняется загрузка условия ПОДЧИН_АДР + ЗАПИСЬ в TWDR. Необходимо помнить, что TWDR используется для хранения как адреса, так и данных. После загрузки в TWDR желаемого значения ПОДЧИН_АДР + ЗАПИСЬ в регистр TWCR должно быть записано специфическое значение, которое служит командой для передачи значения ПОДЧИН_АДР + ЗАПИСЬ, хранящегося в TWDR. Какое именно значение необходимо записать будет сказано позже. Однако необходимо следить, чтобы в записываемом в регистр значении был установлен бит TWINT. Запись логической 1 в TWINT приводит к сбросу этого флага. TWI не начнет работу до тех пор, пока установлен бит TWINT в регистре TWCR. Сразу после сброса флага TWINT инициируется передача адресного пакета.

После передачи адресного пакета устанавливается флаг TWINT в регистре TWCR, а содержимое регистра TWSR обновляется кодом состояния, индицирующего успешность передачи адресного пакета. В коде состояния также отражается было ли подтверждение приема адресного пакета со стороны подчиненного или нет.

Выполняется программная проверка значения TWSR, чтобы убедиться в успешности передачи адресного пакета и что бит подтверждения ПОДТВ имеет ожидаемое значение. Если TWSR индицирует иную ситуацию, то при необходимости выполняются особые действия, например, вызывается процедура обработки ошибочных ситуаций. Если же код состояния имеет ожидаемое значение, то программа записывает пакет данных в TWDR. Впоследствии в регистр TWCR записывается специфическое значение, которое служит командой для TWI и вызывает аппаратную передачу данных, записанных в TWDR. Необходимо учесть, что в записываемом значении должен быть установлен бит

TWINT. Запись логической 1 в TWINT приводит к сбросу этого флага. TWI не начнет работу до тех пор пока будет установлен бит TWINT в регистре TWCR. Сразу после сброса TWINT начинается передача пакета данных.

После передачи пакета данных устанавливается флаг TWINT в регистре TWCR, а содержимое регистра TWSR обновляется значением кода состояния, который сигнализирует об успешной передачи пакета данных. В коде состояния также отражается: было ли принято подтверждение от подчиненного или нет.

Выполняется программная проверка значения в TWSR, чтобы убедиться в успешности передачи пакета данных и в том, что бит ПОДТВ имеет ожидаемое значение. Если TWSR индицирует иную ситуацию, то программа выполняет особые действия, в т. ч. вызывает процедуру обработки прерывания. Если код состояния имеет ожидаемое значение, то выполняется запись специального значения в TWCR, которое служит командой для TWI и инициирует передачу условия СТОП. Какое именно значение необходимо записать, сказано далее. Однако следует учесть, что во время записи должна быть произведена установка бита TWINT. Запись логической 1 в TWINT приводит к очистке этого флага. TWI не начнет работу до тех пор, пока установлен бит TWINT в регистре TWCR. Сразу после сброса флага TWINT инициируется передача условия СТОП. Следует обратить внимание, что флаг TWINT не устанавливается по завершении передачи условия СТОП.

Несмотря на простоту изложенного примера, он показывает принципы, положенные в основу любой передачи через TWI. Из вышеизложенного можно сделать следующие выводы:

- по завершении работы регистра TWI устанавливается флаг TWINT и далее ожидается реакция со стороны программы. Линия находится в низком состоянии, пока сброшен флаг TWINT;
- если флаг TWINT установлен, то пользователь может обновлять любой из регистров TWI значением, которое относится к следующему этапу работы шины TWI. Например, в TWDR загружается значение, которое необходимо передать на следующем цикле шины;
- после обновления всех регистров TWI и завершении других задач выполняется запись в TWCR. Во время записи TWCR необходимо, чтобы был установлен бит TWINT. В этом случае запись логической 1 в TWINT приведет к сбросу данного флага. TWI выполняет действия в соответствии с установкой регистра TWCR.

Далее показан пример на языке «Ассемблере» и «Си». В примере предполагается, что все символьные обозначения определены в присоединенном файле.

Пример кода на языке «Ассемблер»	Пример кода на языке «Си»	Комментарий
1 ldi r16, (1<<TWINT) (1<<TWSTA) (1<<TWEN) out TWCR, r16	TWCR = (1<<TWINT) (1<<TWSTA) (1<<TWEN)	Передача условия СТАРТ
2 wait1: in r16,TWCR sbrs r16,TWINT rjmp wait1	while (!(TWCR & (1<<TWINT))) ;	Ожидание установки флага TWINT. Этим индицируется завершение передачи условия СТАРТ
3 in r16,TWSR andi r16, 0xF8 cpi r16, START brne ERROR	if ((TWSR & 0xF8) != START) ERROR();	Проверка кода состояния TWI. Маскирующий бит предделителя. Если код состояния не равен СТАРТ, то переход на ERROR
ldi r16, SLA_W out TWDR, r16 ldi r16, (1<<TWINT) (1<<TWEN) out TWCR, r16	TWDR = SLA_W; TWCR = (1<<TWINT) (1<<TWEN);	Загрузка ПОДЧИН_АДР + ЗАПИСЬ в регистр TWDR. Сброс бита TWINT в TWCR для начала передачи адреса

Пример кода на языке «Ассемблере»	Пример кода на языке «Си»	Комментарий
4 wait2: in r16,TWCR sbrs r16,TWINT rjmp wait2	while (!(TWCR & (1<<TWINT))) ;	Ожидание установки флага TWINT. Этим сигнализируется завершение передачи ПОДЧИН_АДР + ЗАПИСЬ и получение/неполучение подтверждения (ПОДТВ/НЕТ ПОДТВ).
5 in r16,TWSR andi r16, 0xF8 cpi r16, MT_SLA_ACK brne ERROR	if ((TWSR & 0xF8) != MT_SLA_ACK) ERROR();	Проверка значения регистра состояния. Маскирование бит предделителя. Если состояние отличается от MT_SLA_ACK, то переход на ERROR
	ldi r16, DATA out TWDR, r16 ldi r16, (1<<TWINT) (1<<TWEN) out TWCR, r16	TWDR = DATA; TWCR = (1<<TWINT) (1<<TWEN);
6 wait3: in r16,TWCR sbrs r16,TWINT rjmp wait3	while (!(TWCR & (1<<TWINT))) ;	Ожидание установки флага TWINT. Этим индицируется, что данные были переданы и принято/не принято подтверждение (ПОДТВ/НЕТ ПОДТВ).
7 in r16,TWSR andi r16, 0xF8 cpi r16, MT_DATA_ACK brne ERROR	if ((TWSR & 0xF8) != MT_DATA_ACK) ERROR();	Проверка значения регистра состояния TWI. Маскирование бит предделителя. Если состояние отличается от MT_DATA_ACK, то переход на ERROR
	ldi r16, (1<<TWINT) (1<<TWEN) (1<<TWSTO) out TWCR, r16	TWCR = (1<<TWINT) (1<<TWEN) (1<<TWSTO);
Примечание - Если фактический регистр расположен в расширенной памяти ввода-вывода, то инструкции "IN", "OUT", "SBIS", "SBIC", "CBI" и "SBI" необходимо заменить теми инструкциями, которые эффективны для данной области памяти. Обычно это инструкции "LDS" и "STS" в сочетании с "SBRS", "SBRC", "SBR" и "CBR".		

Режимы передачи

Регистр TWI может работать в одном из 4-х режимов работы. Они называются: ведущий передатчик (MT), ведущий приемник (MR), подчиненный передатчик (ST) и подчиненный приемник (SR). Некоторые из этих режимов могут использоваться в рамках одного и того же приложения. Например, TWI может использовать режим MT для записи данных в двупроводное последовательное ЭСППЗУ, а режим MR для считывания данных из ЭСППЗУ. Если в системе имеются другие ведущие (мастера), один из которых передает данные, то у остальных используется режим SR. Какой из режимов должен использоваться определяется программно.

Далее описан каждый из этих режимов. Возможные значения кодов состояния представлены рядом с рисунками, детализирующими процесс передачи данных в каждом из режимов. На рисунках используются следующие аббревиатуры:

- СТАРТ: Условие СТАРТ;
- ПОВТ. СТАРТ: Условие ПОВТОРНЫЙ СТАРТ;

- ЧТЕНИЕ: Бит "Чтение" (высокий уровень на SDA);
- ЗАПИСЬ: Бит "Запись" (низкий уровень на SDA);
- ПОДТВ.: Бит подтверждения (низкий уровень на SDA);
- НЕТ ПОДТВ.: Нет бита подтверждения (высокий уровень на SDA);
- ДАННЫЕ: 8-разрядный байт данных;
- СТОП: Условие СТОП;
- ПОДЧИН_АДР.: Подчиненный адрес.

На рисунках 3.85 окружности используются для индикации установки флага TWINT. Число, записанное внутри окружности, является кодом состояния из регистра TWSR, в котором замаскированы к нулю биты предделителя. В данном состоянии ожидается действие со стороны программы для завершения передачи TWI. Передача TWI приостанавливается до тех пор, пока программно не будет сброшен флаг TWINT.

После установки флага TWINT по значению кода состояния из регистра TWSR определяется, какое действие выполнить программе. В таблицах 88-91 представлена информация о том, какие программные действия должны быть предприняты при различных значениях кода состояния. Следует обратить внимание, что в таблицах биты предделителя замаскированы нулевыми значениями.

Режим ведущего передатчика

В режиме ведущего передатчика (MT) байты данных передаются подчиненному приемнику (SR) (см. рисунок 3.84). Для ввода режима ведущего необходимо передать условие СТАРТ. Формат следующего адресного пакета определяет какой режим вводится: ведущий передатчик (MT) или ведущий приемник (MR). Если передается ПОДЧИН_АДР+ ЗАПИСЬ, то вводится режим MT (ведущий передатчик), а если ПОДЧИН_АДР + ЧТЕНИЕ, то вводится режим MR (ведущий приемник). Все упоминаемые в этом разделе коды состояния в позиции бит предделителя имеют нулевые значения.

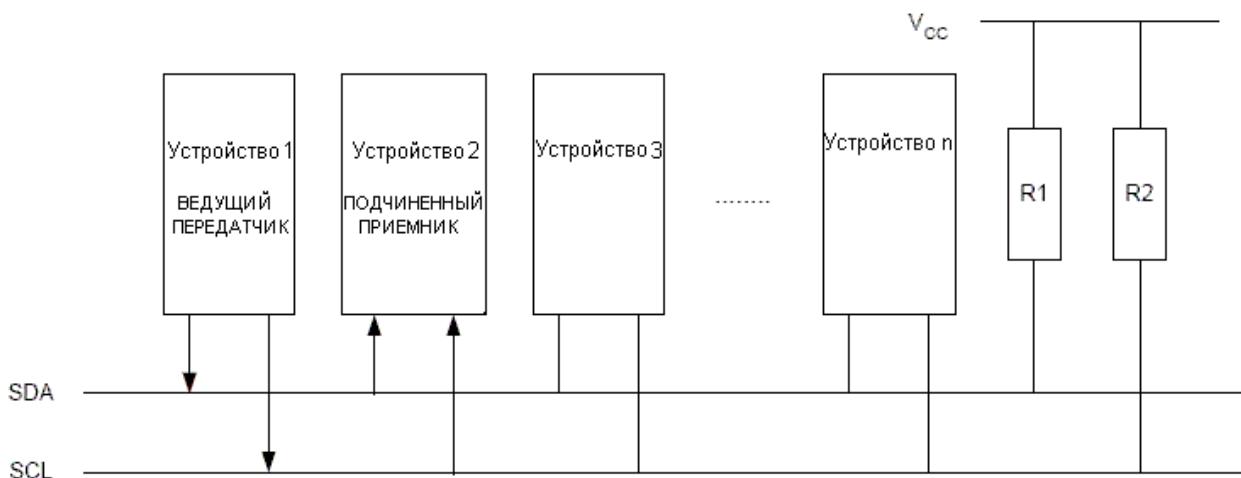


Рисунок 3.84 - Передача данных в режиме ведущего передатчика

Передача условия СТАРТ инициируется путем записи в TWCR следующего значения:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

Для разрешения работы двухпроводного последовательного интерфейса необходимо установить бит TWEN. Запись логической 1 в TWSTA инициирует передачу усло-

вия СТАРТ, а запись логической 1 в TWINT приводит к сбросу флага TWINT. После записи данного значения TWI тестирует двухпроводную последовательную шину и генерирует условие СТАРТ сразу после освобождения шины. После передачи условия СТАРТ аппаратно устанавливается флаг INT, а в регистр TWSR помещается код состояния \$08 (см. таблицу 3.74). Для перевода в режим ведущего передатчика необходимо передать ПОДЧИН_АДР + ЗАПИСЬ. Это выполняется путем записи значения ПОДЧИН_АДР + ЗАПИСЬ в регистр TWDR. После этого необходимо сбросить флаг TWINT (путем записи в него логической 1) для продолжения сеанса связи. Данное выполняется путем записи следующего значения в TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	0	x	1	0	x

После передачи ПОДЧИН_АДР + ЗАПИСЬ и приема бита подтверждения флаг TWINT снова устанавливается, а в регистр TWSR помещается код состояния, который может иметь несколько значений. В режиме ведущего код состояния может быть \$18, \$20 или \$38. Для каждого из этих кодов состояний необходимо выполнить адекватные действия, что отражено в таблице 3.74.

После успешной передачи ПОДЧИН_АДР + ЗАПИСЬ должен быть передан пакет данных. Его передача инициируется записью байта данных в TWDR. Доступ на запись к TWDR разрешен только тогда, когда флаг TWINT равен 1. В противном случае доступ блокируется и устанавливается флаг ошибочной записи TWWC в регистре TWCR. После обновления TWDR необходимо сбросить бит TWINT (путем записи в него логической 1) для продолжения сеанса связи. Данное можно выполнить путем записи следующего значения в регистр TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	0	x	1	0	x

Данная последовательность повторяется до тех пор, пока не будет передан последний байт. После этого генерируется условие СТОП или ПОВТОРНЫЙ СТАРТ. Условие СТОП генерируется путем записи следующего значения TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	1	x	1	0	x

Условие ПОВТОРНЫЙ СТАРТ генерируется путем записи следующего значения в TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

После передачи условия ПОВТОРНОГО СТАРТА (состояние \$10) двухпроводной последовательный интерфейс может обращаться к тому же подчиненному устройству или же к новому, при этом не требуется передача условия СТОП. Таким образом, ПОВТОРНЫЙ СТАРТ полезно использовать для смены подчиненного устройства в режимах ведущий передатчик и ведущий приемник без потери управления шиной.

Таблица 3.74 - Коды состояния в режиме ведущего передатчика

Код состояния (TWCSR), биты предделителя равны 0	Состояние двухпроводной последовательной шины и схемы двухпроводного последовательного интерфейса	Программные действия					Следующее действие, выполняемое схемой ТМ	
		В/из TWDR		В TWCR				
		STA	STO	TWINT	TWEAD			
\$08	Передано условие СТАРТ	Загрузка ПОДЧИН_АДР+ЗАПИСЬ	0	0	1	x	Передается ПОДЧИН_АДР + ЗАПИСЬ Принимается ПОДТВ. или НЕТ ПОДТВ.	
\$10	Передано условие ПОВТОРНЫЙ СТАРТ	Загрузка ПОДЧИН_АДР+ЗАПИСЬ или ПОДЧИН_АДР+ЧТЕНИЕ	0	0	1	x	Передается ПОДЧИН_АДР + ЗАПИСЬ; Принимается ПОДТВ или НЕТ ПОДТВ; Передается ПОДЧИН_АДР + ЧТЕНИЕ; Переход на режим ведущего приемника	
\$18	Передано ПОДЧИН_АДР+ЗАПИСЬ и принято ПОДТВерждение	Загрузка байта данных или действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR	0	0	1	x	Передается байт данных, принимается или не принимается ПОДТВерждение Передается ПОВТОРНЫЙ СТАРТ Передается условие СТОП и сбрасывается флаг TWSTO Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO	
\$20	Передано ПОДЧИН_АДР+ЗАПИСЬ и принято НЕТ ПОДТВ	Загрузка байта данных или действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR	0	0	1	x	Передается байт данных, принимается или не принимается ПОДТВерждение Передается ПОВТОРНЫЙ СТАРТ Передается условие СТОП и сбрасывается флаг TWSTO Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO	
\$28	Передается байт данных; принимается ПОДТВерждение	Загрузка байта данных или действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR	0	0	1	x	Передается байт данных, принимается или не принимается ПОДТВерждение Передается ПОВТОРНЫЙ СТАРТ Передается условие СТОП и сбрасывается флаг TWSTO Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO	
\$30	Передается байт данных; принимается НЕТ ПОДТВерждения	Загрузка байта данных или действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR	0	0	1	x	Передается байт данных, принимается или не принимается ПОДТВерждение Передается ПОВТОРНЫЙ СТАРТ Передается условие СТОП и сбрасывается флаг TWSTO Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO	

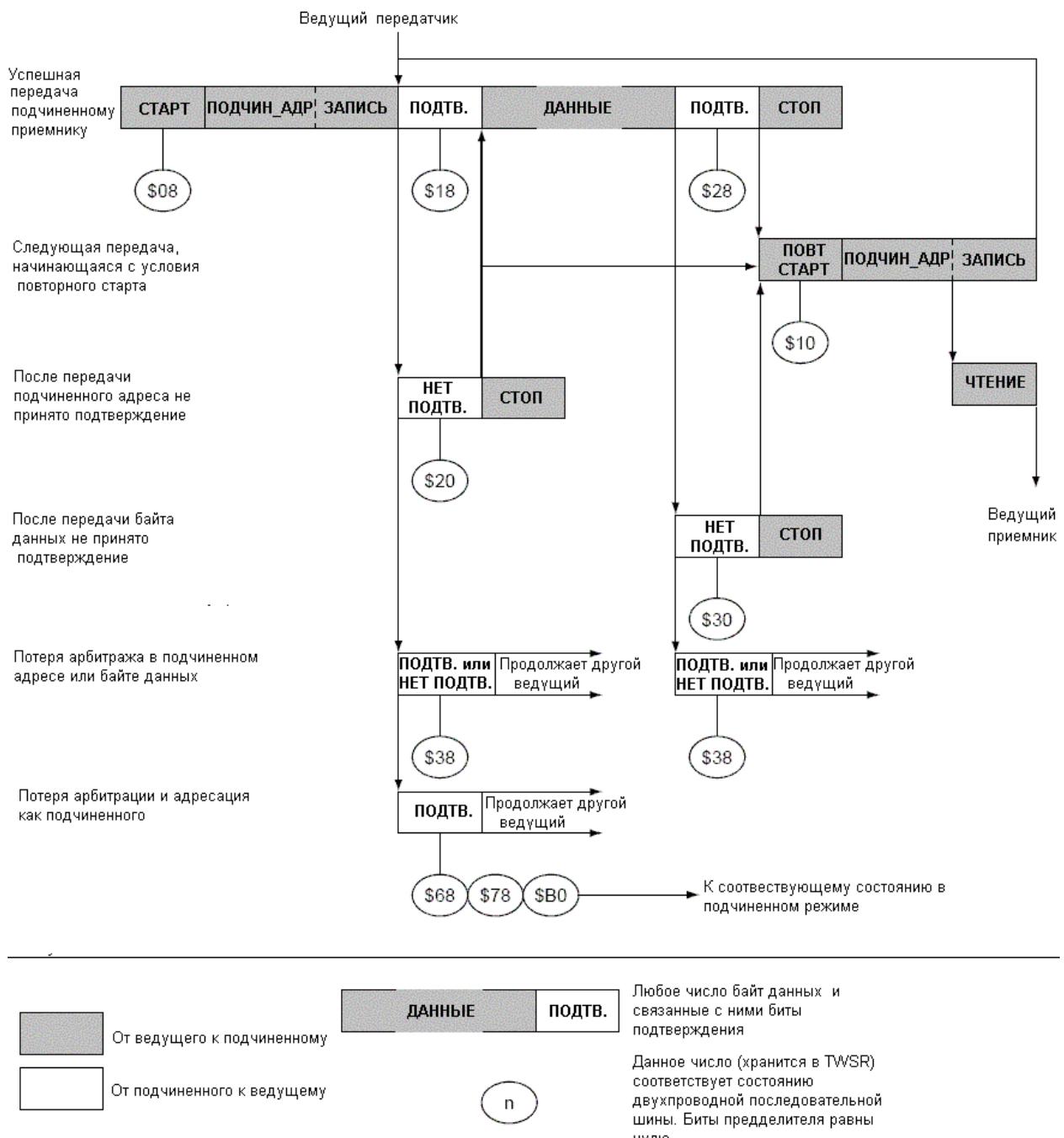


Рисунок 3.85 - Форматы и состояния в режиме ведущего передатчика

Режим «ведущего приемника»

В режиме ведущего приемника принимается несколько байт данных от подчиненного приемника (см. рисунок 3.86). Для ввода режима ведущего необходимо передать условие СТАРТ. Формат следующего адресного пакета определит, будет ли введенный режим ведущий передатчик или приемник. Если передается ПОДЧИН_АДР+ЗАПИСЬ, то вводится режим ведущий передатчик, если же передается ПОДЧИН_АДР+ЧТЕНИЕ, то вводится режим ведущий приемник. Во всех кодах состояния, приведенных в этом подразделе, не учитываются биты предделителя, и они равны нулю.

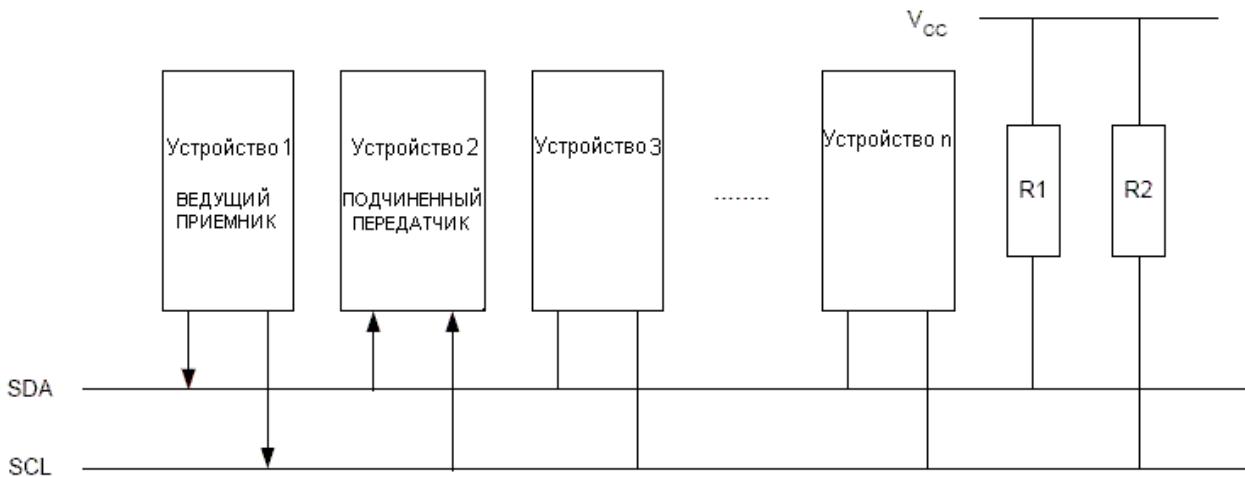


Рисунок 3.86 - Передача данных в режиме ведущего приемника

Передача условия СТАРТ инициируется путем записи следующего значения в TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

Для разрешения работы двухпроводного последовательного интерфейса необходимо установить бит TWEN. Передача условия СТАРТ инициируется записью логической 1 в TWSTA. Для сброса флага TWINT необходимо записать в него логическую 1. TWI выполнит генерацию условия СТАРТ только после тестирования шины и определения ее освобождения. После передачи условия СТАРТ флаг TWINT устанавливается аппаратно, а в регистр TWSR помещается код состояния \$08 (см. таблицу 3.75). Для ввода режима ведущий приемник необходимо передать ПОДЧИН_АДР+ЧТЕНИЕ, выполняемое путем записи значения ПОДЧИН_АДР+ЧТЕНИЕ в TWDR. После этого необходимо сбросить флаг TWINT (путем записи в него логической 1) для продолжения сеанса связи. Для этого в регистр TWCR необходимо поместить следующее значение:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	0	x	1	0	x

После передачи ПОДЧИН_АДР+ЧТЕНИЕ и приема бита подтверждения устанавливается снова флаг TWINT, а в регистр TWSR помещается код состояния, который может иметь несколько значений: \$38, \$40 или \$48. Действия, которые выполняются при каждом из этих значений, представлены в таблице 97. Принятые данные хранятся в регистре TWDR после аппаратной установки флага TWINT. Данная последовательность повторяется до приема последнего байта. После этого ведущий приемник информирует подчиненный передатчик отправкой НЕТ ПОДТВ (нет подтверждения) после приема последнего принятого байта данных. Сеанс связи завершается генерацией условия СТОП или ПОВТОРНЫЙ СТАРТ. Условие СТОП генерируется путем записи в регистр TWCR следующего значения:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	1	x	1	0	x

Условие ПОВТОРНЫЙ СТАРТ генерируется путем записи в TWCR следующего значения:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

После генерации условия ПОВТОРНЫЙ СТАРТ (состояние \$10) двухпроводной последовательный интерфейс может обращаться к тому же подчиненному или к новому подчиненному без генерации условия СТОП. ПОВТОРНЫЙ СТАРТ позволяет ведущему переключаться между подчиненными, режимом ведущего передатчика и ведущего приемника без потери управления над шиной.

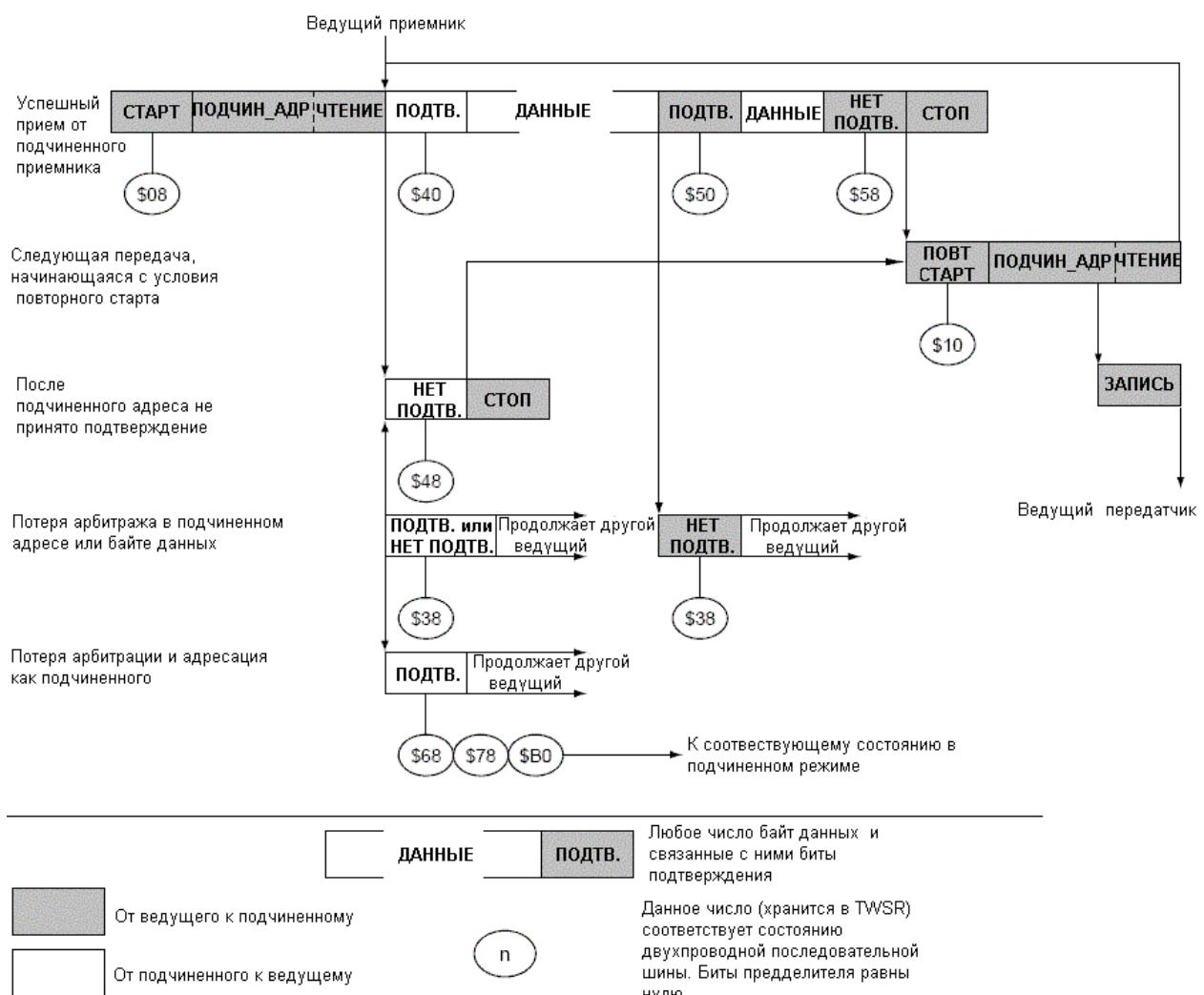


Рисунок 3.87 - Форматы и состояния в режиме ведущего приемника

Таблица 3.75 - Коды состояния для режима ведущего приемника

Код состояния (TWSR), биты предделителя равны 0	Состояние двухпроводной последовательной шины и схемы двухпроводного последовательного интерфейса	Программные действия				Следующее действие, выполняемое схемой ТМ	
		В из TWDR		В TWCR			
		STA	STO	TWINT	TWEA		
\$08	Передано условие СТАРТ	Загрузка ПОДЧИН_АДР+ЧТЕНИЕ	0	0	1	x	Передается ПОДЧИН_АДР + ЧТЕНИЕ Принимается ПОДТВ. или НЕТ ПОДТВ.
\$10	Передано условие ПОВТОРНЫЙ СТАРТ	Загрузка ПОДЧИН_АДР+ЧТЕНИЕ или ПОДЧИН_АДР+ЗАПИСЬ	0	0	1	X	Передается ПОДЧИН_АДР+ЧТЕНИЕ, принимается ПОДТВ. или НЕТ ПОДТВ. Передается ПОДЧИН_АДР+ЗАПИСЬ, переключение на режим «Ведущий передатчик»
\$38	Потеря арбитрации во время передачи бит ПОДЧИН_АДР + ЧТЕНИЕ или бита НЕТ ПОДТВ.	Действия без загрузки TWDR или действия без загрузки TWDR	0	0	1	X	Шина освобождается и вводится безадресный подчиненный режим Условие СТАРТ передается после освобождения шины
\$40	Передано ПОДЧИН_АДР+ЧТЕНИЕ и принято ПОДТВ	Действия без загрузки TWDR или действия без загрузки TWDR	0	0	1	0	Принимается байт данных, возвращается бит НЕТ_ПОДТВ. Принимается байт данных, возвращается бит ПОДТВ.
\$48	Передано ПОДЧИН_АДР+ЧТЕНИЕ и принято НЕТ ПОДТВ	Действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR	1	0	1	X	Передается ПОВТОРНЫЙ СТАРТ Передается условие СТОП и сбрасывается флаг TWSTO Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO
\$50	Принят байт данных и возвращается ПОДТВерждение	Чтение байта данных или чтение байта данных	0	0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВ Принимается байт данных и возвращается ПОДТВ
\$58	Принят байт данных и возвращается НЕТ ПОДТВерждения	Чтение байта данных или чтение байта данных или чтение байта данных	1	0	1	X	Передается ПОВТОРНЫЙ СТАРТ Передается СТОП и сбрасывается флаг TWSTO Вслед за условием СТАРТ передается условие СТОП и сбрасывается флаг TWSTO

Режим подчиненного приемника

В режиме подчиненного приемника принимается несколько байт данных от ведущего передатчика (смотрите рисунок 3.88). Во всех кодах состояния, приведенных в этом подразделе, не учитываются биты предделителя, и они равны нулю.

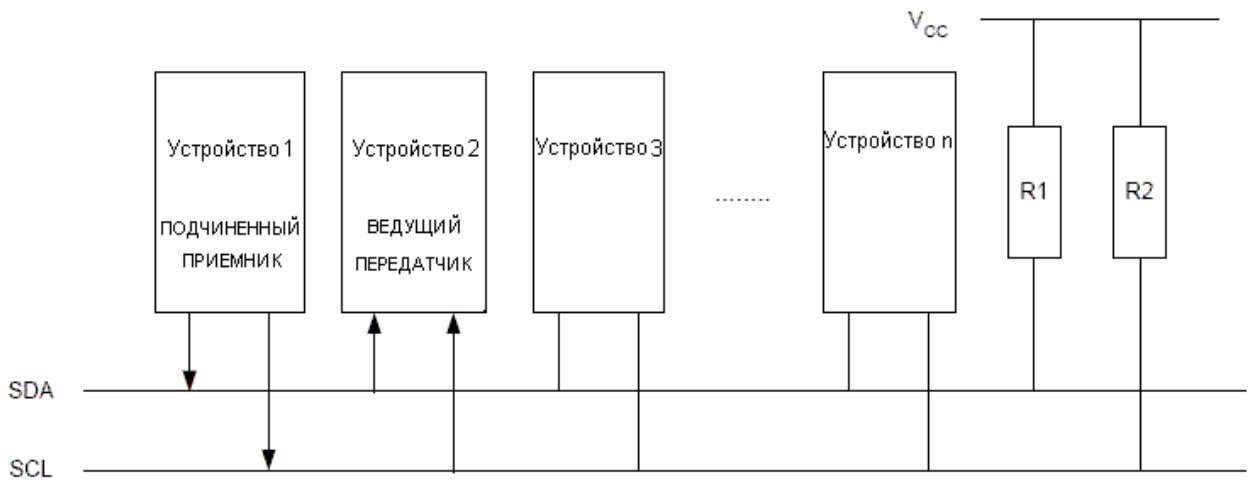


Рисунок 3.88 - Передача данных в режиме подчиненного приемника

Для ввода режима подчиненного приемника необходимо выполнить инициализацию регистров TWAR и TWCR следующим образом:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Значение	Собственный подчиненный адрес устройства							

Старшие семь разрядов образуют адрес. С помощью него определяется к какому двухпроводному интерфейсу адресуется мастер. Если в младшем разряде записана логическая 1, то TWI будет отвечать на адрес общего вызова (\$00). В противном случае он игнорирует адрес общего вызова.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	0	1	0	0	0	1	0	x

Для разрешения работы TWI необходимо записать логическую 1 в TWEN. Для разрешения подтверждения собственно подчиненного адреса или адреса общего вызова записывается 1 в TWEA. Битам TWSTA и TWSTO необходимо присвоить нулевое значение.

После инициализации TWAR и TWCR схема TWI ожидает получения собственно подчиненного адреса (или, если разрешено, адреса общего вызова), а вслед за ним - бита направления данных. Если бит направления равен "0" (запись), то TWI переходит в режим "подчиненный приемник", в противном случае вводится режим "подчиненный передатчик". После приема собственного подчиненного адреса и бита записи устанавливается флаг TWINT, а в регистр TWSR помещается код состояния. По коду состояния определяется какие программные действия необходимо предпринять. В таблице 3.76 собрана информация о предпринимаемых действиях для каждого возможного значения кода состояния. Режим подчиненного приемника также вводится, если теряется арбитрация, когда TWI находился в режиме ведущего (смотрите состояния \$68 и \$78).

Если бит TWEA сбросить во время передачи, то TWI ответит "НЕТ ПОДТВ." ("1") на линии SDA после приема следующего байта данных. Данное свойство может использоваться для сигнализации состояния, когда подчиненный не может больше принимать байты данных. Если TWEA равен нулю, то TWI не подтверждает свой подчиненный адрес. Однако последовательная шина остается под контролем и функция распознавания адреса может быть активизирована в любой момент путем установки бита TWEA. Это означает, что бит TWEA можно использовать для временной изоляции TWI от двухпроводной последовательной шины.

Во всех режимах сна, кроме холостого хода (Idle), синхронизация TWI отключается. Если бит TWEA установлен, то интерфейс останется способным подтверждать прием своего собственного подчиненного адреса или адреса общего вызова за счет использования сигнала синхронизации шины в качестве тактового источника. При обнаружении запроса микроконтроллер выходит из режима сна, при этом линия SCL остается на низком уровне в процессе пробуждения и до сброса флага TWINT (записью в него "1"). Далее выполняется прием данных обычным способом при обычной системной синхронизации. Учтите, что если для микроконтроллера выбрано большое время запуска, то линия SCL может оказаться длительно в низком состоянии и заблокировать другой обмен информацией.

Обратите внимание, что после пробуждения регистр данных TWDR не отражает последний байт, присутствовавший нашине во время выхода из указанных выше режимов сна.

Таблица 3.76 - Коды состояния для режима подчиненный приемник

Код состояния (TWSR), биты предделителя равны 0	Состояние двухпроводной последовательной шины и схемы двухпроводного последовательного интерфейса	Программные действия				Следующее действие, выполняемое схемой ТМ
		В/из TWDR		В TWCR		
		STA	STO	TWINT	TWEA	
\$60	Принимается собственный ПОДЧИН_АДР+ЗАПИСЬ; возвращено ПОДТВ.	Действия без загрузки TWDR или действия без загрузки TWDR	x 0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
			x 0	1	1	Принимается байт данных и возвращается ПОДТВерждение
\$68	Потеряна арбитрация во время передачи ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ как ведущего, принят собственный ПОДЧИН_АДР+ЗАПИСЬ, возвращено ПОДТВ	Действия без загрузки TWDR или действия без загрузки TWDR	x 0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
			x 0	1	1	Принимается байт данных и возвращается ПОДТВерждение
\$70	Принят адрес общего вызова; возвращено подтверждение	Действия без загрузки TWDR или действия без загрузки TWDR	x 0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
			x 0	1	1	Принимается байт данных и возвращается ПОДТВерждение
\$78	Потеряна арбитрация во время передачи ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ как ведущего, принят адрес общего вызова, возвращено ПОДТВ	Действия без загрузки TWDR или действия без загрузки TWDR	x 0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
			x 0	1	1	Принимается байт данных и возвращается ПОДТВерждение
\$80	Предварительная адресация собственным ПОДЧИН_АДР+ЗАПИСЬ; принятые данные; возвращено ПОДТВерждение	Чтение байта данных или чтение байта данных	x 0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
			x 0	1	1	Принимается байт данных и возвращается ПОДТВерждение
\$88	Предварительная адресация собственным ПОДЧИН_АДР+ЗАПИСЬ; принятые данные; возвращено НЕТ ПОДТВерждение	Чтение байта данных или чтение байта данных или чтение байта данных или чтение байта данных	0 0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова
			0 0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"
			1 0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова, передается условие СТАРТ после освобождения шины
			1 0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"; передается условие СТАРТ после освобождения шины
\$90	Предварительная адресация адресом общего вызова; принятые данные; возвращено ПОДТВерждение	Чтение байта данных или чтение байта данных	x 0	1	0	Принимается байт данных и возвращается НЕТ ПОДТВерждение
			x 0	1	1	Принимается байт данных и возвращается ПОДТВерждение

Окончание таблицы 3.76

\$98	Предварительная адресация адресом общего вызова; принятые данные; возвращено НЕТ ПОДТВЕРЖДЕНИЯ	Чтение байта данных или чтение байта данных или чтение байта данных или чтение байта данных	0	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова
			0	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"
			1	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова; передается условие СТАРТ после освобождения шины
			1	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"; передается условие СТАРТ после освобождения шины
\$A0	Принято условие СТОП или повторный СТАРТ во время подчиненной адресации	Нет действий	0	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова
			0	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"
			1	0	1	0	Подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"
			1	0	1	1	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова; передается условие СТАРТ после освобождения шины
			1	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"; передается условие СТАРТ после освобождения шины

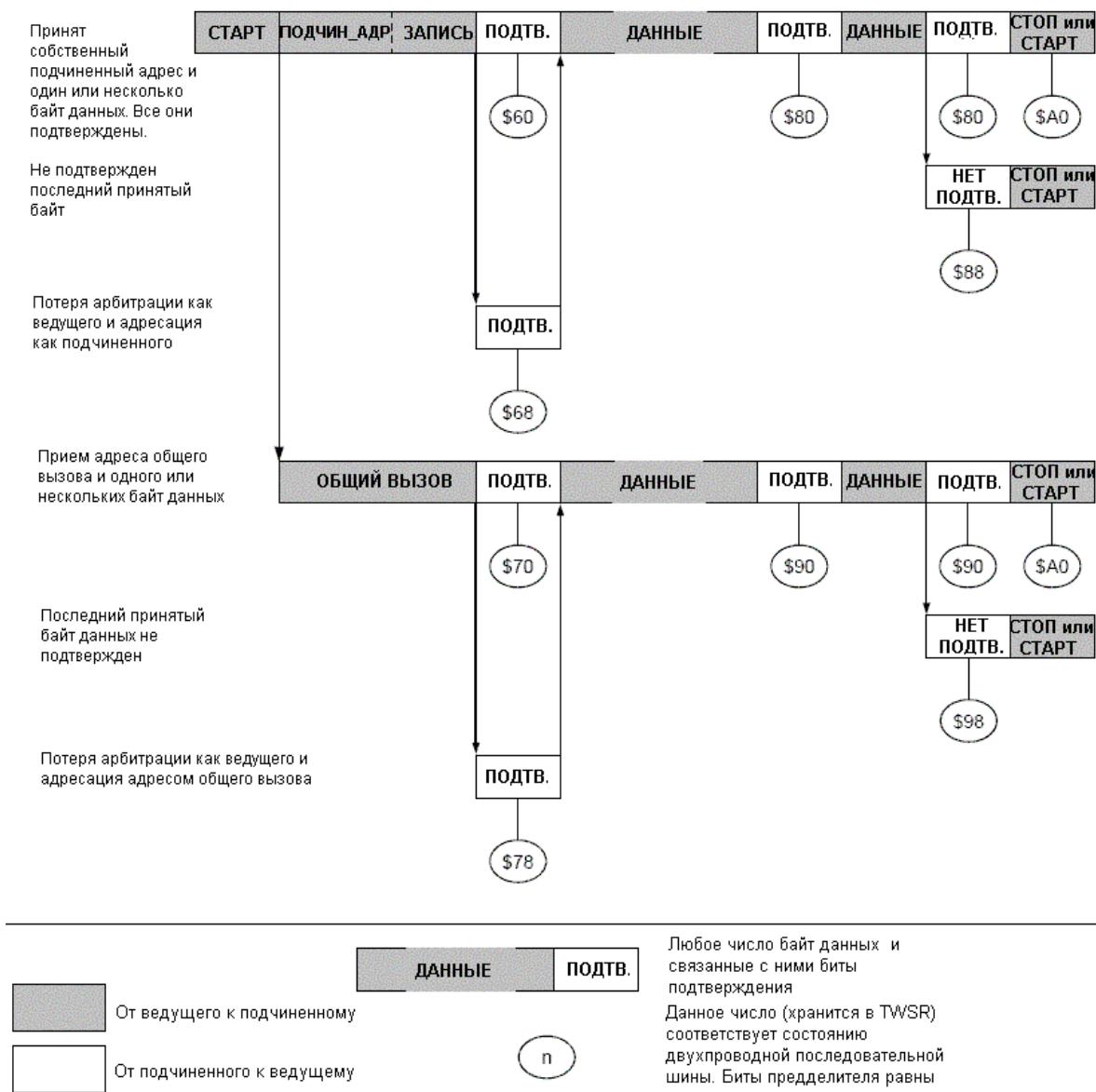


Рисунок 3.89 - Форматы и состояния в режиме подчиненного приемника

Режим подчиненного передатчика

В режиме подчиненного передатчика выполняется передача нескольких байт данных ведущему приемнику (см. рисунок 3.90). Во всех кодах состояния, приведенных в этом разделе, не учитываются биты предделителя и они равны нулю.

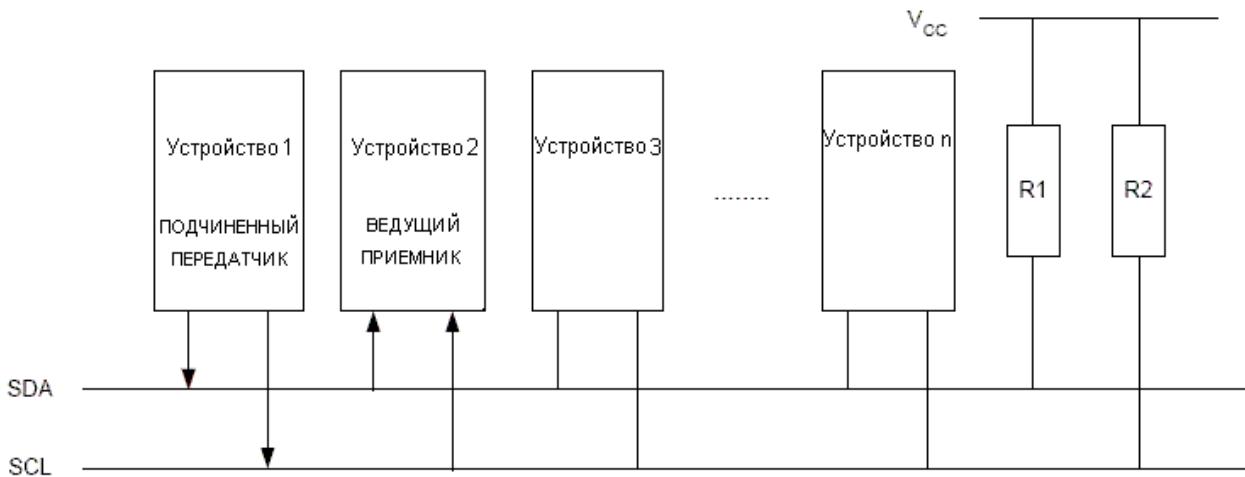


Рисунок 3.90 - Передача данных в режиме подчиненного передатчика

Для ввода режима подчиненного передатчика необходимо инициализировать регистры TWAR и TWCR следующим образом:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Значение	Собственный подчиненный адрес устройства							

Старшие семь разрядов образуют адрес. С помощью него определяется к какому двухпроводному интерфейсу адресуется ведущий. Если в младшем разряде записана лог. 1, то TWI будет отвечать на адрес общего вызова (\$00). В противном случае он игнорирует адрес общего вызова.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	0	1	0	0	0	1	0	x

Для разрешения работы TWI необходимо записать логическую 1 в TWEN. Для разрешения подтверждения собственного подчиненного адреса или адреса общего вызова записывается 1 в TWEA. Битам TWSTA и TWSTO необходимо присвоить нулевое значение.

После инициализации TWAR и TWCR схема TWI ожидает получения собственного подчиненного адреса (или, если разрешен, адреса общего вызова), а вслед за ним - бита направления данных. Если бит направления равен "1" (чтение), то TWI переходит в режим "подчиненный передатчик", иначе вводится режим "подчиненный приемник". После приема собственно подчиненного адреса и бита записи устанавливается флаг TWINT, а в регистр TWSR помещается код состояния. Код состояния позволяет определить, какие программные действия необходимо выполнить. Подробности по использованию кодов состояний представлены в таблице 3.77. Режим "подчиненный передатчик" также вводится, если теряется арбитрация, когда TWI находился в режиме ведущего (см. состояние \$B0). Если бит TWEA обнулить во время передачи, то TWI передаст последний байт. Вводится состояние \$C0 или состояние \$C8 в зависимости от того принял ПОДТВ. или НЕТ ПОДТВ. ведущий приемник за последним байтом. TWI переходит в безадресный подчиненный режим и далее игнорирует ведущего, если тот продолжает передачу. Таким образом, ведущий приемник принимает все "1" как последовательные данные. Состояние \$C8 вводится, если ведущий требует передачи дополнительных байт данных (путем передачи ПОДТВ.), даже если подчиненный передал последний байт (TWEA равен нулю и ожидается прием НЕТ ПОДТВ. от ведущего).

Пока TWEA равен нулю, TWI не отвечает на собственный подчиненный адрес. Однако последовательная шина остается под контролем и функция распознавания адреса может быть активизирована в любой момент путем установки бита TWEA. Это означает, что бит TWEA можно использовать для временной изоляции TWI от двухпроводной последовательной шины.

Во всех режимах сна, кроме холостого хода (Idle), синхронизация TWI отключается. Если бит TWEA установлен, то интерфейс останется способным подтверждать прием своего собственного подчиненного адреса или адреса общего вызова за счет использования сигнала синхронизации шины в качестве тактового источника. При обнаружении запроса микроконтроллер выходит из режима сна, при этом линия SCL остается на низком уровне в процессе пробуждения и до сброса флага TWINT (записью в него "1"). Далее выполняется прием данных обычным способом при обычной системной синхронизации. Необходимо учесть, что если для микроконтроллера выбрано большое время запуска, то линия SCL может оказаться длительно в низком состоянии и заблокировать другой обмен информацией.

После пробуждения регистр данных TWDR не отражает последний байт, присутствовавший на шине во время выхода из указанных выше режимов сна.

Таблица 3.77 - Коды состояния для режима подчиненный передатчик

Код состояния (TWSR), биты предделителя равны 0	Состояние двухпроводной последовательной шины и схемы двухпроводного последовательного интерфейса	Программные действия					Следующее действие, выполняемое схемой ТМ	
		В/из TWDR		В TWCR				
		STA	STO	TWINT	TWEA			
\$A8	Принимается собственный ПОДЧИН_АДР+ЧТЕНИЕ; возвращено ПОДТВ.	Загрузка байта данных или загрузка байта данных	x	0	1	0	Передается последний байт данных и должно быть принято НЕТ ПОДТВ	
\$B0	Потеряна арбитрация во время передачи ПОДЧИН_АДР+ЧТЕНИЕ/ ЗАПИСЬ как ведущего, принят собственный ПОДЧИН_АДР+ЧТЕНИЕ, возвращено ПОДТВ	Загрузка байта данных или загрузка байта данных	x	0	1	0	Передается последний байт данных и должно быть принято НЕТ ПОДТВ	
\$B8	Передан байт данных из TWDR; принято ПОДТВерждение	Загрузка байта данных или загрузка байта данных	x	0	1	0	Передается последний байт данных и должно быть принято НЕТ ПОДТВ	
\$C0	Передан байт данных из TWDR; принято НЕТ ПОДТВерждение	Действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR	0	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова	
			0	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"	
			1	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова; передается условие СТАРТ после освобождения шины	
			1	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"; передается условие СТАРТ после освобождения шины	
\$C8	Передан последний байт данных из TWDR (TWEA = "0"); принято ПОДТВерждение	Действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR или действия без загрузки TWDR	0	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова	
			0	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"	
			1	0	1	0	Переключение на безадресный подчиненный режим; не распознается собственный ПОДЧИН_АДР или адрес общего вызова; передается условие СТАРТ после освобождения шины	
			1	0	1	1	Переключение на безадресный подчиненный режим; собственный ПОДЧИН_АДР распознается; адрес общего вызова распознается, если TWGCE = "1"; передается условие СТАРТ после освобождения шины	

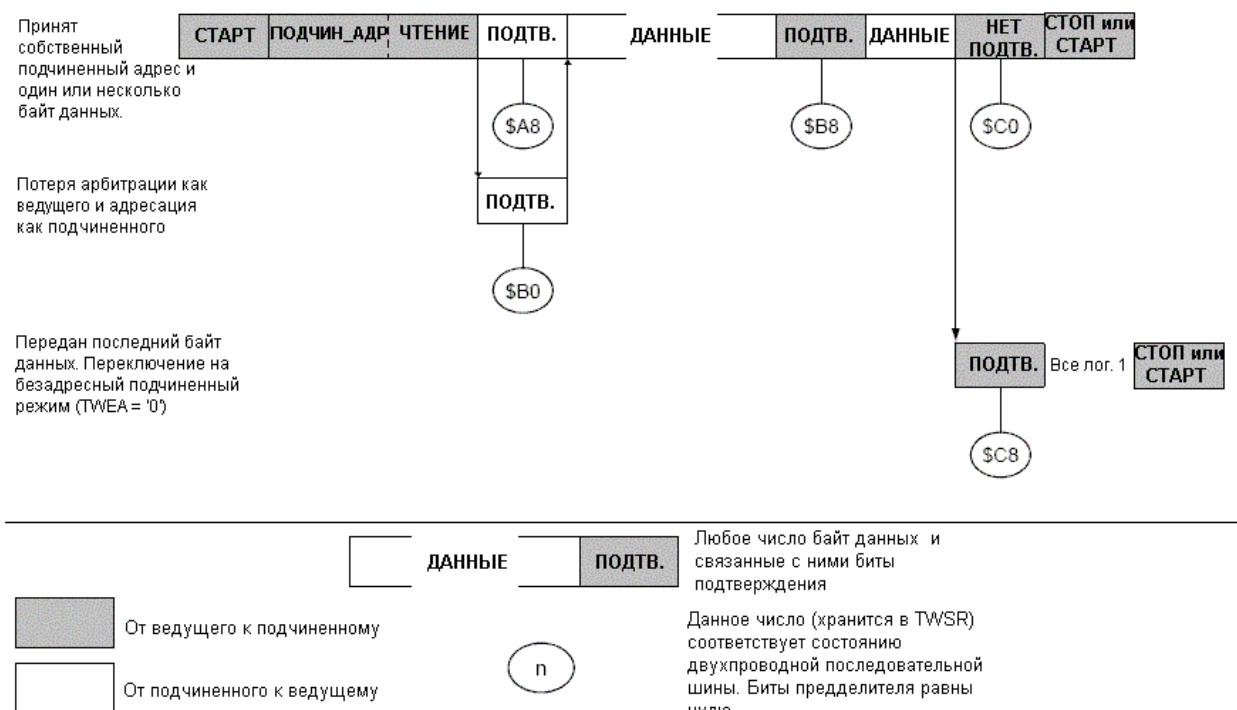


Рисунок 3.91 - Форматы и состояния в режиме подчиненного передатчика

Прочие состояния

Имеются несколько кодов состояний, которые отличаются от упомянутых выше (см. таблицу 3.77). Состояние \$F8 индицирует, что нет доступной информации, т. к. не установлен флаг TWINT. Это может произойти между другими состояниями и когда TWI не участвует в последовательной передаче данных.

Состояние \$00 индицирует, что во время последовательной передачи данных нашине возникла ошибка. Ошибка возникает, если условия СТАРТ или СТОП возникают в неверной позиции формата посылки. Примеры таких неточных позиций могут существовать во время передачи адресного байта, байта данных и бита подтверждения. После возникновения ошибки устанавливается флаг TWINT. Для выхода из состояния ошибки необходимо установить флаг TWSTO и сбросить TWINT путем записи в него "1". Это приводит к переводу TWI в безадресный режим и к сбросу флага TWSTO (другие биты в TWCR не затрагиваются). Линии SDA и SCL освобождаются и условие СТОП не передается.

Сочетание нескольких режимов

В некоторых случаях сочетаются несколько режимов TWI для обеспечения желаемого действия. В качестве примера рассмотрим чтение данных из последовательного ЭСППЗУ. Обычно, такой сеанс связи организуется в такой последовательности:

- инициируется сеанс связи;
- в ЭСППЗУ отправляется инструкция с указанием адреса считываемой ячейки;
- выполняется чтение;
- завершается сеанс связи.

Следует обратить внимание, что данные передаются как от ведущего к подчиненному, так и обратно, от подчиненного к ведущему. Ведущий инструктирует подчиненного какую ячейку он желает считать, для чего используется режим "ведущий передатчик". В дальнейшем данные передаются подчиненным, что требует использования режима "ведущий приемник". Следовательно, направление передачи данных изменяется. Ведущий должен сохранить управление надшиной на каждом из этапов, а каждый из шагов

должен быть выполнен как элементарное действие. Если данный принцип нарушить в многомастерной системе, то другой ведущий может обратиться к ЭСППЗУ на шагах 2 и 3 и изменить указатель данных. Это приведет к тому, что ведущий считывает данные из ячейки с неверным адресом. Таким образом, направление передачи данных необходимо изменять только передачей условия ПОВТОРНЫЙ СТАРТ между передачей адресного байта и приемом байта. Передачей ПОВТОРНОГО СТАРТА мастер сохранит свое "господство" на шине. На рисунке 3.92 представлена структура потока данной передачи.



Рисунок 3.92 - Сочетание нескольких режимов TWI для обмена данными с последовательным ЭСППЗУ

Если к одной шине подключено несколько ведущих, то передача может быть инициирована одновременно одним или несколькими из них. Стандарт TWI гарантирует, что в таких ситуациях разрешается передача только одному ведущему, при этом не будет происходить потеря данных. Пример арбитрирования такой ситуации представлен ниже, где два ведущих пытаются передавать данные подчиненному приемнику.

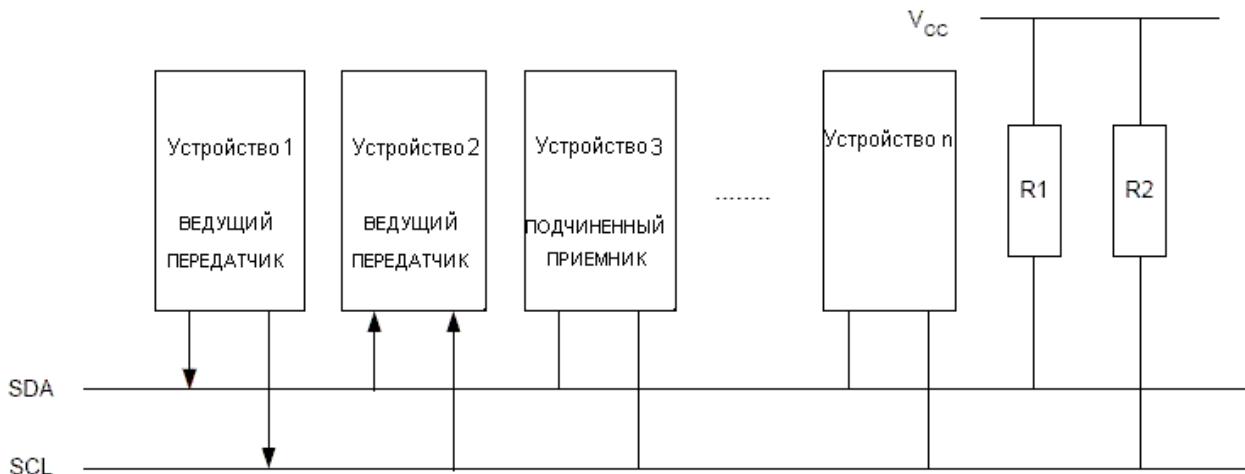


Рисунок 3.93 - Пример арбитризации

Ниже приведены несколько различных сценариев, возникающих в процессе арбитрирования:

- Два или более ведущих выполняют идентичную связь с одним и тем же подчиненным. В этом случае ни один подчиненный и ни один из ведущих не узнает об этой конфликтной ситуации.
- Два или более ведущих обращаются к тому же подчиненному с различными данными или битом направления данных. В этом случае возникает арбитрирование во время передачи бита ЧТЕНИЕ/ЗАПИСЬ или же бит данных. Одни ведущие выводят на SDA логическую 1, а другие выводят логический 0. Последние теряют арбитрацию. Ведущие, которые проиграли арбитрирование, переходят в безадресный подчиненный режим или ожидают освобождения шины и затем передают новое условие СТАРТ, что зависит от программных действий.

– Два или более ведущих обращаются к различным подчиненным. В этом случае арбитрирование возникает во время передачи бит ПОДЧИН_АДР. Одни ведущие выводят на SDA логическую 1, а другие выводят логический 0. Последние теряют арбитрацию. Ведущие, которые проиграли арбитрирование во время передачи ПОДЧИН_АДР, переходят в подчиненный режим для проверки: не обращается ли к ним выигравший арбитраж ведущий. Если такая адресация действительно выполняется, то они переключаются в режим "подчиненный приемник" или "подчиненный передатчик" в зависимости от значения бита ЧТЕНИЕ/ЗАПИСЬ. Если адресации не было, то они перейдут в безадресный подчиненный режим или будут ожидать освобождения шины и после этого передадут новое условие СТАРТ (задается программно).

Данные действия обобщены на рисунке 3.94. Возможные коды состояний представлены внутри окружностей.

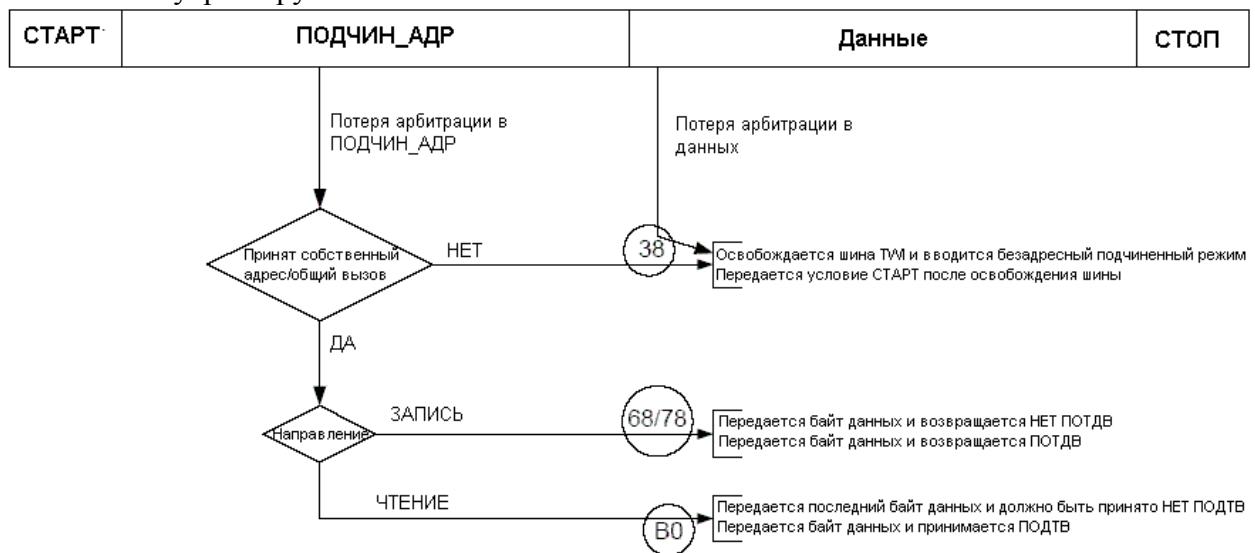


Рисунок 3.94 - Возможные коды состояний при потере арбитрации

3.17 Аналоговый компаратор

Аналоговый компаратор сравнивает уровни напряжений на неинвертирующем входе AIN0 и инвертирующем входе AIN1. Если напряжение на неинвертирующем входе AIN0 превышает напряжение на инвертирующем входе AIN1, то выход аналогового компаратора ACO принимает единичное состояние. Выход компаратора может быть настроен для использования в качестве источника входного сигнала для схемы захвата фронтов таймера-счетчика 1. Кроме того, компаратор может генерировать собственный запрос на обработку прерывания. Пользователь может выбрать несколько событий, по которым возникает прерывание: нарастающий, падающий фронт на выходе компаратора или любое его изменение. Функциональная схема компаратора и связанной с ним логики представлена на рисунке 3.95. Параметры работы компаратора представлены в таблице 3.79.1.

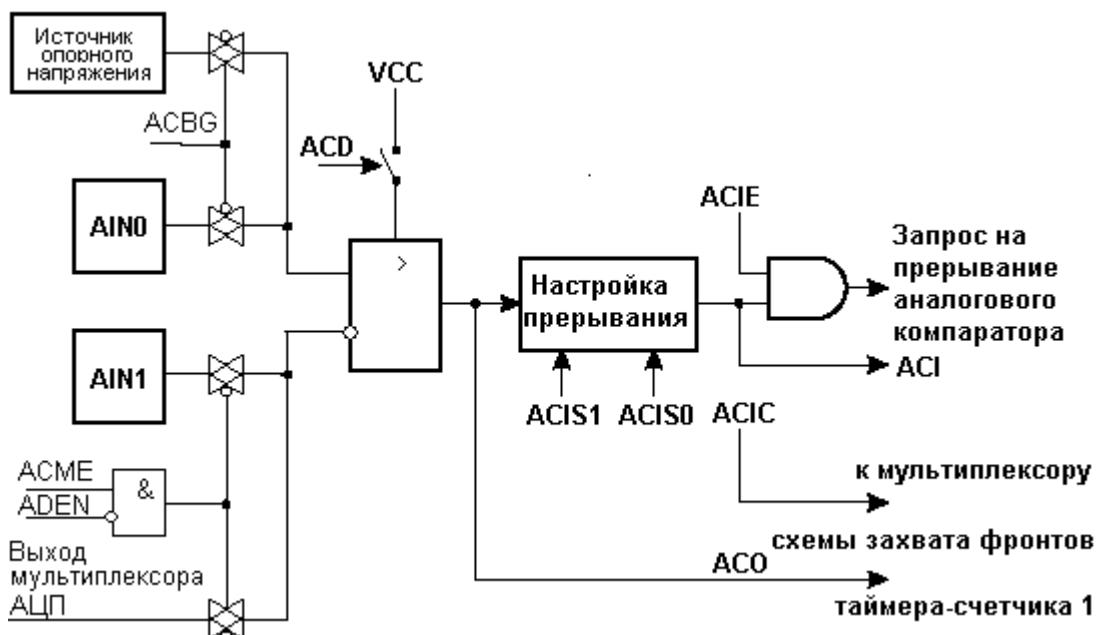


Рисунок 3.95 – Функциональная схема аналогового компаратора

Регистр специальных функций ввода-вывода – SFIOR

Разряд	7	6	5	4	3	2	1	0	SFIOR
	TSM	-	-	-	ACME	PUD	PSR0	PSR321	
Чтение/ЗаписьЧт./Зп.	Чт.	Чт.	Чт.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Начальное знач. 0	0	0	0	0	0	0	0	0	

Разряд 3: ACME - Выбор мультиплексора на входе аналогового компаратора

Если выключен аналогово-цифровой преобразователь (ADEN=0 в регистре ADCSRA) и в данный разряд записана логическая 1, то к инвертирующему входу аналогового компаратора подключен выход аналогового мультиплексора АЦП. Запись в данный разряд логического 0 приведет к подключению инвертирующего входа аналогового компаратора к выводу микроконтроллера AIN1.

Регистр состояния и управления аналогового компаратора ACSR:

Разряд	7	6	5	4	3	2	1	0	ACSR
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	
Чтение/Запись	Чт./Зп.	Чт./Зп.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Начальное знач.	0	0	x	0	0	0	0	0	

Разряд 7: ACD - Отключение аналогового компаратора

Запись в данный разряд логической 1 приводит к снятию питания с аналогового компаратора. Данный разряд можно устанавливать в любой момент при необходимости отключения аналогового компаратора. Его использование позволяет снизить энергопотребление в активном режиме и режиме холостого хода. Перед изменением бита ACD необходимо отключить прерывание по аналоговому компаратору путем сброса бита ACIE в регистре ACSR. В противном случае может возникнуть прерывание после изменения значения данного бита.

Разряд 6: ACBG - Подключение источника опорного напряжения к аналоговому компаратору

После установки данного бита к неинвертирующему входу компаратора подключается источник опорного напряжения. После сброса данного разряда неинвертирующий вход компаратора связан с выводом AIN0 микроконтроллера.

Разряд 5: ACO - Выход аналогового компаратора

Данный бит выхода аналогового компаратора связан непосредственно с выходом ACO через цепь синхронизации. Синхронизация реализована как временная задержка на 1 – 2 машинных цикла.

Разряд 4: ACI - Флаг прерывания аналогового компаратора

Данный разряд устанавливается аппаратно, при возникновении события в соответствии с установками бит ACIS1 и ACIS0. Запрос на обработку прерывания аналогового компаратора выполняется, если установлены биты ACIE и I в регистре SREG. ACI сбрасывается аппаратно при переходе на соответствующий вектор обработки прерывания. Альтернативно бит ACI можно сбросить программно путем записи логической 1 в данный флаг.

Разряд 3: ACIE - Разрешение прерывания аналогового компаратора

Если в данный разряд записана логическая 1 и установлен бит I в регистре статуса, то прерывание по аналоговому компаратору активизируется. Запись в данный разряд логического 0 приводит к отключению данного прерывания.

Разряд 2: ACIC - Подключение аналогового компаратора к схеме захвата фронтов

Установка данного разряда приводит к разрешению совместной работы схемы захвата фронтов таймера-счетчика 1 и аналогового компаратора. В этом случае выход аналогового компаратора непосредственно подключен к входному каскаду схемы захвата фронтов, позволяя к компаратору добавить функции подавления шумов и настройки фронтов прерывания по захвату фронта таймером-счетчиком 1. После записи в данный

разряд логического 0 связь между аналоговым компаратором и схемой захвата фронтов разрывается. Для активизации прерывания схемы захвата фронтов таймера-счетчика 1 по срабатыванию аналогового компаратора необходимо установить бит TICIE1 в регистре маски прерывания таймера (TIMSK).

Разряды 1, 0: ACIS1, ACIS0 - Выбор события прерывания аналогового компаратора

Данные разряды определяют какое событие приводит к генерации запроса на прерывание аналогового компаратора. Варианты установок данных разрядов и их назначение представлены в таблице 3.78.

Таблица 3.78 - Установки разрядов ACIS1, ACIS0

ACIS1	ACIS0	Событие
0	0	Прерывание по любому изменению на выходе компаратора
0	1	Зарезервировано
1	0	Прерывание по падающему фронту на выходе компаратора
1	1	Прерывание по нарастающему фронту на выходе компаратора

Перед изменением бит ACIS1/ACIS0 необходимо отключить прерывание по аналоговому компаратору путем сброса бита разрешения прерывания в регистре ACSR. В противном случае может возникнуть прерывание при изменении значений данных бит.

Мультиплексированный вход аналогового компаратора

Имеется возможность использовать выводы ADC7...0 в качестве неинвертирующих входов аналогового компаратора. Для организации такого ввода используется мультиплексор АЦП и, следовательно, в этом случае АЦП должен быть отключен. Если установлен бит разрешения подключения мультиплексора к аналоговому компаратору (бит ACME в SFIOR) и выключен АЦП (ADEN=0 в регистре ADCSRA), то состояние разрядов MUX2...0 регистра ADMUX определяют какой вывод микроконтроллера подключен к неинвертирующему входу аналогового компаратора (см. таблицу 3.79). Если ACME сброшен или установлен ADEN, то в качестве неинвертирующего входа аналогового компаратора используется вывод микроконтроллера AIN1.

Таблица 3.79 – Мультиплексированный вход аналогового компаратора

ACME	ADEN	MUX2...0	Неинвертирующий вход аналогового компаратора
0	x	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

Таблица 3.79.1 – Параметры работы аналогового компаратора

Наименование параметра, единица измерения, режим измерения	Буквен- ное обоз- значение	Норма		
		не менее	тип	не более
1	2	3	4	5
1 Напряжение смещения по входу аналогового компаратора, мВ; $U_{\#VCC} = U_{\eta VCC} = 5,0$ В; $V_{IN} = U_{\eta VCC}/2$	V_{ACIO}			40
2 Токи утечки по входу аналогового компаратора, нА $U_{\#VCC} = U_{\eta VCC} = 5,0$ В; $V_{IN} = U_{\eta VCC}/2$	I_{ACLK}	-50		50
3 Задержка распространения аналогового компаратора, нс. $U_{\#VCC} = U_{\eta VCC} = 4,0$ В	t_{ACID}		500	

3.18 Аналого-цифровой преобразователь

Отличительные особенности:

- 10-разрядное разрешение;
- интегральная нелинейность 0,5 младший разряд;
- абсолютная погрешность ± 2 младший разряд;
- время преобразования (65 – 260) мкс;
- частота преобразования до 15 тысяч преобразований в секунду при максимальном разрешении;
- 8 мультиплексированных однополярных входов;
- 7 дифференциальных входных каналов;
- 2 дифференциальных входных канала с optionalным усилением на 10 и 200;
- представление результата с левосторонним или правосторонним выравниванием в 16-разрядном слове;
- диапазон входного напряжения АЦП 0... $U_{\eta VCC}$;
- выборочный внутренний источник опорного напряжения на 2,56 В;
- режимы одиночного преобразования и автоматического перезапуска;
- прерывание по завершении преобразования АЦП;
- механизм подавления шумов в режиме сна.

ИМС содержит 10-разрядный АЦП последовательного приближения. АЦП связан с 8-канальным аналоговым мультиплексором, 8 однополярных входов которого связаны с линиями порта F. Общий вывод входных сигналов должен иметь потенциал 0 В (т. е. связан с GND). АЦП также поддерживает ввод 16 дифференциальных напряжений. Два дифференциальных входа (ADC1, ADC0 и ADC3, ADC2) содержат каскад со ступенчатым программируемым усилением: 0 дБ (1x), 20 дБ (10x) или 46 дБ (200x). Семь дифференциальных аналоговых каналов используют общий инвертирующий вход (ADC1), а все остальные входы АЦП выполняют функцию неинвертирующих входов. Если выбрано усиление 1x или 10x, то можно ожидать 8-разрядное разрешение, а если 200x, то 7-разрядное.

АЦП содержит УВХ (устройство выборки-хранения), которое поддерживает на постоянном уровне напряжение на входе АЦП во время преобразования. Функциональная схема АЦП показана на рисунке 3.96.

АЦП имеет отдельный вывод питания $\cap VCC$ (анalogовое питание). $U_{\eta VCC}$ не должен отличаться более чем на $\pm 0,3$ В от $U_{\#VCC}$.

В качестве внутреннего опорного напряжения может выступать напряжение от внутреннего ИОНа на 2,56 В или напряжение $U_{\eta VCC}$. Если требуется использование внешнего ИОН, то он должен быть подключен к выводу AREF с подключением к этому выводу блокировочного конденсатора для улучшения шумовых характеристик.

Параметры работы АЦП представлены в таблице 3.79.2.

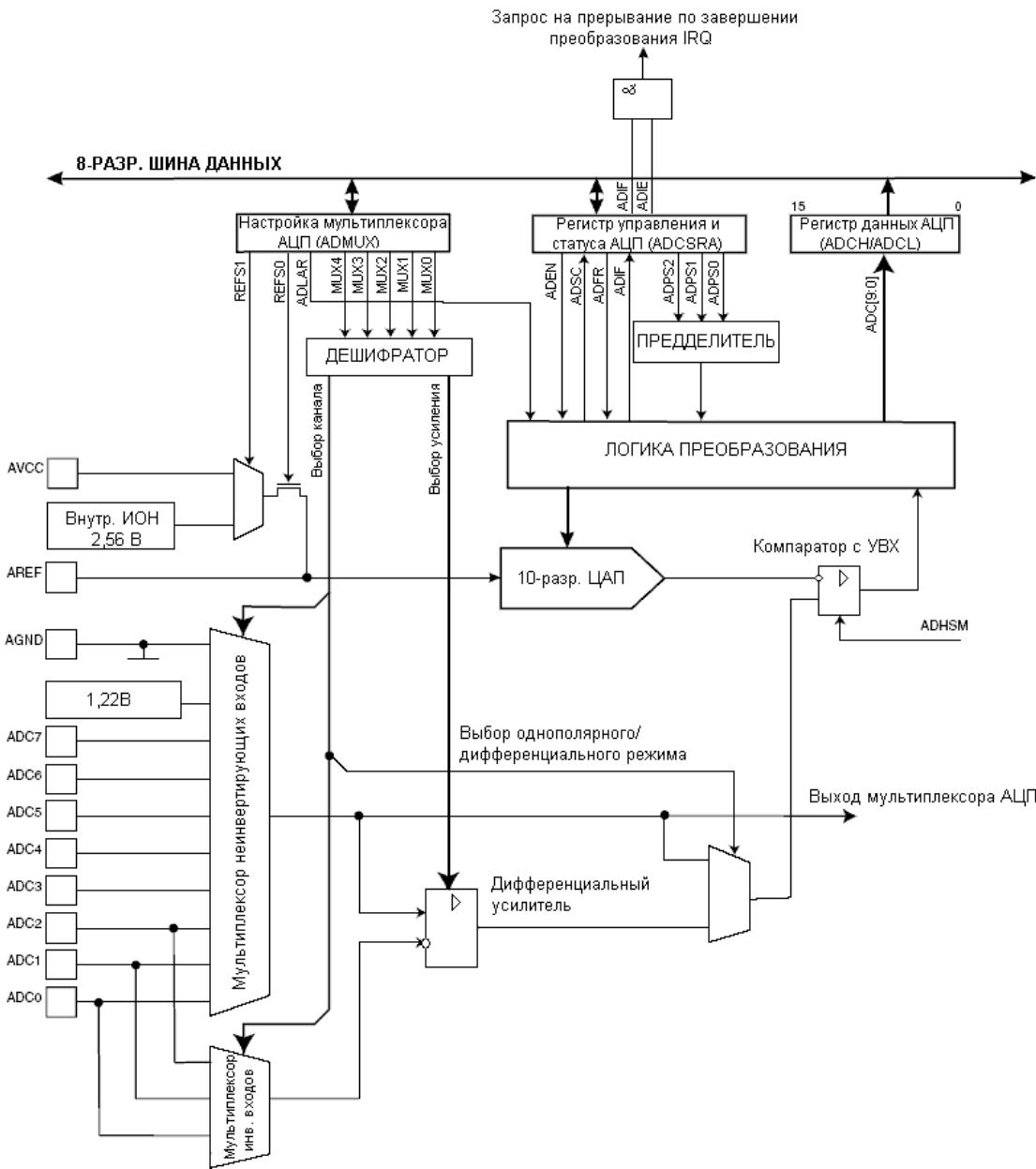


Рисунок 3.96 – Функциональная схема АЦП

3.18.1 Принцип действия АЦП

АЦП преобразовывает входное аналоговое напряжение в 10-разрядный код методом последовательных приближений. Минимальное значение соответствует уровню GND, а максимальное уровню AREF минус 1000000 разрядов К выводу AREF опционально может быть подключено напряжение U_{NVC} или внутренний ИОН на 1,22 В путем записи соответствующих значений в биты REFSn в регистр ADMUX. Несмотря на то, что ИОН на 2,56 В находится внутри микроконтроллера, к его выходу может быть подключен блокировочный конденсатор для снижения чувствительности к шумам, т. к. он связан с выводом AREF.

Канал аналогового ввода и каскад дифференциального усиления выбираются путем записи бит MUX в регистр ADMUX. В качестве однополярного аналогового входа

АЦП может быть выбран один из входов ADC0...ADC7, а также GND и выход фиксированного источника опорного напряжения 1,22 В. В режиме дифференциального ввода предусмотрена возможность выбора инвертирующих и неинвертирующих входов к дифференциальному усилителю.

Если выбран дифференциальный режим аналогового ввода, то дифференциальный усилитель будет усиливать разность напряжений между выбранной парой входов на заданный коэффициент усиления. Усиленное таким образом значение поступает на аналоговый вход АЦП. Если выбирается однополярный режим аналогового ввода, то каскад усиления пропускается.

Работа АЦП разрешается путем установки бита ADEN в ADCSRA. Выбор опорного источника и канала преобразования не возможно выполнить до установки ADEN. Если ADEN = 0, то АЦП не потребляет ток, поэтому при переводе в экономичные режимы сна рекомендуется предварительно отключить АЦП.

АЦП генерирует 10-разрядный результат, который помещается в пару регистров данных АЦП ADCH и ADCL. По умолчанию результат преобразования размещается в младших 10-ти разрядах 16-разрядного слова (выравнивание справа), но может быть опционально размещен в старших 10-ти разрядах (выравнивание слева) путем установки бита ADLAR в регистре ADMUX.

Практическая полезность представления результата с выравниванием слева существует, когда достаточно 8-разрядное разрешение, т. к. в этом случае необходимо считать только регистр ADCH. В другом же случае необходимо первым считать содержимое регистра ADCL, а затем ADCH, чем гарантируется, что оба байта являются результатом одного и того же преобразования. Как только выполнено чтение ADCL, блокируется доступ к регистрам данных со стороны АЦП. Это означает, что если считан ADCL и преобразование завершается перед чтением регистра ADCH, то ни один из регистров не может модифицироваться и результат преобразования теряется. После чтения ADCH доступ к регистрам ADCH и ADCL со стороны АЦП снова разрешается.

АЦП генерирует собственный запрос на прерывание по завершении преобразования. Если между чтением регистров ADCH и ADCL запрещен доступ к данным для АЦП, то прерывание возникнет, даже если результат преобразования будет потерян.

Запуск преобразования

Одиночное преобразование запускается путем записи логической 1 в бит запуска преобразования АЦП ADSC. Данный бит остается в высоком состоянии в процессе преобразования и сбрасывается по завершении преобразования. Если в процессе преобразования переключается канал аналогового ввода, то АЦП автоматически завершит текущее преобразование прежде, чем переключит канал.

В режиме автоматического перезапуска АЦП непрерывно оцифровывает аналоговый сигнал и обновляет регистр данных АЦП. Данный режим задается путем записи логической 1 в бит ADFR регистра ADCSRA. Первое преобразование инициируется путем записи логической 1 в бит ADSC регистра ADCSRA. В данном режиме АЦП выполняет последовательные преобразования, независимо от того сбрасывается флаг прерывания АЦП ADIF или нет.

3.18.2 Предделитель АЦП и временная диаграмма преобразования

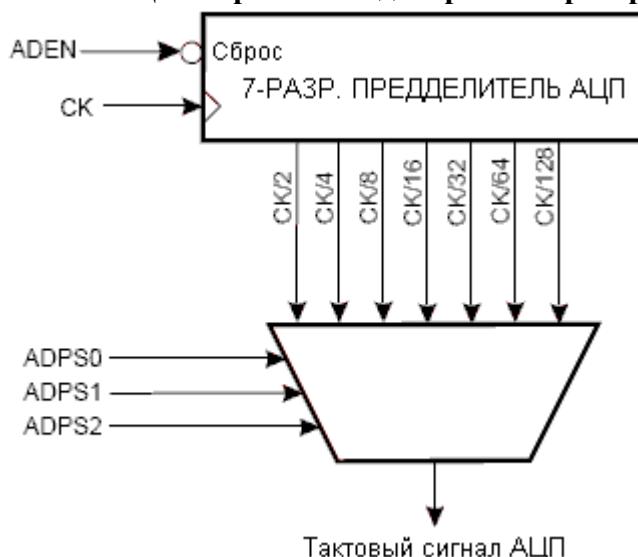


Рисунок 3.97 – Предделитель АЦП

Если требуется максимальная разрешающая способность (10 разрядов), то частота на входе схемы последовательного приближения должна быть в диапазоне 50...200 кГц. Если достаточно разрешение менее 10 разрядов, но требуется более высокая частота преобразования, то частота на входе АЦП может быть установлена выше 200 кГц.

Модуль АЦП содержит предделитель, который формирует производные частоты выше 100 кГц по отношению к частоте синхронизации ЦПУ. Коэффициент деления устанавливается с помощью бит ADPS в регистре ADCSRA. Предделитель начинает счет с момента включения АЦП установкой бита ADEN в регистре ADCSRA. Предделитель работает пока бит ADEN = 1 и сброшен, когда ADEN=0.

Если инициируется однополярное преобразование установкой бита ADSC в регистре ADCSRA, то преобразование начинается со следующего нарастающего фронта тактового сигнала АЦП.

Нормальное преобразование требует 13 тактов синхронизации АЦП. Первое преобразование после включения АЦП (установка ADEN в ADCSRA) требует 25 тактов синхронизации АЦП за счет необходимости инициализации аналоговой схемы.

После начала нормального преобразования на выборку-хранение затрачивается 1,5 такта синхронизации АЦП, а после начала первого преобразования – 13,5 тактов. По завершении преобразования результат помещается в регистры данных АЦП и устанавливается флаг ADIF. В режиме одиночного преобразования одновременно сбрасывается бит ADSC. Программно бит ADSC может быть снова установлен и новое преобразование будет инициировано первым нарастающим фронтом тактового сигнала АЦП.

В режиме автоматического перезапуска новое преобразование начинается сразу по завершении предыдущего, при этом ADSC остается в высоком состоянии.

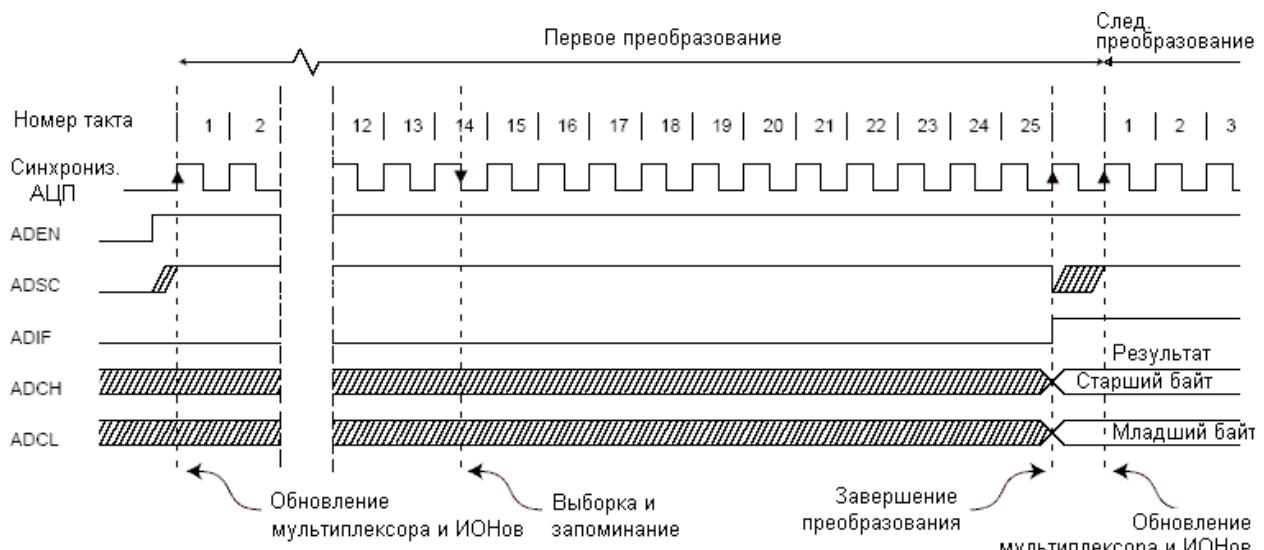


Рисунок 3.98 – Временная диаграмма работы АЦП при первом преобразовании в режиме одиночного преобразования

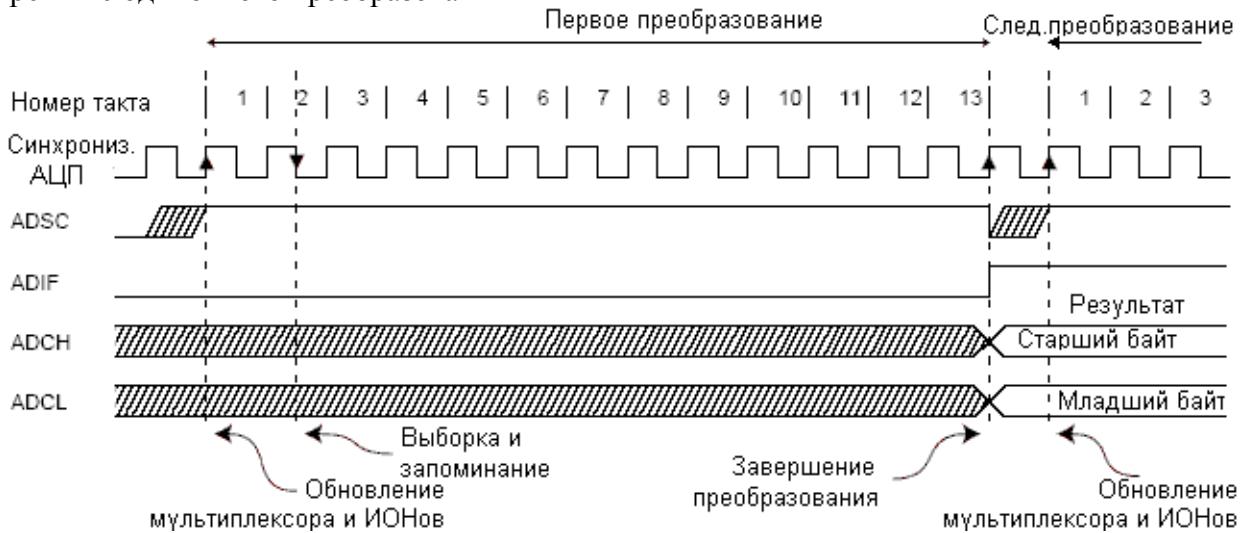


Рисунок 3.99 – Временная диаграмма работы АЦП в режиме одиночного преобразования

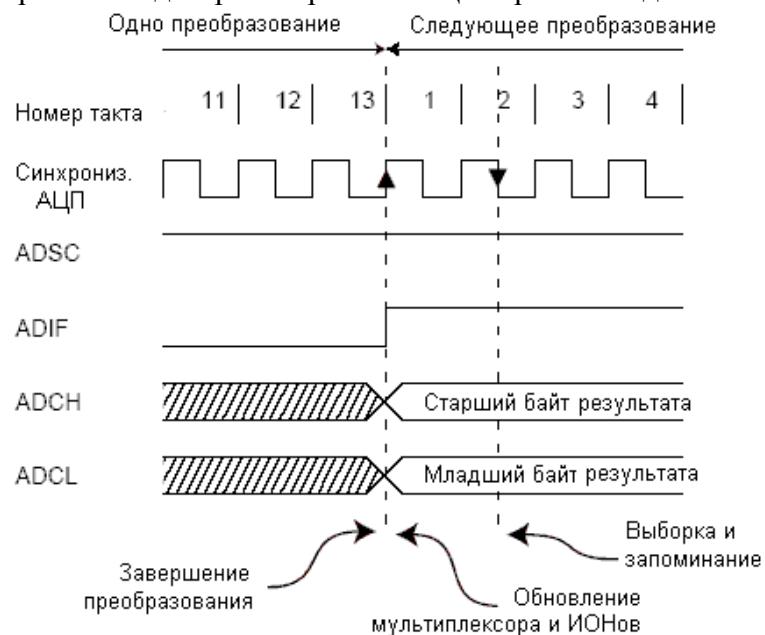


Рисунок 3.100 – Временная диаграмма работы АЦП в режиме автоматического перезапуска

3.18.3 Каналы дифференциального усиления

Если используются каналы дифференциального усиления, то необходимо принять во внимание некоторые особенности.

Дифференциальные преобразования синхронизированы по отношению к внутренней синхронизации СКАЦП2, частота которого равна половине частоты синхронизации АЦП. Данная синхронизация выполняется автоматически интерфейсом АЦП таким образом, чтобы выборка-хранение инициировалась определенным фронтом СКАЦП2. Если преобразование (все одиночные преобразования и первое преобразование в режиме автоматического перезапуска) инициировалось пользователем, когда СКАЦП2 находился в низком логическом состоянии, то его длительность будет эквивалента однополярному преобразованию (13 тактов синхронизации АЦП). Если преобразование инициируется пользователем, когда СКАЦП2 равен логической 1, оно будет длиться 14 тактов синхронизации АЦП вследствие работы механизма синхронизации. В режиме автоматического перезапуска новое преобразование инициируется сразу по завершении предыдущего, а т. к. в этот момент СКАЦП2 равен логической 1, то все преобразования, которые были автоматически перезапущены (т. е. все, кроме первого), будут длиться 14 тактов синхронизации АЦП. Усилительный каскад оптимизирован под частотный диапазон до 4 кГц для любых коэффициентов усиления. Усиление сигналов более высоких частот будет нелинейным. Поэтому, если входной сигнал содержит частотные составляющие выше частотного диапазона усилительного каскада, то необходимо установить внешний фильтр низких частот. Следует обратить внимание, что частота синхронизации АЦП не связана с ограничением по частотному диапазону усилительного каскада. Например, период синхронизации АЦП может быть 6 мкс, при котором частота преобразования канала равна 12 тысячам преобразований в секунду, независимо от частотного диапазона этого канала.

Изменение канала или выбор опорного источника

Биты MUXn и REFS1:0 в регистре ADMUX поддерживают одноступенчатую буферизацию через временный регистр. Этим гарантируется, что новые настройки канала преобразования и опорного источника вступят в силу в безопасный момент для преобразования. До начала преобразования любые изменения канала и опорного источника вступают в силу сразу после их модификации. Как только начинается процесс преобразования, доступ к изменению канала и опорного источника блокируется, чем гарантируется достаточность времени на преобразование для АЦП. Непрерывность модификации возвращается на последнем такте АЦП перед завершением преобразования (перед установкой флага ADIF в регистре ADCSRA). Следует обратить внимание, что преобразование начинается следующим нарастающим фронтом тактового сигнала АЦП после записи ADSC. Таким образом, пользователю не рекомендуется записывать новое значение канала или опорного источника в ADMUX до 1-го такта синхронизации АЦП после записи ADSC.

Особые меры необходимо предпринять при изменении дифференциального канала. Как только осуществлен выбор дифференциального канала, усилительному каскаду требуется 125 мкс для стабилизации нового значения. Следовательно в течение первых после переключения дифференциального канала 125 мкс не должно стартовать преобразование. Если же в этот период преобразования все-таки выполнялись, то их результат необходимо игнорировать.

Такую же задержку на установление необходимо ввести при первом дифференциальном преобразовании после изменения опорного источника АЦП (за счет изменения бит REFS1:0 в ADMUX).

Входные каналы АЦП

При переключении входного канала необходимо учесть некоторые рекомендации, которые исключают некорректность переключения:

- в режиме одиночного преобразования переключение канала необходимо выполнять перед началом преобразования. Переключение канала может произойти только в течение одного такта синхронизации АЦП после записи логической 1 в ADSC. Однако самым простым методом является ожидание завершения преобразования перед выбором нового канала.
- в режиме автоматического перезапуска канал необходимо выбирать перед началом первого преобразования. Переключение канала происходит аналогично - в течение одного такта синхронизации АЦП после записи логической 1 в ADSC. Но самым простым методом является ожидание завершения первого преобразования, а затем переключение канала. Поскольку следующее преобразование уже запущено автоматически, то следующий результат будет соответствовать предыдущему каналу. Последующие преобразования отражают результат для нового канала.
- при переключении на дифференциальный канал первое преобразование будет характеризоваться плохой точностью из-за переходного процесса в схеме автоматической регулировки смещения. Следовательно, первый результат такого преобразования рекомендуется игнорировать.

3.18.4 Источник опорного напряжения АЦП

Источник опорного напряжения (ИОН) для АЦП ($V_{\text{ИОН}}$) определяет диапазон преобразования АЦП. Если уровень однополярного сигнала выше $U_{V_{\text{ИОН}}}$, то результатом преобразования будет 0x3FF. В качестве $U_{V_{\text{ИОН}}}$ могут выступать U_{NVCC} , внутренний ИОН 2,56 В или внешний ИОН, подключенный к выводу AREF. NVCC подключается к АЦП через пассивный ключ. Внутреннее опорное напряжение 2,56 В генерируется внутренним эталонным источником VBG, буферизованным внутренним усилителем. В любом случае внешний вывод AREF связан непосредственно с АЦП, и поэтому можно снизить влияние шумов на опорный источник за счет подключения конденсатора между выводом AREF и общим. Напряжение $V_{\text{ИОН}}$ также может быть измерено на выводе AREF высокоменным вольтметром. Необходимо обратить внимание, что $V_{\text{ИОН}}$ является высокоомным источником и поэтому внешне к нему может быть подключена только емкостная нагрузка.

Если пользователь использует внешний опорный источник, подключенный к выводу AREF, то не допускается использование другой опции опорного источника, т. к. это приведет к шунтированию внешнего опорного напряжения. Если к выводу AREF не приложено напряжение, то пользователь может выбрать U_{NVCC} и 2,56 В качестве опорного источника. Результат первого преобразования после переключения опорного источника может характеризоваться плохой точностью и пользователю рекомендуется его игнорировать.

Если используются дифференциальные каналы, то выбранный опорный источник должен быть меньше уровня U_{NVCC} .

3.18.5 Подавитель шумов АЦП

АЦП характеризуется возможностью подавления шумов, которые вызваны работой ядра ЦПУ и периферийных устройств ввода-вывода. Подавитель шумов может быть использован в режиме снижения шумов АЦП и в режиме холостого хода. При использовании данной функции необходимо придерживаться следующей процедуры:

- следует убедиться, что работа АЦП разрешена и он не выполняет преобразования, выбирать режим одиночного преобразования и разрешить прерывание по завершении преобразования;
- ввести режим уменьшения шумов АЦП (или режим холостого хода). АЦП запустит преобразование как только остановится ЦПУ;
- если до завершения преобразования не возникает других прерываний, то АЦП вызовет прерывание ЦПУ и программа перейдет на вектор обработки прерывания по завершении преобразования АЦП. Если до завершения преобразования другое прерывание пробуждает микроконтроллер, то это прерывание обрабатывается, а по завершении преобразования генерируется соответствующий запрос на прерывание. АЦП остается в активном режиме пока не будет выполнена очередная команда sleep;
- следует братить внимание, что АЦП не отключается автоматически при переводе во все режимы сна, кроме режима холостого хода и снижения шумов АЦП. Поэтому пользователь должен предусмотреть запись логического 0 в бит ADEN перед переводом в такие режимы сна во избежание чрезмерного энергопотребления. Если работа АЦП была разрешена в таких режимах сна и пользователь желает выполнить дифференциальное преобразование, то после пробуждения необходимо включить, а затем выключить АЦП для инициации расширенного преобразования, чем будет гарантировано получение действительного результата.

3.18.6 Схема аналогового входа

Схема аналогового входа для однополярных каналов представлена на рисунке 3.101. Независимо от того, какой канал подключен к АЦП, аналоговый сигнал, подключенный к выводу ADCn, нагружается емкостью вывода и входным сопротивлением утечки. После подключения канала к АЦП аналоговый сигнал будет связан с конденсатором выборки-хранения через последовательный резистор, сопротивление которого эквивалентно всей входной цепи.

АЦП оптимизирован под аналоговые сигналы с выходным сопротивлением не более 10 кОм. Если используется такой источник сигнала, то время выборки незначительно. Если же используется источник с более высоким выходным сопротивлением, то время выборки будет определяться временем, которое требуется для зарядки конденсатора выборки-хранения источником аналогового сигнала. Рекомендуется использовать источники только с малым выходным сопротивлением и медленно изменяющимися сигналами, т. к. в этом случае будет достаточно быстрым заряд конденсатора выборки-хранения.

По отношению к каналам с дифференциальным усилением рекомендуется использовать сигналы с внутренним сопротивлением до нескольких сотен кОм. Следует предусмотреть, чтобы в предварительных каскадах формирования аналогового сигнала ко входу АЦП не вносились частоты выше $f_{\text{АЦП}}/2$, в противном случае результат преобразования может быть некорректным. Если вероятность проникновения высоких частот существует, то рекомендуется перед АЦП установить фильтр низких частот.

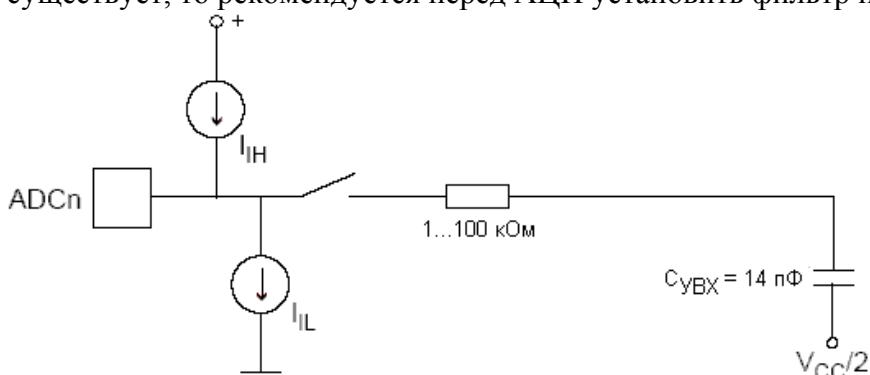


Рисунок 3.101 – Схема аналогового входа

Таблица 3.79.2 – Параметры работы АЦП

Наименование параметра, единица измерения	Режим измерения	Буквенное обозначение	Норма		
			не менее	тип	не более
1	2	3	4	5	6
Разрешение	Несимметричный канал			10	
	Дифференциальный канал (коэффициент усиления 1x, 10x)			8	
	Дифференциальный канал (коэффициент усиления 200x)			7	
Абсолютная точность, младший значащий разряд (МЗР)	Несимметричный канал $U_{\text{REF}} = 4\text{В}$ частота АЦП = 200кГц			1	
Интегральная нелинейность, МЗР	$U_{\text{REF}} = 4\text{В}$			0,5	
Дифференциальная нелинейность, МЗР	$U_{\text{REF}} = 4\text{В}$			0,5	
Смещение нуля	$U_{\text{REF}} = 4\text{В}$			1	
Время преобразования, мкс	Режим непрерывного преобразования		65		260
Тактовая частота, кГц			50		200
Аналоговое напряжение питания, В		U_{VCC}	$U_{\text{VCC}} - 0.3^{(1)}$		$U_{\text{VCC}} + 0.3^{(2)}$
Опорное напряжение, В	Несимметричные каналы	U_{REF}	2,0		U_{VCC}
	Дифференциальные каналы		2,0		$U_{\text{VCC}} - 0,2$
Напряжение входных сигналов, В	Несимметричные каналы	U_{IN}	0		U_{REF}
	Дифференциальные каналы		-		-
Напряжение внутреннего ИОН, В		U_{INT}	2,4	2,56	2,7
Входное сопротивление канала опорного напряжения, кОм		R_{REF}	6	10	13
Входное сопротивление аналогового входа		R_{AIN}		100	

Рекомендации по снижению влияния шумов на результат преобразования

Работа цифровых узлов внутри и снаружи микроконтроллера связана с генерацией электромагнитных излучений, которые могут негативно сказаться на точность измерения аналогового сигнала. Если точность преобразования является критическим параметром, то уровень шумов можно снизить, придерживаясь следующих рекомендаций:

- выполнять путь аналоговых сигналов как можно более коротко. Следить, чтобы аналоговые сигналы проходили над плоскостью (слоем) с аналоговой землей (экраном) и далеко от проводников, передающих высокочастотные цифровые сигналы;
- вывод $\cap VCC$ необходимо связать с цифровым питанием #VCC через LC-цепь в соответствии с рисунком 3.102;
- следует использовать функцию подавления шумов АЦП, внесенных работой ядра ЦПУ;
- если какой-либо из выводов АЦП используется как цифровой выход, то чрезвычайно важно не допустить переключение состояния этого выхода в процессе преобразования.

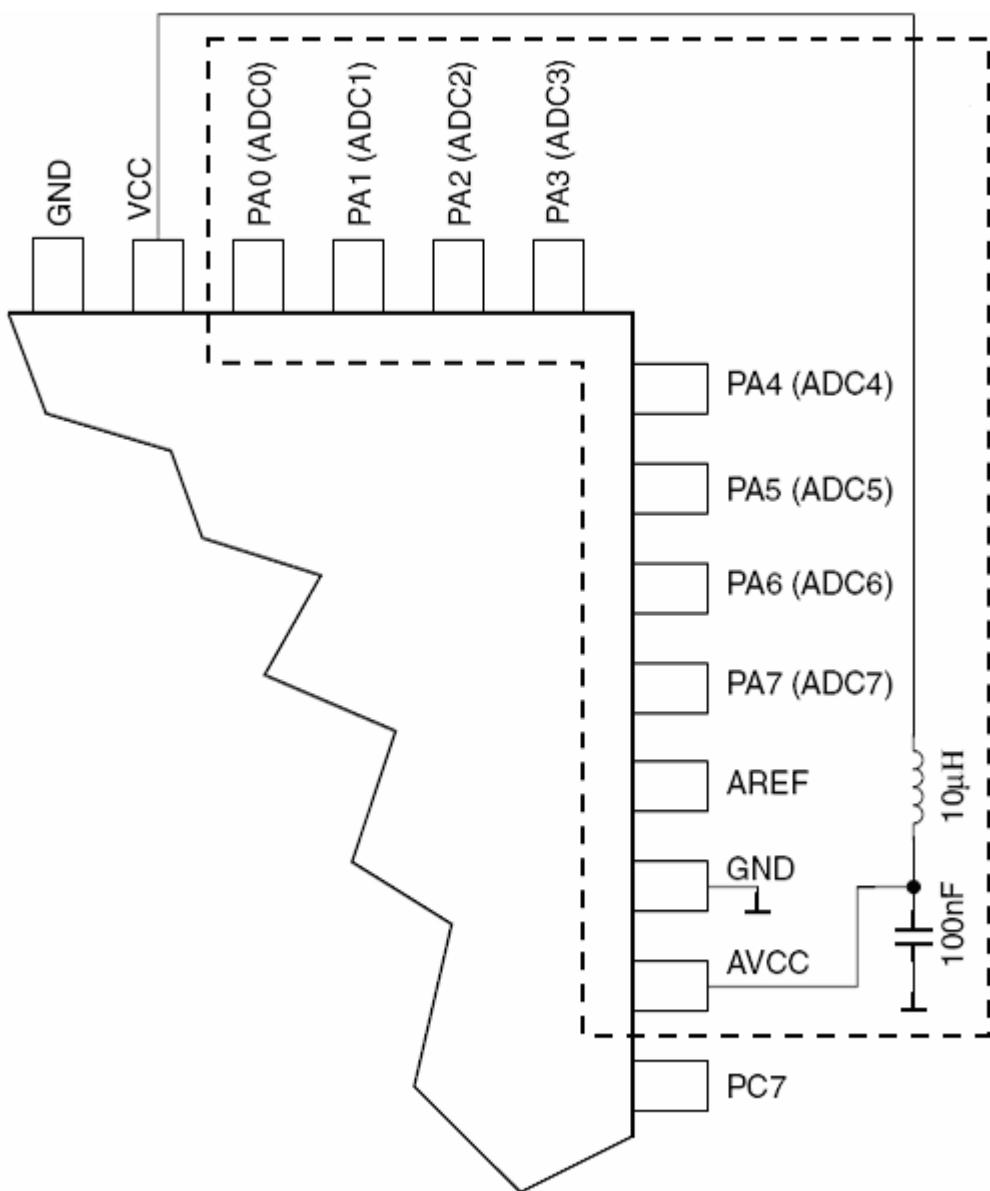


Рисунок 3.102 – Подключение питания АЦП

3.18.7 Методы компенсации смещения

Усилительный каскад имеет встроенную схему компенсации смещения, которая стремится максимально приблизить к нулю смещение дифференциального измерения. Оставшееся смещение можно измерить, если в качестве дифференциальных входов АЦП выбрать один и тот же вывод микроконтроллера. Измеренное таким образом остаточное смещение можно программно вычесть из результата преобразования. Использование программного алгоритма коррекции смещения позволяет уменьшить смещение ниже одного младшего разряда.

Определения погрешностей аналогово-цифрового преобразования

n -разрядный однополярный АЦП преобразовывает напряжение линейно между GND и $V_{ион}$ с количеством шагами 2^n (младших разрядов). Минимальный код = 0, максимальный = $2^n - 1$. Основные погрешности преобразования являются отклонением реальной функции преобразования от идеальной. К ним относятся:

Смещение – отклонение первого перехода (с 0x000 на 0x001) по сравнению с идеальным переходом (т. е. при 0,5 младшего разряда). Идеальное значение: 0 младший разряд.

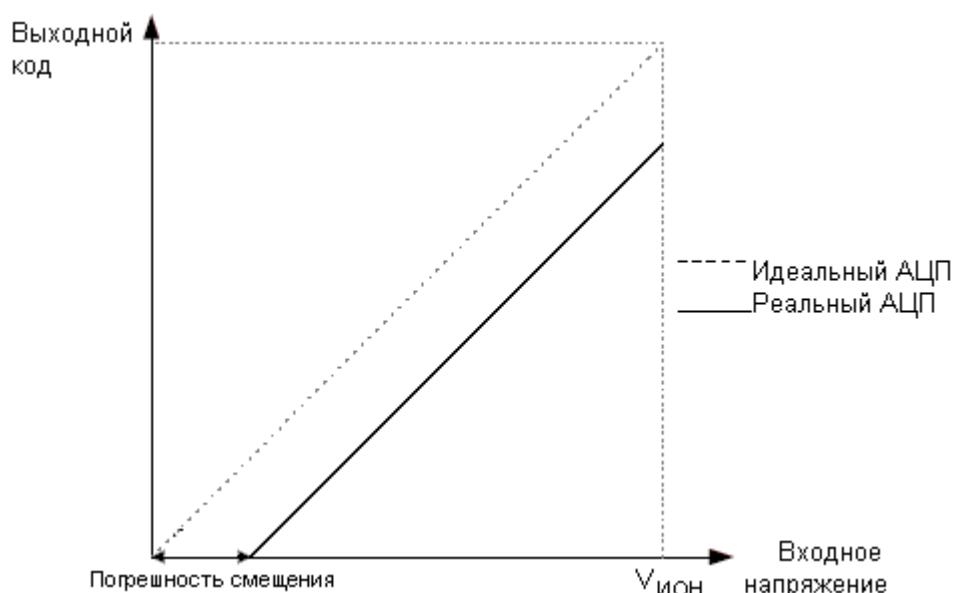


Рисунок 3.103 – Погрешность смещения

Погрешность усиления. После корректировки смещения погрешность усиления представляет собой отклонение последнего перехода (с 0x3FE на 0x3FF) от идеального перехода (т. е. отклонение при максимальном значении минус 1,5 младших разрядов). Идеальное значение: 0 младших разрядов.

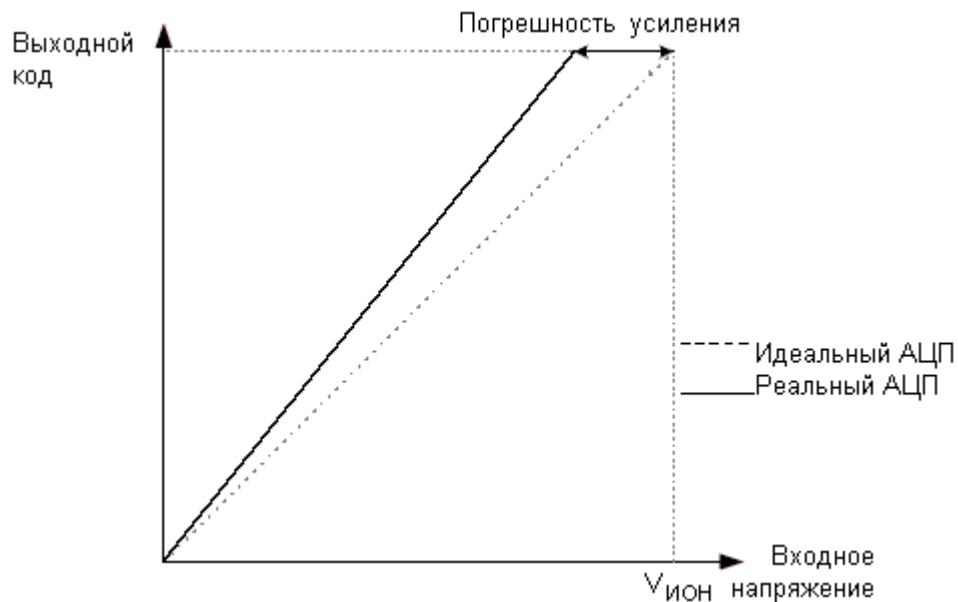


Рисунок 3.104 – Погрешность усиления

Интегральная нелинейность (ИНЛ). После корректировки смещения и погрешности усиления ИНЛ представляет собой максимальное отклонение реальной функции преобразования от идеальной для любого кода. Идеальное значение ИНЛ = 0 младший разряд.

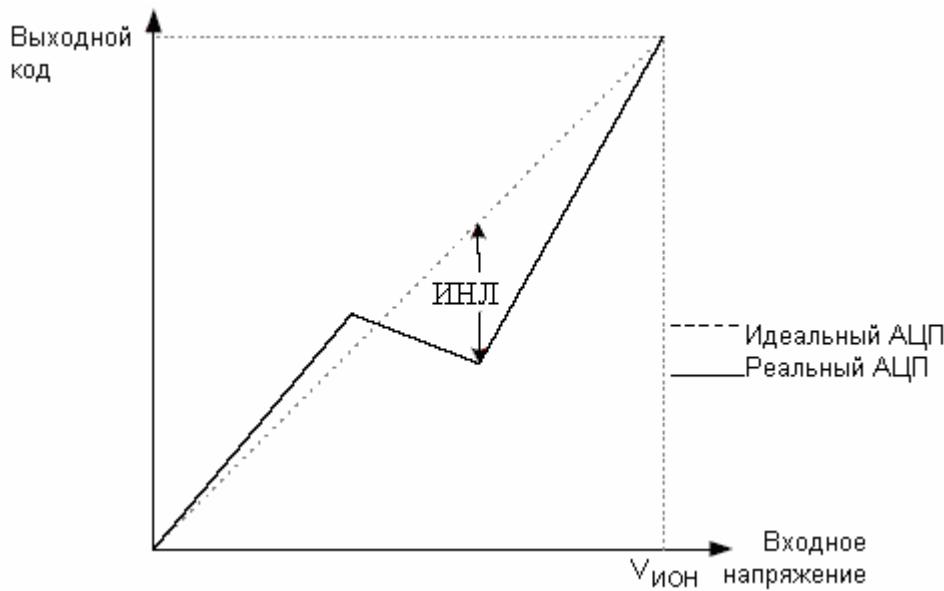


Рисунок 3.105 - Интегральная нелинейность (ИНЛ)

Дифференциальная нелинейность (ДНЛ) - максимальное отклонение между шириной фактического кода (интервал между двумя смежными переходами) от ширины идеального кода (1 младший разряд). Идеальное значение: 0 младший разряд.

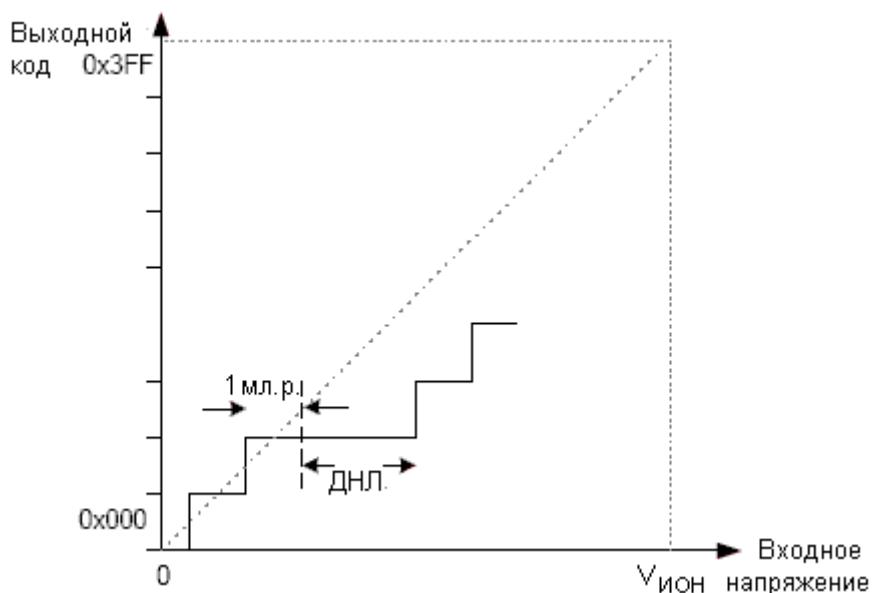


Рисунок 3.106 - Дифференциальная нелинейность (ДНЛ)

Погрешность квантования – возникает из-за преобразования входного напряжения в конечное число кодов. Погрешность квантования – это интервал входного напряжения протяженностью 1 младший разряд (шаг квантования по напряжению), который характеризуется одним и тем же кодом. Всегда равен $\pm 0,5$ младших разрядов.

Абсолютная погрешность – максимальное отклонение реальной (без подстройки) функции преобразования от реальной при любом коде. Является результатом действия нескольких эффектов: смещение, погрешность усиления, дифференциальная погрешность, нелинейность и погрешность квантования. Идеальное значение: $\pm 0,5$ младшего разряда.

Результат преобразования АЦП

По завершении преобразования ($ADIF = 1$) результат может быть считан из пары регистров результата преобразования АЦП ($ADCL$, $ADCH$).

Для однополярного преобразования:

$$ADC = U_{V_{BX}} \times 1024 / U_{V_{ИОН}}$$

где $U_{V_{BX}}$ – уровень напряжения на подключенном к АЦП входу;

$U_{V_{ИОН}}$ – напряжение выбранного источника опорного напряжения. Код 0x000 соответствует уровню аналоговой земли, а 0x3FF – уровню напряжения ИОН минус 1 шаг квантования по напряжению.

3.19 Поддержка загрузчика – самопрограммирование

Для поддержки самопрограммирования вся область памяти программ логически разделена на две секции: секцию прикладной программы (Application Section) и секцию загрузчика (Boot Loader Section). Изменение памяти программ осуществляется программой - загрузчиком, расположенной в одноименной секции. Для загрузки нового содержимого памяти программ, а также для выгрузки старого содержимого, программа-загрузчик может использовать любой интерфейс передачи данных (USART/UART, SPI, TWI), имеющийся в составе конкретного микроконтроллера. Следует отметить, что загрузчик может изменять содержимое обеих секций. Это позволяет ему модифицировать собственный код и даже удалять себя из памяти, если надобность в нем отпадет. Уровень доступа (чтение/запись) к каждой из секций задается пользователем с помощью ячеек защиты BLB02:BLB01 и BLB12:BLB11.

Переход к программе-загрузчику может осуществляться различным образом. В частности, она может быть вызвана из основной программы командами CALL/JMP. Другим способом является перемещение вектора сброса в начало секции загрузчика. В этом случае запуск программы-загрузчика будет осуществляться автоматически после каждого сброса микроконтроллера. Положение вектора сброса определяется состоянием конфигурационной ячейки BOOTRST. Если в ней содержится “1”, вектор сброса располагается в начале памяти программ по адресу \$0000. При запрограммированной ячейке, когда в ней содержится “0”, вектор сброса располагается в начале секции загрузчика.

Размер секции загрузчика и размер секции прикладной программы задается с помощью двух конфигурационных ячеек BOOTSZ1:BOOTSZ0.

Таблица 3.81 – Конфигурация области загрузчика и области приложений в зависимости от бит BOOTSZ0 и BOOTSZ1

BOOTSZ1	BOOTSZ0	Размер области загрузчика	Страниц	Область приложений	Область загрузчика	Окончание области приложений	Стартовый адрес области загрузчика
1	0	128 слов	4	0x000 – 0xF7F	0xF80 – 0xFFFF	0xF7F	0xF80
1	1	256 слов	8	0x000 – 0xEFF	0xF00 – 0xFFFF	0xEFF	0xF00
0	0	512 слов	16	0x000 – 0xDFF	0xE00 – 0xFFFF	0xDFF	0xE00
0	1	1024 слов	32	0x000 – 0xBFF	0xC00 – 0xFFFF	0xBFF	0xC00

В микроконтроллере вся память программ разбита на две области фиксированного размера, называемых “чтение во время записи” (RWW) и “нет чтения при записи” (NRWW). Отличие между этими областями заключается в различном поведении центрального процессора при изменении расположенных в них данных:

- во время выполнения операции стирания или записи страницы памяти программ, расположенной в области RWW, процессор может осуществлять чтение только из области NRWW;

- во время выполнения операции стирания или записи страницы памяти программ, расположенной в области NRWW, процессор останавливается до окончания этой операции.

Таким образом, во время изменения содержимого страницы памяти программ, расположенной в области RWW, чтение этой области запрещено. Попытка обратиться во время программирования к коду, находящемуся в области RWW (в результате выполнения команд CALL/JMP/LPM или в результате прерывания), может привести к непредска-

зуемым последствиям. Во избежание этого следует либо запретить прерывания, либо перенести таблицу векторов прерываний в секцию загрузчика, которая всегда находится в области NRWW.

Для определения того, разрешено чтение из области RWW или нет, предназначен флаг RWWSB регистра SPMCR. Установленный в «1» флаг означает, что область RWW заблокирована для чтения. По окончании операции программирования флаг RWWSB должен быть сброшен программно (см. описание регистра SPMCR).

Напротив, код, расположенный в области NRWW, может быть считан во время изменения страницы памяти программ, расположенной в области RWW. А при изменении содержимого области NRWW процессор останавливается до завершения операции.

Таблица 3.82 – Ограничение на размер секции “Чтение во время записи”

Секция	Страниц	Адрес
Чтение во время записи (RWW)	96	0x000 – 0xBFF
Нет чтения во время записи (NRWW)	32	0xC00 – 0xFFFF

3.19.1 Регистр управления SPMCR

Разряд	7	6	5	4	3	2	1	0	
	SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCR
	Чт/Зап								
Начальное значение	0	0	0	0	0	0	0	0	

Бит 7: SPMIE - Разрешение прерывания SPM

Когда в этот бит записана логическая единица и I-бит в статусном регистре установлен, SPM прерывание разрешено. SPM готовность к прерыванию будет выполняться до тех пор, пока бит SPMEN в SPMCR очищен.

Бит 6: RWWSB - Бит запрещения доступа к RWW секции

Когда самопрограммирование в RWW секции инициализировано, RWWSB бит аппаратно устанавливается. Когда он установлен, секция не может быть доступна. RWWSB бит очищается, когда в RWWSRE бит записывается логическая единица после того, как процедура самопрограммирования завершена. Альтернативно RWWSB будет автоматически очищен, если инициализирована операция загрузки страницы.

Бит 5: Res - Резервированный бит

Бит 4: RWWSRE - Чтение RWW секции разрешено

Когда программируется RWW секция, она блокирована для чтения. Для получения доступа к этой секции пользователь должен подождать окончания программирования. Таким образом, если RWWSRE бит записывается логической единицей в тоже время, когда и SPMEN, следующая SPM инструкция, проведенная в течение 4-х циклов разрешит доступ к RWW секции. RWW секция не может быть доступна, в то время пока flash занята процедурой страничного стирания или записи. Если RWWSRE бит записан, в то время как flash-память программируется, операция загрузки flash-памяти будет игнорироваться и данные будут потеряны.

Бит 3: BLBSET - Установка бит блокировки загрузчика

Если в этот бит одновременно с битом SPMEN записывается логическая единица, SPM инструкция, проведенная в следующие за установкой 4 такта, проведет установку бит блокировки загрузчика в соответствии с данными в регистре R0. Данные в R1 и адрес в Z-указателе будут игнорированы. BLBSET бит будет очищен до тех пор, пока установка бит блокировки загрузчика не будет завершена или если SPM инструкция не будет проведена в следующие за установкой SPMEN 4 цикла.

LPM инструкция, проведенная в течение 3-х циклов после установки BLBSET и SPMEN, будет читать либо биты блокировки, либо конфигурационные биты (в зависимости от Z0 в Z-указателе) в регистр назначения.

Бит 2: PGWRT - Запись страницы

Если в этот бит одновременно с битом SPMEN записывается логическая единица, SPM инструкция, проведенная в следующие за установкой 4 такта, проведет запись страницы данными, записанными во временной буфер. Адрес страницы располагается в старших битах Z указателя. Данные в R0 и в R1 будут игнорированы. Бит PGWRT будет аппаратно очищен после завершения процедуры записи или в случае, если SPM инструкция не будет проведена в следующие за установкой SPMEN 4 цикла. ЦПУ будет остановлен во время процедуры записи страницы, если адресуется NRWW секция.

Бит 1: PGERS - Стирание страницы

Если в этот бит одновременно с битом SPMEN записывается логическая единица, SPM инструкция, проведенная в следующие за установкой 4 такта, проведет стирание страницы. Адрес страницы располагается в старших битах Z указателя. Данные в R0 и в R1 будут игнорированы. Бит PGERS будет аппаратно очищен после завершения процедуры стирания или в случае, если SPM инструкция не будет проведена в следующие за установкой SPMEN 4 цикла. ЦПУ будет остановлен во время процедуры записи страницы, если адресуется NRWW секция.

Бит 0: SPMEN - Разрешение SPM

Этот бит разрешает SPM инструкцию в течение следующих 4-х циклов. Если этот бит устанавливается вместе с RWWSRE, BLBSET, PGWRT или PGERS, следующая команда SPM будет иметь специальное значение. Если устанавливается только SPMEN, следующая команда SPM будет сохранять значение R1: R0 во временной страничный буфер, адресуемый Z-указателем. Младшие биты Z-указателя игнорируются. SPMEN бит будет автоматически очищен после завершения SPM инструкции или в случае, если SPM инструкция не будет проведена в следующие за установкой SPMEN 4 цикла. Во время процедур стирания или записи страницы бит SPMEN будет установлен до завершения операции.

Запись в младшие пять разрядов регистра значений, отличных от “10001”, “01001”, “00101”, “00011” и “00001”, не вызывает никакого эффекта.

Следует обратить внимание на то, что во время записи в EEPROM память изменение регистра SPMCR невозможно. Поэтому перед тем как записать какое-либо значение в регистр SPMCR, рекомендуется дождаться сброса флага Eewe регистра EECR. Для адресации памяти программ при использовании команды SPM используется индексный регистр Z, получаемый объединением двух старших регистров общего назначения R30 (младший байт) и R31 (старший байт). Поскольку память программ имеет страничную организацию, счетчик команд можно условно разбить на две части. Первая часть (младшие разряды) адресует ячейку на странице, а вторая часть определяет страницу. После

запуска операции программирования содержимое регистра Z фиксируется и его можно использовать для других целей.

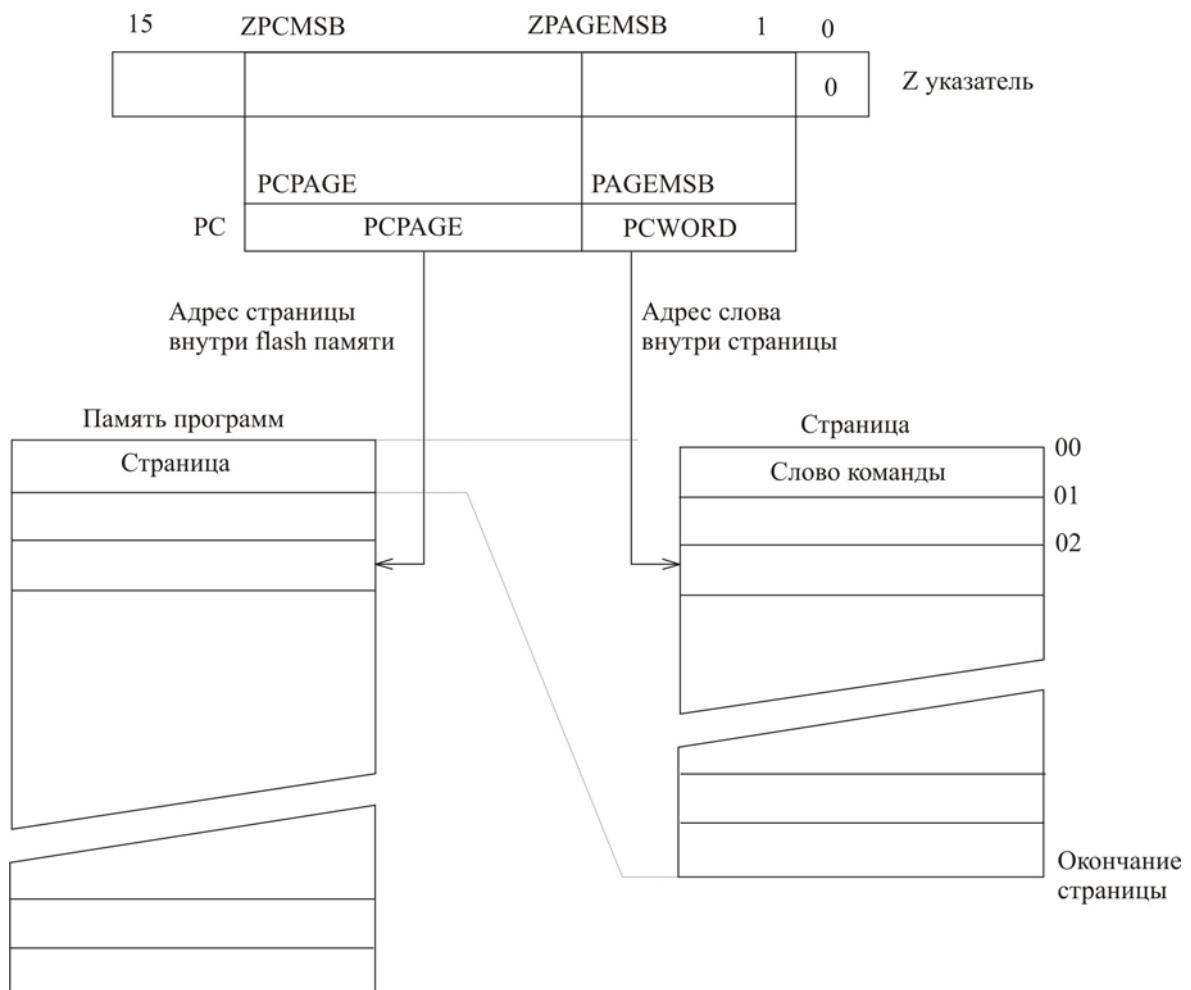


Рисунок 3.107 – Адресация к памяти программ через Z-указатель

3.19.2 Изменение памяти программ

Изменение содержимого памяти программ осуществляется в следующей последовательности:

- 1 Заполнение временного буфера страницы новым содержимым.
- 2 Очистка страницы.
- 3 Перенос содержимого буфера в память программ.

Следует заметить, что очистка страницы может выполняться как после заполнения буфера, так и перед его заполнением. Однако при необходимости изменить только часть страницы приведенный порядок действий является, по понятным причинам, единственным возможным. В этом случае содержимое ячеек, не требующих изменения, сохраняется в буфере перед очисткой страницы. Для определения момента окончания выполнения операций можно либо опрашивать состояние флага SPMEN регистра SPMCR, дожидаясь его сброса, либо воспользоваться прерыванием “готовность SPM”. Это прерывание генерируется все время, пока флаг SPMEN сброшен. В последнем случае таблица векторов прерываний должна находиться в секции загрузчика, а это прерывание должно быть разрешено установкой флага SPMIE регистра SPMCR.

Для стирания страницы памяти программ необходимо занести адрес страницы в регистр Z (секция PCPAGE), записать значение “x0000011” в регистр SPMCR и в течение

четырех машинных циклов выполнить команду SPM. Содержимое регистров RI и R0 при этом игнорируется.

Для занесения слова команды в буфер следует загрузить адрес ячейки в регистр Z (секция PCWORD), а код операции в регистры RI:R0. После этого необходимо записать значение «x0000011» в регистр SPMCR и в течение четырех машинных циклов выполнить команду SPM. Очистка буфера осуществляется автоматически по окончании записи страницы вручную записью логической 1 в разряд RWWSRE регистра SPMCR. Запись по одному и тому же адресу в буфере невозможна без его очистки.

Запись содержимого буфера в память программ осуществляется аналогично. В регистр Z (секция PCPAGE) заносится адрес страницы, в регистр SPMCR записывается значение «x00000101» и в течение четырех машинных циклов выполняется команда SPM. Содержимое регистров RI и R0 при этом игнорируется.

3.19.3 Изменение ячеек защиты загрузчика

Изменение ячеек защиты загрузчика BLB12:BLB11 и BLB02:BLB01 также осуществляется командой SPM. Для этого необходимо загрузить в регистр R0 требуемое значение в соответствии с рисунком 3.108 (сброшенный разряд означает программирование соответствующей ячейки).

	7	6	5	4	3	2	1	0
R0	1	1	BLB12	BLB11	BLB02	BLB01	1	1

Рисунок 3.108 – Изменение состояния ячеек защиты регистра R0 по команде SPM

После этого необходимо записать значение «x0001001» в регистр SPMCR и в течение четырех машинных циклов выполнить команду SPM. Содержимое регистра Z при этом игнорируется, однако для совместимости с будущими устройствами рекомендуется записывать в него значение \$0001. Во время программирования ячеек защиты можно обращаться к любой области памяти программ.

3.19.4 Чтение конфигурационных ячеек и ячеек защиты

Помимо программирования микроконтроллера загрузчик может также считывать содержимое конфигурационных ячеек и ячеек защиты. Так для чтения байта защиты следует загрузить в регистр Z число \$0001, записать в регистр SPMCR значение «x0001001» и в течение трех машинных циклов выполнить команду LPM. В результате содержимое байта защиты будет занесено в заданный регистр общего назначения.

Соответствие разрядов регистра ячейкам приведено в таблице 3.83.

Чтение конфигурационных байтов осуществляется аналогично. В регистр Z загружается адрес байта (\$0000 младший байт, \$0003 старший байт, \$0002 дополнительный байт), после чего необходимо записать в регистр SPMCR значение «x0001001» и в течение трех машинных циклов выполнить команду LPM. В результате выполнения команды содержимое выбранного байта конфигурации будет занесено в регистр общего назначения.

3.20 Программирование памяти

ИМС 1887ВЕ1У поддерживает следующие режимы программирования:

- последовательное программирование при низком напряжении (по интерфейсу SPI);
- параллельное программирование при высоком напряжении.

Под «высоким» напряжением здесь понимается управляющее напряжение (12 В), подаваемое на вывод RESET# микроконтроллера для перевода последнего в режим программирования.

3.20.1 Защита кода и данных

Содержимое FLASH - памяти (памяти программ), а также содержимое EEPROM-памяти (память данных) может быть защищено от записи и/или чтения посредством программирования ячеек защиты (Lock Bits). Возможные режимы защиты, соответствующие различным состояниям этих ячеек, приведены в таблице 3.83.

Таблица 3.83 - Биты блокировки

Биты блокировки	Номер бита	Описание	Значение по умолчанию
	7	-	1 (незапрограммирован)
	6	-	1 (незапрограммирован)
BLB12	5	Бит блокировки загрузчика	1 (незапрограммирован)
BLB11	4	Бит блокировки загрузчика	1 (незапрограммирован)
BLB02	3	Бит блокировки загрузчика	1 (незапрограммирован)
BLB01	2	Бит блокировки загрузчика	1 (незапрограммирован)
LB2	1	Бит блокировки	1 (незапрограммирован)
LB1	0	Бит блокировки	1 (незапрограммирован)

Таблица 3.84 – Режимы защиты бит блокировки LB

Биты защиты памяти			Тип защиты
LB режим	LB2	LB1	
1	1	1	Память доступна для чтения и записи
2	1	0	Дальнейшее программирование Flash и EEPROM, бит конфигурации запрещено в параллельном и последовательном режиме
3	0	0	Дальнейшее программирование и верификация Flash и EEPROM, бит конфигурации запрещено в параллельном и последовательном режиме

Таблица 3.85 - Режимы защиты бит блокировки BLB0

Биты защиты памяти			Тип защиты
BLB0 режим	BLB02	BLB01	
BLB0 режим	BLB02	BLB01	Нет ограничений для SPM и LPM доступа секции приложений
1	1	1	SPM не разрешена для записи в секцию приложений
2	1	0	SPM не разрешена для записи в секцию приложений и LPM, выполняемая из секции загрузчика, не разрешена для чтения из области приложений. Если векторы прерываний расположены в области загрузчика, они запрещены во время выполнения программы из области приложений
3	0	0	LPM, выполняемая из области загрузчика, не может производить чтение из области приложений. Если векторы прерываний расположены в области загрузчика, они запрещены во время выполнения программы из области приложений

Таблица 3.86 - Режимы защиты бит блокировки BLB1

Биты защиты памяти			Тип защиты
BLB1 режим	BLB12	BLB11	
BLB1 режим	BLB12	BLB11	Нет ограничений для SPM и LPM доступа секции загрузчика
1	1	1	SPM не разрешена для записи в секцию загрузчика
2	1	0	SPM не разрешена для записи в секцию загрузчика и LPM, выполняемая из секции приложений, не разрешена для чтения из области загрузчика. Если векторы прерываний расположены в области приложений, они запрещены во время выполнения программы из области загрузчика
3	0	0	LPM, выполняемая из секции приложений, не разрешена для чтения из области загрузчика. Если векторы прерываний расположены в области приложений, они запрещены во время выполнения программы из области загрузчика

При использовании параллельного режима программирования в режимах 2 и 3 запрещается также изменение конфигурационных ячеек. Поэтому включение защиты следует выполнять в самую последнюю очередь, после программирования остальных областей памяти микроконтроллера.

В исходном (незапрограммированном) состоянии в этих ячейках содержится «1», после программирования — «0».

3.20.2 Конфигурационные ячейки

Таблица 3.87 – Старший конфигурационный байт

Старший конфигурационный байт	Номер бита	Описание	Значение по умолчанию
S8535C	7	Выбор режима совместимости с AT90S8535	1 (незапрограммирован)
WDTON	6	WDR всегда включен	1 (незапрограммирован)
SPIEN ¹⁾	5	Разрешение последовательного программирования и загрузки данных	0 (запрограммирован) SPI программирование разрешено
CKOPT ²⁾	4	Опции осциллятора	1 (незапрограммирован)
EESAVE	3	EEPROM не очищается процедурой ChipErase	1 (незапрограммирован)
BOOTSZ1	2	Выбор размера массива области загрузчика	0 (запрограммирован) ³⁾
BOOTSZ2	1	Выбор размера массива области загрузчика	0 (запрограммирован) ³⁾
BOOTRST	0	Выбор вектора сброса	1 (незапрограммирован)

¹⁾ SPIEN бит не доступен в режиме последовательного программирования.

²⁾ CKOPT бит функционально зависит от установки CKSEL бит.

³⁾ Значение по умолчанию для битов BOOTSZ1...2 выбирает максимальный размер массива области загрузчика.

Таблица 3.88 – Младший конфигурационный байт

Младший конфигурационный байт	Номер бита	Описание	Значение по умолчанию
BODLEVEL	7	Пороговый уровень BOD	1 (незапрограммирован)
BODEN	6	Разрешение работы BOD	1 (незапрограммирован)
SUT1	5	Выбор времени старта	1 (незапрограммирован)
SUT0	4	Выбор времени старта	0 (запрограммирован)
CKSEL3	3	Выбор источника тактирования	0 (запрограммирован)
CKSEL2	2	Выбор источника тактирования	0 (запрограммирован)
CKSEL1	1	Выбор источника тактирования	0 (запрограммирован)
CKSEL0	0	Выбор источника тактирования	1 (незапрограммирован)
Примечания			
1 Значение по умолчанию SUT1...0 выбирает максимальное время старта			
2 Значение по умолчанию CKSEL3...0 выбирает внутренний RC генератор с частотой равной 1 МГц.			

Как следует из названия, конфигурационные ячейки (Fuse Bits) определяют ряд параметров конфигурации микроконтроллера. Эти ячейки расположены в отдельном адресном пространстве, доступном только при программировании.

3.20.3 Идентификатор

Микроконтроллер имеет три 8-разрядные ячейки, содержимое которых позволяет идентифицировать устройство. Как и конфигурационные ячейки, ячейки идентификатора расположены в отдельном адресном пространстве, доступ к которому возможен только в режиме программирования. Однако в отличие от конфигурационных ячеек, ячейки идентификатора, по понятным причинам, доступны только для чтения.

3.20.4 Калибровочная ячейка

ИМС содержит четыре различных калибровочных значения для внутреннего RC-осцилятора. Эти байты располагаются в старших байтах сигнатурного ряда по адресам: 0x000, 0x001, 0x002 и 0x003 для 1, 2, 4 и 8 МГц соответственно. Во время сброса значение 1 МГц автоматически загружается в OSCCAL регистр. Если необходима другая частота, калибровочное значение должно быть загружено пользователем вручную.

3.20.5 Параллельное программирование

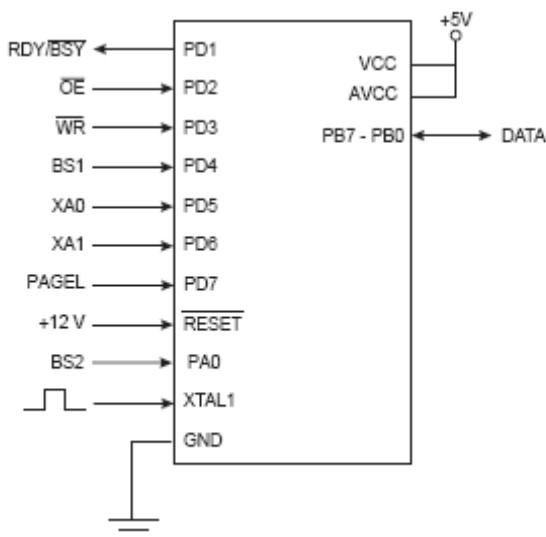


Рисунок 3.109 – Включение схемы при параллельном программировании

Таблица 3.89 – Управляющие сигналы программирования

Наименование сигналов	Имя вывода	Направление (I/O)	Функция
RDY/BSY#	PD1	O	0: Программирование 1: Устройство готово для новой команды
OE#	PD2	I	Чтение разрешено
WR#	PD3	I	Запись разрешена
BS1	PD4	I	Выбор байта 1: 0 – младший, 1 – старший
XA0	PD5	I	Бит действия 0 XTAL
XA1	PD6	I	Бит действия 1 XTAL
PAGEL	PD7	I	Бит страничной загрузки Flash и EEPROM
BS2	PA0	I	Выбор байта 2: 0 – младший, 1 – старший
DATA	PB7- PB 0	I/O	Двунаправленная шина данных

Таблица 3.90 – Значения выводов при входе в режим программирования

Вывод	Символ	Значение
PAGEL	Разрешение программирования [3]	0
XA1	Разрешение программирования [2]	0
XA0	Разрешение программирования [1]	0
BS1	Разрешение программирования [0]	0

Таблица 3.91 – Кодирование XA1 и XA0

XA1	XA0	Действие, производимое на импульс XTAL
0	0	Загрузка Flash и EEPROM адреса
0	1	Загрузка данных
1	0	Загрузка инструкции
1	1	Нет действия

Таблица 3.92 – Коды команд байта инструкций программирования

Байт команды	Выполняемое действие
1000 0000	Стирание памяти
0100 0000	Запись битов конфигурации
0010 0000	Запись битов блокировки
0001 0000	Запись Flash
0001 0001	Запись EEPROM
0000 1000	Чтение идентификационного и калибровочного байтов
0000 0100	Чтение конфигурационных и блокировочных битов
0000 0010	Чтение Flash
0000 0011	Чтение EEPROM

Таблица 3.93 – Количество слов в странице и количество страниц в массиве Flash

Размер массива Flash	Размер страницы	PCWORD	Количество страниц	Количество страниц РС	PCMSB
4К слов	32 слова	PC[4:0]	128	PC[11:5]	11

Таблица 3.94 – Количество слов в странице и количество страниц в массиве EEPROM

Размер массива EEPROM	Размер страницы	PCWORD	Коллчествово страниц	Колличествово страниц РС	EEAMSB
512 байт	4 байта	EEA[1:0]	128	PC[8:2]	8

Вход в режим программирования

Для входа в режим параллельного программирования выполняется следующий алгоритм:

- 1 Подать напряжение питания и ожидание не менее 100 мкс.
- 2 Установить RESET# в низкое логическое состояние и переключить XTAL не менее 6 раз.
- 3 Установить вывода разрешения программирования в значение “0000” и ожидать не менее 100 нс.
- 4 Подать (11,5 – 12,5) В на RESET#. Любое действие на выводах разрешения программирования в течение 100 нс после подачи напряжения на RESET# приведет к сбою входа в режим программирования.

Следует заметить, если тактирование схемы сконфигурировано для работы от внешнего кварца или внешней RC цепочки, это может привести к невозможности работы с XTAL. Для избегания этого должен быть выполнен следующий алгоритм:

- 1 Установить выводы разрешения программирования, указанные в таблице 3.90 в значение “0000”.
- 2 Подать напряжение питания, подключить общий вывод одновременно с подачей (11,5-12,5) В на вывод RESET#.
- 3 Подождать 100 нс.
- 4 Перепрограммировать конфигурационные биты для тактирования схемы от внешнего тактового генератора(CKSEL3:0 = 0b0000). Если биты блокировки запрограммированы, процедура ChipErase (стирание памяти) должна предшествовать программированию конфигурационных битов.
- 5 Выйти из режима программирования, выполнив либо отключение питания, либо подачу низкого логического уровня на вывод RESET#.
- 6 Войти в режим программирования, используя первоначальный алгоритм, описанный выше.

Условия эффективного программирования

Загруженные команда и адрес остаются в устройстве в течение цикла программирования. Для эффективного программирования должны быть учтены следующие моменты:

- команду необходимо загружать только один раз при проведении неоднократных процедур записи/чтения;
- необходимо избегать записи значения 0xFF, которое является содержимым массива Flash и EEPROM после процедуры ChipErase;
- старший байт адреса должен быть загружен только при программировании или чтении новых 256-слов в массив Flash или новых 256-байт в массив EEPROM.

Стирание памяти (ChipErase)

Процедура ChipErase будет стирать целиком массивы Flash и EEPROM плюс биты блокировки. Биты блокировки не сбрасывают своих значений до тех пор, пока память программ полностью не стерта. Конфигурационные биты не изменяют своих значений. Процедура ChipErase проводится перед программированием Flash и/или EEPROM.

Необходимо заметить, что процедура ChipErase не производит стирания массива EEPROM в случае, если конфигурационный бит EESAVE запрограммирован.

Загрузка команды ChipErase:

- 1 Установить XA1 и XA0 в значение “10”. Это разрешит загрузку команды.
- 2 Установить BS1 в “0”.
- 3 Установить данные в “1000 0000”. Это команда для ChipErase.
- 4 Подать на BQ1 положительный импульс. Это приведет к загрузке команды.
- 5 Подать отрицательный импульс на WR# - начало процедуры ChipErase. RDY/BSY# перейдет в низкое логическое состояние.
- 6 Дождаться перехода RDY/BSY# в высокое логическое состояние перед подачей новой команды.

Программирование Flash

Массив Flash организован постранично. Когда программируется Flash-память, программируемые данные защелкиваются в буфер страницы. Это позволяет программировать страницу данных одновременно. Следующая процедура описывает шаги программирования массива Flash.

- 1 Загрузка команды “Write Flash”:
 - установить XA1, XA0 в значение “10”. Это разрешит загрузку команды “Write Flash”;
 - установить BS0 в значение “0”;
 - установить вход данных в “0001 0000”. Это код команды Write Flash;
 - подать положительный импульс на BQ1. Это приведет к загрузке команды.
- 2 Загрузка младшего байта адреса:
 - установить XA1, XA0 в значение “00”. Это разрешит загрузку адреса;
 - установить BS0 в значение “0”;
 - установить вход данных в значение младшего байта адреса;
 - подать положительный импульс на BQ1. Это приведет к загрузке младшего байта адреса.
- 3 Загрузка младшего байта данных:
 - установить XA1, XA0 в значение “01”. Это разрешит загрузку данных;
 - установить вход данных в значение младшего байта данных;
 - подать положительный импульс на BQ1. Это приведет к загрузке младшего байта данных.
- 4 Загрузка старшего байта данных:
 - установить BS0 в значение “1”. Это значение выбирает старший байт данных;
 - установить XA1, XA0 в значение “01”. Это разрешит загрузку данных;
 - установить вход данных в значение старшего байта данных;
 - подать положительный импульс на BQ1. Это приведет к загрузке старшего байта данных.
- 5 Защелкивание данных:
 - установить BS0 в значение “1”. Это значение выбирает старший байт данных;
 - подать на вход PAGEL положительный импульс. Это приведет к защелкиванию байта данных.

6 Повторить шаги с 2 по 5 до тех пор, пока весь буфер не заполнится или до тех пор, пока все данные не будут загружены в страницу.

В то время как младшие биты адреса обращаются к слову внутри страницы, старшие биты обращаются к странице внутри массива Flash. Следует отметить, что если менее чем 8 бит требуется для адресации слова в странице, то наиболее значащие биты в младшем байте адреса использованы для адресации страницы, когда представлена страничная запись.

7 Загрузка старшего байта адреса:

- установить XA1, XA0 в значение “00”. Это разрешит загрузку адреса;
- установить BS0 в значение “1”;
- установить вход данных в значение старшего байта адреса ;
- подать положительный импульс на BQ1. Это приведет к загрузке старшего байта адреса.

8 Программирование страницы:

- установить BS0 в значение “0”;
- подать отрицательный импульс на WR# - начало процедуры записи. RDY/BSY# перейдет в низкое логическое состояние;
- дождаться перехода RDY/BSY# в высокое логическое состояние перед подачей новой команды.

9 Повторить шаги с 2 по 8 до тех пор, пока весь массив или все данные не будут запрограммированы.

10 Завершение страничного программирования:

- установить XA1, XA0 в значение “10”. Это разрешит загрузку команды;
- установить вход данных в “0000 0000”. Это код “холостой” команды;
- подать положительный импульс на BQ1. Это приведет к загрузке команды и внутренний сигнал записи будет сброшен.

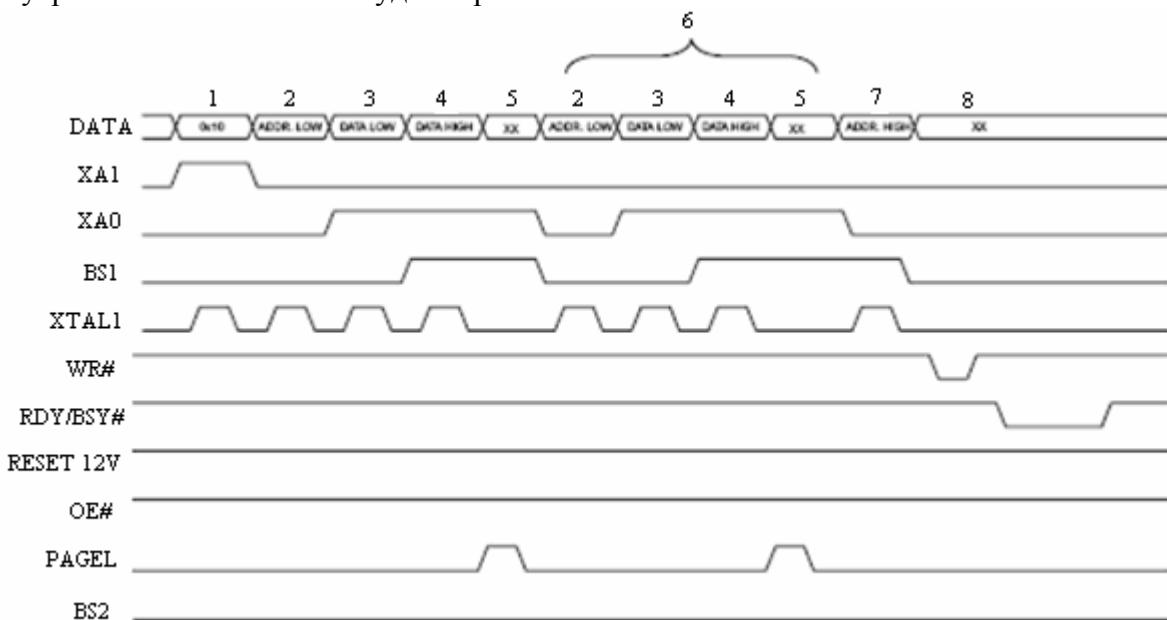


Рисунок 3.110 – Временная диаграмма программирования Flash памяти

Программирование EEPROM

Память данных EEPROM организована в страничном режиме. Когда программируется массив EEPROM, данные защелкиваются в страничном буфере. Это позволяет программировать страницу данных одновременно. Ниже приведен алгоритм загрузки страницы EEPROM:

- 1 Загрузить команду “0001 0001”.
- 2 Загрузить старший байт адреса.
- 3 Загрузить младший байт адреса.
- 4 Загрузить данные.
- 5 Выполнить защелкивание данных (подать положительный импульс PAGEL).
- 6 Повторить шаги с 3 по 5 до тех пор, пока вся страница не будет заполнена.

Для программирования массива EEPROM необходимо выполнить следующие шаги:

- 1 Установить BS0 в значение “1”.
- 2 Подать отрицательный импульс на WR# - начало процедуры записи. RDY/BSY# перейдет в низкое логическое состояние.
- 3 Дождаться перехода RDY/BSY# в высокое логическое состояние перед подачей новой команды.

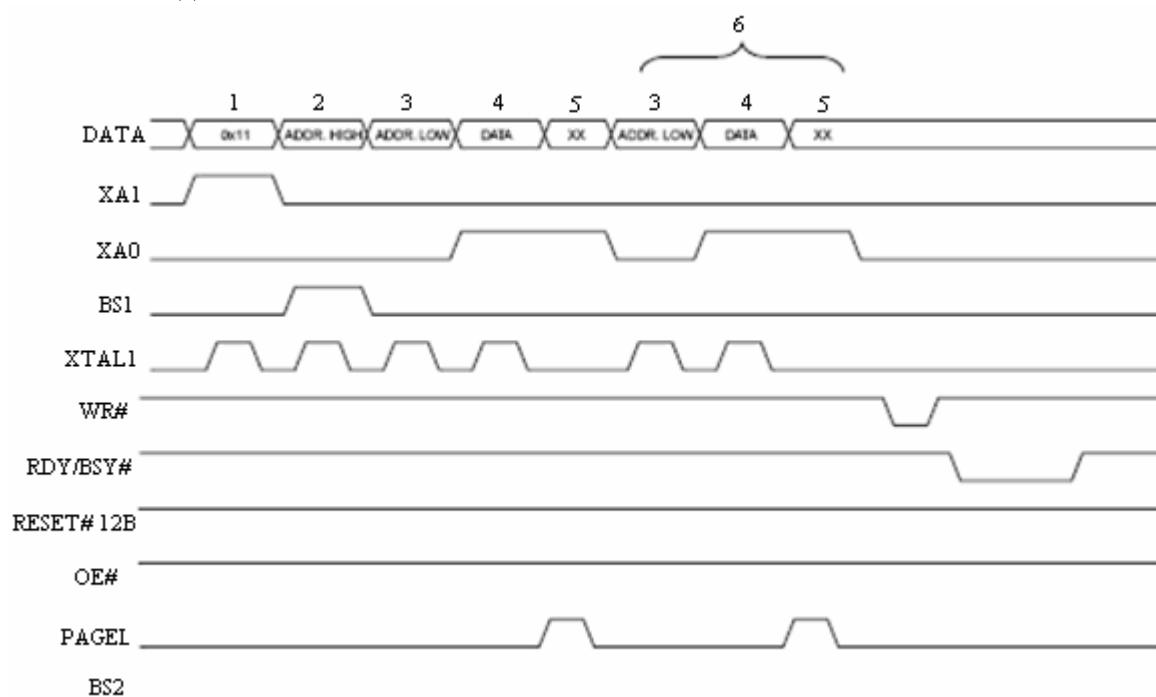


Рисунок 3.111 – Временная диаграмма программирования массива EEPROM

Чтение Flash-памяти

- 1 Загрузить команду “0000 0010”.
- 2 Загрузить старший байт адреса.
- 3 Загрузить младший байт адреса.
- 4 Установить OE# в значение “0” и BS1 в значение “0”. Младший байт слова Flash памяти может быть прочитан на выводе DATA.
- 5 Установить BS1 в значение “1”. Старший байт слова Flash памяти может быть прочитан на выводе DATA.
- 6 Установить OE# в значение “1”.

Чтение массива EEPROM

- 1 Загрузить команду “0001 0011”.
- 2 Загрузить старший байт адреса.
- 3 Загрузить младший байт адреса.
- 4 Установить OE# в значение “0” и BS1 в значение “0”. Байт данных EEPROM - памяти может быть прочитан на выводе DATA.
- 5 Установить OE# в значение “1”.

Программирование младшего конфигурационного байта

- 1 Загрузить команды “0100 0000”(см. рисунок 3.112).
- 2 Загрузить младший байт данных. Бит n = 0 программирует, а бит n = 1 стирает конфигурационный бит.
- 3 Установить BS1 в значение “0” и BS2 в значение “0”. Это выберет младший байт данных.
- 4 Подать отрицательный импульс на WR# - начало процедуры записи – и ожидать, пока сигнал RDY/BSY# перейдет в высокое логическое состояние.

Программирование старшего конфигурационного байта

- 1 Загрузить команду “0100 0000”(см. рисунок 3.112).
- 2 Загрузить старший байт данных. Бит n = 0 программирует, а бит n = 1 стирает конфигурационный бит.
- 3 Установить BS1 в значение “1” и BS2 в значение “0”. Это выберет старший байт данных.
- 4 Подать отрицательный импульс на WR# - начало процедуры записи – и ожидать, пока сигнал RDY/BSY# перейдет в высокое логическое состояние.
- 5 Установить BS1 в значение “0””. Это выберет младший байт данных.

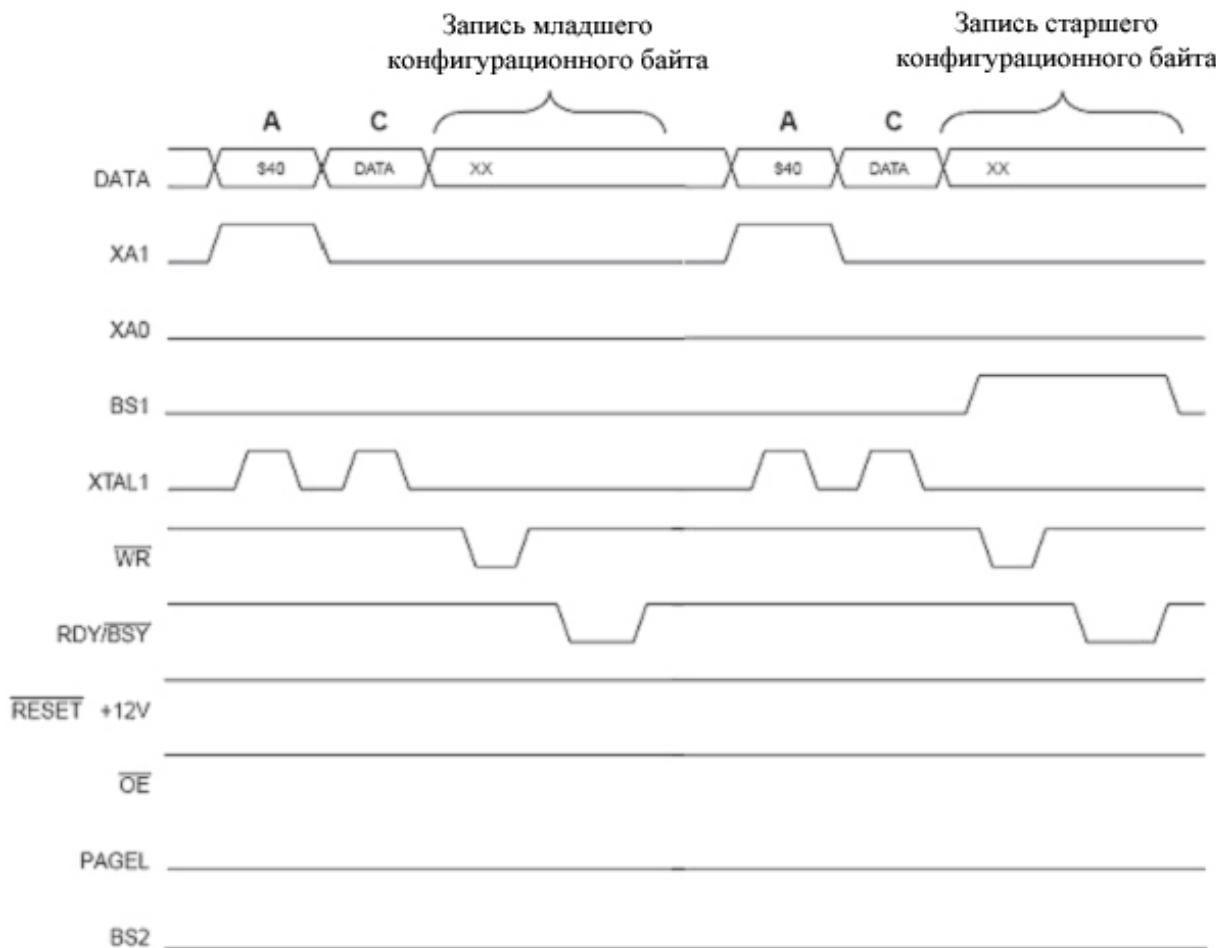


Рисунок 3.112 – Временная диаграмма программирования конфигурационных байтов

Программирование битов блокировки

- 1 Загрузить команду “0010 0000”.
- 2 Загрузить младший байт данных. Бит $n = 0$ программирует бит блокировки.
- 3 Подать отрицательный импульс на WR# - начало процедуры записи, и ожидать, пока сигнал RDY/BSY# перейдет в высокое логическое состояние.

Чтение конфигурационных битов и битов блокировки

- 1 Загрузить команду “0000 0100”(см. рисунок 3.113).
- 2 Установить OE# в значение “0”, BS2 в “0”, BS1 в “0”. Значения младших конфигурационных битов могут быть прочитано на выводе DATA (“0” означает запрограммированный бит).
- 3 Установить OE# в значение “0”, BS2 в “1”, BS1 в “1”. Значения старших конфигурационных битов могут быть прочитано на выводе DATA (“0” означает запрограммированный бит).
- 4 Установить OE# в значение “0”, BS2 в “0”, BS1 в “1”. Значения битов блокировки могут быть прочитано на выводе DATA (“0” означает запрограммированный бит).
- 5 Установить OE# в значение “1”.

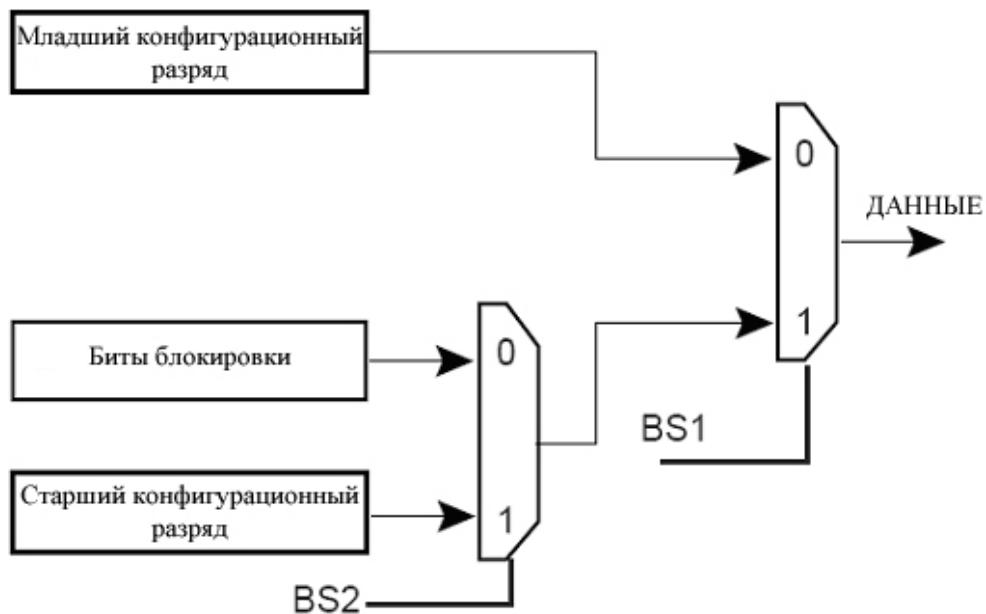


Рисунок 3.113 – Мультиплексирование конфигурационных битов и битов блокировки битами BS1 и BS2 при чтении на выводе “Данные”

Чтение байтов сигнатуры

- 1 Загрузить команду “0000 1000”.
- 2 Загрузить младший байт адреса (0x00 – 0x02).
- 3 Установить OE# в значение “0”, и BS1 в “0”. Теперь байт сигнатуры может быть прочитан на выводе DATA.
- 4 Установить OE# в значение “1”.

Чтение калибровочного байта

- 1 Загрузить команду “0000 1000”.
- 2 Загрузить младший байт адреса, 0x00.
- 3 Установить OE# в значение “0”, BS1 в значение “1”. Калибровочный байт может быть прочитан на выводе DATA.
- 4 Установить OE# в значение “1”.

Характеристики параллельного программирования

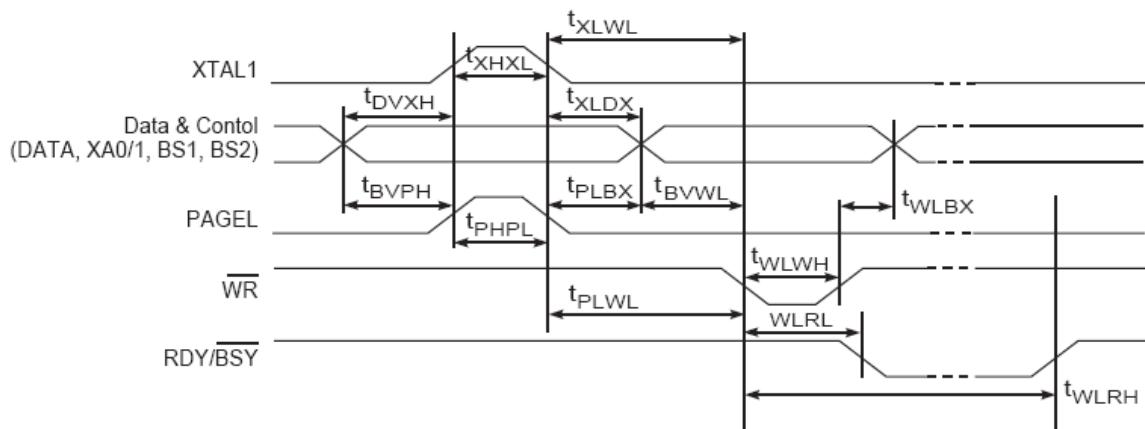


Рисунок 3.114 – Временная диаграмма управляющих сигналов при программировании, включая некоторые общие временные требования

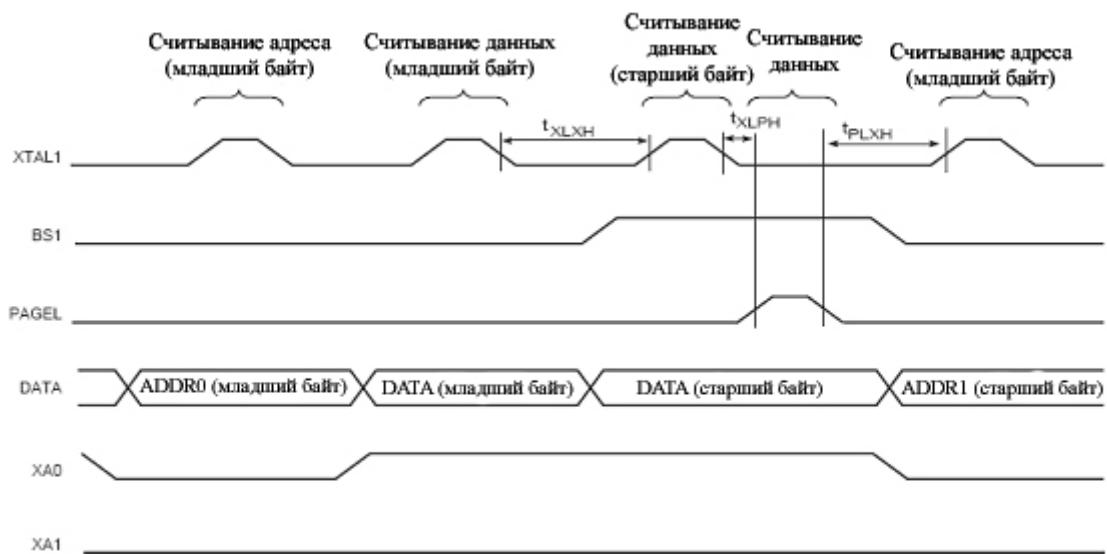


Рисунок 3.115 – Временные требования к последовательности загрузки

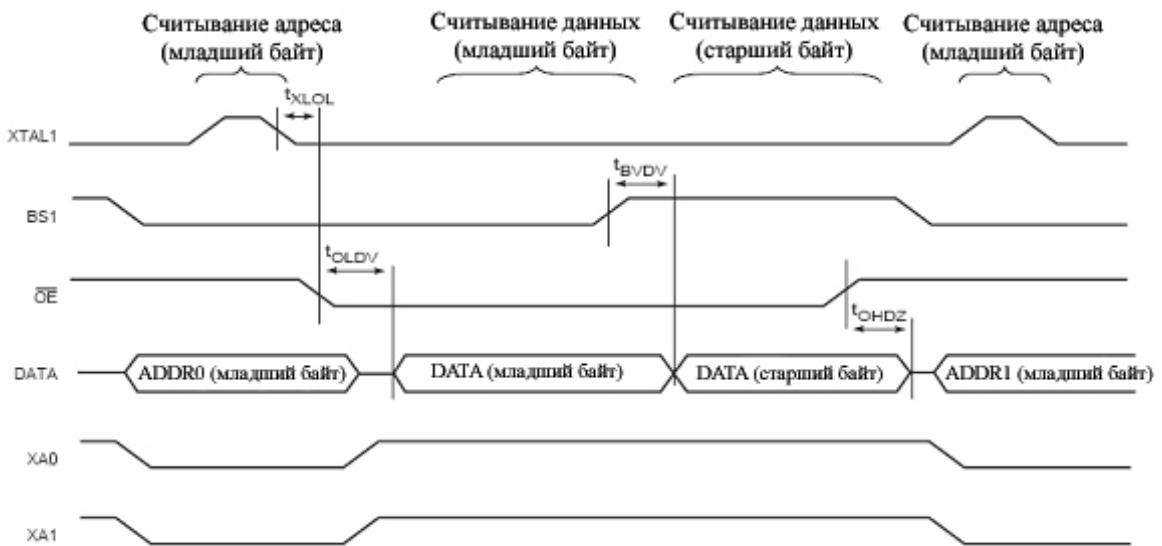


Рисунок 3.116 – Временные требования к последовательности чтения данных

Динамические характеристики, указанные на рисунках 3.114 - 3.116, представлены в таблице 3.95.

Таблица 3.95 – Динамические характеристики параллельного программирования

Символ	Параметр	Мин	Тип	Макс	Единицы
V _{PP}	Напряжение программирования	11,5		12,5	В
I _{PP}	Ток программирования			250	мА
t _{DVXH}	Предустановка данных и управления к моменту перехода BQ1 в высокое логическое состояние	67			нс
t _{XLXH}	Интервал между двумя положительными фронтами BQ1	200			нс
t _{XHXL}	Ширина положительного импульса BQ1	150			нс
t _{XLDX}	Задержка данных и управления после перехода BQ1 в низкое логическое состояние	67			нс
t _{XLWL}	Интервал между переключением BQ1 в низкое логическое состояние и переключением WR# в низкое логическое состояние	0			нс
t _{XLPH}	Интервал между переключением BQ1 в низкое логическое состояние и переключением PAGEL в высокое логическое состояние	0			нс
t _{PLXH}	Интервал между переключением PAGEL в низкое логическое состояние и переключением BQ1 в высокое логическое состояние	150			нс
t _{BVPH}	Предустановка BS1 к моменту перехода PAGEL в высокое логическое состояние	67			нс
t _{PHPL}	Ширина положительного импульса PAGEL	150			нс
t _{PLBX}	Удержание BS1 после перехода PAGEL в низкое логическое состояние	67			нс
t _{WL BX}	Удержание BS2/1 после перехода WR# в низкое логическое состояние	67			нс
t _{PLWL}	Интервал между переключением PAGEL в низкое логическое состояние и переключением WR# в низкое логическое состояние	67			нс
t _{BVWL}	Предустановка BS1 к моменту перехода WR# в низкое логическое состояние	67			нс
t _{WLWH}	Ширина отрицательного импульса WR#	150			нс
t _{WLRL}	Интервал между переключением WR# в низкое логическое состояние и переключением RDY/BSY# в низкое логическое состояние	0		1	мкс
t _{WLRH}	Интервал между переключением WR# в низкое логическое состояние и переключением RDY/BSY# в высокое логическое состояние ¹⁾	3,7		4,5	мс
t _{WLRH_CE} ²⁾	Интервал между переключением WR# в низкое логическое состояние и переключением RDY/BSY# в высокое логическое состояние для процедуры Chip Erase ²⁾	7,5		9	мс
t _{XLOL}	Интервал между переключением BQ1 в низкое логическое состояние и переключением OE# в низкое логическое состояние	0			нс
t _{BVDV}	Предустановка BS1 к моменту установки данных	0		250	нс
t _{OLDV}	Предустановка OE# в низкое логическое состояние к моменту установки данных			250	нс
t _{OHDZ}	Предустановка OE# в высокое логическое состояние к моменту переключению шины данных в состояние высокого импеданса.			250	нс

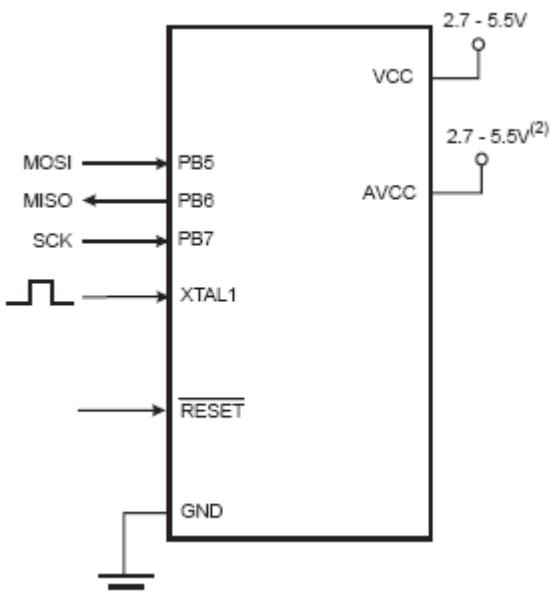
¹⁾ t_{WLRH} распространяется на команды Write Flash, Write EEPROM, Write Fuse (запись бит конфигурации), Write Lock Bits (запись бит блокировки).²⁾ t_{WLRH_CE} используется для команды Chip Erase.

3.20.6 Программирование по последовательному каналу

Оба массива Flash и EEPROM могут быть запрограммированы, используя последовательный SPI интерфейс и подключение вывода RESET# к низкому логическому уровню. Последовательный интерфейс включает выводы SCK, MOSI (вход) и MISO (выход). После переключения RESET# в низкое логическое состояние необходимо выполнить инструкцию Programming Enable перед проведением любых процедур программирования.

Таблица 3.96 – Назначение выводов при последовательном программировании

Символ	Вывод	Направление (I/O)	Описание
MOSI	PB5	I	Последовательный вход данных
MISO	PB6	O	Последовательный выход данных
SCK	PB7	I	Тактовый сигнал



Примечания

1 Если микросхема тактируется внутренним осциллятором, нет необходимости подключать внешний источник тактовых импульсов к выводу BQ1.

2 $U_{\#VCC} - 0,3 < U_{\#VCC} < U_{\#VCC} + 0,3$. Однако $U_{\#VCC}$ должен всегда находиться внутри диапазона (4,5 – 5,5) В.

Рисунок 3.117 – Включение схемы при последовательном программировании и верификации

При программировании EEPROM процедура автостирания всегда следует перед проведением операции программирования, поэтому нет необходимости выполнения команды Chip Erase (только для последовательного программирования!). Процедура Chip Erase устанавливает значения всех ячеек массивов Flash и EEPROM в значение 0xFF.

В зависимости от конфигурации CKSEL должен быть представлен соответствующий тактовый сигнал. Минимальная длительность положительного и отрицательного полупериодов тактовых импульсов SCK должна соответствовать 2-м периодам тактового импульса ЦПУ.

Алгоритм последовательного программирования

При записи последовательных данных события происходят по возрастающему фронту SCK.

При чтении последовательных данных события происходят по спадающему фронту SCK.

Ниже представлена рекомендуемая последовательность шагов при проведении последовательного программирования и верификации памяти программ и данных.

1 Последовательность подачи напряжения питания:

- питание подается при наличии низкого потенциала на выводах RESET# и SCK.

2 Ожидать в течение 20 мс и затем подать команду Programming Enable для разрешения последовательного программирования.

3 Последовательное программирование не будет выполняться, если подключение не синхронизировано. При синхронизации второй байт (0x53) будет давать отклик в то время, когда будет загружаться третий байт команды Programming Enable. Корректен отклик или нет, все четыре байта инструкции должны быть переданы. Если второй байт (0x53) не вызвал отклика, подайте положительный импульс RESET# и выполните загрузку новой команды Programming Enable.

4 Flash – память программируется постранично. Страница памяти загружается побайтно подачей 6 младших значимых бит адреса и данных вместе с командой Load Program Memory Page. Для уверенности корректной загрузки страницы, младший байт данных должен быть загружен перед тем, как старший байт данных назначен для данного адреса. Страница памяти программ сохраняется при загрузке команды Write Program Memory Page и подаче 8-ми старших значимых бит адреса. Если опрос статуса программирования не производится, пользователь должен выждать время не менее t_{WD_FLASH} перед выдачей новой страницы данных. Доступ к последовательному интерфейсу программирования до того как завершится процедура записи во Flash, может привести к некорректному выполнению процесса программирования.

5 Массив EEPROM программируется побайтно назначением адреса и данных вместе с соответствующей командой Write. Ячейка памяти EEPROM предварительно автоматически стирается перед процедурой записи новых данных. Если опрос об окончании процедуры программирования не производится, пользователь должен выждать не менее t_{WD_EEPROM} перед выполнением программирования следующего байта.

6 Любая ячейка памяти может быть верифицирована, используя команду чтения Read, которая выдаст содержимое выбранного адреса на последовательный вывод MISO.

7 По завершению процесса программирования вывод RESET# должен быть установлен в высокое логическое состояние для начала нормального режима функционирования ИМС.

8 Отключение питания (при необходимости):

- установить вывод RESET# в высокое логическое состояние;
- отключить напряжение питания от вывода #VCC.

Опрос данных Flash памяти

Когда страница данных программируется во Flash память, чтение адреса внутри страницы будет возвращать значение 0xFF. Когда устройство готово для загрузки следующей страницы, чтение будет возвращать корректные данные. Это используется для определения окончания процедуры записи страницы. Процедура опроса данных не работает для значения 0xFF, поэтому когда программируется это значение, пользователь должен выждать время не менее t_{WD_FLASH} перед тем, как програмировать следующую страницу.

Опрос данных массива EEPROM

Когда новый байт программируется в массив EEPROM, чтение адреса будет возвращать значение 0xFF. Когда устройство готово для загрузки следующей страницы, чтение будет возвращать корректные данные. Это используется для определения окончания процедуры записи страницы. Это не будет работать для значения 0xFF: процедура Chip Erase будет “записывать” во все ячейки памяти значение 0xFF, поэтому программирование адресов, которые должны содержать значение 0xFF может быть пропущено. Это не выполняется, если EEPROM перепрограммируется без Chip Erase процедуры. В этом случае, опрос данных не может быть использован для значения 0xFF, и пользователь должен будет выждать не менее t_{WD_EEPROM} перед программированием следующего байта данных.

Таблица 3.97 – Минимальные времена задержки перед проведением следующего цикла программирования Flash и EEPROM памяти

Символ	Минимальное время задержки, мс
t_{WD_FLASH}	4,5
t_{WD_EEPROM}	9
t_{WD_ERASE}	9
t_{WD_FUSE}	4,5

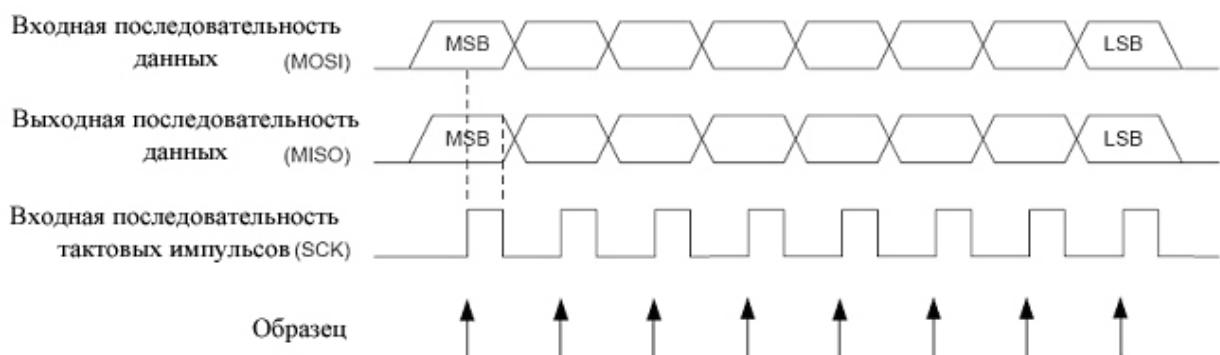


Рисунок 3.118 – Временная диаграмма выводов при последовательном программировании

Таблица 3.98 – Команды последовательного программирования

Команда	Формат инструкции				Операция
	Байт 1	Байт 2	Байт 3	Байт 4	
1	2	3	4	5	6
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Разрешение последовательного программирования после переключения RESET# в “0”
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Стирание flash и EEPROM массивов
Read program memory	0010 H000	0000 aaaa	bbbb bbbb	0000 0000	Чтение Н (старшего или младшего) байта данных o из памяти программ по адресу a:b
Load program memory	0100 H000	0000 xxxx	xxxb bbbb	iiii iiii	Запись Н (старшего или младшего) байта данных i в страничный буфер по адресу b

Продолжение таблицы 3.98

1	2	3	4	5	6
Write program memory	0100 1100	0000 aaaa	bbbb xxxx	xxxx xxxx	Запись страницы памяти программ по адресу a:b
Read EEPROM memory	1010 0000	00xx xxxx a	bbbb bbbb	0000 0000	Чтение данных i из EEPROM памяти по адресу a:b
Write EEPROM memory	1100 0000	00xx xxxx a	bbbb bbbb	iiii iiii	Запись данных i в память EEPROM по адресу a:b
Read Lock Bits	0101 1000	0000 0000	xxxx xxxx	xx00 0000	Чтение бит блокировки: “0” - запрограммирован, “1” – незапрограммирован
Write Lock Bits	1010 1100	111x xxxx	xxxx xxxx	11ii iiii	Запись бит блокировки: “0” - программирование бита.
Read signature Byte	0011 0000	00xx xxxx	xxxx xx bb	0000 0000	Чтение байта сигнатуры o по адресу b .
Write fuse bits	1010 1100	1010 0000	xxxx xxxx	iiii iiii	Запись младших конфигурационных бит. “0” - запрограммирован, “1” – незапрограммирован
Write fuse high bits	1010 1100	1010 1000	xxxx xxxx	iiii iiii	Запись младших конфигурационных бит. “0” - запрограммирован, “1” – незапрограммирован
Read fuse bits	0101 0000	0000 0000	xxxx xxxx	0000 0000	Чтение младших конфигурационных бит. “0” - запрограммирован, “1” – незапрограммирован
Read fuse high bits	0101 1000	0000 1000	xxxx xxxx	0000 0000	Чтение младших конфигурационных бит. “0” - запрограммирован, “1” – незапрограммирован
Read calibration byte	0011 1000	00xx xxxx	0000 00 bb	0000 0000	Чтение калибровочного байта

4 Адресация памяти

4.1 Прямая адресация

При прямой адресации адреса операндов содержатся непосредственно в слове команды. В соответствии со структурой памяти данных существуют следующие разновидности прямой адресации: прямая адресация одного регистра общего назначения (РОН), прямая адресация двух РОН, прямая адресация регистров ввода - вывода РВВ, прямая адресация ОЗУ.

4.1.1 Прямая адресация одного РОН

Этот способ адресации используется в командах, оперирующих с одним из регистров общего назначения. При этом адрес регистра-операнда (его номер) содержится в пяти разрядах слова команды.

Примером команд, использующих этот способ адресации, являются команды работы со стеком (PUSH, POP), команды инкремента (INC), декремента (DEC), а также некоторые команды арифметических операций.

4.1.2 Прямая адресация двух РОН

Этот способ адресации используется в командах, оперирующих одновременно с двумя регистрами общего назначения. При этом адрес регистра-источника содержится в разрядах 9, 3...0 (5 разрядов), а адрес регистра-приемника в разрядах 8...4 (5 разрядов) слова команды.

К командам, использующим этот способ адресации, относятся команда пересылки данных из регистра в регистр (MOV), а также большинство команд арифметических операций.

Здесь необходимо сделать одно замечание. Дело в том, что некоторые команды, имеющие только один регистр-операнд, тем не менее, используют рассматриваемый способ адресации. Просто в этом случае источником и приемником является один и тот же регистр. В качестве примера можно привести команду очистки регистра (CLR Rd), которая в действительности выполняет операцию “Исключающее ИЛИ”, регистра с самим собой (EOR Rd, Rd).

4.1.3 Прямая адресация РВВ

Данный способ адресации используется командами пересылки данных между регистром ввода-вывода, расположенного в основном пространстве ввода-вывода, и регистровым файлом IN и OUT. В этом случае адрес регистра ввода-вывода содержится в разрядах 10, 9, 3...0 (6 разрядов), а адрес РОН - в разрядах 8...4 (5 разрядов) слова команды.

4.1.4 Прямая адресация ОЗУ

Данный способ используется для обращения ко всему адресному пространству памяти данных.

В системе команд микроконтроллеров семейства имеется только две команды, использующие этот способ адресации. Это команды пересылки байта между одним из РОН и ячейкой ОЗУ - LDS и STS. Каждая из этих команд занимает в памяти программ два слова (32 разряда). В первом слове содержится код операции и адрес регистра общего назначения (в разрядах с 8-го по 4-й). Во втором слове находится адрес ячейки памяти, к которому происходит обращение.

4.2 Косвенная адресация

При косвенной адресации адрес ячейки памяти находится в одном из индексных регистров X, Y или Z, в зависимости от дополнительных манипуляций, которые производятся над содержимым индексного регистра, различают следующие разновидности косвенной адресации: простая косвенная адресация, относительная косвенная адресация, косвенная адресация с преддекрементом и косвенная адресация с постинкрементом.

4.2.1 Простая косвенная адресация

При использовании команд простой косвенной адресации обращение производится к ячейке памяти, адрес которой находится в индексном регистре. Никаких действий с содержимым индексного регистра при этом не производится.

Микроконтроллеры поддерживают 6 команд (по 2 для каждого индексного регистра) простой косвенной адресации: LD Rd, X/Y/Z (пересылка байта из ОЗУ в РОН) и ST X/Y/Z, Rd (пересылка байта из РОН в ОЗУ). Адрес регистра общего назначения содержится в разрядах 8...4 слова команды.

4.2.2 Относительная косвенная адресация

При использовании команд относительной косвенной адресации адрес ячейки памяти, к которой производится обращение, получается суммированием содержимого индексного регистра (Y или Z) и константой задаваемой в команде. Другими словами, производится обращение по адресу, указанному в команде, относительно адреса, находящегося в индексном регистре.

Микроконтроллеры семейства Mega поддерживают 4 команды относительной косвенной адресации (две для регистра Y и две для регистра Z): LDD Rd, Y+q/Z+q (пересылка байта из ОЗУ в РОН) и ST Y+q/Z+q, Rr (пересылка байта из РОН в ОЗУ). Адрес регистра общего назначения содержится в разрядах 8...4 слова команды, а величина смещения в разрядах 13, 11, 10, 2...0. Поскольку под значение смещения отводится только 6 разрядов, оно не может превышать 64.

4.2.3 Косвенная адресация с преддекрементом

При использовании команд косвенной адресации и с преддекрементом содержимое индексного регистра сначала уменьшается на 1, а затем производится обращение по полученному адресу.

Микроконтроллеры семейства поддерживают 6 команд по 2 для каждого индексного регистра) косвенной адресации с преддекрементом LD Rd, -X/-Y/-Z (пересылка байта из ОЗУ в РОН) и ST -X/-Y/-Z, Rd (пересылка байта из РОН в ОЗУ). Адрес регистра общего назначения содержится в разрядах 8...4 слова команды.

4.2.4 Косвенная адресация с постинкрементом

При использовании команд косвенной адресации с постинкрементом, после обращения по адресу, который находится в индексном регистре, содержимое индексного регистра увеличивается на 1.

Микроконтроллеры семейства поддерживают 6 команд (по 2 для каждого индексного регистра) косвенной адресации с постинкрементом: LD Rd, X+/Y+/Z+ (пересылка байта из ОЗУ в РОН) и ST X+/Y+/Z+, Rd (пересылка байта из РОН в ОЗУ). Адрес регистра общего назначения содержится в разрядах 8...4 слова команды.

5 Введение в систему команд ИМС 1887ВЕ1У

Система команд микроконтроллера насчитывает 133 различных инструкций. Несмотря на то, что микроконтроллер является микроконтроллером с RISC-архитектурой (процессор с сокращенным набором команд), по количеству реализованных инструкций и их разнообразию он больше похож на микроконтроллер с CISC-архитектурой (процессор с полным набором команд). Практически каждая из команд (за исключением команд, у которых одним из операндов является 16-разрядный адрес) занимает только одну ячейку памяти программ. Причем это достигнуто не за счет сокращения количества команд процессора, а за счет увеличения разрядности памяти программ.

Все множество команд микроконтроллера можно разбить на несколько групп:

- команды логических операций;
- команды арифметических операций и команды сдвига;
- команды операций с битами;
- команды пересылки данных;
- команды передачи управления;
- команды управления системой.

5.1 Команды логических операций

Команды логических операций позволяют выполнять стандартные логические операции над байтами, такие как логическое умножение (И), логическое сложение (ИЛИ), операцию «исключающее И», а также вычисление обратного (дополнение до единицы) и дополнительного (дополнение до двух) кодов числа. К этой группе можно отнести также команды очистки/установки регистров и команду перестановки тетрад. Операции производятся между регистрами общего назначения либо между регистром и константой; результат сохраняется в РОН. Все команды из этой группы выполняются за один машинный цикл.

5.2 Команды арифметических операций и команды сдвига

К данной группе относятся команды, позволяющие выполнять такие базовые операции, как сложение, вычитание, сдвиг (вправо и влево), инкремент и декремент. В микроконтроллере также имеются команды, позволяющие осуществлять умножение 8-разрядных значений. Все операции производятся только над регистрами общего назначения. При этом микроконтроллер позволяет легко оперировать как знаковыми, так и беззнаковыми числами, а также работать с числами, представленными в дополнительном коде.

Почти все команды рассматриваемой группы выполняются за один машинный цикл. Команды умножения и команды, оперирующие двухбайтовыми значениями, выполняются за два цикла.

5.3 Команды операций с битами

К данной группе относятся команды, выполняющие установку или сброс заданного разряда РОН или РВВ. Причем для изменения разрядов регистра состояния SREG имеются также дополнительные команды (точнее говоря, эквивалентные мнемонические обозначения общих команд), т. к. проверка состояния разрядов именно этого регистра производится чаще всего. Условно к этой группе можно отнести также две команды передачи управления типа «проверка/пропуск», которые пропускают следующую команду в зависимости от состояния разряда РОН или РВВ.

Все задействованные разряды РВВ имеют свои символические имена. Определения этих имен описаны в том же включаемом файле, что и определения символьических

имен адресов регистров. Таким образом, после включения в программу указанного файла в командах вместо числовых значений номеров разрядов можно будет указывать их символические имена. Следует помнить, что в командах CBR и SBR операндом является битовая маска, а не номер разряда. Для получения битовой маски из номера разряда следует воспользоваться ассемблерным оператором “сдвиг влево” (`<<<`), как показано в следующем примере:

```
sbr rl6, (1<<SE) + (1<<SM)
out MCUCR, rl6 ;Установить флаги SE и SM регистра MCUCR.
```

Всем командам данной группы требуется один машинный цикл для выполнения, за исключением случаев, когда в результате проверки происходит пропуск команды. В этом случае команда выполняется за 2 или 3 машинных цикла в зависимости от пропускаемой команды.

5.4 Команды пересылки данных

Команды этой группы предназначены для пересылки содержимого ячеек, находящихся в адресном пространстве памяти данных. Разделение адресного пространства на три части (РОН, РВВ, ОЗУ) предопределило разнообразие команд данной группы. Пересылка данных, выполняемая командами группы, может производиться в следующих направлениях:

- РОН <=> РОН;
- РОН <=> РВВ;
- РОН <=> память данных.

Также к данной группе можно отнести стековые команды PUSH и POP, позволяющие сохранять в стеке и восстанавливать из стека содержимое РОН.

На выполнение команд данной группы требуется, в зависимости от команды, от одного до трех машинных циклов.

5.5 Команды передачи управления

В эту группу входят команды перехода, вызова подпрограмм и возврата из них и команды типа «проверка/пропуск», пропускающие следующую за ними команду при выполнении некоторого условия. Также к этой группе относятся команды сравнения, формирующие флаги регистра SREG и предназначенные, как правило, для работы совместно с командами условного перехода.

В системе команд микроконтроллера имеются команды как безусловного, так и условного переходов. Команды относительного перехода (RJMP), а также косвенного (IJMP) и абсолютного (JMP) безусловного перехода являются самыми простыми в этой группе. Их функция заключается только в записи нового адреса в счетчик команд. Команды условного перехода также изменяют содержимое счетчика команд, однако это изменение происходит только при выполнении некоторого условия или, точнее, при определенном состоянии разрядных флагов регистра SREG.

Все команды условного перехода можно разбить на две подгруппы: первая подгруппа - команды условного перехода общего назначения, эту подгруппу входят две команды BRBS s,k и BRBC s, k, в которых явно задается номер тестируемого флага регистра SREG. Соответственно, переход осуществляется при SREG.s = 0 (BRBC) или REG.s = 1 (BRBS). Другую подгруппу составляют 18 специализированных команд, каждая из которых выполняет переход по какому-либо конкретному условию («равно», «больше или равно», «был перенос» и т. п.). Причем одни команды используются после сравнения беззнаковых чисел, другие — после сравнения чисел со знаком.

Команды вызова подпрограммы (ICALL, RCALL и CALL) работают практически так же, как и команды безусловного перехода. Отличие заключается в том, что, перед тем как выполнить переход, значение счетчика команд сохраняется в стеке. Кроме того,

подпрограмма должна заканчиваться командой возврата, как показано в следующем примере:

```

...
rcall sp_test      ; вызов подпрограммы sp_test
...
sp_test          ; текст основной программы
push r2           ; метка подпрограммы
push r2           ; сохранить r2 в стеке
...
pop r2            ; выполнение подпрограммы
pop r2            ; восстановить r2 из стека
ret               ; возврат из подпрограммы

```

В приведенном примере команда RET заменяет адрес, находящийся в счетчике команд, адресом команды, следующей за командой CALL.

Очевидно, что команды передачи управления нарушают нормальное (линейное) выполнение основной программы. Каждый раз, когда выполняется команда из этой группы (кроме команд сравнения), нормальное функционирование конвейера нарушается. Перед загрузкой в конвейер нового адреса производится остановка, и очистка выполняемой последовательности команд, соответственно, реинициализация конвейера приводит к необходимости использования нескольких машинных циклов для выполнения таких команд.

5.6 Команды управления системой

В эту группу входят всего 3 команды:

- NOP - пустая команда;
- SLEEP - перевод микроконтроллера в режим пониженного энергопотребления;
- WDR - сброс сторожевого таймера.

Все команды этой группы выполняются за один машинный цикл.

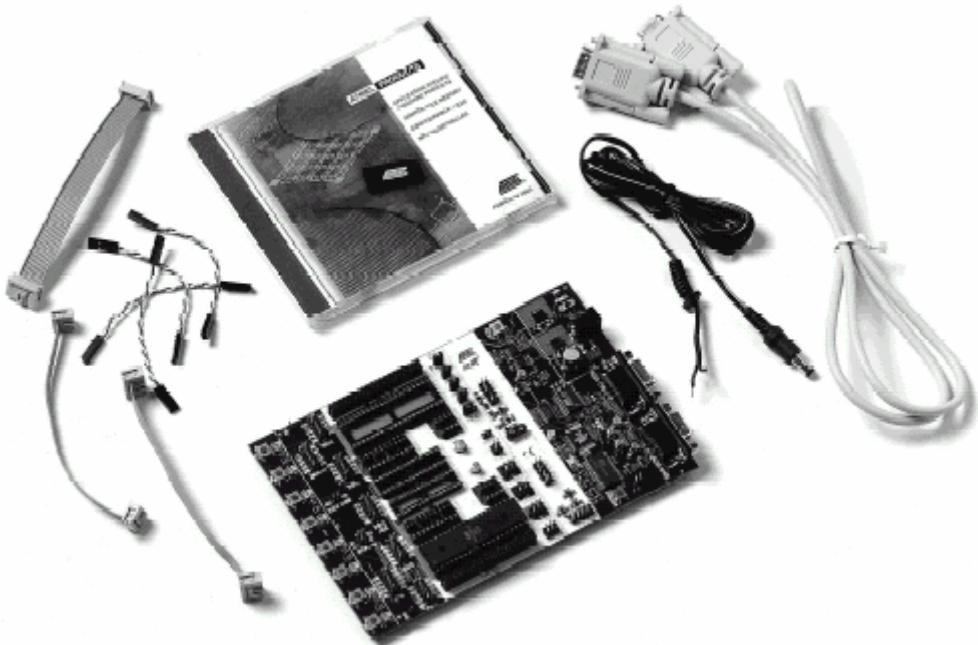
6 Отладочные средства

STK500 – завершенный стартовый набор, средство программирования и система проектирования для Atmel AVR микроконтроллеров. STK500 дает пользователям AVR устройств полную свободу в разработке и тестировании проектов на AVR устройствах и их макетов.

Отличительные особенности:

- Работа под управлением AVR Studio.
- Последовательное внутрисистемное программирование.
- Внутрисистемное программирование во внешней целевой системе.
- Параллельное и последовательное программирование с повышенным напряжением.
- Интерфейс RS-232 для связи с ПК.
- Разъемы 8-, 2П0-, 28- и 40-выводные AVR устройств.
- Гибкость тактирования, питания и системного сброса.
- Светодиоды и кнопка для исследований.
- Все порты ввода-вывода легкодоступны на разъеме.
- RS-232 драйвер и разъем.
- Обновление выполняется из AVR Studio.
- Разъемы расширения для вставляемых модулей и области макетирования.
- Целевое напряжение (1,8 – 6,0) В.
- Напряжение питания (9 – 12) В.

Внешний вид STK500



STK500 поддерживает все режимы программирования AVR микроконтроллеров, в т. ч. программирование на панелях и внутрисистемное программирование в целевой системе.

Интерфейс программирования STK500 интегрирован в AVR Studio. Флэш-память, ЭППЗУ, а также биты конфигурации и защиты могут программироваться индивидуально или с помощью опции автоматического программирования последовательности.

Частота тактирования и напряжение питания также управляются из AVR Studio. Программа для программирования под DOS также входит в комплект для эффективного программирования в производственной среде.

Активный код симулятора или эмулятора может быть легко загружен в STK500 выполнением щелчка мыши в AVR Studio.

Поддерживаются различные типы ИМС, в том числе: AT90S8535, ATmega8535 и др.

Примечания

1 Данное устройство поддерживается через ISP программирование внешнего цепевого устройства или с помощью модуля расширения STK501.

2 Данное устройство поддерживается через ISP программирование внешнего цепевого устройства или с помощью модуля расширения STK502.

3 Данное устройство поддерживается через ISP программирование внешнего цепевого устройства.

4 Также поддерживаются версии с пониженным питанием:

- поставляется с адаптером питания (100 – 240) В, 50/60 Гц.

Внутрисхемный эмулятор AVR ICE50

ICE50 и пользовательский интерфейс AVR Studio4 обеспечивают пользователю эффективное управление внутренними ресурсами микроконтроллера, уменьшение времени проектирования за счет упрощения процесса отладки.

Отличительные особенности:

- Работа под управлением AVR Studio (версия 4 и выше).
- Полная эмуляция как аналоговых функций, так и цифровых;.
- Встроенная самопроверка.
- Встроенный буфер трассировки (128K × 144) бит.
- Профилировщик кода и определение зоны кода.
- Интерфейс с ПК через USB и RS-232.
- Полная поддержка языков Ассемблер и Си.
- Неограниченное число точек прерывания программы.
- Реально-временное считывание памяти.
- Работа и прерывания программы выполняются в реальном времени.
- Обновление через AVR Studio.
- Целевое питание (2,2 – 5,5) В.

AVR Studio

AVR Studio - это интегрированная отладочная среда разработки приложений (IDE) для микроконтроллеров семейства AVR (AT90S, ATmega, ATTiny) фирмы Atmel. IDE AVR Studio содержит:

- транслятор языка ассемблера (Atmel AVR macroassembler);
- отладчик (Debugger);
- программное обеспечение верхнего уровня для поддержки внутрисхемного программирования (In-System Programming, ISP).

Отладчик AVR Studio поддерживает все типы микроконтроллеров AVR и имеет два режима работы: режим программной симуляции и режим управления различными типами внутрисхемных эмуляторов (In-Circuit Emulators) производства фирмы Atmel. Важно отметить, что интерфейс пользователя не изменяется в зависимости от выбранного режима отладки.

Указанные средства поставляются ООО "ЭФО".

Офис в г. Санкт - Петербург:
 194021, г. Санкт - Петербург, ул. Политехническая 21, офис 331
 Телефон: (812) 327-8654,
 Факс: (812) 320-1819
 E-mail: zav@efo.ru,
 www: <http://www.efo.ru>

Офис в г. Москва:
 127486, Москва, Коровинское шоссе, д.10, стр. 2, офис 30
 Телефон/факс: (495)933-0743
 E-mail: moscow@efo.ru,
 www: <http://www.efo.ru>

Средства программирования

Для программирования встроенной Flash-памяти многие фирмы выпускают программаторы.

Можно рекомендовать программатор *ChipProg+*, выпускаемый ОАО «Фитон», г. Москва. Дополнительно приобретается адаптер – переходник под соответствующий тип корпуса.

Программное обеспечение *ChipProg+* работает в среде WindowsTM. Интерфейс пользователя - многооконный, со свободной конфигурацией окон. Одновременно может быть открыто несколько буферов, содержащих различные данные для программирования.

ChipProg+ аппаратно реализован на базе микросхем загружаемой логики. Программно конфигурируемая аппаратура программатора позволяет легко расширять список программируемых устройств путем простого обновления версии программного обеспечения. Пополнение списка производится постоянно по мере появления новых типов микросхем.

Конструктивно программатор выполнен в виде автономного устройства размером (163×95×35) мм и подключается к персональному компьютеру через стандартный принтерный порт. Питание подается от сети переменного тока 220 В.



Описание программного обеспечения *ChipProg+* :

- Одна 40-выводная DIP-колодка с нулевым усилием.
- Дополнительные адаптеры для других типов корпуса (*приобретаются отдельно*).
- Связь с компьютером через принтерный порт.
- Программа управления под WINDOWS.
- Мощная программная поддержка.
- Полнофункциональный двоичный редактор.

- Самотестирование при включении питания.
- Тестирование правильности установки микросхемы.
- Работа с файлами в форматах: Standard/Extended intel HEX, Binary image, Motorola S.

- Представление данных в шестнадцатиричном, восьмеричном, двоичном, символьном (ASCII) форматах.

- Запись/чтение/верификация/контроль/стирание любой области ПЗУ.

- Полнофункциональный двоичный редактор:

- заполнение указанной области строкой данных;

- поиск и поиск/замена строки данных;

- инвертирование блока данных;

- копирование блока данных;

- сравнение блока данных.

- Подсчет контрольной суммы.

- Логические операции AND, OR, XOR с блоком данных.

Указанные средства поставляются ООО "Фирма Фитон" город Москва.

Реквизиты ООО "Фирма ФИТОН":

Адрес: Россия, Москва, 127015, ул. Новодмитровская, д. 5а, 11 этаж, оф. 1103

Телефон/факс: (495) 730-75-84 (многоканальный)

Электронная почта: PHYTON@phyton.ru

Интернет-телефон ([Skype](#)): phytonmsk

7 Заключение

В настоящем техническом описании КФДЛ.431295.025ТО приведено подробное описание архитектуры, функционального построения, системы команд и особенностей применения ИМС 1887ВЕ1У, которая представляет собой 8-разрядный микроконтроллер с AVR RISC-архитектурой, тактовой частотой 8 МГц, содержащий 8К байт Flash ЗУ программ, ЭСППЗУ (512×8) бит, ОЗУ (512 × 8) бит, 8-канальный 10-разрядный АЦП, последовательный периферийный интерфейс SPI, двухпроводной последовательный интерфейс TWI, последовательный порт USART.

Микроконтроллер предназначен для применения в системах управления и специальных применениях, для управления робототехническими комплексами, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, телекоммуникационной технике и т. д.

Все значения электрических параметров ИМС приведены в АЕЯР.431280.537ТУ на изделие 1887ВЕ1У. Значения параметров, приведенные в настоящем ТО, являются справочными.

КФДЛ.431295.025ТО может служить практическим руководством по применению микроконтроллера для разработчиков систем на основе ИМС 1887ВЕ1У и программистов.

Приложение А

(обязательное)

Описание системы команд

Таблица А.1 - Группа команд логических операций

Мнемоника	Описание	Операция	Циклы	Флаги
AND Rd, Rr	Логическое И двух РОН	$Rd = Rd * Rr$	1	Z, N, V
ANDI Rd, K	Логическое И РОН и константы	$Rd = Rd * K$	1	Z, N, V
EOR Rd, Rr	Исключающее ИЛИ двух РОН	$Rd = Rd \Delta Rr$	1	Z, N, V
OR Rd, Rr	Логическое ИЛИ двух РОН	$Rd = Rd \vee Rr$	1	Z, N, V
ORI Rd, K	Логическое ИЛИ РОН и константы	$Rd = Rd \vee K$	1	Z, N, V
COM Rd	Перевод в обратный код	$Rd = \$FF-Rd$	1	Z, N, V, C
NEG Rd	Перевод в дополнительный код	$Rd = \$00-Rd$	1	Z, N, V, C, H
CLR Rd	Сброс всех разрядов РОН	$Rd = Rd \wedge Rd$	1	Z, N, V
SER Rd	Установка всех разрядов РОН	$Rd = \$FF$	1	-
TST Rd	Проверка РОН на отрицательное и нулевое значение	$Rd * Rd$	1	Z, N, V
SWAP Rd	Обмен местами тетрад в РОН	$Rd[3:0] = Rd[7:4]$ $Rd[7:4] = Rd[3:0]$	1	-

Таблица А.2 - Группа команд арифметических операций

Мнемоника	Описание	Операция	Циклы	Флаги
ADD Rd, Rr	Сложение двух РОН	$Rd = Rd + Rr$	1	Z, C, N, V, H
ADC Rd, Rr	Сложение двух РОН с переносом	$Rd = Rd + Rr + C$	1	Z, C, N, V, H
ADIW Rd, K	Сложение регистровой пары с константой	$Rdh:Rdl = =Rdh:Rdl + K$	2	Z, C, N, V, S
SUB Rd, Rr	Вычитание двух РОН	$Rd = Rd - Rr$	1	Z, C, N, V, H
SUBI Rd, K	Вычитание константы из РОН	$Rd = Rd - K$	1	Z, C, N, V, H
SBC Rd, Rr	Вычитание двух РОН с заемом	$Rd = Rd - Rr - C$	1	Z, C, N, V, H
SBCI Rd, K	Вычитание константы из РОН с заемом	$Rd = Rd - K - C$	1	Z, C, N, V, H
SBIW Rd, K	Вычитание константы из регистровой пары	$Rdh:Rdl = Rdh:Rdl - K$	2	Z, C, N, V, S
DEC Rd	Декремент РОН	$Rd = Rd - 1$	1	Z, N, V
INC Rd	Инкремент РОН	$Rd = Rd + 1$	1	Z, N, V
ASR Rd	Арифметический сдвиг вправо	$Rd[n] = Rd[n+1], n=0..6$	1	Z, N, V, C
LSL Rd	Логический сдвиг влево	$Rd[n+1] = Rd[n], Rd[0]=0$	1	Z, N, V, C
LSR Rd	Логический сдвиг вправо	$Rd[n] = Rd[n+1], Rd[7]=0$	1	Z, N, V, C
ROL Rd	Сдвиг влево через перенос	$Rd[0]=C, Rd[n+1] = Rd[n], C = Rd[7]$	1	Z, N, V, C
ROR Rd	Сдвиг вправо через перенос	$Rd[7]=C, Rd[n] = Rd[n+1], C = Rd[0]$	1	Z, N, V, C
MUL Rd, Rr	Умножение беззнаковых чисел	$R1:R0 = Rd \times Rr$	2	Z, C
MULS Rd, Rr	Умножение чисел со знаком	$R1:R0 = Rd \times Rr$	2	Z, C
MULSU Rd, Rr	Умножение беззнакового числа на число со знаком	$R1:R0 = Rd \times Rr$	2	Z, C
FMUL Rd, Rr	Умножение дробных беззнаковых чисел	$R1:R0 = (Rd \times Rr) << 1$	2	Z, C
FMULS Rd, Rr	Умножение дробных чисел со знаком	$R1:R0 = (Rd \times Rr) << 1$	2	Z, C
FMULSU Rd, Rr	Умножение дробного беззнакового числа и дробного числа со знаком	$R1:R0 = (Rd \times Rr) << 1$	2	Z, C

Таблица А.3 - Группа команд операций над битами

Мнемоника	Описание	Операция	Циклы	Флаги
CBR Rd, K	Сброс разряда(ов) РОН	Rd = Rd*(\$FF-K)	1	Z, N, V
SBR Rd, K	Установка разряда(ов) РОН	Rd = Rd ^ K	1	Z, N, V
CBI A, b	Сброс разряда(ов) РВВ	A,b = 0	2	-
SBI A, b	Установка разряда(ов) РВВ	A,b = 1	2	-
BCLS s	Сброс флага	SREG,s = 0	1	SREG, s
BSET s	Установка флага	SREG,s = 1	1	SREG, s
BLD Rd, b	Загрузка разряда РОН из флага Т	Rd,b = T	1	-
BST Rd, b	Запись разряда РОН в флаг Т	T = Rr,b	1	T
CLC	Сброс флага переноса	C = 0	1	C
SEC	Установка флага переноса	C = 1	1	C
CLN	Сброс флага отр. числа	N = 0	1	N
SEN	Установка флага отр. числа	N = 1	1	N
CLZ	Сброс флага нуля	Z = 0	1	Z
SEZ	Установка флага нуля	Z = 1	1	Z
CLI	Общее запрещение прерываний	I = 0	1	I
SEI	Общее разрешение прерываний	I = 1	1	I
CLS	Сброс флага знака	S = 0	1	S
SES	Установка флага знака	S = 1	1	S
CLV	Сброс флага переполнения дополнения кода	V = 0	1	V
SEV	Установка флага переполнения дополнения кода	V = 1	1	V
CLT	Сброс флага Т	T = 0	1	T
SET	Установка флага Т	T = 1	1	T
CLH	Сброс флага половинного переноса	H = 0	1	H
SEH	Установка флага половинного переноса	H = 1	1	H

Таблица А.4 - Группа команд пересылки данных

Мнемоника	Описание	Операция	Циклы	Флаги
MOV Rd, Rr	Пересылка между РОН	Rd = Rr	2	-
MOVW Rd, Rr	Пересылка двухбайтовых значений	Rd+1:Rd = Rr+1:Rd	2	-
LDI Rd, K	Загрузка константы в РОН	Rd = K	2	-
LD Rd, X	Косвенное чтение	Rd = [X]	2	-
LD Rd, X+	Косвенное чтение с постинкрементом	Rd = [X], X = X + 1	2	-
LD Rd, -X	Косвенное чтение с преддекрементом	X = X - 1, Rd = [X]	2	-
LD Rd, Y	Косвенное чтение	Rd = [Y]	2	-
LD Rd, Y+	Косвенное чтение с постинкрементом	Rd = [Y], Y = Y + 1	2	-
LD Rd, -Y	Косвенное чтение с преддекрементом	Y = Y-1, Rd = [Y]	2	-
LDD Rd, Y+q	Косвенное относительное чтение	Rd = [Y+q]	2	-
LD Rd, Z	Косвенное чтение	Rd = [Z]	2	-
LD Rd, Z+	Косвенное чтение с постинкрементом	Rd = [Z], Z = Z + 1	2	-
LD Rd, -Z	Косвенное чтение с преддекрементом	Z = Z - 1, Rd[Z]	2	-
LDD Rd, Z+q	Косвенное относительное чтение	Rd = [Z+q]	2	-
LDS Rd, K	Непосредственное чтение из ОЗУ	Rd = [K]	2	-
ST X, Rr	Косвенная запись	[X] = Rr	2	-
ST X+, Rr	Косвенная запись с постинкрементом	[X] = Rr, X = X + 1	2	-
ST -X, Rr	Косвенная запись с преддекрементом	X = X - 1, [X] = Rr	2	-
ST Y, Rr	Косвенная запись	[Y] = Rr	2	-
ST Y+, Rr	Косвенная запись с постинкрементом	[Y] = Rr, Y = Y + 1	2	-
ST -Y, Rr	Косвенная запись с преддекрементом	Y = Y - 1, [Y] = Rr	2	-
STD Y+q, Rr	Косвенная относительная запись	[Y+q] = Rr	2	-
ST Z, Rr	Косвенная запись	[Z] = Rr	2	-
ST Z+, Rr	Косвенная запись с постинкрементом	[Z] = Rr, Z = Z + 1	2	-
ST -Z, Rr	Косвенная запись с преддекрементом	Z = Z - 1, [Z] = Rr	2	-
STD Z+q, Rr	Косвенная относительная запись	[Z+q] = Rr	2	-
STS K, Rr	Непосредственная запись в ОЗУ	[k] = Rr	2	-
LPM	Загрузка данных из памяти программ	R0 = {Z}	3	-
LPM, Rd, Z	Загрузка данных из памяти программ	Rb = {Z}	3	-
LPM Rd, Z+	Загрузка данных из памяти программ с постинкрементом	Rb = {Z}, Z = Z + 1	3	-
ELPM	Расширенная загрузка данных из памяти программ	R0 = {RAMPZ:Z}	3	-
ELPM Rd, Z	Расширенная загрузка данных из памяти программ	Rb = {RAMPZ:Z}	3	-
ELPM Rd, Z+	Расширенная загрузка данных из памяти программ с постинкрементом	Rb = {RAMPZ:Z}, RAMPZ:Z = RAMPZ:Z + 1	3	-
SPM	Запись в память программ	{Z} = R1:R0	-	-
IN Rd, A	Пересылка РВВ в РОН	Rd = A	1	-
OUT A, Rr	Пересылка РОН в РВВ	A = Rr	1	-
PUSH Rr	Сохранение байта в стеке	STACK = Rr	2	-
POP Rd	Извлечение байта из стека	Rd = STACK	2	-

Таблица А.5 - Группа команд передачи управления

Мнемоника	Описание	Операция	Циклы	Флаги
1	2	3	4	5
RJMP k	Относительный безусловный переход	PC = PC + K + 1	2	-
IJMP	Косвенный безусловный переход	PC = Z	2	-
JMP k	Абсолютный переход	PC = K	3	-
RCALL	Относительный вызов подпрограммы	PC = PC + K + 1	3	-
ICALL	Косвенный вызов подпрограммы	PC = Z	3	-
CALL k	Абсолютный вызов подпрограммы	PC = K	4	-
RET	Возврат из подпрограммы	PC = STACK	4	-
RETI	Возврат из подпрограммы обработки прерывания	PC = STACK	4	1
CP Rd, Rr	Сравнение РОН	Rd – Rr	1	Z, N, V, C, H
CPC Rd, Rr	Сравнение РОН с учетом переноса	Rd – Rr – C	1	Z, N, V, C, H
CPI Rd, K	Сравнение РОН с константой	Rd – K	1	Z, N, V, C, H
CPSE Rd, Rr	Сравнение и пропуск следующей команды при равенстве	При Rd = Rr PC = PC + 2(3)	1/2/3	-
SBRC Rr, b	Пропуск след. команды, если разряд РОН сброшен	При Rr.b = 0 PC = PC + 2(3)	1/2/3	-
SBRS Rr, b	Пропуск след. команды, если разряд РОН установлен	При Rr.b = 1 PC = PC + 2(3)	1/2/3	-
SBIC A, b	Пропуск след. команды, если разряд РВВ сброшен	При A.b = 0 PC = PC + 2(3)	1/2/3	-
SBIS A, b	Пропуск след. команды, если разряд РВВ установлен	При A.b = 1 PC = PC + 2(3)	1/2/3	-
BRBC s, k	Переход, если флаг регистра SREG сброшен	При SREG.s = 0 PC = PC + K + 1	1 / 2	-
BRBS s, k	Переход, если флаг регистра SREG установлен	При SREG.s = 1 PC = PC + K + 1	1 / 2	-
BRCS k	Переход по переносу	При C = 1 PC = PC + K + 1	1 / 2	-
BRCC k	Переход, если нет переноса	При C = 0 PC = PC + K + 1	1 / 2	-
BREQ k	Переход по равенству	При Z = 1 PC = PC + K + 1	1 / 2	-
BRNE k	Переход, если нет равенства	При Z = 0 PC = PC + K + 1	1 / 2	-
BRSH k	Переход по “больше или равно”	При C = 0 PC = PC + K + 1	1 / 2	-
BRLO k	Переход по “меньше”	При C = 1 PC = PC + K + 1	1 / 2	-
BRMI	Переход по “отрицательное значение”	При N = 1 PC = PC + K + 1	1 / 2	-
BRPL	Переход по “положительное значение”	При N = 0 PC = PC + K + 1	1 / 2	-
BRGE	Переход по “больше или равно” (числа со знаком)	При (N^V) = 0 PC = PC + K + 1	1 / 2	-
BRLT	Переход по “меньше или равно” (числа со знаком)	При (N^V) = 1 PC = PC + K + 1	1 / 2	-
BRHS	Переход по половинному переносу	При H = 1 PC = PC + K + 1	1 / 2	-
BRHC	Переход, если нет половинного переноса	При H = 0 PC = PC + K + 1	1 / 2	-

Окончание таблицы А.5

1	2	3	4	5-
BRTS	Переход, если флаг T установлен	При $T = 1$ $PC = PC + K + 1$	1 / 2	
BRTC	Переход, если флаг T сброшен	При $T = 0$ $PC = PC + K + 1$	1 / 2	-
BRVS	Переход по переполнению доп. кода	При $V = 1$ $PC = PC + K + 1$	1 / 2	-
BRVC	Переход, если нет переполнения доп. кода	При $V = 0$ $PC = PC + K + 1$	1 / 2	-
BRID	Переход, если прерывания запрещены	При $I = 1$ $PC = PC + K + 1$	1 / 2	-
BRIE	Переход, если прерывания разрешены	При $I = 0$ $PC = PC + K + 1$	1 / 2	-

Таблица А.6 - Группа команд управления системой

Мнемоника	Описание	Операция	Циклы	Флаги
NOP	Нет операции	-	1	-
SLEEP	Переход в “спящий” режим		3	-
WDR	Сброс сторожевого таймера		1	-

Принятые обозначения

Регистр статуса (SREG):

SREG:	Регистр статуса
C:	Флаг переноса
Z:	Флаг нулевого значения
N:	Флаг отрицательного значения
V:	Флаг - указатель переполнения дополнения до двух
S:	$N \oplus V$, Для проверок со знаком
H:	Флаг полупереноса
T:	Флаг пересылки, используемый командами BLD и BST
I:	Флаг разрешения/запрещения глобального прерывания

Регистры и операнды:

Rd:	Регистр назначения (и источник) в регистровом файле
Rr:	Регистр источник в регистровом файле
R:	Результат выполнения команды
K:	Литерал или байт данных (8 бит)
k:	Данные адреса константы для счетчика программ
b:	Бит в регистровом файле или I/O регистр (3 бита)
s:	Бит в регистре статуса (3 бита)
X, Y, Z:	Регистр косвенной адресации ($X=R27:R26$, $Y=R29:R28$, $Z=R31:R30$)
P:	Адрес I/O порта
q:	Смещение при прямой адресации (6 бит)

I/O регистры:

RAMPX, RAMPY, RAMPZ:	Регистры, связанные с X-, Y- и Z-регистрами, обеспечивающие косвенную адресацию всей области СОЗУ микроконтроллера с объемом СОЗУ более 64К байт
---	--

Стек:

STACK:	Стек для адреса возврата и опущенных в стек регистров
SP:	Указатель стека

Флаги:

<=>	Флаг, на который воздействует команда
0:	Очищенный командой Флаг
1:	Установленный командой флаг
-:	Флаг, на который не воздействует команда

Команда ADC - сложить с переносом

Описание:

Сложение двух регистров и содержимого флага переноса (C), размещение результата в регистре назначения Rd.

Операция:

(i) $Rd \leftarrow Rd + Rr + C$

Синтаксис

Операнды:

Счетчик программ:

(i) ADC Rd,Rr

$0 < d < 31$,
 $0 < r < 31$

$PC < PC + 1$

16-разрядный код операции:

0001	11rd	dddd	rrrr
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	<=>	<=>	<=>	<=>	<=>	<=>
---	---	-----	-----	-----	-----	-----	-----

H: $Rd3 * Rr3 + Rr3 + R3 + R3 * Rd3$

Устанавливается, если есть перенос из бита 3, в ином случае очищается

S: NEV, Для проверок со знаком

V: $Rd7 * Rr7 * R7 + Rd7 * Rr7 * R7$

Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $Rd7 * Rr7 * R7 * R7 * Rd7$

Устанавливается, если результат \$00, в ином случае очищается

C: $Rd7 * Rr7 + Rr7 + R7 + R7 * Rd7$

Устанавливается, если есть перенос из MSB результата, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

; Сложить R1 : R0 с R3 : R2

add r2, r0 ; Сложить младший байт

adc r3, r1 ; Сложить старший байт с переносом

Слов: 1 (2 байта)

Циклов: 1

Команда ADD - сложить без переноса

Описание:

Сложение двух регистров без добавления содержимого флага переноса (C), размещение результата в регистре назначения Rd.

Операция:

(i) $Rd \leftarrow Rd + Rr$

Синтаксис

Операнды:

Счетчик программ:

(i) ADD Rd,Rr

$0 < d < 31,$
 $0 < r < 31$

$PC < PC + 1$

16-разрядный код операции:

0000	11rd	dddd	rrrr
------	------	------	------

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	<=>	<=>	<=>	<=>	<=>	<=>

H: $Rd3 * Rr3 + Rr3 + R3 + R3 * Rd3$

Устанавливается, если есть перенос из бита 3, в ином случае очищается

S: NEV, Для проверок со знаком

V: $Rd7 * Rr7 * R7 + Rd7 * Rr7 * R7$

Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если результат \$00, в ином случае очищается

C: $Rd7 * Rr7 + Rr7 * R7 + R7 * Rd7$

Устанавливается, если есть перенос из MSB результата, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

add r1,r2 ; Сложить r2 с r1 ($r1=r1+r2$)

adc r28,r28 ; Сложить r28 с самим собой

($r28=r28+r28$)

Слов: 1 (2 байта)

Циклов: 1

Команда ADIW - сложить непосредственное значение со словом**Описание:**

Сложение непосредственного значения (0-63) с парой регистров и размещение результата в паре регистров. Команда работает с четырьмя верхними парами регистров, удобна для работы с регистрами указателями.

Операция:(i) $Rdh:Rdl \leftarrow Rdh:Rdl + K$

Синтаксис

Операнды:

Счетчик программ:

(i) ADIW Rdl,K dl O {24,26,28,30},
 0 < K < 63 PC < PC + 1

16-разрядный код операции:

1001	0110	KKdd	KKKK
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	<=>	<=>	<=>	<=>	<=>
---	---	---	-----	-----	-----	-----	-----

S: NEV, Для проверок со знаком**V:** Rdh7 R15

Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R15

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: R15*R14*R13*R12*R11*R10*R9*R8*R7*R6*R5*R4*R3*R2

Устанавливается, если результат \$0000, в ином случае очищается

C: R15*Rdh7

Устанавливается, если есть перенос из MSB результата, в ином случае очищается

R: (Результат) соответствует Rdh:Rdl после выполнения команды (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0 = R7-R0)

Пример:

adiw r24, 1 ; Сложить 1 с r25:r24

adiw r30, 63 ; Сложить 63 с Z указателем (r31 : r30)

Слов: 1 (2 байта)

Циклов: 2

Команда AND - выполнить логическое AND

Описание:

Выполнение логического AND между содержимым регистров Rd и Rr и помещение результата в регистр назначения Rd.

Операция:

(i) $Rd \leftarrow Rd * Rr$

Синтаксис

Операнды:

Счетчик программ:

(i) AND Rd,Rr

$0 < d < 31$,
 $0 < r < 31$

$PC \leftarrow PC + 1$

16-разрядный код операции:

0010	00rd	dddd	rrrr
------	------	------	------

Булевы выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	<=>	0	<=>	<=>	-
---	---	---	-----	---	-----	-----	---

S: NEV, Для проверок со знаком

V: 0

Очищен

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если результат \$00, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

and r2, r3 ; Поразрядное and r2 и r3, результат поместить в r2

ldi r16, 1 ; Установить маску 0000 0001 в r16

and r2, r16 ; Выделить бит 0 в r2

Слов: 1 (2 байта)

Циклов: 1

Команда ANDI - выполнить логическое AND с непосредственным значением

Описание:

Выполнение логического AND между содержимым регистра Rd и константой и помещение результата в регистр назначения Rd.

Операция:

(i) $Rd \leftarrow Rd * K$

Синтаксис

(i) ANDI Rd,K

Операнды:

$16 < d < 31,$
 $0 < K < 255$

Счетчик программ:

$PC \leftarrow PC + 1$

16-разрядный код операции:

0111	KKKK	dddd	KKKK
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	<=>	0	<=>	<=>	-

S: NEV, Для проверок со знаком

V: 0

Очищен

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

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

Устанавливается, если результат \$00, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

andi r17, \$0F ; Очистить старший nibбл r17

andi r18, \$10 ; Выделить бит 4 в r18

andi r19, \$AA ; Очистить нечетные биты r19

Слов: 1 (2 байта)

Циклов: 1

Команда ASR - арифметически сдвинуть вправо

Описание:

Выполнение сдвига всех битов Rd на одно место вправо. Состояние бита 7 не изменяется. Бит 0 загружается во флаг переноса (C) регистра состояния (SREG). Эта команда эффективно делит значение дополнения до двух на два, без изменения знака. Флаг переноса может быть использован для округления результата.

16-разрядный код операции:

1001	010d	dddd	0101
------	------	------	------

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	<=>	<=>	<=>	<=>	<=>

S: NEV, Для проверок со знаком

V: NEC (Для N и C после сдвига)

Устанавливается, если (N устанавливается и C очищается) или (N очищается, а C устанавливается). В ином случае очищается (при наличии значений N и C после сдвига)

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

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

Устанавливается, если результат \$00, в ином случае очищается

C: Rd0

Устанавливается, если перед сдвигом были установлены LSB или Rd

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
ldi r16, $10 ; Загрузить десятичное значение 16 в r16
asr r16 ; r16=r16 / 2
ldi r17, $FC ; Загрузить -4 в r17
asr r17 ; r17=r17 / 2
```

Слов: 1 (2 байта)

Циклов: 1

Команда BCLR - очистить бит в регистре статуса (SREG)

Описание:

Очистка одного флага в регистре статуса

Операция:

- (i) SREG(s) <- 0

Синтаксис Операнды: Счетчик программ:

- (i) BCLR s 0 < S < 7 PC <- PC + 1

16-разрядный код операции:

1001	0100	1sss	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

<=>	<=>	<=>	<=>	<=>	<=>	<=>	<=>
-----	-----	-----	-----	-----	-----	-----	-----

I: 0, если s = 7: в ином случае не изменяется**T:** 0, если s = 6: в ином случае не изменяется**H:** 0, если s = 5: в ином случае не изменяется**S:** 0, если s = 4: в ином случае не изменяется**V:** 0, если s = 3: в ином случае не изменяется**N:** 0, если s = 2: в ином случае не изменяется**Z:** 0, если s = 1: в ином случае не изменяется**C:** 0, если s = 0: в ином случае не изменяется

Пример:

```
bclr 0 ; Очистить флаг переноса
bclr 7 ; Запретить прерывания
```

Слов: 1 (2 байта)

Циклов: 1

Команда BLD - загрузить содержимое Т флага регистра статуса (SREG) в бит регистра

Описание:

Копирование содержимого Т флага регистра статуса в бит b регистра Rd

Операция:

(i) $Rd(b) \leftarrow T$

Синтаксис

Операнды:

Счетчик программ:

(i) BLD Rd,b

$0 < d < 31,$
 $0 < b < 7$

$PC \leftarrow PC + 1$

16-разрядный код операции:

1111	100d	dddd	0bbb
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

; Скопировать бит
bst r1, 0 ; Сохранить бит 2 регистра r1 во флаге T
bld r0, 4 ; Загрузить T в бит 4 регистра r0

Слов: 1 (2 байта)

Циклов: 1

Команда BRBC – перейти, если бит в регистре статуса очищен

Описание:

Условный относительный переход. Тестируется один из битов регистра статуса и, если бит очищен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If SREG(s) = 0 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

(i) BRBC	s, k	$0 < s < 7$, $-64 < k < +63$	$PC \leftarrow PC + 1$, если условия не соблюдены
----------	------	----------------------------------	--

16-разрядный код операции:

1111	01kk	kkkk	ksss
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

срі r20, 5 ;Сравнить r20 со значением 5
brbc 1,noteq ;Перейти, если флаг нуля очищен

.....
noteq: nop ;Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRBS – перейти, если бит в регистре статуса установлен

Описание:

Условный относительный переход. Тестируется один из битов регистра статуса и, если бит установлен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If SREG(s) = 1 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

(i) BRBS s, k	$0 \leq s \leq 7$,	$PC \leftarrow PC + 1$,
	$-64 < k < +63$	если условия не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	ksss
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

bst r0, 3 ;Загрузить T битом 3 регистра r0
brbs 6,bitset ;Перейти если бит T установлен

.....

bitset: nop ;Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRCC – перейти, если флаг переноса очищен

Описание:

Условный относительный переход. Тестируется бит флага переноса (C) регистра статуса и, если бит очищен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $C = 0$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

(i) BRCC k	$-64 < k < +63$	$PC \leftarrow PC + 1$ если условия не соблюдены
------------	-----------------	--

16-разрядный код операции:

1111	01kk	kkkk	k000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

```
add r22, r23 ; Сложить r23 с r22
brcc nocarry ; Перейти если перенос очищен
```

.....

nocarry: nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRCS – перейти, если флаг переноса установлен

Описание:

Условный относительный переход. Тестируется бит флага переноса (C) регистра статуса и, если бит установлен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $C=1$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

$PC \leftarrow PC + k + 1$

- (i) BRCS k $-64 < k < +63$ $PC \leftarrow PC + 1$,
если условия
не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

```
cpi r26, $56 ; Сравнить r26 с $56
brcs carry ; Перейти если перенос установлен
```

.....

carry: nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BREQ – перейти, если равно

Описание:

Условный относительный переход. Тестируется бит флага нулевого значения (Z) регистра статуса и, если бит установлен, выполняется переход относительно состояния счетчика программ. Если команда выполняется непосредственно после выполнения любой из команд CP, CPI, SUB или SUBI переход произойдет если, и только если, двоичное число, со знаком или без знака, представленное в Rd, эквивалентно двоичному числу, со знаком или без знака, представленному в Rr. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $Rd = Rr$ ($Z = 1$) then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

		$PC \leftarrow PC + k + 1$
(i) BREQ k	$-64 < k < +63$	$PC \leftarrow PC + 1$, если условия не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k001
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
cp r1, r0 ; Сравнить регистры r1 и r0
breq equal ; Перейти если содержимое регистров совпадает
```

.....

```
equal: nop ; Перейти по назначению (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRGE – перейти, если больше или равно (с учетом знака)**Описание:**

Условный относительный переход. Тестируется бит флага знака (S) регистра статуса и, еслибит очищен, выполняется переход относительно состояния счетчика программ. Если команда выполняется непосредственно после выполнения любой из команд CP, CPI, SUB или SUBI переход произойдет если, и только если, двоичное число, со знаком представленное в Rd, больше или эквивалентно двоичному числу со знаком, представленному в Rr. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ (PC-64 < назначение < PC+63). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $Rd > Rr$ ($NEV = 0$) then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис

Операнды:

Счетчик программ:

- (i) BRGE k -64 < k <+63

 $PC \leftarrow PC + k + 1$
 $PC \leftarrow PC + 1,$

если условия

не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k100
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

```
cp r11, r12 ; Сравнить регистры r11 и r12
brge greateq ; Перейти, если r11 >= r12 (со знаком)
```

.....

```
greateq: nop ; Перейти по назначению (пустая операция)
```

Слов: 1 (2 байта)**Циклов:** 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRHC – перейти, если флаг полупереноса очищен

Описание:

Условный относительный переход. Тестируется бит флага полупереноса (Н) регистра статуса и, если бит очищен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $H = 0$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

16-разрядный код операции:

1111	01kk	kkkk	k101
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

- - - - - - - -

Пример:

brhc hclear ; Перейти, если флаг полупереноса очищен

hclear: nop : Перейти по назначению (пустая операция)

Словъ 1 (2 байта)

Циклов: 1 - если условия не соблюdenы, 2 - при соблюдении правильных условий

Команда BRHS – перейти, если флаг полупереноса установлен

Описание:

Условный относительный переход. Тестируется бит флага полупереноса (H) регистра статуса и, если бит установлен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $H = 1$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

- (i) BRHS k $-64 < k < +63$ $PC \leftarrow PC + k + 1$
 $PC \leftarrow PC + 1$,
если условия
не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k101
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

brhs hset ; Перейти, если флаг полупереноса установлен

.....

hset: nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRID – перейти, если глобальное прерывание запрещено

Описание:

Условный относительный переход. Тестируется бит флага глобального прерывания (I) регистра статуса и, если бит сброшен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 << \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If I = 0 then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

- (i) BRID k $-64 < k < +63$ $PC \leftarrow PC + k + 1$
 $PC \leftarrow PC + 1$,
если условия
не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k111
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

brid intdis ; Перейти, если глобальное прерывание запрещено

.....
intdis: nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRIE – перейти, если глобальное прерывание разрешено

Описание:

Условный относительный переход. Тестируется бит флага глобального прерывания (I) регистра статуса и, если бит установлен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $I = 1$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

- (i) BRIE k $-64 < k < +63$ $PC \leftarrow PC + k + 1$
 $PC \leftarrow PC + 1$,
если условия
не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k111
------	------	------	------

Булевые выражения регистра статуса (SREG)

1	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

brie inten ;Перейти, если глобальное прерывание разрешено

.....

inten: nop ;Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRLO – перейти, если меньше (без знака)

Описание:

Условный относительный переход. Тестируется бит флага переноса (C) регистра статуса и, если бит установлен, выполняется переход относительно состояния счетчика программ. Если команда выполняется непосредственно после выполнения любой из команд CP, CPI, SUB или SUBI переход произойдет если, и только если, двоичное число без знака, представленное в Rd, меньше двоичного числа без знака, представленного в Rr. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $Rd < Rr$ ($C = 1$) then $PC <- PC + k + 1$, else $PC <- PC + 1$

Синтаксис	Операнды:	Счетчик программ:
(i) BRLO k	-64 < k < +63	$PC <- PC + k + 1$ $PC <- PC + 1$, если условия не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k000
------	------	------	------

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```

      eor r19, r19 ; Очистить r19
loop: inc r19    ; Увеличить на 1 r19
      ....
      cpi r19, $10 ; Сравнить r19 с $10
      brlo loop   ; Перейти, если r19 < $10 (без знака)
      nop       ; Выйти из петли (пустая операция)

```

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRLT - перейти, если меньше чем (со знаком)

Описание:

Условный относительный переход. Тестируется бит флага знака (S) регистра статуса и, если бит установлен, выполняется переход относительно состояния счетчика программы. Если команда выполняется непосредственно после выполнения любой из команд CP, CPI, SUB или SUBI переход произойдет, если, и только если, двоичное число со знаком, представленное в Rd, меньше двоичного числа со знаком, представленного в Rr. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $Rd < Rr$ ($N \square V = 1$) then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис	Операнды:	Счетчик программ:
(i) BRLT k	-64 < k < +63	$PC \leftarrow PC + k + 1$ $PC \leftarrow PC + 1$, если условия не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k100
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
cp r16, r1 ; Сравнить r16 с r1
brlt less ; Перейти, если r16 < r1 (со знаком)
```

```
.....
less nop ; Перейти по назначению (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRMI - перейти, если минус

Описание:

Условный относительный переход. Тестируется бит флага отрицательного значения (N) регистра статуса и, если бит установлен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $N = 1$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

- (i) BRMI k $-64 < k < +63$ $PC \leftarrow PC + k + 1$
 $PC \leftarrow PC + 1$,
 если условия
 не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k010
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

subi r18, 4 ; Вычесть 4 из r18
 brmi negative ; Перейти, если результат отрицательный

negative:
 nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRNE – перейти, если не равно

Описание:

Условный относительный переход. Тестируется бит флага нулевого значения (Z) регистра статуса и, если бит очищен, выполняется переход относительно состояния счетчика программ. Если команда выполняется непосредственно после выполнения любой из команд CP, CPI, SUB или SUBI, переход произойдет, если, и только если, двоичное число со знаком или без знака, представленное в Rd, не равно двоичному числу со знаком или без знака, представленному в Rr. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $Rd \neq Rr (Z = 0)$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис	Операнды:	Счетчик программ:
(i) BRNE k	$-64 < k < +63$	$PC \leftarrow PC + k + 1$ $PC \leftarrow PC + 1$, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k001
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```

      eor r27, r27 ; Очистить r27
loop: inc r27    ; Увеличить на 1 r27
      .....
      cpi r27, 5 ; Сравнить r27 с 5
      brne loop ; Перейти, если r27 <> 5
      por    ; Выйти из петли (пустая операция)
  
```

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRPL – перейти, если плюс

Описание:

Условный относительный переход. Тестируется бит флага отрицательного значения (N) регистра статуса и, если бит очищен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $N = 0$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

- (i) BRPL k $-64 < k < +63$ $PC \leftarrow PC + k + 1$
 $PC \leftarrow PC + 1$,
если условия
не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k010
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

subi r26, \$50 ; Вычесть \$50 из r26
brpl positive ; Перейти, если r26 положителен

positive:
nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRSH - перейти, если равно или больше (без знака)**Описание:**

Условный относительный переход. Тестируется бит флага перехода (C) регистра статуса и, если бит очищен, выполняется переход относительно состояния счетчика программ. Если команда выполняется непосредственно после выполнения любой из команд CP, CPI, SUB или SUBI, переход произойдет, если и только если, двоичное число без знака, представленное в Rd, больше или равно двоичному числу без знака, представленному в Rr. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ (PC-64 < назначение < PC+63). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $Rd > Rr$ ($C = 0$) then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис	Операнды:	Счетчик программ:
(i) BRSH k	-64 < k < +63	$PC \leftarrow PC + k + 1$ $PC \leftarrow PC + 1$, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

subi r19, 4 ; Вычесть 4 из r19
 brsh highsm ; Перейти, если r2 ≥ 4 (без знака)

.....
 highsm: nor ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRTC – перейти, если флаг T очищен

Описание:

Условный относительный переход. Тестируется бит флага пересылки (T) регистра статуса и, если бит очищен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $T = 0$ then then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис	Операнды:	Счетчик программ:
(i) BRTC k	$-64 < k < +63$	$PC \leftarrow PC + k + 1$ $PC \leftarrow PC + 1$, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k110
------	------	------	------

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

bst r3, 5 ; Сохранить бит 5 регистра r3 во флаге T
brtc tclear ; Перейти, если этот бит очищен

.....
tclear: por ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRTS – перейти, если флаг Т установлен

Описание:

Условный относительный переход. Тестируется бит флага пересылки (T) регистра статуса и, если бит установлен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $T = 1$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

- (i) BRTS k $-64 < k < +63$ $PC \leftarrow PC + k + 1$
 $PC \leftarrow PC + 1,$
если условия
не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k110
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

- - - - - - - -

Пример:

bst r3, 5 ; Сохранить бит 5 регистра r3 во флаге T
brts tset ; Перейти, если этот бит установлен

tset: nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1 - если условия не соблюдены, 2 - при соблюдении правильных условий

Команда BRVC – перейти, если переполнение очищено

Описание:

Условный относительный переход. Тестируется бит флага переполнения (V) регистра статуса и, если бит очищен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $V = 0$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

- (i) BRVC k $-64 < k < +63$ $PC \leftarrow PC + k + 1$
 $PC \leftarrow PC + 1$,
если условия
не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k011
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

add r3, r4 ; Сложить r4 с r3
brvc noover ; Перейти, если нет переполнения

.....
noover: nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BRVS – перейти, если переполнение установлено

Описание:

Условный относительный переход. Тестируется бит флага переполнения (V) регистра статуса и, если бит установлен, выполняется переход относительно состояния счетчика программ. Данная команда выполняет переход в любом направлении относительно состояния счетчика программ ($PC-64 < \text{назначение} < PC+63$). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

- (i) If $V = 1$ then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

- (i) BRVS k $-64 < k < +63$ $PC \leftarrow PC + k + 1$
 $PC \leftarrow PC + 1$,
 если условия
 не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k011
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

add r3, r4 ; Сложит r4 с r3
 brvs overfl ; Перейти, если есть переполнение

.....
 overfl: nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, 2 - при соблюдении правильных условий.

Команда BSET - установить бит в регистре статуса (SREG)**Описание:**

Установка одного флага в регистре статуса.

Операция:

(i) SREG(s)<-- 1

Синтаксис Операнды: Счетчик программ:

(i) BSET s 0 < s < 7 PC <- PC + 1

16-разрядный код операции:

1001	0100	0sss	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

<=>	<=>	<=>	<=>	<=>	<=>	<=>	<=>
-----	-----	-----	-----	-----	-----	-----	-----

I : 1, если s = 7: в ином случае не изменяется**T** : 1, если s = 6: в ином случае не изменяется**H** : 1, если s = 5: в ином случае не изменяется**S**: 1, если s = 4: в ином случае не изменяется**V**: 1, если s = 3: в ином случае не изменяется**N**: 1, если s = 2: в ином случае не изменяется**Z**: 1, если s = 1: в ином случае не изменяется**C**: 1, если s = 0: в ином случае не изменяется**Пример:**

bset 6 ; Установить флаг Т

bset 7 ; Разрешить прерывание

Слов: 1 (2 байта)

Циклов: 1

Команда BST - переписать бит из регистра во флаг T регистра статуса (SREG)

Описание:

Перезапись бита b из регистра Rd в флаг T регистра статуса (SREG)

Операция:

(i) $T \leftarrow Rd(b)$

Синтаксис Операнды: Счетчик программ:

(i) BST Rd,b $0 < d < 31, 0 < b < 7$ PC \leftarrow PC + 1

16-разрядный код операции:

1111	101d	dddd	Xbbb
------	------	------	------

Булевы выражения регистра статуса (SREG)

I T H S V N Z C

-	<=>	-	-	-	-	-	-
---	-----	---	---	---	---	---	---

T : 0, если бит b в Rd очищен: в ином случае устанавливается 1

Пример:

; Копировать бит

bst r1, 2 ; Сохранить бит 2 регистра r1 во флаге T

bld r0, 4 ; Загрузить T в бит 4 регистра r0

Слов: 1 (2 байта)

Циклов: 1

Команда CALL - выполнить длинный вызов подпрограммы

Описание:

Вызов подпрограммы из памяти программ. Адрес возврата (к команде после CALL) сохраняется в стеке.

Операция:

- (i) PC <-- Приборы с 16-разрядным счетчиком программ, максимальный объем памяти k программ 128K.
- (i) PC <-- Приборы с 22-разрядным счетчиком программ, максимальный объем памяти k программ 8M.

Синтаксис	Операнды:	Счетчик программ:
(i) CALL k	0 < k < 64K	PC <-- kSTACK <-- PC + 2 SP <-- SP-2, (2 байта, 16 битов)
(i) CALL k	0 < k < 4M	PC <-- kSTACK <-- PC + 2 SP <-- SP-3, (3 байта, 22 бита)

16-разрядный код операции:

1001	010k	kkkk	111k
kkkk	kkkk	kkkk	kkkk

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
mov r16, r0 ; Копировать r0 в r16
call check ; Вызвать подпрограмму
nop ; Продолжать (пустая операция)
```

```
...
check: cpi r16, $42 ; Проверить, содержит ли r16 заданное значение
breq error ; Перейти, если содержит
ret ; Вернуться из подпрограммы
```

```
...
error: rjmp error ; Бесконечная петля
```

Слов: 2 (4 байта)

Циклов: 4

Команда CBI - очистить бит в регистре I/O

Описание:

Очистка определенного бита в регистре ввода-вывода. Команда работает с младшими 32 регистрами ввода-вывода - адреса с 0 по 31.

Операция:

(i) I/O(P,b)<-- 0

Синтаксис

Операнды:

Счетчик программ:

(i) CBI P,b $0 < P < 31, 0 < b < 7$

$PC \leftarrow PC + 1$

16-разрядный код операции:

1001	1000	pppp	pbbb
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

cbi \$12.7 ; Очистить бит 7 в Порте D

Слов: 1 (2 байта)

Циклов: 2

Команда CBR - очистить биты в регистре

Описание:

Очистка определенных битов регистра Rd. Выполняется логическое AND между содержимым регистра Rd и комплементом постоянной K.

Операция:

(i) $Rd \leftarrow Rd * (\$FF - K)$

Синтаксис Операнды: Счетчик программ:

(i) CBR Rd $16 < d < 31, 0 < K < 255$ PC \leftarrow PC + 1

16-разрядный код операции:

Смотри команду ANDI с комплементом K

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	\leftrightarrow	0	\leftrightarrow	\leftrightarrow	-
---	---	---	-------------------	---	-------------------	-------------------	---

S: N^V , Для проверок со знаком

V: 0

Очищен

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7*R6*R5*R4*R3*R2*R1*R0$

Устанавливается, если результат \$00, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

cbr r16, \$F0 ; Очистить старший nibбл регистра r16

cbr r18, 1 ; Очистить бит в r18

Слов: 1 (2 байта)

Циклов: 1

Команда CLC - очистить флаг переноса в регистре статуса (SREG)

Описание:

Очистка флага переноса (C) в регистре статуса (SREG)

Операция:

(i) C <- 0

Синтаксис Операнды: Счетчик программ:

(i) CLC None PC <- PC + 1

16-разрядный код операции:

1001	0100	1000	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	0
---	---	---	---	---	---	---	---

C: 0

Флаг переноса очищен

Пример:

add r0, r0 ; Сложить r0 с самим собой
clc ; Очистить флаг переноса

Слов: 1 (2 байта)

Циклов: 1

Команда CLH - очистить флаг полупереноса в регистре статуса (SREG)

Описание:

Очистка флага полупереноса (H) в регистре статуса (SREG).

Операция:

- (i) H <-- 0

Синтаксис Операнды: Счетчик программ:

- (i) CLH None PC <- PC + 1

16-разрядный код операции:

1001	0100	1101	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	0	-	-	-	-	-
---	---	---	---	---	---	---	---

H: 0

Флаг полупереноса очищен

Пример:

clh ; Очистить флаг полупереноса

Слов: 1 (2 байта)

Циклов: 1

Команда CLI - очистить флаг глобального прерывания в регистре статуса (SREG)

Описание:

Очистка флага глобального прерывания (I) в регистре статуса (SREG).

Операция:

(i) $I \leftarrow 0$

Синтаксис Операнды: Счетчик программ:

(i) CLI None $PC \leftarrow PC + 1$

16-разрядный код операции:

1001	0100	1111	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

0	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

I: 0

Флаг глобального прерывания очищен

Пример:

```
cli      ; Запретить прерывания
in r11, $16 ; Считать Порт В
sei      ; Разрешить прерывания
```

Слов: 1 (2 байта)

Циклов: 1

Команда CLN - очистить флаг отрицательного значения в регистре статуса (SREG)

Описание:

Очистка флага отрицательного значения (N) в регистре статуса (SREG).

Операция:

(i) $N \leftarrow 0$

Синтаксис Операнды: Счетчик программ:

(i) CLN None PC \leftarrow PC + 1

16-разрядный код операции:

1001	0100	1010	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	0	-	-
---	---	---	---	---	---	---	---

N: 0

Флаг отрицательного значения очищен

Пример:

add r2, r3 ; Сложить r3 с r2

cln ; Очистить флаг отрицательного значения

Слов: 1 (2 байта)

Циклов: 1

Команда CLR - очистить регистр

Описание:

Очистка регистра. Команда выполняет Exclusive OR содержимого регистра с самим собой. Это приводит к очистке всех битов регистра.

Операция:

(i) $Rd \leftarrow Rd \square Rd$

Синтаксис Операнды: Счетчик программ:

(i) CLR Rd $0 < d < 31$ $PC \leftarrow PC + 1$

16-разрядный код операции:

0010	01dd	dddd	dddd
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	0	0	0	1	-

S: 0

Очищен

V: 0

Очищен

N: 0

Очищен

Z: 1

Устанавливается

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
clr r18 ; Очистить r18
loop: inc r18 ; Увеличить на 1 r18
```

```
...
cpi r18, $50 ; Сравнить r18 с $50
brne loop
```

Слов: 1 (2 байта)

Циклов: 1

Команда CLS - очистить флаг знака

Описание:

Очистка флага знака (S) в регистре статуса (SREG).

Операция:

(i) S <-- 0

Синтаксис Операнды: Счетчик программ:

(i) CLS None PC <- PC + 1

16-разрядный код операции:

1001	0100	1100	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	0	-	-	-	-
---	---	---	---	---	---	---	---

S: 0

Очищен

Пример:

```
add r2, r3 ; Сложить r3 с r2
cls        ; Очистить флаг знака
```

Слов: 1 (2 байта)

Циклов: 1

Команда CLT - очистить Т флаг

Описание:

Очистка флага пересылки (Т) в регистре статуса (SREG).

Операция:

(i) T <-- 0

Синтаксис Операнды: Счетчик программ:

(i) CLT None PC <- PC + 1

16-разрядный код операции:

1001	0100	1110	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	0	-	-	-	-	-	-
---	---	---	---	---	---	---	---

T: 0

Очищен

Пример:

clt ; Очистить Т флаг

Слов: 1 (2 байта)

Циклов: 1

Команда CLV - очистить флаг переполнения

Описание:

Очистка флага переполнения (V) в регистре статуса (SREG).

Операция:

- (i)
- $V \leftarrow 0$

Синтаксис Операнды: Счетчик программ:

- (i) CLV None PC
- \leftarrow
- PC + 1

16-разрядный код операции:

1001	0100	1011	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	0	-	-	-

V: 0

Очищен

Пример:

```
add r2, r3 ; Сложить r3 с r2
clv      ; Очистить флаг переполнения
```

Слов: 1 (2 байта)

Циклов: 1

Команда CLZ - очистить флаг нулевого значения

Описание:

Очистка флага нулевого значения (Z) в регистре статуса (SREG).

Операция:

(i) $Z \leftarrow 0$

Синтаксис Операнды: Счетчик программ:

(i) CLZ None PC \leftarrow PC + 1

16-разрядный код операции:

1001	0100	1001	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	0	-
---	---	---	---	---	---	---	---

Z: 0

Очищен

Пример:

```
add r2, r3 ; Сложить r3 с r2
clz ; Очистить флаг нулевого значения
```

Слов: 1 (2 байта)

Циклов: 1

Команда СОМ - выполнить дополнение до единицы

Описание:

Команда выполняет дополнение до единицы (реализует обратный код) содержимого регистра Rd.

Операция:

(i) $Rd \leftarrow \$FF * Rd$

Синтаксис Операнды: Счетчик программ:

(i) COM Rd $0 < d < 31$ PC $\leftarrow PC + 1$

16-разрядный код операции:

1001	010d	dddd	0000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	<=>	0	<=>	<=>	1

S: N^V , Для проверок со знаком

V: 0

Очищен

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7*R6*R5*R4*R3*R2*R1*R0$

Устанавливается, если результат \$00, в ином случае очищается

C: 1

Установлен

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
com r4 ; Выполнить дополнение до единицы r4
breq zero ; Перейти, если ноль
```

```
...
zero:    nop      ; Перейти по назначению (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1

Команда СР - сравнить

Описание:

Команда выполняет сравнение содержимого двух регистров Rd и Rr. Содержимое регистров не изменяется. После этой команды можно выполнять любые условные переходы.

Операция:

(i) $Rd = Rr$

Синтаксис Операнды: Счетчик программ:

(i) Cp Rd, Rr $0 < d < 31$, $0 < r < 31$ PC \leftarrow PC + 1

16-разрядный код операции:

0001	01rd	dddd	rrrr
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	<=>	<=>	<=>	<=>	<=>	<=>

H: $Rd3 * Rr3 + Rr3 * R3 + R3 * Rd3$

Устанавливается, если есть заем из бита 3, в ином случае очищается

S: N^V , Для проверок со знаком

V: $Rd7 * Rd7 * R7 + Rd7 * Rr7 * R7$

Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $Rd7 * Rr7 + Rr7 * R7 + R7 * Rd7$

Устанавливается, если результат \$00, в ином случае очищается

C: $Rd7 * Rr7 + Rr7 * R7 + R7 * Rd7$

Устанавливается, если абсолютное значение Rr больше абсолютного значения Rd, в ином случае очищается

R: (Результат) после выполнения команды

Пример:

```
ср r4, r19 ; Сравнить r4 с r19
brne noteq ; Перейти, если r4 <> r19
```

...

noteq: пор ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1

Команда CPC – сравнить с учетом переноса

Описание:

Команда выполняет сравнение содержимого двух регистров Rd и Rr и учитывает также предшествовавший перенос. Содержимое регистров не изменяется. После этой команды можно выполнять любые условные переходы.

Операция:

(i) $Rd = Rr = C$

Синтаксис Операнды: Счетчик программ:

(i) CPC Rd, Rr $0 < d < 31$, $0 < r < 31$ $PC \leftarrow PC + 1$

16-разрядный код операции:

0000	01rd	dddd	rrrr
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	<=>	<=>	<=>	<=>	<=>	<=>

H: $Rd3 * Rr3 + Rr3 * R3 + R3 * Rd3$

Устанавливается, если есть заем из бита 3, в ином случае очищается

S: N^V , Для проверок со знаком

V: $Rd7 * Rd7 * R7 + Rd7 * Rr7 * R7$

Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0 * Z$

Предшествующее значение остается неизменным, если результатом является ноль, в ином случае очищается

C: $Rd7 * Rr7 + Rr7 * R7 + R7 * Rd7$

Устанавливается, если абсолютное значение Rr плюс предшествовавший перенос больше абсолютного значения Rd, в ином случае очищается

R: (Результат) после выполнения команды

Пример:

```
; Сравнить r3 : r2 с r1 : r0
cp r2, r0 ; Сравнить старший байт
cpc r3, r1 ; Сравнить младший байт
brne noteq ; Перейти, если не равно
```

```
...  
noteq: por ; Перейти по назначению (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1

Команда CPI - сравнить с константой

Описание:

Команда выполняет сравнение содержимого регистра Rd с константой. Содержимое регистра не изменяется. После этой команды можно выполнять любые условные переходы.

Операция:

(i) $Rd = K$

Синтаксис Операнды: Счетчик программ:

(i) CPI Rd, K $16 \leq d \leq 31$, $PC \leftarrow PC + 1$
 $0 < K < 255$

16-разрядный код операции:

0011	KKKK	dddd	KKKK
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	<=>	<=>	<=>	<=>	<=>	<=>

H: $Rd3 * K3 + K3 * R3 + R3 * Rd3$

Устанавливается, если есть заем из бита 3, в ином случае очищается

S: N^V , Для проверок со знаком

V: $Rd7 * K7 * R7 + Rd7 * K7 * R7$

Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если результат \$00, в ином случае очищается

C: $Rd7 * K7 + K7 * R7 + R7 * Rd7$

Устанавливается, если абсолютное значение K больше абсолютного значения Rd, в ином случае очищается

R: (Результат) после выполнения команды

Пример:

```
cpi r19, 3 ; Сравнить r19 с 3
brne error ; Перейти, если r4 <> 3
```

...

error: por ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1

Команда CPSE - сравнить и пропустить, если равно

Описание:

Команда выполняет сравнение содержимого регистров Rd и Rr и пропускает следующую команду, если Rd = Rr.

Операция:

- (i) If $Rd = Rr$ then $PC \leftarrow PC + 2$ (or 3), else $PC \leftarrow PC + 1$

Синтаксис Операнды: Счетчик программ:

- (i) CPSE Rd,Rr $0 < d < 31$,
 $0 < r < 31$

PC<- PC + 1, если
условия не соблюдены,
то пропуска нет

PC<- PC + 2, пропуск
одного слова команды

PC<- PC + 3, пропуск
двух слов команды

16-разрядный код операции:

0001	00rd	dddd	rrrr
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

- - - - - - - -

Пример:

```
inc r4      ; Увеличить на 1 r4
cpse r4, r0 ; Сравнить r4 с r0
neg r4      ; Выполнить, если r4 <> r0
nop         ; Продолжать (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1

Команда DEC - декрементировать

Описание:

Вычитание единицы - 1 - из содержимого регистра Rd и размещение результата в регистре назначения Rd. Флаг переноса регистра статуса данной командой не активируется, что позволяет использовать команду DEC при реализации счетчика циклов для вычислений с повышенной точностью. При обработке чисел без знаков за командой могут выполняться переходы BREQ и BRNE. При обработке значений в форме дополнения до двух допустимы все учитывающие знак переходы.

Операция:

- (i) $Rd \leftarrow Rd - 1$

Синтаксис Операнды: Счетчик программ:

- (i) DEC Rd $0 < d < 31$ $PC \leftarrow PC + 1$

16-разрядный код операции:

1001	010d	dddd	1010
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	<=>	<=>	<=>	<=>	-

S: N^V , Для проверок со знаком

V: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если в результате получено переполнение дополнения до двух, в ином случае очищается. Переполнение дополнения до двух будет, если и только если перед операцией содержимое Rd было \$80.

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если результат \$00, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
ldi r17, $10 ; Загрузить константу в r17
loop: add r1, r2 ; Сложить r2 с r1
      dec r17 ; Уменьшить на 1 r17
      brne loop ; Перейти, если r17 <> 0
      nop      ; Продолжать (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1

Команда EOR - выполнить исключающее OR

Описание:

Выполнение логического исключающего OR между содержимым регистра Rd и регистром Rr и помещение результата в регистр назначения Rd.

Операция:

(i) $Rd \leftarrow Rd \wedge Rr$

Синтаксис Операнды: Счетчик программ:

(i) EOR Rd,Rr $0 < d < 31$, $0 < r < 31$ PC $\leftarrow PC + 1$

16-разрядный код операции:

0010	01rd	dddd	rrrr
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	<=>	0	<=>	<=>	-

S: $N \wedge V$, Для проверок со знаком

V: 0

Очищен

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если результат \$00, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

eor r4, r4 ; Очистить r4

eor r0, r22 ; Поразрядно выполнить исключающее or между r0 и r22

Слов: 1 (2 байта)

Циклов: 1

Команда ICALL - вызвать подпрограмму косвенно

Описание:

Косвенный вызов подпрограммы, указанной регистром-указателем Z (16 разрядов) в регистровом файле. Регистр-указатель Z (16-разрядного формата) позволяет вызывать подпрограмму из текущей секции пространства памяти программ объемом 64К слов (128 Кбайт).

Операция:

- (i) PC(15-0)<-- Z(15-0) Приборы с 16-разрядным счетчиком программ, максимальный объем памяти программ 128К.
- (i) PC(15-0)<-- Z(15-0) Приборы с 22-разрядным счетчиком программ, максимальный объем памяти программ 8М. PC(21-16) не изменяются

Синтаксис Операнды: Счетчик программ: Стек

- | | | | |
|-----------|------|--------------|--|
| (i) ICALL | None | См. Операция | STACK<-- PC + 1
SP<-- SP-2, (2 байта, 16 битов) |
| (i) ICALL | None | См. Операция | STACK<-- PC + 1
SP<-- SP-3, (3 байта, 22 бита) |

16-разрядный код операции:

1001	0101	XXXX	1001
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

mov r30, r0 ; Установить смещение в таблицу вызовов
icall ; Вызвать подпрограмму, указанную r31 : r30

Слов: 1 (2 байта)

Циклов: 3

Команда IJMP - перейти косвенно

Описание:

Выполняется косвенный переход по адресу, указанному регистром-указателем Z (16 разрядов) в регистровом файле. Регистр-указатель Z (16-разрядного формата) позволяет вызвать подпрограмму из текущей секции пространства памяти программ объемом 64К слов (128 Кбайт).

Операция:

- (i) PC<-- Z(15-0) Приборы с 16-разрядным счетчиком программ, максимальный объем памяти программ 128K.
- (ii) PC(15-0)<-- ZZ(15-0) Приборы с 22-разрядным счетчиком программ, максимальный объем памяти программ 8M. PC(21-16) не изменяются

Синтаксис Операнды: Счетчик программ: Стек

- | | | | |
|-------------|------|--------------|------------------|
| (ii) IJMP | None | См. Операция | Не задействуется |
| (iii) ICALL | None | См. Операция | Не задействуется |

16-разрядный код операции:

1001	0100	XXXX	1001
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
    mov r30, r0 ; Установить смещение в таблицу переходов
    ijmp      ; Перейти к подпрограмме, указанной r31 : r30
```

Слов: 1 (2 байта)

Циклов: 2

Команда IN - загрузить данные из порта I/O в регистр

Описание:

Команда загружает данные из пространства входа/выхода (порты, таймеры, регистры конфигурации и т. п.) в регистр Rd регистрационного файла.

Операция:

(i) $Pd \leftarrow P$

Синтаксис Операнды: Счетчик программ:

(i) IN Rd,P $0 < d < 31, 0 < P < 63$ $PC \leftarrow PC + 1$

16-разрядный код операции:

1011	0PPd	dddd	PPPP
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

in r25, \$16 ; Считать Порт В
cpi r25, r4 ; Сравнить считанное значение с константой
breq exit ; Перейти, если r25=4

...

exit: nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1

Команда INC - инкрементировать

Описание:

Добавление единицы - 1 - к содержимому регистра Rd и размещение результата в регистре назначения Rd. Флаг переноса регистра статуса данной командой не активируется, что позволяет использовать команду DEC при реализации счетчика циклов для вычислений с повышенной точностью. При обработке чисел без знаков за командой могут выполняться переходы BREQ и BRNE. При обработке значений в форме дополнения до двух допустимы все учитывающие знак переходы.

Операция:

- (i) $Rd \leftarrow Rd + 1$

Синтаксис Операнды: Счетчик программ:

- (i) INC Rd $0 < d < 31$ $PC \leftarrow PC + 1$

16-разрядный код операции:

1001	010d	dddd	0011
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

S: N^V , Для проверок со знаком

V: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если в результате получено переполнение дополнения до двух, в ином случае очищается. Переполнение дополнения до двух будет, если и только если перед операцией содержимое Rd было \$7F.

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если результат \$00, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

```

clr r22 ; Очистить r22
loop: inc r22 ; Увеличить на 1 r22
      ...
      cpi r22, $4F ; Сравнить r22 с $4F
      brne loop ; Перейти, если не равно
      nop ; Продолжать (пустая операция)

```

Слов: 1 (2 байта)

Циклов: 1

Команда JMP - перейти

Описание:

Выполняется переход по адресу внутри всего объема (4М слов) памяти программ.
См. также команду RJMP.

Операция:

(i) Pd<-- k

Синтаксис Операнды: Счетчик программ: Стек

(i) JMP k 0 < k < 4M PC<-- k Не изменяется

32-разрядный код операции:

1001	010k	kkkk	110k
kkkk	kkkk	kkkk	kkkk

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

mov r1, r0 ; Копировать r0 в r1
jmp farplc ; Безусловный переход

farplc: nop ; Перейти по назначению (пустая операция)

Слов: 2 (4 байта)

Циклов: 4

Команда LD - загрузить косвенно из СОЗУ в регистр с использованием индекса X

Описание:

Загружает косвенно один байт из СОЗУ в регистр. Положение байта в СОЗУ указывается 16-разрядным регистром-указателем X в регистровом файле. Обращение к памяти ограничено текущей страницей объемом 64К байта. Для обращения к другой странице СОЗУ необходимо изменить регистр RAMPX в I/O области. Регистр-указатель X может оставаться неизменным после выполнения команды, но может быть инкрементирован или декрементирован. Использование регистра-указателя X обеспечивает удобную возможность обращения к матрицам, таблицам, указателю стека.

Использование X-указателя:

Операция: Комментарий:

- | | | |
|-------------------|-------------|-----------------------------------|
| (i) Rd <-- (X) | | X: Неизменен |
| (ii) Rd <-- (X) | X <-- X + 1 | X: Инкрементирован впоследствии |
| (iii) X <-- X - 1 | Rd <-- (X) | X: Предварительно декрементирован |

Синтаксис

- | | |
|-----------------|------------|
| (i) LD Rd,X | 0 < d < 31 |
| (ii) LD Rd,X+ | 0 < d < 31 |
| (iii) LDD Rd,-X | 0 < d < 31 |

Операнды:

Счетчик программ:

- | |
|-----------|
| PC<-- + 1 |
| PC<-- + 1 |
| PC<-- + 1 |

16-разрядный код операции:

(i)	1001	000d	dddd	1100
(ii)	1001	000d	dddd	1101
(iii)	1001	000d	dddd	1110

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

```

    clr r27 ;Очистить старший байт X
    ldi r26, $20 ;Установить $20 в младший байт X
    ld r0, X+ ;Загрузить в r0 содержимое SRAM по адресу $20 (X постинкремен-
тируется)
    ld r1, X ;Загрузить в r1 содержимое SRAM по адресу $21
    ldi r26, $23 ;Установить $23 в младший байт X
    ld r2, X ;Загрузить в r2 содержимое SRAM по адресу $23
    ld r3, -X ;Загрузить в r3 содержимое SRAM по адресу $22 (X преддекремен-
тируется)
  
```

Слов: 1 (2 байта)

Циклов: 2

Команда LD (LDD) - загрузить косвенно из СОЗУ в регистр с использованием индекса Y

Описание:

Загружает косвенно, со смещением или без смещения один байт из СОЗУ в регистр. Положение байта в СОЗУ указывается 16-разрядным регистром-указателем Y в регистровом файле. Обращение к памяти ограничено текущей страницей объемом 64К байта. Для обращения к другой странице СОЗУ необходимо изменить регистр RAMPY в I/O области. Регистр-указатель Y может оставаться неизменным после выполнения команды, но может быть инкрементирован или декрементирован. Использование регистра-указателя Y обеспечивает удобную возможность обращения к матрицам, таблицам, указателю стека.

Использование Y-указателя:

Операция: Комментарий:

- | | | |
|---------------------|-------------|-----------------------------------|
| (i) Rd <-- (Y) | | Y: Неизменен |
| (ii) Rd <-- (Y) | Y <-- Y + 1 | Y: Инкрементирован впоследствии |
| (iii) Y <-- Y + 1 | Rd <-- (Y) | Y: Предварительно декрементирован |
| (iv) Rd <-- (Y + q) | | Y: Неизменен, q: смещение |

Синтаксис

Операнды:

Счетчик программ:

- | | | |
|--------------------|--------------------------|-----------|
| (i) LD Rd,Y | 0 < d < 31 | PC<-- + 1 |
| (ii) LD Rd,Y+ | 0 < d < 31 | PC<-- + 1 |
| (iii) LD Rd,-Y | 0 < d < 31 | PC<-- + 1 |
| (iv) LDD Rd, Y + q | 0 < d < 31
0 < q < 63 | PC<-- + 1 |

16-разрядный код операции:

(i)	1000	000d	dddd	1000
(ii)	1001	000d	dddd	1001
(iii)	1001	000d	dddd	1010
(iv)	10q0	qq0d	dddd	1qqq

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

```

clr r29 ;Очистить старший байт Y
ldi r28, $20 ;Установить $20 в младший байт Y
ld r0, Y+ ;Загрузить в r0 содерж. SRAM по адресу $20 (Y постинкрементируется)
ld r1, Y ;Загрузить в r1 содержимое SRAM по адресу $21
ldi r28, $23 ;Установить $23 в младший байт Y
ld r2, Y ;Загрузить в r2 содержимое SRAM по адресу $23
ld r3, -Y ;Загрузить в r3 содерж. SRAM по адресу $22 (Y преддекрементируется)
ldd r4, Y+2 ;Загрузить в r4 содержимое SRAM по адресу $24

```

Слов: 1 (2 байта)

Циклов: 2

Команда LD (LDD) - загрузить косвенно из СОЗУ в регистр с использованием индекса Z

Описание:

Загружает косвенно, со смещением или без смещения один байт из СОЗУ в регистр. Положение байта в СОЗУ указывается 16-разрядным регистром-указателем Z в регистровом файле. Обращение к памяти ограничено текущей страницей объемом 64К байта. Для обращения к другой странице СОЗУ необходимо изменить регистр RAMPZ в I/O области. Регистр-указатель Z может остаться неизменным после выполнения команды, но может быть инкрементирован или декрементирован. Эта особенность очень удобна при использовании регистра-указателя Z в качестве указателя стека, однако, поскольку регистр-указатель Z может быть использован для косвенного вызова подпрограмм, косвенных переходов и табличных преобразований, более удобно использовать в качестве указателя стека регистры-указатели X и Y. Об использовании указателя Z для просмотра таблиц в памяти программ, см. команду LPM.

Использование Z-указателя:

Операция: Комментарий:

- | | | |
|---------------------|-------------|-----------------------------------|
| (i) Rd <-- (Y) | | Y: Неизменен |
| (ii) Rd <-- (Y) | Y <-- Y + 1 | Y: Инкрементирован впоследствии |
| (iii) Y <-- Y + 1 | Rd <-- (Y) | Y: Предварительно декрементирован |
| (iv) Rd <-- (Y + q) | | Y: Неизменен, q: смещение |

Синтаксис

Операнды:

Счетчик программ:

- | | | |
|--------------------|--------------------------|-----------|
| (i) LD Rd,Y | 0 < d < 31 | PC<-- + 1 |
| (ii) LD Rd,Y+ | 0 < d < 31 | PC<-- + 1 |
| (iii) LD Rd,-Y | 0 < d < 31 | PC<-- + 1 |
| (iv) LDD Rd, Y + q | 0 < d < 31
0 < q < 63 | PC<-- + 1 |

16-разрядный код операции:

(i)	1000	000d	dddd	1000
(ii)	1001	000d	dddd	1001
(iii)	1001	000d	dddd	1010
(iv)	10q0	qq0d	dddd	1qqq

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

```

clr r29    ;Очистить старший байт Y
ldi r28,$20 ;Установить $20 в младший байт Y
ld r0,Y+   ;Загрузить в r0 содерж. SRAM по адресу $20 (Y постинкрементируется)
ld r1,Y    ;Загрузить в r1 содержимое SRAM по адресу $21
ldi r28,$23 ;Установить $23 в младший байт Y

```

ld r2, Y ;Загрузить в r2 содержимое SRAM по адресу \$23
 ld r3, -Y ;Загрузить в r3 содержимое SRAM по адресу \$22 (Y преддекрементируется)
 ldd r4, Y+2 ;Загрузить в r4 содержимое SRAM по адресу \$24

Слов: 1 (2 байта)

Циклов: 2

Команда LDI - загрузить непосредственное значение

Описание:

Загружается 8-разрядная константа в регистр от 16 по 31

Операция:

(i) Rd <-- K

Синтаксис

Операнды:

Счетчик программ:

(i) LDI Rd, K $16 < d < 31,$
 $0 < K < 255$

PC<-- + 1

16-разрядный код операции:

1110	KKKK	dddd	KKKK
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

clr r31 ; Очистить старший байт Z
 ldi r30, \$F0 ; Установить \$F0 в младший байт Z
 lpm ; Загрузить константу из программы
 ; Память отмечена в Z

Слов: 1 (2 байта)

Циклов: 1

Команда LDS - загрузить непосредственно из СОЗУ

Описание:

Выполняется загрузка одного байта из СОЗУ в регистр. Можно использовать 16-разрядный адрес. Обращение к памяти ограничено текущей страницей СОЗУ объемом 64К байта. Команда LDS использует для обращения к памяти выше 64К байт регистр RAMPZ.

Операция:

(i) Rd <-- (k)

Синтаксис

Операнды:

Счетчик программ:

(i) LDS Rd,k 0 < d < 31, 0 < k < 65535

PC<-- + 2

32-разрядный код операции:

1001	000d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

```
lds r2, $FF00 ; Загрузить r2 содержимым SRAM по адресу $FF00
add r2, r1 ; Сложить r1 с r2
sts $FF00, r2 ; Записать обратно
```

Слов: 2 (4 байта)

Циклов: 3

Команда LPM - загрузить байт памяти программ**Описание:**

Загружает один байт, адресованный регистром Z, в регистр 0 (R0). Команда обеспечивает эффективную загрузку констант или выборку постоянных данных. Память программ организована из 16-разрядных слов и младший значащий разряд (LSB) 16-разрядного указателя Z выбирает или младший (0), или старший (1) байт. Команда может адресовать первые 64К байта (32К слов) памяти программ.

Операция:

(i) R0<-- (Z)

Комментарий:

Z указывает на память программ

Синтаксис	Операнды:	Счетчик программ:
(i) LPM	None	PC<-- + 1

16-разрядный код операции:							
1001	0101		110X		1000		

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
clr r31      ; Очистить старший байт Z.
ldi r30, $F0 ; Установить младший байт Z
lpm          ; Загрузить константу из памяти программ
               отмеченную Z (r31 : r30)
```

Слов: 1 (2 байта)

Циклов: 3

Команда LSL - логически сдвинуть влево

Описание:

Выполнение сдвига всех битов Rd на одно место влево. Бит 0 стирается. Бит 7 загружается во флаг переноса (C) регистра состояния (SREG). Эта команда эффективно умножает на два значение величины без знака.

Операция:

16-разрядный код операции:

0000	11dd	dddd	dddd
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	<=>	<=>	<=>	<=>	<=>	<=>

H: Rd3

S: N^V, Для проверок со знаком

V: N^C (Для N и C после сдвига)

Устанавливается, если (N устанавливается и C очищается) или (N очищается, а C устанавливается). В ином случае очищается (при наличии значений N и C после сдвига)

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

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

Устанавливается, если результат \$00, в ином случае очищается

C: Rd7

Устанавливается, если перед сдвигом был установлен MSB регистра Rd в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
add r0, r4 ; Сложить r4 с r0
lsl r0      ; Умножить r0 на 2
```

Слов: 1 (2 байта)

Циклов: 1

Команда LSR - логически сдвинуть вправо

Описание:

Сдвиг всех битов Rd на одно место вправо. Бит 7 очищается. Бит 0 загружается во флаг переноса (C) регистра состояния (SREG). Эта команда эффективно делит на два величину без знака на два. Флаг переноса может быть использован для округления результата.

16-разрядный код операции:

1001	010d	dddd	0110
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	<=>	<=>	0	<=>	<=>

S: N^V, Для проверок со знаком

V: N^C (Для N и C после сдвига)

Устанавливается, если (N устанавливается и C очищается) или (N очищается, а C устанавливается). В ином случае очищается (при наличии значений N и C после сдвига)

N: 0

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

Устанавливается, если результат \$00, в ином случае очищается

C: Rd0

Устанавливается, если перед сдвигом был установлен LSB регистра Rd, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
add r0, r4 ; Сложить r4 с r0
lsr r0    ; Разделить r0 на 2
```

Слов: 1 (2 байта)

Циклов: 1

Команда MOV - копировать регистр

Описание:

Команда создает копию одного регистра в другом регистре. Исходный регистр Rr остается неизменным, в регистр назначения Rd загружается копия содержимого регистра Rr.

(i) $Rd \leftarrow Rr$

Синтаксис	Операнды:	Счетчик программ:
(i) $MOV\ Rd,Rr$	$0 < d < 31, 0 < r < 31$	$PC \leftarrow +1$

16-разрядный код операции:

0010	11rd	dddd	rrrr
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
mov r16, r0 ; Копировать r0 в r16
call check ; Вызвать подпрограмму
...
check cpi r16, $11 ; Сравнить r16 с $11
...
ret ; Вернуться из подпрограммы
```

Слов: 1 (2 байта)

Циклов: 1

Команда MUL - перемножить

Описание:

Команда перемножает две 8-разрядные величины без знаков с получением 16-разрядного результата без знака. Множимое и множитель - два регистра - Rr и Rd, соответственно. 16-разрядное произведение размещается в регистрах R1 (старший байт) и R0 (младший байт). Следует отметить, что если в качестве множимого и множителя выбрать R0 или R1, то результат заместит прежние значения сразу после выполнения операции.

Операция:

(i) $R1, R0 \leftarrow Rr \times Rd$

Синтаксис

Операнды:

Счетчик программ:

(i) MUL Rd,Rr $0 < d < 31, 0 < r < 31$

$PC \leftarrow PC + 1$

16-разрядный код операции:

1001	11rd	dddd	r000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	<=>
---	---	---	---	---	---	---	-----

C: R15

Устанавливается, если установлен бит 15 результата, в ином случае очищается

R: (Результат) соответствует R1, R0 после выполнения команды

Пример:

```
mul r6, r5 ; Перемножить r6 и r5
mov r6, r1 ; Вернуть результат обратно в r6:r5
mov r5, r1 ; Вернуть результат обратно в r6:r5
```

Слов: 1 (2 байта)

Циклов: 2

Команда NEG - выполнить дополнение до двух

Описание:

Заменяет содержимое регистра Rd его дополнением до двух. Значение \$80 остается неизменным.

Операция:

(i) $Rd \leftarrow \$00 - Rd$

Синтаксис

Операнды:

Счетчик программ:

(i) NEG Rd

$0 < d < 31$

$PC \leftarrow PC + 1$

16-разрядный код операции:

1001	010d	dddd	0001
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow	\leftrightarrow
---	---	-------------------	-------------------	-------------------	-------------------	-------------------	-------------------

H: R3*Rd3

Устанавливается, если есть заем из бита 3, в ином случае очищается

S: N^V , Для проверок со знаком

V: R7*R6*R5*R4*R3*R2*R1*R0

Устанавливается при переполнении дополнения до двух от подразумеваемого вычитания из нуля, в ином случае очищается. Переполнение дополнения до двух произойдет, если и только если содержимое регистра после операции (результат) будет \$80.

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: Rd7*R6*R5*R4*R3*R2*R1*R0

Устанавливается, если результат \$00, в ином случае очищается

C: R7+R6+R5+R4+R3+R2+R1+R0

Устанавливается, если есть заем в подразумеваемом вычитании из нуля, в ином случае очищается. Флаг C будет устанавливаться во всех случаях, за исключением случая, когда содержимое регистра после выполнения операции будет \$80.

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
sub r11, r0 ; Вычесть r0 из r11
brpl positive ; Перейти, если результат положительный
neg r11      ; Выполнить дополнение до двух r11
positive: nop      ; Перейти по назначению (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1

Команда NOP - выполнить холостую команду

Описание:

Команда выполняется за один цикл без выполнения операции.

(i) No

Синтаксис	Операнды:	Счетчик программ:
(i) NOP	None	PC<-- + 1

16-разрядный код операции:

0000	0000	0000	0000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```

clr r16 ; Очистить r16
ser r17 ; Установить r17
out $18, r16 ; Записать ноль в Порт В
nop ; Ожидать (пустая операция)
out $18, r17 ; Записать 1 в Порт В

```

Слов: 1 (2 байта)

Циклов: 1

Команда OR - выполнить логическое OR

Описание:

Команда выполняет логическое OR содержимого регистров Rd и Rr и размещает результат в регистре назначения Rd.

Операция:

(i) $Rd \leftarrow Rd \vee Rr$

Синтаксис

Операнды:

Счетчик программ:

(i) OR Rd,Rr $0 < d < 31, 0 < r < 31$

$PC \leftarrow PC + 1$

16-разрядный код операции:

0010	10rd	dddd	rtrr
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	<=>	0	<=>	<=>	-
---	---	---	-----	---	-----	-----	---

S: N^V , Для проверок со знаком

V: 0

Очищен

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $Rd7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если результат \$00, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
or r15, r16 ; Выполнить поразрядное or между регистрами
bst r15, 6 ; Сохранить бит 6 регистра 15 во флаге T
brst ok ; Перейти, если флаг T установлен
```

```
...
ok: nop ; Перейти по назначению (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1

Команда ORI - выполнить логическое OR с непосредственным значением

Описание:

Команда выполняет логическое OR между содержимым регистра Rd и константой и размещает результат в регистре назначения Rd.

Операция:

(i) $Rd \leftarrow Rd \vee K$

Синтаксис

(i) ORI Rd,K

Операнды:

$16 < d < 31, 0 < K < 255$

Счетчик программ:

$PC \leftarrow PC + 1$

16-разрядный код операции:

0110	KKKK	dddd	KKKK
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	<=>	0	<=>	<=>	-
---	---	---	-----	---	-----	-----	---

S: N^V , Для проверок со знаком

V: 0

Очищен

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $Rd7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если результат \$00, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
ori r16, $F0 ; Установить старший nibбл r16
ori r17, 1 ; Установить бит 0 регистра r17
```

Слов: 1 (2 байта)

Циклов: 1

Команда OUT - записать данные из регистра в порт I/O

Описание:

Команда сохраняет данные регистра Rr в регистровом файле пространства I/O (порты, таймеры, регистры конфигурации и т. п.).

Операция:

(i) $P \leftarrow Rr$

Синтаксис

Операнды:

Счетчик программ:

(i) OUT P,Rr $0 < r < 31, 0 < P < 63$

$PC \leftarrow +1$

16-разрядный код операции:

1011	1PPr	rrrr	PPPP
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
clr r16 ; Очистить r16
ser r17 ; Установить r17
out $18, r16 ; Записать нули в Порт В
nop ; Ожидать (пустая операция)
out $18, r17 ; Записать единицы в Порт В
```

Слов: 1 (2 байта)

Циклов: 1

Команда POP - записать регистр из стека

Описание:

Команда загружает регистр Rd байтом содержимого стека.

Операция:

(i) $Rd \leftarrow \text{STACK}$

Синтаксис

Операнды:

Счетчик программ:

(i) POP Rd

 $0 < d < 31$ $PC \leftarrow +1$ $SP \leftarrow SP + 1$

16-разрядный код операции:

1001	000d	dddd	1111
------	------	------	------

Булевые выражения регистра состояния (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

call routine ; Вызвать подпрограмму

...

routine: push r14 ; Сохранить r14 в стеке

push r13 ; Сохранить r13 в стеке

...

pop r13 ; Восстановить r13

pop r14 ; Восстановить r14

ret ; Вернуться из подпрограммы

Слов: 1 (2 байта)

Циклов: 2

Команда PUSH - поместить регистр в стек

Описание:

Команда помещает содержимое регистра Rd в стек.

Операция:

(i) STACK <-- Rr

Синтаксис

Операнды:

Счетчик программ:

(i) PUSH Rr

0 < d < 31

PC<-- + 1

SP<-- SP - 1

16-разрядный код операции:

1001	001d	dddd	1111
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

call routine ; Вызвать подпрограмму

...

routine: push r14 ; Сохранить r14 в стеке

push r13 ; Сохранить r13 в стеке

...

pop r13 ; Восстановить r13

pop r14 ; Восстановить r14

ret ; Вернуться из подпрограммы

Слов: 1 (2 байта)

Циклов: 2

Команда RCALL - вызвать подпрограмму относительно

Описание:

Команда вызывает подпрограмму в пределах 2K слов (4K байт). Адрес возврата (после выполнения команды RCALL) сохраняется в стеке (См. также команду [CALL](#)).

Операция:

- (i) PC <-- PC + k + 1 Приборы с 16-разрядным счетчиком команд, максимум 128К байт памяти программ
- (ii) PC <-- PC + k + 1 Приборы с 22-разрядным счетчиком команд, максимум 8М байт памяти программ

Синтаксис	Операнды:	Счетчик программ:	Стек
(i) RCALL k	-2K < k < 2K	PC <-- PC + k + 1	STACK <-- PC + 1 SP <-- SP-2 (2 байта, 16 бит)
(ii) RCALL k	-2K < k < 2K	PC <-- PC + k + 1	STACK <-- PC + 1 SP <-- SP-3 (3 байта, 22 бита)

16-разрядный код операции:

1101	kkkk	kkkk	kkkk
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

rcall routine ; Вызвать подпрограмму

...

routine: push r14 ; Сохранить r14 в стеке

...

pop r14 ; Восстановить r14

ret ; Вернуться из подпрограммы

Слов: 1 (2 байта)

Циклов: 3

Команда RET - вернуться из подпрограммы

Описание:

Команда возвращает из подпрограммы. Адрес возврата загружается из стека.

Операция:

- (i) PC(15-0) <- STACK Приборы с 16-разрядным счетчиком команд, максимум 128К байт памяти программ
- (ii) PC(21-0) <- STACK Приборы с 22-разрядным счетчиком команд, максимум 8М байт памяти программ

Синтаксис	Операнды:	Счетчик программ:	Стек
(i) RET	None	См. операцию	SP <- SP+2 (2 байта, 16 бит)
(ii) RET	None	См. операцию	SP <- SP+3 (3 байта, 22 бита)

16-разрядный код операции:

1001	0101	0XX0	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

call routine ; Вызвать подпрограмму

...

routine: push r14 ; Сохранить r14 в стеке

...

pop r14 ; Восстановить r14

ret ; Вернуться из подпрограммы

Слов: 1 (2 байта)

Циклов: 4

Команда RETI - вернуться из прерывания

Описание:

Команда возвращает из прерывания. Адрес возврата выгружается из стека и устанавливается флаг глобального прерывания.

Операция

- (i) PC(15-0) <- STACK Приборы с 16-разрядным счетчиком команд, максимум 128К байт памяти программ
- (ii) PC(21-0) <- STACK Приборы с 22-разрядным счетчиком команд, максимум 8М байт памяти программ

Синтаксис	Операнды:	Счетчик программ:	Стек
(i) RETI	None	См. операцию	SP <- SP+2 (2 байта, 16 бит)
(ii) RETI	None	См. операцию	SP <- SP+3 (3 байта, 22 бита)

16-разрядный код операции:

1001	0101	0XX1	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
1	-	-	-	-	-	-	-

I: 1

Флаг установлен

Пример:

```
...  
extint: push r0 ; Сохранить r0 в стеке  
...  
pop r0 ; Восстановить r0  
reti ; Вернуться и разрешить прерывания
```

Слов: 1 (2 байта)

Циклов: 4

Команда RJMP - перейти относительно

Описание:

Команда выполняет относительный переход по адресу в пределах 2К слов (4К байт) текущего состояния счетчика команд. В ассемблере вместо относительных операндов используются метки. Для AVR микроконтроллеров с памятью программ не превышающей 4К слов (8К байт) данная команда может адресовать всю память программ.

Операция

(i) $PC \leftarrow PC + k + 1$

Синтаксис	Операнды:	Счетчик программ:	Стек
(i) RJMP k	$-2K < k < 2K$	$PC \leftarrow PC + k + 1$	Стек не меняется

16-разрядный код операции:

1100	kkkk	kkkk	kkkk
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
1	-	-	-	-	-	-	-

I: 1

Флаг установлен

Пример:

```

    cpi r16, $42 ; Сравнить r16 с $42
    brne error ; Перейти, если r16 <> $42
    rjmp ok ; Безусловный переход
error: add r16, r17 ; Сложить r17 с r16
        inc r16 ; Увеличить на 1 r16
ok:    nop      ; Назначение для jmp (пустая операция)

```

Слов: 1 (2 байта)

Циклов: 2

Команда ROL - сдвинуть влево через перенос**Описание:**

Сдвиг всех битов Rd на одно место влево. Флаг переноса (C) регистра состояния (SREG) смещается на место бита 0 регистра Rd. Бит 7 смещается во флаг переноса (C).

Операция:

16-разрядный код операции:

0001	11dd	dddd	dddd
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	<=>	<=>	<=>	<=>	<=>	<=>
---	---	-----	-----	-----	-----	-----	-----

H: Rd3**S:** N^V, Для проверок со знаком

V: N^C (Для N и C после сдвига) Устанавливается, если N устанавливается и C очищается, или N очищается, а C устанавливается. В ином случае очищается (при наличии значений N и C после сдвига)

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: Rd7*R6*R5*R4*R3*R2*R1*R0

Устанавливается, если результат \$00, в ином случае очищается

C: Rd7

Устанавливается, если перед сдвигом был установлен MSB регистра Rd, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды**Пример:**

rol r15 ; Сдвигать влево

brcs oneenc ; Перейти, если установлен перенос

...

oneenc: nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1

Команда ROR - сдвинуть вправо через перенос

Описание:

Сдвиг всех битов Rd на одно место вправо. Флаг переноса (C) регистра состояния (SREG) смещается на место бита 7 регистра Rd. Бит 0 смещается во флаг переноса (C).

Операция:

16-разрядный код операции:

1001	010d	dddd	0111
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
---	---	---	---	---	---	---	---

-	-	-	<=>	<=>	<=>	<=>	<=>
---	---	---	-----	-----	-----	-----	-----

S: N^V, Для проверок со знаком

V: N^C (Для N и C после сдвига)

Устанавливается, если N устанавливается и C очищается или N очищается, а C устанавливается. В ином случае очищается (при наличии значений N и C после сдвига)

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: Rd7*R6*R5*R4*R3*R2*R1*R0

Устанавливается, если результат \$00, в ином случае очищается

C: Rd0

Устанавливается, если перед сдвигом был установлен LSB регистра Rd, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
ror r15 ; Сдвигать вправо
brcc zeroenc ; Перейти, если перенос очищен
```

```
...
zeroenc: nop ; Перейти по назначению (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1

Команда SBC - вычесть с переносом

Описание:

Вычитание содержимого регистра-источника и содержимого флага переноса (C) из регистра Rd, размещение результата в регистре назначения Rd.

Операция:

(i) $Rd \leftarrow Rd - Rr - C$

Синтаксис

(i) SBC Rd,Rr Операнды: $0 < d < 31, 0 < r < 31$

Счетчик программ:

$PC \leftarrow PC + 1$

16-разрядный код операции:

0000	10rd	dddd	rrrr
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	<=>	<=>	<=>	<=>	<=>	<=>
---	---	-----	-----	-----	-----	-----	-----

H: $Rd3 * Rr3 + Rr3 * R3 + R3 * Rd3$

Устанавливается, если есть заем из бита 3, в ином случае очищается

S: N^V , Для проверок со знаком

V: $Rd7 * Rr7 * R7 + Rd7 * Rr7 * R7$

Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0 * Z$

Предшествовавшее значение остается неизменным, если результат равен нулю, в ином случае очищается

C: $Rd7 * Rr7 + Rr7 * R7 + R7 * Rd7$

Устанавливается, если абсолютное значение содержимого Rr плюс предшествовавший перенос больше, чем абсолютное значение Rd, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
; Вычесть r1 : r0 из r3 : r2
sub r2, r0 ; Вычесть младший байт
sbc r3, r1 ; Вычесть старший байт с переносом
```

Слов: 1 (2 байта)

Циклов: 1

Команда SBCI - вычесть непосредственное значение с переносом**Описание:**

Вычитание константы и содержимого регистра переноса (C) из содержимого регистра, размещение результата в регистре назначения Rd.

Операция:(i) $Rd \leftarrow Rd - K - C$

Синтаксис

Операнды:

Счетчик программ:

(i) SBCI Rd,K

 $0 < d < 31, 0 < K < 255$ $PC \leftarrow PC + 1$

16-разрядный код операции:

0100	KKKK	dddd	KKKK
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	<=>	<=>	<=>	<=>	<=>	<=>
---	---	-----	-----	-----	-----	-----	-----

H: $Rd3*K3+K3*R3+R3*Rd3$

Устанавливается, если есть заем из бита 3, в ином случае очищается

S: $N \oplus V$, Для проверок со знаком**V:** $Rd7*K7*R7+Rd7*K7*R7$

Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7*R6*R5*R4*R3*R2*R1*R0*Z$

Предшествовавшее значение остается неизменным, если результат равен нулю, в ином случае очищается

C: $Rd7*K7+K7*R7+R7*Rd7$

Устанавливается, если абсолютное значение константы плюс предшествовавший перенос больше, чем абсолютное значение Rd, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды**Пример:**

```
; Вычесть $4F23 из r17 : r16
subi r16, r23 ; Вычесть младший байт
sbcI r17, $4F ; Вычесть старший байт с переносом
```

Слов: 1 (2 байта)

Циклов: 1

Команда SBI - установить бит в регистр I/O

Описание:

Команда устанавливает заданный бит в регистр I/O. Команда работает с младшими 32 регистрами I/O (адреса с 0 по 31)

Операция:

(i) I/O(P,b) <-- 1

Синтаксис	Операнды:	Счетчик программ:
(i) SBI P,b	$0 < P < 31, 0 < b < 7$	$PC <-- PC + k + 1$

16-разрядный код операции:

1001	1010	pppp	pbbb
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
out $1E, r0 ; Записать адрес EEPROM
sbi $1C, 0 ; Установить бит чтения в EECR
in r1, $1D ; Считать данные EEPROM
```

Слов: 1 (2 байта)

Циклов: 2

Команда SBIC - пропустить если бит в регистре I/O очищен**Описание:**

Команда проверяет состояние бита в регистре I/O и, если этот бит очищен, пропускает следующую команду. Данная команда работает с младшими 32 регистрами I/O (адреса с 0 по 31).

Операция:

- (i) If $I/O(P,b) = 0$ then $PC \leftarrow PC + 2$ (or 3) else $PC \leftarrow PC + 1$

Синтаксис	Операнды:	Счетчик программ:
(i) SBIC P,b	$0 < P < 31, 0 < b < 7$	PC $\leftarrow PC + 1$, если условия не соблюдены, нет пропуска PC $\leftarrow PC + 2$, если следующая команда длиной в 1 слово PC $\leftarrow PC + 3$, если следующие команды JMP или CALL

16-разрядный код операции:

1001	1001	pppp	pbbb
------	------	------	------

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
e2wait: sbic $1C, 1 ; Пропустить следующую команду, если EEWE очищен
        rjmp e2wait ; Запись EEPROM не завершена
        por      ; Продолжать (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, нет пропуска, 2 - если условия соблюдены, выполняется пропуск

Команда SBIS – пропустить, если бит в регистре I/O установлен

Описание:

Команда проверяет состояние бита в регистре I/O и, если этот бит установлен, пропускает следующую команду. Данная команда работает с младшими 32 регистрами I/O (адреса с 0 по 31).

Операция:

- (i) If $I/O(P,b) = 1$ then $PC \leftarrow PC + 2$ (or 3) else $PC \leftarrow PC + 1$

Синтаксис	Операнды:	Счетчик программ:
(i) SBIS P,b	$0 < P < 31, 0 < b < 7$	PC $\leftarrow PC + 1$, если условия не соблюдены, нет пропуска PC $\leftarrow PC + 2$, если следующая команда длиной в 1 слово PC $\leftarrow PC + 3$, пропускает команды JMP или CALL

16-разрядный код операции:

1001	1011	pppp	pbbb
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

waitset: sbis\$10, 0 ; Пропустить следующую команду, если установлен бит 0 в Порте D

rjmp waitset ; Бит не установлен
nop ; Продолжать (пустая операция)

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, нет пропуска, 2 - если условия соблюдены, выполняется пропуск

Команда SBIW - вычесть непосредственное значение из слова**Описание:**

Вычитание непосредственного значения (0-63) из пары регистров и размещение результата в паре регистров. Команда работает с четырьмя верхними парами регистров, удобна для работы с регистрами указателями.

Операция:

- (i) Rdh:Rdl <-- Rdh:Rdl - K

Синтаксис	Операнды:	Счетчик программ:
(i) SBIW Rdl,K	dl $\in \{24, 26, 28, 30\}$, $0 < K < 63$	PC <- PC + 1

16-разрядный код операции:

1001	0111	KKdd	KKKK
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	<=>	<=>	<=>	<=>	<=>
---	---	---	-----	-----	-----	-----	-----

S: N \square V, Для проверок со знаком**V:** Rdh7*R15

Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R15

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: R15*R14*R13*R12*R11*R10*R9*R8*R7*R6*R5*R4*R3*R2*R1*R0

Устанавливается, если результат \$0000, в ином случае очищается

C: R15*Rdh7

Устанавливается, если абсолютное значение константы K больше абсолютного значения содержимого регистра Rd, в ином случае очищается

R: (Результат) соответствует Rdh:Rdl после выполнения команды (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0 = R7-R0)

Пример:

sbiw r24, 1 ; Вычесть 1 из r25:r24

sbiw r28, 63 ; Вычесть 63 из Y указателя (r29 : r28)

Слов: 1 (2 байта)

Циклов: 2

Команда SBR - установить биты в регистре

Описание:

Команда выполняет установку определенных битов в регистре Rd. Команда выполняет логическое ORI между содержимым регистра Rd и маской-константой K и размещает результат в регистре назначения Rd.

Операция:

- (i) $Rd \leftarrow Rd \vee K$

Синтаксис	Операнды:	Счетчик программ:
(i) SBR Rd,K	$16 < d < 31, 0 < K < 255$	$PC \leftarrow PC + 1$

16-разрядный код операции:

0110	KKKK	dddd	KKKK
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	<=>	0	<=>	<=>	-
---	---	---	-----	---	-----	-----	---

S: $N \square V$, Для проверок со знаком

V: 0

Очищен

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если результат \$00, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

sbr r16, 3F0 ; Установить биты 0 и 1 в r16

sbr r17, \$F0 ; Установить старшие 4 бита в r17

Слов: 1 (2 байта)

Циклов: 1

Команда SBRC – пропустить, если бит в регистре очищен

Описание:

Команда проверяет состояние бита в регистре и, если этот бит очищен, пропускает следующую команду.

Операция:

- (i) If Rr (b) = 0 then PC <-- PC + 2 (or 3) else PC <-- PC + 1

Синтаксис	Операнды:	Счетчик программ:
(i) SBRC Rr,b	0 < r < 31, 0 < b < 7	PC <-- PC + 1, если условия не соблюдены, нет пропуска PC <-- PC + 2, если следующая команда длиной в 1 слово PC <-- PC + 3, если следующие команды JMP или CALL

16-разрядный код операции:

1111	110r	rrrr	Xbbb
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
sub r0, r1 ; Вычесть r1 из r0
sbrc r0, 7 ; Пропустить, если бит 7 в r0 очищен
sub r0, r1 ; Выполняется, только если бит 7 в r0 не очищен
por      ; Продолжать (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, нет пропуска, 2 - если условия соблюдены, выполняется пропуск

Команда SBRS - пропустить если бит в регистре установлен

Описание:

Команда проверяет состояние бита в регистре и, если этот бит установлен, пропускает следующую команду.

Операция

(i) If Rr(b) = 1 then PC <-- PC + 2 (or 3) else PC <-- PC + 1

Синтаксис	Операнды:	Счетчик программ:
(i) SBRS Rr,b	0 < r < 31, 0 < b < 7	PC <-- PC + 1, если условия не соблюдены, нет пропуска PC <-- PC + 2, если следующая команда длиной в 1 слово PC <-- PC + 3, пропускает команды JMP или CALL

16-разрядный код операции:

1111	111r	rrrr	Xbbb
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
sub r0, r1 ; Вычесть r1 из r0
sbrs r0, 7 ; Пропустить если бит 7 в r0 установлен
neg r0    ; Выполняется только если бит 7 в r0 не установлен
nop      ; Продолжать (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены, нет пропуска, 2 - если условия соблюдены, выполняется пропуск

Команда SEC - установить флаг переноса**Описание:**

Команда устанавливает флаг переноса (C) в регистре статуса (SREG)

Операция:(i) $C \leftarrow 1$

Синтаксис	Операнды:	Счетчик программ:
(i) SEC	None	$PC \leftarrow PC + 1$

16-разрядный код операции:

1001	0100	0000	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	1
---	---	---	---	---	---	---	---

C: 1

Флаг переноса установлен

Пример:

```
sec      ; Установить флаг переноса
adc r0, r1 ; r0 = r0 + r1 + 1
```

Слов: 1 (2 байта)

Циклов: 1

Команда SEH - установить флаг полупереноса**Описание:**

Команда устанавливает флаг полупереноса (H) в регистре статуса (SREG).

Операция:

(i) H <-- 1

Синтаксис	Операнды:	Счетчик программ:
(i) SEH	None	PC <-- PC + 1

16-разрядный код операции:

1001	0100	0101	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	1	-	-	-	-	-
---	---	---	---	---	---	---	---

H: 1

Флаг полупереноса установлен

Пример:

seh ; Установить флаг полупереноса

Слов: 1 (2 байта)

Циклов: 1

Команда SEI - установить флаг глобального прерывания

Описание:

Команда устанавливает флаг глобального прерывания (I) в регистре статуса (SREG).

Операция:

(i) $I \leftarrow 1$

Синтаксис	Операнды:	Счетчик программ:
(i) SEI	None	$PC \leftarrow PC + 1$

16-разрядный код операции:

1001	0100	0111	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
1	-	-	-	-	-	-	-

I: 1

Флаг глобального прерывания установлен

Пример:

```
cli      ; Запретить прерывания
in r13, $16 ; Считать Порт В
sei      ; Разрешить прерывания
```

Слов: 1 (2 байта)

Циклов: 1

Команда SEN - установить флаг отрицательного значения

Описание:

Команда устанавливает флаг отрицательного значения (N) в регистре статуса (SREG).

Операция:

(i) N <-- 1

Синтаксис	Операнды:	Счетчик программ:
(i) SEN	None	PC <-- PC + 1

16-разрядный код операции:

1001	0100	0010	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	1	-	-

N: 1

Флаг переноса установлен

Пример:

```
add r2, r19 ; Сложить r19 с r2
sen      ; Установить флаг отрицательного значения
```

Слов: 1 (2 байта)

Циклов: 1

Команда SER - установить все биты регистра

Описание:

Значение \$FF заносится непосредственно в регистр назначения Rd.

Операция:

(i) $Rd \leftarrow \$FF$

Синтаксис	Операнды:	Счетчик программ:
(i) SER Rd	$16 < d < 31$	$PC \leftarrow PC + 1$

16-разрядный код операции:

1110	1111	dddd	1111
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

```
clr r16 ; Очистить r16
ser r17 ; Установить r17
out #18, r16 ; Записать нули в Порт В
nop ; Задержка (пустая операция)
out #18, r17 ; Записать единицы в Порт В
```

Слов: 1 (2 байта)

Циклов: 1

Команда SES - установить флаг знака

Описание:

Команда устанавливает флаг учета знака (S) в регистре статуса (SREG).

Операция:

(i) S <-- 1

Синтаксис	Операнды:	Счетчик программ:
(i) SES	None	PC <-- PC + 1

16-разрядный код операции:

1001	0100	0100	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	1	-	-	-	-

S: 1

Флаг учета знака установлен

Пример:

```
add r2, r19 ; Сложить r19 с r2
ses      ; Установить флаг отрицательного значения
```

Слов: 1 (2 байта)

Циклов: 1

Команда SET - установить флаг T

Описание:

Команда устанавливает флаг пересылки (T) в регистре статуса (SREG).

Операция:

(i) $T \leftarrow 1$

Синтаксис	Операнды:	Счетчик программ:
(i) SET	None	$PC \leftarrow PC + 1$

16-разрядный код операции:

1001	0100	0110	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	1	-	-	-	-	-	-
---	---	---	---	---	---	---	---

T: 1

Флаг пересылки установлен

Пример:

set ; Установить T флаг

Слов: 1 (2 байта)

Циклов: 1

Команда SEV - установить флаг переполнения**Описание:**

Команда устанавливает флаг переполнения (V) в регистре статуса (SREG).

Операция:(i) $V \leftarrow 1$

Синтаксис	Операнды:	Счетчик программ:
(i) SEV	None	$PC \leftarrow PC + 1$

16-разрядный код операции:

1001	0100	0011	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	1	-	-	-
---	---	---	---	---	---	---	---

V: 1

Флаг переполнения установлен

Пример:

```
add r2, r19 ; Сложить r19 с r2
sev      ; Установить флаг переполнения
```

Слов: 1 (2 байта)

Циклов: 1

Команда SEZ - установить флаг нулевого значения**Описание:**

Команда устанавливает флаг нулевого значения (Z) в регистре статуса (SREG).

Операция:(i) $Z \leftarrow 1$

Синтаксис	Операнды:	Счетчик программ:
(i) SEZ	None	$PC \leftarrow PC + 1$

16-разрядный код операции:

1001	0100	0001	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	1	-

Z: 1

Флаг нулевого значения установлен

Пример:

```
add r2, r19 ; Сложить r19 с r2
sez      ; Установить флаг нулевого значения
```

Слов: 1 (2 байта)

Циклов: 1

Команда SLEEP - установить режим SLEEP

Описание:

Команда устанавливает схему в SLEEP режим, определяемый регистром управления ЦПУ. Когда прерывание выводит ЦПУ из SLEEP режима, команда, следующая за командой SLEEP, будет выполнена прежде, чем отработает обработчик прерывания.

Операция:

Синтаксис	Операнды:	Счетчик программ:
(i) SLEEP	None	PC <-- PC + 1

16-разрядный код операции:

1001	0101	100X	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
mov r0, r11 ; Копировать r11 в r0
sleep      ; Перевести MCU в режим sleep
```

Слов: 1 (2 байта)

Циклов: 1

Команда ST - записать косвенно из регистра в СОЗУ с использованием индекса X

Описание:

Записывается косвенно один байт из регистра в СОЗУ. Положение байта в СОЗУ указывается 16-разрядным регистром-указателем X в регистровом файле. Обращение к памяти ограничено текущей страницей объемом 64К байта. Для обращения к другой странице СОЗУ необходимо изменить регистр RAMPX в I/O области. Регистр-указатель X может оставаться неизменным после выполнения команды, но может быть инкрементирован или декрементирован. Эта особенность очень удобна при использовании регистра-указателя X в качестве указателя стека.

Использование X-указателя:

Операция:

- | | | |
|------------------|------------|-----------------------------------|
| (i) (X) <- Rr | | Комментарий: |
| (ii) (X) <- Rr | X <- X + 1 | X: Неизменен |
| (iii) X <- X - 1 | (X) <- Rr | X: Инкрементирован впоследствии |
| | | X: Предварительно декрементирован |

Синтаксис

Синтаксис	Операнды:	Счетчик программ:
(i) ST X,Rr	0 < d < 31	PC <- PC + 1
(ii) ST X+,Rr	0 < d < 31	PC <- PC + 1
(iii) ST -X,Rr	0 < d < 31	PC <- PC + 1

16-разрядный код операции:

(i)	1001	001r	rrrr	1100
(ii)	1001	001r	rrrr	1101
(iii)	1001	001r	rrrr	1110

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
clr r27 ; Очистить старший байт X
ldi r26, $20 ; Установить $20 в младший байт X
st X+,r0 ; Сохранить в r0 содержимое SRAM по адресу $20 (X постинкрементируется)
st X, r1 ; Сохранить в r1 содержимое SRAM по адресу $21
ldi r26, $23 ; Установить $23 в младший байт X
st r2, X ; Сохранить в r2 содержимое SRAM по адресу $23
st r3, -X ; Сохранить в r3 содержимое SRAM по адресу $22 (X преддекрементируется)
```

Слов: 1 (2 байта)

Циклов: 2

Команда ST (STD) - записать косвенно из регистра в СОЗУ с использованием индекса Y

Описание:

Записывается косвенно, со смещением или без смещения, один байт из регистра в СОЗУ. Положение байта в СОЗУ указывается 16-разрядным регистром-указателем Y в регистровом файле. Обращение к памяти ограничено текущей страницей объемом 64К байта. Для обращения к другой странице СОЗУ необходимо изменить регистр RAMPY в I/O области. Регистр-указатель Y может оставаться неизменным после выполнения команды, но может быть инкрементирован или декрементирован. Эта особенность очень удобна при использовании регистра-указателя Y в качестве указателя стека.

Использование Y-указателя:

Операция:

- (i) (Y) <-- Rr
- (ii) (Y) <-- Rr Y <-- Y + 1
- (iii) Y <-- Y - 1 (Y) <-- Rr
- (iv) (Y + q) <-- Rr

Комментарий:

- Y: Неизменен
- Y: Инкрементирован впоследствии
- Y: Предварительно декрементирован
- Y: Неизменен, q: смещение

Синтаксис

- | | |
|-----------------|---------------------------|
| (i) ST Y,Rr | 0 < d < 31 |
| (ii) ST Y+,Rr | 0 < d < 31 |
| (iii) ST -Y,Rr | 0 < d < 31 |
| (iv) STD Y+q,Rr | 0 < d < 31,
0 < q < 63 |

Операнды:

Счетчик программ:

- PC <-- PC + 1

16-разрядный код операции:

(i)	1000	001r	rrrr	1000
(ii)	1001	001r	rrrr	1001
(iii)	1001	001r	rrrr	1010
(iv)	10q0	qq1r	rrrr	1qqq

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```

clr r29 ;Очистить старший байт Y
ldi r28, $20 ;Установить $20 в младший байт Y
st Y+, r0 ;Сохранить в r0 содержимое SRAM по адресу $20 (Y постинкремен-
тируется)
st Y, r1 ;Сохранить в r1 содержимое SRAM по адресу $21
ldi r28, $23 ;Установить $23 в младший байт Y
st Y, r2 ;Сохранить в r2 содержимое SRAM по адресу $23
st -Y, r3 ;Сохранить в r3 содержимое SRAM по адресу $22 (Y преддекремен-
тируется)
std Y+2, r4 ;Сохранить в r4 содержимое SRAM по адресу $24

```

Слов: 1 (2 байта)

Циклов: 2

Команда ST (STD) - записать косвенно из регистра в СОЗУ с использованием индекса Z

Описание:

Записывается косвенно, со смещением или без смещения, один байт из регистра в СОЗУ. Положение байта в СОЗУ указывается 16-разрядным регистром-указателем Z в регистровом файле. Обращение к памяти ограничено текущей страницей объемом 64К байта. Для обращения к другой странице СОЗУ необходимо изменить регистр RAMPZ в I/O области. Регистр-указатель Z может оставаться неизменным после выполнения команды, но может быть инкрементирован или декрементирован. Эта особенность очень удобна при использовании регистра-указателя Z в качестве указателя стека, однако, поскольку регистр-указатель Z может быть использован для косвенного вызова подпрограмм, косвенных переходов и табличных преобразований, более удобно использовать в качестве указателя стека регистры-указатели X и Y.

Использование Z-указателя:

Операция:	Комментарий:
(i) (Z) <- Rr	Z: Неизменен
(ii) (Z) <- Rr Z <- Y + 1	Z: Инкрементирован впоследствии
(iii) Z <- Z - 1 (Z) <- Rr	Z: Предварительно декрементирован
(iv) (Z + q) <- Rr	Z: Неизменен, q: смещение

Синтаксис	Операнды:	Счетчик программ:
(i) ST Z,Rr	0 < d < 31	PC <- PC + 1
(ii) ST Z+,Rr	0 < d < 31	PC <- PC + 1
(iii) ST -Z,Rr	0 < d < 31	PC <- PC + 1
(iv) STD Z+q,Rr	0 < d < 31, 0 < q < 63	PC <- PC + 1

16-разрядный код операции:

(i)	1000	001r	rrrr	0000
(ii)	1001	001r	rrrr	0001
(iii)	1001	001r	rrrr	0100
(iv)	10q0	qq1r	rrrr	0qqq

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```

    clr r31 ; Очистить старший байт Z
    ldi r30, $20 ; Установить $20 в младший байт Z
    st Z+,r0 ; Сохранить содержимое r0 в SRAM по adr. $20 (Z постинкрементируется)
    st Z, r1 ; Сохранить содержимое r1 в SRAM по адресу $21
    ldi r30, $23 ; Установить $23 в младший байт Z
    st Z, r2 ; Сохранить содержимое r2 в SRAM по адресу $23
    st -Z, r3 ; Сохранить содержимое r3 в SRAM по adr. $22 (Z преддекрементируется)
    std Z+2, r4 ; Сохранить содержимое r4 в SRAM по адресу $24
  
```

Слов: 1 (2 байта)

Циклов: 2

Команда STS - загрузить непосредственно в СОЗУ**Описание:**

Выполняется запись одного байта из регистра в СОЗУ. Можно использовать 16-разрядный адрес. Обращение к памяти ограничено текущей страницей СОЗУ объемом 64К байта. Команда STS использует для обращения к памяти выше 64К байт регистр RAMPZ.

Операция:

(i) (k) <-- Rr

Синтаксис	Операнды:	Счетчик программ:
(i) STS k,Rr	$0 < r < 31, 0 < k < 65535$	$PC \leftarrow PC + 2$

32-разрядный код операции:

1001	001d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	-	-	-	-	-
---	---	---	---	---	---	---	---

Пример:

```
lds r2, $FF00 ; Загрузить в r2 содержимое SRAM по адресу $FF00
add r2, r1   ; Сложить r1 с r2
sts $FF00, r2 ; Записать обратно
```

Слов: 2 (4 байта)

Циклов: 3

Команда SUB - вычесть без переноса

Описание:

Вычитание содержимого регистра-источника Rr из содержимого регистра Rd, размещение результата в регистре назначения Rd.

Операция:

(i) $Rd \leftarrow Rd - Rr$

Синтаксис	Операнды:	Счетчик программ:
(i) SUB Rd,Rr	$16 < d < 31, 0 < r < 31$	$PC \leftarrow PC + 1$

16-разрядный код операции:

0001	10rd	dddd	rrrr
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	<=>	<=>	<=>	<=>	<=>	<=>

H: $Rd3 * Rr3 + Rr3 * R3 + R3 * Rd3$

Устанавливается, если есть заем из бита 3, в ином случае очищается

S: NEV, Для проверок со знаком

V: $Rd7 * Rr7 * R7 + Rd7 * Rr7 * R7$

Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если результат \$00, в ином случае очищается

C: $Rd7 * Rr7 + Rr7 * R7 + * R7 * Rd7$

Устанавливается, если абсолютное значение содержимого Rr больше, чем абсолютное значение Rd, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
sub r13, r12 ; Вычесть r12 из r13
brne noteq ; Перейти, если r12 <> r13
noteq: nop      ; Перейти по назначению (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1

Команда SUBI - вычесть непосредственное значение

Описание:

Вычитание константы из содержимого регистра, размещение результата в регистре назначения Rd.

Операция:

(i) $Rd \leftarrow Rd - K$

Синтаксис	Операнды:	Счетчик программ:
(i) SUB Rd,K	$16 < d < 31, 0 < K < 255$	$PC \leftarrow PC + 1$

16-разрядный код операции:

0101	KKKK	dddd	KKKK
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	<=>	<=>	<=>	<=>	<=>	<=>

H: $Rd3 * K3 + K3 * R3 + R3 * Rd3$

Устанавливается, если есть заем из бита 3, в ином случае очищается

S: NEV, Для проверок со знаком

V: $Rd7 * K7 * R7 + Rd7 * K7 * R7$

Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: $R7 * R6 * R5 * R4 * R3 * R2 * R1 * R0$

Устанавливается, если результат \$00, в ином случае очищается

C: $Rd7 * K7 + K7 * R7 + R7 * Rd7$

Устанавливается, если абсолютное значение константы больше, чем абсолютное значение Rd, в ином случае очищается

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
subi r22, $11 ; Вычесть $11 из r22
brne noteq ; Перейти, если r22 <> $11
```

...

noteq: nop ; Перейти по назначению (пустая операция)

Слов: 1 (2 байта)

Циклов: 1

Команда SWAP - поменять тетрады местами

Описание:

Команда меняет местами старший и младший nibблы (полубайты) регистра.

Операция:

- (i) R(7-4) <-- Rd(3-0), R(3-0) <-- Rd(7-4)

Синтаксис	Операнды:	Счетчик программ:
(i) SWAP Rd	$0 < d < 31$	$PC \leftarrow PC + k + 1$

16-разрядный код операции:

1001	010d	dddd	0010
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

R: (Результат) соответствует Rd после выполнения команды

Пример:

```

inc r1 ; Увеличить на 1 r1
swap r1 ; Поменять местами nibблы r1
inc r1 ; Увеличить на 1 старший nibбл r1
swap r1 ; Снова поменять местами nibблы r1

```

Слов: 1 (2 байта)

Циклов: 1

Команда TST - проверить на ноль или минус**Описание:**

Регистр проверяется на нулевое или отрицательное состояние. Выполняется логическое AND содержимого регистра с самим собой. Содержимое регистра остается неизменным.

Операция:

- (i) $Rd \leftarrow Rd * Rd$

Синтаксис	Операнды:	Счетчик программ:
(i) TST Rd	$0 < d < 31$	$PC \leftarrow PC + 1$

16-разрядный код операции:

0010	00dd	dddd	dddd
------	------	------	------

Булевые выражения регистра статуса (SREG)

I T H S V N Z C

-	-	-	<=>	0	<=>	<=>	-
---	---	---	-----	---	-----	-----	---

S: NEV, Для проверок со знаком**V:** 0

Очищен

N: R7

Устанавливается, если в результате установлен MSB, в ином случае очищается

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

Устанавливается, если результат \$00, в ином случае очищается

R: (Результат) соответствует Rd**Пример:**

```
tst r0 ; Проверить r0
breq zero ; Перейти, если r0 = 0
```

...

```
zero: nop ; Перейти по назначению (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1

Команда WDR - сбросить сторожевой таймер**Описание:**

Команда сбрасывает сторожевой таймер (Watchdog Timer). Команда может быть выполнена внутри заданного предделителем сторожевого таймера промежутка времени.

Операция:

- (i) Перезапускается WD (сторожевой таймер)

Синтаксис	Операнды:	Счетчик программ:
(i) WDR	None	PC <-- PC + 1

16-разрядный код операции:

1001	0101	101X	1000
------	------	------	------

Булевые выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
wdr ; Сбросить сторожевой таймер
```

Слов: 1 (2 байта)

Циклов: 1