МИКРОСХЕМА ИНТЕГРАЛЬНАЯ

1887ВЕ7Т

Руководство пользователя

2019

Содержание

Введение

В настоящем руководстве пользователя (РП) приведено описание архитектуры, структуры, системы команд и особенностей применения интегральной схемы (ИС) 1887ВЕ7Т, которая представляет собой 8-разрядный микроконтроллер AVR RISC-архитектуры, тактовой частотой 16 МГц и содержит 128 Кбайт встроенной памяти программ (типа EEPROM), 4 Кбайт встроенного ОЗУ, 4 Кбайт встроенной энергонезависимой памяти данных (типа EEPROM), 8-канальный 10-разрядный АЦП, последовательный периферийный интерфейс SPI, двухпроводной последовательный интерфейс TWI, интерфейс JTAG, программируемую защиту кода программ, программируемые последовательные порты USART. Разработанная микросхема будет 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. Импортозамещение электронных компонентов наиболее эффективно в случае использования полных функциональных аналогов изделий микроэлектроники.

Настоящее РП может служить практическим руководством по применению микроконтроллеров для разработчиков систем на основе ИС 1887ВЕ7Т.

1 Назначение и область применения

Микросхема 1887ВЕ7Т предназначена для использования в системах приема, передачи и обработки информации, встроенного управления и в автономных необслуживаемых аппаратах специального назначения. Применение изделия обеспечит технологическую независимость при разработке целого ряда новых перспективных образцов и систем вооружения и военной техники, использующих цифровую технику и цифровые методы обработки сигналов, а также при комплектовании и модернизации действующих систем и комплексов. Применение разработанной СБИС позволит обеспечить требуемые тактико-технические данные важнейших систем и выполнить все необходимые требования по назначению, энергопотреблению и массогабаритным показателям. Это позволит также решить вопрос комплектования радиоэлектронной аппаратуры специального назначения отечественными ИС взамен аналогичных импортных микросхем.

2 Краткое техническое описание ИС 1887ВЕ7Т

Микросхема 1887ВЕ7Т представляет собой 8-разрядный микроконтроллер AVR RISC-архитектуры. За счет выполнения большинства инструкций за один машинный цикл, микросхема достигает производительности 1 миллион операций в секунду на 1 МГц, что позволяет проектировщикам систем оптимизировать соотношение энергопотребления и быстродействия.

Ядро AVR сочетает набор инструкций с 32 регистрами общего назначения. Все 32 регистра непосредственно подключены к арифметико-логическому устройству (АЛУ), которое позволяет выполнять действия, используя значения одного или двух регистров. Данная архитектура обладает большей эффективностью кода за счет достижения производительности в 4 раза превосходящей CISC-микроконтроллеры на той же тактовой частоте.

Микросхема имеет следующие особенности: 128 Кбайт памяти программ с поддержкой чтения во время записи, 4 Кбайт энергонезависимой памяти данных (ЭПД, либо ЭСППЗУ данных – электрически стираемого перепрограммируемого постоянного запоминающего устройства данных), 4 Кбайт статического ОЗУ, 53 универсальные линии ввода-вывода, 32 регистра общего назначения, счетчик реального времени (RTC), четыре гибких таймера/счетчика с режимами сравнения и ШИМ, два универсальных синхронно-асинхронных приемопередатчика, байт-ориентированный двухпроводной последователь-

ный интерфейс, 8-канальный 10-разрядный АЦП с дополнительным дифференциальным входом и программируемым коэффициентом усиления, программируемый сторожевой таймер с внутренним генератором, последовательный интерфейс SPI, тестовый интерфейс JTAG, совместимый со стандартом IEEE 1149.1, который также используется для доступа к встроенной системе отладки и для программирования, а также шесть программно выбираемых энергосберегающих режимов. Режим холостого хода (Idle) останавливает ЦПУ, но при этом поддерживает работу статического ОЗУ, таймеров/счетчиков, SPI-интерфейса и системы прерываний. Режим хранения (Power-down) позволяет сохранить содержимое регистров, но останавливает генератор, блокируя все остальные функции микросхемы до тех пор, пока не произойдет следующее прерывание или аппаратный сброс. В режиме микропотребления (Power-save) асинхронный таймер продолжает работу, позволяя пользователю сохранить функцию счета времени, в то время как остальная часть контроллера находится в режиме сна. Режим снижения шумов АЦП (ADC Noise Reduction) останавливает ЦПУ и все интерфейсы ввода-вывода, за исключением асинхронного таймера и АЦП, для минимизации импульсных шумов в процессе преобразования АЦП. В режиме ожидания (Standby) генератор кварцевого резонатора продолжает работу, а остальная часть микроконтроллера находится в режиме сна. Данный режим характеризуется малой потребляемой мощностью, но при этом позволяет достичь самого быстрого возврата в рабочий режим. В расширенном режиме ожидания (Extended Standby) и основной генератор и асинхронный таймер продолжают работать.

Карта регистров микроконтроллера представлена в приложении В.

В состав микроконтроллера входит энергонезависимая память (программ и данных), которую можно перепрограммировать непосредственно внутри системы через последовательный интерфейс SPI с помощью обычного программатора или автономной программы в секторе загрузчика. Загрузочная программа может использовать любой интерфейс для загрузки прикладной программы в память программ. Программа в загрузочном секторе продолжает работу в процессе обновления секции прикладной памяти программ, поддерживая операцию чтения во время записи. Благодаря сочетанию 8-разрядного RISC ядра с внутрисистемно самопрограммируемой памятью программ в одном кристалле, ИС 1887ВЕ7Т являются микроконтроллерами, позволяющими достичь высокой степени гибкости и эффективной стоимости при проектировании большинства приложений встроенного управления.

Микросхема поддерживается набором программно-аппаратных средств, включая Си-компиляторы, макроассемблеры, программные отладчики/симуляторы, внутрисистемные эмуляторы и макетно-отладочные платы.

2.1 Функциональные возможности

Функциональные возможности микросхемы:

- 8-разрядная RISC архитектура	
- тактовая частота, МГц	16
- встроенный двухцикловый умножитель, бит	8 × 8
- память программ, байт	128К
- объем внутреннего ОЗУ, байт	4К
- объем встроенной энергонезависимой памяти данных, байт	4К
- объем адресуемой внешней памяти, байт	64К
- программируемые последовательные порты USART	2
- последовательный периферийный интерфейс SPI с функцией внутрисхемного программирования	1
- байт-ориентированный двухпроводной последовательный интерфейс TWI	1
- 8-битные таймеры/счетчики с раздельными предделителями и режимами сравнения	2
- 16-битные таймеры/счетчики с раздельными предделителями,	

режимами сравнения и режимами захвата 2
- счетчик реального времени с отдельным генератором 1
- 8-разрядные каналы ШИМ 2
- каналы ШИМ с программируемым разрешением от 2 до 16 бит 6
- встроенный АЦП: разрядность преобразования, бит 10
количество каналов 8
- программируемый сторожевой таймер со встроенным генератором. 1
- встроенный аналоговый компаратор 1
- встроенный калиброванный RC-генератор 1
- количество программируемых линий ввода-вывода 53
- количество внешних и внутренних источников прерываний 21
- количество режимов пониженного энергопотребления 6
- интерфейс JTAG
- полностью статическая работа
- программируемая защита кода программы
- сброс при подаче питания и программируемая схема сброса при снижении напряжения питания.

Микросхема 1887ВЕ7Т разработана в металлокерамическом планарном корпусе с четырехсторонним расположением выводов типа 4203.64–2.

Номинальное значение напряжения питания микросхемы 5,0 В.

Допустимое отклонение напряжения питания от номинального ±10 %.

2.2 Особенности архитектуры

Структурная схема ИС 1887ВЕ7Т приведена на рисунке 2.1.

Условное графическое обозначение микросхемы показано на рисунке 2.2.

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

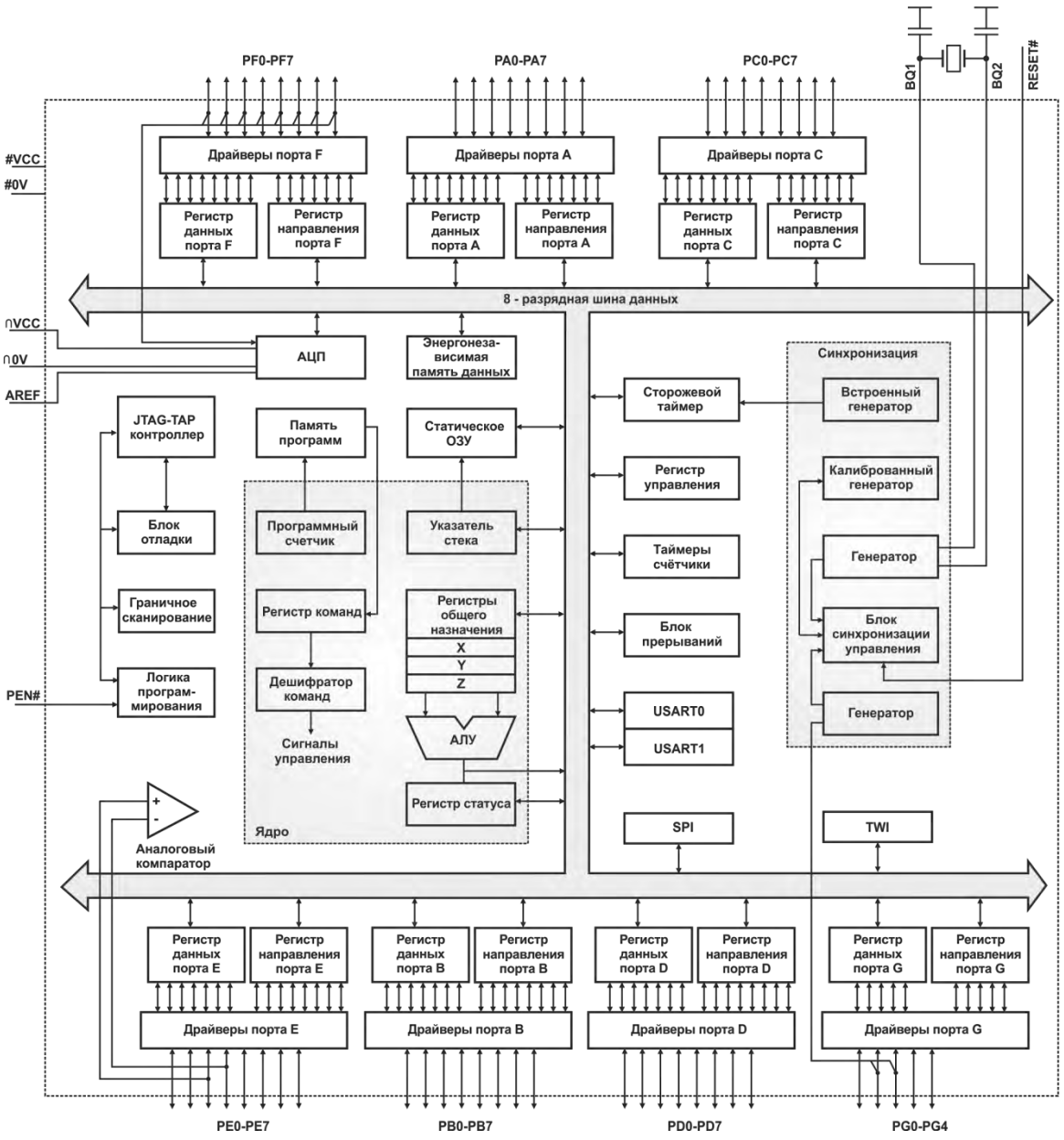


Рисунок 2.1 – Структурная схема ИС 1887ВЕ7Т

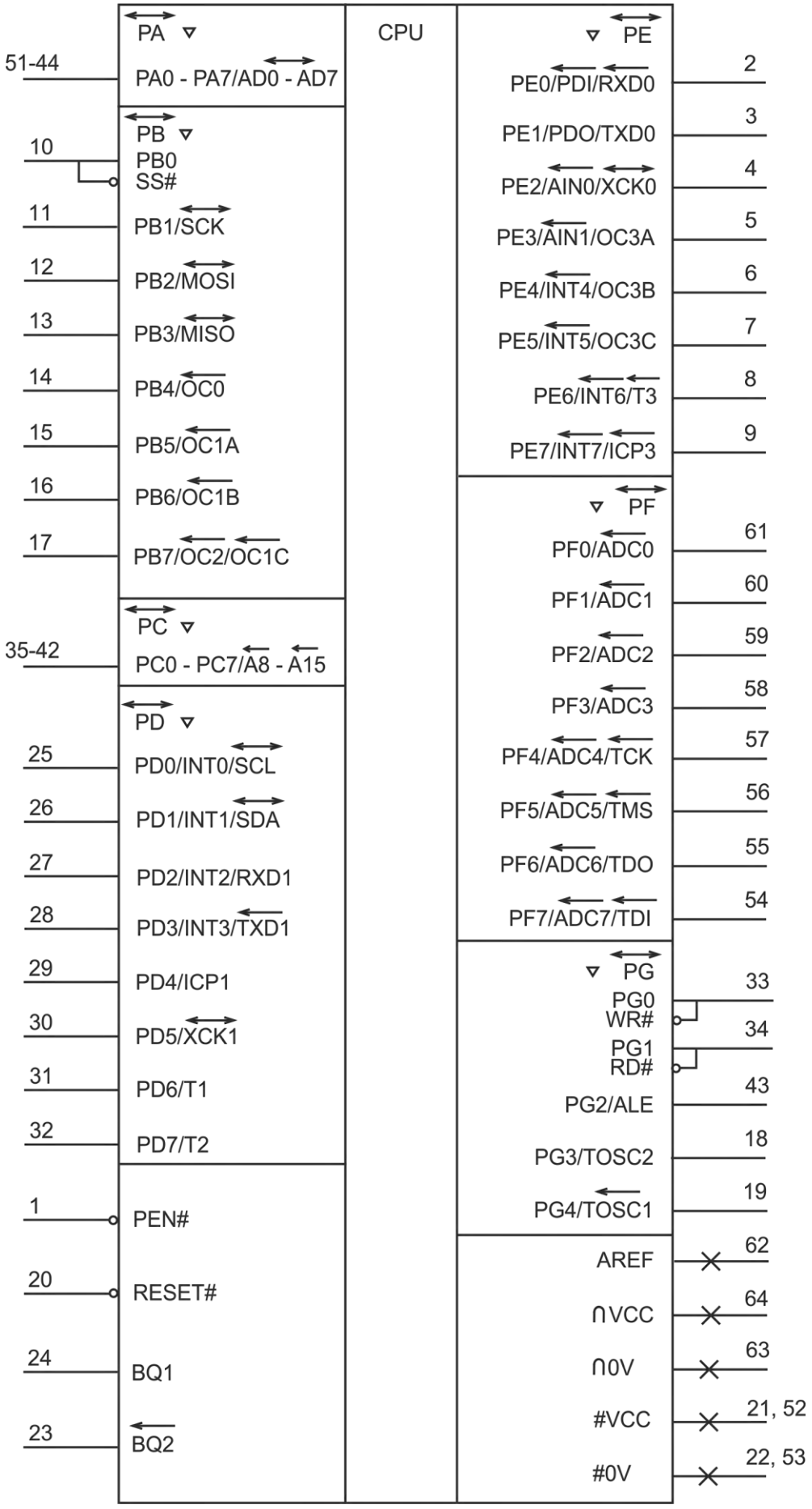


Рисунок 2.2 – Условное графическое обозначение ИС 1887ВЕ7Т

Таблица 2.1 – Функциональное назначение выводов микросхемы

Обозначение вывода	Номер вывода	Функциональное назначение	Тип вывода	Обозначение альтернативной функции вывода
PA0	51	Вход/выход "порт A, 0-й разряд" Вход/выход "разряд 0 адреса/данных интерфейса внешней памяти"	I/O/Z I/O	AD0
PA1	50	Вход/выход "порт A, 1-й разряд" Вход/выход "разряд 1 адреса/данных интерфейса внешней памяти"	I/O/Z I/O	AD1
PA2	49	Вход/выход "порт A, 2-й разряд" Вход/выход "разряд 2 адреса/данных интерфейса внешней памяти"	I/O/Z I/O	AD2
PA3	48	Вход/выход "порт A, 3-й разряд" Вход/выход "разряд 3 адреса/данных интерфейса внешней памяти"	I/O/Z I/O	AD3
PA4	47	Вход/выход "порт A, 4-й разряд" Вход/выход "разряд 4 адреса/данных интерфейса внешней памяти"	I/O/Z I/O	AD4
PA5	46	Вход/выход "порт A, 5-й разряд" Вход/выход "разряд 5 адреса/данных интерфейса внешней памяти"	I/O/Z I/O	AD5
PA6	45	Вход/выход "порт A, 6-й разряд" Вход/выход "разряд 6 адреса/данных интерфейса внешней памяти"	I/O/Z I/O	AD6
PA7	44	Вход/выход "порт A, 7-й разряд" Вход/выход "разряд 7 адреса/данных интерфейса внешней памяти"	I/O/Z I/O	AD7
PB0	10	Вход/выход "порт B, 0-й разряд" Вход "выбор ведомого интерфейса SPI"	I/O/Z I	SS#
PB1	11	Вход/выход "порт B, 1-й разряд" Вход/выход "тактовый сигнал интерфейса SPI"	I/O/Z I/O	SCK
PB2	12	Вход/выход "порт B, 2-й разряд" Вывод "ведущий выход/ведомый вход последовательного периферийного интерфейса SPI "	I/O/Z I/O	MOSI
PB3	13	Вход/выход "порт B, 3-й разряд" Вывод "ведущий вход/ведомый выход последовательного периферийного интерфейса SPI "	I/O/Z I/O	MISO
PB4	14	Вход/выход "порт B, 4-й разряд" Выход "сравнение и ШИМ-выход для таймера/счетчика 0"	I/O/Z O	OC0
PB5	15	Вход/выход "порт B, 5-й разряд" Выход "сравнение и ШИМ-выход A для таймера/счетчика 1"	I/O/Z O	OC1A
PB6	16	Вход/выход "порт B, 6-й разряд" Выход "сравнение и ШИМ-выход B для таймера/счетчика 1"	I/O/Z O	OC1B
PB7	17	Вход/выход "порт B, 7-й разряд" Выход "сравнение и ШИМ-выход для таймера/счетчика 2" Выход "сравнение и ШИМ-выход C для таймера/счетчика 1"	I/O/Z O O	OC2 OC1C
PC0	35	Вход/выход "порт C, 0-й разряд" Выход "8-й разряд старшего байта адреса интерфейса внешней памяти"	I/O/Z O	A8

Продолжение таблицы 2.1

Обозначение вывода	Номер вывода	Функциональное назначение	Тип вывода	Обозначение альтернативной функции вывода
PC1	36	Вход/выход "порт C, 1-й разряд" Выход "9-й разряд старшего байта адреса интерфейса внешней памяти"	I/O/Z O	 A9
PC2	37	Вход/выход "порт C, 2-й разряд" Выход "10-й разряд старшего байта адреса интерфейса внешней памяти"	I/O/Z O	 A10
PC3	38	Вход/выход "порт C, 3-й разряд" Выход "11-й разряд старшего байта адреса интерфейса внешней памяти"	I/O/Z O	 A11
PC4	39	Вход/выход "порт C, 4-й разряд" Выход "12-й разряд старшего байта адреса интерфейса внешней памяти"	I/O/Z O	 A12
PC5	40	Вход/выход "порт C, 5-й разряд" Выход "13-й разряд старшего байта адреса интерфейса внешней памяти"	I/O/Z O	 A13
PC6	41	Вход/выход "порт C, 6-й разряд" Выход "14-й разряд старшего байта адреса интерфейса внешней памяти"	I/O/Z O	 A14
PC7	42	Вход/выход "порт C, 7-й разряд" Выход "15-й разряд старшего байта адреса интерфейса внешней памяти"	I/O/Z O	 A15
PD0	25	Вход/выход "порт D, 0-й разряд" Вход "внешнее прерывание 0" Вход/выход "тактовый сигнал двухпроводного последовательного интерфейса TWI"	I/O/Z I I/O	 INT0 SCL
PD1	26	Вход/выход "порт D, 1-й разряд" Вход "внешнее прерывание 1" Вход/выход "данные двухпроводного последовательного интерфейса TWI"	I/O/Z I I/O	 INT1 SDA
PD2	27	Вход/выход "порт D, 2-й разряд" Вход "внешнее прерывание 2" Вход "данные последовательного асинхронного приемопередатчика UART1"	I/O/Z I I	 INT2 RXD1
PD3	28	Вход/выход "порт D, 3-й разряд" Вход "внешнее прерывание 3" Выход "данные последовательного асинхронного приемопередатчика UART1"	I/O/Z I O	 INT3 TXD1
PD4	29	Вход/выход "порт D, 4-й разряд" "Вход захвата таймера/счетчика 1"	I/O/Z I	 ICP1
PD5	30	Вход/выход "порт D, 5-й разряд" Вход/выход "тактовый сигнал универсального синхронно-асинхронного приемопередатчика USART1"	I/O/Z I/O	 XCK1
PD6	31	Вход/выход "порт D, 6-й разряд" Вход "тактовый сигнал таймера/счетчика 1"	I/O/Z I	 T1
PD7	32	Вход/выход "порт D, 7-й разряд" Вход "тактовый сигнал таймера/счетчика 2"	I/O/Z I	 T2
PE0	2	Вход/выход "порт E, 0-й разряд" Вход "программируемые данные" Вход "данные универсального асинхронного приемопередатчика UART0"	I/O/Z I I	 PDI RXD0

Продолжение таблицы 2.1

Обозначение вывода	Номер вывода	Функциональное назначение	Тип вывода	Обозначение альтернативной функции вывода
PE1	3	Вход/выход "порт E, 1-й разряд" Выход "программируемые данные" Выход "данные универсального асинхронного приемопередатчика UART0"	I/O/Z O O	 PDO TXD0
PE2	4	Вход/выход "порт E, 2-й разряд" Вход "положительный вход аналогового компаратора" Вход/выход "внешний тактовый сигнал универсального синхронно-асинхронного приемопередатчика USART0"	I/O/Z I I/O	 AIN0 XCK0
PE3	5	Вход/выход "порт E, 3-й разряд" Вход "отрицательный вход аналогового компаратора" Выход "сравнение и ШИМ-выход A для таймера/счетчика 3"	I/O/Z I O	 AIN1 OC3A
PE4	6	Вход/выход "порт E, 4-й разряд" Вход "внешнее прерывание 4" Выход "сравнение и ШИМ-выход B для таймера/счетчика 3"	I/O/Z I O	 INT4 OC3B
PE5	7	Вход/выход "порт E, 5-й разряд" Вход "внешнее прерывание 5" Выход "сравнение и ШИМ-выход C для таймера/счетчика 3"	I/O/Z I O	 INT5 OC3C
PE6	8	Вход/выход "порт E, 6-й разряд" Вход "внешнее прерывание 6" Вход "тактовый сигнал таймера/счетчика 3"	I/O/Z I I	 INT6 T3
PE7	9	Вход/выход "порт E, 7-й разряд" Вход "внешнее прерывание 7" Вход "вывод захвата таймера/счетчика 3"	I/O/Z I I	 INT7 ICP3
PF0	61	Вход/выход "порт F, 0-й разряд" Вход АЦП, канал 0	I/O/Z I	 ADC0
PF1	60	Вход/выход "порт F, 1-й разряд" Вход АЦП, канал 1	I/O/Z I	 ADC1
PF2	59	Вход/выход "порт F, 2-й разряд" Вход АЦП, канал 2	I/O/Z I	 ADC2
PF3	58	Вход/выход "порт F, 3-й разряд" Вход АЦП, канал 3	I/O/Z I	 ADC3
PF4	57	Вход/выход "порт F, 4-й разряд" Вход АЦП, канал 4 Вход "JTAG, тактовый сигнал"	I/O/Z I I	 ADC4 TCK
PF5	56	Вход/выход "порт F, 5-й разряд" Вход АЦП, канал 5 Вход "JTAG, выбор тестового режима"	I/O/Z I I	 ADC5 TMS
PF6	55	Вход/выход "порт F, 6-й разряд" Вход АЦП, канал 6 Выход "JTAG, выход тестовых данных"	I/O/Z I O	 ADC6 TDO
PF7	54	Вход/выход "порт F, 7-й разряд" Вход АЦП, канал 7 Вход "JTAG, вход тестовых данных"	I/O/Z I I	 ADC7 TDI
PG0	33	Вход/выход "порт G, 0-й разряд" Выход "строб записи во внешнюю память"	I/O/Z O	 WR#

Окончание таблицы 2.1

Обозначение вывода	Номер вывода	Функциональное назначение	Тип вывода	Обозначение альтернативной функции вывода
PG1	34	Вход/выход "порт G, 1-й разряд" Выход "строб чтения внешней памяти"	I/O/Z O	 RD#
PG2	43	Вход/выход "порт G, 2-й разряд" Выход "разрешение захвата адреса для доступа к внешней памяти"	I/O/Z O	 ALE
PG3	18	Вход/выход "порт G, 3-й разряд" Выход "вывод осциллятора таймера/счетчика 0"	I/O/Z O	 TOSC2
PG4	19	Вход/выход "порт G, 4-й разряд" Вход " вывод осциллятора таймера/счетчика 0"	I/O/Z I	 TOSC1
PEN#	1	Вход "разрешение программирования для последовательного периферийного интерфейса"	 I	
RESET#	20	Вход "сброс"/"напряжение программирования"	I	
BQ1	24	Вход инвертирующего усилителя осциллятора Вход тактового сигнала	I I	
BQ2	23	Выход инвертирующего усилителя осциллятора	O	
∩VCC	64	Вывод питания порта F и АЦП	-	
AREF	62	Вывод опорного напряжения для АЦП	-	
#VCC	21, 52	Вывод питания цифровой части ИС	-	
#0V	22, 53	Общий вывод цифровой части ИС	-	
∩0V	63	Общий вывод аналоговой части ИС	-	

Примечания

1 Обозначения в графе «Тип вывода»: I – вход, O – выход, Z – третье состояние.

2 Все выводы #0V и вывод ∩0V микросхемы объединены в кристалле общей шиной.

2.3 Система команд

Система команд включает в себя полный набор арифметических и логических команд, а также операции над битами, операции перехода и передачи данных с различными способами адресации, команды управления системой – 133 инструкции, большинство из которых выполняется за один машинный цикл. Перечень команд микроконтроллера 1887ВЕ7Т представлен в приложении А.

2.4 Электрические параметры микросхемы

Электрические параметры микросхемы 1887ВЕ7Т 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 ИС 1887ВЕ7Т при приемке и поставке

Наименование параметра, единица измерения, режим измерения		Буквенное обозначение параметра	Норма параметра		Температура среды, °С
			не менее	не более	
1 Выходное напряжение высокого уровня по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, PE0 – PE7, PF0 – PF7, PG0 – PG4, В	$U_{CC1} = U_{CC2} = 4,5$ В, $I_{OH} = -20$ мА	U_{OH}	U_{CC}– 1,2[1)]	–	–60± 3 25± 10 85 ± 3
2 Выходное напряжение низкого уровня по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, PE0 – PE7, PF0 – PF7, PG0 – PG4, В	$U_{CC1} = U_{CC2} = 4,5$ В, $I_{OL} = 20$ мА	U_{OL}	–	0,7	
3 Входное напряжение смещения аналогового компаратора, мВ, $U_I = U_{CC2}/2$	$U_{CC1} = U_{CC2} = 5,5$ В	U_{IO}	–	40	
4 Входной ток утечки входов/ выходов, мкА[2)], $U_{CC1} = U_{CC2} = 5,5$ В	$U_{IL} = 0$ В	I_{ILL}	–10	–	
	$U_{IH} = 5,5$ В	I_{ILH}	–	10	
5 Динамический ток потребления в активном режиме, мА, $f_{CI} = 8$ МГц	$U_{CC1} = U_{CC2} = 5,5$ В	I_{OCC}	–	50	
6 Динамический ток потребления в режиме холостого хода, мА, $f_{CI} = 8$ МГц	$U_{CC1} = U_{CC2} = 5,5$ В	I_{OCCI}	–	22	
7 Ток потребления в режиме хранения, мкА, сторожевой таймер разрешен	$U_{CC1} = U_{CC2} = 5,5$ В	I_{CCS1}	–	200	
8 Ток потребления в режиме хранения, мкА, сторожевой таймер запрещен	$U_{CC1} = U_{CC2} = 5,5$ В	I_{CCS2}	–	150	
9 Функциональный контроль[3)]	$U_{CC1} = U_{CC2} = (4,5; 5,5)$ В, $f_{CI} = 16$ МГц	ФК	–	–	

1) $U_{CC} = U_{CC1} = U_{CC2}$.

2) Параметры I_{ILL}, I_{ILH} при температуре минус 60 ºС не измеряются, а гарантируются нормой при температуре (25 ± 10) ºС.

3) Функциональный контроль осуществляется в соответствии с системой команд микроконтроллера.

Таблица 2.3 – Предельно допустимые и предельные режимы эксплуатации ИС 1887ВЕ7Т в диапазоне рабочих температур от минус 60 до плюс 85 ºС

Наименование параметра режима, единица измерения	Буквенное обозначение параметра	Предельно допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
1	2	3	4	5	6
1 Напряжение питания цифровой части, В[1)]	U_{CC1}	4,5	5,5	–0,5	6,0
2 Напряжение питания аналоговой части, В[1)]	U_{CC2}	4,5	5,5	–0,5	6,0

Продолжение таблицы 2.3

1	2	3	4	5	6
3 Входное напряжение низкого уровня по выводам кроме BQ1 и RESET#, В	U_{IL1}	–0,5	$0,2U_{CC1}$	–0,6	–
4 Входное напряжение низкого уровня на выводе BQ1, В	U_{IL2}	–0,5	$0,1U_{CC1}$	–0,6	–
5 Входное напряжение низкого уровня на выводе RESET#, В, $U_{CC1} = U_{CC2} = (4,5–5,5)$ В	U_{IL3}	–0,5	$0,2U_{CC1}$	–0,6	–
6 Входное напряжение высокого уровня по выводам кроме BQ1 и RESET#, В	U_{IH1}	$0,6U_{CC1}$	$U_{CC1}+0,5$	–	$U_{CC1}+0,5$
7 Входное напряжение высокого уровня по выводу BQ1, В	U_{IH2}	$0,7U_{CC1}$	$U_{CC1}+0,5$	–	$U_{CC1}+0,5$
8 Входное напряжение высокого уровня по выводу RESET#, В	U_{IH3}	$0,85U_{CC1}$	$U_{CC1}+0,5$	–	13,0
9 Напряжение программирования на выводе RESET#, В, $U_{CC1} = U_{CC2} = (4,5–5,5)$ В	U_{PR}	11,5	12,5	–	13,0
10 Выходной ток низкого уровня по выводам кроме BQ1 и RESET#, мА[2], $U_{CC1} = U_{CC2} = (4,5–5,5)$ В	I_{OL}	–	20	–	40
11 Выходной ток высокого уровня по выводам кроме BQ1 и RESET#, мА[2], $U_{CC1} = U_{CC2} = (4,5–5,5)$ В	I_{OH}	–	|–20|	–	|–40|
12 Длительность фронта и спада тактового сигнала, нс, - для входа BQ1 - для остальных входов	t_{LH}, t_{HL}	– –	10 20	– –	500 500
13 Частота следования импульсов тактового сигнала, МГц, $U_{CC1} = U_{CC2} = (4,5–5,5)$ В	f_{CI}	–	16	–	–
14 Емкость нагрузки по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, PE0 – PE7, PF0 – PF7, PG0 – PG4, пФ	C_L	–	80	–	120

1) Между напряжениями питания аналоговой части и цифровой части микросхемы должно сохраняться соотношение $|U_{CC1} – U_{CC2}| \le 0,3$ В.

2) Значение суммарного максимально допустимого тока по всем выводам – не более 400 мА, по выводам PA0 – PA7 – не более 100 мА, PB0 – PB3 – не более 100 мА, PB4 – PB7 – не более 100 мА, PC0 – PC3 – не более 100 мА, PC4 – PC7 – не более 100 мА, PD0 – PD3 и BQ2 – не более 100 мА, PD4 – PD7 – не более 100 мА.

Если выходной ток превышает условия тестирования, выходное напряжение может превышать заявленные значения. Выводы не гарантируют нагрузочный ток, превышающий условия тестирования.

3 Функциональное описание ИС 1887ВЕ7Т

В таблице 3.1 приведено описание выводов ИС 1887ВЕ7Т.

Таблица 3.1 – Описание выводов микросхемы

Название вывода	Описание
#VCC	Напряжение питания цифровой части ИС
#0V	Общий вывод цифровой части ИС
Порт A (PA7–PA0)	Порт A – 8-разрядный двунаправленный порт ввода-вывода с внутренними подтягивающими к питанию резисторами (выбираются отдельно для каждого разряда). Выходные буферы порта A имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. В режиме ввода линии порта A будут действовать как источник тока, если внешне задается низкий уровень и включены подтягивающие резисторы. Выводы порта A находятся в третьем (высокоимпедансном) состоянии при выполнении сброса, даже если не выполняется тактирование. Порт A также выполняет некоторые специальные функции, описываемые далее
Порт B (PB7–PB0)	Порт B – 8-разрядный двунаправленный порт ввода-вывода с внутренними подтягивающими к питанию резисторами (выбираются отдельно для каждого разряда). Выходные буферы порта B имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. В режиме ввода линии порта B будут действовать как источник тока, если внешне задается низкий уровень и включены подтягивающие резисторы. Выводы порта B находятся в третьем (высокоимпедансном) состоянии при выполнении сброса, даже если не выполняется тактирование. Порт B также выполняет некоторые специальные функции, описываемые далее
Порт C (PC7–PC0)	Порт C – 8-разрядный двунаправленный порт ввода-вывода с внутренними подтягивающими к питанию резисторами (выбираются отдельно для каждого разряда). Выходные буферы порта C имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. В режиме ввода линии порта C будут действовать как источник тока, если внешне задается низкий уровень и включены подтягивающие резисторы. Выводы порта C находятся в третьем (высокоимпедансном) состоянии при выполнении сброса, даже если не выполняется тактирование. Порт C также выполняет некоторые специальные функции, описываемые далее
Порт D (PD7–PD0)	Порт D – 8-разрядный двунаправленный порт ввода-вывода с внутренними подтягивающими к питанию резисторами (выбираются отдельно для каждого разряда). Выходные буферы порта D имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. В режиме ввода линии порта D будут действовать как источник тока, если внешне задается низкий уровень и включены подтягивающие резисторы. Выводы порта D находятся в третьем (высокоимпедансном) состоянии при выполнении сброса, даже если не выполняется тактирование. Порт D также выполняет некоторые специальные функции, описываемые далее
Порт E (PE7–PE0)	Порт E – 8-разрядный двунаправленный порт ввода-вывода с внутренними подтягивающими к питанию резисторами (выбираются отдельно для каждого разряда). Выходные буферы порта E имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. В режиме ввода линии порта E будут действовать как источник тока, если внешне задается низкий уровень и включены подтягивающие резисторы

Продолжение таблицы 3.1

Название вывода	Описание
Порт E (PE7–PE0)	Выводы порта E находятся в третьем (высокоимпедансном) состоянии при выполнении сброса, даже если не выполняется тактирование. Порт E также выполняет некоторые специальные функции, описываемые далее
Порт F (PF7–PF0)	Порт F действует как аналоговый вход аналого-цифрового преобразователя. Порт F также может использоваться как 8-разрядный двунаправленный порт ввода-вывода, если АЦП не используется. К каждой линии порта может быть подключен встроенный подтягивающий к питанию резистор (выбирается отдельно для каждого разряда). Выходные буферы порта F имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. В режиме ввода линии порта F будут действовать как источник тока, если внешне задается низкий уровень и включены подтягивающие резисторы. Выводы порта F находятся в третьем (высокоимпедансном) состоянии при выполнении сброса, даже если не выполняется тактирование. Если активизирован интерфейс JTAG, то подтягивающие резисторы на линиях PF7(TDI), PF5(TMS) и PF4(TCK) будут подключены, даже если выполняется сброс. Вывод TDO находится в третьем состоянии, пока TAP контроллер не начнет выдавать данные. Порт F также выполняет функции интерфейса JTAG
Порт G (PG4–PG0)	Порт G – 5-разрядный двунаправленный порт ввода-вывода с внутренними подтягивающими к питанию резисторами (выбираются отдельно для каждого разряда). Выходные буферы порта G имеют симметричную выходную характеристику с одинаковыми втекающим и вытекающим токами. В режиме ввода линии порта G будут действовать как источник тока, если внешне задается низкий уровень и включены подтягивающие резисторы. Выводы порта G находятся в третьем (высокоимпедансном) состоянии при выполнении условия сброса, даже если не выполняется тактирование. Порт G также выполняет некоторые специальные функции.
RESET#	Вход сброса. Если на этот вход подать низкий уровень, длительность которого больше минимально необходимой, то будет выполнен сброс независимо от работы синхронизации. Действие импульса меньшей продолжительности не гарантирует генерацию сброса
BQ1	B0,ход инвертирующего усилителя генератора и вход внутренней синхронизации
BQ2	Выход инвертирующего усилителя генератора
∩VCC	Вход питания порта F и аналого-цифрового преобразователя. Он должен быть внешне подключен к #VCC, если АЦП не используется. При использовании АЦП этот вывод подключается к #VCC через фильтр нижних частот
∩0V	Общий вывод аналоговой части ИС
AREF	Вход подключения источника опорного напряжения АЦП
PEN#	Вход разрешения программирования для режима последовательного программирования через интерфейс SPI. Если во время сброса по включению питания (Power-on-Reset) на этот вход подан низкий логический уровень, то микроконтроллер переходит в режим последовательного программирования через SPI. В рабочем режиме PEN# не выполняет никаких функций

3.1 Ядро микроконтроллера

В данном подразделе описывается архитектура ядра микроконтроллера 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 Обзор архитектуры

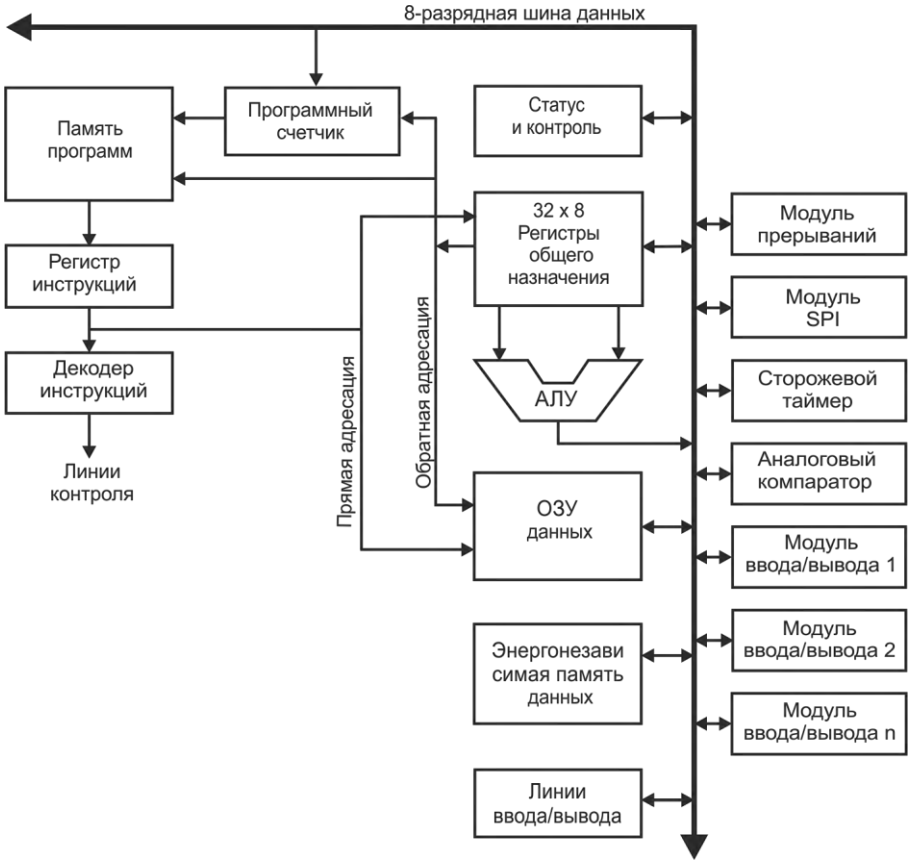


Рисунок 3.1 – Блок-схема архитектуры ядра микропрограммного управления

В целях достижения максимальной производительности и параллелизма используется Гарвардская архитектура с отдельной памятью программ и данных, а также шинами доступа к ним. Команды в памяти программ выполняются с одноуровневой конвейеризацией. В процессе выполнения одной инструкции, следующая инструкция предварительно считывается из памяти программ. Данная концепция позволяет выполнять одну инструкцию за один машинный цикл. Память программ представляет собой внутрисистемно перепрограммируемую память.

Регистровый файл с быстрым доступом содержит тридцать два 8-разрядных рабочих регистра общего назначения с однотактным циклом доступа. Благодаря этому достигается однотактность работы арифметико-логического устройства. При обычной работе АЛУ сначала из регистрового файла загружается два операнда, затем выполняется операция, а после результат отправляется обратно в регистровый файл, и все это происходит за один машинный цикл.

Шесть регистров из 32 могут использоваться как три 16-разрядных регистра косвенного адреса для эффективной адресации в пределах памяти данных. Один из этих указателей может также использоваться как указатель адреса для доступа к памяти программ. Данные 16-разрядные регистры называются X-регистр, Y-регистр и Z-регистр и описываются далее в этом подразделе.

АЛУ поддерживает арифметические и логические операции между регистрами, а также между константой и регистром. Кроме того, АЛУ поддерживает действия с одним регистром. После выполнения арифметической операции регистр статуса обновляется для отображения результата выполнения операции.

Для ветвления программы поддерживаются инструкции условных и безусловных переходов и вызовов процедур, которые способны напрямую обращаться ко всему адресному пространству. Большинство инструкций представляют собой одно 16-разрядное слово. Каждый адрес памяти программ содержит 16- или 32-разрядную инструкцию.

Память программ разделена на две секции: секция загрузчика и секция прикладной программы. Обе секции имеют специальные биты защиты от записи и чтения/записи. Инструкция SPM (запись в секцию прикладной программы) должна использоваться только внутри секции загрузчика.

При генерации прерывания и вызове подпрограмм адрес возврата из программного счетчика записывается в стек. Стек размещен в статическом ОЗУ памяти данных и, следовательно, размер стека ограничен только общим размером статического ОЗУ и его загруженностью. В любой программе сразу после сброса должна быть выполнена инициализация указателя стека SP (т.е. перед выполнением процедуры обработки прерываний или вызовом подпрограмм). Указатель стека – SP – доступен для чтения и записи в пространстве ввода-вывода. Доступ к статическому ОЗУ данных может быть осуществлен через пять различных режимов адресации, поддерживаемых архитектурой.

Модуль прерываний содержит свои управляющие регистры в пространстве ввода-вывода и имеет дополнительный бит общего разрешения работы системы прерываний в регистре статуса. У всех прерываний имеется свой вектор прерывания в соответствии с таблицей векторов. Прерывания имеют приоритет в соответствии с позицией их вектора. Чем ниже адрес вектора прерывания, тем выше его приоритет.

Пространство памяти ввода-вывода содержит 64 адреса с непосредственной адресацией или может адресоваться как память данных, следующая за регистрами по адресам $20 - $5F. Кроме того, ИС 1887ВЕ7Т имеет пространство расширенного ввода-вывода по адресам $60 - $FF в статическом ОЗУ, для доступа к которому могут использоваться только команды ST/STS/STD и LD/LDS/LDD.

3.1.2 Арифметико-логическое устройство

АЛУ микроконтроллера работает в непосредственной связи со всеми 32 регистрами общего назначения. АЛУ позволяет выполнить за один машинный цикл операцию между двумя регистрами или между регистром и константой. Операции АЛУ могут быть классифицированы на три основные группы: арифметические, логические и битовые. Кроме того, ИС 1887ВЕ7Т поддерживает операции умножения со знаком и без знака и дробным форматом. Смотрите приложение А для подробного ознакомления.

3.1.3 Регистр статуса

Регистр статуса содержит информацию о результате только что выполненной арифметической инструкции. Данная информация может использоваться для ветвления программы по условию. Регистр статуса обновляется после выполнения всех операций АЛУ в объеме, предусмотренном для каждой конкретной инструкции. Флаги этого регистра в большинстве случаев позволяют отказаться от использования инструкций сравнения, делая код программы более компактным и быстрым.

Состояние регистра статуса автоматически не запоминается при вызове процедуры обработки прерываний и не восстанавливается при выходе из нее. Это необходимо выполнить программно.

Регистр статуса SREG микроконтроллера имеет структуру, показанную на рисунке 3.2.

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

Рисунок 3.2 – Регистр статуса SREG

Разряд 7 – I: Общее разрешение прерываний

Бит общего разрешения прерываний используется для активации работы системы прерываний. Разрешение отдельных прерываний осуществляется в соответствующих управляющих регистрах. Если бит общего разрешения прерываний сбросить, то ни одно из прерываний не будет активным независимо от их индивидуальной конфигурации. Бит I сбрасывается в ноль аппаратно после генерации запроса на прерывание, а после выполнения инструкции возврата из прерывания RETI снова устанавливается в единицу для выполнения последующих прерываний. Бит I может также сбрасываться и устанавливаться с помощью инструкций CLI и SEI соответственно.

Разряд 6 – T: Хранение копируемого бита

Специальные битовые операции BLD (копирование из T-бита) и BST (копирование в T-бит) используют в качестве источника и получателя данных бит T. Любой бит из регистрового файла может быть скопирован в бит T инструкцией BST, а также содержимое бита T может быть скопировано в любой бит регистрового файла с помощью инструкции BLD.

Разряд 5 – H: Флаг половинного переноса

Данный флаг устанавливается при выполнении некоторых арифметических инструкций и индицирует о возникновении половинного переноса. Как правило, половинный перенос широко используется в двоично-десятичной арифметике. Более подробная информация приведена в описании системы команд.

Разряд 4 – S: бит знака, S = N ⊕ V

Бит S – результат выполнения логической операции исключающего ИЛИ между флагом отрицательного результата N и флагом переполнения двоичного дополнения V. Более подробная информация приведена в описании системы команд.

Разряд 3 – V: Флаг переполнения дополнения до двух

Флаг переполнения дополнения до двух V поддерживает арифметику с двоичным дополнением. Более подробная информация приведена в описании системы команд.

Разряд 2 – N: Флаг отрицательного результата

Флаг отрицательного результата N показывает, что результатом выполнения арифметической или логической операции является отрицательное значение. Более подробная информация приведена в описании системы команд.

Разряд 1 – Z: Флаг нулевого результата

Флаг нулевого результата Z показывает, что результатом выполнения арифметической или логической операции является ноль. Более подробная информация приведена в описании системы команд.

Разряд 0 – C: Флаг переноса

Флаг переноса C свидетельствует о возникновении переноса в результате выполнения арифметической или логической операции. Более подробная информация приведена в описании системы команд.

3.1.4 Файл регистров общего назначения

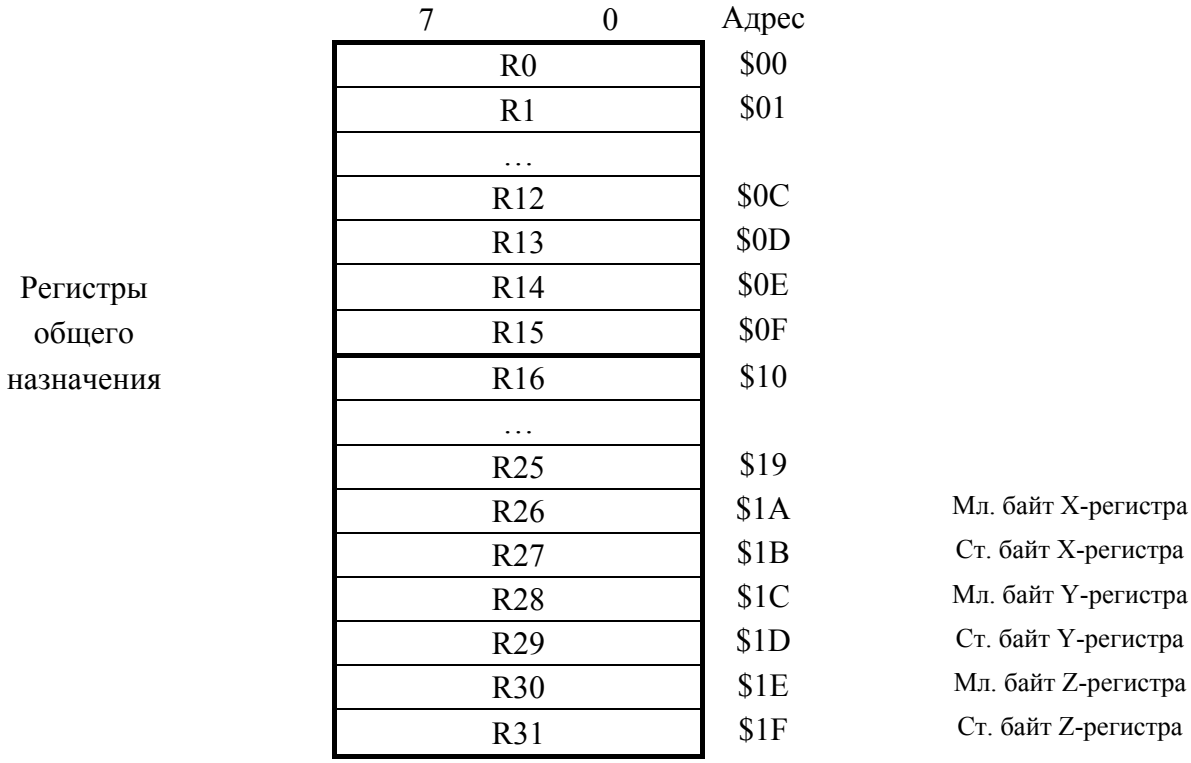
Файл регистров оптимизирован под расширенный набор инструкций микроконтроллера. В целях достижения требуемой производительности и гибкости файлом регистров поддерживаются следующие схемы ввода-вывода:

- один 8-разрядный операнд и один 8-разрядный результат;
- два 8-разрядных операнда и один 8-разрядный результат;
- два 8-разрядных операнда и один 16-разрядный результат;
- один 16-разрядный операнд и один 16-разрядный результат.

На рисунке 3.3 показана структура 32 регистров общего назначения в ЦПУ.

Большинство инструкций, работающих с регистровым файлом, имеют непосредственный доступ ко всем регистрам, чем достигается выполнение их за один машинный цикл.

Как показано на рисунке 3.3, каждый регистр имеет свой адрес в области памяти данных, для чего там отведены первые 32 позиции. Несмотря на физическую реализацию не по адресам статического ОЗУ, данная архитектура памяти обеспечивает гибкость доступа к регистрам благодаря тому, что регистры-указатели X, Y и Z могут быть использованы для обращения к любому регистру в файле.



Условные обозначения: мл. – младший, ст. – старший.

Рисунок 3.3 –Регистры общего назначения

X-регистр, Y-регистр и Z-регистр

Регистры R26–R31 обладают дополнительными функциями кроме их общецелевого использования. Данные регистры являются 16-разрядными указателями адреса для косвенной адресации в пределах памяти данных.

Три регистра косвенной адресации X, Y и Z представлены на рисунке 3.4.

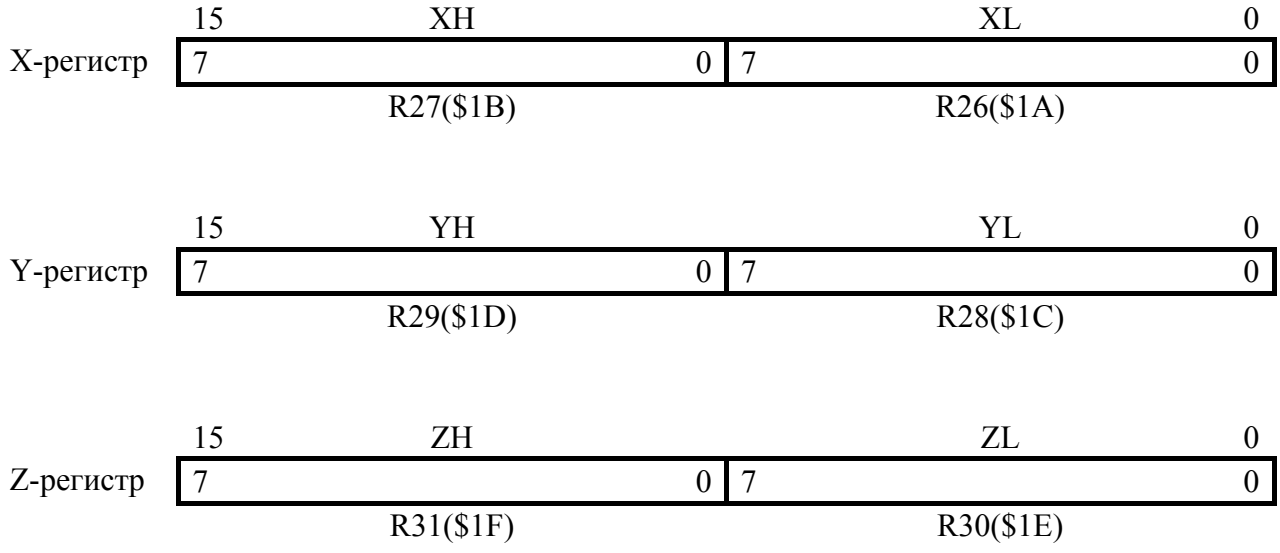


Рисунок 3.4 – X-, Y- и Z-регистры

В различных режимах адресации данные адресные регистры выполняют функции абсолютного смещения, автоматического инкрементирования и автоматического декрементирования (см. описание системы команд для более подробного изучения).

3.1.5 Указатель стека

Стек обычно используется для хранения временных данных, локальных переменных и адресов возврата при прерываниях и вызовах подпрограмм. Регистр указателя стека указывает на вершину стека. Обратите внимание на организацию стека, который направлен от старших к младшим позициям статического ОЗУ. Это означает, что команда помещения в стек PUSH уменьшает значение указателя стека.

Указатель стека определяет область стека в статическом ОЗУ данных, где расположены стеки прерываний и подпрограмм. Данная область стека в статическом ОЗУ памяти данных должна быть определена программно до вызова любой процедуры или разрешения прерываний. Устанавливаемое значение указателя стека должно быть более $60. Указатель стека однократно декрементируется при помещении данных в стек инструкцией PUSH и дважды декрементируется при помещении в стек адреса возврата при вызове подпрограммы или прерывании. Указатель стека однократно инкрементируется при извлечении данных из стека инструкцией POP и дважды инкрементируется при извлечении адреса возврата при выполнении инструкции выхода из подпрограммы RET или выхода из процедуры обработки прерываний RETI.

Указатель стека реализован как два 8-разрядных регистра в области ввода-вывода (см. рисунок 3.5).

Бит	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Чтение/запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Рисунок 3.5 – Указатель стека

Регистр Z выбора страницы памяти программ – RAMPZ

Структура регистра Z выбора страницы памяти – RAMPZ – представлена на рисунке 3.6.

Бит	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	-	RAMPZ0	RAMPZ
Чтение/ запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Рисунок 3.6 – Регистр Z выбора страницы ОЗУ

Разряды 7–1: зарезервированные разряды

Данные разряды считываются как ноль.

Разряд 0 – RAMPZ0: Расширенный Z-указатель страницы памяти программ

Данный регистр используется для выбора страницы памяти программ, доступ к которой осуществляется с помощью инструкций ELPM/SPM. Различные установки бита RAMPZ0 имеют следующий результат:

RAMPZ0 = 0: Инструкции ELPM/SPM осуществляют доступ к памяти программ в диапазоне адресов $0000 - $7FFF (младшие 64 Кбайт)
RAMPZ0 = 1: Инструкции ELPM/SPM выполняют доступ к памяти программ в диапазоне адресов $8000 - $FFFF (старшие 64 Кбайт)

Обратите внимание: действие инструкции LPM не зависит от установки RAMPZ.

3.1.6 Временные диаграммы выполнения команд

Ниже описываются общие концепции временных диаграмм доступа для выполнения команд. Центральное процессорное устройство работает от тактового сигнала clk_{CPU}, сформированного непосредственно от выбранного для микроконтроллера источника тактовых импульсов.

На рисунке 3.7 представлена параллельная процедура выборки команд и их выполнения, разрешенная Гарвардской архитектурой и концепцией регистрового файла с быстрым доступом. Это является базовой концепцией работы в режиме конвейеризации для получения производительности вплоть до 1 миллиона команд в секунду на 1 МГц.

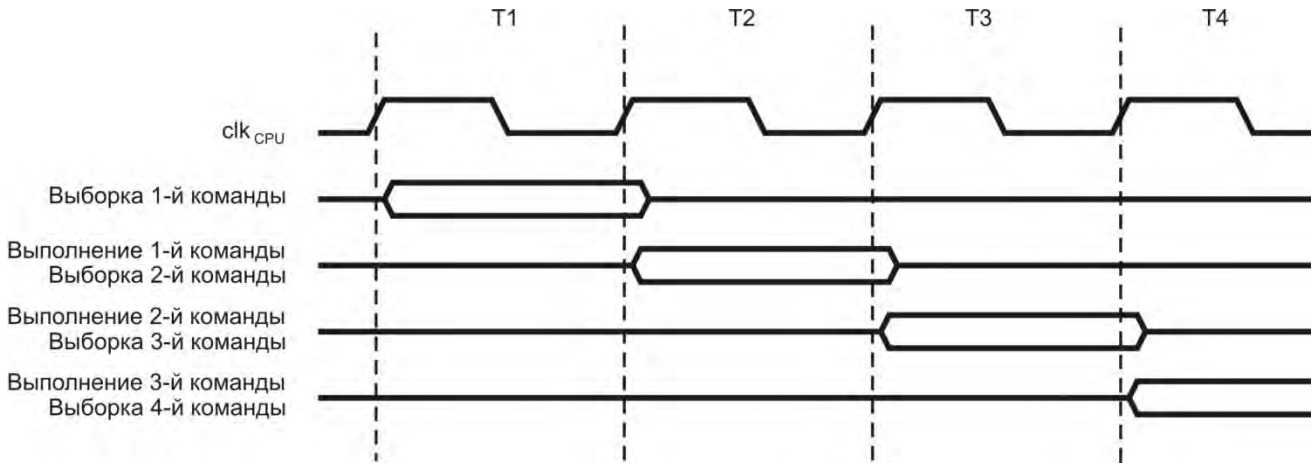


Рисунок 3.7 – Параллельная процедура выборки и выполнения команд

На рисунке 3.8 представлена временная диаграмма для регистрового файла. За один тактовый цикл выполняется операция АЛУ с использованием двух операндов регистра, и результат сохраняется обратно в регистр назначения.

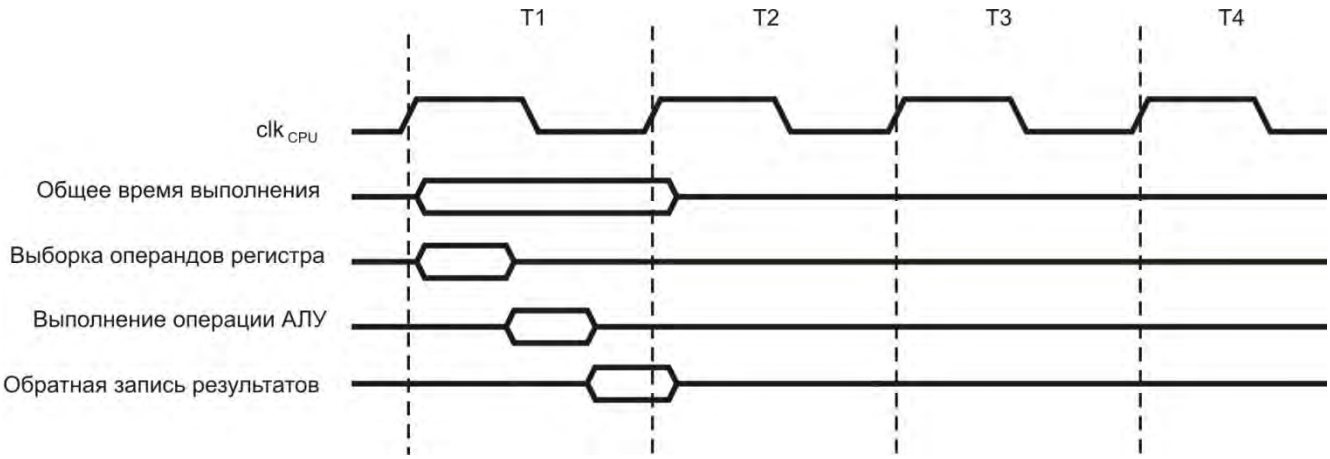


Рисунок 3.8 – Работа АЛУ в одиночном цикле

3.1.7 Сброс и обработка прерываний

ИС 1887ВЕ7Т содержит несколько источников прерываний. Каждое из этих прерываний, а также сброс имеют индивидуальный вектор в памяти программ. Все прерывания имеют индивидуальные биты разрешения, которые должны быть установлены вместе с установкой бита глобального разрешения прерываний в регистре статуса для того, чтобы разрешить прерывание. В зависимости от значения программного счетчика прерывания могут быть автоматически отключены, если запрограммировать биты защиты загрузочного сектора BLB02 или BLB12. Данная функция улучшает защиту программы. См. подраздел 3.21 «Программирование памяти» для уточнения деталей.

Младшие адреса в пространстве памяти программ, по умолчанию, определены как векторы сброса и прерывания. Перечень прерываний устанавливает также уровень приоритетов для различных прерываний. Чем ниже адрес, тем выше уровень приоритета. Сброс (RESET#) имеет наивысший приоритет, а следующим является INT0 – запрос внешнего прерывания. Векторы прерывания могут быть перемещены в начало сектора загрузчика памяти программ установкой бита IVSEL в регистре управления микроконтроллером (MCUCR). См. подраздел 3.6 «Прерывания» для более подробного ознакомления. Вектор сброса может быть также перемещен в начало сектора загрузчика памяти программ путем программирования конфигурационного бита BOOTRST (см. подраздел 3.20 «Самопрограммирование из сектора начальной загрузки с поддержкой чтения во время записи»).

При возникновении прерывания бит I общего разрешения прерываний очищается, и все прерывания блокируются. Пользовательское ПО может записывать логическую единицу в бит I для разрешения вложенных прерываний. Все разрешенные прерывания могут прерывать текущую программу обработки прерываний. Бит I устанавливается автоматически при выполнении возврата из команды прерывания RETI.

Существует два основных типа прерываний. Первый тип запускается событием, которое устанавливает флаг прерывания. Для этого типа прерываний программный счетчик переходит на вектор прерывания для исполнения подпрограммы обработки прерываний, при этом аппаратно сбрасывается соответствующий флаг прерывания. Флаг прерывания также можно сбросить путем записи логической единицы в соответствующий разряд регистра флагов. Если возникает условие прерывания, в то время как соответствующий бит прерывания сброшен, флаг прерывания будет установлен и запомнен до разрешения прерывания или он будет сброшен программным способом. Подобным образом, если возникает одно или несколько условий прерывания при сброшенном бите глобального разрешения прерываний I, соответствующий флаг(и) будет установлен и запомнен, пока не установится I, а затем будет выполнена обработка прерываний в порядке приоритета.

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

После выхода из прерывания микроконтроллер всегда возвращается к основной программе и выполняет еще одну команду до обслуживания любого ждущего прерывания.

Обратите внимание на то, что регистр статуса не сохраняется автоматически при переходе к процедуре обработки прерываний и не восстанавливается при возврате из нее. Это должно выполняться программно.

При использовании команды CLI для блокировки прерывания, прерывания будут немедленно блокированы. 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 появления прерываний во время запланированной последовательности записи ЭПД.

Пример кода на Ассемблере:

```
in r16, SREG       ; чтение значения SREG
cli                ; блокировка прерываний в течение отработки временной
                   ; последовательности
sbi EECR, EEMWE    ; старт записи в ЭПД
sbi EECR, EEWE
out SREG, r16      ; восстановление значения SREG (бит I)
```

Пример кода на Си:

```
char cSREG;
cSREG = SREG;         /* чтение значения SREG */
                      /* блокировка прерываний в течение отработки */
                      /* временной последовательности */
_CLI();
EECR |= (1<<EEMWE);   /* старт записи в память данных */
EECR |= (1<<EEWE);
SREG = cSREG;         /* восстановление значения SREG (бит I)*/
```

При использовании команды SEI для разрешения прерываний инструкция, следующая за SEI, будет выполнена прежде любого ожидающего прерывания, как это показано в примере ниже.

Пример кода на Ассемблере:

```
sei      ; установка глобального разрешения прерываний
sleep    ; включение режима sleep, ожидание прерывания
         ; примечание: переход в режим sleep будет осуществлен
         ; до обработки любого ожидающего прерывания(ий)
```

Пример кода на Си:

```
_SEI();     /* установка глобального разрешения прерываний */
_SLEEP();   /* включение режима sleep, ожидание прерывания */
            /* примечание: переход в режим sleep будет осуществлен */
            /* до обработки любого ожидающего прерывания(ий) */
```

Время отклика на прерывание

Отклик на выполнение прерывания для всех разрешенных прерываний составляет минимум четыре тактовых цикла. Во время периода четырех тактовых циклов значение счетчика программ заносится в стек. Вектор обычно переходит в процедуру обработки пре-

рывания, и этот переход занимает три тактовых цикла. Если прерывание происходит во время исполнения команды, состоящей из нескольких циклов, то команда завершается до того, как обслуживается прерывание. Если прерывание происходит когда микроконтроллер находится в спящем режиме, то время отклика на выполнение прерывания составляет сумму времени выхода из спящего режима и длительности четырех тактовых циклов.

Возвращение из процедуры обработки прерывания обычно занимает четыре тактовых цикла. В течение этих четырех циклов значение программного счетчика (два байта) восстанавливается из стека, указатель стека увеличивается на два и устанавливается бит I в SREG.

3.2 Память

В данном подразделе описываются различные виды памяти ИС 1887ВЕ7Т. В соответствии с Гарвардской архитектурой память микроконтроллера разделена на две области: память данных и память программ. Кроме того, устройство содержит память для энергонезависимого хранения данных. Все три области памяти являются линейными и регулярными.

3.2.1 Память программ

ИС 1887ВЕ7Т содержат 128 Кбайт встроенной памяти программ с возможностью внутрисистемного программирования. Поскольку все инструкции являются 16- или 32-разрядными, то память программ организована как 64 Кбит × 16. Для защиты ПО память программ разделена на два сектора: сектор загрузчика и сектор прикладной программы (см. рисунок 3.9).

Рисунок 3.9 – Карта памяти программ

Количество циклов перезаписи памяти программ – 100 000. Срок хранения данных – 10 лет.

Программный счетчик является 16-разрядным, поэтому позволяет адресовать 64 Кбайт памяти программ. Работа сектора загрузчика и связанных с ним бит блокировки загрузки для защиты ПО детально описана в подразделе 3.20 «Самопрограммирование из сектора загрузчика с поддержкой чтения во время записи». В подразделе 3.21 «Программирование памяти» детально описывается параллельное программирование памяти программ и последовательное программирование через интерфейсы SPI, JTAG.

При записи данных в массив памяти необходимость в предварительном стирании отсутствует. Записываемые ячейки памяти будут содержать обновленную информацию. Содержимое остальных ячеек памяти останется без изменения со времени последней модификации (записи/стирания). Состояние ячеек памяти после команд стирания – 0000. Запись можно производить как постранично (размер страницы – тридцать два 16-разрядных слова), так и пословно (запись одного 16-разрядного слова).

Режим записи одного слова полезен в случае самопрограммирования контроллера. Если необходимо изменить одно или несколько слов на странице памяти программ, то нет необходимости производить предварительное считывание и сохранение в буфере неизменяемых данных со страницы памяти. Подробно описание данной возможности описано в разделе об управлении процессом самопрограммирования микроконтроллера.

Таблицы констант могут располагаться в пределах всего пространства памяти программ.

Временные диаграммы выборки и выполнения инструкций представлены в 3.1.6 «Временные диаграммы выполнения команд».

3.2.2 Память данных статического ОЗУ

ИС 1887ВЕ7Т имеет следующую конфигурацию памяти данных статического ОЗУ: 4096 байт внутренней памяти данных и до 64 Кбайт внешней памяти данных.

На рисунке 3.10 показано, как организована память статического ОЗУ.

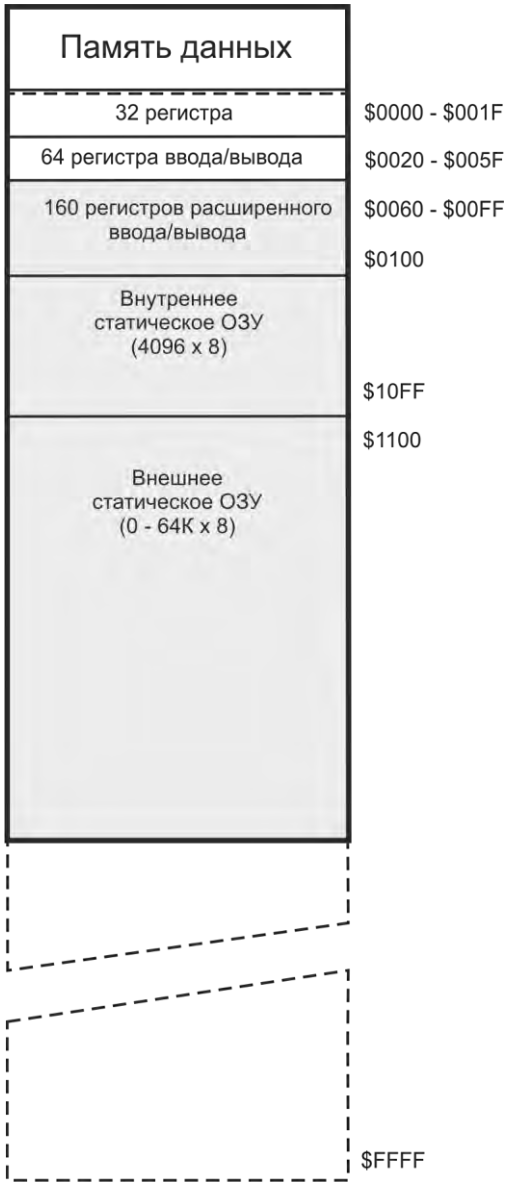


Рисунок 3.10 – Карта памяти данных

ИС 1887ВЕ7Т – микроконтроллер с набором периферийных устройств, которые управляются через 64 регистра, зарезервированные в кодах инструкций IN и OUT. Для расширенной области ввода-вывода в статическом ОЗУ по адресам $60 - $FF необходимо использовать только инструкции ST/STS/STD и LD/LDS/LDD.

Первые 4352 ячейки памяти данных относятся к регистровому файлу, памяти ввода-вывода, расширенной памяти ввода-вывода, и к встроенному статическому ОЗУ данных. В первых 32 ячейках расположен регистровый файл, следующие 64 ячейки занимает стандартная память ввода-вывода, а за ними следуют 160 ячеек расширенной памяти ввода-вывода. Замыкают внутреннюю память данных 4096 ячеек внутреннего статического ОЗУ данных.

В ИС 1887ВЕ7Т существует возможность использования внешнего статического ОЗУ. Это статическое ОЗУ будет занимать оставшуюся часть адресного пространства размером 64 Кбайт. Внешняя память начинается с адреса, следующего за последним адресом внутреннего статического ОЗУ. Регистровый файл, память ввода-вывода, память расширенного ввода-вывода и внутреннее статическое ОЗУ занимают младшие 4352 байт. Таким образом, при использовании внешней памяти размером 64 Кбайт (65536 байт) из них будет доступно 61184 байт. См. 3.2.5 «Интерфейс внешней памяти» для детального изучения методов использования внешней памяти.

Доступ к внешнему статическому ОЗУ осуществляется автоматически с помощью тех же инструкций, что и для внутреннего ОЗУ, если указанное значение адреса находится за пределами внутренней памяти данных. При адресации внутренней памяти сигналы чтения и записи внешней памяти (выводы PG0 и PG1) не активны в процессе всего цикла доступа. Работа внешнего статического ОЗУ разрешается путем установки бита SRE в регистре MCUCR.

Доступ к внешнему статическому ОЗУ требует еще один машинный цикл на байт, по сравнению с доступом к внутреннему статическому ОЗУ. Это означает, что на выполнение команд LD, ST, LDS, STS, LDD, STD, PUSH и POP потребуется один дополнительный цикл. Если стек будет размещен во внешнем статическом ОЗУ, то вызов и возврат из подпрограмм и процедур обработки прерываний будет длиться на три машинных цикла дольше. Это будет происходить вследствие помещения в стек и извлечения из стека двухбайтного счетчика программ, а также не использования во время доступа к внешней памяти преимущества конвейерного доступа к внутренней памяти. Если интерфейс внешнего статического ОЗУ используется с состояниями ожидания (со сниженным быстродействием), то однобайтный внешний доступ потребует 2, 3 или 4 дополнительных машинных цикла для 1, 2 и 3 состояний ожиданий, соответственно. Таким образом, вызов и возврат из прерываний и подпрограмм потребует еще 5, 7 и 9 машинных циклов (в отличие от значений, приведенных в описании системы команд) для 1, 2 и 3 состояний ожидания соответственно.

Реализовано пять различных способов адресации для охвата всей памяти данных: прямая, косвенная со смещением, косвенная, косвенная с предварительным декрементом и косвенная с последующим инкрементом. Регистры с R26 по R31 из регистрового файла используются как регистры-указатели для косвенной адресации.

Прямая адресация позволяет обращаться ко всему пространству памяти данных.

Косвенная адресация со смещением позволяет адресовать 63 ячейки, начиная с адреса, указанного в регистре Y или Z.

При использовании команд косвенной адресации с предварительным декрементом и последующим инкрементом значения адресных регистров X, Y и Z, соответственно, декрементируются до или инкрементируются после выполнения инструкции.

32 рабочих регистра общего назначения, 64 регистра ввода-вывода и 4096 байт внутреннего статического ОЗУ данных в ИС 1887ВЕ7Т доступны с помощью всех этих режимов адресации. Регистровый файл описывается в 3.1.4 «Файл регистров общего назначения».

Доступ к памяти данных внутреннего ОЗУ

Ниже описывается общая концепция доступа к внутренней памяти данных.

Доступ к внутреннему статическому ОЗУ выполняется за два машинных цикла, как показано на рисунке 3.11.

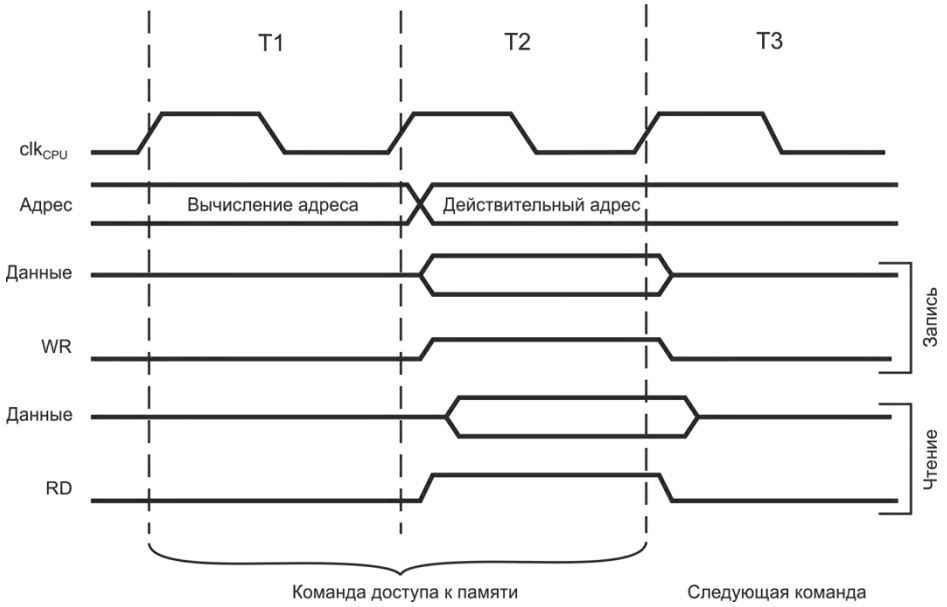


Рисунок 3.11 – Временная диаграмма доступа к данным внутреннего статического ОЗУ

3.2.3 Энергонезависимая память данных (ЭПД)

ИС 1887ВЕ7Т содержат 4 Кбайт энергонезависимой памяти данных. Память организована как отдельная область памяти данных, в которой каждый байт может быть записан и считан.

Количество циклов перезаписи памяти данных – 100 000. Срок хранения данных – 10 лет.

В подразделе 3.21 «Программирование памяти» содержится детальное описание программирования энергонезависимой памяти данных через интерфейсы SPI, JTAG и параллельный интерфейс программирования.

При записи данных в массив памяти необходимость в предварительном стирании отсутствует. Состояние ячеек памяти после команд стирания – ноль. Запись можно производить как постранично с размером страницы 32 байта, так и побайтно.

Чтение/запись ЭПД

Доступ к ЭПД осуществляется через специальные регистры, расположенные в пространстве ввода-вывода.

Время записи в ЭПД приведено в таблице 3.2. Функция выдержки времени записи позволяет программно определить возможность записи следующего байта. Если код программы содержит инструкции записи в ЭПД, то должны быть предприняты некоторые меры предосторожности. У источников питания с хорошей фильтрацией напряжение #VCC медленно нарастает/спадает при подаче/снятии питания. По этой причине напряжение питания микроконтроллера в течение некоторого периода времени может оказаться ниже, чем требуется для корректной работы микросхемы. См. подпункт «Предотвращение повреждения данных в ЭПД» для детального изучения методов разрешения данной проблемы.

Таблица 3.2 – Время программирования ЭПД

Операция	Количество периодов калиброванного RC-генератора*	Типовое время программирования, мс
Запись ЭПД (через ЦПУ)	8448	8,5

_____
* Используется частота 1 МГц, независимо от установки конфигурационного бита CKSEL.

В целях предотвращения неумышленной записи в ЭПД должна быть выполнена специфическая процедура записи. Детально этот процесс рассматривается при описании управляющего регистра ЭПД.

Во время считывания ЭПД ЦПУ задерживается на четыре машинных цикла до выполнения следующей инструкции. Во время записи в ЭПД ЦПУ задерживается на два машинных цикла до выполнения следующей инструкции.

Адресные регистры ЭПД – EEARH и EEARL

Бит	15	14	13	12	11	10	9	8	
	-	-	-	-	EEAR11	EEAR10	EEAR9	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Чтение/ запись	Ч Ч/З	Ч Ч/З	Ч Ч/З	Ч Ч/З	Ч/З Ч/З	Ч/З Ч/З	Ч/З Ч/З	Ч/З Ч/З	
Начальное значение	0 X	0 X	0 X	0 X	X X	X X	X X	X X	

Разряды 15–12: зарезервированы

Данные разряды всегда считываются как ноль.

Разряды 11–0: EEAR11–0: Адрес ячейки ЭПД

Адресные регистры ЭПД – EEARH и EEARL – определяют адрес ячейки ЭПД в пространстве размером 4 Кбайт. Байтные ячейки ЭПД адресуются линейно в диапазоне адресов 0...4096. Начальное значение EEAR неопределенное. Необходимое значение адреса должно быть записано до начала доступа к ЭПД.

Регистр данных ЭПД – EEDR

Бит	7	6	5	4	3	2	1	0	
	Старший бит							Младший бит	EEDR
Чтение/ запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7–0: EEDR7–0: Данные ЭПД

Для выполнения записи в ЭПД в регистр EEDR необходимо помещать данные, которые будут записаны по адресу, указанному в регистре EEAR. При выполнении операции чтения из ЭПД в регистре EEDR содержатся считанные данные из ячейки с адресом, указанным в EEAR.

Регистр управления ЭПД – EECR

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

Разряды 7–4 зарезервированы

Данные разряды считываются как 0.

Разряд 3–EERIE: Разрешение прерывания по готовности ЭПД

Запись единицы в EERIE разрешает прерывание по готовности ЭПД, если установлен бит I в регистре SREG. Запись нуля в EERIE запрещает это прерывание. Прерывание по готовности ЭПД генерируется, если бит EEWE сброшен.

Разряд 2–EEMWE: Разрешение записи в ЭПД

Бит EEMWE разрешает установку бита EEWE, инициирующего запись в ЭПД. Данные будут записаны в ЭПД по указанному адресу, если в EEMWE записать единицу, а затем в течение четырех машинных циклов записать единицу в EEWE. Если EEMWE=0, то запись логической единицы в EEWE не вызовет никаких действий. После программной установки бита EEMWE он автоматически сбрасывается аппаратно по истечении четырех машинных циклов.

Разряд 1 – EEWE: Запись в ЭПД

Для записи в ЭПД после корректной установки адреса и данных необходимо установить бит EEWE. Перед установкой бита EEWE должен быть установлен бит EEMWE, иначе запись в ЭПД не произойдет. При выполнении операции записи в ЭПД необходимо руководствоваться следующей последовательностью (порядок шагов 3 и 4 не важен):

1. Ожидание, пока EEWE не станет равным нулю.
2. Ожидание, пока бит SPMEN в регистре SPMCSR не станет равным нулю.
3. Запись нового адреса ЭПД в EEAR (не обязательно).
4. Запись новых данных в регистр EEDR для записи в ЭПД (не обязательно).
5. Запись логической единицы в EEMWE одновременно с записью нуля в бит EEWE регистра EECR.
6. Запись логической единицы в EEWE в течение четырех машинных циклов после установки EEMWE.

ЭПД нельзя программировать во время записи в память программ из ЦПУ. С учетом этого перед началом новой записи в ЭПД необходимо проверить завершение программирования памяти программ. Шаг 2 необходимо выполнять, если в приложении используется программирование из загрузочного сектора. Если программирование памяти программ под управлением ЦПУ не предусмотрено, то шаг 2 можно пропустить. См. подраздел 3.20 «Самопрограммирование из сектора загрузчика с поддержкой чтения во время записи» для детального изучения программирования из загрузочного сектора.

Прерывание между шагами 5 и 6 может нарушить цикл записи из-за превышения установленного предела времени на выполнение этих шагов. Если процедура обработки прерывания, осуществляющая доступ к ЭПД, прерывается другим доступом к ЭПД, то EEAR или EEDR изменятся, вызывая сбой прерванного цикла доступа. Во избежание этих проблем рекомендуется сбрасывать флаг общего разрешения прерываний при выполнении последних четырех шагов.

По окончании записи бит EEWE сбрасывается аппаратно. Данный бит может опрашиваться программно для определения возможности записи следующего байта (нулевое значение). После установки EEWE ЦПУ останавливается на два машинных цикла перед выполнением следующей инструкции.

Разряд 0 – EERE: Чтение из ЭПД

После записи корректного адреса в регистр адреса EEAR бит EERE должен быть установлен для запуска механизма чтения ЭПД. Для операции чтения из ЭПД требуется одна инструкция, поэтому запрашиваемые данные доступны для считывания сразу по ее завершении. После выполнения чтения из ЭПД ЦПУ останавливается на четыре машинных цикла и только затем выполняет следующую инструкцию.

Пользователь должен опросить флаг EEWE до начала операции чтения. Если осуществляется операция записи, то невозможно не только считать ЭПД, но и изменить регистр адреса EEAR. Для отсчета времени записи в ЭПД используется внутренний калиброванный генератор.

Далее представлены примеры кодов функций записи в ЭПД на языках Ассемблер и Си. В данных примерах предполагается, что ни одно прерывание не возникает в процессе выполнения данных функций (например, путем общего отключения прерываний). Кроме того, считается, что из сектора загрузчика не выполняется программирование памяти программ. В противном случае функция записи в ЭПД должна ожидать окончания действия инструкции SPM.

Пример кода на Ассемблере:

```
EEPROM_write:
; Ожидание окончания предыдущей записи
sbic EECR,EEWE
rjmp EEPROM_write
; Запись адреса (r18:r17) в адресный регистр ЭПД
out EEARH, r18
out EEARL, r17
; Запись данных (r16)  в регистр данных ЭПД
out EEDR,r16
; Запись логической единицы в EEMWE
sbi EECR,EEMWE
; Старт записи в ЭПД путем установки бита EEWE
sbi EECR,EEWE
ret
```

Пример кода на Си:

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
/* Ожидание окончания предыдущей записи */
while(EECR & (1<<EEWE))
;
/* Указание адреса и данных */
EEAR = uiAddress;
EEDR = ucData;
/* Запись логической единицы в EEMWE */
EECR |= (1<<EEMWE);
/* Старт записи в ЭПД путем установки бита EEWE */
EECR |= (1<<EEWE);
}
```

В следующих примерах кодов представлены функции чтения из ЭПД. При разработке примеров учитывалось управление прерываниями таким образом, что ни одно из них не возникает в процессе выполнения этих функций.

Пример кода на Ассемблере:

```
EEPROM_read:
; Ожидание завершения предыдущей записи
sbic EECR,EEWE
rjmp EEPROM_read
; Установка адреса (r18:r17) в адресном регистре
```

```
out EEARH, r18
out EEARL, r17
; Запуск чтения ЭПД путем установки бита EERE
sbi EECR,EERE
; Считывание данных из регистра данных ЭПД
in r16,EEDR
ret
```

Пример кода на Си:

```
unsigned char EEPROM_read(unsigned int uiAddress)
{
/* Ожидание  завершения предыдущей записи*/
while(EECR & (1<<EEWE))
;
/* Установка адресного регистра */
EEAR = uiAddress;
/* Запуск чтения ЭПД путем установки бита EERE */
EECR |= (1<<EERE);
/* Возврат данных из регистра данных ЭПД*/
return EEDR;
}
```

Запись в ЭПД в режиме микропотребления

Когда микроконтроллер переводится в режим микропотребления в процессе выполнения операции записи в ЭПД, операция записи будет продолжена и завершится по истечении требуемого времени доступа для записи. Однако по завершении операции записи кварцевый генератор будет продолжать работу, и как следствие, микроконтроллер будет переведен в режим микропотребления не полностью. С учетом этого рекомендуется проверять окончание операции записи в ЭПД до перевода устройства в режим микропотребления.

Предотвращение повреждения данных в ЭПД

В те моменты, когда напряжение питания находится на уровне, недостаточном для корректной работы ЦПУ и ЭПД, содержимое ЭПД может быть нарушено. Такие проблемы также характерны для устройств, использующих внешнюю ЭПД, поэтому в данном случае необходимо применить аналогичные меры.

При пониженном напряжении питания существует две причины нарушения содержимого ЭПД. Первая причина заключается в том, что для коректного выполнения процедуры записи в ЭПД требуется наличие минимально необходимого напряжения питания. Вторая – в том, что ЦПУ может выполнять инструкции некорректно, если напряжение питания слишком низкое.

Повреждение данных в ЭПД может быть предотвращено, если придерживаться следующих рекомендаций. Микроконтроллер необходимо удерживать в состоянии сброса (низкий уровень на выводе RESET#) при недостаточном уровне питания. Аналогично это можно выполнить, разрешив работу встроенного детектора питания (BOD). Если пороговый уровень встроенного детектора питания не соответствует необходимому порогу, то следует применить внешнюю схему сброса при снижении напряжении питания (супервизор питания). Если сброс возникает в процессе операции записи, то запись будет завершена при условии достаточности уровня питания.

3.2.4 Память ввода-вывода

Все порты ввода-вывода и периферийные устройства в ИС 1887ВЕ7Т размещены в пространстве ввода-вывода. Доступ ко всем ячейкам ввода-вывода может быть осуществлен с помощью инструкций LD/LDS/LDD и ST/STS/STD путем передачи данных между од-

ним из 32 регистров общего назначения и памятью ввода-вывода. Регистры ввода-вывода с адресами $00-$1F могут побитно адресоваться с помощью инструкций SBI и CBI. Состояние одного из разрядов в этих регистрах может тестироваться с помощью инструкций SBIS и SBIC. При использовании специфических команд ввода-вывода IN и OUT необходимо использовать адреса $00-$3F. Если адресоваться к регистрам ввода-вывода как к памяти данных с помощью инструкций LD и ST, то к указанным выше адресам необходимо прибавить $20. В микроконтроллере 1887ВЕ7Т 64 адреса, зарезервированных в кодах операций IN и OUT, не достаточно для поддержки всех имеющихся периферийных устройств. Для расширенной области ввода-вывода, которая находится по адресам $60 - $FF в статическом ОЗУ, необходимо использовать только инструкции ST/STS/STD и LD/LDS/LDD.

В зарезервированные адреса ввода-вывода памяти запись проводиться не должна.

Некоторые флаги статуса сбрасываются путем записи в них логической единицы. Инструкции CBI и SBI работают только с регистрами по адресам $00-$1F.

Регистры управления вводом-выводом и периферийными устройствами описываются в следующих разделах.

3.2.5 Интерфейс внешней памяти

Характеристики интерфейса внешней памяти позволяют использовать его не только для подключения к внешнему статическому ОЗУ или Flash-памяти, но и в качестве интерфейса с внешними периферийными устройствами, например, ЖК-дисплеи, АЦП и ЦАП. Его основными отличительными особенностями являются:

- Возможность задания четырех различных по длительности состояний ожидания, включая отсутствие состояния ожидания.
- Возможность установки различных состояний ожидания для разных секторов внешней памяти (размер сектора конфигурируется).
- Возможность выбора количества задействованных разрядов в старшем адресном байте.
- Устройство запоминания состояния шины для минимизации потребления тока (дополнительно).

Краткий обзор

После разрешения внешней памяти (XMEM) становится доступным адресное пространство за пределами внутреннего статического ОЗУ благодаря специальным выводам. Конфигурация памяти показана на рисунке 3.12.

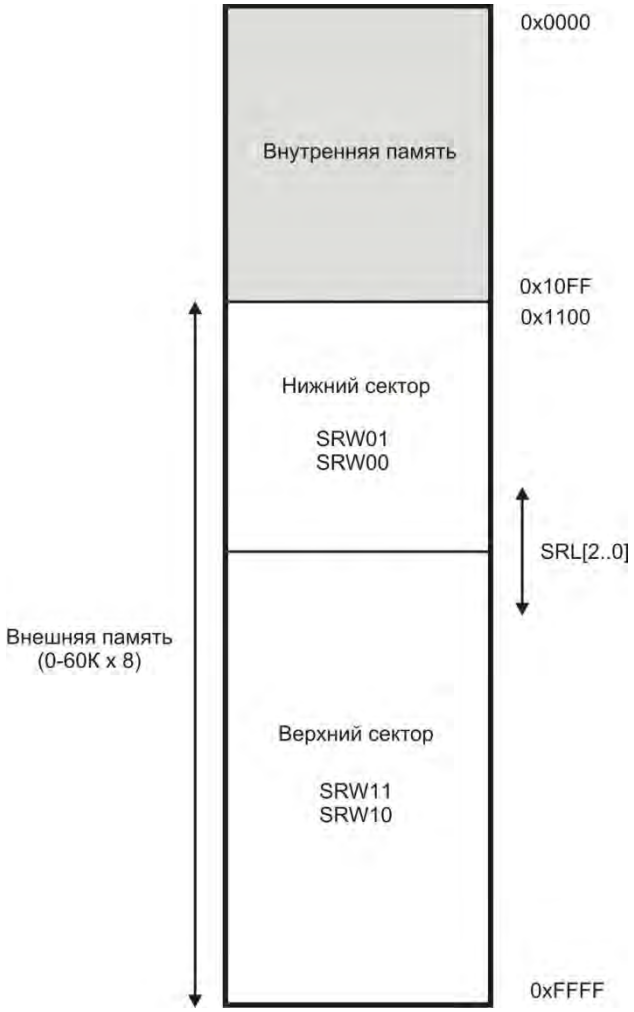


Рисунок 3.12 – Внешняя память с выбором сектора

Использование интерфейса внешней памяти

Интерфейс состоит из:
- AD7–0: мультиплексированной шины младшего адреса/шины данных;
- A15–8: шины старшего адреса (с конфигурируемым числом разрядов);
- ALE: сигнала разрешения фиксации адреса;
- RD#: сигнала чтения из внешней памяти;
- WR#: сигнала записи во внешнюю память.

Биты управления интерфейсом внешней памяти расположены в трех регистрах: регистре управления микроконтроллером (MCUCR), регистре A управления внешней памятью (XMCRA) и регистре B управления внешней памятью (XMCRB).

После разрешения работы интерфейс XMEM переопределяет функцию общего назначения портов ввода-вывода, линии которых предопределены для выполнения функций интерфейса XMEM. Более подробная информация об изменении настроек порта приведена в подразделе 3.7 «Порты ввода-вывода» при рассмотрении альтернативных функций. Интерфейс XMEM автоматически определяет к какой памяти – внешней или внутренней – осуществляется доступ. Во время доступа к внешней памяти интерфейс XMEM будет формировать сигналы шин адреса, данных и управления на линиях порта в соответствии со схемой на рисунке 3.14 (на рисунке представлена временная диаграмма доступа к XMEM без состояний ожидания). При переходе сигнала ALE из единицы в ноль на линиях AD7:0 будет присутствовать действительный адрес. Сигнал ALE находится в низком логическом состоянии во время передачи данных. После разрешения работы интерфейса XMEM доступ к внутренней памяти будет вызывать изменения на шинах данных и адреса, а также сигнала

ALE, при этом сигналы RD# и WR# останутся неизменными. После запрета работы интерфейса внешней памяти используются обычные установки выводов и направления данных. На рисунке 3.13 показывается как подключить внешнее статическое ОЗУ к микроконтроллеру с помощью 8-разрядного регистра, который передает данные напрямую при высоком уровне на входе G.

Требования по фиксации адреса

К основным параметрам, характеризующим фиксацию адреса, относятся:
- Длительность задержки на распространение сигнала с входа D на выход Q (t_{PD}).
- Время установки данных, перед тем как G станет равным нулю (t_{SU}).
- Время удержания данных (адреса) после установки низкого уровня на входе G (t_H).

Интерфейс внешней памяти разработан с учетом того, что после приложения низкого уровня на вход регистра G минимальное время удержания адреса t_H составляет 5 нс. Задержка распространения со входа D на выход Q (t_{PD}) должна быть учтена при вычислении требования по времени доступа к внешней памяти. Время установки данных, перед тем как G станет равным нулю (t_{SU}), должно не превышать разности между временем действительного адреса и низким логическим уровнем ALE (t_{AVLLC}) и задержкой в печатных проводниках (определяют емкостную нагрузку).

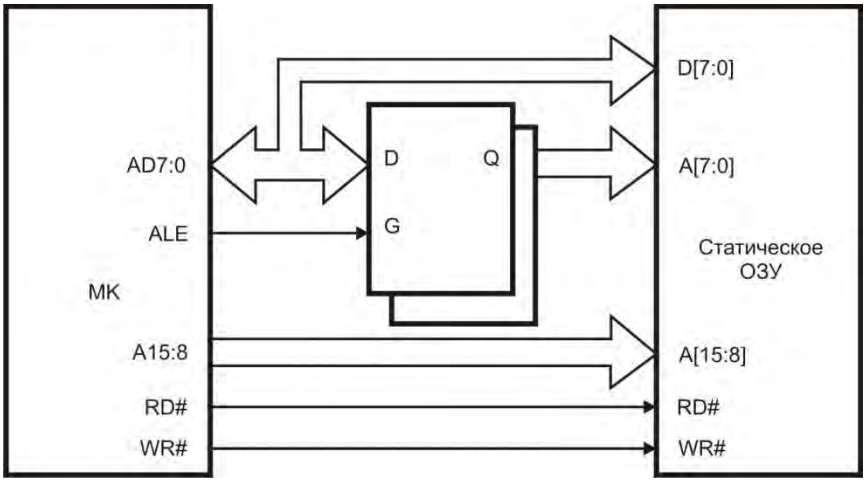


Рисунок 3.13 – Подключение внешнего статического ОЗУ к микроконтроллеру

Подтягивающие резисторы и устройство запоминания состояния шины

Подтягивающие к питанию резисторы на линиях AD7:0 могут быть активированы, если записать логические единицы в регистр соответствующего порта. Для снижения потребляемой мощности в режиме сна рекомендуется отключать подтягивающие резисторы путем записи нуля в регистр порта непосредственно перед входом в режим сна.

Интерфейс XMEM также содержит устройство запоминания состояния шины на линиях AD7:0. Это устройство может быть программно подключено и отключено, как описано в подпункте «Регистр B управления внешней памятью – XMCRB». После активизации устройство запоминания состояния шины будет сохранять предыдущее состояние шины AD7:0 при переводе этих линий интерфейсом XMEM в третье состояние.

Временная диаграмма

Микросхемы внешней памяти характеризуются различными параметрами временных диаграмм. Для удовлетворения этих требований интерфейс XMEM ИС 1887ВЕ7Т обеспечивает различное количество состояний ожидания (см. таблицу 3.4). Перед выбором

состояний ожидания очень важно уточнить требования к временной диаграмме микросхемы внешней памяти. Время доступа к внешней памяти определяется как промежуток времени с момента выбора микросхемы памяти и установки адреса до появления на шине действительных данных, соответствующих указанному адресу. Время доступа не может превышать времени с момента установки сигнала ALE в низкое логическое состояние до стабильного установления данных во время чтения. Различные состояния ожидания устанавливаются программно. Реализована дополнительная функция, которая позволяет разделить внешнюю память на два сектора, и для каждого из них индивидуально выполнить настройку состояний ожидания. Это делает возможным подключить две различные микросхемы памяти с различными требованиями к временной диаграмме доступа через один и тот же интерфейс XMEM.

Обратите внимание, что интерфейс XMEM – асинхронный, и что переключение сигналов на рисунках ниже связано с внутренней синхронизацией. Следовательно, интерфейс XMEM не подходит для синхронной работы.

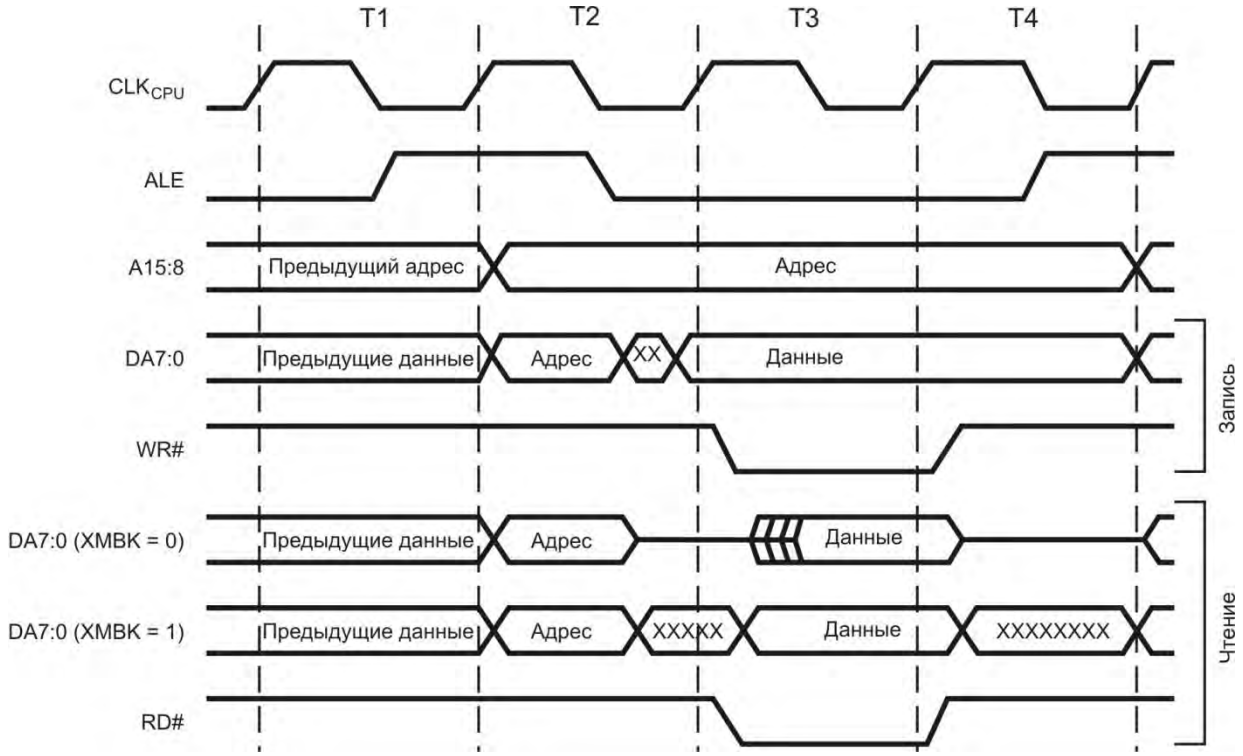


Рисунок 3.14 – Временная диаграмма доступа к внешней памяти
без состояний ожидания (SRWn1=0 и SRWn0=0)

Примечание – SRWn1 равняется SRW11 (верхний сектор) или SRW01 (нижний сектор), SRWn0 равняется SRW10 (верхний сектор) или SRW00 (нижний сектор). Импульс ALE присутствует на такте T4, только если следующая инструкция осуществляет доступ к ОЗУ (внутреннему или внешнему).

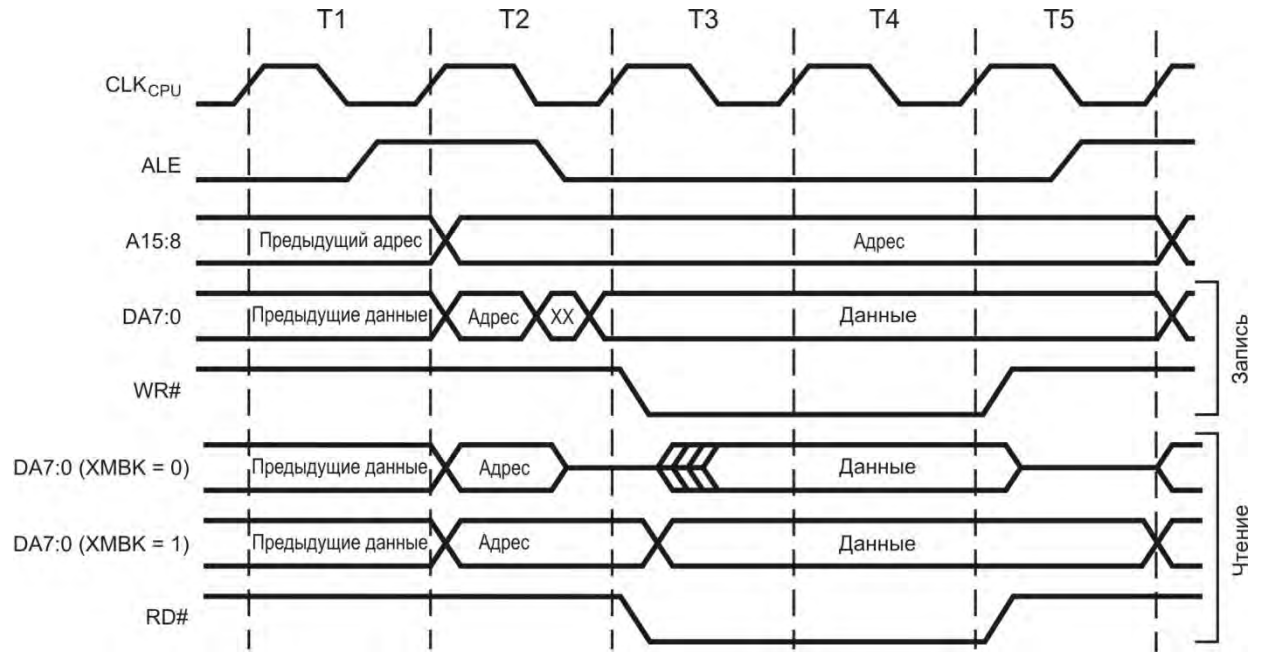


Рисунок 3.15 – Временная диаграмма доступа к внешней памяти при SRWn1=0 и SRWn0=1

Примечание – SRWn1 равняется SRW11 (верхний сектор) или SRW01 (нижний сектор), SRWn0 равняется SRW10 (верхний сектор) или SRW00 (нижний сектор). Импульс ALE присутствует на такте T5, только если следующая инструкция осуществляет доступ к ОЗУ (внутреннему или внешнему).

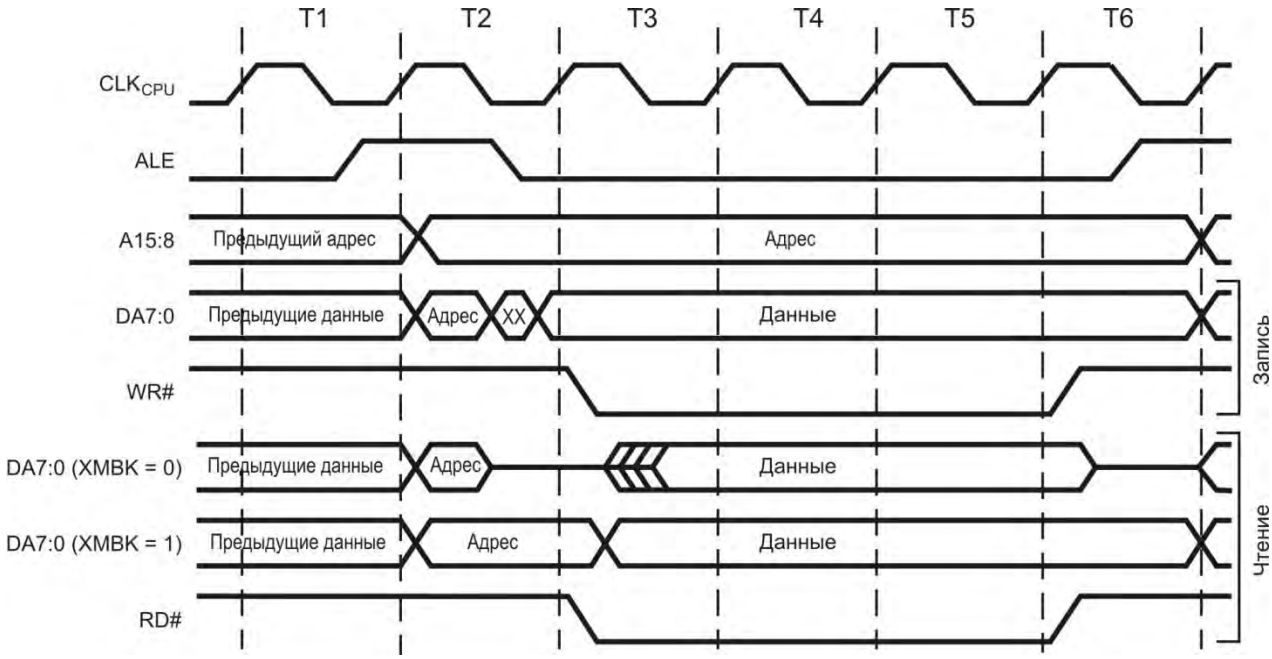


Рисунок 3.16 – Временная диаграмма доступа к внешней памяти при SRWn1=1 и SRWn0=0

Примечание – SRWn1 равняется SRW11 (верхний сектор) или SRW01 (нижний сектор), SRWn0 равняется SRW10 (верхний сектор) или SRW00 (нижний сектор). Импульс ALE присутствует на такте T6, только если следующая инструкция осуществляет доступ к ОЗУ (внутреннему или внешнему).

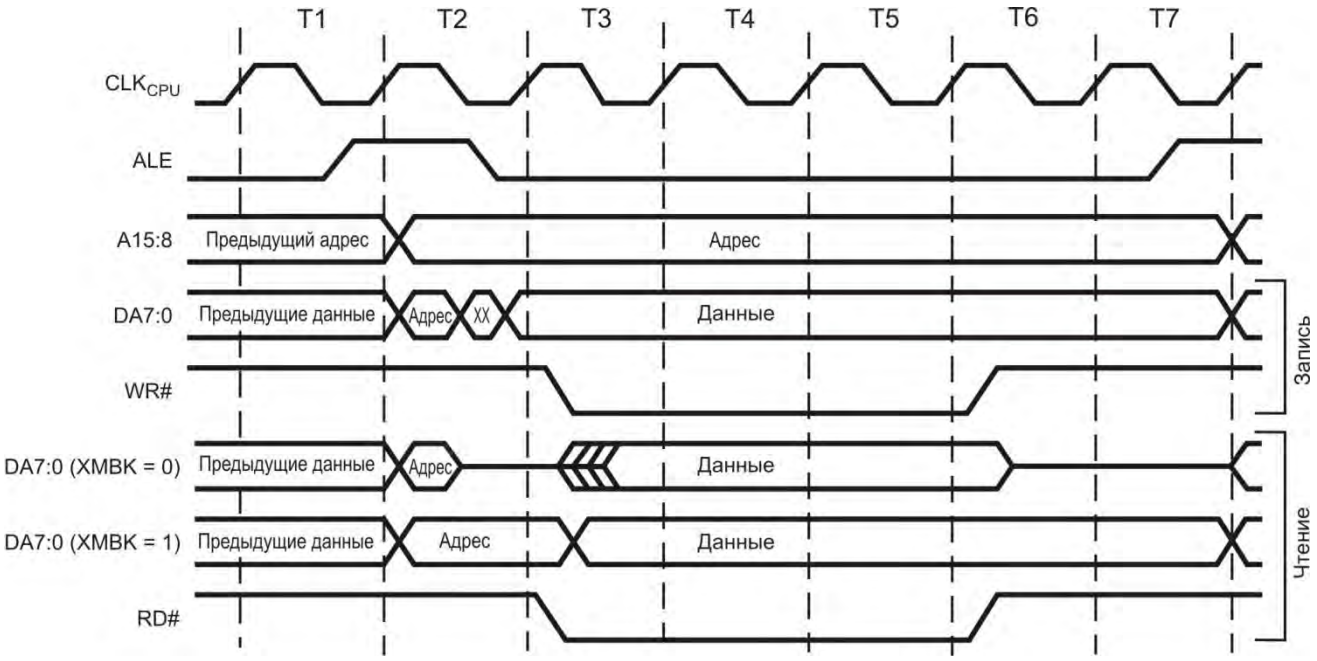


Рисунок 3.17 – Временная диаграмма доступа к внешней памяти при SRWn1=1 и SRWn0=1

Примечание – SRWn1 равняется SRW11 (верхний сектор) или SRW01 (нижний сектор), SRWn0 равняется SRW10 (верхний сектор) или SRW00 (нижний сектор). Импульс ALE присутствует на такте T7, только если следующая инструкция осуществляет доступ к ОЗУ (внутреннему или внешнему).

Регистр управления микроконтроллером – MCUCR

Бит	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	MCUCR
Чтение/Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – SRE: Разрешение внешнего статического ОЗУ/XMEM

Запись в SRE логической единицы разрешает работу интерфейса внешней памяти, после чего выводы AD7:0, A15:8, ALE, WR# и RD# выполняют свои альтернативные функции. После установки бита SRE игнорируются любые установки в соответствующих регистрах направления данных. Запись нуля в SRE отключает интерфейс внешней памяти, после чего вступают в силу обычные функции выводов и установки направления данных.

Разряд 6 – SRW10: Бит выбора состояния ожидания

См. ниже в описании регистра XMCRA.

Регистр А управления внешней памятью – XMCRA

Бит	7	6	5	4	3	2	1	0	
	-	SRL2	SRL1	SRL0	SRW01	SRW00	SRW11	-	XMCRA
Чтение/Запись	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – Зарезервированный бит

Данный бит является зарезервированным и всегда читается как ноль.

Разряды 6–4: SRL2, SRL1, SRL0: Задание границ секторов с состоянием ожидания

Имеется возможность установить различное количество состояний ожидания для различных адресов внешней памяти. Адресное пространство внешней памяти может быть разделено на два сектора, каждый из которых имеет собственные биты выбора состояний ожидания. Биты SRL2, SRL1 и SRL0 определяют разбиение секторов (см. таблицу 3.3 и рисунок 3.12). По умолчанию значение битов SRL2, SRL1 и SRL0 равно нулю и все адресное пространство внешней памяти обслуживается как один сектор. Если все адресное пространство статического ОЗУ конфигурируется как один сектор, то состояния ожидания определяются битами SRW11 и SRW10.

Таблица 3.3 – Границы секторов памяти при различных настройках SRL2–0

SRL2	SRL1	SRL0	Границы сектора
0	0	0	Нижний сектор = Нет Верхний сектор = 0x1100–0xFFFF
0	0	1	Нижний сектор = 0x1100–0x1FFF Верхний сектор = 0x2000–0xFFFF
0	1	0	Нижний сектор = 0x1100–0x3FFF Верхний сектор = 0x4000–0xFFFF
0	1	1	Нижний сектор = 0x1100–0x5FFF Верхний сектор = 0x6000–0xFFFF
1	0	0	Нижний сектор = 0x1100–0x7FFF Верхний сектор = 0x8000–0xFFFF
1	0	1	Нижний сектор = 0x1100–0x9FFF Верхний сектор = 0xA000–0xFFFF
1	1	0	Нижний сектор = 0x1100–0xBFFF Верхний сектор = 0xC000–0xFFFF
1	1	1	Нижний сектор = 0x1100–0xDFFF Верхний сектор = 0xE000–0xFFFF

Разряды 3, 2 – SRW01, SRW00: Биты выбора состояния ожидания для нижнего сектора

Биты SRW01 и SRW00 задают количество состояний ожидания для нижнего сектора адресного пространства внешней памяти (см. таблицу 3.4).

Разряд 1 и разряд 6 регистра MCUCR – SRW11, SRW10: Биты выбора состояния ожидания для верхнего сектора

Биты SRW11 и SRW10 задают количество состояний ожидания для верхнего сектора адресного пространства внешней памяти (см. таблицу 3.4).

Таблица 3.4 – Состояния ожидания

SRWn1	SRWn0	Состояние ожидания
0	0	Нет состояний ожидания
0	1	Задержка на один машинный цикл во время сигнала чтения/записи
1	0	Задержка на два машинных цикла во время сигнала чтения/записи
1	1	Задержка на два машинных цикла во время сигнала чтения/записи и задержка на один машинный цикл перед установкой нового адреса
Примечание – n = 0 или 1 для нижнего или верхнего сектора, соответственно. На рисунках 3.14 - 3.17 представлены временные диаграммы, которые соответствуют различным установкам битов SRW.		

Разряд 0 – зарезервированный бит

Данный бит является зарезервированным и всегда читается как ноль.

Регистр B управления внешней памятью – XMCRB

Бит	7	6	5	4	3	2	1	0	
	XMBK	-	-	-	-	XMM2	XMM1	XMM0	XMCRB
Чтение/Запись	Ч/З	Ч	Ч	Ч	Ч	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – XMBK: Разрешение работы устройства запоминания состояния шины внешней памяти

Запись в XMBK логической единицы разрешает работу устройства запоминания состояния шины на линиях AD7:0. После его активизации AD7:0 будут запоминать последнее установленное состояние, даже если интерфейс XMEM перевел линии в третье состояние. Запись в XMBK логического нуля означает запрет работы устройства запоминания состояния шины. XMBK не зависит от SRE, так что даже если интерфейс XMEM отключен, устройство запоминания состояния шины будет активным, пока XMBK=1.

Разряды 6–3 зарезервированные разряды

Данные разряды являются зарезервированными и всегда читаются как ноль.

Разряды 2–0: XMM2, XMM1, XMM0: Маска старших адресных разрядов внешней памяти

После разрешения внешней памяти все выводы порта C по умолчанию используются в качестве старшего адресного байта. Если нет необходимости адресоваться ко всему 60-Кбайтному пространству внешней памяти, то свободные адресные линии порта C можно использовать в качестве универсального ввода-вывода (см. таблицу 3.5). Использование битов XMMn позволяет адресоваться ко всему пространству 64 Кбайт ячеек внешней памяти (см. ниже подпункт «Использование всего пространства 64 Кбайт внешней памяти»).

Таблица 3.5 – Использование старших адресных сигналов порта С в качестве линий универсального ввода-вывода после разрешения внешней памяти

XMM2	XMM1	XMM0	Число разрядов адреса внешней памяти	Освобождаемые адресные линии порта С
0	0	0	8 (все пространство 60 Кбайт)	Нет
0	0	1	7	PC7
0	1	0	6	PC7 - PC6
0	1	1	5	PC7 - PC5
1	0	0	4	PC7 - PC4
1	0	1	3	PC7 - PC3
1	1	0	2	PC7 - PC2
1	1	1	Старшие биты адреса не используются	Все линии

Использование всех ячеек внешней памяти размером менее 64 Кбайт

Поскольку в соответствии с рисунком 3.12 адресное пространство внешней памяти следует за адресным пространством внутренней памяти, к младшим 4352 ячейкам внешней памяти невозможно адресоваться (адреса 0x0000…0x10FF). Однако, при подключении внешней памяти размером менее 64 Кбайт, например 32 Кбайт, к этим ячейкам можно легко обратиться по адресам 0x8000…0x90FF. Поскольку адресный бит внешней памяти A15 не подключен к внешней памяти, то адреса 0x8000…0x90FF будут выступать в качестве адресов 0x0000…0x10FF для внешней памяти. Адресация по адресам свыше 0x90FF не рекомендуется, так как может затронуть ячейку внешней памяти, доступ к которой уже осуществлялся по другому (меньшему) адресу. Для прикладной программы 32 Кбайта внешней памяти будут представлять линейное адресное пространство с адресами 0x1100…0x90FF (см. рисунок 3.18).

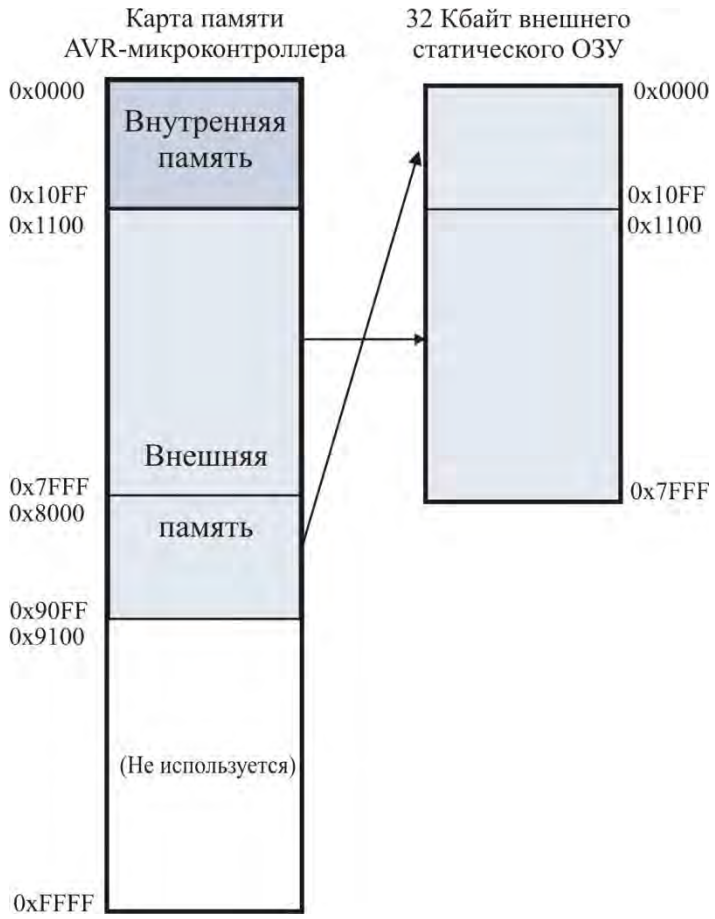


Рисунок 3.18 – Подключение 32 Кбайт внешней памяти

Использование всего пространства 64 Кбайт внешней памяти

Поскольку внешняя память располагается после внутренней памяти, как показано на рисунке 3.12, то только 60 Кбайт внешней памяти доступно по умолчанию (адресное пространство от 0x0000 до 0x10FF зарезервировано для внутренней памяти). Однако имеется возможность использовать весь объем внешней памяти путем маскирования старших адресных разрядов в ноль. Это может быть выполнено с помощью битов XMMn и программного управления старшими адресными разрядами. Интерфейс памяти будет иметь диапазон адресов 0x0000…0x1FFF, если установить на выходе порта C значение 0x00 и выбрать работу старших адресных разрядов как обычных линий ввода-вывода. См. нижеприведенные примеры.

Пример кода на Ассемблере:

```
; Определено смещение OFFSET=0x2000 для гарантирования доступа к внешней
; памяти. Конфигурируем порт C (старший байт адреса) на вывод 0x00, после
; чего настраиваем выводы на выполнение функций обычного порта
ldi r16, 0xFF
out DDRC, r16
ldi r16, 0x00
out PORTC, r16
; Освобождаем PC7:5 от адресных функций
ldi r16, (1<<XMM1)|(1<<XMM0)
sts XMCRB, r16
; Запись  0xAA по адресу 0x0001 внешней памяти
ldi r16, 0xaa
sts 0x0001+OFFSET, r16
; Снова разрешаем работу  PC7:5  как адресных линий
ldi r16, (0<<XMM1)|(0<<XMM0)
sts XMCRB, r16
; Запись 0x55 по адресу (OFFSET + 1) внешней памяти
ldi r16, 0x55
sts 0x0001+OFFSET, r16
```

Пример кода на Си:

```
#define OFFSET 0x2000
void XRAM_example(void)
{
unsigned char *p = (unsigned char *) (OFFSET + 1);
DDRC = 0xFF;
PORTC = 0x00;
XMCRB = (1<<XMM1) | (1<<XMM0);
*p = 0xaa;
XMCRB = 0x00;
*p = 0x55;
}
```

3.3 Системная синхронизация

3.3.1 Системы синхронизации микроконтроллера

На рисунке 3.19 представлены основные системы синхронизации. 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, как описано в подразделе 3.4 «Управление энергопотреблением и спящие режимы». Подробности в отношении систем синхронизации приведены ниже.

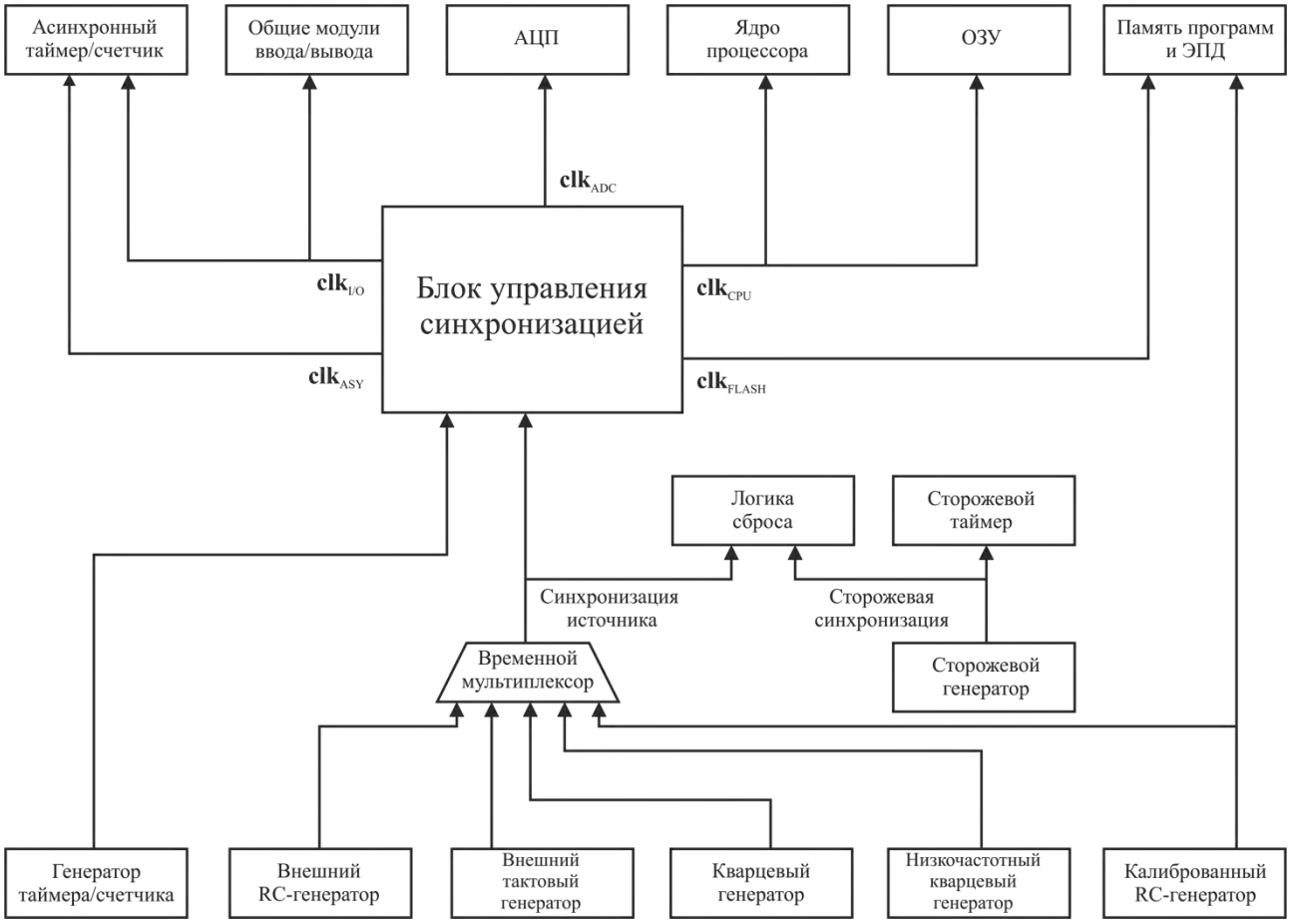


Рисунок 3.19 – Распределение тактовых сигналов

Тактовый сигнал ЦПУ– clk_{CPU}

Тактовый сигнал ЦПУ подсоединен к узлам системы, связанным с работой ядра. Примерами таких модулей являются файл регистров общего назначения, регистр состояний и т. д. Остановка тактового генератора ЦПУ блокирует выполнение ядром операций общего назначения и вычислений.

Тактовый сигнал ввода-вывода – $clk_{I/O}$

Тактовый сигнал ввода-вывода используется большинством периферийных модулей, такими как таймеры/счетчики, SPI и USART. Тактовый сигнал ввода-вывода используется также модулем внешнего прерывания, при этом необходимо обратить внимание на то, что некоторые внешние прерывания детектируются асинхронной логикой, позволяя обнаруживать такие прерывания даже в случае остановки тактового сигнала ввода-вывода. Необходимо также обратить внимание на то, что распознавание адреса в модуле TWI выполняется асинхронно при остановке $clk_{I/O}$, что позволяет обеспечить прием адреса TWI во всех спящих режимах.

Тактовый сигнал памяти программ – clk_{FLASH}

Тактовый сигнал памяти программ clk_{FLASH} управляет работой интерфейса памяти программ. Обычно он активируется одновременно с тактовым генератором ЦПУ.

Тактовый сигнал асинхронного таймера -clk_{ASY}

Тактовый сигнал асинхронного таймера clk_{ASY} обеспечивает непосредственную синхронизацию асинхронного таймера/счетчика с помощью внешнего кварцевого тактового генератора 32 кГц. Специально выделенный канал тактирования 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 канал сигнала тактирования, что позволяет выполнять остановку тактового сигнала ЦПУ и тактового сигнала ввода-вывода для снижения шума, вызываемого цифровыми схемами. Это дает более точные результаты аналого-цифровых преобразований.

Предделитель тактового сигнала BQ

Предделитель тактового сигнала BQ позволяет программно уменьшить частоту сигнала, поступающего от тактового генератора, посредством деления на число в диапазоне от 2 до 129. Эту функцию можно применять для снижения потребления мощности, когда требования к производительности невысоки. Для управления предделителем тактового сигнала используется регистр XDIV.

Бит	7	6	5	4	3	2	1	0	
	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0	XDIV
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – XDIVEN: Разрешение деления BQ

Когда бит XDIVEN равен единице, частота синхронизации ЦПУ и всех периферийных устройств ($clk_{I/O}$, clk_{ADC}, clk_{CPU}, clk_{FLASH}) делится на коэффициент, определяемый битами XDIV6...XDIV0. Бит XDIVEN можно установить во время выполнения программы, чтобы изменить частоту синхронизации на частоту, подходящую для данного приложения.

Разряды 6–0: XDIV6 – XDIV0: Биты выбора деления BQ

Эти разряды определяют коэффициент деления, который используется, когда бит XDIVEN установлен в единицу. Если значение этих разрядов обозначить d, зависимость тактовой частоты f_{CLK} от состояния этих разрядов будет определяться выражением:

$$f_{CLK} = \frac{Частота\ источника}{129 - d}$$

Значение этих битов может быть изменено, только когда XDIVEN = 0. Когда XDIVEN=1, значение, определяемое совокупностью битов XDIV6...XDIV0, рассматривается как коэффициент деления. Когда XDIVEN = 0, это значение игнорируется.

Когда выполняется деление системного синхросигнала, таймер/счетчик 0 можно использовать только с асинхронным тактированием. Частота асинхронного тактирования должна быть ниже четверти частоты синхронизации микроконтроллера. В противном

случае можно пропустить прерывания и тогда значения регистров, имеющих доступ к таймеру/счетчику 0, могут быть нарушены.

3.3.2 Источники синхронизации

Микроконтроллер имеет несколько источников тактового сигнала, выбираемых с помощью конфигурационных битов согласно таблице 3.6. Сигнал синхронизации выбранного источника является входным для тактового генератора микроконтроллера и затем подключается к соответствующим модулям.

Таблица 3.6 – Опции выбора синхронизирующего устройства

Опция выбора синхронизирующего устройства	CKSEL3...0
Внешний кварцевый/керамический резонатор	1111–1010
Внешний низкочастотный кварцевый генератор	1001
Внешний RC-генератор	1000–0101
Калиброванный внутренний RC-генератор	0100–0001
Внешний тактовый генератор	0000
Примечание – Для всех конфигурационных битов «1» означает незапрограммированное состояние, «0» означает запрограммированное состояние.	

Подробное описание каждой из этих опций приведено в следующих разделах. При выходе ЦПУ из режима микропотребления или режима хранения выбранный источник синхронизации используется по истечении времени запуска, гарантируя тем самым стабильность работы генератора перед выполнением первой инструкции. Запуск микроконтроллера после сброса сопровождается дополнительной задержкой для достижения питанием стабильного уровня перед переводом микроконтроллера в нормальный режим работы. Генератор сторожевого таймера используется для синхронизации данного модуля, который формирует задержку при запуске. Длительность генерируемой задержки определяется количеством импульсов генератора сторожевого таймера и для различных случаев приведена в таблице 3.7.

Таблица 3.7 – Количество тактов сторожевого генератора

Типовое время простоя, мс (#VCC = 5,0 В)	Количество тактов
8	4K (4096)
130	64K (65536)

3.3.3 Источник тактовых импульсов по умолчанию

Микроконтроллер поставляется с установками CKSEL = «0001» и SUT = «10». Эти значения соответствуют выбору в качестве источника синхронизации внутреннего RC-генератора с максимальным временем старта. Данная настройка гарантирует всем пользователям возможность установить требуемый источник синхронизации с помощью внутрисистемного или параллельного программатора.

3.3.4 Кварцевый генератор

BQ1 и BQ2 – соответственно вход и выход инвертирующего усилителя, который может быть настроен для использования в качестве встроенного генератора (см. рисунок 3.20). Для задания частоты может использоваться либо кварцевый, либо керамический резонатор. Конфигурационный бит CKOPT выбирает один из двух режимов усилителя гене-

ратора. Если CKOPT запрограммирован, то амплитуда колебаний выходного сигнала генератора будет ограничена уровнями питания. Данный режим рекомендуется использовать при высоком уровне окружающих шумов или при использовании выхода BQ2 в качестве источника синхронизации внешней схемы. Данный режим характеризуется широким частотным диапазоном. Если CKOPT незапрограммирован, то амплитуда выходных колебаний генератора снижается. Использование данного режима позволяет существенно снизить потребляемую мощность, но при этом частотный диапазон ограничен, и нельзя использовать BQ2 для внешней синхронизации.

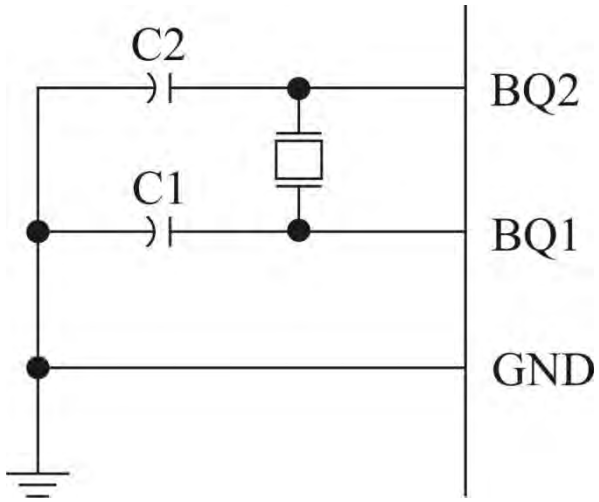


Рисунок 3.20 – Подключение кварцевого генератора

При использовании резонаторов максимальная частота равна 8 МГц, если CKOPT не запрограммирован, и 16 МГц, если CKOPT запрограммирован. Конденсаторы C1 и C2 всегда должны быть одинаковыми, независимо от использования кварцевого или керамического резонатора. Оптимальное значение емкостей конденсаторов зависит от используемого кварцевого или керамического резонатора, от значения паразитной емкости и от окружающего уровня электромагнитного шума. Рекомендации по выбору номиналов конденсаторов приведены в таблице 3.8. Для керамических резонаторов необходимо использовать конденсаторы с номиналом, рекомендуемым производителем.

Генератор может работать в трех различных режимах, каждый из которых оптимизирован для определенного частотного диапазона. Режим работы выбирается с помощью конфигурационных битов CKSEL3–1, как показано в таблице 3.8.

Таблица 3.8 – Режимы работы кварцевого генератора

CKOPT	CKSEL3–1	Частотный диапазон, МГц	Рекомендуемые номиналы конденсаторов C1 и C2 для использования с кварцевыми резонаторами, пФ
1	101*	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 – max	12–22

* Этот вариант не следует использовать с кварцевыми резонаторами, только с керамическими.

Конфигурационный бит CKSEL0 совместно с битами SUT1–0 выбирает время запуска в соответствии с таблицей 3.9.

Таблица 3.9 – Временная задержка при запуске для различных настроек кварцевого генератора

CKSEL0	SUT1–0	Длительность задержки при выходе из режима микропотребления и режима хранения	Дополнительная задержка после сброса, мс (#VCC = 5,0 В)	Рекомендуемое применение
0	00	258 CK[1)]	8	Керамический резонатор, быстро нарастающее питание
0	01	258 CK[1)]	130	Керамический резонатор, медленно нарастающее питание
0	10	1K CK[2)]	-	Керамический резонатор, детектор питания (BOD) включен
0	11	1K CK[2)]	8	Керамический резонатор, быстро нарастающее питание
1	00	1K CK[2)]	130	Керамический резонатор, медленно нарастающее питание
1	01	16K CK	-	Кварцевый генератор, детектор питания (BOD) включен
1	10	16K CK	8	Кварцевый генератор, быстро нарастающее питание
1	11	16K CK	130	Кварцевый генератор, медленно нарастающее питание

[1)] Этот вариант должен 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. Вариант не пригоден для использования с кварцевыми резонаторами.

[2)] Этот вариант предназначен для использования с керамическими резонаторами и обеспечивает стабильность частоты при запуске. Также можно использовать с кварцевыми резонаторами, если не применять при работе на частоте, близкой к предельной, и если стабильность по частоте при запуске не важна для данного вида применения.

3.3.5 Низкочастотный кварцевый генератор

Для использования часового кварцевого резонатора 32,768 кГц в качестве источника синхронизации необходимо выбрать низкочастотный кварцевый генератор путем установки конфигурационных битов CKSEL равными «1001». Подключение кварцевого резонатора показано на рисунке 3.20. Путем программирования конфигурационного бита пользователь может разрешить подключение встроенных конденсаторов к выводам BQ1 и BQ2, тем самым исключая необходимость применения внешних конденсаторов. Внутренние конденсаторы имеют номинал 36 пФ. После выбора данного генератора, длительности задержек при старте определяются конфигурационными битами SUT как показано в таблице 3.10.

Таблица 3.10 – Длительности задержек при старте для низкочастотного кварцевого генератора

SUT1–0	Длительность задержки при выходе из режима микропотребления и режима хранения	Дополнительная задержка после сброса, мс (#VCC = 5,0 В)	Рекомендуемое применение
00	1K CK[1)]	8	Быстро нарастающее питание или включен детектор питания BOD
01	1K CK[1)]	130	Медленно нарастающее питание
10	32K CK	130	Стабильная частота при запуске
11	Зарезервировано		

[1)] Эти варианты следует применять тогда, когда стабильность частоты при старте не важна для приложения.

3.3.6 Внешний RC-генератор

Для приложений, некритичных к стабильности временных характеристик, в качестве источника синхронизации может использоваться внешняя RC-цепь, подключение которой показано на рисунке 3.21. Тактовая частота грубо определяется выражением f = 1/(3RC). Номинал конденсатора C должен быть не менее 22 пФ. Путем программирования конфигурационного бита CKOPT пользователь может разрешить подключение внутреннего конденсатора 36 пФ между BQ1 и #0V (GND), тем самым исключая необходимость применения внешнего конденсатора.

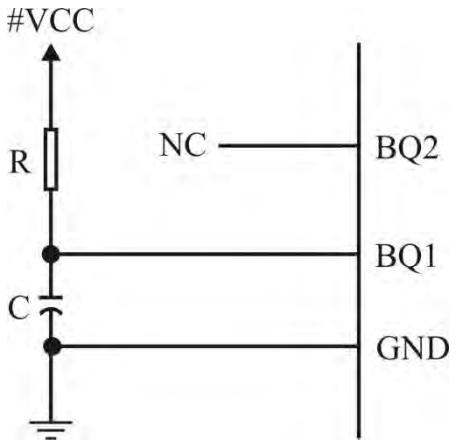


Рисунок 3.21 – Схема подключения внешней задающей RC-цепи

Генератор может работать в четырех различных режимах, каждый из которых ориентирован на определенный частотный диапазон. Рабочий режим выбирается конфигурационными битами CKSEL3–0 (см. таблицу 3.11).

Таблица 3.11 – Режимы работы внешнего RC-генератора

CKSEL3–0	Частотный диапазон, МГц
0101	0,1–0,9
0110	0,9–3,0
0111	3,0–8,0
1000	8,0–12,0

После разрешения работы данного генератора длительность задержки при старте определяется установками конфигурационных битов (см. таблицу 3.12).

Таблица 3.12 – Длительность задержек при старте после выбора внешнего RC-генератора

SUT1–0	Длительность задержки при выходе из режима микропотребления и режима хранения	Дополнительная задержка после сброса, мс (#VCC = 5,0 В)	Рекомендуемое применение
00	18CK	-	Включенный детектор питания BOD
01	18CK	8	Быстро нарастающее питание
10	18CK	130	Медленно нарастающее питание
11	6CK*	8	Быстро нарастающее питание или включенный детектор питания BOD

* Этот вариант не должен использоваться при работе на частоте, близкой к предельной.

3.3.7 Внутренний калиброванный RC-генератор

Внутренний калиброванный RC-генератор формирует фиксированные тактовые частоты 1, 2, 4 или 8 МГц. Данные значения частот являются номинальными и определены для напряжения питания 5 В при температуре 25 °С. Одна из этих частот может быть выбрана в качестве тактовой, если запрограммировать конфигурационные биты CKSEL в соответствии с таблицей 3.13. После выбора микроконтроллер будет работать без внешних компонентов. Конфигурационный бит CKOPT должен быть всегда незапрограммированным, если используется внутренний RC-генератор. В процессе сброса калибровочный байт аппаратно записывается в регистр OSCCAL, тем самым автоматически выполняется калибровка RC-генератора. При питании 5 В, температуре 25 °С и выбранной частоте генератора 1 МГц данный метод калибровки обеспечивает погрешность генерации частоты в пределах ±6 % от номинального значения. Использование методов калибровки во время работы микроконтроллера позволяет достичь точности ±1 % при любой заданной температуре и напряжении питания. Когда этот генератор применяется для синхронизации микросхемы, для сторожевого таймера и задержки сброса будет использоваться генератор сторожевого таймера. Более подробная информация о предварительно запрограммированном калибровочном значении приведена в 3.21.4 «Калибровочный байт».

Таблица 3.13 – Рабочие режимы внутреннего калиброванного RC-генератора

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

_____
* Микроконтроллер поставляется с данной установкой.

После выбора данного генератора длительность задержки при запуске микроконтроллера определяется установками конфигурационных битов SUT (см. таблицу 3.14). Выводы BQ1 и BQ2 должны быть оставлены неподключенными.

Таблица 3.14 – Длительности задержек при запуске с различными настройками внутреннего калиброванного RC-генератора

SUT1–0	Длительность задержки при выходе из режима микропотребления и режима хранения	Дополнительная задержка после сброса, мс (#VCC = 5,0 В)	Рекомендуемое применение
00	6 CK	-	Включенный детектор питания BOD
01	6 CK	8	Быстро нарастающее питание
10*	6 CK	130	Медленно нарастающее питание
11	Зарезервировано		

_____
* Микроконтроллер поставляется с данной установкой.

Регистр калибровки генератора – OSCCAL

Бит	7	6	5	4	3	2	1	0	
	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	OSCCAL
Чтение/Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	Калибровочное значение индивидуально для каждого микроконтроллера								

Разряды 7–0: CAL7–0: Калибровочное значение для генератора

Запись значения калибровочного байта в данный регистр приведет к подстройке генератора на номинальную частоту. В процессе сброса калибровочное значение для выбранной частоты автоматически записывается в регистр OSCCAL. Если необходима более точная подстройка частоты генератора в регистр OSCCAL можно программно загрузить колибровочный коэффициент, который необходимо предварительно сохранить в памяти программ или ЭПД, а после старта считать и записать в регистр OSCCAL. Если в регистр OSCCAL записать ноль, то выбирается минимальная частота. Запись ненулевого значения приводит к повышению частоты генератора. Запись $FF дает максимально доступную частоту. Калиброванный генератор используется для синхронизации доступа к ЭПД памяти программ. Во время выполнения записи в ЭПД или в память программ не следует выполнять установку частоты более, чем на 10 % отличающуюся от номинальной частоты. В противном случае запись в ЭПД или в память программ может быть некорректной. Обратите внимание, что генератор предназначен для калибровки на частоты 1, 2, 4 или 8 МГц. Настройка на другие значения не обеспечена (см. таблицу 3.15).

Таблица 3.15 – Диапазон частот внутреннего RC-генератора

Значение OSCCAL	Минимальная частота в процентах от номинальной частоты	Максимальная частота в процентах от номинальной частоты
$00	50	100
$7F	75	150
$FF	100	200

3.3.8 Внешний тактовый генератор

Если необходимо тактировать микроконтроллер от внешнего источника, то его необходимо подключить к выводу BQ1 (см. рисунок 3.22). В этом случае внешняя синхронизация должна быть разрешена записью в конфигурационные биты CKSEL значения «0000». Если запрограммировать конфигурационный бит CKOPT, то между BQ1 и #0V (GND) будет подключен внутренний конденсатор номиналом 36 пФ.

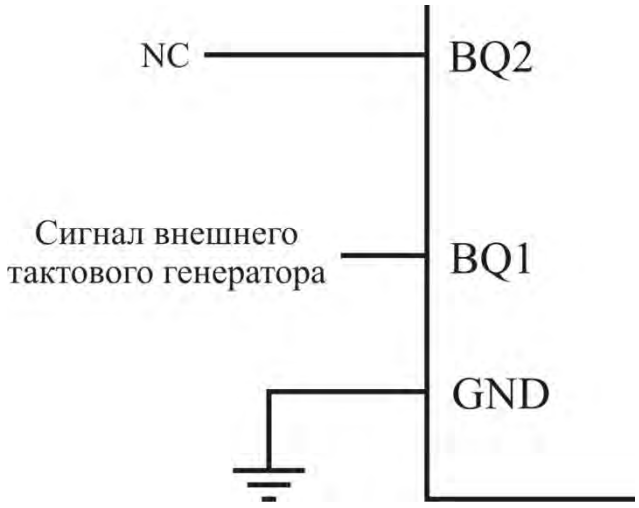


Рисунок 3.22 – Схема подключения внешнего тактового генератора

После выбора данного источника синхронизации длительность задержки при запуске определяется конфигурационными битами SUT, как показано в таблице 3.16.

Таблица 3.16 – Длительность задержки при запуске после выбора внешнего тактового генератора

SUT1–0	Длительность задержки при выходе из режима микропотребления и режима хранения	Дополнительная задержка после сброса, мс (#VCC = 5,0 В)	Рекомендуемое применение
00	6 CK	-	Включен детектор питания BOD
01	6 CK	8	Быстро нарастающее питание
10	6 CK	130	Медленно нарастающее питание
11	Зарезервировано		

После подключения внешнего тактового источника необходимо избегать внезапных изменений его частоты для гарантирования стабильности работы микроконтроллера. Если на следующем такте частота изменится более чем на 2 % по сравнению с предыдущим, то поведение микроконтроллера может стать непредсказуемым. Необходимо гарантировать нахождение микроконтроллера в состоянии сброса в процессе таких изменений частоты синхронизации.

3.3.9 Генератор таймера/счетчика

В состав микроконтроллера входит отдельный генератор таймера/счетчика. Кварцевый резонатор подключается непосредственно между выводами TOSC1 и TOSC2. В этом случае не требуются внешние конденсаторы. Генератор оптимизирован для совместной работы с часовым кварцевым резонатором частотой 32,768 кГц. **Подключение внешнего источника синхронизации к выводу TOSC1 не рекомендуется.**

В генераторе таймера/счетчика используется тот же тип кварцевого резонатора, что и в низкочастотном генераторе, и внутренние конденсаторы имеют такое же номинальное значение 36 пФ.

3.4 Управление энергопотреблением и спящие режимы

Использование спящих режимов позволяет отключать неиспользуемые модули микроконтроллера, уменьшая тем самым потребляемую мощность. Микроконтроллер поддерживает несколько режимов сна, позволяющих программисту оптимизировать энергопотребление под требования приложения.

Для перевода микроконтроллера в один из шести спящих режимов необходимо предварительно установить бит SE в регистре MCUCR, а затем выполнить инструкцию SLEEP. Биты SM2, SM1 и SM0 регистра MCUCR задают, в какой именно режим будет переведен микроконтроллер (холостой ход «Idle», снижение шумов АЦП «ADC Noise Reduction», хранение «Power-down», микропотребление «Power-save», ожидание «Standby» или расширенный режим ожидания «Extended Standby») после выполнения команды SLEEP (см. таблицу 3.17). Выход из режима сна происходит при возникновении разрешенного прерывания. В этом случае помимо времени старта микроконтроллер приостанавливается на четыре машинных цикла, выполняет процедуру обработки прерывания и продолжает выполнять команды, следующие за SLEEP. Содержимое регистрового файла и статического ОЗУ остается неизменным после выхода из спящего режима. Если во время действия режима сна возникает условие сброса, то микроконтроллер пробуждается и исполняет код программы по вектору сброса.

На рисунке 3.19 представлены различные системы синхронизации микроконтроллера 1887ВЕ7Т и их распределение. Данный рисунок может быть полезным при выборе соответствующего спящего режима.

3.4.1 Регистр управления микроконтроллером – MCUCR

Регистр управления микроконтроллером содержит биты управления энергопотреблением.

Бит	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	MCUCR
Чтение/запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 5 – SE: Разрешение перевода в режим сна

В бит SE должна быть записана логическая единица, когда необходимо перевести микроконтроллер в спящий режим командой SLEEP. Во избежание незапланированного программистом перевода микроконтроллера в режим сна рекомендуется устанавливать этот бит непосредственно перед выполнением инструкции SLEEP и сбрасывать сразу после пробуждения.

Разряды 4–2: SM2–0: Биты выбора режима сна

С помощью данных битов можно выбрать один из шести режимов сна в соответствии с таблицей 3.17.

Таблица 3.17 – Выбор режима сна

SM2	SM1	SM0	Наименование режима сна
0	0	0	Холостой ход
0	0	1	Снижение шумов АЦП
0	1	0	Хранение
0	1	1	Микропотребление
1	0	0	Зарезервировано
1	0	1	Зарезервировано
1	1	0	Ожидание*
1	1	1	Расширенный режим ожидания*

* Режим ожидания и расширенный режим ожидания доступны только при использовании внешних кварцевых генераторов или резонаторов.

3.4.2 Режим холостого хода (Idle)

Если значение битов SM2...0 равно «000», то после выполнения инструкции SLEEP микроконтроллер переходит в режим холостого хода, в котором останавливается ЦПУ, но продолжают работу SPI, USART, аналоговый компаратор, АЦП, двухпроводной интерфейс TWI, таймеры/счетчики, сторожевой таймер и система прерываний. По сути, в данном режиме останавливается синхронизация ядра ЦПУ и памяти программ (clk_{CPU} и clk_{FLASH}), а остальные системы синхронизации продолжают работу.

В режиме холостого хода допускается пробуждение от любого внешнего или внутреннего прерывания, например, при переполнении таймера или завершении передачи USART. Если пробуждение по прерыванию аналогового компаратора не требуется, то аналоговый компаратор может быть отключен путем установки бита ACD в регистре управления и состояния аналогового компаратора ACSR. Это позволит уменьшить потребляемую мощность в режиме холостого хода. Если разрешена работа АЦП, то преобразование автоматически запускается после перевода в данный режим.

3.4.3 Режим снижения шумов АЦП (ADC Noise Reduction)

Если битам SM2–0 присвоить значение «001», то выполнение инструкции SLEEP приведет к переводу микроконтроллера в режим снижения шумов АЦП, в котором останавливается ЦПУ, но продолжают работу АЦП, внешние прерывания, наблюдение за адресом двухпроводного последовательного интерфейса TWI, таймер/счетчик 0 и сторожевой таймер (в случае, если были предварительно активизированы). Фактически в данном режиме прекращается синхронизация ввода-вывода ($clk_{I/O}$), ядра ЦПУ (clk_{CPU}) и памяти программ (clk_{FLASH}), а остальная синхронизация продолжает работу.

В этом режиме создаются более благоприятные условия для аналого-цифрового преобразования с повышенной разрешающей способностью за счет снижения влияния шумов на результат измерения. Если разрешена работа АЦП, то преобразование автоматически запускается при переводе в данный режим. Выход из данного режима допускается не только при генерации запроса на прерывание по завершении преобразования АЦП, но также и при внешнем сбросе, сбросе по сторожевому таймеру, сбросе при недопустимом снижении питания, прерывании при обнаружении установленного адреса на двухпроводной последовательной шине, прерывании по таймеру/счетчику 0, прерывании по готовности SPM/EEPROM, прерывании по внешнему уровню на выводах INT7–4 или внешнем прерывании по входам INT3–0.

3.4.4 Режим хранения (Power-down)

Если биты SM2–0 установить в значение «010», то выполнение команды SLEEP означает перевод микроконтроллера в режим хранения. В данном режиме прекращает работу внешний генератор, но остаются активными внешние прерывания, наблюдение за адресом на двухпроводной последовательной шине и сторожевой таймер (при условии, что они активизированы). Выход из данного режима возможен только по внешнему сбросу, сбросу сторожевым таймером, сбросу при недопустимом снижении питания, прерывании при обнаружении установленного адреса на двухпроводной последовательной шине, прерывании по внешнему уровню на выводах INT7–4 или при внешнем прерывании INT3–0. Данный спящий режим фактически останавливает все системы синхронизации, позволяя работать только асинхронным модулям.

Обратите внимание, что если для выхода из прерывания используется прерывание по установке заданного уровня на внешнем входе, то выход из режима сна возможен, только если этот уровень присутствует в течение определенного времени (см. также подраздел 3.8 «Внешние прерывания»).

Выход из режима хранения сопровождается задержкой с момента выполнения условия прерывания до эффективного пробуждения. Данная задержка позволяет перезапустить синхронизацию и дождаться стабильности ее работы. Период пробуждения определяется теми же конфигурационными битами CKSEL, которые определяют период задержки при сбросе (см. 3.3.2 «Источники синхронизации»).

3.4.5 Режим микропотребления (Power-save)

Если установить значения битов SM2–0 равным «011», то действие команды SLEEP приведет к переводу микроконтроллера в режим микропотребления. Данный режим идентичен режиму хранения за одним исключением.

Если таймер/счетчик 0 тактируется асинхронно, то есть установлен бит AS0 в регистре ASSR, то таймер/счетчик 0 в режиме сна продолжит работу. Выход из режима сна возможен либо при переполнении таймера, либо при выполнении условия сравнения, если соответствующее прерывание для таймера/счетчика 0 разрешено в регистре TIMSK, а также установлен бит общего разрешения прерываний в регистре SREG.

Если для асинхронного таймера не включено асинхронное тактирование, то рекомендуется использовать режим хранения вместо режима микропотребления, так как содержимое регистров асинхронного таймера должно рассматриваться как неопределенное после выхода из режима хранения, в котором значение AS0 было равно 0.

Данный спящий режим фактически останавливает всю синхронизацию за исключением асинхронной (clk_{ASY}), позволяя работать только асинхронным модулям, в том числе таймеру/счетчику 0 с разрешенной опцией асинхронного тактирования.

3.4.6 Режим ожидания (Standby)

Если значениям битов SM2–0 присвоить «110» и выбрать опции тактирования от внешнего кварцевого генератора/резонатора, то действие команды SLEEP приведет к переводу микроконтроллера в режим ожидания. Данный режим идентичен режиму хранения за исключением того, что генератор продолжает свою работу. Из режима ожидания микроконтроллер выходит за шесть машинных циклов.

3.4.7 Расширенный режим ожидания (Extended Standby)

Запись в SM2–0 значения «111» с учетом выбора в качестве тактового источника внешнего кварцевого генератора/резонатора означает, что после выполнения команды SLEEP микроконтроллер будет переведен в расширенный режим ожидания. Данный режим идентичен режиму микропотребления за исключением того, что генератор продолжает свою работу. Выход из расширенного режима ожидания происходит за шесть машинных циклов.

Таблица 3.18 – Активные тактируемые модули и источники пробуждения в различных режимах сна

Наименование режима сна	Активные области тактирования					Активные генераторы		Источники пробуждения					
	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Источник основной синхронизации	Генератор таймера	INT7:0	Соответствие адреса TWI	Таймер 0	Готовность SPM/EEPROM	АЦП	Другие выводы/выводы
Холостой ход			x	x	x	x	x[2)]	x	x	x	x	x	x
Снижение шумов				x	x	x	x[2)]	x[3)]	x	x	x	x	
Хранение								x[3)]	x				
Микропотребление					x[2)]		x[2)]	x[3)]	x	x[2)]			
Ожидание [1)]						x		x[3)]	x				
Расширенный режим ожидания [1)]					x[2)]	x	x[2)]	x[3)]	x	x[2)]			

[1)] В качестве источника синхронизации выбран внешний кварцевый генератор или резонатор.
[2)] Если установлен бит AS0 в регистре ASSR.
[3)] Только INT3–0 или прерывание по уровню на INT7–4.

3.4.8 Минимизация энергопотребления

В процессе оптимизации энергопотребления могут возникнуть некоторые проблемы. В общем случае следует по возможности максимально использовать спящие режимы, а сам спящий режим нужно выбирать исходя из поддержки только необходимых функций. Все неиспользуемые функции должны быть отключены. В частности, для следующих модулей в целях достижения минимально возможного энергопотребления должны быть учтены некоторые рекомендации.

Аналого-цифровой преобразователь

Если АЦП был активирован, то он останется активным и во всех спящих режимах. Для снижения мощности рекомендуется отключать АЦП перед входом в режим сна. Если АЦП был отключен, а затем снова включен, то следующее преобразование будет расширенным (см. подраздел 3.18 «Аналого-цифровой преобразователь»).

Аналоговый компаратор

Перед входом в режим холостого хода аналоговый компаратор необходимо выключить, если он не используется. Перед входом в режим снижения шумов АЦП аналоговый компаратор должен быть отключен. При входе в другие спящие режимы аналоговый компаратор отключается автоматически. Однако если к неинвертирующему входу аналогового компаратора выбрано подключение встроенного источника опорного напряжения, то перед входом в любой режим сна аналоговый компаратор необходимо отключать. В противном случае встроенный источник опорного напряжения останется включенным независимо от режима сна (см. подраздел 3.17 «Аналоговый компаратор»).

Супервизор питания

Если нет необходимости использовать супервизор питания, то данный модуль следует выключить. Если супервизор питания активирован конфигурационным битом BODEN, то он останется активным во всех режимах сна и, следовательно, будет постоянно потреблять мощность. При организации режимов глубокого сна отключение данного модуля позволит существенно уменьшить потребляемый ток.

Встроенный источник опорного напряжения

Работа встроенного источника опорного напряжения (ИОН) разрешается, если необходимо использовать супервизор питания, аналоговый компаратор или АЦП. Если данные модули отключены, то встроенный ИОН также отключится и не будет потреблять мощность. При возобновлении его работы программист должен учесть задержку на установление выходного напряжения ИОН перед его использованием. Если ИОН остается включенным в режиме сна, то его можно использовать сразу после пробуждения (см. также 3.5.8 «Встроенный источник опорного напряжения» для уточнения времени его запуска).

Сторожевой таймер

Если нет необходимости в использовании сторожевого таймера, то данный модуль должен быть отключен. Если разрешить работу сторожевого таймера, то он останется активным во всех режимах сна и, следовательно, будет постоянно потреблять мощность. При организации режимов глубокого сна отключение данного модуля позволит существенно уменьшить потребляемый ток (см. также 3.5.9 «Сторожевой таймер»).

Линии портов ввода-вывода

Перед переводом в режим сна все линии портов ввода-вывода должны быть настроены с учетом потребления минимальной мощности. Основное внимание следует уделить отсутствию резистивных нагрузок на выводах. В режимах сна, где отключена синхронизация ввода-вывода ($clk_{I/O}$) и АЦП (clk_{ADC}), входные буферы микроконтроллера отключены. Этим гарантируется отсутствие энергопотребления неиспользуемой в режиме сна входной логикой. В некоторых случаях входная логика необходима для определения условия пробуждения и в этом случае должна быть активной. Если работа входного буфера разрешена, а входной сигнал оказался отключенным или имеет уровень близкий к #VCC/2, то этот входной буфер будет потреблять повышенную мощность.

Интерфейс JTAG и встроенная система отладки

Если работа встроенной системы отладки разрешена конфигурационным битом OCDEN, то даже при переводе микроконтроллера в режим хранения или режим микропотребления источник основной синхронизации продолжит работу. В этом случае микроконтроллер будет потреблять существенный ток даже в этих режимах сна. Избежать этого можно с помощью одного из трех способов:

- сбросить конфигурационный бит OCDEN;
- сбросить конфигурационный бит JTAGEN;
- установить бит JTD в регистре MCUCSR.

После разрешения работы интерфейса JTAG вывод TDO остается плавающим до тех пор, пока TAP-контроллер JTAG не начнет выдавать данные. Если связанная с выводом TDO аппаратная часть не выполняет подтягивание потенциала к питанию, то потребляемая мощность увеличится. Обратите внимание, что вывод TDI следующего микроконтроллера в сканируемой цепи содержит подтягивающий резистор во избежание данной проблемы. Запись логической единицы в бит JTD регистра MCUCSR приводит к отключению интерфейса JTAG так же, как и незапрограммированное состояние конфигурационного бита JTAGEN.

3.5 Управление системой и сброс

3.5.1 Сброс микроконтроллера

В процессе сброса во все регистры ввода-вывода записываются их начальные значения, и выполнение программы начинается с вектора сброса. По вектору сброса должна храниться инструкция абсолютного перехода JMP на метку процедуры обработки сброса. Если в программе не используются источники прерывания, то векторы прерываний не используются, а зарезервированные под них ячейки памяти могут использоваться для равномерного расположения кода программы. Имеется также случай, когда вектор сброса расположен в секции прикладной программы, а векторы прерываний находятся в загрузочном секторе или наоборот. На рисунке 3.23 показана схема организации логики сброса. В таблице 3.19 приведены характеристики схемы сброса.

Порты ввода-вывода микроконтроллера немедленно возвращаются к их первоначальному состоянию, как только один из источников сброса становится активным. Для этого не требуется работа какой-либо синхронизации.

После прекращения действия всех источников сброса вступает в силу счетчик задержки, продлевающий внутренний сброс. Данная последовательность требуется для гарантирования запуска микроконтроллера при достижении напряжением питания стабильного уровня. Длительность задержки при старте определяется конфигурационными битами CKSEL. Различные варианты установок длительностей задержек представлены в 3.3.2 «Источники синхронизации».

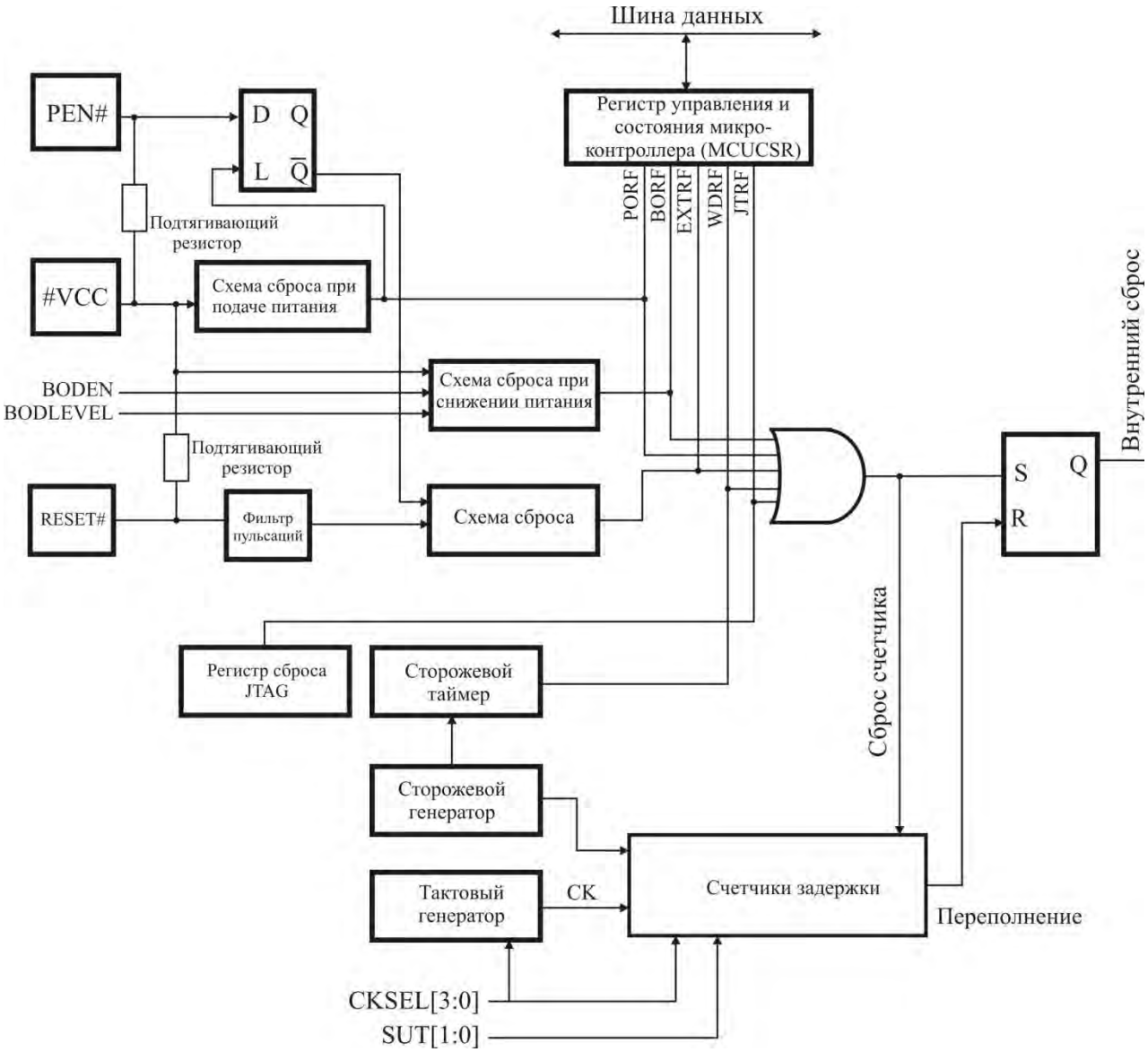


Рисунок 3.23 – Логика сброса

Таблица 3.19 – Характеристики сброса

Обозначение	Параметр	Условие	Минимальное значение	Типовое значение	Максимальное значение	Единица измерения
1	2	3	4	5	6	7
V_{POT}	Пороговое напряжение сброса при подаче питания (нарастающее)			1,4	2,3	В
	Пороговое напряжение сброса при подаче питания (спадающее) [1)]			1,3	2,3	В
V_{RST}	Пороговое напряжение на выводе RESET#		$0,2U_{CC1}$		$0,85U_{CC1}$	В
t_{RST}	Длительность импульса на выводе RESET#		1,5			мкс

Продолжение таблицы 3.19

1	2	3	4	5	6	7
V_{BOT}	Пороговое напряжение сброса при снижении питания	BODLEVEL=1	2,4	2,6	2,9	В
		BODLEVEL=0	3,7	4,0	4,5	
t_{BOD}	Минимальная длительность снижения напряжения для срабатывания схемы контроля питания	BODLEVEL=1		2		мкс
		BODLEVEL=0		2		
V_{HYST}	Ширина петли гистерезиса схемы контроля питания			100		мВ

[1)] Сброс при подаче питания не будет срабатывать, пока напряжение питания не снизится до величины ниже V_{POT} (спадающее).

3.5.2 Источники сброса

ИС 1887ВЕ7Т включает пять источников сброса:

- Сброс при подаче питания. Микроконтроллер переходит в состояние сброса, если напряжение питания ниже порога сброса при подаче питания (V_{POT}).

- Внешний сброс. Микроконтроллер переходит в состояние сброса, если на вывод RESET# подать низкий логический уровень на время, превышающее минимальную длительность импульса сброса.

- Сброс по сторожевому таймеру. Если разрешена работа сторожевого таймера и истек период его срабатывания, то микроконтроллер сбрасывается.

- Сброс при снижении питания. Микроконтроллер сбрасывается, если напряжение питания #VCC становится ниже порогового значения (V_{BOT}) и разрешена работа схемы контроля питания BOD.

- Сброс через интерфейс JTAG. Микроконтроллер находится в состоянии сброса до тех пор, пока записана логическая единица в регистре сброса, одной из сканируемых цепей JTAG-системы. См. пункт 3.19.10 «Периферийное сканирование. Стандарт IEEE 1149.1».

Замечание: После окончания периода time-out задержки сброса напряжение питания микроконтроллера должно находиться в рабочем диапазоне. В случае, если внешняя схема подачи напряжения питания на микроконтроллер не гарантирует установку значения напряжения питания в рабочем диапазоне после завершения задержки сброса, рекомендуется использовать внешнюю схему сброса, которая будет удерживать микроконтроллер в режиме сброса до установки рабочего напряжения питания**.**

3.5.3 Сброс при подаче питания

Импульс сброса при подаче питания (POR) генерируется встроенной схемой. Пороговый уровень сброса приведен в таблице 3.19. POR активируется всякий раз, когда #VCC становится ниже порогового уровня. Схема POR может использоваться для запуска микроконтроллера, а также для выявления нарушения режима питания.

Сигнал POR гарантирует, что микроконтроллер будет сброшен при включении питания. Достижение порогового напряжения сброса при подаче питания активирует счетчик задержки, который устанавливает, как долго устройство будет находиться в состоянии сброса после повышения напряжения. Сигнал сброса активируется снова, безо всяких задержек, когда напряжение питания уменьшается ниже определенного уровня.

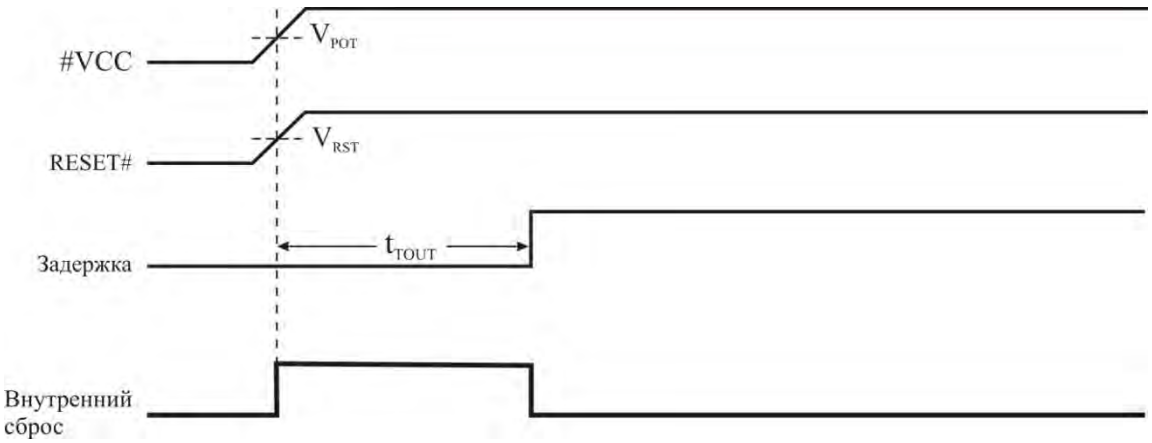


Рисунок 3.24 – Запуск микроконтроллера, RESET# связан с уровнем #VCC

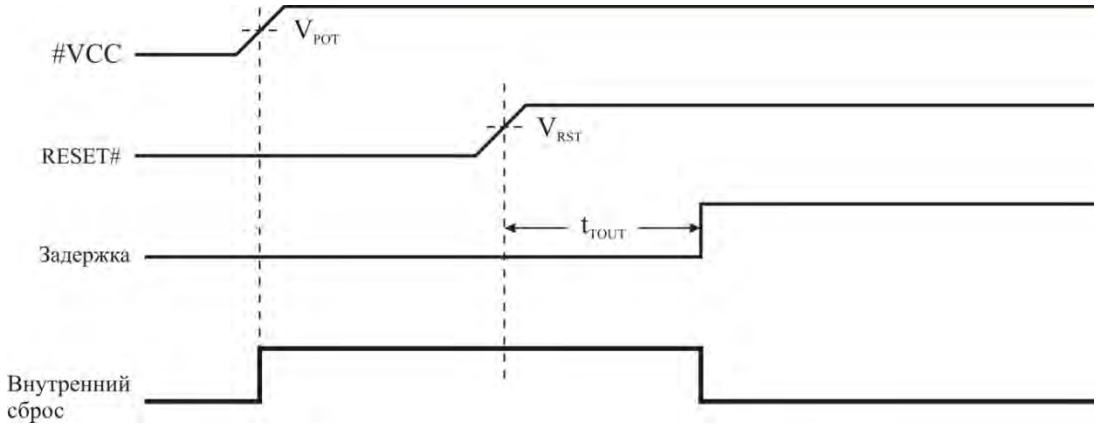


Рисунок 3.25 – Запуск микроконтроллера, RESET# управляется внешне

3.5.4 Внешний сброс

Внешний сброс генерируется, если на вход RESET# подать низкий уровень. Если подать импульс сброса длительностью более t_{RST} (см. таблицу 3.19), то будет сгенерирован сброс, даже если синхронизация не запущена. При подаче импульса сброса длительностью менее t_{RST} сброс не гарантируется. Если сигнал на выводе RESET# достигает порогового напряжения сброса V_{RST} на его положительном фронте, то запускается счетчик задержки, и микроконтроллер начнет работу по истечении периода задержки t_{TOUT}.

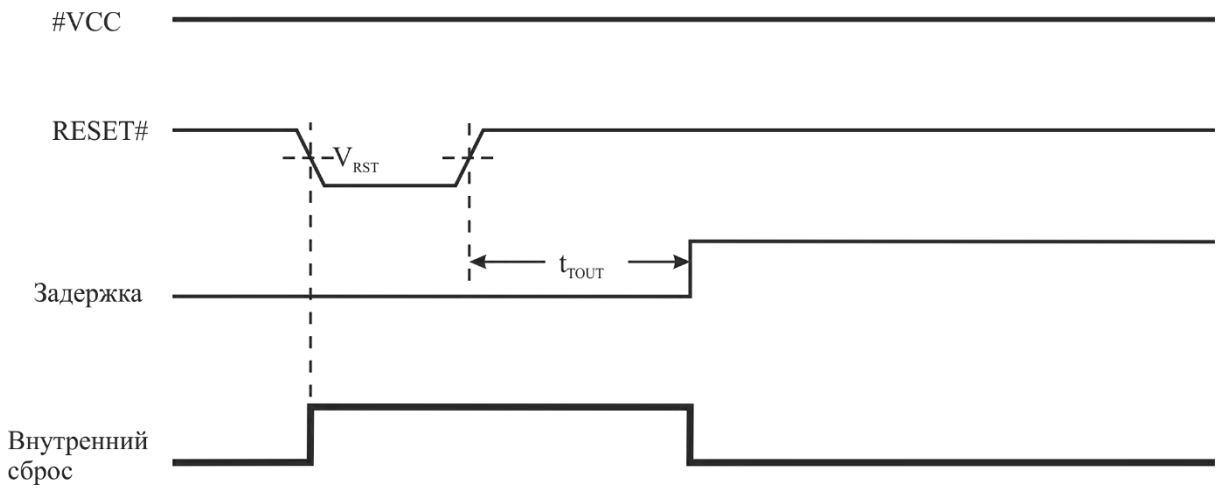


Рисунок 3.26 – Внешний сброс микроконтроллера

3.5.5 Контроль напряжения питания

ИС 1887ВЕ7Т содержит встроенную схему контроля питания (BOD), которая выполняет сравнение уровня #VCC с фиксированным пороговым значением. Порог срабатывания схемы BOD может выбираться с помощью конфигурационного бита BODLEVEL. Порог равен 2,7 В, когда BODLEVEL не запрограммирован, или 4 В, когда BODLEVEL запрограммирован. Для исключения автоколебательного режима схема BOD характеризуется гистерезисом. С учетом гистерезиса результирующие пороги срабатывания следующие: $V_{BOT+} = V_{BOT} + V_{HYST}/2$ и $V_{BOT-} = V_{BOT} – V_{HYST}/2$.

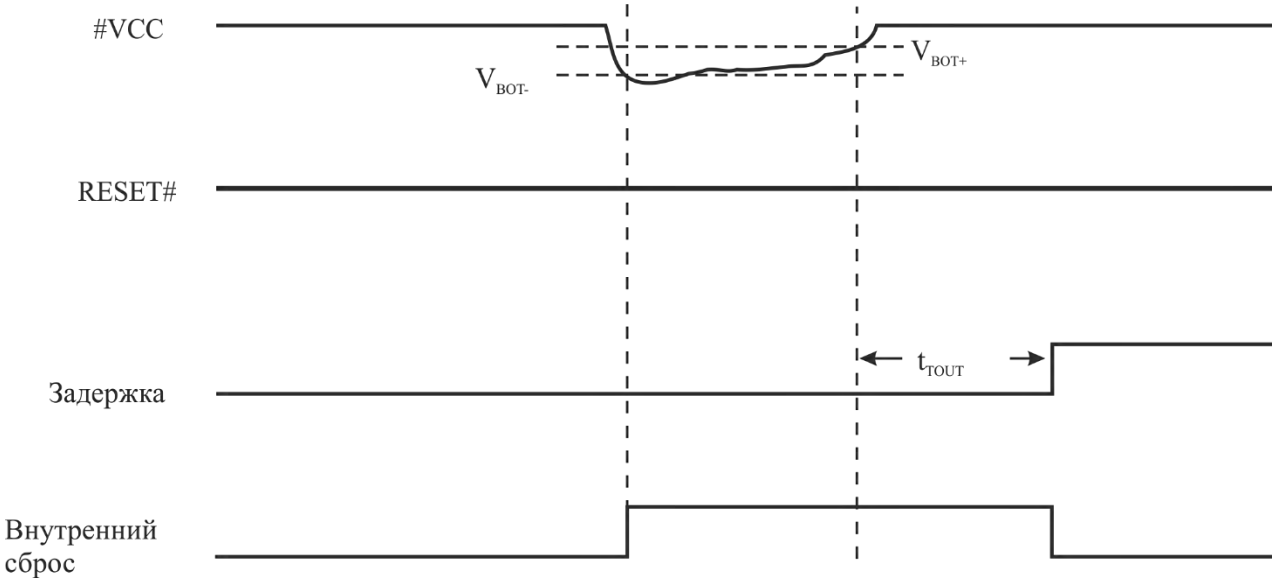


Рисунок 3.27 – Сброс микроконтроллера схемой контроля питания

Схема BOD может быть включена или отключена с помощью конфигурационного бита BODEN. Если разрешена работа BOD (BODEN запрограммирован) и уровень #VCC снизился ниже порога срабатывания (V_{BOT-} на рисунке 3.27), то схема BOD переводит микроконтроллер в состояние сброса. Когда #VCC достигает значения выше порога срабатывания (V_{BOT+} на рисунке 3.27), то запускается счетчик задержки, и микроконтроллер начнет работу по истечении времени t_{TOUT}.

Схема BOD реагирует на снижение #VCC. Если значение напряжения питания находится ниже порога срабатывания в течение периода времени, превышающего t_{BOD} (см. таблицу 3.19), формируется сигнал сброса.

3.5.6 Сброс от сторожевого таймера

По истечении периода переполнения сторожевого таймера генерируется короткий импульс сброса длительностью, равной одному периоду системной синхронизации (1 СК). Спадающим фронтом этого импульса запускается счетчик времени задержки t_{TOUT} (см. рисунок 3.28).

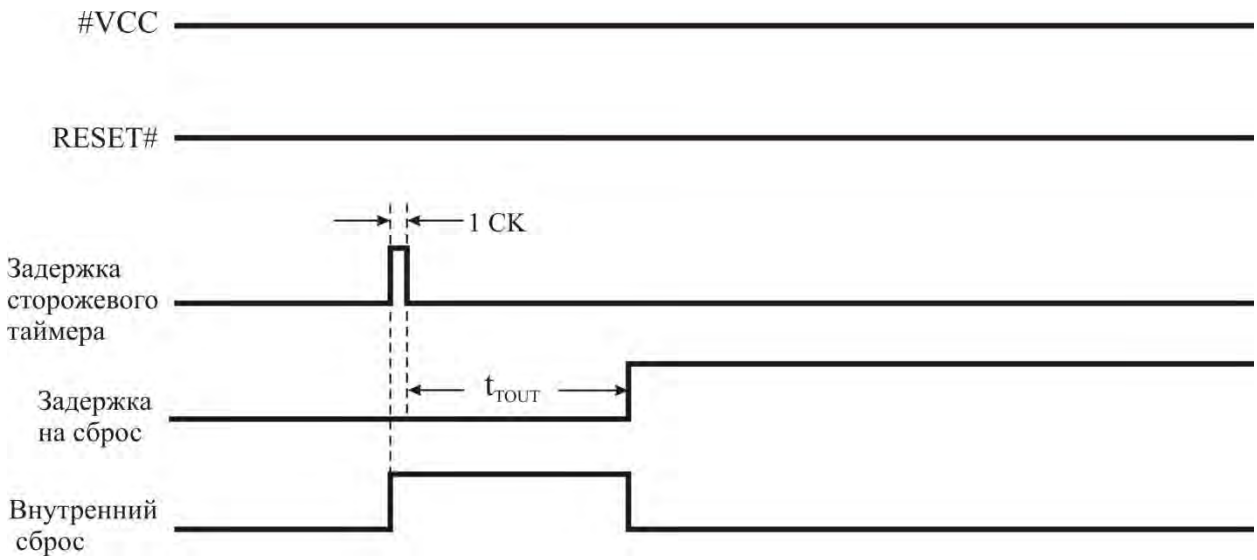


Рисунок 3.28 – Сброс сторожевым таймером

3.5.7 Регистр управления и состояния микроконтроллера – MCUCSR

В регистре управления и состояния микроконтроллера хранится информация об источнике, который вызвал сброс микроконтроллера.

Бит	7	6	5	4	3	2	1	0	
	JTD	-	-	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Чтение/запись	Ч/З	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	См. описание разрядов					

Разряд 4 – JTRF: Флаг индикации сброса через JTAG-интерфейс

Этот бит устанавливается, если сброс вызван записью логической единицы в регистр сброса JTAG с помощью команды AVR_RESET. Данный бит сбрасывается автоматически при подаче питания или путем непосредственной записи логического нуля во флаг.

Разряд 3 – WDRF: Флаг индикации сброса сторожевым таймером

Этот бит устанавливается, если происходит сброс сторожевым таймером. Данный бит сбрасывается автоматически при подаче питания или путем непосредственной записи логического нуля во флаг.

Разряд 2 – BORF: Флаг индикации сброса схемой контроля напряжения питания

Этот бит устанавливается, если происходит сброс при снижении питания. Данный бит сбрасывается автоматически при подаче питания или путем непосредственной записи логического нуля во флаг.

Разряд 1 – EXTRF: Флаг индикации внешнего сброса

Этот бит устанавливается, если происходит внешний сброс. Данный бит сбрасывается автоматически при подаче питания или путем непосредственной записи логического нуля во флаг.

Разряд 0 – PORF: Флаг индикации сброса при подаче питания

Этот бит устанавливается, если происходит сброс при подаче питания. Данный бит сбрасывается только путем непосредственной записи логического нуля во флаг.

Если флаги индикации сброса необходимо использовать для определения причины сброса, то пользователю следует предусмотреть в программе сброс значений флагов в регистре MCUCSR как можно раньше после их опроса. Если регистр очищается перед возникновением другого сброса, то источник нового сброса может быть найден среди флагов сброса.

3.5.8 Встроенный источник опорного напряжения

Микроконтроллер содержит встроенный источник опорного напряжения (ИОН). ИОН используется схемой контроля питания и может быть подключен к входу аналогового компаратора или АЦП. Опорное напряжение АЦП 2,56 В генерируется встроенным ИОН.

Сигналы разрешения и длительность запуска ИОН

ИОН характеризуется задержкой при включении, которая может оказать негативное влияние, если не будет учтена. Длительность задержки приведена в таблице 3.20. Для оптимизации энергопотребления ИОН не всегда находится во включенном состоянии. ИОН остается включенным в следующих случаях:

- Когда разрешена работа схемы контроля питания BOD (запрограммирован конфигурационный бит BODEN).
- Когда ИОН подключен к входу аналогового компаратора (установлен бит ACBG в регистре ACSR).
- Когда разрешена работа АЦП.

Следовательно, когда работа BOD запрещена после установки бита ACBG или разрешения работы АЦП, пользователь всегда должен предусматривать задержку на время запуска ИОН перед использованием выходных данных аналогового компаратора или АЦП. В целях снижения потребляемой мощности в режиме хранения пользователю следует избегать приведенных выше трех условий для гарантирования, что ИОН будет отключен прежде, чем микроконтроллер перейдет в режим хранения.

Таблица 3.20 – Характеристики встроенного ‘ИОН

Обозначение	Параметр	Мин.	Тип.	Макс.	Единица измерения
V_{BG}	Опорное напряжение	1,0	1,23	1,40	В
t_{BG}	Время установки опорного напряжения	-	40	70	мкс
I_{BG}	Ток потребления опорного напряжения	-	10	-	мкА

3.5.9 Сторожевой таймер

Сторожевой таймер тактируется от отдельного встроенного генератора с частотой 0,5 МГц. Данное значение типично для напряжения питания U_{CC1} = 5 В. Период сброса сторожевого таймера можно задавать путем управления предделителем (см. таблицу 3.22). Инструкция WDR выполняет сброс сторожевого таймера. Сторожевой таймер также сбрасывается при выключении и во время сброса микроконтроллера. Период сброса определяют восемь различных коэффициентов деления предделителя. Если период сброса истекает, а

команда WDR отсутствует, микроконтроллер сбрасывается и начинает выполнение программы по вектору сброса.

Для предотвращения неумышленного изменения периода переполнения или выключения сторожевого таймера могут быть использованы три различных уровня безопасности, которые выбираются конфигурационными битами M103C и WDTON, как показано в таблице 3.21. Не существует каких-либо ограничений на разрешение работы сторожевого таймера на любом из уровней безопасности (см. 3.5.10 «Временные последовательности для изменения конфигурации сторожевого таймера»).

Таблица 3.21 – Конфигурация сторожевого таймера при различных настройках битов M103C и WDTON

M103C	WDTON	Уровень безопасности	Исходное состояние сторожевого таймера	Возможность отключения сторожевого таймера	Возможность изменения периода переполнения
Не запрограммирован	Не запрограммирован	1	Отключен	Временная последовательность	Временная последовательность
Не запрограммирован	Запрограммирован	2	Включен	Всегда включен	Временная последовательность
Запрограммирован	Не запрограммирован	0	Отключен	Временная последовательность	Нет ограничений
Запрограммирован	Запрограммирован	2	Включен	Всегда включен	Временная последовательность

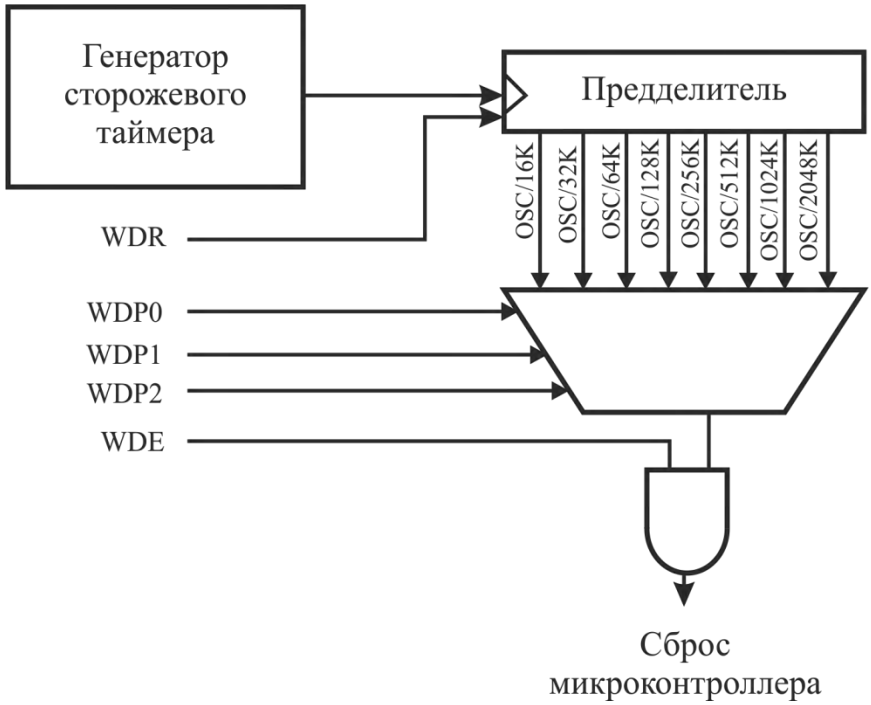


Рисунок 3.29 – Сторожевой таймер

Регистр управления сторожевым таймером – WDTCR

Бит	7	6	5	4	3	2	1	0	
	-	-	-	WDCE	WDE	WDP2	WDP1	WDP0	WDTCR
Чтение/запись	Ч	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7–5 зарезервировано

Эти разряды являются резервными и всегда читаются как ноль.

Разряд 4 – WDCE: Разрешение изменения сторожевого таймера

Данный бит необходимо установить непосредственно перед записью логического нуля в бит WDE. В противном случае запретить работу сторожевого таймера невозможно. После записи в данный бит логической единицы он автоматически аппаратно сбросится по истечении четырех тактов синхронизации микроконтроллера. На уровнях безопасности 1 и 2 данный бит также необходимо устанавливать перед изменением настроек предделителя. См. 3.5.10 «Временные последовательности для изменения конфигурации сторожевого таймера».

Разряд 3 – WDE: Включение сторожевого таймера

Работа сторожевого таймера разрешается (запрещается) путем установки (сброса) бита WDE. Сбросить бит WDE возможно, только если предварительно установить бит WDCE. Для выключения сторожевого таймера необходимо выполнить следующую последовательность:

1. Записать логическую единицу в WDCE и WDE одной инструкцией. Логическая единица должна быть записана в бит WDE, даже если до выполнения данной операции в нем уже была записана логическая единица.

2. В течение следующих четырех тактов записать логический ноль в WDE, что приводит к отключению сторожевого таймера.

На уровне безопасности 2 запретить работу сторожевого таймера невозможно даже с помощью указанной выше последов

ательности. См. пункт 3.5.10 «Временные последовательности для изменения конфигурации сторожевого таймера».

Разряд 2–0: WDP2, WDP1, WDP0: Выбор коэффициента деления

Биты WDP2, WDP1 и WDP0 задают коэффициент деления частоты генератора сторожевого таймера после разрешения работы последнего. Значения коэффициентов деления и соответствующих периодов переполнения приведены в таблице 3.22.

Таблица 3.22 – Настройка предделения генератора сторожевого таймера

WDP2	WDP1	WDP0	Число тактов генератора сторожевого таймера	Типовое время переполнения при U_{CC1} = 5,0 В
0	0	0	16 K (16384)	28,0 мс
0	0	1	32 K (32768)	56,2 мс
0	1	0	64 K (65536)	0,11 мс
0	1	1	128 K (131072)	0,22 с
1	0	0	256 K (262144)	0,45 с
1	0	1	512 K (524288)	0,9 с
1	1	0	1024 K (1048576)	1,8 с
1	1	1	2048 K (2097152)	3,6 с

В следующих примерах приведены функции выключения сторожевого таймера на Ассемблере и Си. В примерах предполагается, что система прерываний настроена таким образом, чтобы во время выполнения функции не возникло прерывание (например, с помощью общего запрета прерываний).

Пример кода на Ассемблере:

```
WDT_off:
; Сброс сторожевого таймера
wdr
in r16, WDTCR
; Запись логической единицы в WDCE и WDE
ori r16, (1<<WDCE)|(1<<WDE)
out WDTCR, r16
; Выключение сторожевого таймера
ldi r16, (0<<WDE)
out WDTCR, r16
ret
```

Пример кода на Си:

```
void WDT_off(void)
{
/* Сброс сторожевого таймера */
_watchdog_reset();
/* Запись логической единицы в WDCE и WDE */
WDTCR |= (1<<WDCE) | (1<<WDE);
/* Выключение сторожевого таймера */
WDTCR = 0x00;
}
```

3.5.10 Временные последовательности для изменения конфигурации сторожевого таймера

Режим управления настройками сторожевого таймера характеризуется тремя уровнями безопасности. Ниже описаны процедуры изменения настроек для каждого из уровней.

Уровень безопасности 0

Сторожевой таймер изначально отключен, но может быть активирован путем записи в бит WDE логической единицы без каких-либо ограничений. Период переполнения таймера может быть изменен в любой момент без ограничений. Для выключения активизированного сторожевого таймера должна быть выполнена процедура, описанная при рассмотрении бита WDE.

Уровень безопасности 1

В данном режиме сторожевой таймер изначально отключен. Его работа может быть разрешена путем записи логической единицы в бит WDE без каких-либо ограничений. Временная последовательность должна быть соблюдена при изменении периода переполнения или выключении активизированного сторожевого таймера. В данном случае должна быть выполнена следующая процедура:

1. Записать логическую единицу в WDCE и WDE одной инструкцией. Логическая единица должна быть записана в WDE независимо от предыдущего значения бита WDE.
2. В течение следующих четырех тактов одной инструкцией записать желаемое значение в биты WDE и WDP, одновременно сбрасывая бит WDCE.

Уровень безопасности 2

В данном режиме сторожевой таймер всегда включен, а значение бита WDE всегда считывается как логическая единица. Временная последовательность должна быть соблюдена при изменении периода переполнения сторожевого таймера. При этом должна быть выполнена следующая процедура:

1. Записать логическую единицу в WDCE и WDE одной инструкцией. Несмотря на то, что бит WDE всегда установлен, для запуска временной последовательности в него все равно необходимо записывать логическую единицу.

2. В течение следующих четырех тактов одной инструкцией записать желаемое значение в биты WDP, одновременно сбрасывая бит WDCE. Значение, записываемое в бит WDE, не оказывает никакого влияния.

3.6 Прерывания

В данном подразделе описывается специфика обработки прерываний, реализованная в ИС 1887ВЕ7Т. Общая информация, касающаяся обработки прерываний микроконтроллера, приведена в пункте 3.1.7 «Сброс и обработка прерываний».

3.6.1 Векторы прерываний

Таблица 3.23 – Векторы сброса и прерываний

Номер вектора	Адрес памяти программ*	Источник	Условие возникновения прерывания
1	$0000**	RESET	Внешний сброс, сброс при подаче питания, сброс при снижении питания, сброс от сторожевого таймера и сброс через JTAG-интерфейс
2	$0002	INT0	Запрос на внешнее прерывание 0
3	$0004	INT1	Запрос на внешнее прерывание 1
4	$0006	INT2	Запрос на внешнее прерывание 2
5	$0008	INT3	Запрос на внешнее прерывание 3
6	$000A	INT4	Запрос на внешнее прерывание 4
7	$000C	INT5	Запрос на внешнее прерывание 5
8	$000E	INT6	Запрос на внешнее прерывание 6
9	$0010	INT7	Запрос на внешнее прерывание 7
10	$0012	TIMER2 COMP	Срабатывание компаратора таймера/счетчика 2
11	$0014	TIMER2 OVF	Переполнение таймера/счетчика 2
12	$0016	TIMER1 CAPT	Захват таймером/счетчиком 1
13	$0018	TIMER1 COMPA	Срабатывание компаратора А таймера/счетчика 1
14	$001A	TIMER1 COMPB	Срабатывание компаратора В таймера/счетчика 1
15	$001C	TIMER1 OVF	Переполнение таймера/счетчика 1
16	$001E	TIMER0 COMP	Срабатывание компаратора таймера/счетчика 0
17	$0020	TIMER0 OVF	Переполнение таймера/счетчика 0
18	$0022	SPI, STC	Завершение последовательной передачи данных интерфейсом SPI
19	$0024	USART0, RX	Завершение приема USART 0
20	$0026	USART0, UDRE	Регистр данных USART 0 свободен
21	$0028	USART0, TX	Завершение передачи USART 0
22	$002A	ADC	Завершение преобразования АЦП
23	$002C	EE READY	Готовность ЭПД
24	$002E	ANALOG COMP	Срабатывание аналогового компаратора
25	$0030	TIMER1 COMPC	Срабатывание компаратора С таймера/счетчика 1
26	$0032	TIMER3 CAPT	Захват таймером/счетчиком 3
27	$0034	TIMER3 COMPA	Срабатывание компаратора А таймера/счетчика 3
28	$0036	TIMER3 COMPB	Срабатывание компаратора В таймера/счетчика 3
29	$0038	TIMER3 COMPC	Срабатывание компаратора С таймера/счетчика 3
30	$003A	TIMER3 OVF	Переполнение таймера/счетчика 3
31	$003C	USART1, RX	Завершение приема USART 1
32	$003E	USART1, UDRE	Регистр данных USART 1 свободен
33	$0040	USART1, TX	Завершение передачи USART 1

Окончание таблицы 3.23

Номер вектора	Адрес памяти программ*	Источник	Условие возникновения прерывания
34	$0042	TWI	Двухпроводной последовательный интерфейс
35	$0044	SPM READY	Готовность записи в память программ

_____

* Если установлен бит IVSEL в регистре MCUCR, то векторы прерываний перемещаются в начало загрузочного сектора памяти программ. В этом случае к адресу каждого вектора прерывания из таблицы прибавляется стартовый адрес загрузочного сектора памяти программ.

** Если конфигурационный бит BOOTRST запрограммирован, то микроконтроллер выполняет переход на адрес сброса в загрузочном секторе, см. подраздел 3.20 «Самопрограммирование из сектора начальной загрузки с поддержкой чтения во время записи».

В таблице 3.24 показано расположение векторов сброса и прерываний в зависимости от различных установок BOOTRST и IVSEL. Если программа не использует прерывания, то она может быть размещена равномерно, используя ячейки с адресами векторов прерываний для хранения программного кода. Возможен также случай, когда вектор сброса располагается в секторе прикладной программы, а векторы прерываний – в секторе загрузчика или наоборот.

Таблица 3.24 – Размещение векторов сброса и прерываний

BOOTRST	IVSEL	Адрес сброса	Начальный адрес вектора прерывания
1	0	$0000	$0002
1	1	$0000	Адрес сброса в секторе загрузчика + $0002
0	0	Адрес сброса в секторе загрузчика	$0002
0	1	Адрес сброса в секторе загрузчика	Адрес сброса в секторе загрузчика + $0002

Примечание – Адрес сброса в секторе загрузчика см. в 3.20.7 «Самопрограммирование памяти программ». Для конфигурационного бита BOOTRST значение «1» означает незапрограммированное состояние, а «0» означает запрограммированное состояние.

Ниже представлена наиболее типичная и общая программная установка для векторов сброса и прерываний в ИС 1887ВЕ7Т.

```
Адрес   Команда      Операнд                Комментарий

$0000     jmp          RESET      ;Переход на обработку сброса
$0002     jmp          EXT_INT0   ;Переход на обработку запроса IRQ0
$0004     jmp          EXT_INT1   ;Переход на обработку запроса IRQ1
$0006     jmp          EXT_INT2   ;Переход на обработку запроса IRQ2
$0008     jmp          EXT_INT3   ;Переход на обработку запроса IRQ3
$000A     jmp          EXT_INT4   ;Переход на обработку запроса IRQ4
$000C     jmp          EXT_INT5   ;Переход на обработку запроса IRQ5
$000E     jmp          EXT_INT6   ;Переход на обработку запроса IRQ6
$0010     jmp          EXT_INT7   ;Переход на обработку запроса IRQ7
$0012     jmp          TIM2_COMP  ;Переход на обработку при срабатывании
                                  ;компаратора таймера 2
$0014     jmp          TIM2_OVF   ;Переход на обработку при переполнении
                                  ;таймера 2
$0016     jmp          TIM1_CAPT  ;Переход на обработку при захвате фронта
                                  ;таймером  1
$0018     jmp          TIM1_COMPA ;Переход на обработку при срабатывании
                                  ;компаратора A таймера 1
$001A     jmp          TIM1_COMPB ;Переход на обработку при срабатывании
                                  ;компаратора B таймера 1
```

```
$001C     jmp         TIM1_OVF      ;Переход на обработку при переполнении
                                    ;таймера 1
$001E     jmp         TIM0_COMP     ;Переход на обработку при выполнения
                                    ;условия сравнения таймера 0
$0020     jmp         TIM0_OVF      ;Переход на обработку при переполнении
                                    ;таймера 0
$0022     jmp         SPI_STC       ;Переход на обработку при завершении
                                    ;передачи данных интерфейсом SPI
$0024     jmp         USART0_RXC    ;Переход на обработку при завершении
                                    ;приема USART 0
$0026     jmp         USART0_DRE    ;Переход на обработку при освобождении
                                    ;регистра данных UDR USART 0
$0028     jmp         USART0_TXC    ;Переход на обработку при завершении
                                    ;передачи USART 0
$002A     jmp            ADC        ;Переход на обработку при завершении
                                    ;преобразования АЦП
$002C     jmp          EE_RDY       ;Переход на обработку при готовности
                                    ;ЭПД
$002E     jmp         ANA_COMP      ;Переход на обработку при срабатывании
                                    ;аналогового компаратора
$0030     jmp         TIM1_COMPC    ;Переход на обработку при срабатывании
                                    ;компаратора С таймера 1
$0032     jmp         TIM3_CAPT     ;Переход на обработку при захвате фронта
                                    ;таймером 3
$0034     jmp         TIM3_COMPA    ;Переход на обработку при срабатывании
                                    ;компаратора А таймера 3
$0036     jmp         TIM3_COMPB    ;Переход на обработку при срабатывании
                                    ;компаратора В таймера 3
$0038     jmp         TIM3_COMPC    ;Переход на обработку при срабатывании
                                    ;компаратора С таймера 3
$003A     jmp         TIM3_OVF      ;Переход на обработку при переполнении
                                    ;таймера 3
$003C     jmp         USART1_RXC    ;Переход на обработку при завершении
                                    ;приема USART 1
$003E     jmp         USART1_DRE    ;Переход на обработку при освобождении
                                    ;регистра данных UDR USART 1
$0040     jmp         USART1_TXC    ;Переход на обработку при завершении
                                    ;передачи USART 1
$0042     jmp            TWI        ;Переход на обработку прерывания по
                                    ;двухпроводному последовательному интерфейсу
$0044     jmp          SPM_RDY      ;Переход на обработку прерывания при
                                    ;готовности выполнения команды SPM
  ;
$0046  RESET:ldi  r16,high(RAMEND)  ;Начало основной программы
$0047     out          SPH,r16      ;Установка указателя стека в конце области
                                    ОЗУ
$0048     ldi      r16, low(RAMEND)
$0049     out          SPL,r16
$004A     sei                       ;Разрешение прерываний
$004B   <instr>          xxx
 ...      ...            ...
```

Если конфигурационный бит BOOTRST не запрограммирован, размер сектора загрузчика установлен 8 Кбайт, и бит IVSEL в регистре MCUCR устанавливается прежде, чем разрешается любое прерывание, то наиболее типичная и общая программная установка для векторов сброса и прерываний выглядит следующим образом:

```
Адрес   Команда       Операнд                    Комментарий

$0000  RESET:ldi  r16,high(RAMEND)  ;Начало основной программы
$0001     out          SPH,r16      ;Установка указателя стека в конце области
                                    ОЗУ
$0002     ldi       r16,low(RAMEND)
```

```
$0003     out          SPL,r16
$0004     sei                        ;Разрешение прерываний
$0005    <instr>         xxx
  ;
   .org $F002
$F002     jmp          EXT_INT0      ;Переход на обработку запроса IRQ0
$F004     jmp          EXT_INT1      ;Переход на обработку запроса IRQ1
 ...      ...            ...
$F044     jmp          SPM_RDY       ;Переход на обработку прерывания при
                                     ;готовности выполнения команды SPM
```

Если конфигурационный бит BOOTRST запрограммирован, и размер загрузочного сектора установлен 8 Кбайт, то наиболее типичная и общая программная установка для векторов сброса и прерываний выглядит следующим образом:

```
Адрес    Команда       Операнд                    Комментарий

   .org $0002
$0002     jmp          EXT_INT0      ;Переход на обработку запроса IRQ0
$0004     jmp          EXT_INT1      ;Переход на обработку запроса IRQ1
 ...      ...            ...
$0044     jmp          SPM_RDY       ;Переход на обработку прерывания при
                                     ;готовности выполнения команды SPM

   .org $F000
$F000   RESET:ldi   r16,high(RAMEND) ;Начало основной программы
$F001     out          SPH,r16       ;Установка указателя стека в конце области
                                     ОЗУ
$F002     ldi       r16,low(RAMEND)
$F003     out          SPL,r16
$F004     sei                        ;Разрешение прерываний
$F005    <instr>         xxx
```

Если конфигурационный бит BOOTRST запрограммирован, размер загрузочного сектора установлен 8 Кбайт, и бит IVSEL в регистре MCUCR устанавливается прежде, чем разрешается любое прерывание, то наиболее типичная и общая программная установка для векторов сброса и прерываний выглядит следующим образом:

```
Адрес    Команда       Операнд                    Комментарий

   .org $F000
$F000     jmp           RESET        ;Переход на обработку сброса
$F002     jmp          EXT_INT0      ;Переход на обработку запроса IRQ0
$F004     jmp          EXT_INT1      ;Переход на обработку запроса IRQ1
 ...      ...            ...
$F044     jmp          SPM_RDY       ;Переход на обработку прерывания при
                                     ;готовности выполнения команды SPM
$F046   RESET:ldi   r16,high(RAMEND) ;Начало основной программы
$F047     out          SPH,r16       ;Установка указателя стека в конце области
                                     ОЗУ
$F048     ldi       r16,low(RAMEND)
$F049     out          SPL,r16
$F04A     sei                        ;Разрешение прерываний
$F04B    <instr>         xxx
```

Перемещение между секторами загрузочной и прикладной программы

Общий регистр управления прерываниями задает размещение таблицы векторов прерываний.

Регистр управления микроконтроллером – MCUCR

Бит	7	6	5	4	3	2	1	0	
	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE	MCUCR
Чтение/запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 1 – IVSEL: Выбор вектора прерывания

Если бит IVSEL сброшен, то векторы прерываний размещаются в начале памяти программ. Если данный бит установлен в единицу, то векторы прерываний перемещаются в начало загрузочного сектора памяти программ. Фактический адрес начала загрузочного сектора определяется значением конфигурационных битов BOOTSZ. См. подраздел 3.20 «Самопрограммирование из сектора начальной загрузки с поддержкой чтения во время записи» для выяснения подробностей. Во избежание непреднамеренных изменений таблицы векторов прерываний, необходимо выполнить специальную последовательность записи при изменении бита IVSEL:

1. Записать логическую единицу в бит разрешения изменения вектора прерывания (IVCE).

2. В течение четырех машинных циклов записать желаемое значение в бит IVSEL, при этом записывая логический ноль в бит IVCE.

Прерывания будут автоматически отключены при выполнении такой последовательности. Прерывания отключаются во время установки IVCE и остаются отключенными до перехода к инструкции, следующей за командой записи в IVSEL. Если бит IVSEL не записан, то прерывания остаются отключенными в течение четырех машинных циклов. Состояние бита I в регистре статуса не изменяется при автоматическом отключении прерываний.

Если векторы прерываний помещены в загрузочный сектор и бит защиты загрузочного сектора BLB02 запрограммирован, то прерывания будут отключены при выполнении программы в прикладном секторе. Если векторы прерываний размещены в прикладном секторе и бит защиты BLB12 запрограммирован, то прерывания будут отключены при выполнении программы в загрузочном секторе. См. также подраздел 3.20 «Самопрограммирование из сектора начальной загрузки с поддержкой чтения во время записи» для более подробного изучения битов защиты.

Разряд 0 – IVCE: Разрешение изменения вектора прерывания

В бит IVCE должна быть записана логическая единица, чтобы разрешить изменение бита IVSEL. IVCE сбрасывается аппаратно через четыре машинных цикла после установки, или когда записывается логическая единица в IVSEL. Установка бита IVCE приведет к отключению прерываний, как описано выше при рассмотрении бита IVSEL. Ниже приведен пример кода.

Пример кода на Ассемблере:

```
Move_interrupts:
; Разрешение изменения векторов прерываний
ldi r16, (1<<IVCE)
out MCUCR, r16
; Перемещение векторов в загрузочный сектор памяти программ
ldi r16, (1<<IVSEL)
out MCUCR, r16
ret
```

Пример кода на Си:

```
void Move_interrupts(void)
{
/* Разрешение изменения векторов прерываний */
MCUCR = (1<<IVCE);
/* Перемещение векторов в загрузочной сектор памяти программ*/
MCUCR = (1<<IVSEL);
}
```

3.7 Порты ввода-вывода

Все порты ввода-вывода работают по принципу чтение-модификация-запись, когда используются в качестве общих портов цифрового ввода-вывода. Это означает, что направление одной линии порта можно изменить с помощью команд SBI и CBI без изменений направления ввода-вывода других линий порта. Это распространяется также и на изменение логического уровня (если линия порта настроена на вывод) или на включение/отключение подтягивающих резисторов (если линия настроена на ввод). Каждый выходной буфер имеет симметричную характеристику как стока, так и истока. Выходной драйвер достаточно мощный, чтобы непосредственно управлять светодиодными индикаторами. Все линии портов имеют индивидуально выбираемые подтягивающие резисторы, сопротивление которых не зависит от напряжения питания. На всех линиях портов ввода-вывода установлены защитные диоды, которые подключены и к напряжению питания, и к земле, как показано на рисунке 3.30.

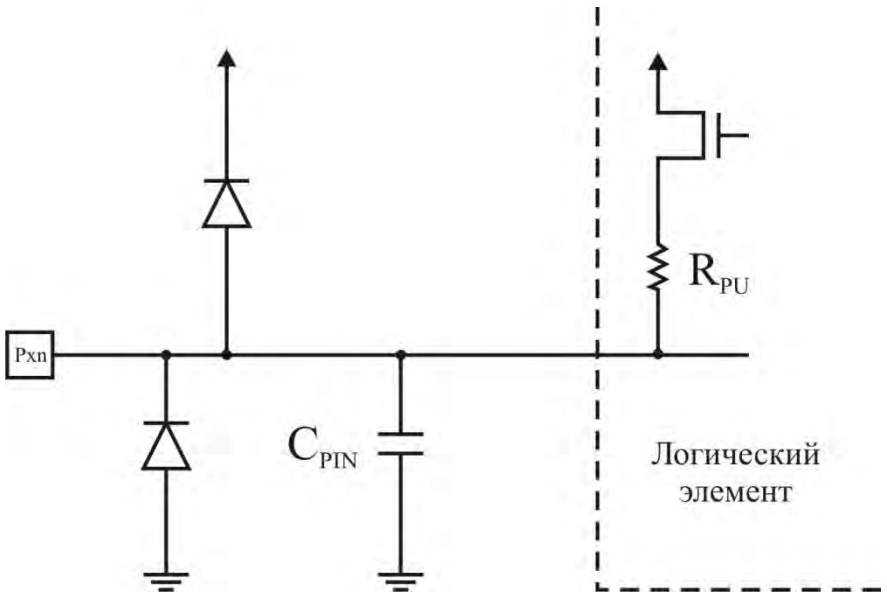


Рисунок 3.30 – Эквивалентная схема линии порта ввода-вывода

Примечание – Подробное описание логического элемента см. на рисунке 3.31.

Ссылки на регистры и биты регистров в данном разделе даны в общей форме. При этом символ «x» заменяет наименование порта ввода-вывода, а символ «n» заменяет номер разряда. Однако при составлении программы необходимо использовать точную форму записи. Например, запись «PORTB3», означающая разряд 3 порта B, в данном документе обозначается как «PORTxn». Физические регистры ввода-вывода и распределение их разрядов представлено в пункте «Описание регистров для портов ввода-вывода».

Для каждого порта в памяти ввода-вывода зарезервировано три ячейки: одна – для регистра данных (PORTx), другая – для регистра направления данных (DDRx) и третья –

для хранения состояния входов порта – PINx. Ячейка, хранящая состояние на входах порта, доступна только для чтения, а регистры данных и направления данных имеют двунаправленный доступ. Кроме того, установка бита выключения подтягивающих резисторов PUD регистра SFIOR отключает функцию подтягивания на всех выводах всех портов.

Ниже приведено описание использования порта для общего цифрового ввода-вывода. Большинство выводов портов поддерживают альтернативные функции встроенных периферийных устройств микроконтроллера. Описание альтернативных функций приведено далее в 3.7.2 «Альтернативные функции портов» (см. также описание функций соответствующих периферийных модулей).

Обратите внимание, что для некоторых портов разрешение альтернативных функций некоторых выводов делает невозможным использование других выводов для общего цифрового ввода-вывода.

3.7.1 Порты в качестве общего цифрового ввода-вывода

Порты являются двунаправленными портами ввода-вывода с дополнительной внутренней нагрузкой. Рисунок 3.31 иллюстрирует функциональную схему одной линии порта ввода-вывода, обозначенной как Pxn.

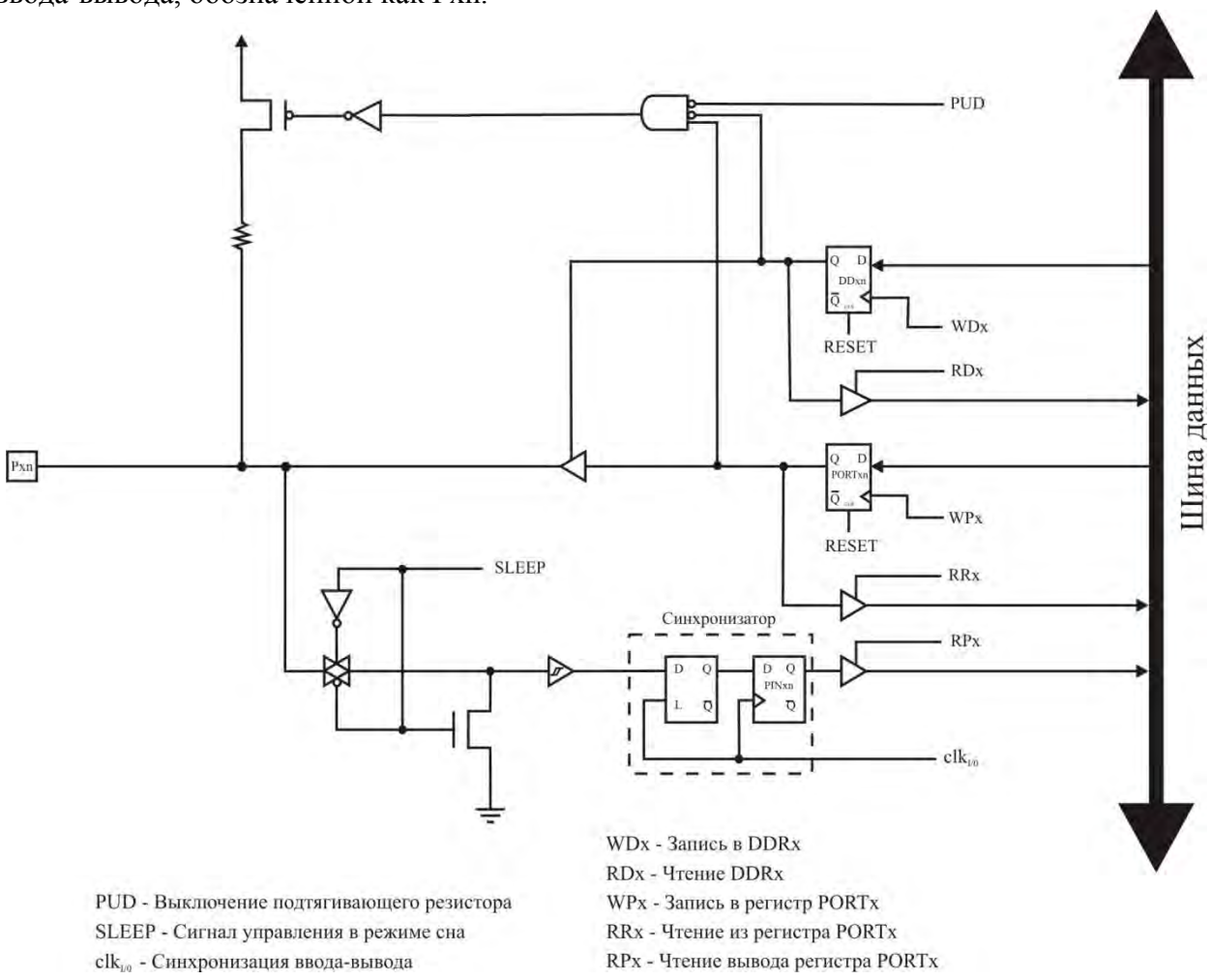


Рисунок 3.31 – Организация общего цифрового ввода-вывода

Примечание – Сигналы WPx, WDx, RRx, RPx и RDx являются общими для всех линий в пределах одного порта. Сигналы $clk_{I/O}$, SLEEP и PUD являются общими для всех портов.

Настройка выводов

Режим и состояние для каждого вывода определяется значением соответствующих разрядов трех регистров: DDxn, PORTxn и PINxn. Как показано в 3.7.3 «Описание регистров для портов ввода-вывода», доступ к битам DDxn возможен по адресу DDRx в пространстве ввода-вывода, к битам PORTxn – по адресу PORTx, а к битам PINxn – по адресу PINx.

Биты DDxn регистра DDRx определяют направленность линии ввода-вывода. Если DDxn = 1, то Pxn конфигурируется на вывод. Если DDxn = 0, то Pxn конфигурируется на ввод.

Если PORTxn = 1 при конфигурации линии порта на ввод, разрешается подключение подтягивающего резистора. Для выключения данного резистора необходимо записать в PORTxn логический ноль или настроить линию порта на вывод. Во время сброса все линии портов находятся в третьем (высокоимпедансном) состоянии, даже если не работает синхронизация.

Если PORTxn = 1 при конфигурации линии порта на вывод, то состояние вывода устанавливается в единицу. Если PORTxn = 0 при конфигурации линии порта на вывод, то состояние вывода устанавливается в ноль.

При переключении между третьим состоянием ({DDxn, PORTxn} = 0b00) и высоким логическим уровнем ({DDxn, PORTxn} = 0b11) должно быть промежуточное состояние либо с подключенным подтягивающим резистором ({DDxn, PORTxn} = 0b01), либо с выводом низкого логического уровня ({DDxn, PORTxn} = 0b10). Как правило, переход через состояние с подключением подтягивающего резистора эквивалентен состоянию вывода высокого уровня, если вывод микроконтроллера связан с высокоимпедансным входом. Если это не так, то можно записать единицу в бит PUD регистра SFIOR, чтобы выключить все подтягивающие резисторы во всех портах.

Переключение между вводом с подтягивающим резистором и выводом низкого уровня связано с такой же проблемой. Поэтому пользователь должен использовать либо третье состояние ({DDxn, PORTxn} = 0b00), либо вывод высокого уровня ({DDxn, PORTxn} = 0b11) в качестве промежуточного шага.

В таблице 3.25 приводится действие управляющих сигналов на состояние вывода.

Таблица 3.25 – Настройка вывода порта

DDxn	PORTxn	PUD (в SFIOR)	Ввод-вывод	Подтягивающий резистор	Комментарий
0	0	X	Ввод	Нет	Третье состояние (Z-состояние)
0	1	0	Ввод	Да	Pxn будет источником тока при подаче внешнего низкого уровня
0	1	1	Ввод	Нет	Третье состояние (Z-состояние)
1	0	X	Вывод	Нет	Вывод низкого уровня (сток)
1	1	X	Вывод	Нет	Вывод высокого уровня (исток)

Считывание состояния вывода

Независимо от значения бита направления данных DDxn состояние вывода порта может быть опрошено через регистровый бит PINxn. Как показано на рисунке 3.31, регистровый бит PINxn и предшествующая ему триггерная защелка составляют синхронизатор. Данный подход позволяет избежать метастабильности, если изменение состояния на выводе происходит около фронта внутренней синхронизации. Однако такой подход связан с возникновением задержки. На рисунке 3.32 представлена временная диаграмма синхронизации при считывании поданного уровня сигнала на разряд n порта x. Длительности максимальной и минимальной задержек на распространение сигнала обозначены как $t_{pd,max}$ и $t_{pd,min}$, соответственно.

Рассмотрим период тактирования, начинающийся вскоре после первого спадающего фронта системной синхронизации. Триггер защелкивается, когда сигнал тактирования имеет низкий уровень, и открывается, когда сигнал тактирования имеет высокий уровень, что показывает заштрихованная область сигнала «Синхронизирующая защелка» на рисунке 3.32. Значение сигнала защелкивается при спадающем фронте тактового сигнала. Это значение вносится в регистровый бит PINxn на последующем нарастающем фронте синхронизации. Как показывают два временных отрезка $t_{pd,max}$ и $t_{pd,min}$, переход отдельного сигнала будет иметь задержку в пределах от половины до полутора периода синхронизации в зависимости от момента установки сигнала.

При считывании состояния вывода, установленного программно, должна быть вложена инструкция nop, как показано на рисунке 3.33. Команда out устанавливает сигнал «Синхронизирующая защелка» на нарастающем фронте тактового сигнала. В этом случае задержка распространения сигнала через синхронизатор t_{pd} будет равна одному периоду тактового сигнала.

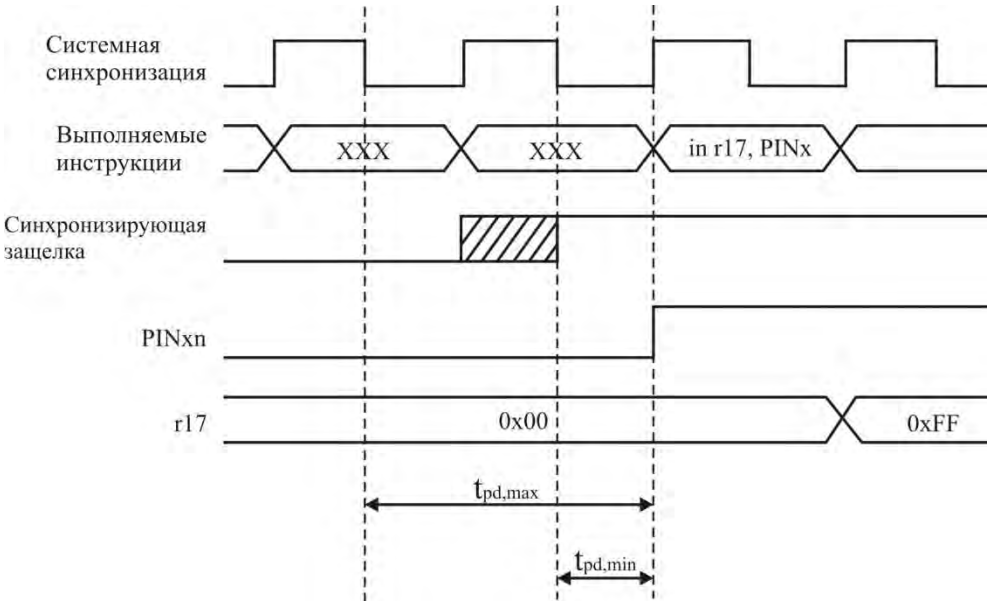


Рисунок 3.32 – Синхронизация при считывании значения, поданного на разряд n порта x ввода-вывода общего назначения

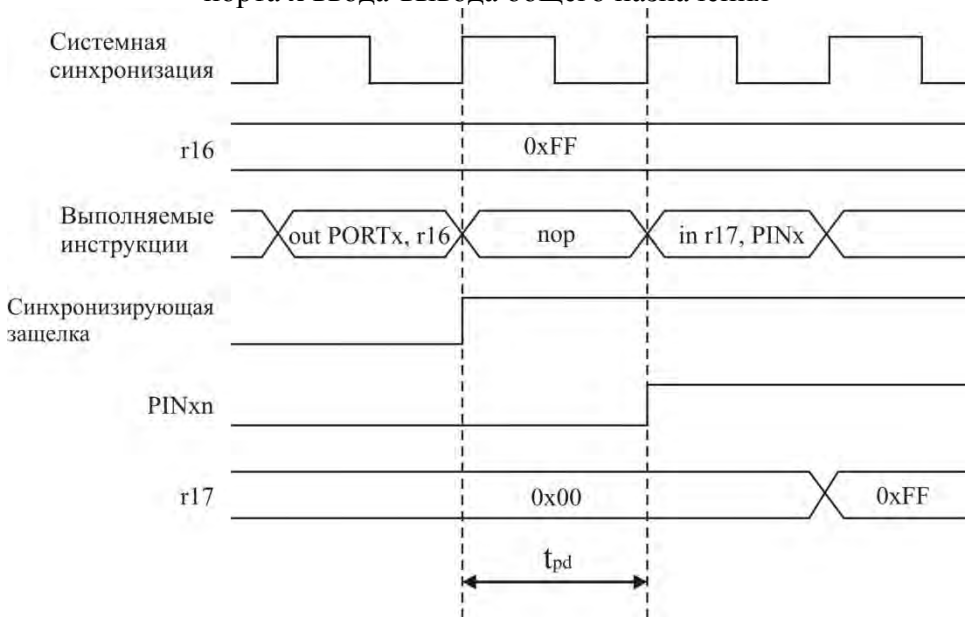


Рисунок 3.33 – Синхронизация при считывании значения, программно установленного при конфигурации порта на вывод данных

В следующих примерах показано как установить на линиях 0 и 1 порта B уровень логической единицы, на линиях 2 и 3 – уровень логического нуля, а также как настроить линии 4…7 на ввод с подключением подтягивающих резисторов на линиях 6 и 7. Результирующее состояние линий считывается обратно, но с учетом сказанного выше, включена инструкция nop для обеспечения возможности обратного считывания только что назначенного состояния некоторых выводов.

Пример кода на Ассемблере:

```
...
; Разрешение подтягивающих резисторов и установка высокого уровня на выводах
; Определение направления для линий порта
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
; Вставка инструкции nop для синхронизации
nop
; Считывание состояния выводов порта
in r16,PINB
...
```

Пример кода на Си:

```
unsigned char i;
...
/* Разрешение подтягивающих резисторов и установка высокого уровня */
/* на выводах */
/* Определение направления для линий порта */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
/* Вставка инструкции nop для синхронизации */
_no_operation();
/* Считывание состояния выводов порта */
i = PINB;
...
```

Примечание – В программе на Ассемблере используются два временных регистра для минимизации интервала времени от настройки подтягивающих резисторов на линиях 0, 1, 6 и 7 до корректной установки битов направления. Биты направления разрешают вывод логического нуля на линиях 2 и 3 и переопределяют линии 0 и 1 как мощные драйверы с высоким уровнем.

Разрешение цифрового ввода и спящие режимы

Как показано на рисунке 3.31, входной цифровой сигнал может быть подключен на «землю» на входе триггера Шмитта. Сигнал, обозначенный на рисунке как SLEEP, устанавливается при переводе микроконтроллера в режим микропотребления, режим хранения, режим ожидания и расширенный режим ожидания. Это позволяет избежать повышенного энергопотребления в случае, если некоторые входные сигналы окажутся в плавающем состоянии, или уровень входного аналогового сигнала будет близок к $U_{CC2}/2$.

Сигнал SLEEP игнорируется линиями порта, задействованными в качестве линий внешних прерываний. Если внешние прерывания запрещены, то SLEEP действует и на эти выводы. SLEEP также игнорируется на других входах при выполнении их альтернативных функций (см. 3.7.2 «Альтернативные функции портов»).

Если присутствует уровень логической единицы на выводе асинхронного внешнего прерывания, который сконфигурирован на прерывание по нарастающему фронту, спадаю-

щему фронту или по любому логическому изменению на линии в то время как внешнее прерывание не разрешено, то при выходе из вышеупомянутых спящих режимов будет установлен соответствующий флаг внешнего прерывания.

Неподключенные выводы

Если некоторые выводы остаются неиспользованными, то рекомендуется обеспечить присутствие на них определенного логического уровня. Несмотря на то, что большинство цифровых входов отключено в режимах глубокого сна, как описано выше, следует избегать наличия плавающих входов, чтобы уменьшить потребление тока во всех других режимах работы микроконтроллера, где цифровой ввод разрешен (сброс, активный режим и режим холостого хода).

Самым простым методом обеспечения присутствия определенного уровня на неиспользуемом выводе является разрешение подключения внутреннего подтягивающего резистора. Однако в этом случае в режиме сброса подтягивающий резистор будет отключен. Если требуется малое энергопотребление и в режиме сброса, то следует использовать внешний резистор, подтягивающий к плюсу или к минусу питания. Подключение неиспользуемых выводов непосредственно к #VCC или #0V не рекомендуется, так как может возникнуть опасный ток короткого замыкания при случайной конфигурации такого вывода на выдачу данных.

3.7.2 Альтернативные функции портов

Большинство выводов поддерживает альтернативные функции в дополнение к функции общего цифрового ввода-вывода. На рисунке 3.34 показано, как функции управляющих сигналов, представленные на упрощенном рисунке 3.31, могут быть переопределены альтернативными функциями. Сигналы переопределения функций могут присутствовать не на всех выводах порта, поэтому данный рисунок необходимо рассматривать как общее описание, применимое ко всем выводам портов.

В таблице 3.26 приведены описания сигналов переопределения функций при активации альтернативных функций. Индексы выводов и портов с рисунка 3.34 не показаны в последующих таблицах. Сигналы переопределения функций генерируются внутренне в модулях, поддерживающих альтернативные функции.

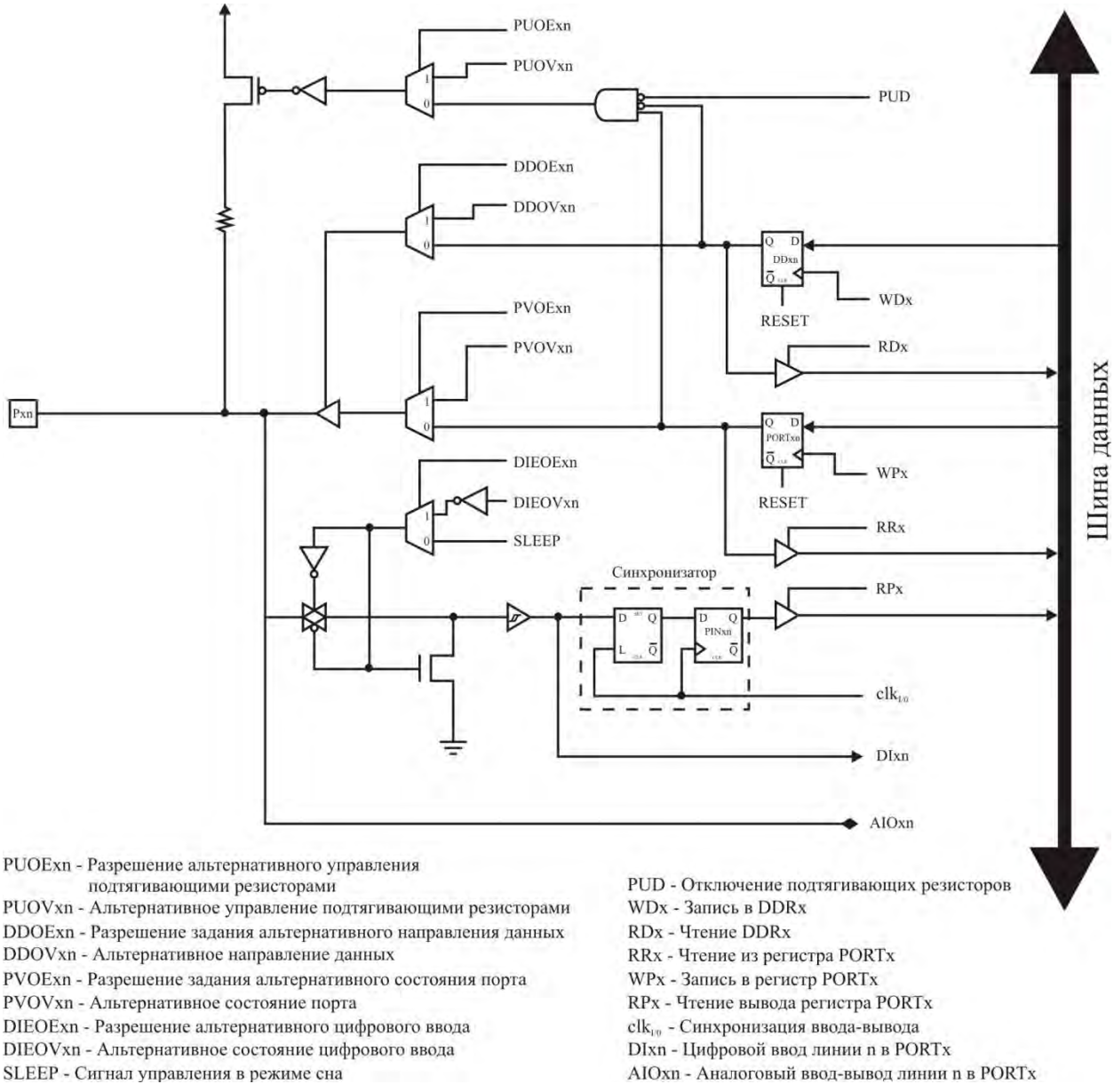


Рисунок 3.34 – Альтернативные функции порта

Примечание – Сигналы WPx, WDx, RRx, RPx и RDx являются общими для всех линий в пределах одного порта. Сигналы $clk_{I/O}$, SLEEP и PUD являются общими для всех портов. Все остальные сигналы индивидуальны для каждого вывода.

Таблица 3.26 – Общее описание сигналов переопределения функций для активизации альтернативных функций

Обозначение сигнала	Полное наименование	Описание
PUOE	Разрешение альтернативного управления подтягивающими резисторами	Если данный сигнал установлен, то подключение подтягивающего резистора определяется значением сигнала PUOV. Если данный сигнал сброшен, то подтягивающий резистор подключается, когда {DDxn, PORTxn, PUD} = 0b010
PUOV	Альтернативное управление подтягивающими резисторами	Если PUOE установлен, то подтягивающий резистор подключается/отключается, когда PUOV установлен/сброшен независимо от состояния регистровых битов DDxn, PORTxn и PUD

Продолжение таблицы 3.26

Обозначение сигнала	Полное наименование	Описание
DDOE	Разрешение задания альтернативного направления данных	Если этот сигнал установлен, то разрешение работы выходного драйвера определяется значением сигнала DDOV. Если этот сигнал сброшен, то работа выходного драйвера разрешается регистровым битом DDxn
DDOV	Альтернативное направление данных	Если DDOE установлен, то работа выходного драйвера разрешается/запрещается, когда DDOV устанавливается/ сбрасывается независимо от состояния регистрового бита DDxn
PVOE	Разрешение задания альтернативного состояния порта	Если данный сигнал установлен и разрешена работа выходного драйвера, то состояние на выходе порта определяется сигналом PVOV. Если PVOE сброшен и разрешена работа выходного драйвера, то состояние на выходе порта определяется регистровым битом PORTxn
PVOV	Альтернативное состояние порта	Если PVOE установлен, то выход порта принимает состояние PVOV независимо от установки регистрового бита PORTxn
DIEOE	Разрешение альтернативного цифрового ввода	Если данный бит установлен, то функция разрешения цифрового ввода передается сигналу DIEOV. Если данный сигнал сброшен, то разрешение цифрового ввода определяется состоянием микроконтроллера (нормальный режим, режимы сна)
DIEOV	Альтернативное состояние цифрового ввода	Если DIEOE установлен, то цифровой ввод разрешен/запрещен, когда DIEOV установлен/сброшен независимо от состояния микроконтроллера (нормальный режим, режимы сна)
DI	Цифровой ввод	Сигнал цифрового ввода для альтернативных функций. На рисунке сигнал подключен к входу триггера Шмитта перед синхронизатором. Если цифровой ввод не используется как источник синхронизации, то модуль с альтернативной функцией будет использовать свой собственный синхронизатор
AIO	Аналоговый ввод-вывод	Сигнал аналогового ввода-вывода к модулю/из модуля с альтернативной функцией. Сигнал подключается непосредственно к контактной площадке и может использоваться двунаправленно

В следующих подпунктах кратко описаны альтернативные функции для каждого порта и связь сигналов переопределения функций с альтернативными функциями выводов.

Регистр специальных функций ввода-вывода – SFIOR

Бит	7	6	5	4	3	2	1	0	
	TSM	-	-	-	ACME	PUD	PSR0	PSR321	SFIOR
Чтение/запись	Ч/З	Ч	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 2 – PUD: Отключение подтягивающих резисторов

Если в данный разряд записать логическую единицу, то подтягивающие резисторы на всех портах будут отключены, даже если регистры DDxn и PORTxn настроены на их подключение ({DDxn, PORTxn} = 0b01). См. выше подпункт «Настройка выводов» для детального изучения данной особенности.

Альтернативные функции порта A

Альтернативной функцией порта A является мультиплексированная младшая шина адреса/данных для интерфейса внешней памяти.

Таблица 3.27 – Альтернативные функции выводов порта A

Вывод порта	Альтернативная функция
PA7	AD7 (Разряд 7 шины адреса и шины данных интерфейса внешней памяти)
PA6	AD6 (Разряд 6 шины адреса и шины данных интерфейса внешней памяти)
PA5	AD5 (Разряд 5 шины адреса и шины данных интерфейса внешней памяти)
PA4	AD4 (Разряд 4 шины адреса и шины данных интерфейса внешней памяти)
PA3	AD3 (Разряд 3 шины адреса и шины данных интерфейса внешней памяти)
PA2	AD2 (Разряд 2 шины адреса и шины данных интерфейса внешней памяти)
PA1	AD1 (Разряд 1 шины адреса и шины данных интерфейса внешней памяти)
PA0	AD0 (Разряд 0 шины адреса и шины данных интерфейса внешней памяти)

В таблицах 3.28 и 3.29 показана связь сигналов переопределения, представленных на рисунке 3.34, и альтернативных функций выводов порта A.

Таблица 3.28 – Сигналы переопределения функций для разрешения альтернативных функций на PA7 – PA4

Наименование сигнала	PA7/AD7	PA6/AD6	PA5/AD5	PA4/AD4
PUOE	SRE	SRE	SRE	SRE
PUOV	~(WR# | ADA) · PORTA7 · PUD#	~(WR# | ADA) · PORTA6 · PUD#	~(WR# | ADA) · PORTA5 · PUD#	~(WR# | ADA) · PORTA4 · PUD#
DDOE	SRE	SRE	SRE	SRE
DDOV	WR# | ADA	WR# | ADA	WR# | ADA	WR# | ADA
PVOE	SRE	SRE	SRE	SRE
PVOV	A7 · ADA | D7 OUTPUT · WR#	A6 · ADA | D6 OUTPUT · WR#	A5 · ADA | D5 OUTPUT · WR#	A4 · ADA | D4 OUTPUT · WR#
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	D7 INPUT	D6 INPUT	D5 INPUT	D4 INPUT
AIO	-	-	-	-
Примечание – ADA – это интервал времени, когда выводятся адресные сигналы (см. также 3.2.5 «Интерфейс внешней памяти»).				

Таблица 3.29 – Сигналы переопределения функций для разрешения альтернативных функций на PA3 – PA0

Наименование сигнала	PA3/AD3	PA2/AD2	PA1/AD1	PA0/AD0
PUOE	SRE	SRE	SRE	SRE
PUOV	~(WR# | ADA) · PORTA3 · PUD#	~(WR#| ADA) · PORTA2 · PUD#	~(WR# | ADA) · PORTA1 · PUD#	~(WR# | ADA) · PORTA0 · PUD#
DDOE	SRE	SRE	SRE	SRE
DDOV	WR# | ADA	WR# | ADA	WR# | ADA	WR# | ADA
PVOE	SRE	SRE	SRE	SRE
PVOV	A3 · ADA | D3 OUTPUT · WR#	A2 · ADA | D2 OUTPUT · WR#	A1 · ADA | D1 OUTPUT · WR#	A0 · ADA | D0 OUTPUT · WR#
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	D3 INPUT	D2 INPUT	D1 INPUT	D0 INPUT
AIO	-	-	-	-
Примечание – ADA – это интервал времени, когда выводятся адресные сигналы (см. также 3.2.5 «Интерфейс внешней памяти»).				

Альтернативные функции порта B

Выводы порта B с альтернативными функциями показаны в таблице 3.30.

Таблица 3.30 – Альтернативные функции выводов порта B

Вывод порта	Альтернативная функция
PB7	OC2/OC1C (выход компаратора и ШИМ-выход таймера/счетчика 2 или выход компаратора и ШИМ-выход C таймера/счетчика 1)
PB6	OC1B (выход компаратора и ШИМ-выход B таймера/счетчика 1)
PB5	OC1A (выход компаратора и ШИМ-выход A таймера/счетчика 1)
PB4	OC0 (Выход компаратора и ШИМ-выход таймера/счетчика 0)
PB3	MISO (Вход ведущего/выход ведомого шины SPI)
PB2	MOSI (Выход ведущего/вход ведомого шины SPI)
PB1	SCK (Синхронизация шины SPI)
PB0	SS#(Вход выбора ведомого интерфейса SPI)

Ниже дано описание альтернативных функций выводов.

OC2/OC1C, разряд 7

OC2 – выход компаратора таймера/счетчика 2. Для выполнения данной функции вывод PB7 конфигурируется как выход (DDB7 = 1). Вывод OC2 также выполняет функцию выхода, когда таймер переводится в режим ШИМ.

OC1C – выход компаратора C таймера/счетчика 1. Для выполнения данной функции вывод PB7 конфигурируется как выход (DDB7 = 1). Вывод OC1C также выполняет функцию выхода, когда таймер переводится в режим ШИМ.

OC1B, разряд 6

OC1B – выход компаратора B таймера/счетчика 1. Для выполнения данной функции вывод PB6 конфигурируется как выход (DDB6 = 1). Вывод OC1B также выполняет функцию выхода, когда таймер переводится в режим ШИМ.

OC1A, разряд 5

OC1A – выход компаратора A таймера/счетчика 1. Для выполнения данной функции вывод PB5 конфигурируется как выход (DDB5 = 1). Вывод OC1A также выполняет функцию выхода, когда таймер переводится в режим ШИМ.

OC0, разряд 4

OC0 – выход компаратора таймера/счетчика 0. Для выполнения данной функции вывод PB4 конфигурируется как выход (DDB4 = 1). Вывод OC0 также выполняет функцию выхода, когда таймер переводится в режим ШИМ.

MISO – порт B, разряд 3

MISO – вход данных в режиме ведущего, выход данных в режиме ведомого интерфейса SPI. Если разрешена работа SPI как ведущего, то данный вывод настраивается на ввод, независимо от состояния DDB3. Если разрешена работа SPI как ведомого, то направление передачи данных задается DDB3. Если вывод принудительно настраивается на ввод данных, то подключение подтягивающего резистора останется под управлением бита PORTB3.

MOSI – порт B, разряд 2

MOSI – выход данных в режиме ведущего, вход данных в режиме ведомого интерфейса SPI. Если разрешена работа SPI как ведомого, то данный вывод настраивается на ввод, независимо от значения DDB2. Если разрешена работа SPI как ведущего, то направление передачи данных определяется DDB2. Если вывод принудительно настраивается на ввод данных, то подключение подтягивающего резистора останется под управлением бита PORTB2.

SCK – порт B, разряд 1

SCK – выход синхронизации в режиме ведущего, вход синхронизации в режиме ведомого интерфейса SPI. Если разрешена работа SPI как ведомого, то данный вывод настраивается на ввод, независимо от состояния DDB1. Если разрешена работа SPI как ведущего, то направление передачи данных задается DDB1. Если вывод принудительно настраивается на ввод данных, то подключение подтягивающего резистора останется под управлением бита PORTB1.

SS# – порт B, разряд 0

SS# – вход выбора ведомого порта. Если разрешена работа SPI как ведомого, то данный вывод настраивается на ввод, независимо от установки DDB0. Работа SPI как ведомого активизируется, если подать низкий уровень на этот вход. Если разрешена работа SPI как ведущего, то направление передачи данных на этом выводе задается DDB0. Если вывод принудительно настраивается на ввод данных, то подключение подтягивающего резистора останется под управлением бита PORTB0.

В таблицах 3.31 и 3.32 показаны значения сигналов переопределения (см. рисунок 3.34) в различных альтернативных функциях порта B. SPI MSTR INPUT (вход ведущего SPI) и SPI SLAVE OUTPUT (выход ведомого SPI) составляют сигнал MISO, а сигнал MOSI разделен на SPI MSTR OUTPUT (выход ведущего SPI) и SPI SLAVE INPUT (вход ведомого SPI).

Таблица 3.31 – Сигналы переопределения функций для разрешения альтернативных функций на PB7 – PB4

Наименование сигнала	PB7/OC2/OC1C	PB6/OC1B	PB5/OC1A	PB4/OC0
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	Разрешение OC2/OC1C*	Разрешение OC1B	Разрешение OC1A	Разрешение OC0
PVOV	OC2/OC1C*	OC1B	OC1A	OC0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	-	-	-	-

_____

* См. подраздел 3.13 «Модулятор сравнения (OCM1C2)» для получения подробного описания.

Таблица 3.32 – Сигналы переопределения функций для разрешения альтернативных функций на PB3 – PB0

Наименование сигнала	PB3/MISO	PB2/MOSI	PB1/SCK	PB0/SS#
PUOE	SPE · MSTR	SPE · MSTR#	SPE · MSTR#	SPE · MSTR#
PUOV	PORTB3 · PUD#	PORTB2 · PUD#	PORTB1 · PUD#	PORTB0 · PUD#
DDOE	SPE · MSTR	SPE · MSTR#	SPE · MSTR#	SPE · MSTR#
DDOV	0	0	0	0
PVOE	SPE · MSTR#	SPE · MSTR	SPE · MSTR	0
PVOV	SPI SLAVE OUTPUT	SPI MSTR OUTPUT	SCK OUTPUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SPI MSTR INPUT	SPI SLAVE INPUT	SCK INPUT	SPI SS#
AIO	-	-	-	-

Альтернативные функции порта C

Альтернативной функцией порта C является шина старшего адреса интерфейса внешней памяти.

Таблица 3.33 – Альтернативные функции выводов порта C

Вывод порта	Альтернативная функция
PC7	A15 (Разряд 15 шины адреса интерфейса внешней памяти)
PC6	A14 (Разряд 14 шины адреса интерфейса внешней памяти)
PC5	A13 (Разряд 13 шины адреса интерфейса внешней памяти)
PC4	A12 (Разряд 12 шины адреса интерфейса внешней памяти)
PC3	A11 (Разряд 11 шины адреса интерфейса внешней памяти)
PC2	A10 (Разряд 10 шины адреса интерфейса внешней памяти)
PC1	A9 (Разряд 9 шины адреса интерфейса внешней памяти)
PC0	A8 (Разряд 8 шины адреса интерфейса внешней памяти)

В таблицах 3.34 и 3.35 показывается связь между альтернативными функциями выводов порта и сигналами переопределения, приведенными на рисунке 3.34.

Таблица 3.34 – Сигналы переопределения функций для разрешения альтернативных функций на PC7 – PC4

Наименование сигнала	PC7/A15	PC6/A14	PC5/A13	PC4/A12
PUOE	SRE · (XMM<1)	SRE · (XMM<2)	SRE · (XMM<3)	SRE · (XMM<4)
PUOV	0	0	0	0
DDOE	SRE · (XMM<1)	SRE · (XMM<2)	SRE · (XMM<3)	SRE · (XMM<4)
DDOV	1	1	1	1
PVOE	SRE · (XMM<1)	SRE · (XMM<2)	SRE · (XMM<3)	SRE · (XMM<4)
PVOV	A15	A14	A13	A12
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	-	-	-	-

Таблица 3.35 – Сигналы переопределения функций для разрешения альтернативных функций на PC3 – PC0

Наименование сигнала	PC3/A11	PC2/A10	PC1/A9	PC0/A8
PUOE	SRE · (XMM<5)	SRE · (XMM<6)	SRE · (XMM<7)	SRE · (XMM<8)
PUOV	0	0	0	0
DDOE	SRE · (XMM<5)	SRE · (XMM<6)	SRE · (XMM<7)	SRE · (XMM<8)
DDOV	1	1	1	1
PVOE	SRE · (XMM<5)	SRE · (XMM<6)	SRE · (XMM<7)	SRE · (XMM<8)
PVOV	A11	A10	A9	A8
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	-	-	-	-

Альтернативные функции порта D

Выводы порта D с альтернативными функциями представлены в таблице 3.36.

Таблица 3.36 – Альтернативные функции выводов порта D

Вывод порта	Альтернативная функция
PD7	T2 (вход синхронизации таймера/счетчика 2)
PD6	T1 (вход синхронизации таймера/счетчика 1)
PD5	XCK1 (вход/выход внешней синхронизации USART 1)
PD4	ICP1 (вход события таймера/счетчика 1)
PD3	INT3/TXD1 (вход внешнего прерывания 3 или вывод передачи USART 1)
PD2	INT2/RXD1 (вход внешнего прерывания 2 или вывод приема USART 1)
PD1	INT1/SDA (вход внешнего прерывания 1 или ввод/вывод последовательных данных TWI)
PD0	INT0/SCL (вход внешнего прерывания 0 или синхронизация последовательной связи TWI)

T2 – порт D, разряд 7

T2 – счетный вход таймера/счетчика 2.

T1 – порт D, разряд 6

T1 – счетный вход таймера/счетчика 1.

XCK1 – порт D, разряд 5

XCK1 – внешняя синхронизация USART 1. Регистр направления данных (DDD4) определяет, является ли синхронизация выходной (DDD4 = 1) или входной (DDD4 = 0). Вывод XCK1 активен, только когда USART 1 работает в синхронном режиме.

ICP1 – порт D, разряд 4

ICP1 – вход события таймера/счетчика 1.

INT3/TXD1 – порт D, разряд 3

INT3 – источник внешнего прерывания 3. Вывод PD3 может служить источником внешнего прерывания микроконтроллера.

TXD1 – передача данных (вывод данных для USART 1). Если разрешена работа передатчика USART 1, то данный вывод настраивается как выход независимо от значения DDD3.

INT2/RXD1 – порт D, разряд 2

INT2 – источник внешнего прерывания 2. Вывод PD2 может служить источником внешнего прерывания микроконтроллера.

RXD1 – прием данных (ввод данных для USART 1). Если разрешена работа приемника USART 1, то данный вывод настраивается на ввод независимо от значения DDD2. Если USART 1 настраивает данный вывод на ввод данных, то подключение подтягивающего резистора останется под управлением бита PORTD2.

INT1/SDA – порт D, разряд 1

INT1 – источник внешнего прерывания 1. Вывод PD1 может служить источником внешнего прерывания микроконтроллера.

SDA – ввод-вывод данных двухпроводного последовательного интерфейса TWI. После установки бита TWEN в регистре TWCR разрешается работа TWI, вывод PD1 отключается от порта и становится линией ввода-вывода последовательных данных для TWI. В этом режиме на входе активизируется помехоподавляющий фильтр, который пресекает входные импульсы длительностью менее 50 нс, а передача организована драйвером с открытым стоком и ограниченной скоростью изменения сигнала.

INT0/SCL – порт D, разряд 0

INT0 – источник внешнего прерывания 0. Вывод PD0 может служить источником внешнего прерывания микроконтроллера.

SCL – синхронизация двухпроводного последовательного интерфейса TWI. После установки бита TWEN в регистре TWCR разрешается работа TWI, вывод PD0 отключается от порта и становится входом/выходом синхронизации последовательной связи для TWI. В этом режиме на входе активизируется помехоподавляющий фильтр, который пресекает входные импульсы длительностью менее 50 нс, а передача организована драйвером с открытым стоком и ограниченной скоростью изменения сигнала.

В таблицах 3.37 и 3.38 показывается связь между альтернативными функциями выводов порта D и сигналами переопределения функций, приведенными на рисунке 3.34.

Таблица 3.37 – Сигналы переопределения функций для разрешения альтернативных функций на PD7 – PD4

Наименование сигнала	PD7/T2	PD6/T1	PD5/XCK1	PD4/ICP1
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	UMSEL1	0
PVOV	0	0	XCK1 OUTPUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	T2 INPUT	T1 INPUT	XCK1 INPUT	ICP1 INPUT
AIO	-	-	-	-

Таблица 3.38 – Сигналы переопределения функций для разрешения альтернативных функций на PD3 – PD0

Наименование сигнала	PD3/INT3/TXD1	PD2/INT2/RXD1	PD1/INT1/SDA	PD0/INT0/SCL
PUOE	TXEN1	RXEN1	TWEN	TWEN
PUOV	0	PORTD2 · PUD#	PORTD1 · PUD#	PORTD0 · PUD#
DDOE	TXEN1	RXEN1	TWEN	TWEN
DDOV	1	0	SDA_OUT	SCL_OUT
PVOE	TXEN1	0	TWEN	TWEN
PVOV	TXD1	0	0	0
DIEOE	Разрешение INT3	Разрешение INT2	Разрешение INT1	Разрешение INT0
DIEOV	1	1	1	1
DI	INT3 INPUT	INT2 INPUT/RXD1	INT1 INPUT	INT0 INPUT
AIO	-	-	SDA INPUT	SCL INPUT

Примечание – После разрешения работы TWI активизируется схема управления скоростью изменения выходных сигналов на выводах PD0 и PD1. Данная функция не учтена в таблице. Кроме того, помехоподавляющий фильтр подключается между выходом AIO (см. рисунок 3.34) и цифровой логикой модуля TWI.

Альтернативные функции порта E

Альтернативные функции порта E представлены в таблице 3.39.

Таблица 3.39 – Альтернативные функции выводов порта E

Вывод порта	Альтернативная функция
PE7	INT7/ICP3 (вход внешнего прерывания 7 или вход события таймера/счетчика 3)
PE6	INT6/T3 (вход внешнего прерывания 6 или вход синхронизации таймера/счетчика 3)
PE5	INT5/OC3C (вход внешнего прерывания 5 или выход C компаратора и ШИМ таймера/счетчика 3)
PE4	INT4/OC3B (вход внешнего прерывания 4 или выход B компаратора и ШИМ таймера/счетчика 3)
PE3	AIN1/OC3A (инвертирующий вход аналогового компаратора или выход A компаратора и ШИМ таймера/счетчика 3)
PE2	AIN0/XCK0 (неинвертирующий вход аналогового компаратора или вход/выход внешней синхронизации USART 0)
PE1	PDO/TXD0 (выход программируемых данных или выход передатчика USART 0)
PE0	PDI/RXD0 (вход программируемых данных или вход приемника USART 0)

INT7/ICP3 – Порт E, разряд 7

INT7 – источник внешнего прерывания 7. Вывод PE7 может служить источником внешнего прерывания микроконтроллера.

ICP3 – вход события таймера/счетчика 3.

INT6/T3 – Порт E, разряд 6

INT6 – источник внешнего прерывания 6. Вывод PE6 может служить источником внешнего прерывания микроконтроллера.

T3 – счетный вход таймера/счетчика 3.

INT5/OC3C – Порт E, разряд 5

INT5 – источник внешнего прерывания 5. Вывод PE5 может служить источником внешнего прерывания микроконтроллера.

OC3C – выход компаратора C таймера/счетчика 3. Для выполнения данной функции вывод должен быть настроен как выход (DDE5 = 1). Вывод OC3C также выполняет функцию выхода, когда таймер переведен в режим ШИМ.

INT4/OC3B – Порт E, разряд 4

INT4 – источник внешнего прерывания 4 микроконтроллера.

OC3B – выход компаратора B таймера/счетчика 3. Для выполнения данной функции вывод должен быть настроен как выход (DDE4 = 1). Вывод OC3B также выполняет функцию выхода, когда таймер переведен в режим ШИМ.

AIN1/OC3A – Порт E, разряд 3

AIN1 – инвертирующий вход аналогового компаратора. Данный вывод непосредственно подключен к инвертирующему входу аналогового компаратора.

OC3A – выход компаратора A таймера/счетчика 3. Для выполнения данной функции вывод должен быть настроен как выход (DDE3 = 1). Вывод OC3A также выполняет функцию выхода, когда таймер переведен в режим ШИМ.

AIN0/XCK0 – Порт E, разряд 2

AIN0 – неинвертирующий вход аналогового компаратора. Данный вывод непосредственно подключен к неинвертирующему входу аналогового компаратора.

XCK0 – внешняя синхронизация USART 0. Регистр направления данных (DDE2) определяет, является ли синхронизация выходной (DDE2 = 1) или входной (DDE2 = 0). Вывод XCK0 активен, только когда USART 0 работает в синхронном режиме.

PDO/TXD0 – Порт E, разряд 1

PDO – вывод последовательно программируемых через SPI интерфейс данных. В процессе последовательного программирования этот вывод используется как линия вывода данных из микроконтроллера.

TXD0 – вывод передачи данных USART 0.

PDI/RXD0 – Порт E, разряд 0

PDI – ввод последовательно программируемых через SPI данных. В процессе последовательного программирования данный вывод используется как линия ввода данных в микроконтроллер.

RXD0 – вывод приема данных USART 0. Если разрешена работа приемника USART 0, то данный вывод настраивается как вход независимо от состояния DDRE0. Если USART 0 принудительно настраивает данный вывод на ввод данных, то запись логической единицы в PORTE0 включит подтягивающий резистор на данном выводе.

В таблицах 3.40 и 3.41 описывается связь альтернативных функций выводов порта E и сигналов переопределения функций, представленных на рисунке 3.34.

Таблица 3.40 – Сигналы переопределения функций для разрешения альтернативных функций на PE7 – PE4

Наименование сигнала	PE7/INT7/ICP3	PE6/INT6/T3	PE5/INT5/OC3C	PE4/INT4/OC3B
PUOE	0	0	0	0
PUOV	0	0	0	0
DDOE	0	0	0	0
DDOV	0	0	0	0
PVOE	0	0	Разрешение OC3C	Разрешение OC3B
PVOV	0	0	OC3C	OC3B
DIEOE	Разрешение INT7	Разрешение INT6	Разрешение INT5	Разрешение INT4
DIEOV	1	1	1	1
DI	INT7 INPUT/ICP3 INPUT	INT6 INPUT/T3 INPUT	INT5 INPUT	INT4 INPUT
AIO	-	-	-	-

Таблица 3.41 – Сигналы переопределения функций для разрешения альтернативных функций на PE3 – PE0

Наименование сигнала	PE3/AIN1/OC3A	PE2/AIN0/XCK0	PE1/PDO/TXD0	PE0/PDI/RXD0
PUOE	0	0	TXEN0	RXEN0
PUOV	0	0	0	PORTE0 · PUD#
DDOE	0	0	TXEN0	RXEN0
DDOV	0	0	1	0
PVOE	Разрешение OC3A	UMSEL0	TXEN0	0
PVOV	OC3A	XCK0 OUTPUT	TXD0	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	0	XCK0 INPUT	-	RXD0
AIO	AIN1 INPUT	AIN0 INPUT	-	-

Альтернативные функции порта F

Альтернативной функцией порта F является аналоговый вход АЦП (см. таблицу 3.42). Если некоторые выводы порта F сконфигурированы как выходы, то необходимо следить, чтобы во время преобразования АЦП не происходило их переключение. Иначе результат преобразования может быть некорректным. Если разрешена работа интерфейса JTAG, то подтягивающие резисторы на выводах PF7(TDI), PF5(TMS) и PF4(TCK) останутся подключенными, даже если произойдет сброс микроконтроллера.

Таблица 3.42 – Альтернативные функции выводов порта F

Вывод порта	Альтернативная функция
PF7	ADC7/TDI (Входной канал 7 АЦП или вход данных интерфейса JTAG)
PF6	ADC6/TDO (Входной канал 6 АЦП или выход данных интерфейса JTAG)
PF5	ADC5/TMS (Входной канал 5 АЦП или выбор режима интерфейса JTAG)
PF4	ADC4/TCK (Входной канал 4 АЦП или синхронизация интерфейса JTAG)
PF3	ADC3 (Входной канал 3 АЦП)
PF2	ADC2 (Входной канал 2 АЦП)
PF1	ADC1 (Входной канал 1 АЦП)
PF0	ADC0 (Входной канал 0 АЦП)

TDI, ADC7 – Порт F, разряд 7

ADC7 – Аналого-цифровой преобразователь, канал 7.

TDI – вход данных интерфейса JTAG. Последовательный ввод данных происходит в регистр инструкций или регистр данных (цепи сканирования). После разрешения работы JTAG-интерфейса данный вывод не может использоваться в качестве линии ввода-вывода.

TDO, ADC6 – Порт F, разряд 6

ADC6 – Аналого-цифровой преобразователь, канал 6.

TDO – выход данных интерфейса JTAG. Последовательный вывод данных из регистра инструкций или регистра данных. После разрешения работы JTAG-интерфейса данный вывод не может использоваться в качестве линии ввода-вывода. Вывод TDO будет оставаться в состоянии высокого импеданса, пока TAP-контроллер не начнет выдавать данные.

TMS, ADC5 – Порт F, разряд 5

ADC5 – Аналого-цифровой преобразователь, канал 5.

TMS – выбор режима интерфейса JTAG. Данный вывод используется для управления автоматом конечных состояний TAP-контроллера. После разрешения работы JTAG-интерфейса данный вывод не может использоваться в качестве линии ввода-вывода.

TCK, ADC4 – Порт F, разряд 4

ADC4 – Аналого-цифровой преобразователь, канал 4.

TCK – синхронизация интерфейса JTAG. Работа JTAG-интерфейса синхронизируется в соответствии с сигналом TCK. После разрешения работы JTAG-интерфейса данный вывод не может использоваться в качестве линии ввода-вывода.

ADC3 – ADC0 – Порт F, разряды 3 – 0

ADC7 – Аналого-цифровой преобразователь, каналы 3 – 0.

Таблица 3.43 – Сигналы переопределения функций для разрешения альтернативных функций на PF7 – PF4

Наименование сигнала	PF7/ADC7/TDI	PF6/ADC6/TDO	PF5/ADC5/TMS	PF4/ADC4/TCK
PUOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
PUOV	1	0	1	1
DDOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
DDOV	0	SHIFT_IR + SHIFT_DR	0	0
PVOE	0	JTAGEN	0	0
PVOV	0	TDO	0	0
DIEOE	JTAGEN	JTAGEN	JTAGEN	JTAGEN
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	TDI/ADC7 INPUT	ADC6 INPUT	TMS/ADC5 INPUT	TCK/ADC4 INPUT

Таблица 3.44 – Сигналы переопределения функций для разрешения альтернативных функций на PF3 – PF0

Наименование сигнала	PF3/ADC3	PF2/ADC2	PF1/ADC1	PF0/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 INPUT	ADC2 INPUT	ADC1 INPUT	ADC0 INPUT

Альтернативные функции порта G

В таблице 3.45 приведены альтернативные функции порта G.

Таблица 3.45 – Альтернативные функции выводов порта G

Вывод порта	Альтернативная функция
PG4	TOSC1 (Вывод осциллятора счетчика реального времени таймера/счетчика 0)
PG3	TOSC2 (Вывод осциллятора счетчика реального времени таймера/счетчика 0)
PG2	ALE (Разрешение фиксации адреса внешней памяти)
PG1	RD# (Сигнал чтения из внешней памяти)
PG0	WR# (Сигнал записи во внешнюю память)

TOSC1 – Порт G, разряд 4

TOSC1 – вывод 1 осциллятора таймера. После установки бита AS0 в регистре ASSR разрешается работа асинхронного тактирования таймера/счетчика 0, а вывод PG4 отключается от порта и становится входом инвертирующего усилителя осциллятора. В этом режиме кварцевый генератор подключен к выводу PG4, который теперь не может использоваться как линия ввода-вывода.

TOSC2 – Порт G, разряд 3

TOSC2 – вывод 2 осциллятора таймера. После установки бита AS0 в регистре ASSR разрешается работа асинхронного тактирования таймера/счетчика 0, а вывод PG3 отключается от порта и становится выходом инвертирующего усилителя осциллятора. В этом режиме кварцевый генератор подключен к выводу PG3, который теперь не может использоваться как линия ввода-вывода.

ALE – Порт G, разряд 2

ALE – сигнал разрешения фиксации адреса во внешней памяти данных.

RD# – Порт G, разряд 1

RD# – сигнал управления чтением из внешней памяти данных.

WR# – Порт G, разряд 0

WR# – сигнал управления записью во внешнюю память данных.

В таблице 3.46 представлена связь альтернативных функций порта G и сигналов переопределения функций, приведенных на рисунке 3.34.

Таблица 3.46 – Сигналы переопределения функций для разрешения альтернативных функций на PG4 – PG1

Наименование сигнала	PG4/TOSC1	PG3/TOSC2	PG2/ALE	PG1/RD#	PG0/WR#
PUOE	AS0	AS0	SRE	SRE	SRE
PUOV	0	0	0	0	0
DDOE	AS0	AS0	SRE	SRE	SRE
DDOV	0	0	1	1	1
PVOE	0	0	SRE	SRE	SRE
PVOV	0	0	ALE	RD#	WR#
DIEOE	AS0	AS0	0	0	0
DIEOV	0	0	0	0	0
DI	-	-	-	-	-
AIO	T/C0 OSC INPUT	T/C0 OSC OUTPUT	-	-	-

3.7.3 Описание регистров для портов ввода-вывода

Регистр данных порта A – PORTA

Бит	7	6	5	4	3	2	1	0
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Регистр направления данных порта A – DDRA

Бит	7	6	5	4	3	2	1	0
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Адрес входов порта A – PINA

Бит	7	6	5	4	3	2	1	0
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
Чтение/ Запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч
Начальное значение	-	-	-	-	-	-	-	-

Регистр данных порта B – PORTB

Бит	7	6	5	4	3	2	1	0
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Регистр направления данных порта B – DDRB

Бит	7	6	5	4	3	2	1	0
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Адрес входов порта B – PINB

Бит	7	6	5	4	3	2	1	0
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
Чтение/ Запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч
Начальное значение	-	-	-	-	-	-	-	-

Регистр данных порта C – PORTC

Бит	7	6	5	4	3	2	1	0
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Регистр направления данных порта C – DDRC

Бит	7	6	5	4	3	2	1	0
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Адрес входов порта C – PINC

Бит	7	6	5	4	3	2	1	0
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
Чтение/ Запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч
Начальное значение	-	-	-	-	-	-	-	-

Регистр данных порта D – PORTD

Бит	7	6	5	4	3	2	1	0
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Регистр направления данных порта D – DDRD

Бит	7	6	5	4	3	2	1	0
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Адрес входов порта D – PIND

Бит	7	6	5	4	3	2	1	0
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
Чтение/ Запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч
Начальное значение	-	-	-	-	-	-	-	-

Регистр данных порта E – PORTE

Бит	7	6	5	4	3	2	1	0
	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Регистр направления данных порта E – DDRE

Бит	7	6	5	4	3	2	1	0
	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Адрес входов порта E – PINE

Бит	7	6	5	4	3	2	1	0
	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0
Чтение/ Запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч
Начальное значение	-	-	-	-	-	-	-	-

Регистр данных порта F – PORTF

Бит	7	6	5	4	3	2	1	0
	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Регистр направления данных порта F – DDRF

Бит	7	6	5	4	3	2	1	0
	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Адрес входов порта F – PINF

Бит	7	6	5	4	3	2	1	0
	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0
Чтение/ Запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч
Начальное значение	-	-	-	-	-	-	-	-

Регистр данных порта G – PORTG

Бит	7	6	5	4	3	2	1	0
	-	-	-	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0
Чтение/ Запись	Ч	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Регистр направления данных порта G – DDRG

Бит	7	6	5	4	3	2	1	0
	-	-	-	DDG4	DDG3	DDG2	DDG1	DDG0
Чтение/ Запись	Ч	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Адрес входов порта G – PING

Бит	7	6	5	4	3	2	1	0
	-	-	-	PING4	PING3	PING2	PING1	PING0
Чтение/ Запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч
Начальное значение	0	0	0	-	-	-	-	-

3.8 Внешние прерывания

Выводы INT7–0 генерируют внешние прерывания. После разрешения внешние прерывания будут генерироваться, даже если линии INT7–0 настроены как выходы. Данная особенность может использоваться для генерации программного прерывания. Внешние прерывания могут генерироваться по спадающему или нарастающему фронту, а также по низкому логическому уровню. Одна из этих установок задается в регистрах управления внешними прерываниями EICRA (INT3–0) и EICRB (INT7–4). Если внешнее прерывание разрешено и настроено на срабатывание при низком логическом уровне, то прерывание будет инициироваться постоянно, пока на выводе будет оставаться низкий уровень. Для распознавания спадающего или нарастающего фронтов на INT7–4 необходимо наличие

синхронизации ввода-вывода, описанной в 3.3.1 «Системы синхронизации микроконтроллера». Прерывания по низкому уровню и фронтам на INT3–0 определяются асинхронно. Это означает, что данные прерывания могут использоваться для пробуждения микроконтроллера из режимов глубокого сна. Синхронизация ввода-вывода останавливается во всех спящих режимах, за исключением режима холостого хода.

Обратите внимание, что при использовании прерывания по уровню для пробуждения микроконтроллера из режима хранения, необходимо удерживать измененный уровень в течение некоторого времени. Это делает микроконтроллер менее чувствительным к шумам. Оценка изменения состояния уровня выполняется по двум его выборкам с интервалом, соответствующим периоду сторожевого таймера, который равен 2 мкс (номинальное значение) при 5,0 В и 25 °С. Частота сторожевого таймера зависит от напряжения. Пробуждение микроконтроллера наступает, когда на входе присутствует требуемый уровень в процессе выборок или если он удерживается до окончания времени запуска. Время запуска определяется конфигурационными битами SUT (см. 3.3.1 «Системы синхронизации микроконтроллера»). Если дважды была выполнена выборка низкого логического уровня, но этот уровень исчез до окончания времени запуска, то пробуждение микроконтроллера наступит, но прерывание не будет сгенерировано. Для активизации прерывания по уровню необходимо, чтобы этот уровень удерживался в течение времени, достаточного для пробуждения микроконтроллера.

3.8.1 Регистр А управления внешними прерываниями – EICRA

Бит	7	6	5	4	3	2	1	0	
	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7–0: Биты выбора условия генерации внешнего прерывания 3–0

Внешние прерывания 3–0 активируются через внешние выводы INT3–0, если установлены флаг I в регистре статуса SREG и соответствующая маска прерывания в регистре EIMSK. Выбор уровня или фронта для активизации внешнего прерывания осуществляется в соответствии с таблицей 3.47. Фронты сигналов на INT3 – INT0 регистрируются асинхронно. Прерывание по выводам INT3–0 генерируется, если длительность импульса будет больше минимально необходимой (см. таблицу 3.48). При возникновении импульсов меньшей длительности генерация прерывания не гарантируется. Если выбрано прерывание по низкому уровню, то для генерации прерывания необходимо, чтобы этот уровень оставался на прежнем низком уровне до момента завершения выполнения текущей инструкции. После разрешения прерывания по уровню оно будет генерироваться непрерывно до тех пор, пока на входе присутствует низкий уровень.

Таблица 3.47 – Задание условия генерации запроса на прерывание

ISCn1	ISCn0	Описание
0	0	Низкий уровень на INTn генерирует запрос на прерывание
0	1	Зарезервировано
1	0	Спадающий фронт на INTn генерирует запрос на асинхронное прерывание
1	1	Нарастающий фронт на INTn генерирует запрос на асинхронное прерывание

Примечание – n = 3, 2, 1 или 0. Перед изменением битов ISCn1/ISCn0 необходимо запретить работу прерывания путем очистки соответствующего бита разрешения прерывания в регистре EIMSK. В противном случае может возникнуть прерывание после изменения этих битов.

При изменении бита ISCn может возникнуть прерывание. Поэтому рекомендуется вначале отключить прерывание INTn путем сброса соответствующего бита разрешения прерывания в регистре EIMSK. После этого значение бита ISCn может быть изменено. И, наконец, перед возобновлением работы прерываний необходимо сбросить флаг прерывания INTn путем записи логической единицы во флаг прерывания (INTFn) в регистре EIFR.

Таблица 3.48 – Характеристики асинхронного внешнего прерывания

Обозначение	Параметр	Типичное значение	Единица измерения
t_{INT}	Минимальная длительность импульса для генерации асинхронного внешнего прерывания	100	нс

3.8.2 Регистр B управления внешними прерываниями – EICRB

Бит	7	6	5	4	3	2	1	0	
	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7–0: Биты выбора условия генерации внешнего прерывания 7–4

Внешние прерывания 7–4 активируются через внешние выводы INT7–4, если установлены флаг I в регистре статуса SREG и соответствующая маска прерывания в регистре EIMSK. Условие, по которому генерируется прерывание, выбирается исходя из данных таблицы 3.49. Для определения фронтов на выводах INT7–4 осуществляется выборка их состояний. Если выбрано прерывание по фронту или изменению уровня, то импульс, длительность которого больше одного периода синхронизации, генерирует прерывание. При действии на входе более коротких импульсов, генерация прерывания не гарантируется. Обратите внимание, что частота синхронизации ЦПУ может быть ниже, чем частота BQ, если разрешена работа делителя частоты BQ. Если выбрано прерывание по низкому уровню, то этот уровень должен удерживаться до окончания выполнения текущей команды, чтобы сгенерировалось прерывание. Если разрешено прерывание по уровню, то оно будет генерироваться непрерывно, пока на входе присутствует низкий уровень.

Таблица 3.49 – Задание условия генерации запроса на прерывание

ISCn1	ISCn0	Описание
0	0	Низкий уровень на INTn генерирует запрос на прерывание
0	1	Любое изменение логического состояния на INTn генерирует запрос на прерывание
1	0	Спадающий фронт между двумя выборками на INTn генерирует запрос на прерывание
1	1	Нарастающий фронт между двумя выборками на INTn генерирует запрос на прерывание
Примечание – n = 7, 6, 5 или 4. Перед изменением битов ISCn1/ISCn0 необходимо запретить работу прерывания путем очистки соответствующего бита разрешения прерывания в регистре EIMSK. В противном случае может возникнуть прерывание после изменения этих битов.		

3.8.3 Регистр маски внешнего прерывания – EIMSK

Бит	7	6	5	4	3	2	1	0	
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
Чтение/Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7–0: Разрешение запроса на внешнее прерывание 7–0

Когда в бит INT7 – INT0 записана логическая единица и установлен бит I регистра статуса SREG, то разрешается работа внешнего прерывания по соответствующему выводу. Биты выбора условия генерации прерывания в регистрах управления внешними прерываниями EICRA и EICRB определяют, по какому условию генерируется прерывание: по нарастающему фронту, по спадающему фронту или по уровню. Активность на любом из выводов инициирует запрос на прерывание, даже если вывод настроен как выход. Данная особенность может использоваться для генерации программного прерывания.

3.8.4 Регистр флагов внешних прерываний – EIFR

Бит	7	6	5	4	3	2	1	0	
	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0	EIFR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7–0: Флаги внешних прерываний 7–0

Если фронт или изменение логического состояния на INT7–0 инициирует запрос на прерывание, то соответствующий флаг INTF7–0 принимает единичное состояние. Если установлены бит I регистра статуса SREG и соответствующий бит разрешения прерывания INT7–0 в регистре EIMSK, то микроконтроллер выполнит переход на вектор прерывания. Флаг сбрасывается аппаратно после выполнения процедуры обработки прерывания. Альтернативно флаг может быть сброшен программно путем записи логической единицы в соответствующий бит. Если выводы INT7–0 настроены на генерацию прерывания по уровню, то флаги постоянно находятся в сброшенном состоянии. Обратите внимание, что при переходе в режим сна с отключенными прерываниями INT3–0 входные буферы этих выводов будут отключены. В свою очередь это может вызвать изменение внутреннего состояния сигналов, которое приведет к установке флагов INTF3–0.

3.9 8-разрядный таймер/счетчик 0 с широтно-импульсной модуляцией и асинхронной работой

Таймер/счетчик 0 представляет собой одноканальный 8-разрядный модуль общего назначения. Основные отличительные особенности:

- Одноканальный счетчик.
- Режим сброса таймера при совпадении (автоматическая перезагрузка).
- Широтно-импульсная модуляция без генерации ложных импульсов при записи нового значения сравнения в OCR0 (двойная буферизация) и с фазовой коррекцией.
- Генератор частоты.
- 10-разрядный предделитель тактовой частоты.
- Генерация прерываний при переполнении и выполнении условия сравнения (TOV0 и OCF0).
- Возможность тактирования от внешнего кварцевого генератора частотой 32 кГц независимо от синхронизации ввода-вывода.

3.9.1 Краткий обзор

Упрощенная структурная схема 8-разрядного таймера/счетчика 0 представлена на рисунке 3.35. Связи с регистрами, к которым осуществляет доступ ЦПУ, в том числе биты ввода-вывода и линии ввода-вывода, показаны жирной линией. Специфические для данного

устройства регистры, расположение и назначение их битов приведены в 3.9.8 «Описание регистров таймера/счетчика 0».

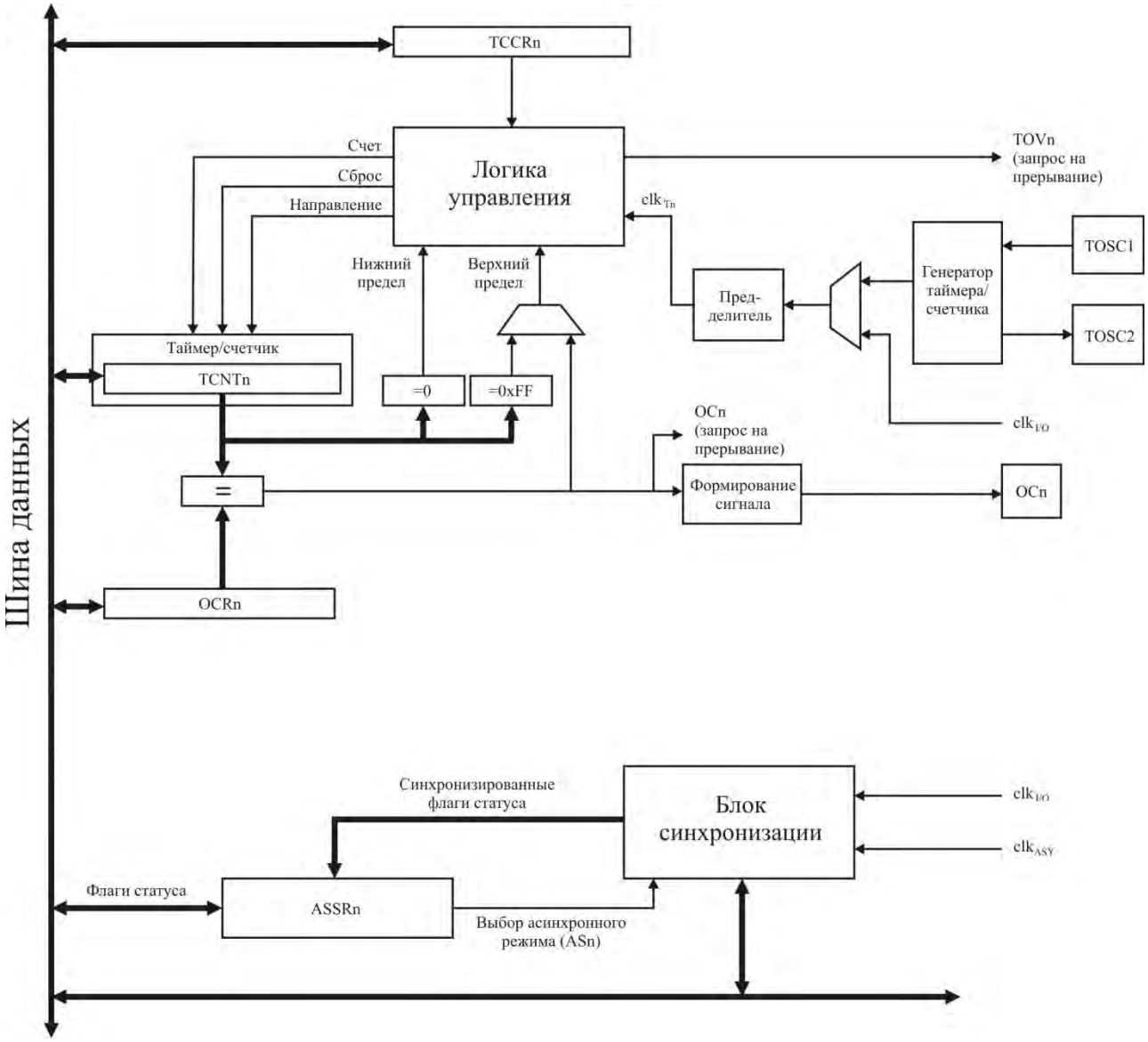


Рисунок 3.35 – Структурная схема 8-разрядного таймера/счетчика 0

Регистры

Счетный регистр таймера/счетчика (TCNT0) и регистр сравнения (OCR0) – это 8-разрядные регистры. Сигналы запроса на прерывание представлены как флаги прерываний таймера в регистре TIFR. Все прерывания индивидуально маскируются с помощью регистра масок прерываний таймеров (TIMSK). Регистры TIFR и TIMSK не представлены на структурной схеме, так как они используются также и другими таймерами микроконтроллера.

Таймер/счетчик может тактироваться, как от внутреннего (через предделитель), так и от внешнего (асинхронно – через выводы TOSC1/2) источника тактового сигнала, что описано в следующих пунктах. Асинхронная работа управляется регистром асинхронного состояния (ASSR). Логический блок выбора тактирования определяет, какой тактовый источник используется для инкрементирования (или декрементирования) значения таймера/счетчика. Если источник тактирования не задан, то таймер/счетчик находится в неактив-

ном состоянии. Выход логики выбора тактирования обозначен как синхронизация таймера (clk_{T0}).

Значение регистра сравнения с двойной буферизацией (OCR0) непрерывно сравнивается со значением таймера/счетчика. Результат сравнения может использоваться формирователем сигналов для генерации импульсов с ШИМ или импульсов переменной частоты на выводе OC0. См. 3.9.4 «Блок сравнения» для изучения деталей. Совпадение значения сравнения со значением таймера/счетчика приводит к установке флага сравнения (OCF0), который может использоваться для генерации запроса на прерывание по результату сравнения.

Определения

Многие обозначения регистров и битов в данном подразделе представлены в общей форме. Латинская строчная буква «n» заменяет номер таймера/счетчика, в данном случае 0. Однако при использовании обозначений регистров или битов в программе необходимо применять точную форму (например, TCNT0 для доступа к значению таймера/счетчика 0 и т.д.).

Определения, приведенные в таблице 3.50, также широко используются на протяжении всего подраздела документа.

Таблица 3.50 – Определения

Нижний предел	Счетчик достигает нижнего предела, когда его значение становится равным нулю (0x00)
Максимум	Счетчик достигает максимума, когда его значение становится равным 0xFF (десятичное число 255)
Верхний предел	Счетчик доходит до верхнего предела, когда он достигает наивысшего значения в последовательности счета. Верхнему пределу может быть присвоено фиксированное значение 0xFF (максимум) или значение, хранящееся в регистре OCR0. Присваивание зависит от режима работы

3.9.2 Источники тактирования таймера/счетчика 0

Таймер/счетчик 0 может тактироваться от внутреннего синхронного или внешнего асинхронного источника тактирования. По умолчанию используется тактовый сигнал clk_{T0}, эквивалентный тактовому сигналу микроконтроллера $clk_{I/O}$. Если в бит AS0 регистра ASSR записать логическую единицу, то в качестве источника синхронизации выступает генератор таймера/счетчика, подключенный к выводам TOSC1 и TOSC2. Более подробно асинхронная работа описана ниже в подпункте «Регистр состояния асинхронного режима – ASSR». См. 3.9.10 «Предделитель таймера/счетчика 0» для получения подробностей об источниках тактирования и предделителе.

3.9.3 Блок счетчика

Главным компонентом 8-разрядного таймера/счетчика 0 является программируемый двунаправленный счетчик. На рисунке 3.36 приведена структурная схема счетчика и окружающих его элементов.

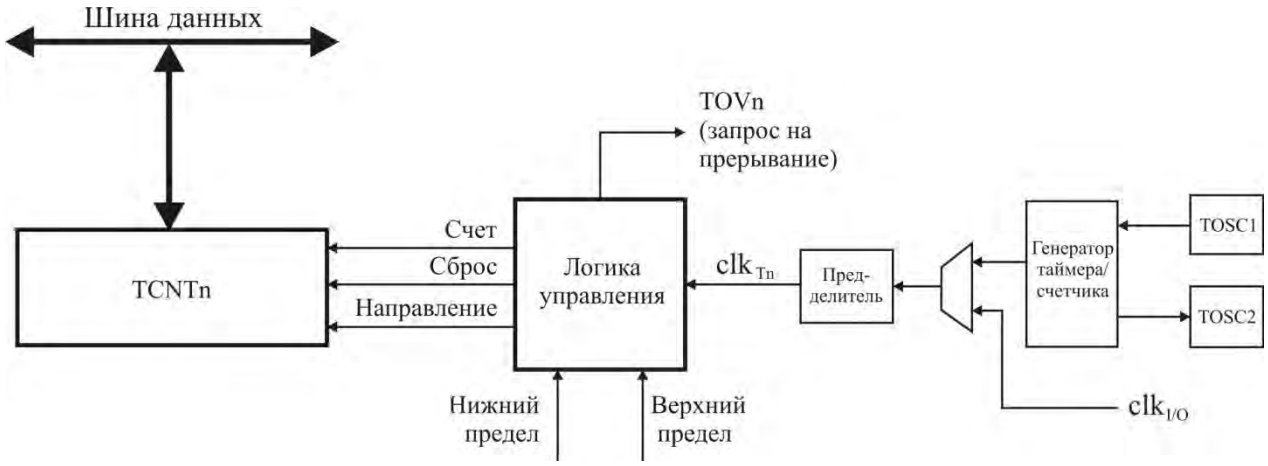


Рисунок 3.36 – Структурная схема блока счетчика

В таблице 3.51 представлено описание внутренних сигналов.

Таблица 3.51 – Описание внутренних сигналов

Наименование сигнала	Описание
Счет	Инкрементирует или декрементирует значение TCNT0 на 1
Направление	Инкремент и декремент
Сброс	Сбрасывает содержимое TCNT0 (устанавливает все разряды в ноль)
clk_{T0}	Синхронизация таймера/счетчика
Верхний предел	Показывает, что содержимое TCNT0 достигло максимального значения
Нижний предел	Показывает, что содержимое TCNT0 достигло минимального значения (нуля)

В зависимости от выбранного режима работы счетчик сбрасывается, инкрементируется или декрементируется на каждом такте синхронизации (clk_{T0}). Тактовый сигнал clk_{T0} может быть выбран между внутренним или внешним источником тактирования, который определяется битами выбора частоты синхронизации (CS02–0). Если источник синхронизации не задан (CS02–0 = 0), то таймер останавливается. Однако значение TCNT0 доступно ЦПУ независимо от того, работает синхронизация таймера или нет. Запись в регистр таймера через ЦПУ перекрывает любые действия самого счетчика (сброс или счет), то есть имеет более высокий приоритет.

Последовательность счета определяется установкой битов режима работы таймера WGM01 и WGM00, расположенных в регистре управления таймером/счетчиком (TCCR0). Существует точная связь между поведением счетчика (алгоритмом счета) и генерируемой формой сигнала на выходе OC0. Более подробно об алгоритмах счета и формировании сигналов написано в 3.9.6 «Режимы работы».

Флаг переполнения таймера/счетчика (TOV0) устанавливается в соответствии с режимом работы, который выбирается битами WGM01 и WGM00. Флаг TOV0 может использоваться для генерации прерывания ЦПУ.

3.9.4 Блок сравнения

8-разрядный цифровой компаратор непрерывно выполняет сравнение содержимого счетного регистра таймера/счетчика TCNT0 с регистром сравнения OCR0. Всякий раз, когда значение TCNT0 совпадает со значением OCR0, компаратор устанавливает флаг совпадения OCF0 на последующем такте синхронизации таймера. Если разрешено прерывание (OCIE0 = 1), то установка флага совпадения вызывает запрос на прерывание. Флаг OCF0

автоматически сбрасывается, когда выполняется процедура обработки прерывания. Альтернативно, флаг OCF0 можно сбросить программно путем записи логической единицы в соответствующий разряд регистра TIFR. Формирователь сигналов использует результат сравнения для генерации выходных импульсов в соответствии с режимом работы, заданным битами WGM01–0 и битами режима формирования выходного сигнала COM01–0. Сигналы «Верхний предел» и «Нижний предел» используются формирователем сигналов для обработки особенных случаев экстремальных значений в некоторых режимах работы (см. 3.9.6 «Режимы работы»). На рисунке 3.37 приведена структурная схема блока сравнения.

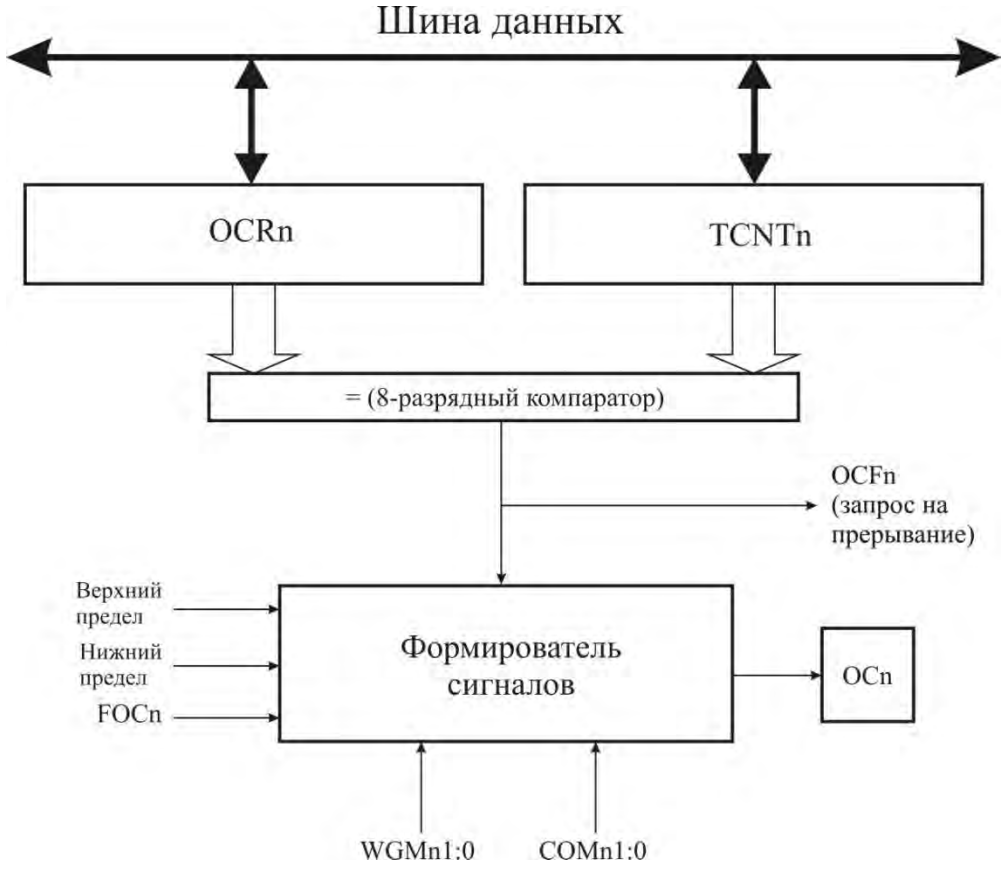


Рисунок 3.37 – Структурная схема блока сравнения

Регистр OCR0 выполнен по схеме двойной буферизации при использовании режимов с широтно-импульсной модуляцией (ШИМ). В нормальном режиме и режиме сброса таймера при совпадении (СТС) схема двойной буферизации отключается. Двойная буферизация позволяет синхронизировать обновление регистра сравнения OCR0 по достижении либо верхнего, либо нижнего предела счета. Такая синхронизация предотвращает возможность возникновения несимметричных ШИМ-сигналов нечетной длины, гарантируя тем самым отсутствие на выходе паразитных импульсов.

Доступ к регистру OCR0 может показаться сложным, но это не так. Если разрешена двойная буферизация, ЦПУ получает доступ к буферному регистру OCR0, а если двойная буферизация отключена, то ЦПУ обращается непосредственно к регистру OCR0.

Установка принудительного совпадения

В режимах формирования сигналов без ШИМ результат сравнения компаратора может быть установлен путем записи логической единицы в бит установки принудительного совпадения FOC0. Установка принудительного совпадения компаратора не приводит к установке флага OCF0 или перезагрузке/сбросу таймера, но обновляет состояние вывода

OC0 (который будет устанавливаться, сбрасываться или переключаться в зависимости от выбранной установки битов COM01–0), как если бы произошло реальное событие совпадения.

Блокирование результата сравнения путем записи в TCNT0

Если ЦПУ осуществляет запись в регистр TCNT0, то результат сравнения будет блокироваться на следующем такте синхронизации таймера, даже если таймер остановлен. Данная функция позволяет устанавливать в регистре OCR0 то же значение, что и в TCNT0, без генерации запроса на прерывание, когда включена синхронизация таймера/счетчика.

Использование блока сравнения

Поскольку запись в TCNT0 блокирует любые действия по сравнению на один такт синхронизации таймера в любом режиме работы, то когда изменяется TCNT0 при использовании канала сравнения (независимо от того, работает таймер/счетчик или нет), необходимо учесть следующие особенности. Если в регистр TCNT0 записано значение, равное значению OCR0, то игнорирование совпадения приведет к генерации сигнала неправильной формы. Аналогично следует избегать записи в TCNT0 значения, равного нижнему пределу (0x00), когда счетчик работает как вычитающий.

Установка OC0 должна быть выполнена перед настройкой линии на вывод в регистре направления данных. Самый легкий путь установки значения OC0 – использование бита установки принудительного совпадения (FOC0) в нормальном режиме. Регистр OC0 сохраняет свое значение, даже если происходит изменение между режимами формирования сигналов.

Учтите, что биты COM01 и COM00 не содержат схемы двойной буферизации и на любые изменения реагируют мгновенно.

3.9.5 Блок формирования выходного сигнала

Биты задания режима формирования выходного сигнала (COM01–0) имеют две функции. С одной стороны, они используются формирователем сигнала и определяют, какое логическое состояние должно быть на выводе OC0 при возникновении следующего совпадения. С другой стороны, биты COM01 и COM00 управляют источником выходного сигнала для вывода OC0. На рисунке 3.38 представлена упрощенная логическая схема, на которой показана функция бит COM01–0. На рисунке показаны только те регистры управления портами ввода-вывода (DDR и PORT), на которые оказывают влияние биты COM01 и COM00. При упоминании состояния OC0 речь идет о внутреннем регистре OC0, а не о выводе OC0.

Функция порта ввода-вывода общего назначения переопределяется функцией выхода формирователя сигнала OC0, если хотя бы один из битов COM01–0 установлен. Однако, направление вывода OC0 (вход или выход) по-прежнему контролируется соответствующим битом регистра направления данных порта. Бит регистра направления данных для вывода OC0 (DDR_OC0) должен быть настроен как выход, прежде чем значение регистра OC0 появится на данном выводе. Управление вводом альтернативной функции не зависит от режима формирования сигналов.

Схемотехника выходной логики позволяет инициализировать состояние регистра OC0 перед разрешением настройки вывода OC0 в качестве выхода. Обратите внимание, что некоторые настройки битов COM01–0 зарезервированы для определенных режимов работы. См. 3.9.8 «Описание регистров таймера/счетчика 0».

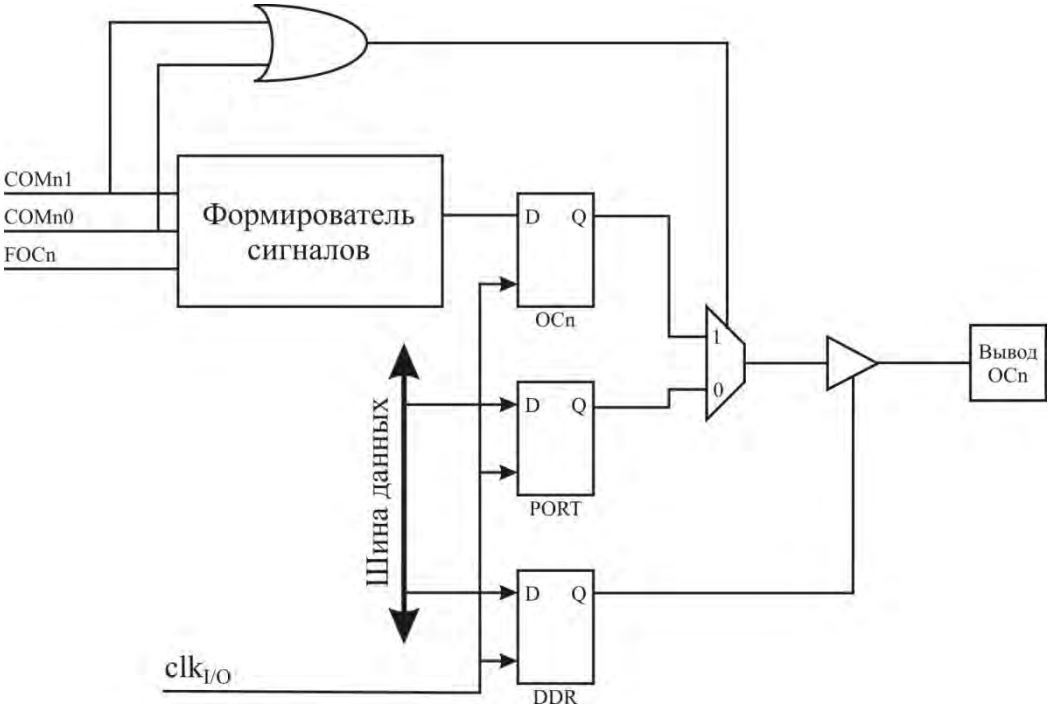


Рисунок 3.38 – Схема блока формирования выходного сигнала

Режим сравнения и формирование сигналов

Формирователь сигналов использует биты COM01–0 по-разному в нормальном режиме, режиме сброса таймера при совпадении и режимах с ШИМ. Во всех режимах установка COM01–0 = 0 сообщает формирователю сигналов, что никакое действие в регистре OC0 не должно выполняться при возникновении следующего совпадения. В таблице 3.53 описаны действия при сравнении в режимах без ШИМ. Аналогичная информация для режима с быстрой ШИМ приведена в таблице 3.54, а для режима с ШИМ с фазовой коррекцией – в таблице 3.55.

Изменение состояния битов COM01–0 окажет эффект при возникновении первого совпадения после записи этих битов. В режимах без ШИМ можно принудительно получить незамедлительный эффект, используя бит FOC0.

3.9.6 Режимы работы

Режим работы, то есть поведение таймера/счетчика и линий выходных сигналов, определяется комбинацией битов режима работы таймера (WGM01–0) и битов режима формирования выходного сигнала (COM01–0). Биты COM01–0 не влияют на последовательность счета. Биты режима работы таймера WGM01–0 оказывают влияние на последовательность счета. Биты COM01 и COM00 определяют, должен ли выход ШИМ быть инвертированным или нет (широтно-импульсная модуляция с инверсией или без). Для режимов без ШИМ биты COM01–0 определяют, какое действие необходимо осуществить при выполнении условия сравнения: установить, сбросить или переключить выход (см. также 3.9.5 «Блок формирования выходного сигнала»).

Более подробная информация по временным диаграммам работы таймера приведена в 3.9.7 «Временные диаграммы таймера/счетчика 0».

Нормальный режим работы

Самым простым режимом работы является нормальный режим (WGM01–0 = 0). В данном режиме счетчик всегда работает как суммирующий (инкрементирующий), при этом

сброс счетчика не выполняется. Счетчик просто переполняется, когда достигает своего максимального 8-разрядного значения (верхний предел равен 0xFF), и затем перезапускается, начиная счет от нижнего предела (0x00). В нормальном режиме работы флаг переполнения таймера/счетчика TOV0 будет установлен на том же такте синхронизации, когда TCNT0 примет нулевое значение. Флаг TOV0 в данном случае ведет себя подобно девятому биту, за исключением того, что он только устанавливается и не сбрасывается. Однако программно это свойство может быть использовано для повышения разрешающей способности таймера, если использовать прерывание по переполнению таймера, при возникновении которого флаг TOV0 сбрасывается автоматически. Для нормального режима работы не существует каких-либо особых ситуаций, связанных с записью нового состояния счетчика, когда требовалось бы учесть меры предосторожности.

Блок сравнения может использоваться для генерации прерываний. Не рекомендуется использовать выход OC0 для генерации сигналов в нормальном режиме работы, так как в этом случае будет затрачена значительная часть процессорного времени.

Режим сброса таймера при совпадении (CTC)

В режиме CTC (WGM01–0 = 2) регистр OCR0 используется для управления разрешающей способностью счетчика. Если задан режим CTC и значение счетчика (TCNT0) совпадает со значением регистра OCR0, то счетчик обнуляется. Таким образом, OCR0 задает вершину счета для счетчика, а, следовательно, и его разрешающую способность. В данном режиме обеспечивается более широкий диапазон регулировки частоты генерируемых импульсов. Он также упрощает операцию счета внешних событий.

Временная диаграмма для режима CTC показана на рисунке 3.39. Значение счетчика (TCNT0) увеличивается до тех пор, пока оно не становится равным значению в регистре OCR0, после чего счетчик (TCNT0) обнуляется.

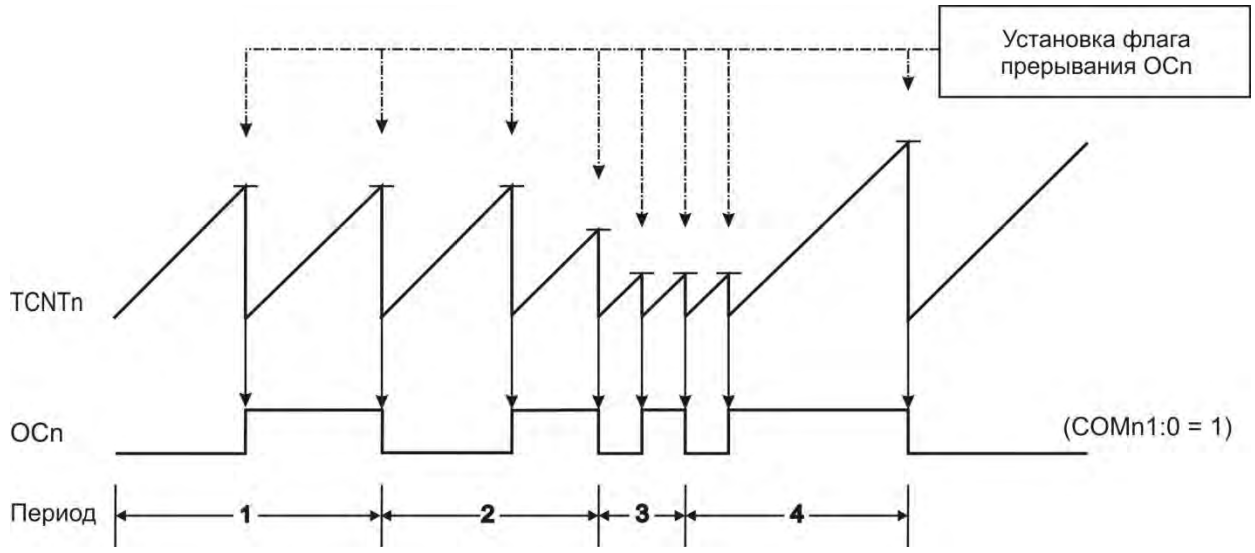


Рисунок 3.39 – Временная диаграмма для режима CTC

С помощью флага OCF0 прерывание может генерироваться всякий раз, когда счетчик достигает своего верхнего предела счета. Если разрешено прерывание, то процедура обработки прерывания может использоваться для обновления значения вершины счета. Однако, задание значения вершины счета, близкого к значению нижнего предела, когда счетчик работает без предделения или с малым значением предделения, необходимо выполнять с особой осторожностью, так как в режиме CTC нет двойной буферизации. Если новое значение, записанное в OCR0, будет меньше текущего значения TCNT0, то счетчик пропустит событие совпадения. В таком случае счетчик должен будет досчитать до своего максималь-

ного значения (0xFF) и перезапуститься, начав счет с 0x00, пока он не достигнет нового значения в регистре OCR0.

В режиме CTC выход компаратора OC0 используется для вывода генерированного сигнала, который изменяет логический уровень при каждом совпадении. Для этого необходимо задать режим переключения (COM01–0 = 1). Значение OC0 не будет видимым на выводе порта, если направление данных для этого вывода не установлено как выход. Генерируемый сигнал будет иметь максимальную частоту $f_{OC0} = f_{clk\_I/O}/2$, когда значение регистра OCR0 станет равным нулю (0x00). Частота сигналов определяется по формуле:

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

где переменная N задает коэффициент деления предделителя (1, 8, 32, 64, 128, 256 или 1024).

Как и в нормальном режиме работы, флаг TOV0 устанавливается на том же самом такте таймера, когда значение счетчика меняется с максимального на 0x00.

Режим быстрой широтно-импульсной модуляции

Режим быстрой широтно-импульсной модуляции (WGM01–0 = 3) позволяет генерировать высокочастотные ШИМ-сигналы. От другого режима с ШИМ данный режим отличается однонаправленностью счета. Счетчик считает от нижнего предела до максимального значения, а затем перезапускается, начиная счет от нижнего предела. В неинвертирующем режиме сравнения вывод OC0 сбрасывается при совпадении TCNT0 и OCR0 и устанавливается при достижении нижнего предела счета. В инвертирующем режиме сравнения вывод OC0 устанавливается при совпадении TCNT0 и OCR0 и сбрасывается при достижении нижнего предела счета. Благодаря однонаправленности счета рабочая частота для режима быстрой ШИМ может быть в два раза выше по сравнению с режимом ШИМ с фазовой коррекцией, где используется двунаправленный счет. Возможность генерации высокочастотных ШИМ-сигналов делает использование данного режима полезным в задачах стабилизации питания, коррекции и цифро-аналогового преобразования. Высокая частота при этом позволяет использовать внешние элементы физически малых размеров (катушки индуктивности, конденсаторы), снижая тем самым общую стоимость системы.

В режиме быстрой ШИМ содержимое счетчика инкрементируется до тех пор, пока не достигнет максимального значения (0xFF). Далее, на следующем такте синхронизации таймера счетчик сбрасывается. Временная диаграмма для режима быстрой ШИМ показана на рисунке 3.40. Значение TCNT0 на временной диаграмме показано в виде гистограммы для иллюстрации однонаправленности счета. На диаграмме показаны как неинвертированный, так и инвертированный ШИМ-выходы. Короткими горизонтальными линиями на графике TCNT0 обозначены положения, где значения OCR0 и TCNT0 совпадают.

Флаг переполнения таймера/счетчика TOV0 устанавливается всякий раз, когда счетчик достигает своего максимального значения (0xFF). Если прерывание разрешено, то процедура обработки прерывания может использоваться для обновления сравниваемого значения.

В режиме быстрой ШИМ блок сравнения позволяет формировать ШИМ-сигналы на выводе OC0. Установка COM01–0 = 2 обеспечит генерацию неинвертированного ШИМ-сигнала. Инвертированный ШИМ-сигнал можно сгенерировать установкой COM01–0 = 3 (см. таблицу 3.54). Фактическое значение OC0 будет присутствовать на выводе порта, только если для данного вывода задано направление выхода. ШИМ-сигнал генерируется путем установки (или сброса) регистра OC0 при совпадении значений OCR0 и TCNT0, а также путем сброса (или установки) регистра OC0 на том же такте синхронизации таймера, когда

счетчик сбрасывается (состояние счетчика изменяется от максимального значения до нижнего предела счета).

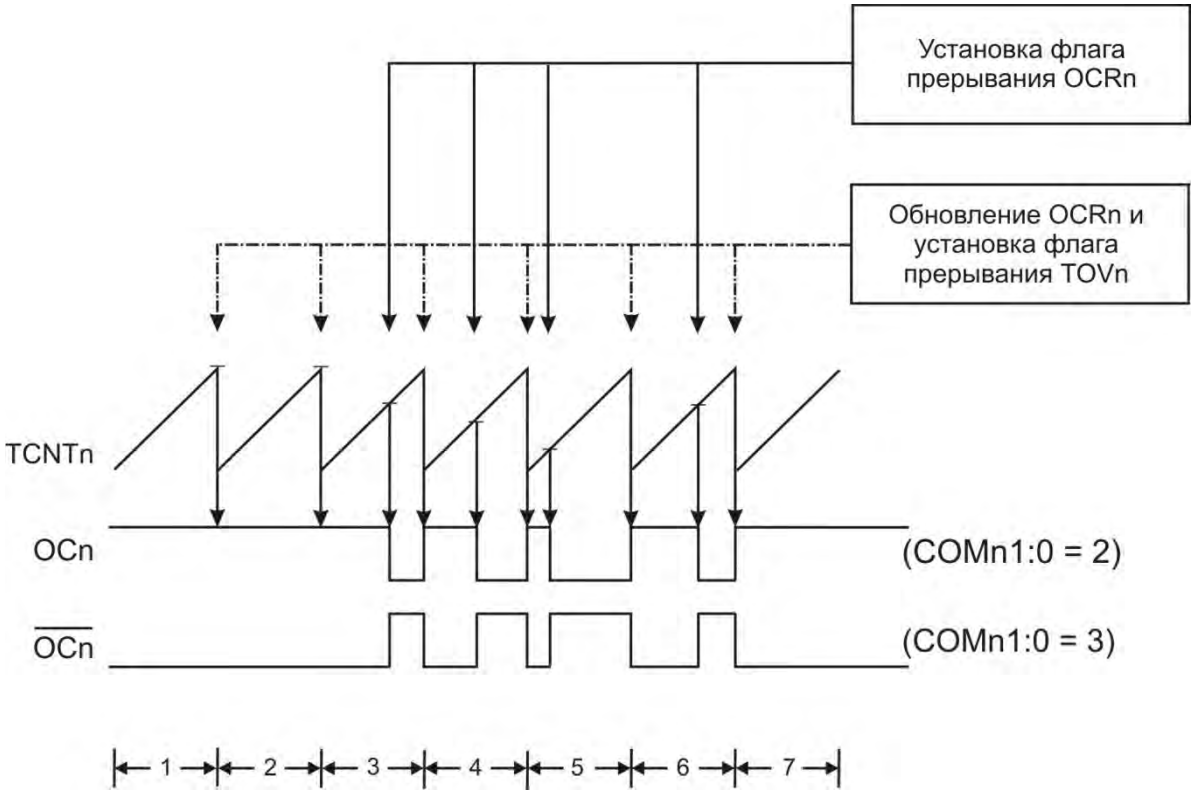


Рисунок 3.40 – Временная диаграмма для режима быстрой ШИМ

Частота выходного ШИМ-сигнала может быть вычислена по формуле:

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256},$$

где переменная N задает коэффициент деления предделителя (1, 8, 32, 64, 128, 256 или 1024).

Предельные значения регистра OCR0 представляют собой особые случаи при формировании ШИМ-сигналов в режиме быстрой ШИМ. Если в регистр OCR0 записать значение, равное нижнему пределу счета, то через каждые 256 тактов синхронизации таймера на выходе будет генерироваться узкий импульсный сигнал. Если установить значение OCR0 равным максимальному значению, то на выходе будет постоянный логический уровень «1» или «0» (зависит от полярности сигнала на выходе, определяемой битами COM01–0).

Частотный выходной сигнал (с пятидесятипроцентным коэффициентом заполнения) в режиме быстрой ШИМ может быть достигнут путем установки OC0 на переключение своего логического уровня при каждом совпадении (COM01–0 = 1). Сгенерированный сигнал будет иметь максимальную частоту $f_{OC0} = f_{clk\_I/O}/2$, когда значение регистра OCR0 равно нулю. Данная функция похожа на переключение OC0 в режиме CTC за исключением того, что в режиме быстрой ШИМ разрешена двойная буферизация в блоке сравнения.

Режим широтно-импульсной модуляции с фазовой коррекцией (ШИМ ФК)

Режим ШИМ ФК (WGM01–0 = 1) позволяет выполнять фазовую коррекцию ШИМ-сигнала с высокой разрешающей способностью. Режим ШИМ ФК основан на двунаправленной работе таймера/счетчика. Счетчик постоянно выполняет счет в направлении от

нижнего предела до максимального значения, а затем обратно от максимального значения к нижнему пределу. В неинвертирующем режиме сравнения выход OC0 сбрасывается/устанавливается при совпадении значений TCNT0 и OCR0 во время прямого/обратного счета соответственно. В инвертирующем режиме сравнения, наоборот, во время прямого счета происходит установка, а во время обратного – сброс выхода OC0. При двунаправленном счете максимальная частота ШИМ-сигнала меньше, чем при однонаправленном счете. Однако, благодаря симметричности режимов ШИМ с двунаправленным счетом, данные режимы предпочтительны при решении задач управления приводами.

Разрешающая способность ШИМ для данного режима фиксирована и равна 8 битам. В режиме ШИМ ФК счетчик инкрементируется, пока не достигнет максимального значения (0xFF). По достижении максимального значения счетчик меняет направление счета. Значение TCNT0 остается равным максимальному значению в течение одного такта синхронизации таймера. На рисунке 3.41 показана временная диаграмма работы таймера/счетчика для режима ШИМ ФК. Значение TCNT0 представлено в виде гистограммы для иллюстрации двунаправленности работы счетчика. Диаграмма отображает состояние неинвертированного и инвертированного ШИМ-выхода. Короткие горизонтальные линии на графике изменения TCNT0 указывают положения, где значения OCR0 и TCNT0 совпадают.

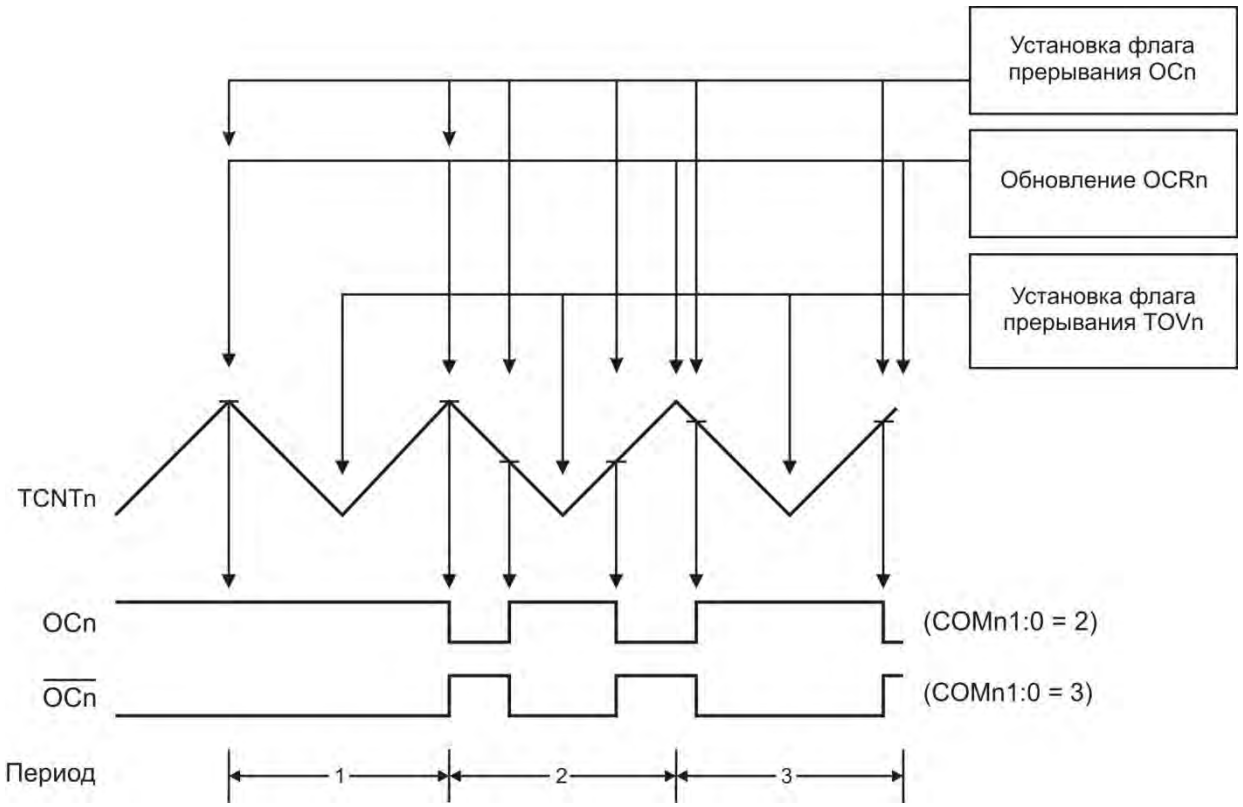


Рисунок 3.41 – Временная диаграмма для режима ШИМ ФК

Флаг переполнения таймера/счетчика (TOV0) устанавливается всякий раз, когда счетчик достигает нижнего предела, и может использоваться для генерации прерывания.

В режиме ШИМ ФК блок сравнения позволяет формировать ШИМ-сигналы на выводе OC0. Установка битов COM01–0 = 2 приводит к генерации неинвертированного ШИМ-сигнала. Инвертированный ШИМ-сигнал может быть получен установкой COM01–0 = 3 (см. таблицу 3.55). Фактическое значение OC0 будет видимым на соответствующем выводе порта, только если направление данных для этого вывода установлено как выход. ШИМ-сигнал генерируется путем сброса (или установки) регистра OC0 при совпадении OCR0 и TCNT0 во время прямого счета и путем установки (или сброса) реги-

стра OC0 при совпадении OCR0 и TCNT0 во время обратного счета. Результирующая частота ШИМ-сигнала в режиме ШИМ ФК может быть вычислена по формуле:

$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510},$$

где переменная N задает коэффициент деления предделителя (1, 8, 32, 64, 128, 256 или 1024).

Предельные значения регистра OCR0 представляют собой особые случаи при формировании ШИМ-сигналов в режиме ШИМ ФК. Если установить значение регистра OCR0 равным нижнему пределу, то выход будет постоянно иметь низкий уровень, а если установить равным максимальному значению, то выход будет постоянно иметь высокий уровень в неинвертирующем режиме широтно-импульсной модуляции. В инвертирующем режиме выход будет иметь противоположные логические значения.

В самом начале периода 2 (на рисунке 3.41) сигнал OCn переходит из единицы в ноль, несмотря на то, что совпадения нет. Точка данного перехода должна гарантировать симметричность относительно нижней границы счета. Существует два случая изменения логического уровня сигнала, когда нет совпадения значений OCR0 и TCNT0:

- OCR0 изменяет свое значение с максимального, как показано на рисунке 3.41. Когда значение OCR0 равно максимальному, значение вывода OCn будет таким же, как результат совпадения при обратном счете. Для обеспечения симметрии относительно нижнего предела значение OCn при максимальном значении счетчика должно соответствовать результату совпадения при прямом счете.

- Таймер начинает счет со значения, большего по сравнению с OCR0, по этой причине пропуская событие совпадения. Отсюда следует изменение OCn, которое должно было бы произойти во время прямого счета.

3.9.7 Временные диаграммы таймера/счетчика 0

На рисунках представлена информация о том, когда происходит установка флагов прерывания. На рисунке 3.42 показаны временные показатели основной работы таймера/счетчика. Данный рисунок иллюстрирует счетную последовательность в области максимального значения счетчика (0xFF) для всех режимов, кроме режима ШИМ ФК. На рисунке 3.43 показаны те же временные характеристики, но при включенном предделителе.

Представленные диаграммы соответствуют синхронному по отношению к системной частоте режиму тактирования таймера/счетчика. Однако они будут полностью соответствовать асинхронному режиму работы, если заменить $clk_{I/O}$ на сигнал генератора таймера/счетчика.

На рисунке 3.44 отображается установка флага OCF0 во всех режимах, кроме режима CTC. На рисунке 3.45 отображается установка флага OCF0 и сброс TCNT0 в режиме CTC.

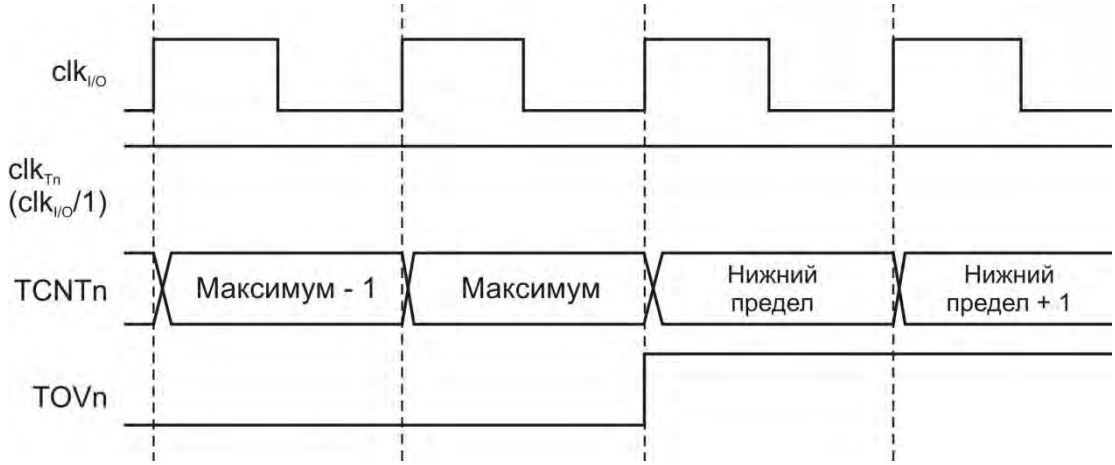


Рисунок 3.42 – Временная диаграмма таймера/счетчика без предделения

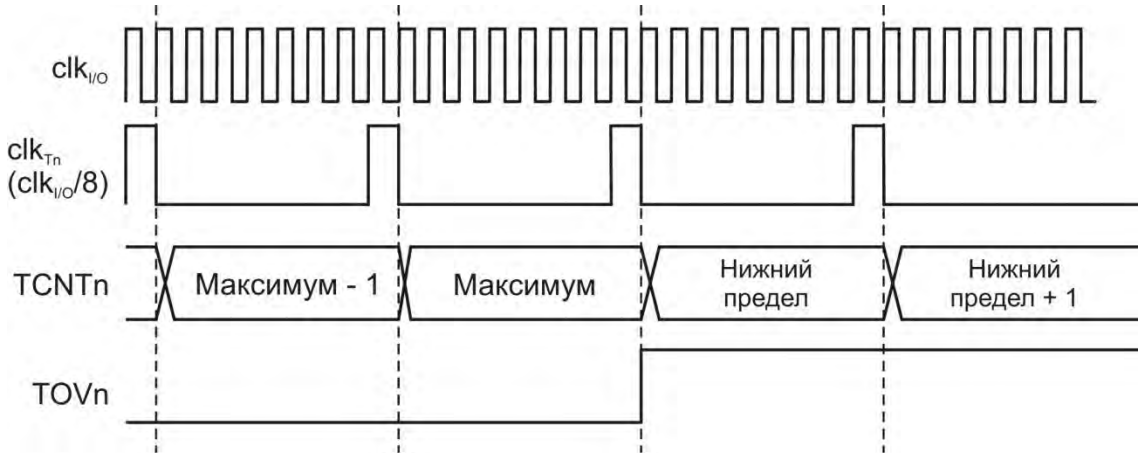


Рисунок 3.43 – Временная диаграмма таймера/счетчика с предделением ($f_{clk\_I/O}/8$)

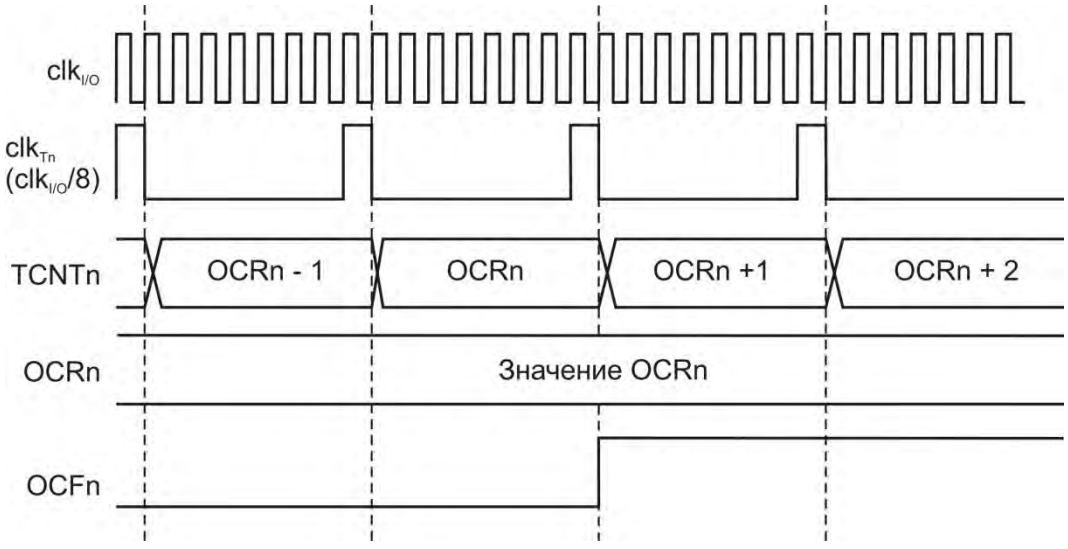


Рисунок 3.44 – Временная диаграмма таймера/счетчика, установка флага OCF0, с предделением ($f_{clk\_I/O}/8$)

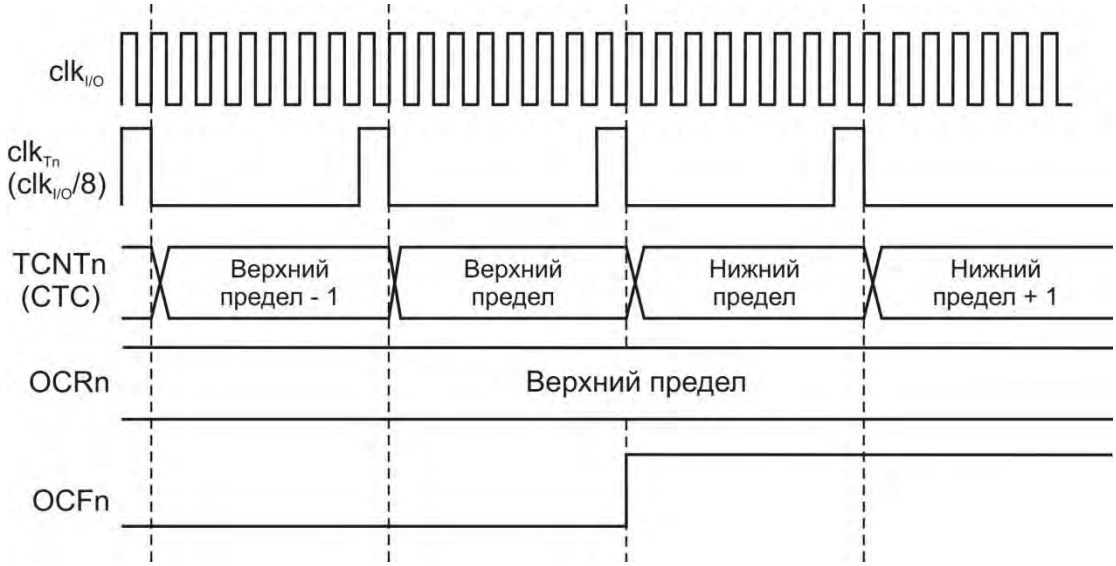


Рисунок 3.45 – Временная диаграмма таймера/счетчика, с предделением ($f_{clk\_I/O}/8$), в режиме сброса таймера при совпадении

3.9.8 Описание регистров таймера/счетчика 0

Регистр управления таймером/счетчиком 0 – TCCR0

Бит	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Чтение/ Запись	З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – FOC0: Установка принудительного совпадения

Функция бита FOC0 активна, только если с помощью битов WGM задан один из режимов, где нет широтно-импульсной модуляции. Если записать логическую единицу в бит FOC0, то это приведет к установке принудительного совпадения на входе блока формирования выходного сигнала. Выход OC0 изменяется в соответствии с установками битов COM01, COM00. Значение, представленное в COM01:0, определяет эффект принудительного совпадения.

Бит FOC0 не генерирует каких-либо прерываний, а также не вызывает сброс таймера в режиме CTC, где регистр OCR0 задает верхний предел счета.

Бит FOC0 всегда считывается как ноль.

Разряды 6, 3 – WGM01–0: Режим работы таймера/счетчика 0

Эти биты определяют счетную последовательность счетчика, источник для верхнего предела значения счетчика и тип генерируемых сигналов. Режимами работы, поддерживаемыми блоком таймера/счетчика, являются: нормальный режим, режим сброса таймера при совпадении и два типа режимов с широтно-импульсной модуляцией (см. таблицу 3.52 и 3.9.6 «Режимы работы»).

Таблица 3.52 – Описание битов режима работы таймера/счетчика 0

Режим	WGM01	WGM00	Режим работы таймера/ счетчика	Верхний предел	Обновление значения OCR0	Установка флага TOV0
0	0	0	Нормальный	0xFF	немедленно	при достижении максимального значения
1	0	1	ШИМ ФК	0xFF	при достижении верхнего предела	при достижении нижнего предела
2	1	0	СТС	OCR0	немедленно	при достижении максимального значения
3	1	1	Быстрая ШИМ	0xFF	при достижении нижнего предела	при достижении максимального значения

Разряды 5,4 – COM01–0: Режим формирования выходного сигнала

Эти биты управляют поведением вывода выходного сравнения (OC0). Если один или оба бита COM01–0 установлены, то активируется функция выхода OC0, которая переопределяет нормальное функционирование линии ввода-вывода, к которой он подключен. Однако обратите внимание, что бит регистра направления данных (DDR), соответствующий выводу OC0, должен быть установлен, чтобы включить выходной драйвер.

После активизации альтернативной функции назначение битов COM01–0 зависит от установки битов WGM01–0. В таблице 3.53 отображаются функции битов COM01–0, когда с помощью WGM01–0 выбран нормальный режим работы или режим СТС (режимы без ШИМ).

Таблица 3.53 – Режим формирования выходного сигнала, режим без ШИМ

COM01	COM00	Описание
0	0	Нормальная работа порта, OC0 отключен
0	1	Переключение OC0 при совпадении
1	0	Сброс OC0 при совпадении
1	1	Установка OC0 при совпадении

В таблице 3.54 отображаются функции битов COM01–0, когда с помощью WGM01–0 выбран режим быстрой широтно-импульсной модуляции.

Таблица 3.54 – Режим формирования выходного сигнала, режим быстрой ШИМ

COM01	COM00	Описание
0	0	Нормальная работа порта, OC0 отключен
0	1	Зарезервировано
1	0	Сброс OC0 при совпадении, установка OC0 при достижении нижнего предела (неинвертирующий режим)
1	1	Установка OC0 при совпадении, сброс OC0 при достижении нижнего предела (инвертирующий режим)

Примечание – Особый случай возникает, когда значение OCR0 равно верхнему пределу и установлен бит COM01. В данном случае совпадение игнорируется, но установка или сброс происходят при достижении нижнего предела (см. выше подпункт «Режим быстрой широтно-импульсной модуляции» для изучения деталей).

Таблица 3.55 отображает функционирование битов COM01–0, когда с помощью WGM01–0 выбран режим широтно-импульсной модуляции с фазовой коррекцией.

Таблица 3.55 – Режим формирования выходного сигнала, режим ШИМ ФК

COM01	COM00	Описание
0	0	Нормальная работа порта, OC0 отключен
0	1	Зарезервировано
1	0	Сброс OC0 при совпадении во время прямого счета. Установка OC0 при совпадении во время обратного счета.
1	1	Установка OC0 при совпадении во время прямого счета. Сброс OC0 при совпадении во время обратного счета.

Примечание – Особый случай возникает, когда значение OCR0 равно верхнему пределу и установлен бит COM01. В данном случае совпадение игнорируется, но установка или сброс происходят при достижении верхнего предела (см. выше подпункт «Режим широтно-импульсной модуляции с фазовой коррекцией (ШИМ ФК)» для изучения деталей).

Разряды 2–0 – CS02–0: Выбор частоты синхронизации таймера

С помощью трех настроечных битов имеется возможность выбрать различные тактовые частоты, кратные исходной частоте синхронизации (см. таблицу 3.56).

Таблица 3.56 – Выбор частоты синхронизации таймера/счетчика 0

CS02	CS01	CS00	Описание
0	0	0	Нет синхронизации (таймер/счетчик 0 остановлен)
0	0	1	clk_{T0S}/1 (без предделения)
0	1	0	clk_{T0S}/8 (с предделением)
0	1	1	clk_{T0S}/32 (с предделением)
1	0	0	clk_{T0S}/64 (с предделением)
1	0	1	clk_{T0S}/128 (с предделением)
1	1	0	clk_{T0S}/256 (с предделением)
1	1	1	clk_{T0S}/1024 (с предделением)

Счетный регистр таймера/счетчика 0 – TCNT0

Бит	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр таймера/счетчика 0 обеспечен прямым доступом как для операции чтения, так и для операции записи. Запись в регистр TCNT0 блокирует совпадение на последующем такте синхронизации таймера. Изменение содержимого счетчика (TCNT0) во время счета связано с риском потери результата сравнения между TCNT0 и регистром OCR0.

Регистр сравнения – OCR0

Бит	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр значения сравнения содержит 8-разрядное значение, которое постоянно сравнивается со значением счетчика (TCNT0). Событие совпадения значений может ис-

пользоваться для генерации прерывания по выполнению условия сравнения или для генерации прямоугольных импульсов на выводе OC0.

3.9.9 Асинхронная работа таймера/счетчика 0

Регистр состояния асинхронного режима – ASSR

Бит	7	6	5	4	3	2	1	0	
	-	-	-	-	AS0	TCN0UB	OCR0UB	TCR0UB	ASSR
Чтение/ Запись	Ч	Ч	Ч	Ч	Ч/З	Ч	Ч	Ч	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 3 – AS0: Разрешение асинхронного тактирования таймера/счетчика 0

Если AS0 = 0, таймер/счетчик 0 тактируется сигналом синхронизации ввода-вывода ($clk_{I/O}$). Если AS0 = 1, таймер/счетчик 0 тактируется кварцевым генератором, подключенным к выводу генератора таймера 1 (TOSC1). При изменении значения AS0 содержимое регистров TCNT0, OCR0 и TCCR0 может быть нарушено.

Разряд 2 – TCN0UB: Флаг занятости таймера/счетчика 0 при обновлении значения

Если таймер/счетчик 0 работает асинхронно и выполняется запись в TCNT0, то данный флаг устанавливается. После завершения обновления TCNT0 из регистра временного хранения флаг TCN0UB сбрасывается аппаратно. Логический ноль во флаге показывает, что регистр TCNT0 готов к новому обновлению своего значения.

Разряд 1 – OCR0UB: Флаг занятости регистра значения сравнения при обновлении значения

Если таймер/счетчик 0 работает асинхронно и выполняется запись в OCR0, то данный флаг устанавливается. После завершения обновления OCR0 из регистра временного хранения флаг OCR0UB сбрасывается аппаратно. Логический ноль во флаге показывает, что регистр OCR0 готов к новому обновлению своего значения.

Разряд 0 – TCR0UB: Флаг занятости регистра управления таймером/счетчиком 0 при обновлении значения

Если таймер/счетчик 0 работает асинхронно и выполняется запись в TCCR0, то данный флаг устанавливается. После завершения обновления TCCR0 из регистра временного хранения флаг TCR0UB сбрасывается аппаратно. Логический ноль во флаге показывает, что регистр TCCR0 готов к новому обновлению своего значения.

Если выполнять запись в любой из трех регистров таймера/счетчика 0, когда соответствующий флаг занятости установлен, то обновленное значение может повредиться и стать причиной возникновения непреднамеренного прерывания.

Механизм считывания значений TCNT0, OCR0 и TCCR0 различный. Если выполняется чтение TCNT0, то считывается фактическое значение таймера. Если же выполняется чтение OCR0 или TCCR0, то считывается содержимое регистра временного хранения.

Асинхронная работа таймера/счетчика 0

Когда таймер/счетчик работает асинхронно, необходимо учесть некоторые особенности.

1 При переключении между асинхронным и синхронным тактированием таймера/счетчика содержимое регистров TCNT0, OCR0 и TCCR0 может быть нарушено. Во из-

бежание этого необходимо придерживаться следующей безопасной последовательности переключения:

а) Отключить прерывания таймера/счетчика 0 путем сброса битов OCIE0 и TOIE0.

б) Выбрать необходимый источник синхронизации с помощью бита AS0.

в) Выполнить запись новых значений в TCNT0, OCR0 и TCCR0.

г) При переходе к асинхронному режиму тактирования дождаться сброса флагов TCN0UB, OCR0UB и TCR0UB.

д) Сбросить флаги прерываний таймера/счетчика 0.

е) При необходимости разрешить прерывания.

2 Генератор оптимизирован для использования совместно с часовым кварцевым резонатором частотой 32,768 кГц. Подключение внешнего тактового сигнала к выводу TOSC1 может привести к некорректной работе таймера. Частота основной синхронизации ЦПУ должна быть не менее чем в четыре раза выше частоты данного генератора.

3 При выполнении записи в любой из регистров TCNT0, OCR0 или TCCR0 значение перемещается во временный регистр и фиксируется после двух положительных фронтов сигнала TOSC1. Пользователю не следует записывать новое значение, прежде чем содержимое временного регистра переместится в соответствующий пункт назначения. Каждый из трех вышеупомянутых регистров имеет свой индивидуальный временный регистр. Это означает, что, например, запись в TCNT0 не помешает выполняемой в данный момент записи в регистр OCR0. Для того чтобы определить, что произошло перемещение именно в заданный регистр, и был реализован регистр ASSR – регистр состояния асинхронного режима.

4 При входе в режим хранения или расширенный режим ожидания после записи в TCNT0, OCR0 или TCCR0 программист должен дождаться завершения обновления записанного регистра, если таймер/счетчик 0 используется для пробуждения из этих режимов. Иначе микроконтроллер перейдет в режим сна раньше, чем вступят в силу желаемые изменения. Это особенно важно, если прерывание по результату сравнения таймера/счетчика 0 используется для пробуждения микроконтроллера, так как функция выходного сравнения отключается во время записи в OCR0 или TCNT0. Если цикл записи не закончен и микроконтроллер входит в режим сна прежде, чем бит OCR0UB возвращается в ноль, то устройство никогда не получит прерывания по результату сравнения и, следовательно, не сможет пробудиться.

5 Если таймер/счетчик 0 используется для пробуждения микроконтроллера из режима хранения или расширенного режима ожидания, то должны быть приняты меры предосторожности, если пользователь пожелает вновь перевести устройство в один из этих режимов. Для сброса логики прерываний требуется один такт TOSC1. Если интервал времени между пробуждением микроконтроллера и повторным вводом режима сна меньше чем один период TOSC1, то прерывание в дальнейшем не возникнет и устройство не сможет пробудиться. Если программист не уверен в прохождении достаточного количества времени перед повторным вводом в режим хранения или расширенный режим ожидания, то можно применять следующий алгоритм, который гарантирует прохождение одного периода TOSC1:

а) Запись значения в TCCR0, TCNT0 или OCR0.

б) Ожидание сброса соответствующего флага занятости при обновлении в регистре ASSR.

в) Ввод режима хранения или расширенного режима ожидания.

6 Когда выбрана асинхронная работа, генератор таймера/счетчика 0 частотой 32768 Гц включен постоянно, за исключением режима хранения и режима ожидания микроконтроллера. После сброса при подаче питания или пробуждения из режима хранения или режима ожидания пользователь должен учесть, что для возобновления нормальной стабильной работы данного генератора требуется минимум одна секунда. Пользователю рекомендуется подождать по крайней мере одну секунду перед использованием таймера/счетчика 0 после подачи питания или выхода из режима хранения или режима ожидания. Содержимое всех

регистров таймера/счетчика 0 необходимо рассматривать как потерянное после пробуждения из указанных выше режимов из-за нестабильности тактового сигнала при запуске независимо от того, какой асинхронный источник используется (генератор или внешний сигнал на выводе TOSC1).

7 Выход микроконтроллера из режима микропотребления и расширенного режима ожидания при асинхронном тактировании таймера/счетчика 0 происходит в приведенной далее последовательности. Если выполняется условие прерывания, то процесс пробуждения начинается на следующем такте синхронизации таймера, то есть таймер минимум на единицу изменит свое состояние, прежде чем процессор получит доступ к значению счетчика. После пробуждения микроконтроллер останавливается на четыре такта синхронизации ЦПУ, выполняет обработку прерывания, а затем возвращается к выполнению инструкции, следующей за SLEEP.

8 Чтение регистра TCNT0 вскоре после пробуждения из режима микропотребления может дать некорректный результат. Поскольку TCNT0 тактируется от асинхронного источника TOSC, то чтение TCNT0 должно выполняться через регистр, согласованный по времени с внутренней синхронизацией ввода-вывода. Синхронизация выполняется на каждом нарастающем фронте сигнала TOSC1. При пробуждении из режима микропотребления снова активизируется синхронизация ввода-вывода ($clk_{I/O}$), и при чтении TCNT0 фактически будет считываться предыдущее значение (которое было записано перед вводом в режим сна) до следующего нарастающего фронта TOSC1. Фаза тактового сигнала TOSC после выхода из режима микропотребления совершенно непредсказуема, так как она зависит от момента пробуждения микроконтроллера. С учетом этого, при чтении содержимого TCNT0 рекомендуется придерживаться следующего алгоритма:

а) Записать произвольное значение в регистр OCR0 или в TCCR0.

б) Дождаться сброса соответствующего флага занятости при обновлении.

в) Выполнить чтение TCNT0.

9 Во время асинхронной работы синхронизация флагов прерываний асинхронного таймера требует три такта ЦПУ плюс один такт синхронизации таймера. Таким образом, таймер должен изменить свое состояние как минимум на единицу, прежде чем процессор сможет считать его значение, вызывающее установку флага прерывания. Выход генератора импульсов OC0 изменяется в зависимости от синхронизации таймера и не согласован по времени с тактированием ЦПУ.

Регистр масок прерываний таймеров/счетчиков – TIMSK

Бит	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 1 – OCIE0: Разрешение прерывания по совпадению таймера/счетчика 0

Если OCIE0 = 1, а также установлен бит I в регистре статуса, то прерывание по совпадению таймера/счетчика 0 разрешается. В этом случае прерывание возникает, если обнаруживается совпадение значения таймера/счетчика 0 с значением сравнения, т.е. когда установлен флаг OCF0 в регистре флагов прерываний таймеров/счетчиков TIFR.

Разряд 0 - TOIE0: Разрешение прерывания по переполнению таймера/счетчика 0

Если TOIE0 = 1, а также установлен бит I в регистре статуса, то прерывание по переполнению таймера/счетчика 0 разрешается. В этом случае прерывание возникает, если обнаруживается переполнение таймера/счетчика 0, т.е. когда установлен бит TOV0 в регистре флагов прерываний таймеров/счетчиков TIFR.

Регистр флагов прерываний таймеров/счетчиков – TIFR

Бит	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 1 – OCF0: Флаг совпадения таймера/счетчика 0

OCF0 устанавливается в единицу, когда обнаруживается совпадение между значением таймера/счетчика 0 и данными в регистре OCR0 (регистре значения сравнения). OCF0 сбрасывается аппаратно при переходе на соответствующий вектор обработки прерывания. Альтернативно, флаг OCF0 можно сбросить путем записи логической единицы в соответствующий разряд регистра. Если установлены бит I в регистре SREG, бит OCIE0 (разрешено прерывание по совпадению таймера/счетчика 0) и флаг OCF0, то генерируется прерывание по совпадению таймера/счетчика 0.

Разряд 0 – TOV0: Флаг переполнения таймера/счетчика 0

Флаг TOV0 устанавливается в единицу, когда в таймере/счетчике 0 возникает переполнение. Флаг TOV0 сбрасывается аппаратно при переходе на соответствующий вектор обработки прерывания. Альтернативно, флаг TOV0 можно сбросить путем записи логической единицы в соответствующий разряд регистра. Если установлены бит I в регистре SREG, бит TOIE0 (разрешено прерывание по переполнению таймера/счетчика 0) и флаг TOV0, то генерируется прерывание по переполнению таймера/счетчика 0. В режиме с ШИМ данный флаг устанавливается, когда таймер/счетчик 0 изменяет направление счета при значении $00.

3.9.10 Предделитель таймера/счетчика 0

Тактовый сигнал таймера/счетчика 0 обозначен как clk_{T0}. По умолчанию clk_{T0} подключен к системному источнику синхронизации ввода-вывода $clk_{I/O}$. В соответствии с установкой бита AS0 в регистре ASSR таймер/счетчик 0 тактируется асинхронно с вывода TOSC1. Это позволяет использовать таймер/счетчик 0 в качестве счетчика реального времени (RTC). Когда установлен бит AS0, выводы TOSC1 и TOSC2 отключаются от порта C, и тогда между этими выводами может быть подключен кварцевый резонатор в качестве независимого тактового источника таймера/счетчика 0. Генератор оптимизирован для использования совместно с кварцевым резонатором частотой 32,768 кГц. Подключение внешнего тактового источника к выводу TOSC1 не рекомендуется.

Предделитель таймера/счетчика 0 позволяет выбрать следующие тактовые сигналы: $clk_{T0S}/8$, $clk_{T0S}/32$, $clk_{T0S}/64$, $clk_{T0S}/128$, $clk_{T0S}/256$ и $clk_{T0S}/1024$. Кроме того, имеется возможность остановить синхронизацию. Установка бита PSR0 в регистре SFIOR сбрасывает предделитель. Данная функция позволяет программисту работать с предсказуемым поведением предделителя.

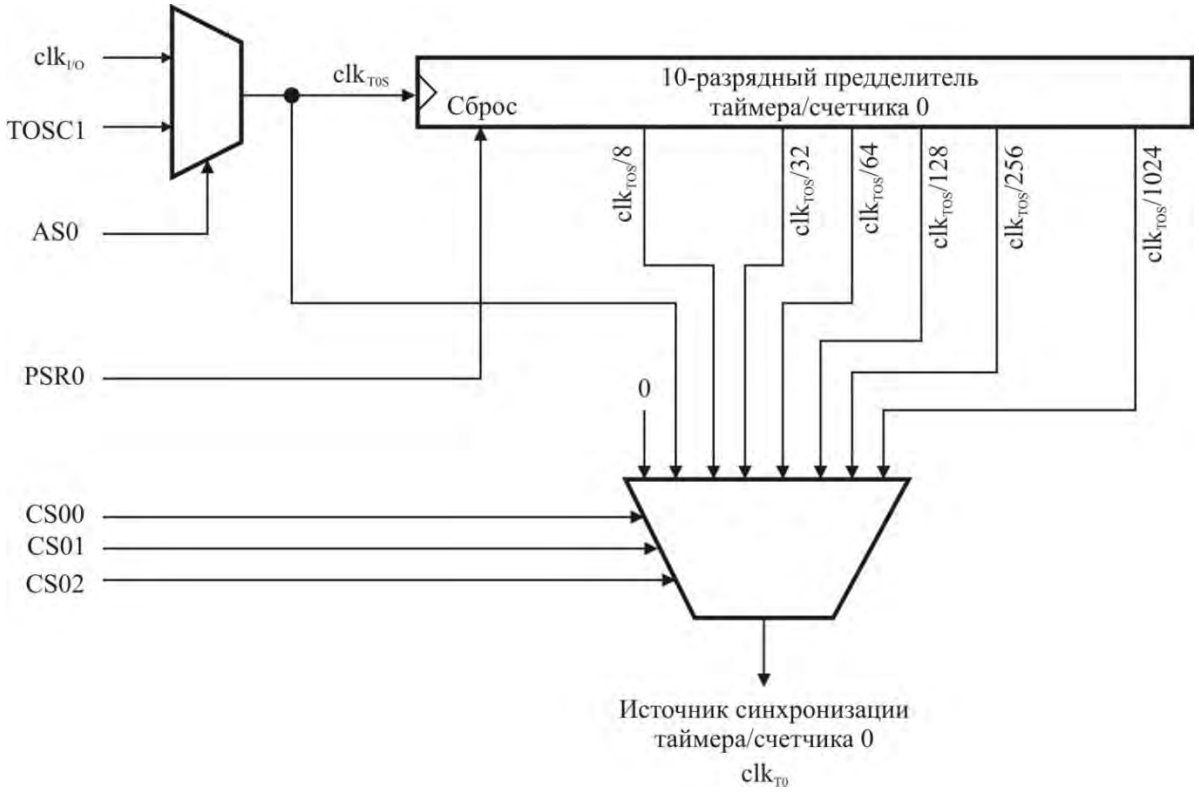


Рисунок 3.46 – Предделитель таймера/счетчика 0

Регистр специальных функций ввода-вывода – SFIOR

Бит	7	6	5	4	3	2	1	0	
	TSM	-	-	-	ACME	PUD	PSR0	PSR321	SFIOR
Чтение/ Запись	Ч/З	Ч	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – TSM: Режим синхронизации таймера/счетчика

Запись логической единицы в бит TSM активирует режим синхронизации таймера/счетчика. В этом режиме значение, записанное в биты PSR0 и PSR321, сохраняется и, следовательно, удерживается установка соответствующих сигналов сброса предделителя. Это гарантирует, что соответствующие таймеры/счетчики остановлены и могут быть сконфигурированы на одно и то же значение без риска изменения состояния одного из них во время настройки. Когда TSM = 0, биты PSR0 и PSR321 сбрасываются аппаратно, а таймеры/счетчики начинают счет одновременно.

Разряд 1 – PSR0: Сброс предделителя таймера/счетчика 0

Когда этот бит равен логической единице, предделитель таймера/счетчика 0 сбрасывается. Бит PSR0 обычно сбрасывается аппаратно сразу после установки. Если данный бит устанавливается, когда таймер/счетчик 0 работает в асинхронном режиме, то он останется равным логической единице, пока не сбросится предделитель. Бит PSR0 не сбрасывается аппаратно, если установлен бит TSM.

3.10 16-разрядный таймер/счетчик (таймер/счетчик 1 и таймер/счетчик 3)

16-разрядный таймер/счетчик предназначен для точного задания временных интервалов, формирования импульсов и генерации сигналов с ШИМ, измерения временных характеристик импульсных сигналов.

Основные отличительные особенности:

- Разрядность 16 бит (т. е. имеется возможность организации 16-разрядной ШИМ).
- Три независимых блока сравнения.
- Двойная буферизация регистров значения сравнения.
- Один блок захвата.
- Подавитель шумов на входе блока захвата.
- Режим сброса таймера при совпадении с значением сравнения (автоматическая перезагрузка).
- Широтно-импульсная модуляция с фазовой коррекцией без импульсных помех.
- Регулируемый период ШИМ.
- Частотный генератор.
- Счетчик внешних событий.
- 10 независимых источников прерываний (TOV1, OCF1A, OCF1B, OCF1C, ICF1, TOV3, OCF3A, OCF3B, OCF3C и ICF3).

На рисунке 3.47 приведена структурная схема таймера/счетчика.

Расположение и назначение выводов таймеров/счетчиков 1 и 3 приведены в таблицах 3.30 и 3.39.

3.10.1 Краткий обзор

Многие обозначения регистров и битов в данном подразделе представлены в общей форме. Латинская строчная буква «n» заменяет номер таймера/счетчика (в данном случае 1 или 3), а буква «x» заменяет наименование канала сравнения (A, B или C). Однако при использовании обозначений регистров или битов в программе необходимо применять точную форму (например, TCNT1 для доступа к значению таймера/счетчика 1 и т.д.).

Упрощенная структурная схема 16-разрядного таймера/счетчика представлена на рисунке 3.47. Связи с регистрами, к которым осуществляет доступ ЦПУ, в том числе биты ввода-вывода и линии ввода-вывода, показаны жирной линией. Специфические для данного устройства регистры, расположение и назначение их битов приведены в 3.10.10 «Описание регистров 16-разрядного таймера/счетчика».

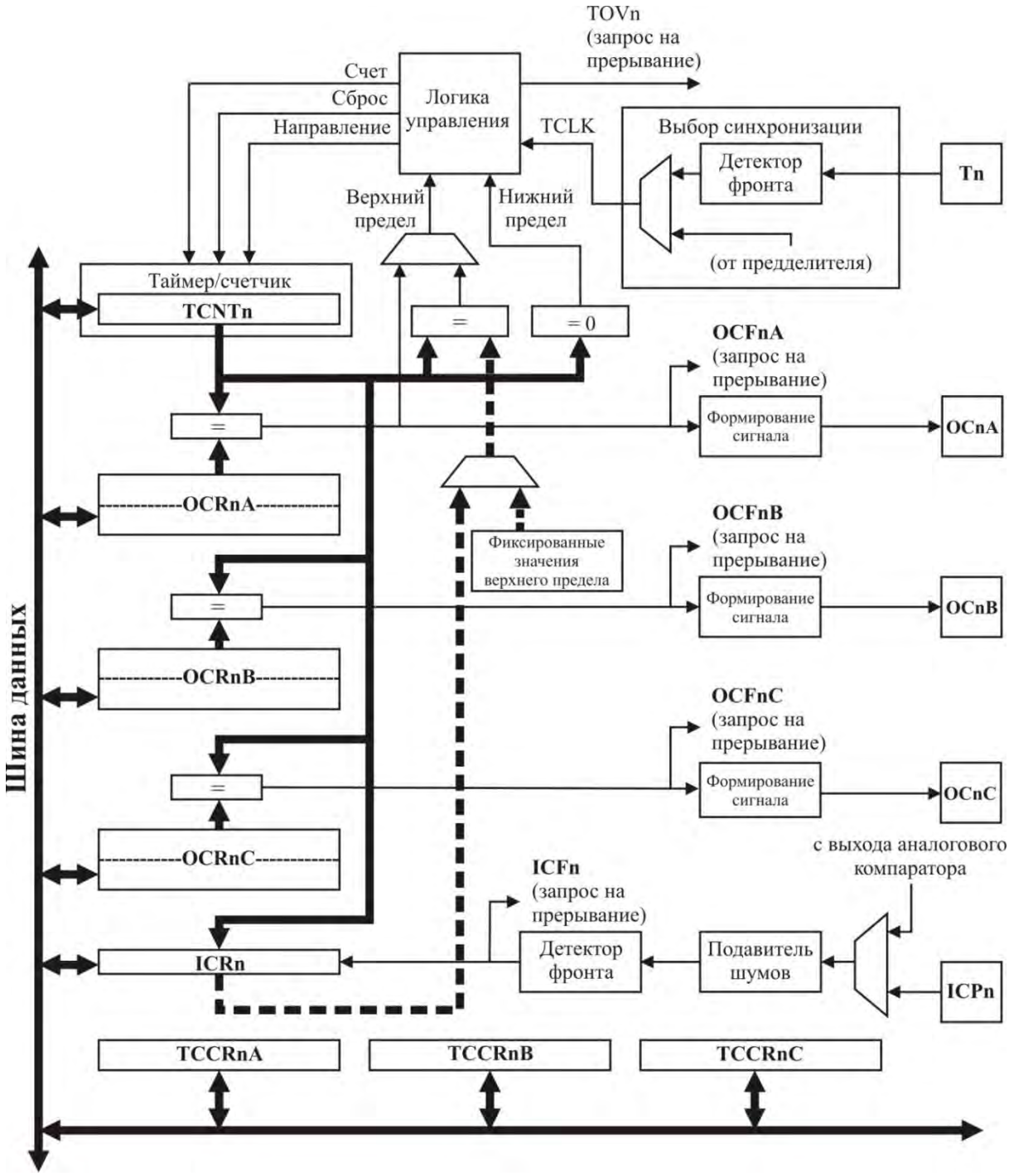


Рисунок 3.47 – Структурная схема 16-разрядного таймера/счетчика

Регистры

Счетный регистр таймера/счетчика (TCNTn), регистры сравнения (OCRnA/B/C), а также регистр захвата (ICRn) являются 16-разрядными регистрами. В связи с этим, во время доступа к данным регистрам должны быть соблюдены специальные процедуры (см. 3.10.2 «Доступ к 16-разрядным регистрам»). Регистры управления таймером (TCCRnA/B/C) являются 8-разрядными регистрами, поэтому доступ к ним со стороны ЦПУ не связан с какими-либо ограничениями. Все сигналы запросов на прерывание представлены в регистре флагов прерываний таймеров (TIFR) и расширенном регистре флагов прерываний таймеров (ETIFR). Все прерывания индивидуально маскируются регистром масок прерываний таймеров (TIMSK) и расширенным регистром масок прерываний (ETIMSK). Регистры (E)TIFR

и (E)TIMSK не представлены на структурной схеме, так как они используются также и другими таймерами микроконтроллера.

Таймер/счетчик может тактироваться от внутреннего тактового сигнала через предделитель или от внешнего источника тактового сигнала, подключенного к выводу Tn. Блок выбора синхронизации позволяет выбрать тактовый источник и фронт, по которому будет изменяться состояние таймера/счетчика. Если тактовый источник не задан, то таймер/счетчик находится в неактивном состоянии. Сигнал на выходе блока выбора синхронизации является тактовым сигналом таймера (clk_{Tn}).

Значение регистров сравнения (OCRnA/B/C) непрерывно сравнивается со значением счетчика. Результат сравнения может использоваться для генерации прямоугольных импульсов с ШИМ или с переменной частотой на выходах сравнения OCnA/B/C. См. также 3.10.6 «Блоки сравнения». При совпадении значений сравниваемых регистров устанавливается соответствующий флаг прерывания (OCFnA/B/C), который в свою очередь может служить источником прерывания.

Регистр захвата позволяет запомнить состояние таймера/счетчика при возникновении заданного внешнего события (фронт внешнего сигнала) либо на входе захвата фронта ICPn, либо на выводах аналогового компаратора (см. подраздел 3.17 «Аналоговый компаратор»). В блоке захвата предусмотрен модуль цифровой фильтрации (подавитель шумов) для снижения риска срабатывания схемы от помех.

Верхний предел или максимальное значение таймера/счетчика в зависимости от режима работы таймера могут определяться либо значением в OCRnA или ICRn, либо могут иметь фиксированные значения. Если регистр OCRnA задает верхний предел счета в режиме ШИМ, то он не может использоваться для генерации ШИМ-сигналов. Однако верхний предел в этом случае имеет двойную буферизацию, тем самым допуская изменение его значения в процессе работы. Если верхний предел счета является фиксированным значением, то альтернативно можно использовать регистр ICRn, освобождая регистр OCRnA для функции широтно-импульсной модуляции.

Определения

В таблице 3.57 представлены определения, которые широко используются на протяжении всего подраздела документа.

Таблица 3.57 – Определения

Нижний предел	Счетчик достигает нижнего предела, когда его значение становится равным нулю (0x0000)
Максимум	Счетчик достигает максимума, когда его значение становится равным 0xFFFF (десятичное число 65535)
Верхний предел	Счетчик доходит до верхнего предела, когда он достигает наивысшего значения в последовательности счета. Верхнему пределу может быть присвоено одно из фиксированных значений 0x00FF, 0x01FF, 0x03FF или же значение, хранящееся в регистре OCRnA или ICRn. Присваиваемое значение зависит от режима работы

3.10.2 Доступ к 16-разрядным регистрам

Регистры TCNTn, OCRnA/B/C и ICRn являются 16-разрядными, поэтому ЦПУ может получить к ним доступ через 8-разрядную шину данных с помощью двух инструкций чтения или двух инструкций записи. Каждый 16-разрядный таймер имеет отдельный 8-разрядный регистр для временного хранения старшего байта данных. Поэтому в пределах одного таймера все 16-разрядные регистры совместно используют один и тот же временный регистр. Получение доступа к младшему байту инициирует 16-разрядную операцию чтения или записи. Если выполняется запись младшего байта 16-разрядного регистра, то за один такт ЦПУ одновременно записываются и младший байт, и старший байт из

временного регистра. Если выполняется чтение младшего байта 16-разрядного регистра, то за один такт ЦПУ параллельно с чтением младшего байта происходит копирование старшего байта 16-разрядного регистра во временный регистр.

Не все 16-разрядные регистры используют временный регистр для копирования старшего байта. Чтение 16-разрядных регистров OCRnA/B/C не связано с использованием временного регистра.

Таким образом, чтобы записать данные в 16-разрядный регистр, необходимо сначала записать старший байт, а затем младший. А при чтении 16-разрядного регистра, наоборот, сначала должен считываться младший байт, а затем старший.

Ниже приведены примеры кода на Ассемблере и Си, показывающие как осуществлять доступ к 16-разрядным регистрам таймера. В примерах предполагается, что во время обновления временного регистра не возникает прерываний. Аналогично может быть выполнен доступ к регистрам OCRnA/B/C и ICRn.

Пример кода на Ассемблере:

```
...
; Установка TCNTn = 0x01FF
ldi r17,0x01
ldi r16,0xFF
out TCNTnH,r17
out TCNTnL,r16
; Чтение TCNTn в r17:r16
in r16,TCNTnL
in r17,TCNTnH
...
```

Пример кода на Си:

```
unsigned int i;
...
/* Установка TCNTn = 0x01FF */
TCNTn = 0x1FF;
/* Чтение TCNTn в i */
i = TCNTn;
...
```

Примечания

1 При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

2 В примере на Ассемблере значение TCNTn возвращается в пару регистров r17–r16.

Важно отметить, что операции доступа (чтение и запись) к 16-разрядным регистрам являются неделимыми операциями. Если после выполнения первой инструкции происходит прерывание, и код прерывания обновляет временный регистр при наличии доступа к этому же или любому другому регистру в пределах одного таймера, то выполнение второй инструкции приведет к некорректному результату. Поэтому, когда и в основной программе и в прерываниях происходит обновление временного регистра, то в основной программе перед инициализацией доступа к 16-разрядному регистру необходимо запретить прерывания.

В следующем примере показано, как корректно выполнить неделимую операцию чтения регистра TCNTn без риска изменения содержимого временного регистра при возникновении прерывания. Чтение регистров OCRnA/B/C и ICRn можно осуществить, применяя этот же принцип.

Пример кода на Ассемблере:

```
TIM16_ReadTCNTn:
; Запоминание состояния общего флага прерываний
in r18,SREG
; Запрет прерываний
cli
; Чтение TCNTn в r17:r16
in r16,TCNTnL
in r17,TCNTnH
; Восстановление состояния общего флага прерываний
out SREG,r18
ret
```

Пример кода на Си:

```
unsigned int TIM16_ReadTCNTn( void )
{
unsigned char sreg;
unsigned int i;
/* Запоминание состояния общего флага прерываний */
sreg = SREG;
/* Запрет прерываний */
_disable_interrupt();
/* Чтение  TCNTn в i */
i = TCNTn;
/* Восстановление состояния общего флага прерываний */
SREG = sreg;
return i;
}
```

Примечания

1 При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

2 В примере на Ассемблере значение TCNTn возвращается в пару регистров r17–r16.

В следующем примере показано, как корректно выполнить неделимую операцию записи в регистр TCNTn без риска изменения содержимого временного регистра при возникновении прерывания. Запись в регистры OCRnA/B/C и ICRn можно осуществить, применяя этот же принцип.

Пример кода на Ассемблере:

```
TIM16_WriteTCNTn:
; Запоминание состояния общего флага прерываний
in r18,SREG
; Запрет прерываний
cli
; Копирование  TCNTn в r17:r16
out TCNTnH,r17
out TCNTnL,r16
; Восстановление состояния общего флага прерываний
out SREG,r18
ret
```

Пример кода на Си:

```
void TIM16_WriteTCNTn( unsigned int i )
{
unsigned char sreg;
unsigned int i;
/* Запоминание состояния общего флага прерываний */
sreg = SREG;
/* Запрет прерываний */
_disable_interrupt();
/* Копирование TCNTn в i */
TCNTn = i;
/* Восстановление состояния общего флага прерываний */
SREG = sreg;
}
```

Примечания

1 При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

2 В примере на Ассемблере требуется, чтобы пара регистров r17–r16 содержала значение, которое должно быть записано в TCNTn.

Повторное использование временного регистра для хранения старшего байта данных

Если выполняется запись в несколько 16-разрядных регистров, и при этом значение старшего байта одинаково для всех записываемых регистров, то достаточно однократно выполнить запись старшего байта. Однако обратите внимание, что и в данном случае также применяется вышеупомянутое правило неделимой операции.

3.10.3 Источники синхронизации таймера/счетчика

Таймер/счетчик может тактироваться от внутреннего или внешнего источника синхронизации. Источник тактового сигнала выбирается соответствующей схемой микроконтроллера под управлением битов выбора синхронизации (CSn2:0), которые находятся в регистре B управления таймером/счетчиком (TCCRnB). Более подробная информация относительно тактовых источников и предделителя приведена в подразделе 3.11 «Предделители таймера/счетчика 1, таймера/счетчика 2 и таймера/счетчика 3».

3.10.4 Блок счетчика

Основным элементом 16-разрядного таймера/счетчика является программируемый двунаправленный 16-разрядный счетчик. На рисунке 3.48 представлена структурная схема счетчика и окружающих его элементов.

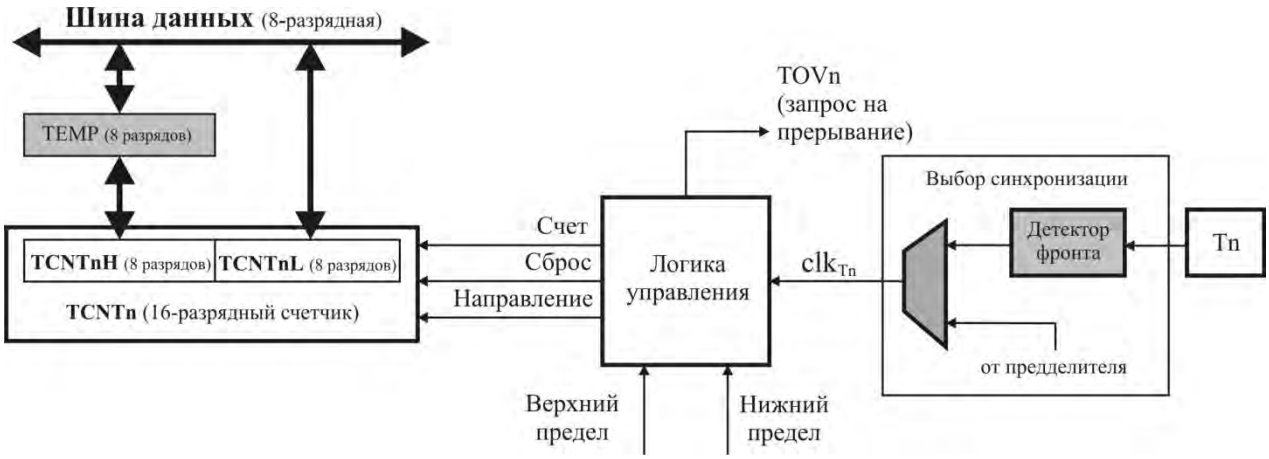


Рисунок 3.48 – Структурная схема счетчика

Описание внутренних сигналов:

Счет – инкремент или декремент значения TCNTn на единицу.

Направление – выбор прямого счета (инкрементирование) или обратного счета (декрементирование).

Сброс – сброс TCNTn (установка всех разрядов в логический ноль).

clk_{Tn} – синхронизация таймера/счетчика.

Верхний предел – показывает, что регистр TCNTn достиг максимального значения.

Нижний предел – показывает, что регистр TCNTn достиг минимального значения (нуля).

Содержимое 16-разрядного счетчика разбито на две 8-разрядные ячейки, расположенные в памяти ввода-вывода: старший байт счетчика (TCNTnH), в котором хранятся старшие 8 разрядов счетчика, и младший байт счетчика (TCNTnL), в котором хранятся младшие 8 разрядов. ЦПУ не имеет прямого доступа к регистру TCNTnH. Когда ЦПУ обращается к ячейке ввода-вывода, содержащей старший байт счетчика TCNTnH, то фактически он получает доступ к временному регистру (TEMP). Во временный регистр копируется значение TCNTnH, когда выполняется чтение регистра TCNTnL. А в регистр TCNTnH копируется содержимое временного регистра, когда выполняется запись в TCNTnL. Такой механизм реализован для считывания/записи 16-разрядного значения счетчика за один такт ЦПУ через 8-разрядную шину данных. Следует обратить внимание, что в некоторых случаях запись в регистр TCNTn во время счета будет давать непредсказуемый результат. Такие случаи описаны в последующих пунктах.

В зависимости от используемого режима работы счетчик будет сбрасываться, инкрементироваться или декрементироваться на каждом такте синхронизации таймера clk_{Tn}. Сигнал clk_{Tn} может быть сгенерирован внешним или внутренним источником, который определяется битами выбора синхронизации (CSn2–0). Если тактовый источник не задан (CSn2–0 = 0), то таймер останавливается. Однако содержимое TCNTn остается доступным для ЦПУ независимо от наличия сигнала clk_{Tn}. Запись ЦПУ блокирует все операции счета и сброса счетчика (запись имеет более высокий приоритет).

Счетная последовательность определяется значением битов режима работы таймера (WGMn3–0), расположенных в регистрах А и В управления таймером/счетчиком (TCCRnA и TCCRnB). Имеется четкая связь между счетной последовательностью счетчика и формой генерируемого на выходе OCnx сигнала. Более подробная информация об этом приведена ниже в пункте 3.10.8 «Режимы работы».

Флаг переполнения таймера/счетчика (TOVn) устанавливается согласно режиму работы, выбираемому битами WGMn3–0. Флаг TOVn может использоваться для генерации прерывания ЦПУ.

3.10.5 Блок захвата

Таймер/счетчик содержит блок захвата, который запоминает состояние счетчика при появлении внешнего события, тем самым определяя время его возникновения. В качестве события/событий выступает внешний сигнал, подаваемый на вывод ICPn. Альтернативно, только для таймера/счетчика 1, может использоваться аналоговый компаратор в качестве источника внешнего события. Результат захвата состояния таймера может использоваться для вычисления частоты, скважности импульсов и других параметров импульсных сигналов. Альтернативно это значение может использоваться для создания журнала событий.

Структурная схема блока захвата представлена на рисунке 3.49. Некоторые элементы на структурной схеме, которые не относятся напрямую к блоку захвата, залиты серым цветом. Символ «n» в наименованиях битов и регистров заменяет номер таймера/счетчика.

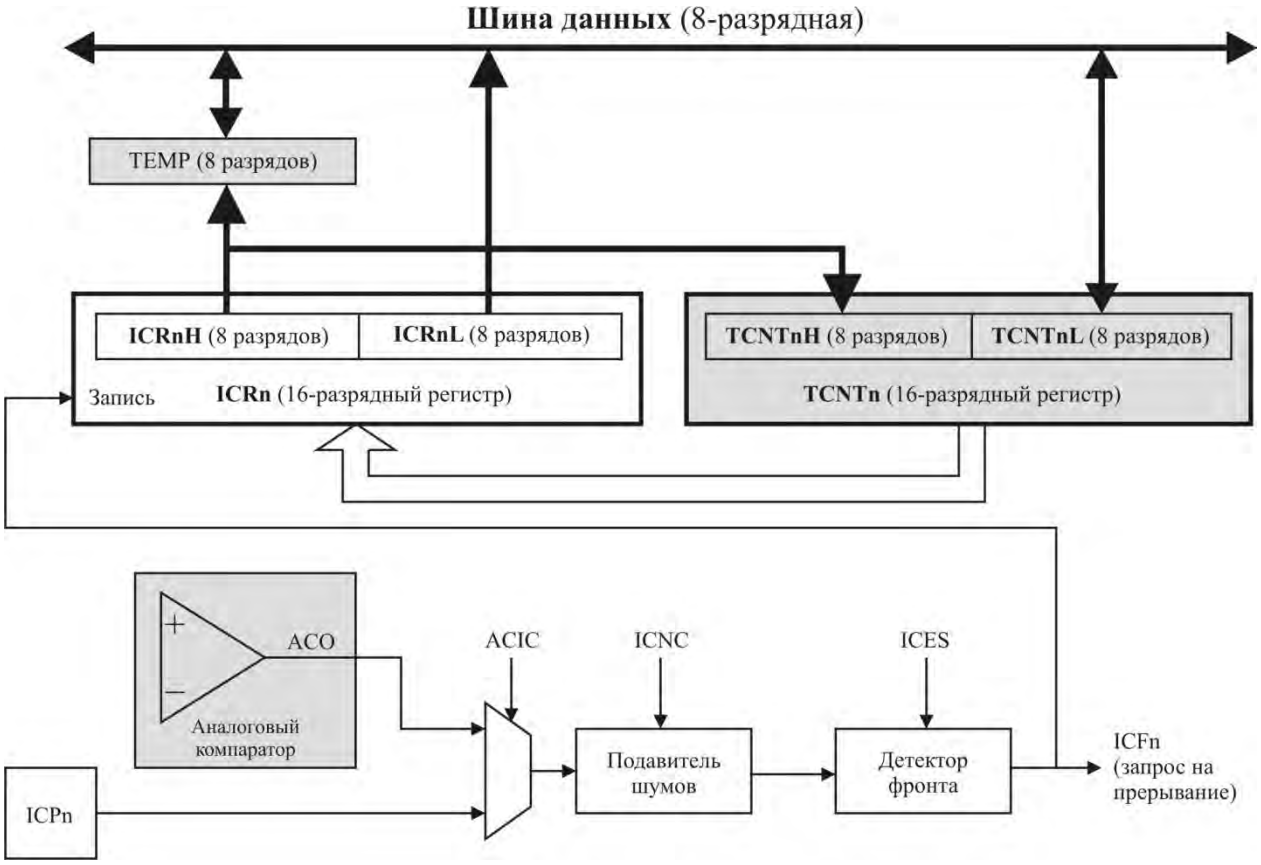


Рисунок 3.49 – Структурная схема блока захвата

Примечание – Выход аналогового компаратора (ACO) только у таймера/счетчика 1 может выступать в качестве сигнала захвата. У таймера/счетчика 3 эта возможность отсутствует.

Если на входе захвата (ICPn) или альтернативно на выходе аналогового компаратора (ACO) возникает изменение логического уровня (событие), которое соответствует установкам детектора фронта, то выполняется захват состояния таймера. При этом 16-разрядное значение содержимого таймера (TCNTn) помещается в регистр захвата (ICRn). Флаг захвата (ICFn) устанавливается на том же такте ЦПУ, на котором произошло копирование значения TCNTn в ICRn. Если TICIEn = 1, то флаг захвата генерирует прерывание. Флаг ICFn автоматически сбрасывается при переходе на вектор прерывания. Альтернативно, флаг ICFn можно сбросить программно путем записи логической единицы в соответствующий разряд регистра флагов.

Считывание 16-разрядного значения регистра захвата (ICRn) выполняется чтением сначала младшего байта (ICRnL), а затем старшего байта (ICRnH). При выполнении команды чтения младшего байта значение старшего байта автоматически копируется во временный регистр. Если ЦПУ выполняет команду чтения регистра ICRnH, то фактически считывается содержимое временного регистра.

Запись в регистр ICRn возможна только тогда, когда выбран режим работы таймера, в котором значение регистра ICRn определяет верхний предел счета. В этом случае необходимо выполнить соответствующую установку битов режима работы (WGMn3–0), а только затем осуществлять запись значения верхнего предела в регистр ICRn. Запись 16-разрядного значения в регистр ICRn выполняется путем записи сначала старшего байта в ICRnH, а только затем младшего байта в ICRnL (см. также 3.10.2 «Доступ к 16-разрядным регистрам»).

Источник срабатывания механизма захвата

Основным источником, инициирующим захват состояния таймера/счетчика, является вывод захвата (ICPn). Таймер/счетчик 1 также альтернативно может использовать выход аналогового компаратора в качестве источника сигнала захвата. Для этого необходимо установить бит разрешения захвата аналоговым компаратором (ACIC) в регистре состояния и управления аналоговым компаратором (ACSR). Учтите, что изменение источника сигнала захвата может привести к возникновению ложного захвата. Поэтому после изменения источника должен быть сброшен флаг захвата.

Входные каскады, связанные с выводом захвата (ICPn) и выходом аналогового компаратора (ACO), выполнены по аналогии с каскадом, связанным с выводом Tn (см. рисунок 3.60). Детекторы фронта также идентичны. Однако если разрешена работа подавителя шумов, то последовательно перед детектором фронта включается дополнительная логика, которая увеличивает задержку на четыре такта ЦПУ. Обратите внимание, что вход подавителя шумов и детектора фронта всегда разрешен, если таймер/счетчик не находится в режиме, где регистр ICRn определяет вершину счета.

Захват можно инициировать программно путем управления настройками порта на выводе ICPn.

Подавитель шумов

Подавитель шумов позволяет повысить помехозащищенность за счет использования простой схемы цифровой фильтрации. Вход подавителя шумов контролируется четырьмя выборками, и все четыре выборки должны быть идентичными для изменения выхода, который в свою очередь используется детектором фронта.

Работа подавителя шумов разрешается путем установки бита разрешения подавления шумов на входе захвата (ICNCn) в регистре B управления таймером/счетчиком (TCCRnB). С разрешением работы подавителя шумов вводится задержка на четыре такта между возникновением изменения входного уровня и обновлением регистра ICRn. Подавитель шумов использует системную синхронизацию и поэтому не зависит от настройки предделителя.

Использование блока захвата

Основная проблема при использовании блока захвата заключается в необходимости выделения достаточного процессорного времени на обработку возникшего события. Время между двумя событиями является критическим параметром. Если процессор не успевает считать содержимое ICRn до момента возникновения следующего события, то в регистр ICRn будет записано новое значение поверх старого. В этом случае результат захвата будет некорректным.

Если используется прерывание по захвату, то в процедуре обработки прерываний регистр ICRn необходимо считать как можно раньше. Несмотря на то, что прерывание по захвату имеет относительно высокий приоритет, время реакции на прерывание будет зависеть от максимального числа тактов, требуемых для выполнения процедуры обработки любого другого прерывания.

Применение блока захвата не рекомендуется, если таймер используется в режиме, когда значение верхнего предела счета (разрешающая способность) активно изменяется в процессе работы.

Для измерения скважности (или коэффициента заполнения) импульсов необходимо после каждого захвата изменять фронт, по которому происходит захват. Данное изменение необходимо выполнять как можно раньше после считывания регистра ICRn. После изменения фронта необходимо программно сбросить флаг захвата ICFn путем записи логической единицы в соответствующий разряд регистра флагов. Для измерения только частоты сброс флага ICFn не требуется (если используется программа обработки прерываний).

3.10.6 Блоки сравнения

16-разрядный цифровой компаратор непрерывно сравнивает значение TCNTn со значением регистра сравнения (OCRnx). Если значение TCNT равно OCRnx, то компаратор формирует сигнал совпадения. На последующем такте таймера установится флаг сравнения (OCFnx). Если OCIEnx = 1, то установка флага сравнения приведет к генерации прерывания по результату сравнения. Флаг OCFnx автоматически сбрасывается после перехода на вектор обработки прерывания. Альтернативно, флаг OCFnx можно сбросить программно путем записи логической единицы в соответствующий разряд регистра флагов. Сигнал совпадения используется формирователем сигналов для генерации выходного импульса в соответствии с режимом, выбранным с помощью битов режима работы таймера (WGMn3–0) и битов режима формирования импульсов (COMnx1–0). Сигналы «Верхний предел» и «Нижний предел» используются формирователем сигналов для обработки особых случаев задания экстремальных значений в некоторых режимах работы (см. 3.10.8 «Режимы работы»).

У канала сравнения A имеется своя отличительная особенность, которая позволяет задать верхний предел счета (т.е. разрешающую способность счетчика). В дополнение к разрешающей способности верхний предел определяет период формируемых импульсов.

На рисунке 3.50 показана структурная схема блока сравнения. Символ «n» в обозначениях битов и регистров заменяет номер таймера/счетчика (1 или 3), а «x» заменяет наименование канала сравнения (A/B/C). Некоторые элементы на структурной схеме, которые не относятся напрямую к блоку сравнения, залиты серым цветом.

Если задан любой из 12 режимов широтно-импульсной модуляции, то в этом случае регистр OCRnx содержит двойную буферизацию. Если таймер работает в нормальном режиме или режиме сброса при совпадении (СТС), то двойная буферизация отключается. Двойная буферизация синхронизирует обновление регистра значения сравнения OCRnx по достижении либо верхнего, либо нижнего предела счетной последовательности. Такая синхронизация предотвращает возможность возникновения несимметричных ШИМ-импульсов нечетной длины, гарантируя тем самым отсутствие импульсных помех на выходе.

Доступ к регистру OCRnx может показаться сложным, но это не так. Если включена двойная буферизация, то ЦПУ фактически осуществляет доступ к буферному регистру OCRnx. Если же двойная буферизация отключена, то ЦПУ обращается непосредственно к регистру OCRnx. Содержимое регистра OCR1x (в том числе и буферного) может измениться только путем непосредственной записи в него (таймер/счетчик не обновляет содержимое данного регистра автоматически, как регистры TCNTn и ICRn).

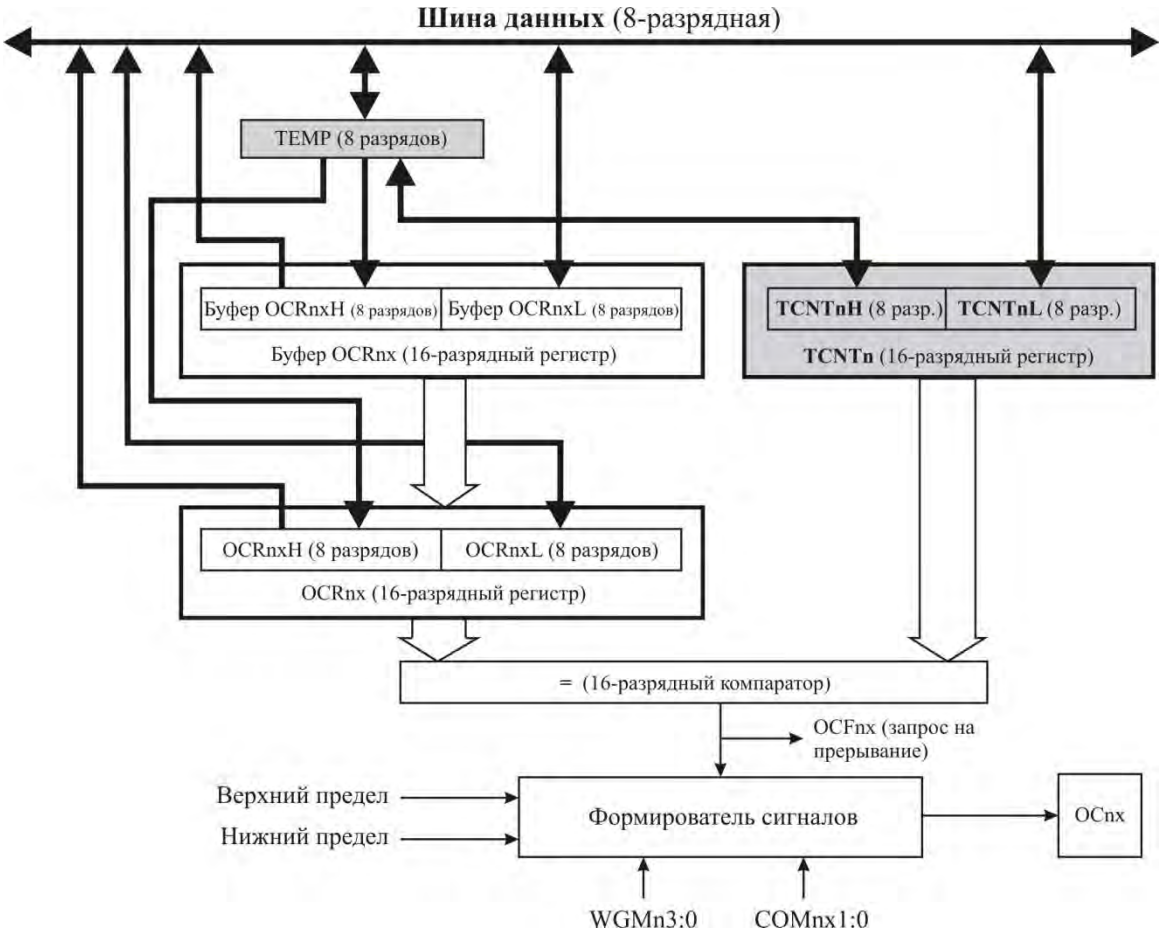


Рисунок 3.50 – Структурная схема блока сравнения

Таким образом, OCRnx считывается напрямую, а не через временный регистр старшего байта. Тем не менее, это хороший прием – считывать сначала младший байт, как при доступе к другим 16-разрядным регистрам. Запись регистров OCRnx необходимо выполнять через временный регистр TEMP, поскольку сравнение всех 16 битов производится непрерывно. Первым должен быть записан старший байт OCRnxH. Когда ЦПУ выполняет запись ячейки старшего байта, временный регистр обновляется в соответствии с новым значением. Затем, когда выполняется запись младшего байта (OCRnxL), параллельно копируется содержимое временного регистра в старшие 8 разрядов либо буферного регистра OCRnx, либо регистра значения сравнения OCRnx, тем самым обновляя все 16 битов за один такт ЦПУ (см. также 3.10.2 «Доступ к 16-разрядным регистрам»).

Установка принудительного совпадения

В режимах формирования сигналов без ШИМ результат сравнения компаратора может быть установлен путем записи логической единицы в бит установки принудительного совпадения FOCnx. Установка принудительного совпадения компаратора не приводит к установке флага OCFnx или перезагрузке/сбросу таймера, но обновляет состояние вывода OCnx (который будет устанавливаться, сбрасываться или переключаться в зависимости от выбранной установки битов COMnx1–0), как если бы произошло реальное событие совпадения.

Блокирование результата сравнения путем записи в TCNTn

Если ЦПУ осуществляет запись в регистр TCNTn, то результат сравнения будет блокироваться на следующем такте синхронизации таймера, даже если таймер остановлен.

Данная функция позволяет устанавливать в регистре OCRnx то же значение, что и в TCNTn, без генерации запроса на прерывание, когда включена синхронизация таймера/счетчика.

Использование блока сравнения

Поскольку запись в TCNTn блокирует действие сравнения на один такт синхронизации таймера в любом режиме работы, то когда изменяется TCNTn при использовании любого канала сравнения (независимо от того, работает таймер/счетчик или нет), необходимо учесть следующие особенности. Если в регистр TCNTn записано значение, равное значению OCRnx, то игнорирование совпадения приведет к генерации сигнала неправильной формы. Следует избегать записи в TCNTn значения, равного верхнему пределу, в ШИМ-режимах с переменным значением верхнего предела. В этом случае совпадение по достижении верхнего предела игнорируется, и счет продолжается до 0xFFFF. Аналогично следует избегать записи в TCNTn значения, равного нижнему пределу, когда счетчик работает как вычитающий.

Установка OCnx должна быть выполнена перед настройкой линии на вывод в регистре направления данных. Самый легкий путь установки значения OCnx – использование бита установки принудительного совпадения (FOCnx) в нормальном режиме. Регистр OCnx сохраняет свое значение, даже если происходит изменение между режимами формирования сигналов.

Учтите, что биты COMnx1 и COMnx0 не содержат схемы двойной буферизации и на любые изменения реагируют мгновенно.

3.10.7 Блок формирования выходного сигнала

Биты задания режима формирования выходного сигнала (COMnx1–0) имеют двойное назначение. С одной стороны биты COMnx1–0 используются формирователем сигнала и определяют, какое логическое состояние должно быть на выходе OCnx при возникновении следующего совпадения. С другой стороны, эти биты используются для разрешения/запрета альтернативной функции вывода порта OCnx. На рисунке 3.51 представлена упрощенная логическая схема, на которую воздействуют биты COMnx1–0. На рисунке показаны только те регистры управления портом ввода-вывода (DDR и PORT), на которые оказывают влияние биты COMnx1–0. При упоминании состояния OCnx речь идет о внутреннем регистре OCnx, а не о выводе OCnx. Если происходит системный сброс, то регистр OCnx принимает нулевое состояние.

Функция порта ввода-вывода общего назначения переопределяется функцией выхода формирователя сигнала OCnx, если установлен хотя бы один из битов COMnx1–0. Однако, контроль направления вывода OCnx (вход или выход) в этом случае остается за соответствующим регистром направления данных (DDR). Бит регистра направления данных для вывода OCnx (DDR_OCnx) необходимо определить, как выход, чтобы значение регистра OCnx появилось на выводе OCnx. Управление вводом альтернативной функции не зависит от режима работы таймера за некоторыми исключениями (см. таблицы 3.58–3.60).

Схемотехника выходной логики позволяет инициализировать состояние регистра OCnx перед разрешением настройки вывода OCnx в качестве выхода. Обратите внимание, что некоторые установки битов COMnx1–0 зарезервированы для определенных режимов работы (см. 3.10.10 «Описание регистров 16-разрядного таймера/счетчика»). Установки битов COMnx1–0 не оказывают никакого влияния на работу блока захвата.

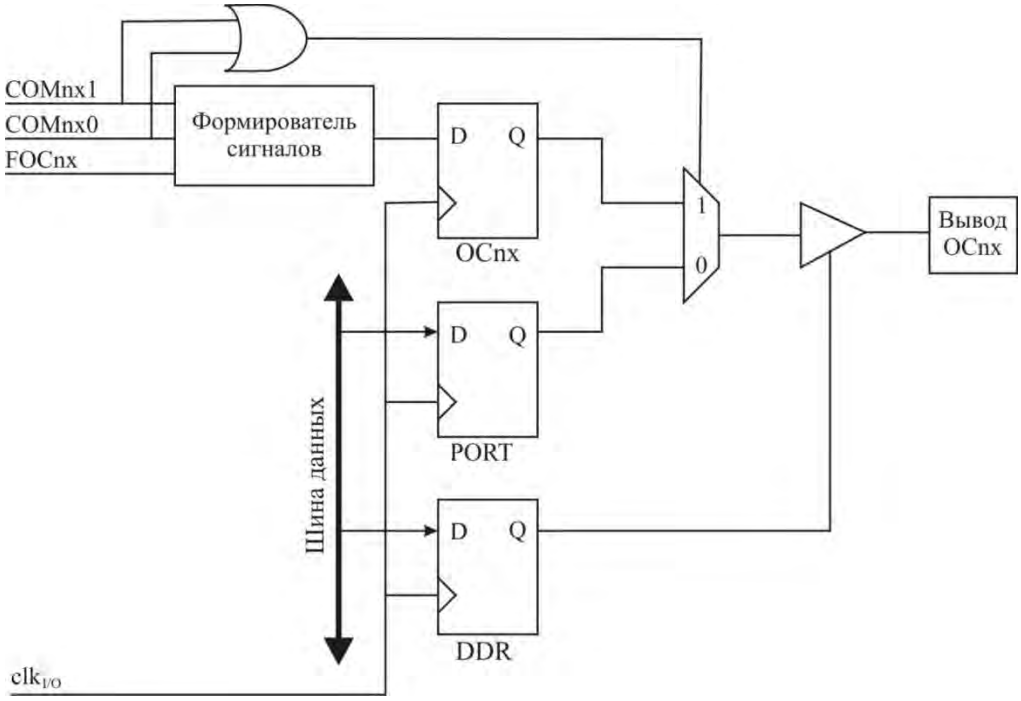


Рисунок 3.51 – Схема блока формирования выходного сигнала

Режим сравнения и формирование сигналов

Формирователь сигналов использует биты COMnx1–0 по-разному в нормальном режиме, режиме сброса таймера при совпадении и режиме с ШИМ. Во всех режимах установка COMnx1–0 = 0 сообщает формирователю сигналов, что никакое действие в регистре OCnx не будет выполнено при возникновении следующего совпадения. В таблице 3.58 описаны действия при сравнении в режимах без ШИМ. Аналогичная информация для режима с быстрой ШИМ приведена в таблице 3.59, а для режимов с ШИМ с фазовой коррекцией и с фазовой и частотной коррекцией – в таблице 3.60.

Изменение состояния битов COMnx1–0 окажет эффект при возникновении первого совпадения после записи этих битов. В режимах без ШИМ можно принудительно получить незамедлительный эффект, используя бит FOCnx.

3.10.8 Режимы работы

Режим работы, то есть поведение таймера/счетчика и линий выходных сигналов, определяется комбинацией битов режима работы таймера (WGMn3–0) и битов режима формирования выходного сигнала (COMnx1–0). Биты COM01–0 не влияют на последовательность счета. Биты режима работы таймера WGM01–0 оказывают влияние на последовательность счета. Биты COMnx1 и COMnx0 определяют, должен ли выход ШИМ быть инвертированным или нет (широтно-импульсная модуляция с инверсией или без). Для режимов без ШИМ биты COMnx1–0 определяют, какое действие необходимо осуществить при выполнении условия совпадения: установить, сбросить или переключить выход (см. также 3.10.7 «Блок формирования выходного сигнала»).

Более подробная информация по временным диаграммам работы приведена в 3.10.9 «Временные диаграммы 16-разрядного таймера/счетчика».

Нормальный режим работы

Самым простым режимом работы является нормальный режим (WGMn3–0 = 0). В данном режиме счетчик всегда работает как суммирующий (инкрементирующий), при этом

сброс счетчика не выполняется. Счетчик просто переполняется, когда достигает своего максимального 16-разрядного значения (верхний предел равен 0xFFFF), и затем перезапускается, начиная счет от нижнего предела (0x0000). В нормальном режиме работы флаг переполнения таймера/счетчика TOVn будет установлен на том же такте синхронизации, когда TCNTn примет нулевое значение. Флаг TOVn в данном случае ведет себя подобно семнадцатому биту, за исключением того, что он только устанавливается и не сбрасывается. Однако программно это свойство может быть использовано для повышения разрешающей способности таймера, если использовать прерывание по переполнению таймера, при возникновении которого флаг TOVn сбрасывается автоматически. Для нормального режима работы не существует каких-либо особых ситуаций, связанных с записью нового состояния счетчика, когда требовалось бы учесть меры предосторожности.

В нормальном режиме можно использовать блок захвата. Однако при этом необходимо следить, чтобы максимальный интервал времени между возникновениями внешних событий не превышал разрешающую способность счетчика. Если интервал между событиями слишком большой, то следует использовать прерывание по переполнению таймера или предделитель, чтобы увеличить разрешающую способность для блока захвата.

Блоки сравнения могут использоваться для генерации прерываний в определенный момент. Не рекомендуется использовать выход OCnx для генерации сигналов в нормальном режиме работы, так как в этом случае будет затрачена значительная часть процессорного времени.

Режим сброса таймера при совпадении (CTC)

В режиме сброса таймера при совпадении (WGMn3–0 = 4 или 12) разрешающая способность таймера задается регистрами OCRnA или ICRn. В режиме CTC счетчик сбрасывается в ноль, когда его значение (TCNTn) совпадает либо со значением регистра OCRnA (WGMn3–0 = 4), либо со значением ICRn (WGMn3–0 = 12). Значение регистра OCRnA или ICRn определяет верхний предел счета, а, следовательно, и разрешающую способность таймера. В данном режиме обеспечивается более широкий диапазон регулировки частоты генерируемых прямоугольных импульсов. Он также упрощает операцию счета внешних событий.

Временная диаграмма работы таймера в режиме CTC показана на рисунке 3.52. Значение счетчика (TCNTn) увеличивается до тех пор, пока не возникнет совпадение либо со значением OCRnA, либо со значением ICRn, а затем счетчик (TCNTn) сбрасывается.

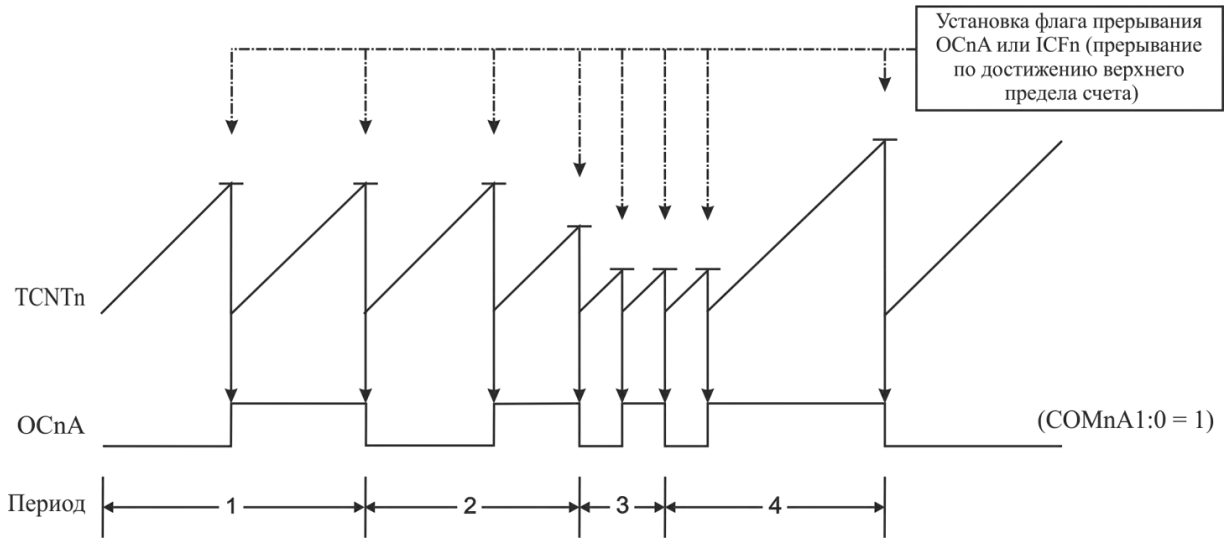


Рисунок 3.52 – Временная диаграмма для режима сброса таймера при совпадении

Прерывание может генерироваться всякий раз, когда счетчик достигает верхнего предела, с помощью либо флага OCFnA, либо ICFn в зависимости от регистра, определяющего верхний предел счета. Если прерывание разрешено, то процедура обработки прерывания может использоваться для обновления верхнего предела счета. Однако, задание верхнего предела счета, близкого к значению нижнего предела, когда счетчик работает без предделения или с малым значением предделения, необходимо выполнять с особой осторожностью, поскольку в режиме CTC нет двойной буферизации. Если значение, записанное в OCRnA или ICRn, меньше текущего значения TCNTn, то счетчик пропустит событие совпадения. В таком случае счетчик должен будет досчитать до своего максимального значения (0xFFFF) и перезапуститься, начав счет с 0x0000, пока он не достигнет нового значения в регистре OCRnA или ICRn. Во многих случаях возникновение такой ситуации не желательно. В качестве альтернативы необходимо использовать режим быстрой ШИМ, где регистр OCRnA определяет верхний предел счета (WGMn3–0 = 15), т.к. в этом случае OCRnA имеет двойную буферизацию.

Для генерации сигнала в режиме CTC выход OCnA может использоваться для изменения логического уровня при каждом совпадении, для чего необходимо задать режим переключения (COMnA1–0 = 1). Значение OCnA будет присутствовать на выводе порта, только если для данного вывода задано выходное направление (DDR_OCnA = 1). Сгенерированный сигнал будет иметь максимальную частоту, равную $f_{OCnA} = f_{clk\_I/O}/2$, когда регистр OCRnA имеет нулевое значение (0x0000). Частота сигналов определяется по формуле

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

где переменная N задает коэффициент деления предделителя (1, 8, 64, 256 или 1024).

Как и в нормальном режиме работы, флаг TOVn устанавливается на том же самом такте таймера, когда значение счетчика меняется с 0xFFFF на 0x0000.

Режим быстрой широтно-импульсной модуляции

Режим быстрой широтно-импульсной модуляции (WGMn3–0 = 5, 6, 7, 14 или 15) позволяет генерировать высокочастотные ШИМ-сигналы. От других режимов ШИМ данный режим отличается однонаправленностью счета. Счетчик считает от нижнего предела до верхнего предела, а затем перезапускается, начиная счет от нижнего предела. В неинвертирующем режиме вывод OCnx сбрасывается при совпадении значений TCNTn и OCRnx и устанавливается при достижении нижнего предела счета. В инвертирующем режиме вывод OCnx устанавливается при совпадении TCNTn и OCRnx и сбрасывается при достижении нижнего предела счета. Благодаря однонаправленности счета рабочая частота для режима быстрой ШИМ может быть в два раза выше по сравнению с режимами ШИМ с фазовой коррекцией и с фазовой и частотной коррекцией, где используется двунаправленный счет. Возможность генерации высокочастотных ШИМ-сигналов делает использование данного режима полезным в задачах стабилизации питания, коррекции и цифро-аналогового преобразования. Высокая частота при этом позволяет использовать внешние элементы физически малых размеров (катушки индуктивности, конденсаторы), снижая тем самым общую стоимость системы.

Разрешающая способность ШИМ может быть фиксированной 8, 9 или 10 разрядов или может задаваться либо регистром ICRn, либо OCRnA. Минимальная разрешающая способность равняется двум разрядам (установка значения 0x0003 в регистре ICRn или OCRnA), а максимальная – 16 разрядам (установка значения 0xFFFF в регистре ICRn или OCRnA). Разрешающую способность ШИМ в битах можно вычислить по формуле

$$R_{FPWM} = \frac{\log(Верхний\ предел + 1)}{\log(2)},$$

В режиме быстрой ШИМ счетчик инкрементируется, пока его значение не совпадет с одним из фиксированных значений 0x00FF, 0x01FF, 0x03FF (WGMn3–0 = 5, 6, 7) или со значением в регистре ICRn (WGMn3–0 = 14) или OCRnA (WGMn3–0 = 15). Затем счетчик сбрасывается на следующем такте синхронизации таймера. Временная диаграмма для режима быстрой ШИМ представлена на рисунке 3.53. Рисунок отображает режим быстрой ШИМ, когда для задания верхнего предела используется регистр OCRnA или ICRn. Значение TCNTn на временной диаграмме показано в виде гистограммы для иллюстрации однонаправленности счета. Диаграмма включает неинвертированный и инвертированный ШИМ-выходы. Короткими горизонтальными линиями обозначены положения на графике TCNTn, где совпадают значения регистров OCRnx и TCNTn. Флаг прерывания OCnx устанавливается при возникновении совпадения.

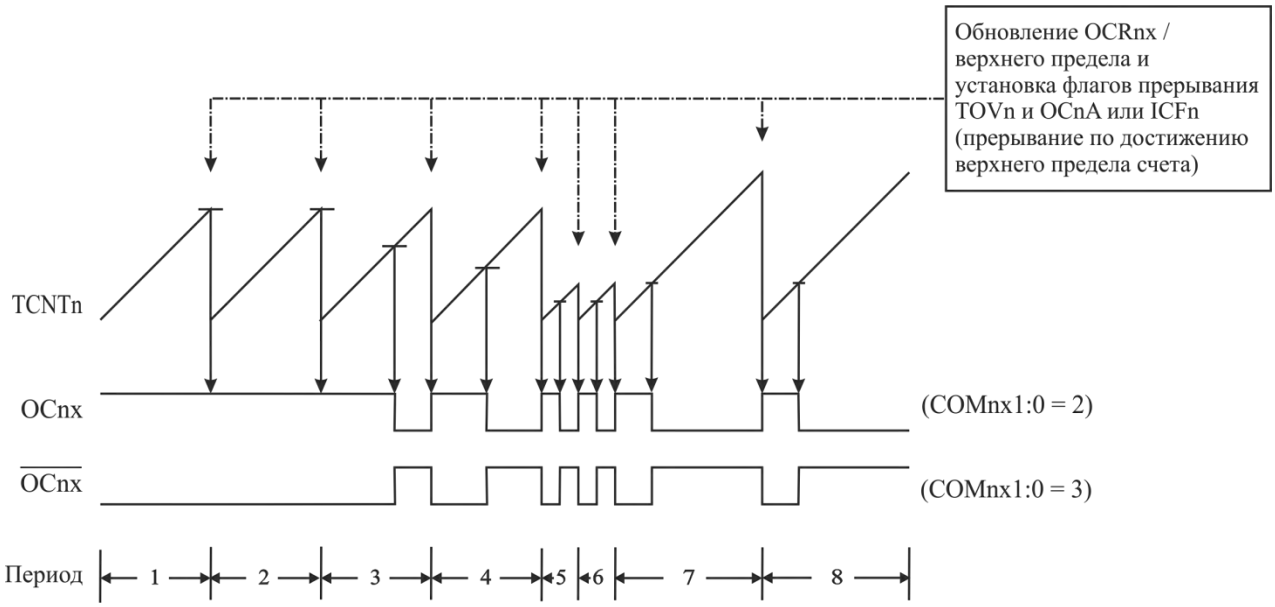


Рисунок 3.53 – Временная диаграмма для режима быстрой ШИМ

Флаг переполнения таймера/счетчика (TOVn) устанавливается всякий раз, когда счетчик достигает верхнего предела. Помимо этого, на том же самом такте таймера, когда устанавливается флаг TOVn, устанавливается также и флаг OCnA или ICFn, если для задания верхнего предела используется либо регистр OCRnA, либо ICRn, соответственно. Если разрешено одно из прерываний, то в процедуре обработки прерывания может быть выполнено обновление верхнего предела счета и значения сравнения.

При изменении значения верхнего предела счета программа должна гарантировать, что записываемое новое значение верхнего предела больше или равно значению каждого из регистров сравнения. Если значение верхнего предела будет меньше значения какого-либо регистра сравнения, то совпадение между TCNTn и OCRnx никогда не возникнет. Обратите внимание, что при использовании фиксированных значений верхнего предела неиспользуемые биты маскируются нулем, когда производится запись в любой из регистров OCRnx.

Процедура обновления регистра ICRn, когда он используется для задания верхнего предела, отличается от аналогичной процедуры с регистром OCRnA. Регистр ICRn не имеет двойной буферизации. Это означает, что если в ICRn записывается небольшое значение при работе счетчика без предделения или с малым значением предделения, то существует риск записи в регистр ICRn значения, которое окажется меньше текущего значения TCNTn. Как результат, в такой ситуации будет пропущено совпадение. В таком случае счетчик должен

будет считать до своего максимального значения (0xFFFF) и перезапуститься, начав счет с 0x0000, пока он не достигнет нового значения в регистре ICRn. Регистр OCRnA содержит схему двойной буферизации, поэтому его можно модифицировать в любой момент времени. При выполнении записи в регистр OCRnA записываемое значение фактически помещается в буферный регистр OCRnA. При совпадении значений TCNTn и верхнего предела счета информация из буферного регистра будет скопирована в регистр сравнения OCRnA на последующем такте таймера. Обновление регистра выполняется на том же такте, когда происходит сброс TCNTn и устанавливается флаг TOVn.

Использование регистра ICRn для задания верхнего предела целесообразно, когда верхний предел счета является константой. В этом случае также освобождается регистр OCRnA для генерации ШИМ-сигнала на выходе OCnA. Однако если базовая частота широтно-импульсной модуляции активно изменяется (за счет изменения значения верхнего предела), то в данной ситуации определенно лучше использовать регистр OCRnA для задания верхнего предела, поскольку он поддерживает двойную буферизацию.

В режиме быстрой ШИМ блоки сравнения позволяют генерировать ШИМ-сигналы на выводах OCnx. Если COMnx1–0 = 2, то задается ШИМ без инверсии выхода, а если COMnx1–0 = 3, то задается режим ШИМ с инверсией на выходе (см. таблицу 3.59). Фактическое значение OCnx можно наблюдать на выводе порта, только если для этого вывода задано выходное направление (DDR_OCnx). ШИМ-сигнал генерируется путем установки (или сброса) регистра OCnx при возникновении совпадения между OCRnx и TCNTn, а также путем сброса (или установки) регистра OCnx, когда счетчик сбрасывается (т.е. изменяет свое значение от верхнего предела счета до нижнего предела).

Частота ШИМ выходного сигнала определяется выражением

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + Верхний\ предел)},$$

где переменная N задает коэффициент деления предделителя (1, 8, 64, 256 или 1024).

Запись предельного значения в регистр OCRnx представляет собой особый случай при генерации ШИМ импульсов. Если значение OCRnx установить равным нижнему пределу (0x0000), то на выходе будет возникать узкий импульс на каждом (верхний предел + 1) такте таймера. Запись в OCRnx значения, равного верхнему пределу, приведет к установке постоянного уровня логической единицы или нуля на выходе (в зависимости от полярности выходного сигнала, выбранной битами COMnx1–0).

Если требуется генерация меандра (импульсной последовательности с коэффициентом заполнения 50 %) высокой частоты, необходимо использовать режим быстрой ШИМ с установкой битов COMnA1–0 = 1, что вызывает переключение (инвертирование) логического уровня на выходе OCnA при каждом совпадении. Это применимо, только если OCRnA используется для задания верхнего предела (WGMn3–0 = 15). Сгенерированный сигнал будет иметь максимальную частоту, равную $f_{OCnA} = f_{clk\_I/O}/2$, когда регистр OCRnA имеет нулевое значение (0x0000). Данная особенность подобна переключению OCnA в режиме CTC за исключением двойной буферизации, которая имеется в режиме быстрой ШИМ.

Режим широтно-импульсной модуляции с фазовой коррекцией

Режим широтно-импульсной модуляции с фазовой коррекцией (ШИМ ФК) (WGMn3–0 = 1, 2, 3, 10 или 11) предназначен для генерации ШИМ-сигнала с фазовой коррекцией и высокой разрешающей способностью. Режим ШИМ ФК, как и режим ШИМ с фазовой и частотной коррекцией, основан на двунаправленной работе таймера/счетчика. Счетчик выполняет счет циклически от нижнего предела (0x0000) до верхнего предела, а затем обратно от верхнего предела к нижнему пределу. В неинвертирующем режиме выход

OCnx сбрасывается при возникновении совпадения между TCNTn и OCRnx во время прямого счета и устанавливается при возникновении совпадения во время обратного счета. В инвертирующем режиме, наоборот, во время прямого счета происходит установка, а во время обратного – сброс выхода OCnx. При двунаправленном счете максимальная частота ШИМ-сигнала меньше, чем при однонаправленном счете. Однако, благодаря симметричности режимов ШИМ с двунаправленной работой, данные режимы предпочтительны при решении задач управления приводами.

Разрешающая способность ШИМ может быть фиксированной 8, 9, 10 разрядов или может задаваться либо регистром ICRn, либо OCRnA. Минимальная разрешающая способность равняется двум разрядам (установка значения 0x0003 в регистре ICRn или OCRnA), а максимальная – 16 разрядам (установка значения 0xFFFF в регистре ICRn или OCRnA). Разрешающую способность ШИМ в битах можно вычислить по формуле

$$R_{PCPWM} = \frac{\log(\textit{Верхний предел} + 1)}{\log(2)}.$$

В режиме ШИМ ФК счетчик инкрементируется, пока его значение не совпадет с одним из фиксированных значений 0x00FF, 0x01FF, 0x03FF (WGMn3–0 = 1, 2, 3) или со значением в регистре ICRn (WGMn3–0 = 10) или OCRnA (WGMn3–0 = 11). Затем счетчик изменяет направление счета. Значение TCNTn остается равным верхнему пределу в течение одного такта синхронизации таймера. Временная диаграмма для режима ШИМ ФК представлена на рисунке 3.54. Рисунок отображает режим ШИМ ФК, когда для задания верхнего предела используется регистр OCRnA или ICRn. Значение TCNTn на временной диаграмме показано в виде гистограммы для иллюстрации двунаправленности счета. Диаграмма включает неинвертированный и инвертированный ШИМ-выходы. Короткими горизонтальными линиями обозначены положения на графике TCNTn, в которых значения регистров OCRnx и TCNTn совпадают. Флаг прерывания OCnx устанавливается при возникновении совпадения.

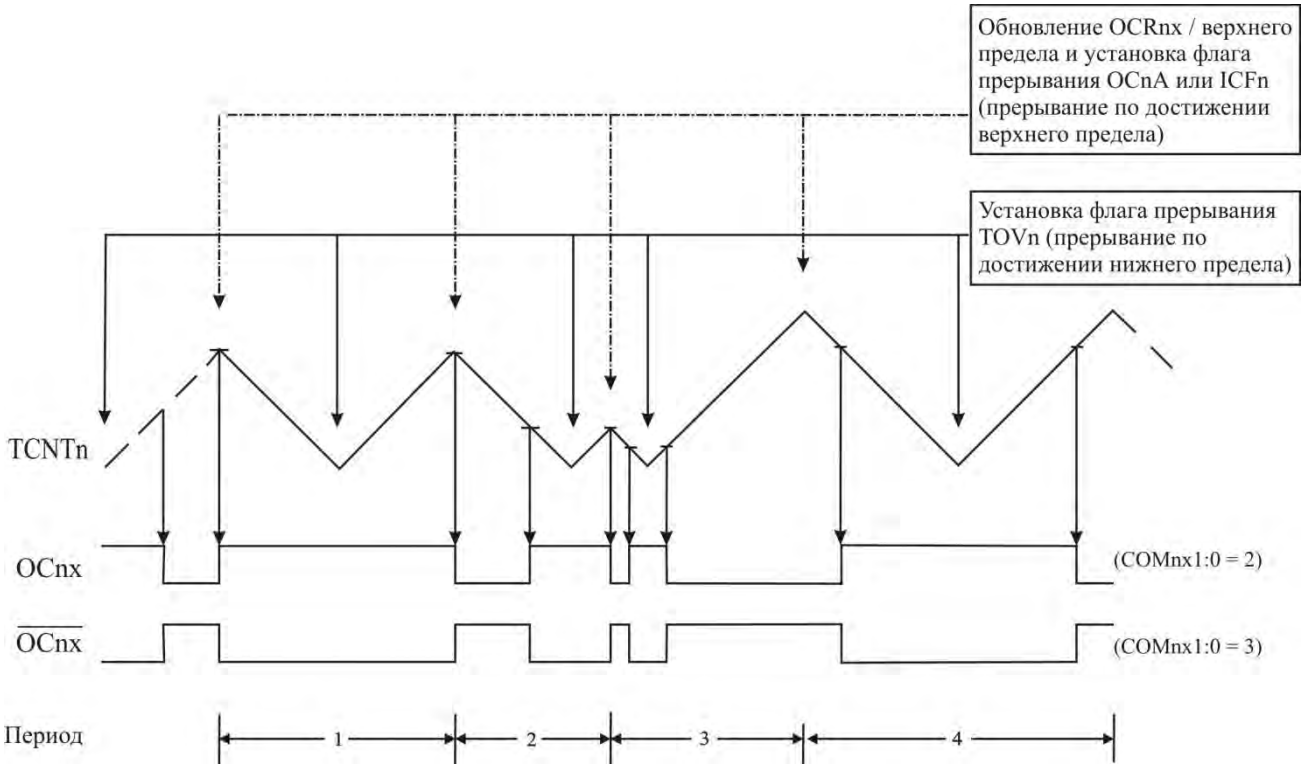


Рисунок 3.54 – Временная диаграмма для режима ШИМ с фазовой коррекцией

Флаг переполнения таймера/счетчика (TOVn) устанавливается всякий раз, когда счетчик достигает нижнего предела. Если для задания верхнего предела используется регистр OCRnA или ICRn, то устанавливается либо флаг OCnA, либо ICFn, соответственно, на том же такте, когда происходит обновление регистра OCRnx значением из буферного регистра (на вершине счета). Флаги прерывания могут использоваться для генерации прерывания всякий раз, когда счетчик достигает значения верхнего или нижнего предела.

При изменении значения верхнего предела счета программа должна гарантировать, что записываемое новое значение верхнего предела больше или равно значению каждого из регистров сравнения. Если значение верхнего предела будет меньше значения какого-либо регистра сравнения, то совпадение между TCNTn и OCRnx никогда не возникнет. Обратите внимание, что при использовании фиксированных значений верхнего предела неиспользуемые биты маскируются нулем, когда производится запись в любой из регистров OCRnx. Третий период на рисунке 3.54 иллюстрирует случай, когда изменение верхнего предела счета во время работы таймера/счетчика может привести к генерации несимметричного импульса. Причина этого кроется в моменте обновления регистра OCRnx. Поскольку обновление OCRnx происходит на вершине счета, то и период ШИМ начинается и заканчивается на вершине счета. Это подразумевает, что длительность обратного счета определяется предыдущим значением верхнего предела, в то время как длительность прямого счета определяется новым значением верхнего предела. Если предыдущее и новое значение верхнего предела отличаются друг от друга, то и длительности обратного и прямого счета будут также отличаться. Разница в длительности счета приводит к несимметричности выходных импульсов.

При изменении значения верхнего предела счета во время работы таймера/счетчика рекомендуется использовать режим ШИМ с фазовой и частотной коррекцией вместо режима только с фазовой коррекцией. Если же используется статическое значение верхнего предела, то между данными режимами практически нет отличий.

В режиме ШИМ ФК блоки сравнения позволяют генерировать ШИМ-сигналы на выводах OCnx. Если COMnx1–0 = 2, то задается ШИМ без инверсии выхода, а если COMnx1–0 = 3, то задается режим ШИМ с инверсией на выходе (см. таблицу 3.60). Фактическое значение OCnx можно наблюдать на выводе порта, только если для этого вывода задано выходное направление (DDR_OCnx). ШИМ-сигнал генерируется путем установки (или сброса) регистра OCnx при возникновении совпадения между OCRnx и TCNTn во время прямого счета, а также путем сброса (или установки) регистра OCnx при возникновении совпадения между OCRnx и TCNTn во время обратного счета. Частота ШИМ выходного сигнала определяется выражением

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot Верхний\ предел},$$

где переменная N задает коэффициент деления предделителя (1, 8, 64, 256 или 1024).

Запись предельного значения в регистр OCRnx представляет собой особый случай при генерации ШИМ-импульсов. Если значение OCRnx установить равным нижнему пределу, то на выходе будет постоянно присутствовать низкий логический уровень. А если значение OCRnx установить равным верхнему пределу, то на выходе будет постоянно присутствовать высокий логический уровень. Это относится к режиму ШИМ без инверсии. В режиме ШИМ с инверсией на выходе будут присутствовать противоположные логические уровни.

Если для задания значения верхнего предела используется регистр OCRnA (WGMn3–0 = 11) и COMnA1–0 = 1, то на выходе OCnA будет генерироваться меандр.

Режим широтно-импульсной модуляции с фазовой и частотной коррекцией

Режим широтно-импульсной модуляции с фазовой и частотной коррекцией (ШИМ ФЧК) (WGMn3–0 = 8 или 9) предназначен для генерации ШИМ-импульсов высокой разрешающей способности с фазовой и частотной коррекцией. Режим ШИМ ФЧК, как и режим ШИМ с фазовой коррекцией, основан на двунаправленной работе таймера/счетчика. Счетчик циклически выполняет счет от нижнего предела (0x0000) до верхнего предела, а затем обратно от верхнего предела к нижнему пределу. В неинвертирующем режиме выход OCnx сбрасывается при возникновении совпадения между TCNTn и OCRnx во время прямого счета и устанавливается при возникновении совпадения во время обратного счета. В инвертирующем режиме, наоборот, во время прямого счета происходит установка, а во время обратного – сброс выхода OCnx. При двунаправленном счете максимальная частота ШИМ-сигнала меньше, чем при однонаправленном счете. Однако, благодаря симметричности режимов ШИМ с двунаправленной работой, данные режимы предпочтительны при решении задач управления приводами.

Основное отличие между режимами ШИМ ФК и ШИМ ФЧК состоит в моменте обновления регистра OCRnx значением из буферного регистра OCRnx (см. рисунки 3.54 и 3.55).

Разрешающая способность ШИМ может задаваться либо регистром ICRn, либо OCRnA. Минимальная разрешающая способность равняется двум разрядам (установка значения 0x0003 в регистре ICRn или OCRnA), а максимальная – 16 разрядам (установка значения 0xFFFF в регистре ICRn или OCRnA). Разрешающую способность ШИМ в битах можно вычислить по формуле

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

В режиме ШИМ ФЧК счетчик инкрементируется, пока его значение не совпадет либо со значением в регистре ICRn (WGMn3–0 = 8), либо со значением в регистре OCRnA (WGMn3–0 = 9). Затем счетчик достигает верхнего предела и изменяет направление счета. Значение TCNTn остается равным верхнему пределу в течение одного такта синхронизации таймера. Временная диаграмма для режима ШИМ ФЧК представлена на рисунке 3.55. Рисунок отображает режим ШИМ ФЧК, когда для задания верхнего предела используется регистр OCRnA или ICRn. Значение TCNTn на временной диаграмме показано в виде гистограммы для иллюстрации двунаправленности счета. Диаграмма включает неинвертированный и инвертированный ШИМ-выходы. Короткими горизонтальными линиями обозначены положения на графике TCNTn, где значения регистров OCRnx и TCNTn совпадают. Флаг прерывания OCnx устанавливается при возникновении совпадения.

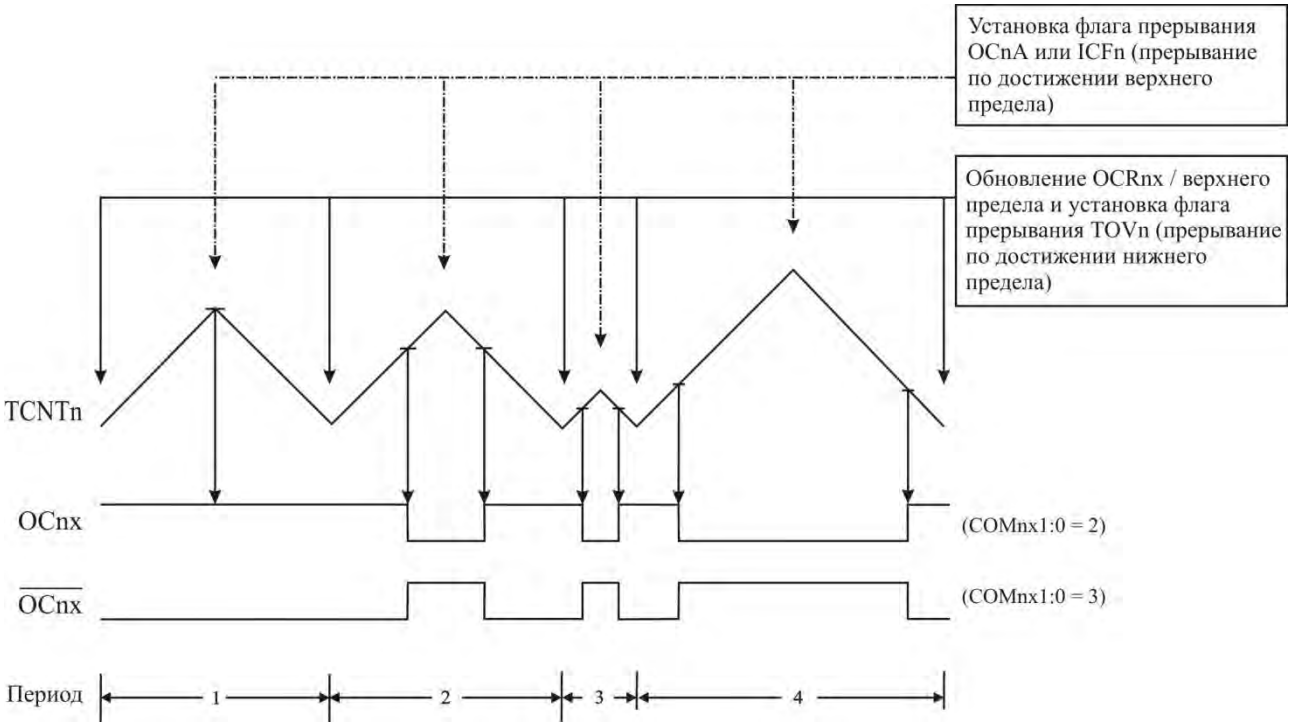


Рисунок 3.55 – Временная диаграмма для режима широтно-импульсной модуляции с фазовой и частотной коррекцией

Флаг переполнения таймера/счетчика (TOVn) устанавливается на том же такте, когда регистр OCRnx обновляется значением из буферного регистра (на нижнем пределе счета). Если для задания верхнего предела используется регистр OCRnA или ICRn, то устанавливается либо флаг OCnA, либо ICFn, соответственно, когда значение регистра TCNTn достигает верхнего предела. Флаги прерывания могут использоваться для генерации прерывания всякий раз, когда счетчик достигает значения верхнего или нижнего предела.

При изменении значения верхнего предела счета программа должна гарантировать, что записываемое новое значение верхнего предела больше или равно значению каждого из регистров сравнения. Если значение верхнего предела будет меньше значения какого-либо регистра сравнения, то совпадение между TCNTn и OCRnx никогда не возникнет.

На рисунке 3.55 видно, что в режиме ШИМ с фазовой и частотной коррекцией, в отличие от режима ШИМ ФК, генерируемый выходной сигнал симметричен на всех периодах. Поскольку регистры OCRnx обновляются на нижнем пределе, длительности прямого и обратного счета всегда будут равны. Это дает симметричные импульсы на выходе, а, следовательно, и корректную частоту.

Использование регистра ICRn для задания верхнего предела целесообразно, когда верхний предел счета является константой. В этом случае также освобождается регистр OCRnA для генерации ШИМ-сигнала на выходе OCnA. Однако если базовая частота широтно-импульсной модуляции активно изменяется (за счет изменения значения верхнего предела), то в данной ситуации определенно лучше использовать регистр OCRnA для задания верхнего предела, поскольку он поддерживает двойную буферизацию.

В режиме ШИМ ФЧК блоки сравнения позволяют генерировать ШИМ-сигналы на выводах OCnx. Если COMnx1–0 = 2, то задается ШИМ без инверсии на выходе, а если COMnx1–0 = 3, то задается режим ШИМ с инверсией на выходе (см. таблицу 3.60). Фактическое значение OCnx можно наблюдать на выводе порта, только если для этого вывода задано выходное направление (DDR_OCnx). ШИМ-сигнал генерируется путем установки (или сброса) регистра OCnx при возникновении совпадения между OCRnx и TCNTn во время прямого счета, а также путем сброса (или установки) регистра OCnx при возникнове-

нии совпадения между OCRnx и TCNTn во время обратного счета. Частота ШИМ выходного сигнала определяется выражением

$$f_{OCnxPFCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot Верхний\ предел},$$

где переменная N задает коэффициент деления предделителя (1, 8, 64, 256 или 1024).

Запись предельного значения в регистр OCRnx представляет собой особый случай при генерации ШИМ-импульсов. Если значение OCRnx установить равным нижнему пределу, то на выходе будет постоянно присутствовать низкий логический уровень. А если значение OCRnx установить равным верхнему пределу, то на выходе будет постоянно присутствовать высокий логический уровень. Это относится к режиму ШИМ без инверсии. В режиме ШИМ с инверсией на выходе будут присутствовать противоположные логические уровни.

Если для задания значения верхнего предела используется регистр OCRnA (WGMn3–0 = 9) и COMnA1–0 = 1, то на выходе OCnA будет генерироваться меандр.

3.10.9 Временные диаграммы 16-разрядного таймера/счетчика

На рисунках представлена информация о том, когда происходит установка флагов прерывания и обновление регистра OCRnx значением из буферного регистра OCRnx (только для режимов с двойной буферизацией). На рисунке 3.56 представлена временная диаграмма с установкой флага OCFnx. На рисунке 3.57 показаны те же временные характеристики, но при включенном предделителе.

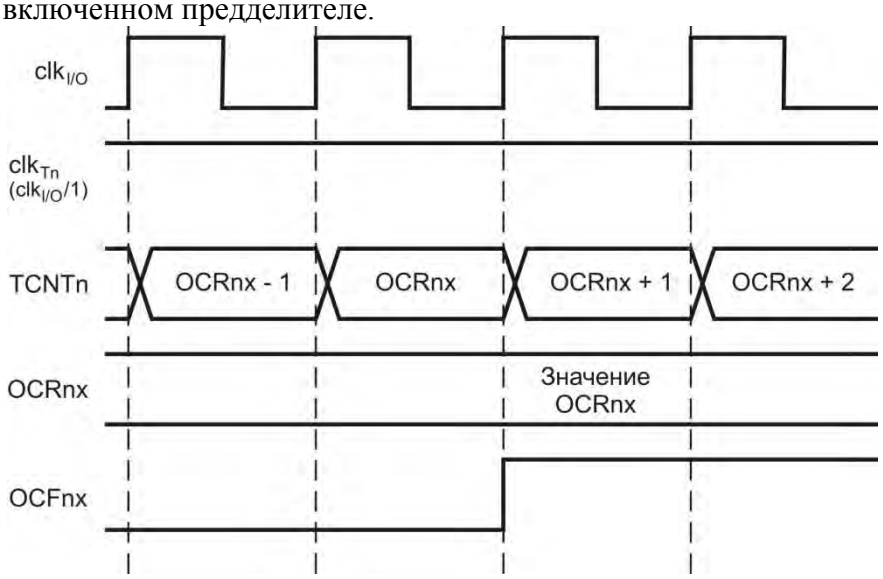


Рисунок 3.56 – Временная диаграмма таймера/счетчика с установкой флага OCFnx, без предделения

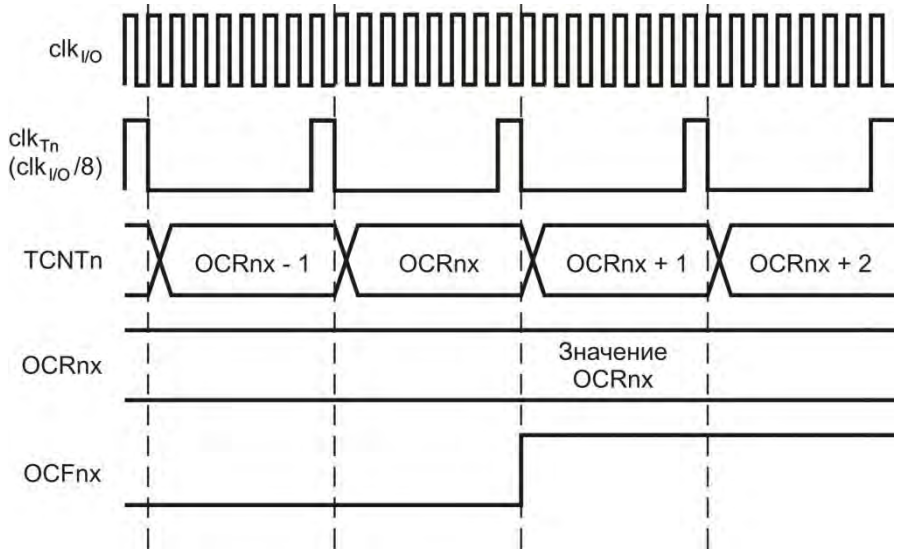


Рисунок 3.57 – Временная диаграмма таймера/счетчика с установкой флага OCFnx и предделением ($f_{clk\_I/O}/8$)

На рисунке 3.58 иллюстрируется счетная последовательность в районе верхнего предела в различных режимах. Если используется режим ШИМ ФЧК, то регистр OCRnx обновляется на нижнем пределе. В этом случае временная диаграмма будет такой же, но обозначение «Верхний предел» необходимо заменить на «Нижний предел», «Верхний предел - 1» на «Нижний предел + 1» и т.д. Аналогичные переименования необходимо применить в режимах, где флаг TOVn устанавливается на нижнем пределе. На рисунке 3.59 показаны те же временные характеристики, но при включенном предделителе.

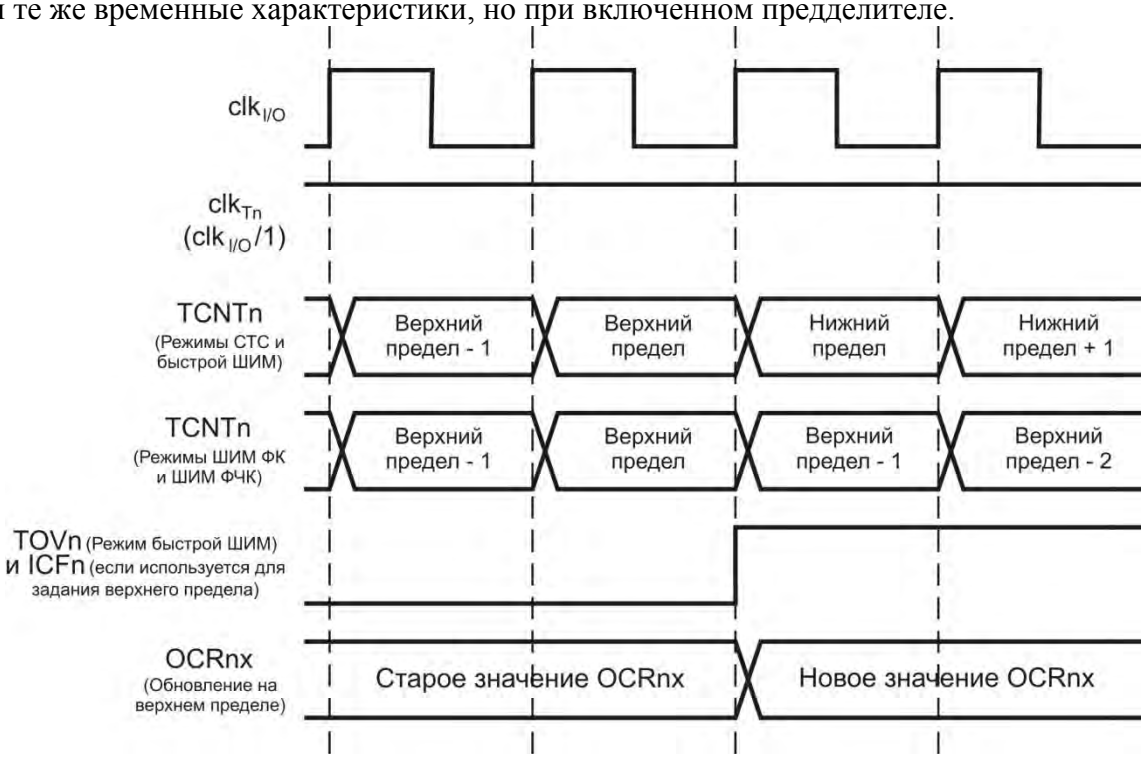


Рисунок 3.58 – Временная диаграмма таймера/счетчика без предделения

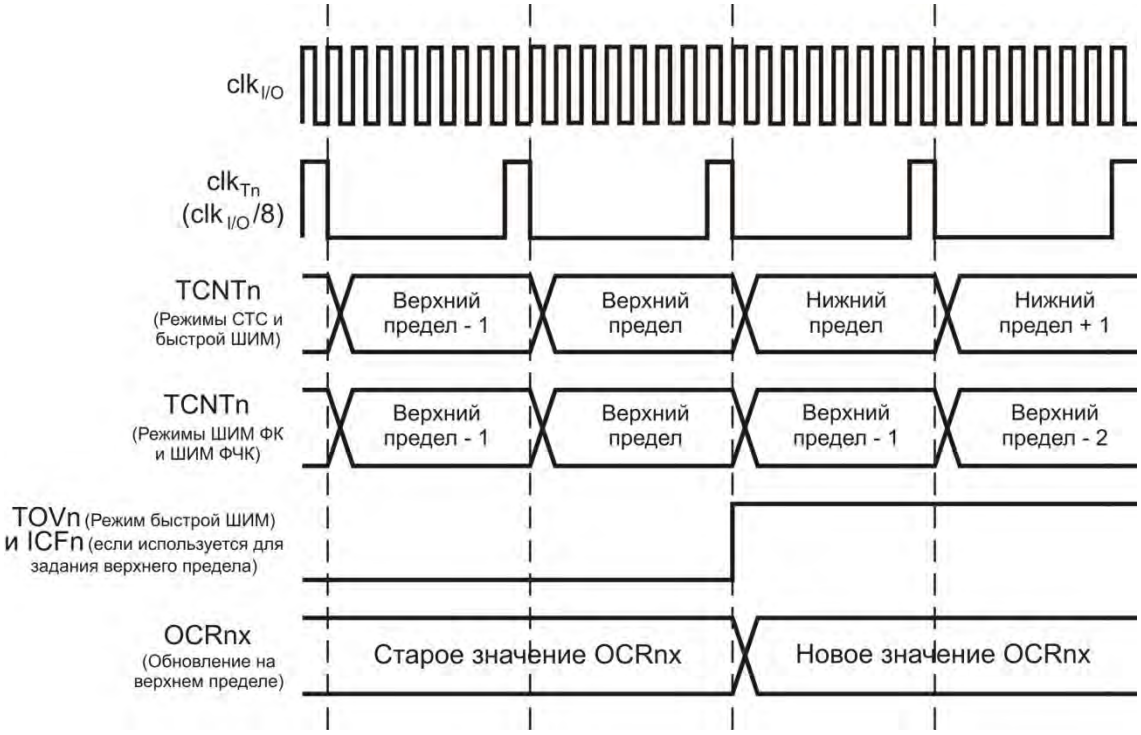


Рисунок 3.59 - Временная диаграмма таймера/счетчика с предделением ($f_{clk\_I/O}/8$)

3.10.10 Описание регистров 16-разрядного таймера/счетчика

Регистр A управления таймером/счетчиком 1 – TCCR1A

Бит	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Регистр A управления таймером/счетчиком 3 – TCCR3A

Бит	7	6	5	4	3	2	1	0
	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Разряды 7, 6 – COMnA1, 0: Задание режима формирования выходного сигнала для канала A

Разряды 5, 4 – COMnB1, 0: Задание режима формирования выходного сигнала для канала B

Разряды 3, 2 – COMnC1, 0: Задание режима формирования выходного сигнала для канала C

Биты COMnA1–0, COMnB1–0 и COMnC1–0 управляют поведением выводов OCnA, OCnB и OCnC, соответственно. Если один или оба бита COMnA1–0 равны единице, то выход OCnA начинает выполнять альтернативную функцию вывода порта, к которому он подключен. Аналогичные изменения происходят с выводами OCnB и OCnC во время записи логической единицы в один или оба бита COMnB1–0 и COMnC1–0, соответственно. Од-

нако обратите внимание, что бит регистра направления данных (DDR), соответствующий выводу OCnA, OCnB или OCnC, должен быть установлен в единицу для включения выходного драйвера.

Если выбрано подключение сигнала OCnA, OCnB или OCnC к выводу микроконтроллера, то функционирование битов COMnx1–0 зависит от установки битов режима работы таймера/счетчика WGMn3–0. В таблице 3.58 отображается функционирование битов COMnx1–0, когда с помощью битов WGMn3–0 выбран нормальный режим или режим сброса таймера при совпадении (СТС), т.е. режимы без широтно-импульсной модуляции.

Таблица 3.58 – Режимы формирования выходного сигнала при работе таймера без ШИМ

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Описание
0	0	Нормальная работа порта, сигналы OCnA/OCnB/OCnC отключены
0	1	Переключение (инвертирование) OCnA/OCnB/OCnC при совпадении
1	0	Сброс OCnA/OCnB/OCnC при совпадении (установка низкого уровня на выходе)
1	1	Установка OCnA/OCnB/OCnC при совпадении (установка высокого уровня на выходе)

В таблице 3.59 отображается функционирование битов COMnx1:0, когда с помощью битов WGMn3:0 выбран режим быстрой ШИМ.

Таблица 3.59 – Режим формирования выходного сигнала при работе таймера с быстрой ШИМ

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Описание
0	0	Нормальная работа порта, сигналы OCnA/OCnB/OCnC отключены
0	1	WGMn3:0 = 15: Переключение (инвертирование) OCnA при совпадении, сигналы OCnB/OCnC отключены (нормальная работа порта). Для всех остальных установок WGMn эта комбинация соответствует нормальной работе порта, когда сигналы OCnA/OCnB/OCnC отключены
1	0	Сброс OCnA/OCnB/OCnC при совпадении, установка OCnA/OCnB/OCnC на нижнем пределе (неинвертирующий режим)
1	1	Установка OCnA/OCnB/OCnC при совпадении, сброс OCnA/OCnB/OCnC на нижнем пределе (инвертирующий режим)
Примечание – Возникает особая ситуация, когда значение регистра OCRnA/OCRnB/OCRnC равно верхнему пределу счета и установлен бит COMnA1/COMnB1/COMnC1. В этом случае возникшее совпадение игнорируется, но установка или сброс выполняется на нижнем пределе (см. выше подпункт «Режим быстрой широтно-импульсной модуляции»).		

В таблице 3.60 отображается функционирование битов COMnx1–0, когда с помощью битов WGMn3–0 выбран режим ШИМ ФК или ШИМ ФЧК.

Таблица 3.60 – Режим формирования выходного сигнала при работе таймера в режиме ШИМ ФК или ШИМ ФЧК

COMnA1/COMnB1/ COMnC1	COMnA0/COMnB0/ COMnC0	Описание
0	0	Нормальная работа порта, сигналы OCnA/OCnB/OCnC отключены
0	1	WGMn3:0 = 9 или 11: Переключение (инвертирование) OCnA при совпадении, сигналы OCnB/OCnC отключены (нормальная работа порта). Для всех остальных установок WGMn эта комбинация соответствует нормальной работе порта, когда сигналы OCnA/OCnB/OCnC отключены
1	0	Сброс OCnA/OCnB/OCnC при совпадении во время прямого счета, установка OCnA/OCnB/OCnC при совпадении во время обратного счета
1	1	Установка OCnA/OCnB/OCnC при совпадении во время прямого счета, сброс OCnA/OCnB/OCnC при совпадении во время обратного счета
Примечание – Возникает особая ситуация, когда значение регистра OCRnA/OCRnB/OCRnC равно верхнему пределу счета и установлен бит COMnA1/COMnB1/COMnC1. Для получения подробностей см. выше подпункт «Режим широтно-импульсной модуляции с фазовой коррекцией».		

Разряды 1, 0 – WGMn1, 0: Задание режима работы таймера/счетчика

В сочетании с битами WGMn3–2 из регистра TCCRnB данные биты определяют последовательность счета, источник для задания максимального значения счетчика (верхнего предела) и тип генерируемой формы сигнала (см. таблицу 3.61). Таймер/счетчик поддерживает следующие режимы работы: нормальный режим, режим сброса таймера при совпадении и три режима с широтно-импульсной модуляцией (см. 3.10.8 «Режимы работы»).

Таблица 3.61 – Описание битов задания режима работы таймера/счетчика

Режим	WGMn3	WGMn2	WGMn1	WGMn0	Режим работы таймера/ счетчика	Верхний предел	Обновление OCRnx	Установка флага TOVn
1	2	3	4	5	6	7	8	9
0	0	0	0	0	Нормальный	0xFFFF	Немедленно	На максимальном значении
1	0	0	0	1	8-разрядная ШИМ ФК	0x00FF	На верхнем пределе	На нижнем пределе
2	0	0	1	0	9-разрядная ШИМ ФК	0x01FF	На верхнем пределе	На нижнем пределе
3	0	0	1	1	10-разрядная ШИМ ФК	0x03FF	На верхнем пределе	На нижнем пределе
4	0	1	0	0	СТС	OCRnA	Немедленно	На максимальном значении
5	0	1	0	1	8-разрядная быстрая ШИМ	0x00FF	На нижнем пределе	На верхнем пределе
6	0	1	1	0	9-разрядная быстрая ШИМ	0x01FF	На нижнем пределе	На верхнем пределе
7	0	1	1	1	10-разрядная быстрая ШИМ	0x03FF	На нижнем пределе	На верхнем пределе

Продолжение таблицы 3.61

1	2	3	4	5	6	7	8	9
8	1	0	0	0	ШИМ ФЧК	ICRn	На нижнем пределе	На нижнем пределе
9	1	0	0	1	ШИМ ФЧК	OCRnA	На нижнем пределе	На нижнем пределе
10	1	0	1	0	ШИМ ФК	ICRn	На верхнем пределе	На нижнем пределе
11	1	0	1	1	ШИМ ФК	OCRnA	На верхнем пределе	На нижнем пределе
12	1	1	0	0	СТС	ICRn	Немедленно	На максимальном значении
13	1	1	0	1	(Зарезервировано)	-	-	-
14	1	1	1	0	Быстрая ШИМ	ICRn	На нижнем пределе	На верхнем пределе
15	1	1	1	1	Быстрая ШИМ	OCRnA	На нижнем пределе	На верхнем пределе

Регистр B управления таймером/счетчиком 1 – TCCR1B

Бит	7	6	5	4	3	2	1	0
	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10
Чтение/ Запись	Ч/З	Ч/З	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Регистр B управления таймером/счетчиком 3 – TCCR3B

Бит	7	6	5	4	3	2	1	0
	ICNC3	ICES3	-	WGM33	WGM32	CS32	CS31	CS30
Чтение/ Запись	Ч/З	Ч/З	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З
Начальное значение	0	0	0	0	0	0	0	0

Разряд 7 – ICNCn: Подавитель шумов на входе блока захвата

Установка данного бита (запись логической единицы) активизирует подавитель шумов на входе блока захвата. После активизации подавителя шумов сигнал с вывода ICPn пропускается через цифровой фильтр. Функционирование фильтра требует четыре последовательные равные по значению выборки с вывода ICPn для изменения его выходного состояния. Поэтому когда включен подавитель шумов, сигнал с входа захвата задерживается на четыре такта системной синхронизации.

Разряд 6 – ICESn: Выбор детектируемого фронта на входе блока захвата

Данный бит позволяет задать, какой фронт на выводе ICPn приведет к захвату состояния таймера. Если ICESn = 0, то спадающий (отрицательный) фронт приводит к захвату состояния таймера, а если ICESn = 1, то нарастающий (положительный) фронт инициирует захват.

Когда захват инициируется в соответствии с установкой бита ICESn, содержимое счетчика копируется в регистр захвата ICRn. При этом также устанавливается флаг захвата ICFn, который может использоваться для генерации прерывания по захвату (если данное прерывание разрешено).

Когда регистр ICRn используется для хранения значения верхнего предела счета (см. таблицу 3.61), вход захвата ICPn отключается от соответствующего вывода микроконтроллера и, следовательно, функция захвата блокируется.

Разряд 5 – Зарезервированный бит

Данный разряд является зарезервированным. Во время записи регистра TCCRnB в этот бит необходимо записывать ноль.

Разряды 4,3 – WGMn3, 2: Задание режима работы таймера/счетчика
См. описание регистра TCCRnA.

Разряды 2–0: CSn2–0: Выбор частоты синхронизации таймера
С помощью трех настроечных битов имеется возможность выбрать различные тактовые частоты, кратные исходной частоте синхронизации (см. рисунки 3.56 и 3.57).

Таблица 3.62 – Описание битов выбора частоты синхронизации таймера/счетчика

CSn2	CSn1	CSn0	Описание
0	0	0	Нет синхронизации (таймер/счетчик остановлен)
0	0	1	$clk_{I/O}$/1 (без предделения)
0	1	0	$clk_{I/O}$/8 (с предделением)
0	1	1	$clk_{I/O}$/64 (с предделением)
1	0	0	$clk_{I/O}$/256 (с предделением)
1	0	1	$clk_{I/O}$/1024 (с предделением)
1	1	0	Внешний тактовый источник на выводе Tn. Синхронизация по спадающему фронту
1	1	1	Внешний тактовый источник на выводе Tn. Синхронизация по нарастающему фронту

Если для тактирования таймера/счетчика выбран внешний вывод Tn, то переключения на выводе Tn будут синхронизировать счетчик, даже если этот вывод сконфигурирован как выход. Данная функция позволяет управлять счетом программно.

Регистр C управления таймером/счетчиком 1 – TCCR1C

Бит	7	6	5	4	3	2	1	0
	FOC1A	FOC1B	FOC1C	-	-	-	-	-
Чтение/ Запись	З	З	З	Ч	Ч	Ч	Ч	Ч
Начальное значение	0	0	0	0	0	0	0	0

Регистр C управления таймером/счетчиком 3 – TCCR3C

Бит	7	6	5	4	3	2	1	0
	FOC3A	FOC3B	FOC3C	-	-	-	-	-
Чтение/ Запись	З	З	З	Ч	Ч	Ч	Ч	Ч
Начальное значение	0	0	0	0	0	0	0	0

Разряд 7 – FOCnA: Бит установки принудительного совпадения для канала A
Разряд 6 – FOCnB: Бит установки принудительного совпадения для канала B
Разряд 5 – FOCnC: Бит установки принудительного совпадения для канала C
Биты FOCnA/FOCnB/FOCnC становятся активными, только если с помощью битов WGMn3–0 задан один из режимов, где нет широтно-импульсной модуляции. Если записать логическую единицу в бит FOCnA/FOCnB/FOCnC, то это приведет к немедленной установке принудительного совпадения на входе блока формирования выходного сигнала. Выход OCnA/OCnB/OCnC изменяется в соответствии с настройкой битов COMnx1–0. Значение, представленное в битах COMnx1–0, определяет эффект принудительного совпадения.

Биты FOCnA/FOCnB/FOCnC не генерируют каких-либо прерываний, а также не вызывают сброс таймера в режиме CTC, где регистр OCRnA задает верхний предел счета.

Биты FOCnA/FOCnB/FOCnC всегда считываются как ноль.

Разряды 4 – 0 зарезервированные биты

Данные разряды являются зарезервированными. Во время записи регистра TCCRnC в эти биты необходимо записывать ноль.

Регистр таймера/счетчика 1 – TCNT1 (старший байт TCNT1H и младший байт TCNT1L)

Бит	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр таймера/счетчика 3 – TCNT3 (старший байт TCNT3H и младший байт TCNT3L)

Бит	7	6	5	4	3	2	1	0	
	TCNT3[15:8]								TCNT3H
	TCNT3[7:0]								TCNT3L
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Две ячейки в области ввода-вывода (TCNTnH и TCNTnL, вместе TCNTn) предоставляют прямой доступ к 16-разрядному счетчику как для чтения, так и для записи. В целях гарантирования одновременного чтения и записи старшего и младшего байтов регистра TCNTn доступ организован с использованием 8-разрядного временного регистра TEMP. Временный регистр является общим для всех 16-разрядных регистров таймера (см. также 3.10.2 «Доступ к 16-разрядным регистрам»).

Изменение содержимого счетчика TCNTn в течение его работы (счета) связано с риском пропуска совпадения между TCNTn и одним из регистров OCRnx. Запись в регистр TCNTn блокирует возникновение совпадения на последующем такте синхронизации для всех блоков сравнения.

Регистр A сравнения таймера/счетчика 1 – OCR1A (старший байт OCR1AH и младший байт OCR1AL)

Бит	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр B сравнения таймера/счетчика 1 – OCR1B (старший байт OCR1BH и младший байт OCR1BL)

Бит	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	OCR1B[7:0]								OCR1BL
Чтение/Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр C сравнения таймера/счетчика 1 – OCR1C (старший байт OCR1CH и младший байт OCR1CL)

Бит	7	6	5	4	3	2	1	0	
	OCR1C[15:8]								OCR1CH
	OCR1C[7:0]								OCR1CL
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр A сравнения таймера/счетчика 3 – OCR3A (старший байт OCR3AH и младший байт OCR3AL)

Бит	7	6	5	4	3	2	1	0	
	OCR3A[15:8]								OCR3AH
	OCR3A[7:0]								OCR3AL
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр B сравнения таймера/счетчика 3 – OCR3B (старший байт OCR3BH и младший байт OCR3BL)

Бит	7	6	5	4	3	2	1	0	
	OCR3B[15:8]								OCR3BH
	OCR3B[7:0]								OCR3BL
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр C сравнения таймера/счетчика 3 – OCR3C (старший байт OCR3CH и младший байт OCR3CL)

Бит	7	6	5	4	3	2	1	0	
	OCR3C[15:8]								OCR3CH
	OCR3C[7:0]								OCR3CL
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

В регистрах сравнения хранится 16-разрядное значение, которое непрерывно сравнивается со значением счетчика (TCNTn). Возникающее совпадение может использоваться для генерации прерывания по результату сравнения или для формирования меандра на выводе OCnx.

Регистры сравнения являются 16-разрядными. В целях гарантирования одновременного чтения и записи старшего и младшего байтов регистра OCRnx доступ организован с использованием 8-разрядного временного регистра TEMP. Временный регистр является общим для всех 16-разрядных регистров таймера (см. также 3.10.2 «Доступ к 16-разрядным регистрам»).

Регистр захвата таймера/счетчика 1 – ICR1 (старший байт ICR1H и младший байт ICR1L)

Бит	7	6	5	4	3	2	1	0	
	ICR1[15:8]								ICR1H
	ICR1[7:0]								ICR1L
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр захвата таймера/счетчика 3 – ICR3 (старший байт ICR3H и младший байт ICR3L)

Бит	7	6	5	4	3	2	1	0	
	ICR3[15:8]								ICR3H
	ICR3[7:0]								ICR3L
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр захвата обновляется содержимым счетчика (TCNTn) при каждом выполнении условия захвата на входе ICPn (или дополнительно на выходе аналогового компаратора для таймера/счетчика 1). Регистры захвата могут использоваться для задания верхнего предела счета.

Регистры захвата являются 16-разрядными. В целях гарантирования одновременности чтения и записи старшего и младшего байтов регистра ICRn доступ организован с использованием 8-разрядного временного регистра TEMP. Временный регистр является общим для всех 16-разрядных регистров таймера (см. также 3.10.2 «Доступ к 16-разрядным регистрам»).

Регистр масок прерываний таймеров/счетчиков – TIMSK

Бит	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Примечание – Этот регистр содержит биты управления прерываниями для нескольких таймеров/счетчиков, но в данном подпункте описаны только биты таймера/счетчика 1. Характеристики остальных битов смотрите в описаниях соответствующих таймеров.

Разряд 5 – TICIE1: Разрешение прерывания по захвату состояния таймера/счетчика 1

Когда TICIE1 = 1 и установлен флаг I в регистре статуса SREG, то разрешается прерывание по захвату состояния таймера/счетчика 1. Когда устанавливается флаг ICF1 в регистре TIFR, программа переходит на соответствующий вектор прерывания (см. подраздел 3.6 «Прерывания»).

Разряд 4 – OCIE1A: Разрешение прерывания по совпадению для канала А таймера/счетчика 1

Когда OCIE1A = 1 и установлен флаг I в регистре статуса SREG, то разрешается прерывание по результату совпадения для канала A таймера/счетчика 1. Когда устанавливается флаг OCF1A в регистре TIFR, программа переходит на соответствующий вектор прерывания (см. подраздел 3.6 «Прерывания»).

Разряд 3 – OCIE1B: Разрешение прерывания по совпадению для канала B таймера/счетчика 1

Когда OCIE1B = 1 и установлен флаг I в регистре статуса SREG, то разрешается прерывание по результату совпадения для канала B таймера/счетчика 1. Когда устанавливается флаг OCF1B в регистре TIFR, программа переходит на соответствующий вектор прерывания (см. подраздел 3.6 «Прерывания»).

Разряд 2 – TOIE1: Разрешение прерывания по переполнению таймера/счетчика 1

Когда TOIE1 = 1 и установлен флаг I в регистре статуса SREG, то разрешается прерывание по переполнению таймера/счетчика 1. Когда устанавливается флаг TOV1 в регистре TIFR, программа переходит на соответствующий вектор прерывания (см. подраздел 3.6 «Прерывания»).

Расширенный регистр масок прерываний таймеров/счетчиков – ETIMSK

Бит	7	6	5	4	3	2	1	0	
	-	-	TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	OCIE1C	ETIMSK
Чтение/ Запись	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7,6 зарезервированные биты

Данные разряды являются зарезервированными. Во время записи регистра ETIMSK в эти биты необходимо записывать ноль.

Разряд 5 – TICIE3: Разрешение прерывания по захвату состояния таймера/счетчика 3

Когда TICIE3 = 1 и установлен флаг I в регистре статуса SREG, то разрешается прерывание по захвату состояния таймера/счетчика 3. Когда устанавливается флаг ICF3 в регистре ETIFR, программа переходит на соответствующий вектор прерывания (см. подраздел 3.6 «Прерывания»).

Разряд 4 – OCIE3A: Разрешение прерывания по совпадению для канала A таймера/счетчика 3

Когда OCIE3A = 1 и установлен флаг I в регистре статуса SREG, то разрешается прерывание по результату совпадения для канала A таймера/счетчика 3. Когда устанавливается флаг OCF3A в регистре ETIFR, программа переходит на соответствующий вектор прерывания (см. подраздел 3.6 «Прерывания»).

Разряд 3 – OCIE3B: Разрешение прерывания по совпадению для канала B таймера/счетчика 3

Когда OCIE3B = 1 и установлен флаг I в регистре статуса SREG, то разрешается прерывание по результату совпадения для канала B таймера/счетчика 3. Когда устанавливается флаг OCF3B в регистре ETIFR, программа переходит на соответствующий вектор прерывания (см. подраздел 3.6 «Прерывания»).

Разряд 2 – TOIE3: Разрешение прерывания по переполнению

таймера/счетчика 3

Когда TOIE3 = 1 и установлен флаг I в регистре статуса SREG, то разрешается прерывание по переполнению таймера/счетчика 3. Когда устанавливается флаг TOV3 в регистре ETIFR, программа переходит на соответствующий вектор прерывания (см. подраздел 3.6 «Прерывания»).

Разряд 1 – OCIE3C: Разрешение прерывания по совпадению для канала C таймера/счетчика 3

Когда OCIE3C = 1 и установлен флаг I в регистре статуса SREG, то разрешается прерывание по результату совпадения для канала C таймера/счетчика 3. Когда устанавливается флаг OCF3C в регистре ETIFR, программа переходит на соответствующий вектор прерывания (см. подраздел 3.6 «Прерывания»).

Разряд 0 – OCIE1C: Разрешение прерывания по совпадению для канала C таймера/счетчика 1

Когда OCIE1C = 1 и установлен флаг I в регистре статуса SREG, то разрешается прерывание по результату совпадения для канала C таймера/счетчика 1. Когда устанавливается флаг OCF1C в регистре ETIFR, программа переходит на соответствующий вектор прерывания (см. подраздел 3.6 «Прерывания»).

Регистр флагов прерываний таймеров/счетчиков – TIFR

Бит	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Примечание – Этот регистр содержит биты флагов прерываний для нескольких таймеров/счетчиков, но в данном подпункте описаны только биты таймера/счетчика 1. Характеристики остальных битов смотрите в описаниях соответствующих таймеров.

Разряд 5 – ICF1: Флаг захвата состояния таймера/счетчика 1

Флаг устанавливается, когда на входе ICP1 определяется условие захвата. Если с помощью битов WGMn3–0 в качестве источника верхнего предела счета выбирается регистр захвата ICR1, то флаг ICF1 устанавливается, когда счетчик достигает значения верхнего предела.

Флаг ICF1 автоматически сбрасывается при переходе на вектор прерывания по захвату состояния таймера/счетчика. Альтернативно, флаг ICF1 можно сбросить путем записи логической единицы в соответствующий разряд регистра.

Разряд 4 – OCF1A: Флаг совпадения для канала A таймера/счетчика 1

Данный флаг устанавливается на такте, следующем после совпадения значений счетчика (TCNT1) и регистра A значения сравнения (OCR1A).

Обратите внимание, что сигнал установки принудительного совпадения (FOC1A) не устанавливает флаг OCF1A.

Флаг OCF1A автоматически сбрасывается при переходе на вектор прерывания по результату сравнения для канала A. Альтернативно, флаг OCF1A можно сбросить путем записи логической единицы в соответствующий разряд регистра.

Разряд 3 – OCF1B: Флаг совпадения для канала B таймера/счетчика 1

Данный флаг устанавливается на такте, следующем после совпадения значений счетчика (TCNT1) и регистра B значения сравнения (OCR1B).

Обратите внимание, что сигнал установки принудительного совпадения (FOC1B) не устанавливает флаг OCF1B.

Флаг OCF1B автоматически сбрасывается при переходе на вектор прерывания по результату совпадения для канала B. Альтернативно, флаг OCF1B можно сбросить путем записи логической единицы в соответствующий разряд регистра.

Разряд 2 – TOV1: Флаг переполнения таймера/счетчика 1

Установка данного флага зависит от настройки битов WGMn3–0. В нормальном режиме и режиме CTC флаг TOV1 устанавливается при переполнении таймера/счетчика. См. таблицу 3.61 для изучения поведения флага TOV1 при задании других значений WGMn3:0.

Флаг TOV1 автоматически сбрасывается при переходе на вектор прерывания по переполнению таймера/счетчика 1. Альтернативно, флаг TOV1 можно сбросить путем записи логической единицы в соответствующий разряд регистра.

Расширенный регистр флагов прерываний таймеров/счетчиков – ETIFR

Бит	7	6	5	4	3	2	1	0	
	-	-	ICF3	OCF3A	OCF3B	TOV3	OCF3C	OCF1C	ETIFR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7, 6 зарезервированные биты

Данные разряды являются зарезервированными. Во время записи в регистр ETIFR в эти биты необходимо записывать ноль.

Разряд 5 – ICF3: Флаг захвата состояния таймера/счетчика 3

Флаг устанавливается, когда на входе ICP3 определяется условие захвата. Если с помощью битов WGMn3–0 в качестве источника верхнего предела счета выбирается регистр захвата ICR3, то флаг ICF3 устанавливается, когда счетчик достигает значения верхнего предела.

Флаг ICF3 автоматически сбрасывается при переходе на вектор прерывания по захвату состояния таймера/счетчика. Альтернативно, флаг ICF3 можно сбросить путем записи логической единицы в соответствующий разряд регистра.

Разряд 4 – OCF3A: Флаг совпадения для канала A таймера/счетчика 3

Данный флаг устанавливается на такте, следующем после совпадения значений счетчика (TCNT3) и регистра A значения сравнения (OCR3A).

Обратите внимание, что сигнал установки принудительного совпадения (FOC3A) не устанавливает флаг OCF3A.

Флаг OCF3A автоматически сбрасывается при переходе на вектор прерывания по результату совпадения для канала A. Альтернативно, флаг OCF3A можно сбросить путем записи логической единицы в соответствующий разряд регистра.

Разряд 3 – OCF3B: Флаг совпадения для канала B таймера/счетчика 3

Данный флаг устанавливается на такте, следующем после совпадения значений счетчика (TCNT3) и регистра B значения сравнения (OCR3B).

Обратите внимание, что сигнал принудительной установки совпадения (FOC3B) не устанавливает флаг OCF3B.

Флаг OCF3B автоматически сбрасывается при переходе на вектор прерывания по результату сравнения для канала B. Альтернативно, флаг OCF3B можно сбросить путем записи логической единицы в соответствующий разряд регистра.

Разряд 2 – TOV3: Флаг переполнения таймера/счетчика 3

Установка данного флага зависит от настройки битов WGMn3:0. В нормальном режиме и режиме CTC флаг TOV3 устанавливается при переполнении таймера/счетчика. См. таблицу 3.61 для изучения поведения флага TOV3 при задании других значений WGMn3:0.

Флаг TOV3 автоматически сбрасывается при переходе на вектор прерывания по переполнению таймера/счетчика 3. Альтернативно, флаг TOV3 можно сбросить путем записи логической единицы в соответствующий разряд регистра.

Разряд 1 – OCF3C: Флаг совпадения для канала C таймера/счетчика 3

Данный флаг устанавливается на такте, следующем после совпадения значений счетчика (TCNT3) и регистра C значения сравнения (OCR3C).

Обратите внимание, что сигнал принудительной установки совпадения (FOC3C) не устанавливает флаг OCF3C.

Флаг OCF3C автоматически сбрасывается при переходе на вектор прерывания по результату совпадения для канала C. Альтернативно, флаг OCF3C можно сбросить путем записи логической единицы в соответствующий разряд регистра.

Разряд 0 – OCF1C: Флаг совпадения для канала C таймера/счетчика 1

Данный флаг устанавливается на такте, следующем после совпадения значений счетчика (TCNT1) и регистра C значения сравнения (OCR1C).

Обратите внимание, что сигнал принудительной установки совпадения (FOC1C) не устанавливает флаг OCF1C.

Флаг OCF1C автоматически сбрасывается при переходе на вектор прерывания по результату совпадения для канала C. Альтернативно, флаг OCF1C можно сбросить путем записи логической единицы в соответствующий разряд регистра.

3.11 Предделители таймера/счетчика 1, таймера/счетчика 2 и таймера/счетчика 3

Таймеры/счетчики 1, 2 и 3 используют один и тот же модуль предделителя, но могут иметь различные установки предварительного деления. Приведенное ниже описание распространяется на все упомянутые таймеры.

3.11.1 Внутренний тактовый источник

Тактовый вход таймера/счетчика может быть непосредственно связан с системной синхронизацией, если установить CSn2–0 = 1. В данном случае достигается самая быстрая работа таймера/счетчика на максимальной частоте, равной частоте системной синхронизации ($f_{CLK\_I/O}$). Альтернативно, один из четырех сигналов предделителя может использоваться в качестве источника синхронизации. Поделенный тактовый сигнал имеет частоту $f_{CLK\_I/O}/8$, $f_{CLK\_I/O}/64$, $f_{CLK\_I/O}/256$ или $f_{CLK\_I/O}/1024$.

3.11.2 Сброс предделителя

Предделитель работает в автономном режиме, то есть независимо от логики выбора синхронизации таймера/счетчика, и является общим для таймеров 1, 2 и 3. Поскольку выбор синхронизации таймера/счетчика не оказывает влияния на предделитель, состояние предделителя будет иметь последствия для ситуаций, где используется деленный тактовый сигнал. Как пример можно привести неопределенность, которая возникает, когда таймер включен и его работа синхронизируется предделителем (6 > CSn2:0 > 1). Число тактов системной синхронизации с момента включения таймера до возникновения первого счетного импульса может быть от 1 до N+1, где N – коэффициент деления предделителя (8, 64, 256 или 1024).

Можно использовать сброс предделителя для синхронизации работы таймера/счетчика и выполнения программ. Однако следует проявить осторожность, если другой таймер, использующий этот же предделитель, работает с предделением. Сброс предделителя повлияет на период предделителя для всех таймеров/счетчиков, подключенных к нему.

3.11.3 Внешний тактовый источник

Сигнал внешнего тактового источника, подаваемый на вывод Tn, может использоваться как сигнал синхронизации для таймеров/счетчиков (clk_{T1}/clk_{T2}/clk_{T3}). Вывод Tn опрашивается на каждом такте системной синхронизации логикой данного вывода. Считанный таким образом сигнал затем проходит через детектор фронта. На рисунке 3.60 представлена функционально эквивалентная структурная схема логики синхронизации Tn и детектора фронта. Регистры тактируются положительным фронтом внутренней системной синхронизации ($clk_{I/O}$).

Детектор фронта генерирует один тактовый импульс clk_{T1}/clk_{T2}/clk_{T3} на каждом положительном (CSn2–0 = 7) или отрицательном (CSn2–0 = 6) фронте, который он распознает.

Логика синхронизации и детектора фронта вносит задержку на 2,5–3,5 такта системной синхронизации с момента появления фронта сигнала на выводе Tn до обновления счетчика.

Включение и выключение входа синхронизации необходимо выполнять, когда вывод Tn находится в устойчивом состоянии, по крайней мере, в течение одного такта системной синхронизации, в противном случае существует риск генерации ложного тактового импульса таймера/счетчика.

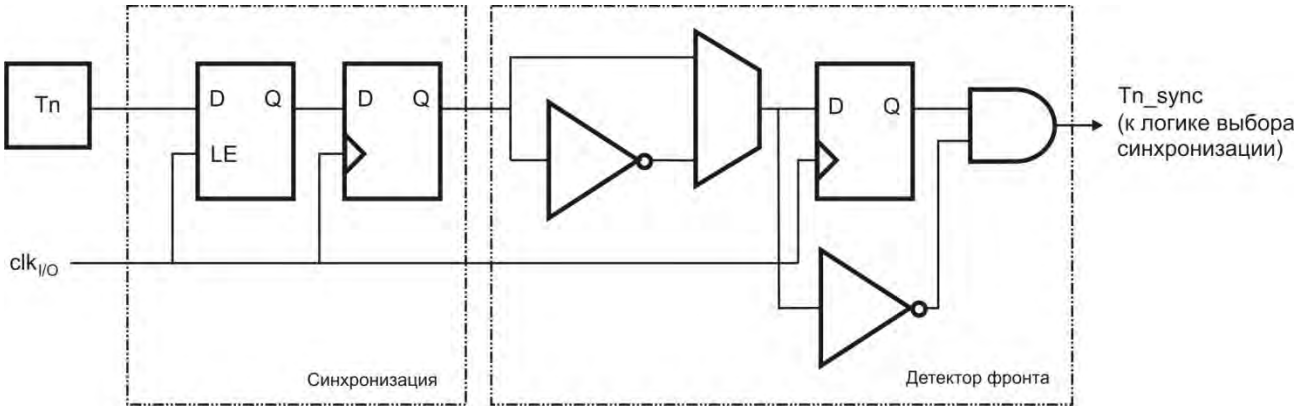


Рисунок 3.60 – Опрос вывода Tn

Для корректной работы логики преобразования каждый полупериод приложенного внешнего тактового сигнала должен быть больше одного периода системной синхронизации. Таким образом, внешний тактовый сигнал должен быть меандром (коэффициент заполнения 50 %) с частотой минимум вдвое меньше частоты системной синхронизации ($f_{ExtClk} < f_{clk\_I/O}/2$). Поскольку детектор фронта использует выборку, то максимальная частота, которую он может определить, равна половине частоты выборки (теорема отсчетов Найквиста). Однако, из-за колебаний частоты системной синхронизации и коэффициента заполнения импульсной последовательности, вызванных допустимыми отклонениями тактового генератора (погрешностями кварцевого резонатора, керамического резонатора и конденсаторов), рекомендуется, чтобы максимальная частота внешнего тактового сигнала была меньше $f_{clk\_I/O}/2{,}5$.

Частота внешнего тактового сигнала не может быть поделена внутренним предделителем (см. рисунок 3.61).

Примечание – Логика синхронизации на входах T3/T2/T1 показана на рисунке 3.60.

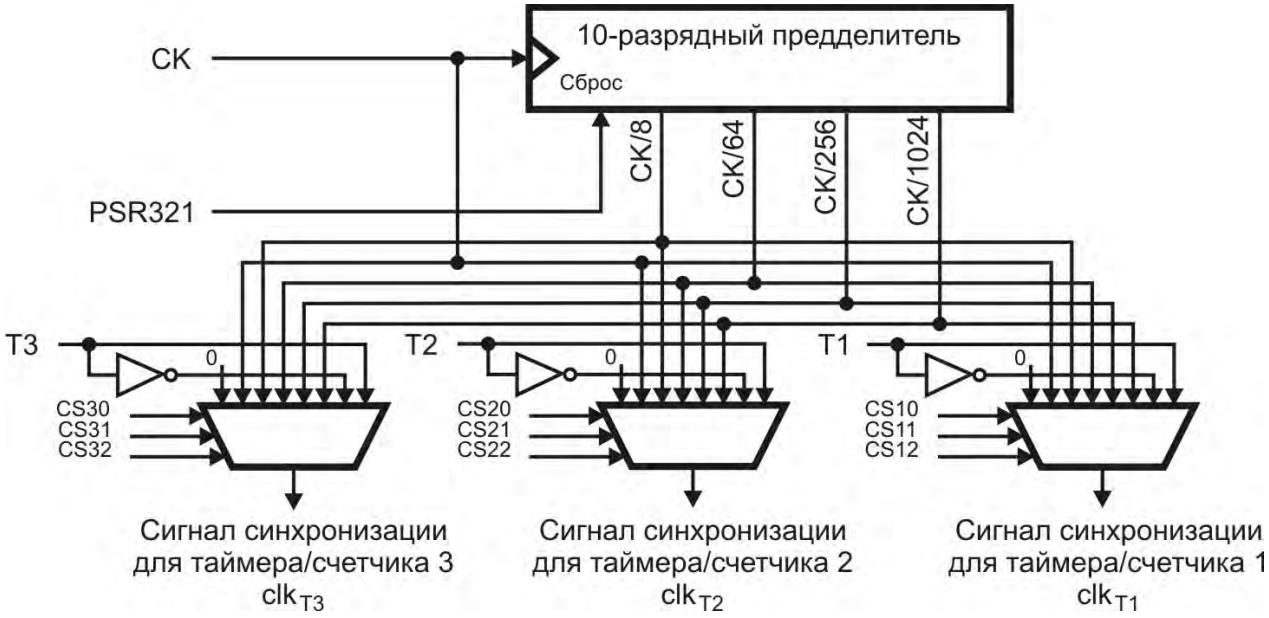


Рисунок 3.61 – Предделитель для таймеров/счетчиков 1, 2 и 3

3.11.4 Регистр специальных функций ввода-вывода – SFIOR

Бит	7	6	5	4	3	2	1	0	
	TSM	-	-	-	ACME	PUD	PSR0	PSR321	SFIOR
Чтение/ Запись	Ч/З	Ч	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – TSM: Режим синхронизации таймеров/счетчиков

Запись логической единицы в данный бит активизирует режим синхронизации таймеров/счетчиков. В этом режиме запоминаются значения, записанные в биты PSR0 и PSR321, и, следовательно, соответствующие сигналы сброса предделителя сохраняются установленными. Этим гарантируется, что соответствующие таймеры будут остановлены и им можно будет присвоить одинаковые значения без риска их модификации в процессе конфигурации. Если в TSM записать логический ноль, то биты PSR0 и PSR321 сбросятся аппаратно, и таймеры/счетчики начнут счет одновременно.

Разряд 0 – PSR321: Сброс предделителя таймеров/счетчиков 1, 2 и 3

Если бит PSR321 равен логической единице, то предделитель таймеров/счетчиков 1, 2 и 3 будет сброшен. Данный бит обычно сразу сбрасывается аппаратно за исключением случая, когда установлен бит TSM. Обратите внимание, что таймеры/счетчики 1, 2 и 3 используют один и тот же предделитель и сброс этого предделителя оказывает влияние на все три таймера.

3.12 8-разрядный таймер/счетчик 2 с широтно-импульсной модуляцией

Таймер/счетчик 2 представляет собой одноканальный 8-разрядный модуль общего назначения. Основные отличительные особенности:

- Одноканальный счетчик.
- Режим сброса таймера при совпадении (автоматическая перезагрузка).
- Широтно-импульсная модуляция без генерации ложных импульсов с фазовой коррекцией.

- Генератор частоты.
- Счетчик внешних событий.
- 10-разрядный предделитель тактовой частоты.
- Генерация прерываний при переполнении и совпадении (TOV2 и OCF2).

3.12.1 Краткий обзор

Упрощенная структурная схема 8-разрядного таймера/счетчика 2 представлена на рисунке 3.62. Связи с регистрами, к которым осуществляет доступ ЦПУ, в том числе линии ввода-вывода, показаны жирной линией. Специфические для данного устройства регистры, расположение и назначение их битов приведены в 3.12.8 «Описание регистров таймера/счетчика 2».

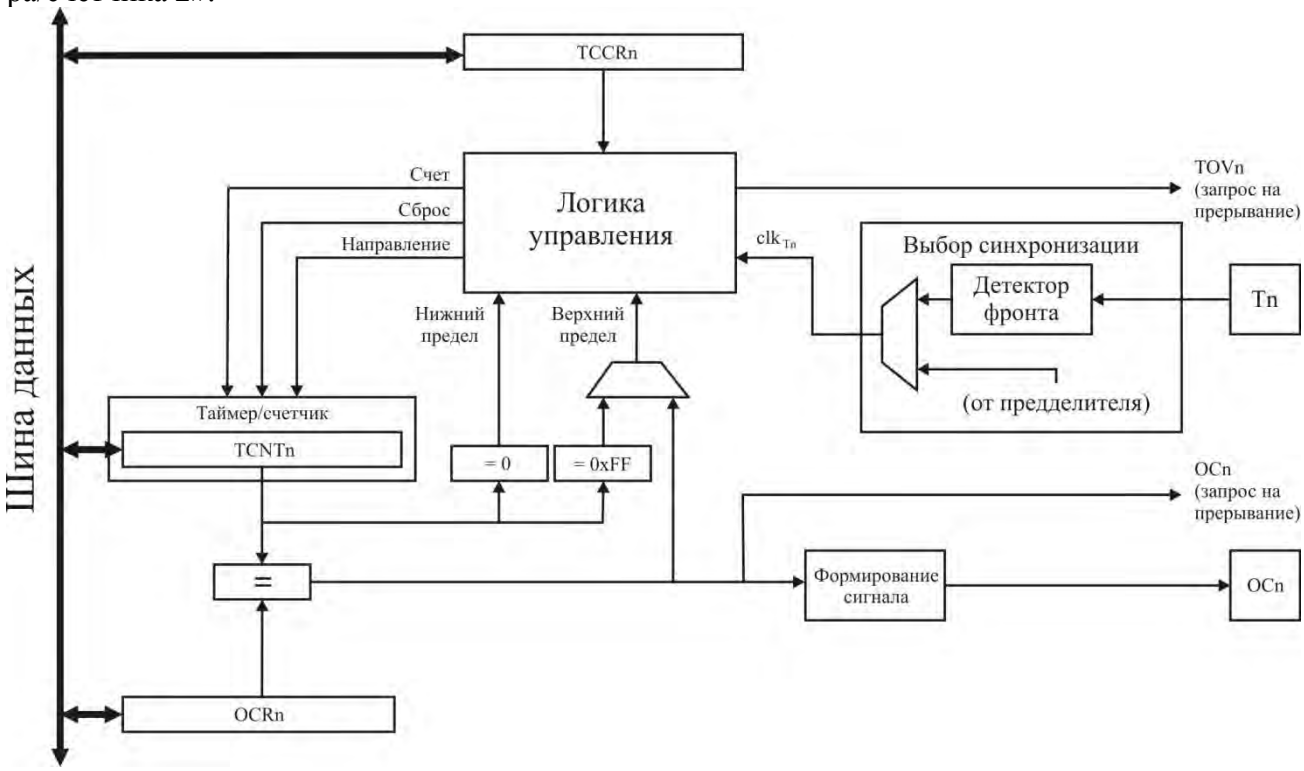


Рисунок 3.62 – Структурная схема 8-разрядного таймера/счетчика 2

Регистры

Счетный регистр таймера/счетчика (TCNT2) и регистр сравнения (OCR2) – это 8-разрядные регистры. Сигналы запроса на прерывание представлены как флаги прерываний таймера в регистре TIFR. Все прерывания индивидуально маскируются с помощью регистра масок прерываний таймеров (TIMSK). Регистры TIFR и TIMSK не представлены на структурной схеме, так как они используются также и другими таймерами микроконтроллера.

Таймер/счетчик может тактироваться от внутреннего источника (через предделитель) или от внешнего источника синхронизации, подключенный к выводу T2. Логический блок выбора синхронизации определяет, какой тактовый источник и фронт используется для инкрементирования (или декрементирования) значения таймера/счетчика. Если источник тактирования не задан, то таймер/счетчик находится в неактивном состоянии. Выход логики выбора тактирования обозначен как синхронизация таймера (clk_{T2}).

Значение регистра сравнения с двойной буферизацией (OCR2) непрерывно сравнивается со значением таймера/счетчика. Результат сравнения может использоваться формирователем сигналов для генерации импульсов с ШИМ или импульсов переменной частоты на выводе OC2. См. 3.12.4 «Блок сравнения» для изучения деталей. Совпадение значения

сравнения со значением таймера/счетчика приводит к установке флага сравнения (OCF2), который может использоваться для генерации запроса на прерывание по результату сравнения.

Определения

Многие обозначения регистров и битов в данном подразделе представлены в общей форме. Латинская строчная буква «n» заменяет номер таймера/счетчика, в данном случае 2. Однако при использовании обозначений регистров или битов в программе необходимо применять точную форму (например, TCNT2 для доступа к значению таймера/счетчика 2).

Определения, приведенные в таблице 3.63, также широко используются на протяжении всего подраздела документа.

Таблица 3.63 – Определения

Нижний предел	Счетчик достигает нижнего предела, когда его значение становится равным нулю (0x00)
Максимум	Счетчик достигает максимума, когда его значение становится равным 0xFF (десятичное число 255)
Верхний предел	Счетчик доходит до верхнего предела, когда он достигает наивысшего значения в последовательности счета. Верхнему пределу может быть присвоено фиксированное значение 0xFF (максимум) или значение, хранящееся в регистре OCR2. Присваивание зависит от режима работы

3.12.2 Источники тактирования таймера/счетчика 2

Таймер/счетчик 2 может тактироваться от внутреннего или внешнего источника синхронизации. Источник тактирования определяется логикой выбора синхронизации, которая контролируется битами выбора частоты синхронизации (CS22–0), расположенными в регистре управления таймером/счетчиком 2 (TCCR2). Для получения подробностей об источниках тактирования и предделителе см. подраздел 3.11 «Предделители таймера/счетчика 1, таймера/счетчика 2 и таймера/счетчика 3».

3.12.3 Блок счетчика

Главным компонентом 8-разрядного таймера/счетчика 2 является программируемый двунаправленный счетчик. На рисунке 3.63 иллюстрируется структурная схема счетчика и окружающих его элементов.

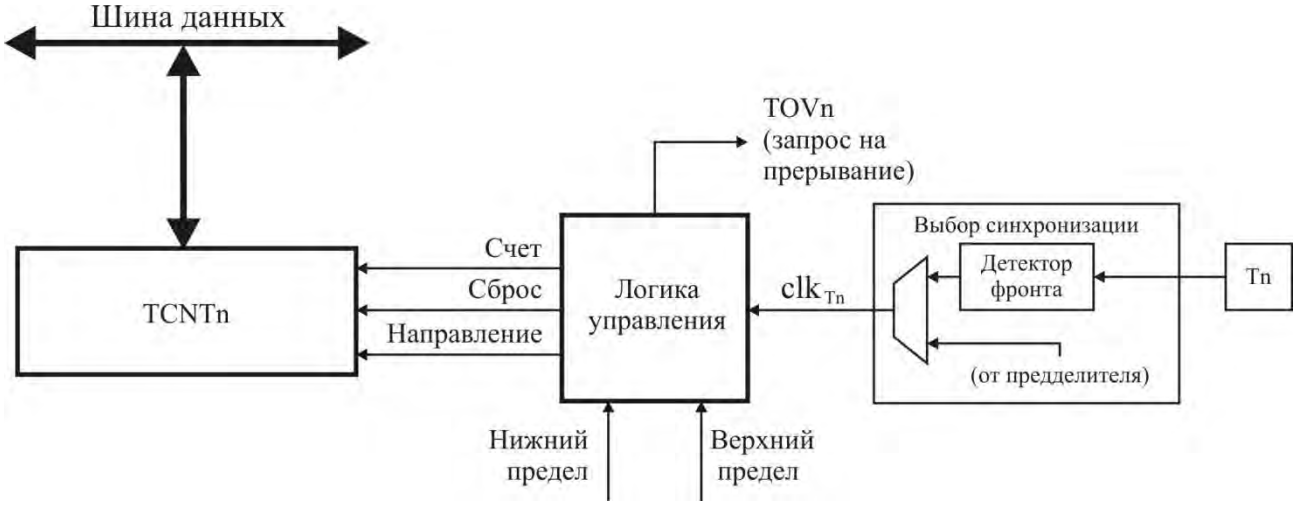


Рисунок 3.63 – Структурная схема блока счетчика

Описание внутренних сигналов

Счет – изменение значения TCNT2 на единицу.

Направление – выбор прямого счета (инкрементирование) или обратного счета (декрементирование).

Сброс – сброс TCNT2 (установка всех разрядов в логический ноль).

clk_{Tn} – синхронизация таймера/счетчика 2, именуемая далее как clk_{T2}.

Верхний предел – показывает, что регистр TCNT2 достиг максимального значения.

Нижний предел – показывает, что регистр TCNT2 достиг минимального значения (нуля).

В зависимости от выбранного режима работы счетчик сбрасывается, инкрементируется или декрементируется на каждом такте синхронизации (clk_{T2}). Тактовый сигнал clk_{T2} может быть сгенерирован внутренним или внешним источником тактирования, который определяется битами выбора частоты синхронизации (CS22–0). Если источник синхронизации не задан (CS22–0 = 0), то таймер останавливается. Однако значение TCNT2 доступно ЦПУ независимо от того, работает синхронизация таймера или нет. Запись в регистр таймера через ЦПУ перекрывает любые действия самого счетчика (сброс или счет), то есть имеет более высокий приоритет.

Последовательность счета определяется установкой битов режима работы таймера WGM21 и WGM20, расположенных в регистре управления таймером/счетчиком (TCCR2). Существует точная связь между поведением счетчика (алгоритмом счета) и генерируемой формой сигнала на выходе OC2. Более подробно об алгоритмах счета и формировании сигналов написано в 3.12.6 «Режим работы».

Флаг переполнения таймера/счетчика (TOV2) устанавливается в соответствии с режимом работы, который выбирается битами WGM21:0. Флаг TOV2 может использоваться для генерации прерывания ЦПУ.

3.12.4 Блок сравнения

8-разрядный цифровой компаратор непрерывно выполняет сравнение содержимого регистра таймера/счетчика TCNT2 с регистром сравнения OCR2. Всякий раз, когда значение TCNT2 совпадает со значением OCR2, компаратор устанавливает флаг совпадения OCF2 на последующем такте синхронизации таймера. Если разрешено прерывание (OCIE2 = 1 и установлен флаг I в регистре SREG), то установка флага совпадения вызывает запрос на прерывание. Флаг OCF2 автоматически сбрасывается, когда выполняется процедура обработки прерывания. Альтернативно, флаг OCF2 можно сбросить программно путем записи логической единицы в соответствующий разряд регистра TIFR. Формирователь сигналов использует результат сравнения для генерации выходных импульсов в соответствии с режимом работы, заданным битами WGM21–0 и битами режима формирования выходного сигнала COM21–0. Сигналы «Верхний предел» и «Нижний предел» используются формирователем сигналов для обработки особенных случаев крайних значений в некоторых режимах работы (см. 3.12.6 «Режимы работы»). На рисунке 3.64 приведена структурная схема блока сравнения.

Регистр OCR2 выполнен по схеме двойной буферизации при использовании режимов с широтно-импульсной модуляцией (ШИМ). В нормальном режиме и режиме сброса таймера при совпадении (CTC) схема двойной буферизации отключается. Двойная буферизация позволяет синхронизировать обновление регистра сравнения OCR2 по достижении либо верхнего, либо нижнего предела счета. Такая синхронизация предотвращает возможность возникновения несимметричных ШИМ-сигналов, гарантируя тем самым отсутствие на выходе паразитных импульсов.

Доступ к регистру OCR2 может показаться сложным, но это не так. Если разрешена двойная буферизация, ЦПУ получает доступ к буферному регистру OCR2, а если двойная буферизация отключена, то ЦПУ обращается непосредственно к регистру OCR2.

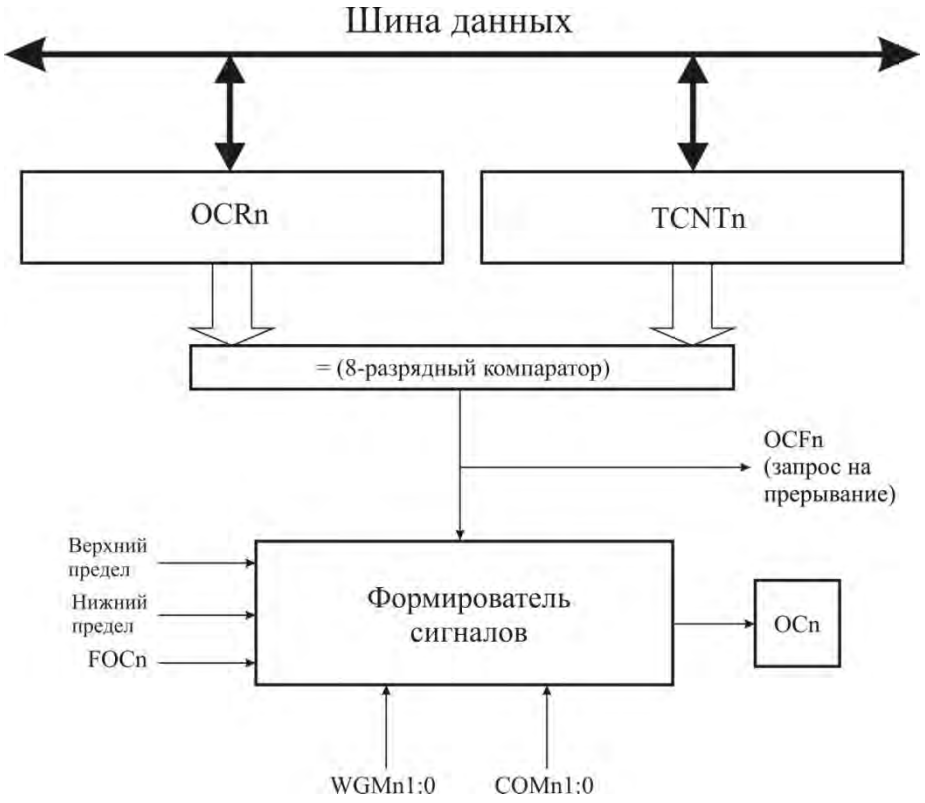


Рисунок 3.64 – Структурная схема блока сравнения

Установка принудительного совпадения

В режимах формирования сигналов без ШИМ результат совпадения компаратора может быть установлен путем записи логической единицы в бит установки принудительного совпадения FOC2. Установка принудительного совпадения компаратора не приводит к установке флага OCF2 или перезагрузке/сбросу таймера, но обновляет состояние вывода OC2 (который будет устанавливаться, сбрасываться или переключаться в зависимости от выбранной установки битов COM21–0), как если бы произошло реальное событие совпадения.

Блокирование сравнения путем записи в TCNT2

Если ЦПУ осуществляет запись в регистр TCNT2, то сравнение будет блокироваться на следующем такте синхронизации таймера, даже если таймер остановлен. Данная функция позволяет устанавливать в регистре OCR2 то же значение, что и в TCNT2, без генерации запроса на прерывание, когда включена синхронизация таймера/счетчика.

Использование блока сравнения

Поскольку запись в TCNT2 блокирует сравнение на один такт синхронизации таймера в любом режиме работы, то когда изменяется TCNT2 при использовании канала сравнения (независимо от того, работает таймер/счетчик или нет), необходимо учесть следующие особенности. Если в регистр TCNT2 записано значение, равное значению OCR2, то игнорирование совпадения приведет к генерации сигнала неправильной формы. Аналогично следует избегать записи в TCNT2 значения, равного нижнему пределу (0x00), когда счетчик работает как вычитающий.

Установка OC2 должна быть выполнена перед настройкой линии на вывод в регистре направления данных. Самый легкий путь установки значения OC2 – использование бита установки принудительного совпадения (FOC2) в нормальном режиме. Регистр OC2 сохраняет свое значение, даже если происходит изменение между режимами формирования сигналов.

Следует учитывать, что биты COM21 и COM20 не содержат схемы двойной буферизации и на любые изменения реагируют мгновенно.

3.12.5 Блок формирования выходного сигнала

Биты задания режима формирования выходного сигнала (COM21–0) имеют две функции. С одной стороны, они используются формирователем сигнала и определяют, какое логическое состояние должно быть на выводе OC2 при возникновении следующего совпадения. С другой стороны, биты COM21 и COM20 управляют источником выходного сигнала для вывода OC2. На рисунке 3.65 представлена упрощенная логическая схема, на которую воздействуют биты COM21–0. На рисунке показаны только те регистры управления портами ввода-вывода (DDR и PORT), на которые оказывают влияние биты COM21 и COM20. При упоминании состояния OC2 речь идет о внутреннем регистре OC2, а не о выводе OC2. При возникновении системного сброса регистр OC2 сбрасывается в ноль.

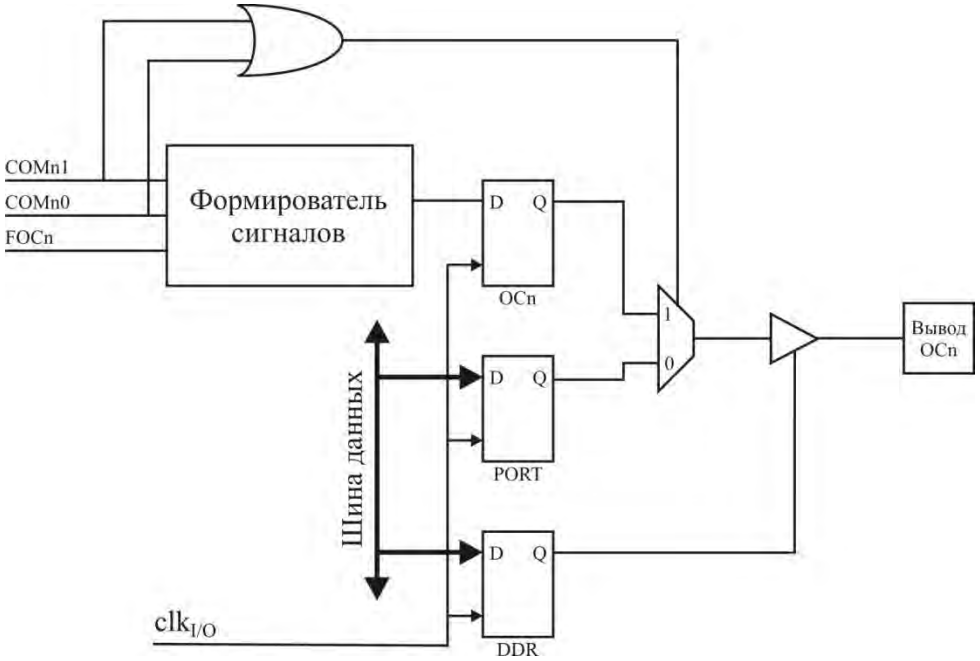


Рисунок 3.65 – Схема блока формирования выходного сигнала

Функция порта ввода-вывода общего назначения переопределяется на функцию выхода формирователя сигнала OC2, если хотя бы один из битов COM21–0 установлен. Однако, направление вывода OC2 (вход или выход) по-прежнему контролируется соответствующим битом регистра направления данных порта. Бит регистра направления данных для вывода OC2 (DDR_OC2) должен быть настроен как выход, прежде чем значение регистра OC2 появится на данном выводе. Управление вводом альтернативной функции не зависит от режима формирования сигналов.

Схемотехника выходной логики позволяет инициализировать состояние регистра OC2 перед разрешением настройки вывода OC2 в качестве выхода. Обратите внимание, что некоторые настройки битов COM21–0 зарезервированы для определенных режимов работы. См. 3.12.8 «Описание регистров таймера/счетчика 2».

Режим сравнения и формирования сигналов

Формирователь сигналов использует биты COM21–0 по-разному в нормальном режиме, режиме сброса таймера при совпадении и режимах с ШИМ. Во всех режимах установка COM21–0 = 0 сообщает формирователю сигналов, что никакое действие в регистре OC2 не должно выполняться при возникновении следующего совпадения. В таблице 3.65

описаны действия при сравнении в режимах без ШИМ. Аналогичная информация для режима с быстрой ШИМ приведена в таблице 3.66, а для режима с ШИМ с фазовой коррекцией – в таблице 3.67.

Изменение состояния битов COM21:0 окажет эффект при возникновении первого совпадения после записи этих битов. В режимах без ШИМ можно принудительно получить незамедлительный эффект, используя бит FOC2.

3.12.6 Режимы работы

Режим работы, то есть поведение таймера/счетчика и линий выходных сигналов, определяется комбинацией битов режима работы таймера (WGM21–0) и битов режима формирования выходного сигнала (COM21–0). Биты COM21–0 не влияют на последовательность счета. Биты режима работы таймера WGM21–0 оказывают воздействие на последовательность счета. Биты COM21 и COM20 определяют, должен ли выход ШИМ быть инвертированным или нет (широтно-импульсная модуляция с инверсией или без). Для режимов без ШИМ биты COM21–0 определяют, какое действие необходимо осуществить при выполнении совпадения: установить, сбросить или переключить выход (см. также 3.12.5 «Блок формирования выходного сигнала»).

Более подробная информация по временным диаграммам работы таймера приведена в 3.12.7 «Временные диаграммы таймера/счетчика 2».

Нормальный режим работы

Самым простым режимом работы является нормальный режим (WGM21–0 = 0). В данном режиме счетчик всегда работает как суммирующий (инкрементирующий), при этом сброс счетчика не выполняется. Счетчик просто переполняется, когда достигает своего максимального 8-разрядного значения (верхний предел равен 0xFF), и затем перезапускается, начиная счет от нижнего предела (0x00). В нормальном режиме работы флаг переполнения таймера/счетчика TOV2 будет установлен на том же такте синхронизации, когда TCNT2 примет нулевое значение. Флаг TOV2 в данном случае ведет себя подобно девятому биту, за исключением того, что он только устанавливается и не сбрасывается. Однако это свойство может быть использовано программно для повышения разрешающей способности таймера, если использовать прерывание по переполнению таймера, при возникновении которого флаг TOV2 сбрасывается автоматически. Для нормального режима работы не существует каких-либо особых ситуаций, связанных с записью нового состояния счетчика, когда требовалось бы учесть меры предосторожности.

Блок сравнения может использоваться для генерации прерываний. Не рекомендуется использовать выход OC2 для генерации сигналов в нормальном режиме работы, так как в этом случае будет затрачена значительная часть процессорного времени.

Режим сброса таймера при совпадении (CTC)

В режиме CTC (WGM21–0 = 2) регистр OCR2 используется для управления разрешающей способностью счетчика. Если задан режим CTC и значение счетчика (TCNT2) совпадает со значением регистра OCR2, то счетчик обнуляется. Таким образом, OCR2 задает верхний предел счета для счетчика, а, следовательно, и его разрешающую способность. В данном режиме обеспечивается более широкий диапазон регулировки частоты генерируемых импульсов. Он также упрощает операцию счета внешних событий.

Временная диаграмма для режима CTC показана на рисунке 3.66. Значение счетчика (TCNT2) увеличивается до тех пор, пока оно не становится равным значению в регистре OCR2, после чего счетчик (TCNT2) обнуляется.

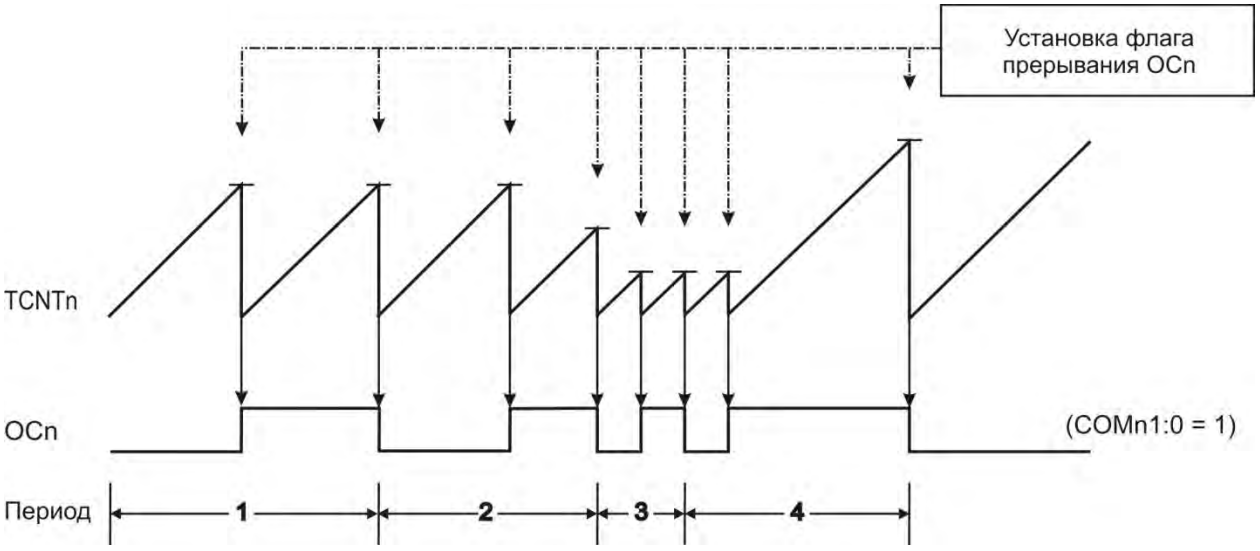


Рисунок 3.66 – Временная диаграмма для режима CTC

С помощью флага OCF2 прерывание может генерироваться всякий раз, когда счетчик достигает своего верхнего предела счета. Если разрешено прерывание, то процедура обработки прерывания может использоваться для обновления значения верхнего предела счета. Однако, задание значения верхнего предела счета, близкого к значению нижнего предела, когда счетчик работает без предделения или с малым значением предделения, необходимо выполнять с особой осторожностью, так как в режиме CTC нет двойной буферизации. Если новое значение, записанное в OCR2, будет меньше текущего значения TCNT2, то счетчик пропустит событие совпадения. В таком случае счетчик должен будет досчитать до своего максимального значения (0xFF) и перезапуститься, начав счет с 0x00, пока он не достигнет нового значения в регистре OCR2.

Для генерации сигнала в режиме CTC выход OC2 может использоваться для изменения логического уровня при каждом совпадении, для чего необходимо задать режим переключения (COM21–0 = 1). Значение OC2 не будет видимым на выводе порта, если направление данных для этого вывода не установлено как выход. Генерируемый сигнал будет иметь максимальную частоту $f_{OC2} = f_{clk\_I/O}/2$, когда значение регистра OCR2 станет равным нулю (0x00). Частота сигналов определяется по формуле:

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

где переменная N задает коэффициент деления предделителя (1, 8, 64, 256 или 1024).

Как и в нормальном режиме работы, флаг TOV2 устанавливается на том же самом такте таймера, когда значение счетчика меняется с 0xFF на 0x00.

Режим быстрой широтно-импульсной модуляции

Режим быстрой широтно-импульсной модуляции (WGM21–0 = 3) позволяет генерировать высокочастотные ШИМ-сигналы. От другого режима с ШИМ данный режим отличается однонаправленностью счета. Счетчик считает от нижнего предела до максимального значения, а затем перезапускается, начиная счет от нижнего предела. В неинвертирующем режиме сравнения вывод OC2 сбрасывается при совпадении TCNT2 и OCR2 и устанавливается при достижении нижнего предела счета. В инвертирующем режиме сравнения вывод OC2 устанавливается при совпадении TCNT2 и OCR2 и сбрасывается при достижении нижнего предела счета. Благодаря однонаправленности счета рабочая частота для режима быстрой ШИМ может быть в два раза выше по сравнению с режимом ШИМ с фазовой коррекцией, где используется двунаправленный счет. Возможность генерации высокочастот-

ных ШИМ-сигналов делает использование данного режима полезным в задачах стабилизации питания, коррекции и цифро-аналогового преобразования. Высокая частота при этом позволяет использовать внешние элементы физически малых размеров (катушки индуктивности, конденсаторы), снижая тем самым общую стоимость системы.

В режиме быстрой ШИМ содержимое счетчика инкрементируется до тех пор, пока не достигнет максимального значения (0xFF). Далее на последующем такте синхронизации таймера счетчик сбрасывается. Временная диаграмма для режима быстрой ШИМ показана на рисунке 3.67. Значение TCNT2 на временной диаграмме показано в виде гистограммы для иллюстрации однонаправленности счета. На диаграмме показаны как неинвертированный, так и инвертированный ШИМ-выходы. Короткими горизонтальными линиями обозначены положения на графике TCNT2, где значения OCR2 и TCNT2 совпадают.

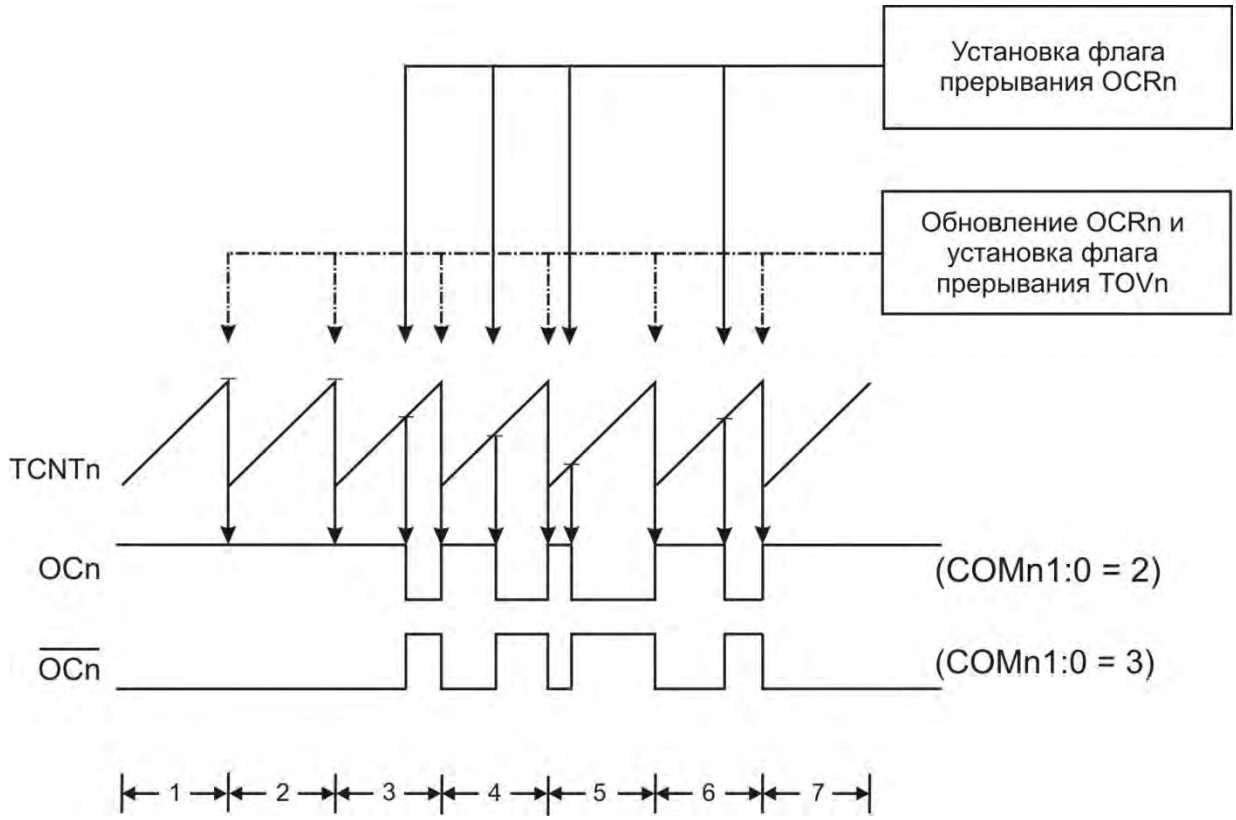


Рисунок 3.67 – Временная диаграмма для режима быстрой ШИМ

Флаг переполнения таймера/счетчика TOV2 устанавливается всякий раз, когда счетчик достигает своего максимального значения (0xFF). Если прерывание разрешено, то процедура обработки прерывания может использоваться для обновления сравниваемого значения.

В режиме быстрой ШИМ блок сравнения позволяет формировать ШИМ-сигналы на выводе OC2. Установка COM21–0 = 2 обеспечит формирование неинвертированного ШИМ-сигнал, инвертированный ШИМ-сигнал можно сформировать установкой COM21–0 = 3 (см. таблицу 3.66). Фактическое значение OC2 будет присутствовать на выводе порта, только если для данного вывода задано выходное направление. ШИМ-сигнал генерируется путем установки (или сброса) регистра OC2 при совпадении значений OCR2 и TCNT2, а также путем сброса (или установки) регистра OC2 на том же такте синхронизации таймера, когда счетчик сбрасывается (состояние счетчика изменяется от максимального значения до нижнего предела счета).

Частота выходного ШИМ-сигнала может быть вычислена по формуле

$$f_{OCnPWM} = \frac{f_{clk\_I/O}}{N \cdot 256},$$

где переменная N задает коэффициент деления предделителя (1, 8, 64, 256 или 1024).

Предельные значения регистра OCR2 представляют собой особые случаи при формировании ШИМ-сигналов в режиме быстрой ШИМ. Если в регистр OCR2 записать значение, равное нижнему пределу счета, то через каждые 256 тактов синхронизации таймера на выходе будет генерироваться узкий импульсный сигнал. Если установить значение OCR2 равным максимальному значению, то выход будет иметь постоянный логический уровень 1 или 0 (зависит от полярности сигнала на выходе, определяемой битами COM21–0).

Частотный выходной сигнал (с пятидесятипроцентным коэффициентом заполнения) в режиме быстрой ШИМ может быть достигнут путем установки OC2 на переключение своего логического уровня при каждом совпадении (COM21–0 = 1). Генерированный сигнал будет иметь максимальную частоту $f_{OC2} = f_{clk\_I/O}/2$, когда значение регистра OCR2 равно нулю. Данная функция похожа на переключение OC2 в режиме CTC за исключением того, что в режиме быстрой ШИМ разрешена двойная буферизация в блоке сравнения.

Режим широтно-импульсной модуляции с фазовой коррекцией (ШИМ ФК)

Режим ШИМ ФК (WGM21–0 = 1) позволяет выполнять фазовую коррекцию ШИМ-сигнала с высокой разрешающей способностью. Режим ШИМ ФК основан на двунаправленной работе таймера/счетчика. Счетчик циклически выполняет счет в направлении от нижнего предела до максимального значения, а затем обратно от максимального значения к нижнему пределу. В неинвертирующем режиме сравнения выход OC2 сбрасывается/устанавливается при совпадении значений TCNT2 и OCR2 во время прямого/обратного счета соответственно. В инвертирующем режиме сравнения, наоборот, во время прямого счета происходит установка, а во время обратного – сброс выхода OC2. При двунаправленном счете максимальная частота ШИМ-сигнала меньше, чем при однонаправленном счете. Однако, благодаря симметричности режимов ШИМ с двунаправленной работой, данные режимы предпочтительны при решении задач управления приводами.

Разрешающая способность ШИМ для данного режима фиксирована и равна 8 битам. В режиме ШИМ ФК счетчик инкрементируется, пока не достигнет максимального значения (0xFF). По достижении максимального значения счетчик меняет направление счета. Значение TCNT2 остается равным максимальному значению в течение одного такта синхронизации таймера. На рисунке 3.68 показана временная диаграмма работы таймера/счетчика для режима ШИМ ФК. Значение TCNT2 представлено в виде гистограммы для иллюстрации двунаправленности работы счетчика. Диаграмма включает неинвертированный и инвертированный ШИМ-выходы. Короткие горизонтальные линии на графике изменения TCNT2 указывают положения, где значения OCR2 и TCNT2 совпадают.

Флаг переполнения таймера/счетчика (TOV2) устанавливается всякий раз, когда счетчик достигает нижнего предела, и может использоваться для генерации прерывания.

В режиме ШИМ ФК блок сравнения позволяет формировать ШИМ-сигналы на выводе OC2. Установка битов COM21–0 = 2 обеспечивает генерацию неинвертированного ШИМ-сигнала. Инвертированный ШИМ-сигнал может быть сгенерирован установкой COM21–0 = 3 (см. таблицу 3.67). Фактическое значение OC2 будет видимым на соответствующем выводе порта, только если направление данных для этого вывода установлено как выход. ШИМ-сигнал генерируется путем сброса (или установки) регистра OC2 при совпадении OCR2 и TCNT2 во время прямого счета и путем установки (или сброса) регистра OC2 при совпадении OCR2 и TCNT2 во время обратного счета.

Результирующая частота ШИМ-сигнала в режиме ШИМ ФК может быть вычислена по формуле

$$f_{OCnPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510},$$

где переменная N задает коэффициент деления предделителя (1, 8, 64, 256 или 1024).

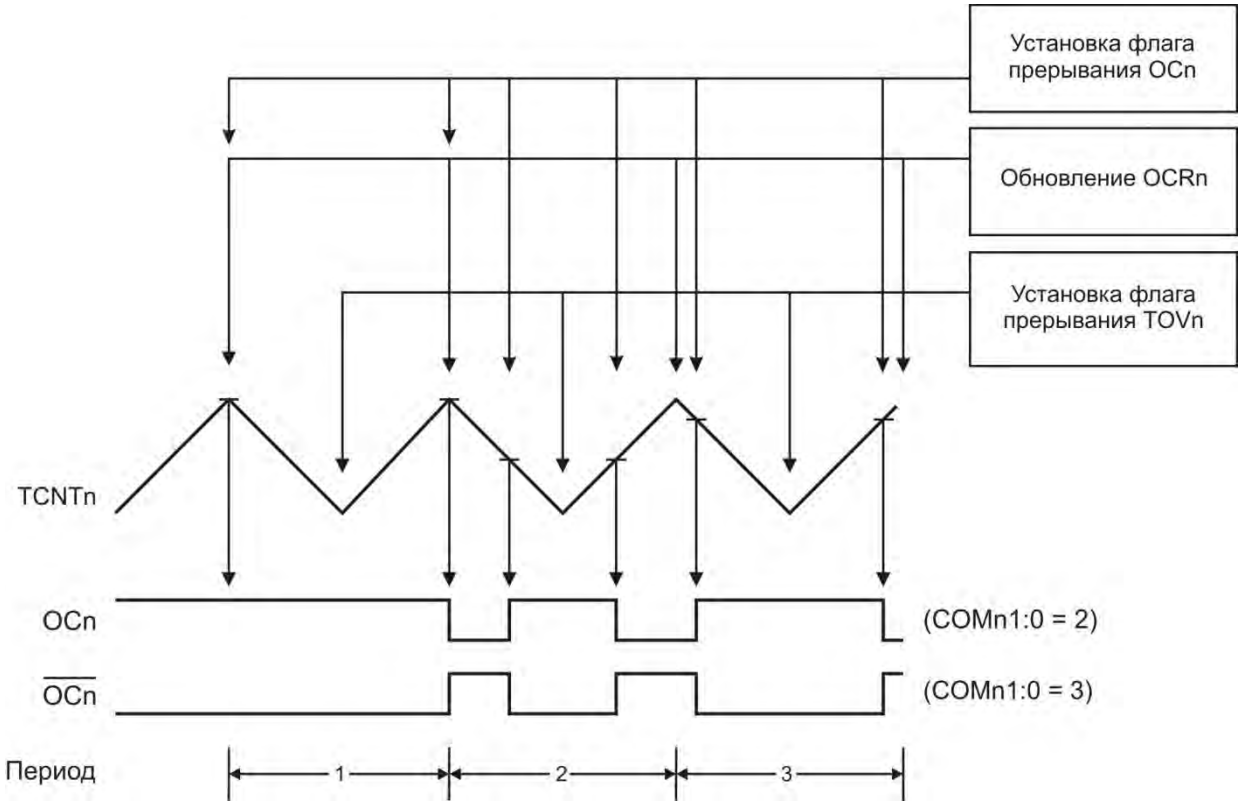


Рисунок 3.68 – Временная диаграмма для режима ШИМ ФК

Предельные значения регистра OCR2 представляют собой особые случаи при формировании ШИМ-сигналов в режиме ШИМ ФК. Если установить значение регистра OCR2 равным нижнему пределу, то выход будет постоянно иметь низкий уровень, а если установить равным максимальному значению, то выход будет постоянно иметь высокий уровень в неинвертирующем режиме широтно-импульсной модуляции. В инвертирующем режиме выход будет иметь противоположные логические значения.

В самом начале периода 2 на рисунке 3.68 сигнал OCn переходит из единицы в ноль, несмотря на то, что совпадения нет. Точка данного перехода должна гарантировать симметричность относительно нижней границы счета. Существует два случая изменения логического уровня сигнала, когда нет совпадения значений OCR2 и TCNT2:

- OCR2 изменяет свое значение с максимального, как на рисунке 3.68. Когда значение OCR2 равно максимальному, значение вывода OCn будет таким же, как результат совпадения при обратном счете. Для обеспечения симметрии относительно нижнего предела значение OCn при максимальном значении счетчика должно соответствовать результату совпадения при прямом счете.

- Таймер начинает счет со значения, большего по сравнению с OCR2, по этой причине пропуская событие совпадения. Отсюда следует изменение OCn, которое должно было бы произойти во время прямого счета.

3.12.7 Временные диаграммы таймера/счетчика 2

На рисунках представлена информация о том, когда происходит установка флагов прерывания. На рисунке 3.69 показаны временные характеристики основной работы таймера/счетчика. Данный рисунок иллюстрирует счетную последовательность в области максимального значения счетчика (0xFF) для всех режимов, кроме режима ШИМ ФК. На рисунке 3.70 показаны те же временные характеристики, но при включенном предделителе.

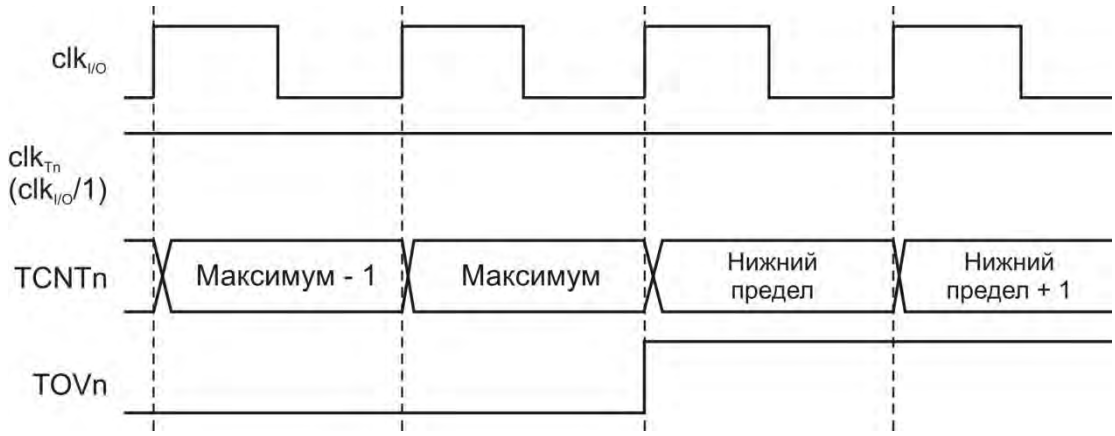


Рисунок 3.69 – Временная диаграмма таймера/счетчика без предделения

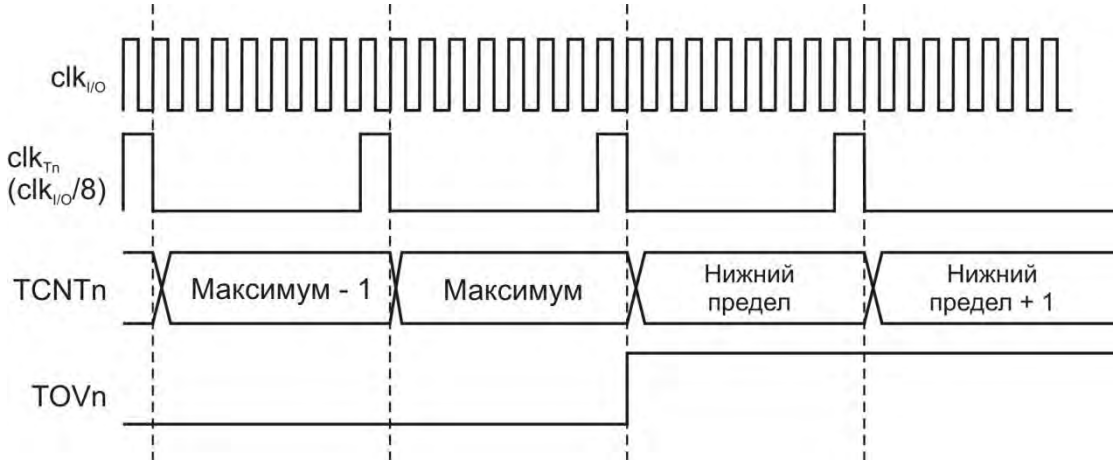


Рисунок 3.70 – Временная диаграмма таймера/счетчика с предделением ($f_{clk\_I/O}/8$)

На рисунке 3.71 отображена установка флага OCF2 во всех режимах, кроме режима СТС.

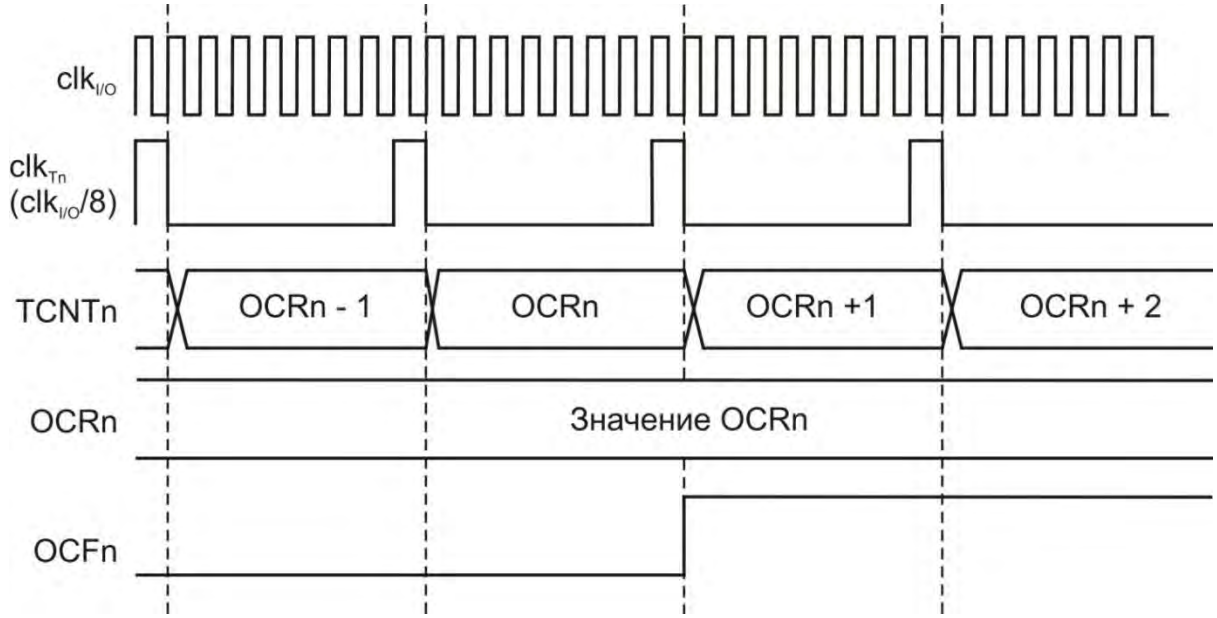


Рисунок 3.71 – Временная диаграмма таймера/счетчика, установка флага OCF2, с предделением ($f_{clk\_I/O}/8$)

На рисунке 3.72 отображена установка флага OCF2 и сброс TCNT2 в режиме CTC.

Рисунок 3.72 – Временная диаграмма таймера/счетчика, с предделением ($f_{clk\_I/O}/8$), в режиме сброса таймера при совпадении

3.12.8 Описание регистров таймера/счетчика 2

Регистр управления таймером/счетчиком 2 – TCCR2

Бит	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Чтение/ Запись	З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – FOC2: Установка принудительного совпадения

Функция бита FOC2 активна, только если бит WGM20 задает один из режимов, где нет широтно-импульсной модуляции. Если записать логическую единицу в бит FOC2, то это приведет к немедленной установке принудительного совпадения на входе блока формирования выходного сигнала. Выход OC2 изменяется в соответствии с установками битов COM21, COM20. Значение, представленное в COM21–0, определяет эффект принудительного совпадения.

Бит FOC2 не генерирует каких-либо прерываний, а также не вызывает сброс таймера в режиме CTC, где регистр OCR2 задает верхний предел счета.

Бит FOC2 всегда считывается как ноль.

Разряды 6, 3 – WGM21–0: Режим работы таймера/счетчика 2

Эти биты определяют счетную последовательность счетчика, источник верхнего предела значения счетчика и тип генерируемых сигналов. Режимами работы, поддерживаемыми блоком таймера/счетчика, являются: нормальный режим, режим сброса таймера при совпадении и два типа режимов с широтно-импульсной модуляцией (см. таблицу 3.64 и 3.12.6 «Режимы работы»).

Таблица 3.64 – Описание битов режима работы таймера/счетчика 2

Режим	WGM21	WGM20	Режим работы таймера/ счетчика	Верхний предел	Обновление значения OCR2	Установка флага TOV2
0	0	0	Нормальный	0xFF	немедленно	при достижении максимального значения
1	0	1	ШИМ ФК	0xFF	при достижении верхнего предела	при достижении нижнего предела
2	1	0	CTC	OCR2	немедленно	при достижении максимального значения
3	1	1	Быстрая ШИМ	0xFF	при достижении нижнего предела	при достижении максимального значения

Разряды 5, 4 – COM21–0: Режим формирования выходного сигнала

Эти биты управляют поведением вывода сравнения (OC2). Если один или оба бита COM21–0 установлены, то таймер-счетчик переопределяет нормальное функционирование линии ввода-вывода, к которой подключен выход OC2. Однако обратите внимание, что бит регистра направления данных (DDR), соответствующий выводу OC2, должен быть установлен, чтобы включить выходной драйвер.

После активизации альтернативной функции назначение битов COM21–0 зависит от установки битов WGM21–0. В таблице 3.65 отображается функционирование битов COM21–0, когда с помощью WGM21–0 выбран нормальный режим работы или режим CTC (режимы без ШИМ).

Таблица 3.65 – Режим формирования выходного сигнала, режим без ШИМ

COM21	COM20	Описание
0	0	Нормальная работа порта, OC2 отключен
0	1	Переключение OC2 при совпадении
1	0	Сброс OC2 при совпадении
1	1	Установка OC2 при совпадении

В таблице 3.66 отображается функционирование битов COM21–0, когда с помощью WGM21–0 выбран режим быстрой широтно-импульсной модуляции.

Таблица 3.66 – Режим формирования выходного сигнала, режим быстрой ШИМ

COM21	COM20	Описание
0	0	Нормальная работа порта, OC2 отключен
0	1	Зарезервировано
1	0	Сброс OC2 при совпадении, установка OC2 при достижении нижнего предела (неинвертирующий режим)
1	1	Установка OC2 при совпадении, сброс OC2 при достижении нижнего предела (инвертирующий режим)

Примечание – Особый случай возникает, когда значение OCR2 равно верхнему пределу и установлен бит COM21. В данном случае совпадение игнорируется, но установка или сброс происходят при достижении нижнего предела (см. выше подпункт «Режим быстрой широтно-импульсной модуляции» для изучения деталей).

В таблице 3.67 отображается функционирование битов COM21:0, когда с помощью WGM21:0 выбран режим широтно-импульсной модуляции с фазовой коррекцией.

Таблица 3.67 – Режим формирования выходного сигнала, режим ШИМ ФК

COM21	COM20	Описание
0	0	Нормальная работа порта, OC2 отключен
0	1	Зарезервировано
1	0	Сброс OC2 при совпадении во время прямого счета. Установка OC2 при совпадении во время обратного счета
1	1	Установка OC2 при совпадении во время прямого счета. Сброс OC2 при совпадении во время обратного счета

Примечание – Особый случай возникает, когда значение OCR2 равно верхнему пределу и установлен бит COM21. В данном случае совпадение игнорируется, но установка или сброс происходят при достижении верхнего предела (см. выше подпункт «Режим широтно-импульсной модуляции с фазовой коррекцией (ШИМ ФК)» для изучения деталей).

Разряды 2–0 – CS22–0: Выбор частоты синхронизации таймера

С помощью трех настроечных битов имеется возможность выбрать различные тактовые частоты, кратные исходной частоте синхронизации (см. таблицу 3.68).

Таблица 3.68 – Выбор частоты синхронизации таймера/счетчика 2

CS22	CS21	CS20	Описание
0	0	0	Нет синхронизации (таймер/счетчик 2 остановлен)
0	0	1	$clk_{I/O}$/1 (без предделения)
0	1	0	$clk_{I/O}$/8 (с предделением)
0	1	1	$clk_{I/O}$/64 (с предделением)
1	0	0	$clk_{I/O}$/256 (с предделением)
1	0	1	$clk_{I/O}$/1024 (с предделением)

Окончание таблицы 3.68

CS22	CS21	CS20	Описание
1	1	0	Внешний тактовый источник на выводе T2. Синхронизация по спадающему фронту
1	1	1	Внешний тактовый источник на выводе T2. Синхронизация по нарастающему фронту

Если для тактирования таймера/счетчика выбран внешний вывод T2, то переключения на выводе T2 будут синхронизировать счетчик, даже если этот вывод сконфигурирован как выход. Данная функция позволяет управлять счетом программно.

Счетный регистр таймера/счетчика 2 – TCNT2

Бит	7	6	5	4	3	2	1	0	
	TCNT2[7:0]								TCNT2
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Счетный регистр таймера/счетчика 2 предоставляет прямой доступ как для операции чтения, так и для операции записи. Запись в регистр TCNT2 блокирует совпадение на последующем такте синхронизации таймера. Изменение содержимого счетчика (TCNT2) во время счета связано с риском потери результата сравнения между TCNT2 и регистром OCR2.

Регистр сравнения – OCR2

Бит	7	6	5	4	3	2	1	0	
	OCR2[7:0]								OCR2
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр сравнения содержит 8-разрядное значение, которое постоянно сравнивается со значением счетчика (TCNT2). Событие совпадения значений может использоваться для генерации прерывания по выполнению условия сравнения или для генерации прямоугольных импульсов на выводе OC2.

Регистр масок прерываний таймеров/счетчиков – TIMSK

Бит	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – OCIE2: Разрешение прерывания по совпадению таймера/счетчика 2

Если OCIE2 = 1, а также установлен бит I в регистре статуса, то прерывание по совпадению таймера/счетчика 2 разрешается. В этом случае прерывание возникает, если обнаруживается совпадение значения таймера/счетчика 2 с значением сравнения, т.е. когда установлен флаг OCF2 в регистре флагов прерываний таймеров/счетчиков TIFR.

Разряд 6 – TOIE2: Разрешение прерывания по переполнению таймера/счетчика 2

Если TOIE2 = 1, а также установлен бит I в регистре статуса, то прерывание по переполнению таймера/счетчика 2 разрешается. В этом случае прерывание возникает, если обнаруживается переполнение таймера/счетчика 2, т.е. когда установлен бит TOV2 в регистре флагов прерываний таймеров/счетчиков TIFR.

Регистр флагов прерываний таймеров/счетчиков – TIFR

Бит	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – OCF2: Флаг совпадения таймера/счетчика 2

Флаг OCF2 устанавливается в единицу, когда обнаруживается совпадение между значением таймера/счетчика 2 и данными в регистре OCR2 (регистре сравнения). Флаг OCF2 сбрасывается аппаратно при переходе на соответствующий вектор обработки прерывания. Альтернативно, флаг OCF2 можно сбросить путем записи логической единицы в соответствующий разряд регистра. Если установлены бит I в регистре SREG, бит OCIE2 (разрешено прерывание по выполнению условия сравнения таймера/счетчика 2) и флаг OCF2, то генерируется прерывание по выполнению условия сравнения таймера/счетчика 2.

Разряд 6 – TOV2: Флаг переполнения таймера/счетчика 2

Флаг TOV2 устанавливается в единицу, когда в таймере/счетчике 2 возникает переполнение. Флаг TOV2 сбрасывается аппаратно при переходе на соответствующий вектор обработки прерывания. Альтернативно, флаг TOV2 можно сбросить путем записи логической единицы в соответствующий разряд регистра. Если установлены бит I в регистре SREG, бит TOIE2 (разрешено прерывание по переполнению таймера/счетчика 2) и флаг TOV2, то генерируется прерывание по переполнению таймера/счетчика 2. В режиме с ШИМ данный флаг устанавливается, когда таймер/счетчик 2 изменяет направление счета при значении $00.

3.13 Модулятор выходов сравнения (OCM1C2)

3.13.1 Краткий обзор

Модулятор выходов сравнения (OCM) позволяет генерировать прямоугольные импульсы, модулированные по несущей частоте. Модулятор использует выходы канала сравнения С 16-разрядного таймера/счетчика 1 и выход блока сравнения 8-разрядного таймера/счетчика 2.

Если работа модулятора разрешена, то сигналы с выходов каналов сравнения модулируются вместе, как показано на рисунке 3.73.

3.13.2 Описание

Блоки сравнения 1С и 2 используют один и тот же вывод порта PB7 в качестве выхода. Выходы блоков сравнения (OC1C и OC2) блокируют обычную функцию регистра PORTB7 после разрешения работы одного из них (то есть когда бит COMnx1–0 не равен нулю). Когда одновременно разрешается работа и OC1C, и OC2, модулятор включается автоматически.

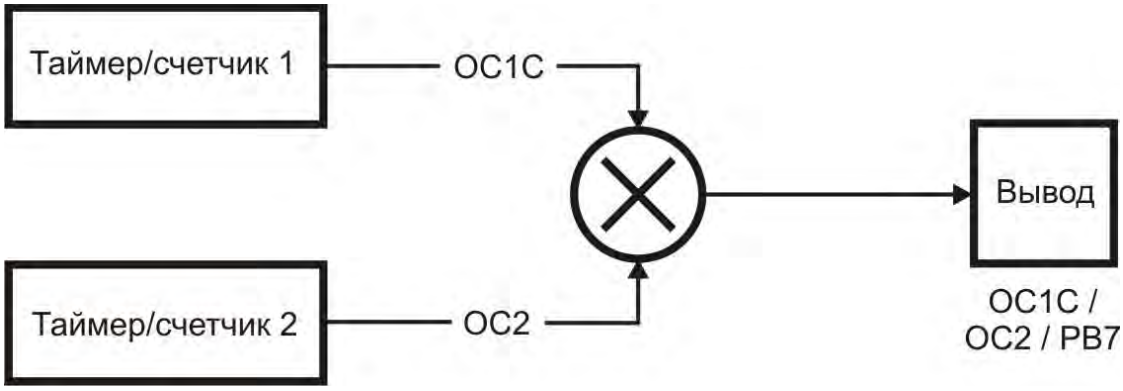


Рисунок 3.73 – Блок-схема модулятора сравнения

Эквивалентная функциональная схема модулятора выходов сравнения представлена на рисунке 3.74. На рисунке показаны элементы блоков таймеров/счетчиков и схема выходного драйвера линии 7 порта B.

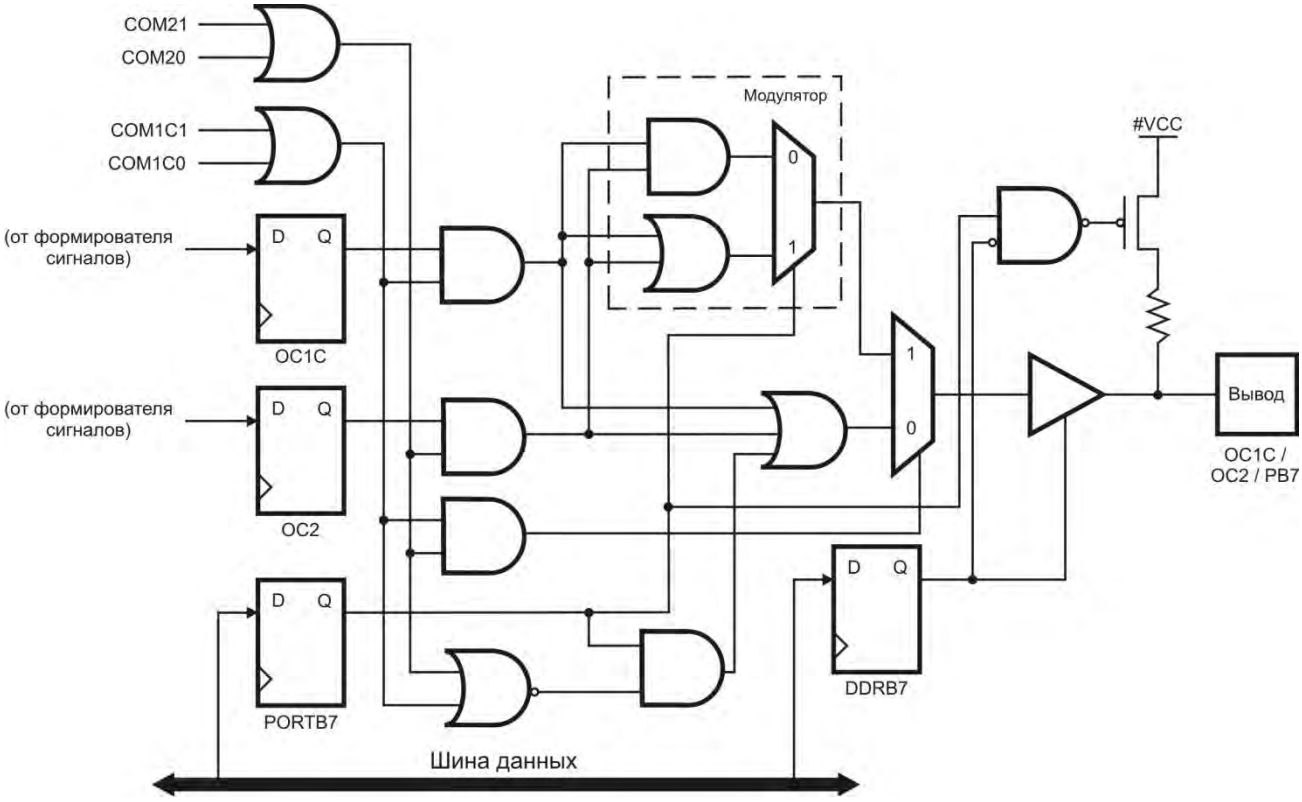


Рисунок 3.74 – Эквивалентная функциональная схема модулятора выходов сравнения

После разрешения работы модулятора необходимо выбрать тип модуляции (логическое умножение (операция И) или логическое сложение (операция ИЛИ)) с помощью регистра PORTB7. **Обратите внимание,** что DDRB7 управляет направлением порта независимо от установок битов COMnx1:0.

Пример временной диаграммы

Рисунок 3.75 иллюстрирует работу модулятора. В данном примере таймер/счетчик 1 настроен на работу в режиме быстрой ШИМ (без инверсии), а таймер/счетчик 2 использует режим СТС с переключением выхода компаратора при совпадении (COMnx1–0 = 1).

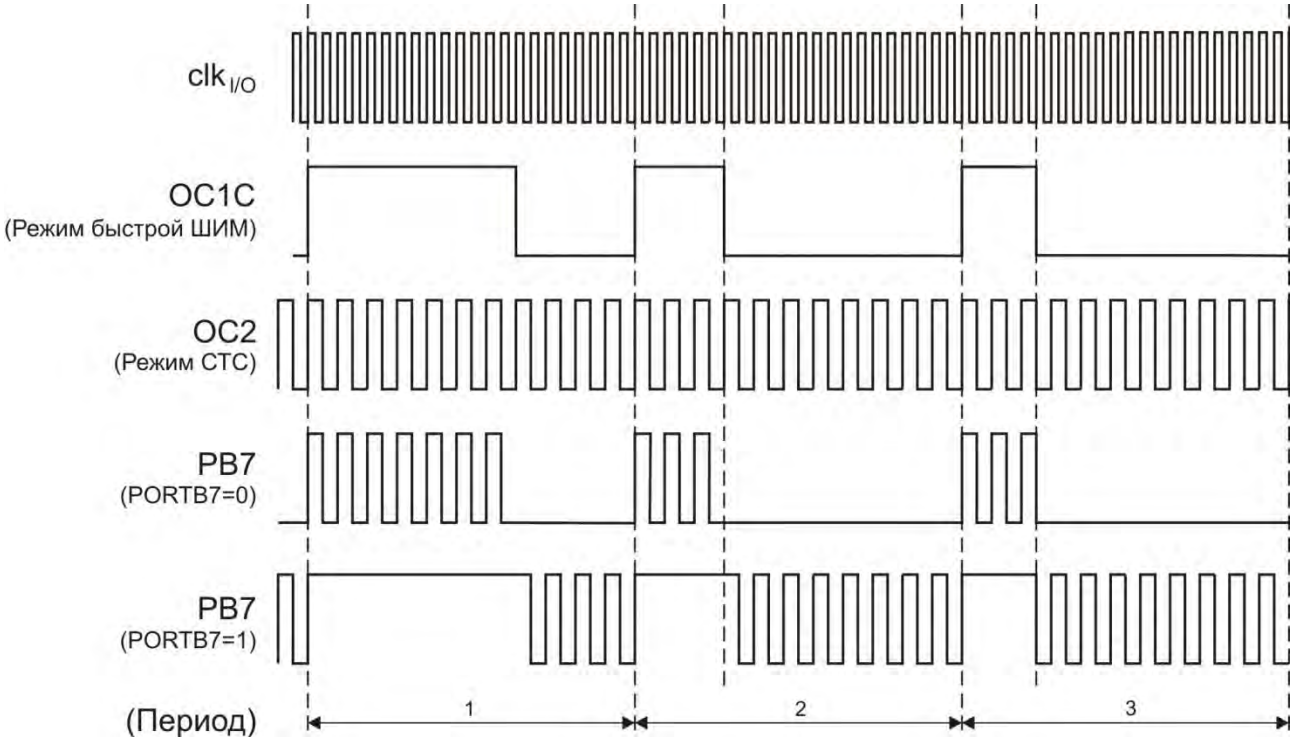


Рисунок 3.75 – Временная диаграмма работы модулятора сравнения

В данном примере таймер/счетчик 2 обеспечивает несущий сигнал, в то время как модулирующий сигнал генерируется каналом сравнения С таймера/счетчика 1.

Разрешающая способность ШИМ-сигнала (OC1C) снижается за счет модуляции. Коэффициент снижения эквивалентен числу тактовых импульсов системной синхронизации в течение одного периода несущего сигнала (OC2). В данном примере коэффициент снижения равен двум. Причина снижения разрешающей способности показана на рисунке 3.75 во втором и третьем периодах сигнала на выходе PB7, когда состояние PORTB7 равно нулю. Длительность высокого уровня сигнала OC1C во втором периоде на один такт системной синхронизации дольше длительности высокого уровня данного сигнала в третьем периоде, но форма сигнала на выводе PB7 идентична в каждом из этих периодов.

3.14 Последовательный периферийный интерфейс

Последовательный периферийный интерфейс (SPI) обеспечивает высокоскоростную синхронную передачу данных между ИС 1887ВЕ7Т и периферийными устройствами или между несколькими микроконтроллерами. Последовательный периферийный интерфейс в ИС 1887ВЕ7Т имеет следующие особенности:

- Полнодуплексная трехпроводная синхронная передача данных.
- Работа в качестве ведущего или ведомого.
- Передача первым младшего или старшего значащего бита.
- Семь программируемых скоростей передачи данных.
- Флаг прерывания для индикации окончания передачи данных.
- Флаг защиты от конфликта данных при записи.
- Пробуждение из режима холостого хода (Idle).
- Режим ведущего SPI с удвоением скорости (CK/2).

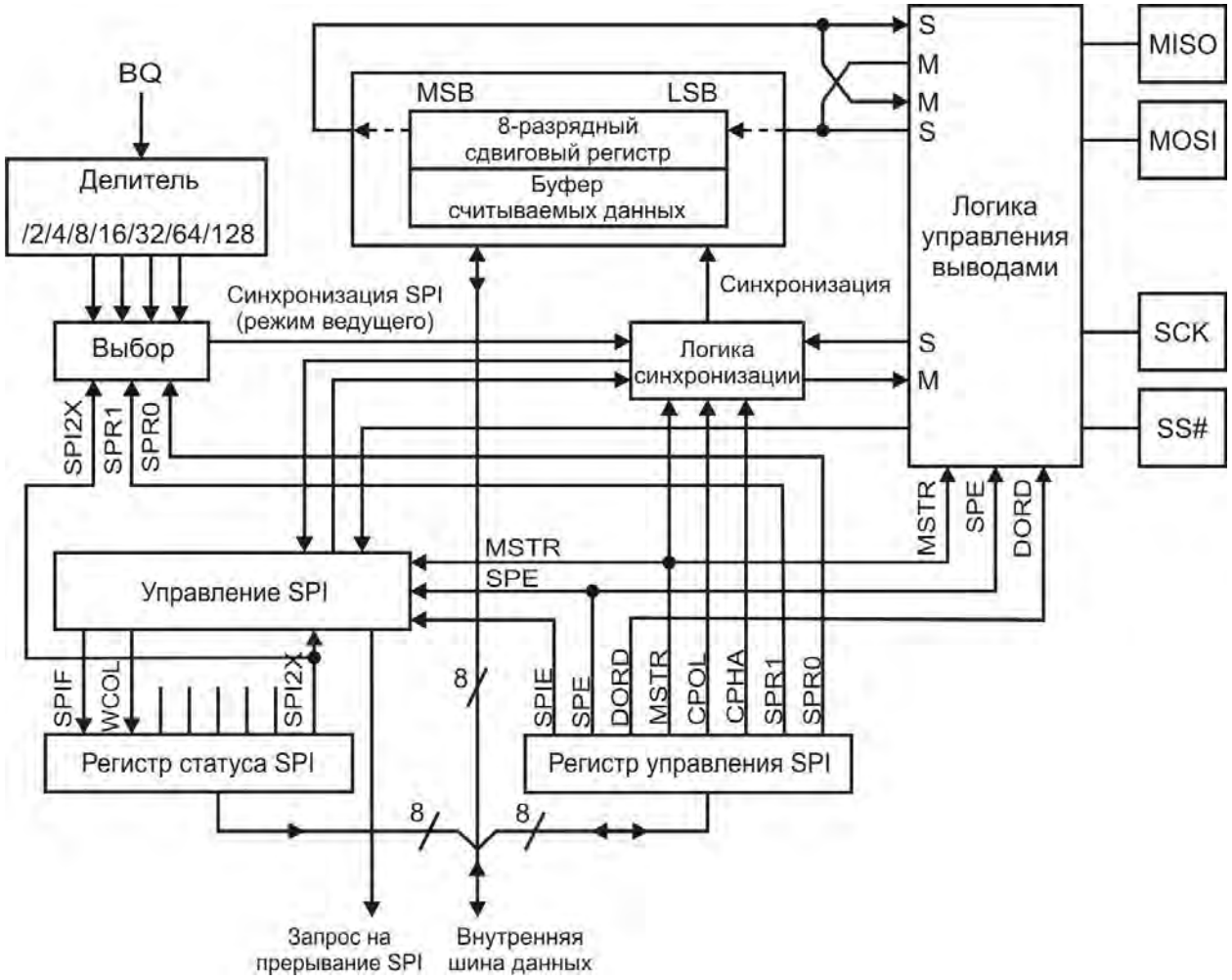


Рисунок 3.76 – Структурная схема последовательного периферийного интерфейса

Соединение двух микроконтроллеров (ведущий - ведомый) через интерфейс SPI показано на рисунке 3.77. Система состоит из двух сдвиговых регистров и тактового генератора ведущего SPI. Ведущий SPI инициирует сеанс связи подачей низкого уровня на вывод SS# того ведомого устройства, с которым необходимо обмениваться данными. Оба устройства (ведущее и ведомое) подготавливают данные к передаче в своем сдвиговом регистре, при этом ведущий SPI генерирует необходимые импульсы синхронизации на линии SCK для обмена данными. По линии MOSI всегда осуществляется передача данных от ведущего устройства к ведомому, а по линии MISO, наоборот, от ведомого к ведущему. По окончании передачи каждого пакета данных ведущий SPI синхронизирует ведомый путем подачи высокого уровня на линию SS#.

Если SPI сконфигурирован как ведущий, то управление линией SS# происходит не автоматически. Данная операция должна быть выполнена программно перед началом сеанса связи. После этого запись байта в регистр данных SPI запускает тактовый генератор SPI, и восемь битов аппаратно сдвигаются в ведомое устройство. После сдвига одного байта тактовый генератор SPI останавливается, при этом устанавливая флаг окончания передачи (SPIF). Если установлен бит разрешения прерывания SPI (SPIE) в регистре SPCR, то генерируется запрос на прерывание. Главное устройство может продолжить сдвигать следующий байт путем записи его в регистр SPDR, или же устройство может подать сигнал окончания передачи пакета данных путем установки низкого уровня на линии SS#. Последний принятый байт сохраняется в буферном регистре для дальнейшего использования.

В режиме ведомого интерфейс SPI находится в состоянии сна, в котором линия MISO пребывает в третьем состоянии, пока на выводе SS# присутствует высокий уровень. В этом состоянии программа может обновлять содержимое регистра данных SPI (SPDR), но

при этом поступающие тактовые импульсы на линии SCK не будут сдвигать данные, пока не появится низкий уровень на выводе SS#. Когда происходит полный сдвиг одного байта, устанавливается флаг окончания передачи SPIF. Если установлен бит разрешения прерывания SPI (SPIE) в регистре SPCR, то генерируется запрос на прерывание. Подчиненное устройство может продолжить размещать новые данные для передачи в регистр SPDR перед чтением входящих данных. Последний принятый байт сохраняется в буферном регистре для дальнейшего использования.

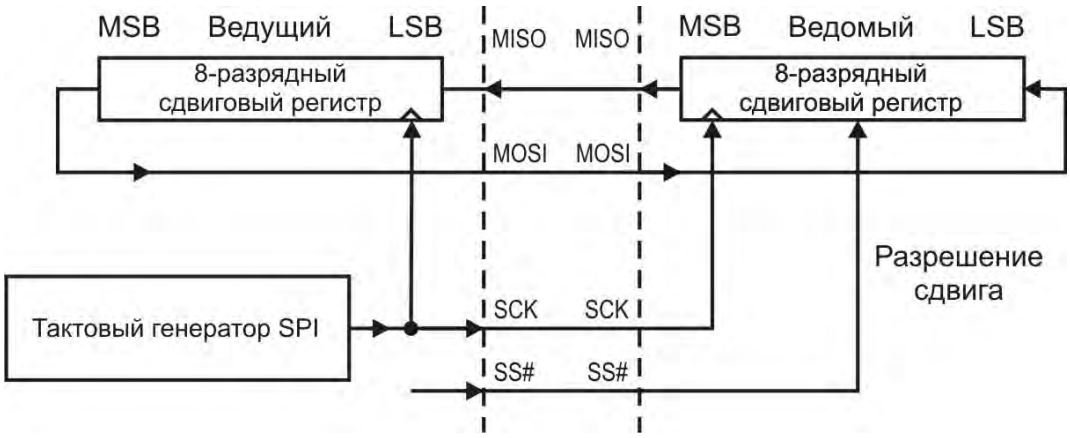


Рисунок 3.77 – Внутреннее соединение ведущего и ведомого SPI

В направлении передачи данных система выполнена как однобуферная, а в направлении приема используется двойная буферизация. Это означает, что передаваемые байты не могут быть записаны в регистр данных SPI, пока не завершится полный цикл сдвига. Однако во время приема информации полученный пакет данных должен быть считан из регистра SPDR перед тем, как произойдет полный сдвиг следующего пакета. В противном случае первый байт будет потерян.

В режиме ведомого SPI логика управления осуществляет выборку входящего сигнала на выводе SCK. Для обеспечения корректной выборки тактового сигнала минимальная длительность как периода низкого уровня, так и периода высокого уровня должна быть дольше, чем два такта системной синхронизации.

Если разрешена работа SPI, то разрешается альтернативная функция выводов MOSI, MISO, SCK и SS# (см. таблицу 3.69).

Таблица 3.69 – Направление выводов SPI

Вывод	Направление (ведущий SPI)	Направление (ведомый SPI)
MOSI	Определяется пользователем	Вход
MISO	Вход	Определяется пользователем
SCK	Определяется пользователем	Вход
SS#	Определяется пользователем	Вход
Примечание – См. подпункт «Альтернативные функции порта B» в подразделе 3.7 «Порты ввода-вывода» для получения подробностей относительно того, как определить направление выводов SPI, задаваемых пользователем.		

В следующих двух примерах показано, как инициализировать SPI в качестве ведущего и выполнить простую передачу данных. Во всех нижеприведенных примерах DDR_SPI должен быть заменен именем фактического регистра направления данных, управляющим выводами SPI. DD_MOSI, DD_MISO и DD_SCK также должны быть заменены именами соответствующих битов направления данных, связанных с этими выводами. Например, если MOSI размещен на выводе PB5, то в программе DD_MOSI необходимо заменить на DDB5, а DDR_SPI на DDRB.

Пример кода на Ассемблере:

```
SPI_MasterInit:
; Настройка линий MOSI и SCK на вывод данных, а всех остальных – на ввод
ldi r17,(1<<DD_MOSI)|(1<<DD_SCK)
out DDR_SPI,r17
; Включение SPI в режиме ведущего, установка тактовой частоты fck/16
ldi r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
out SPCR,r17
ret

SPI_MasterTransmit:
; Запуск передачи данных (r16)
out SPDR,r16
Wait_Transmit:
; Ожидание завершения передачи данных
sbis SPSR,SPIF
rjmp Wait_Transmit
ret
```

Пример кода на Си:

```
void SPI_MasterInit(void)
{
/* Настройка линий MOSI и SCK на вывод данных, а всех остальных – на ввод */
DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
/* Включение SPI в режиме ведущего, установка тактовой частоты fck/16 */
SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
/* Запуск передачи данных */
SPDR = cData;
/* Ожидание завершения передачи данных */
while(!(SPSR & (1<<SPIF)))
;
}
```

Примечание – При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

В следующих двух примерах показано, как инициализировать SPI в качестве ведомого и выполнить простой прием данных.

Пример кода на Ассемблере:

```
SPI_SlaveInit:
; Настройка линии MISO на вывод данных, а всех остальных линий – на ввод
ldi r17,(1<<DD_MISO)
out DDR_SPI,r17
; Включение SPI
ldi r17,(1<<SPE)
out SPCR,r17
ret

SPI_SlaveReceive:
; Ожидание завершения приема данных
```

```
sbis SPSR,SPIF
rjmp SPI_SlaveReceive
; Чтение принятых данных и возврат из процедуры
in r16,SPDR
ret
```

Пример кода на Си:

```
void SPI_SlaveInit(void)
{
/* Настройка линии MISO на вывод данных, а всех остальных линий – на ввод */
DDR_SPI = (1<<DD_MISO);
/* Включение SPI */
SPCR = (1<<SPE);
}

char SPI_SlaveReceive(void)
{
/* Ожидание завершения приема данных */
while(!(SPSR & (1<<SPIF)))
;
/* Возврат к регистру данных */
return SPDR;
}
```

Примечание – При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

3.14.1 Функционирование вывода SS#

Режим ведомого

Когда SPI сконфигурирован как ведомый, вывод выбора режима ведомого SS# всегда работает как вход. Когда сигнал на входе SS# удерживается в низком уровне, активизируется SPI и вывод MISO становится выходом, если так запрограммировано пользователем. Все остальные выводы работают как входы. Если на вход SS# подать высокий уровень, то все выводы станут входами за исключением MISO, который может быть сконфигурирован пользователем в качестве выхода, и SPI перейдет в пассивное состояние, в котором блокируется прием входящих данных. Обратите внимание, что логика SPI сбрасывается, как только на вывод SS# подается высокий логический уровень.

Вывод SS# удобно использовать для пакетной/байтной синхронизации, что позволяет поддерживать синхронность работы счетчика битов ведомого и тактового генератора ведущего устройств. Если на вывод SS# подать высокий логический уровень, то ведомый SPI немедленно сбросит логику передачи и приема и удалит частично полученные данные в сдвиговом регистре.

Режим ведущего

Когда SPI сконфигурирован как ведущий (установлен бит MSTR в регистре SPCR), то пользователь может задавать желаемое направление для вывода SS#.

Если вывод SS# настроен как выход, то он работает как общая линия выхода и не оказывает влияния на SPI-систему. Обычно он используется для управления выводом SS# ведомого SPI.

Если вывод SS# настроен как вход, то он должен удерживаться на высоком логическом уровне для обеспечения работы ведущего SPI. Если на вывод SS# будет подан низкий логический уровень при помощи периферийной схемы, когда SPI сконфигурирован как ве-

дущий с выводом SS#, настроенным как вход, то система SPI интерпретирует это, как будто другой ведущий выбирает SPI в качестве ведомого и начинает передачу данных к нему. Для того чтобы избежать конфликтной ситуации при обращении к шине, система SPI выполняет следующие действия:

1 Бит MSTR в регистре SPCR сбрасывается, и SPI переводится в режим ведомого. В результате выводы MOSI и SCK становятся входами.

2 Устанавливается флаг SPIF в регистре SPSR и, если разрешено прерывание SPI и установлен бит I в регистре SREG, выполняется процедура обработки прерывания.

Таким образом, если используется передача SPI в режиме ведущего с управлением по прерываниям и предусмотрена возможность подачи низкого уровня на вход SS#, то при генерации прерывания необходимо всегда проверять состояние бита MSTR. Если MSTR оказался сброшенным, то это означает, что SPI был переведен в режим ведомого внешним устройством и пользователь должен предусмотреть возобновление режима ведущего SPI программным путем.

Регистр управления SPI – SPCR

Бит	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – SPIE: Разрешение прерывания SPI

Если установлены флаг SPIF в регистре SPSR и бит общего разрешения прерываний I в регистре SREG, то установка бита SPIE приведет к выполнению процедуры обработки прерывания SPI.

Разряд 6 – SPE: Включение SPI

При записи логической единицы в бит SPE разрешается работа SPI. Чтобы выполнять любые операции SPI, данный бит должен быть установлен.

Разряд 5 – DORD: Порядок сдвига данных

Если DORD = 1, то при передаче слова данных первым передается младший значащий разряд LSB. Если же DORD = 0, то первым передается старший значащий разряд MSB.

Разряд 4 – MSTR: Выбор ведущего/ведомого

Когда MSTR = 1, выбирается режим ведущего, а когда MSTR = 0, выбирается режим ведомого. Если вывод SS# настроен как вход и на него подается низкий логический уровень, пока установлен бит MSTR, то MSTR сбросится и установится флаг SPIF в регистре SPSR. Для возобновления режима ведущего SPI пользователь должен предусмотреть программную установку бита MSTR.

Разряд 3 – CPOL: Полярность синхронизации

Когда CPOL = 1, SCK имеет высокий логический уровень в состоянии ожидания. Когда CPOL = 0, SCK имеет низкий логический уровень в состоянии ожидания. См. примеры, иллюстрирующие отличия в полярности синхронизации, на рисунках 3.78 и 3.79. Ниже обобщено функционирование бита CPOL.

Таблица 3.70 – Функционирование бита CPOL

CPOL	Передний фронт	Задний фронт
0	Нарастающий	Спадающий
1	Спадающий	Нарастающий

Разряд 2 – CPHA: Фаза синхронизации

Значение бита фазы синхронизации (CPHA) определяет, по какому фронту сигнала SCK происходит выборка данных: по переднему или заднему. Примеры действия различных установок CPHA приведены на рисунках 3.78 и 3.79. Ниже обобщено функционирование бита CPHA.

Таблица 3.71 – Функционирование бита CPHA

CPHA	Передний фронт	Задний фронт
0	Выборка	Установка
1	Установка	Выборка

Разряды 1, 0 – SPR1, SPR0: Биты 1 и 0 выбора частоты синхронизации SPI

Данные биты задают частоту синхронизации на выводе SCK ведущего SPI. SPR1 и SPR0 не оказывают никакого влияния на ведомое устройство. Связь между частотой SCK и частотой синхронизации генератора f_{OSC} показана в таблице 3.72.

Таблица 3.72 – Связь между частотой сигнала SCK и частотой генератора

SPI2X	SPR1	SPR0	Частота SCK
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

Регистр статуса SPI – SPSR

Бит	7	6	5	4	3	2	1	0	
	SPIF	WCOL	-	-	-	-	-	SPI2X	SPSR
Чтение/ Запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – SPIF: Флаг прерывания SPI

Флаг SPIF устанавливается по завершении последовательной передачи. Прерывание генерируется в том случае, когда установлены бит SPIE в регистре SPCR и бит общего разрешения прерываний I в регистре SREG. Если вывод SS# настроен как вход и на него подается низкий логический уровень, когда SPI находится в режиме ведущего, то это также приводит к установке флага SPIF. Флаг SPIF сбрасывается аппаратно при переходе на соответствующий вектор обработки прерывания. Альтернативно, бит SPIF сбрасывается при чтении регистра статуса SPI с установленным флагом SPIF, с последующим доступом к регистру данных SPI (SPDR).

Разряд 6 – WCOL: Флаг защиты от конфликта данных при записи

Бит WCOL устанавливается, если выполняется запись в регистр данных SPI (SPDR) во время передачи данных. Бит WCOL, а также бит SPIF сбрасываются при чтении регистра статуса SPI с установленным флагом WCOL, с последующим доступом к регистру данных SPI (SPDR).

Разряды 5–1 – Зарезервированные биты

Данные биты являются резервными и всегда считываются как ноль.

Разряд 0 – SPI2X: Бит удвоения скорости SPI

Когда SPI2X = 1, скорость работы SPI (частота SCK) удваивается, если SPI находится в режиме ведущего (см. таблицу 3.72). Это означает, что минимальный период сигнала SCK будет равен двум периодам синхронизации ЦПУ. Когда SPI сконфигурирован как ведомый, то его работа гарантируется только на частоте $f_{OSC}/4$ или ниже.

Интерфейс SPI в микроконтроллере 1887ВЕ7Т также используется для программирования памяти программ и ЭПД.

Регистр данных SPI – SPDR

Бит	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	X	X	X	X	X	X	X	X	Не определено

Регистр данных SPI является регистром чтения/записи и используется для передачи данных между регистровым файлом и сдвиговым регистром SPI. Запись в регистр SPDR инициирует передачу данных. При чтении же данного регистра фактически считывается содержимое приемного буфера сдвигового регистра.

3.14.2 Режимы передачи данных

Комбинации битов CPHA и CPOL определяют четыре возможных режима последовательной передачи данных. Форматы передачи данных для SPI представлены в таблице 3.73, а их временные диаграммы показаны на рисунках 3.78 и 3.79. Биты данных сдвигаются и фиксируются противоположными фронтами синхросигнала SCK, гарантируя тем самым достаточное время для установления сигналов данных. Таким образом, можно обобщить информацию из таблиц 3.70 и 3.71 и представить ее в следующем виде:

Таблица 3.73 – Функционирование битов CPOL и CPHA

Значения битов CPOL и CPHA	Передний фронт	Задний фронт	Режим SPI
CPOL = 0, CPHA = 0	Выборка данных на нарастающем фронте	Установка данных на спадающем фронте	0
CPOL = 0, CPHA = 1	Установка данных на нарастающем фронте	Выборка данных на спадающем фронте	1
CPOL = 1, CPHA = 0	Выборка данных на спадающем фронте	Установка данных на нарастающем фронте	2
CPOL = 1, CPHA = 1	Установка данных на спадающем фронте	Выборка данных на нарастающем фронте	3

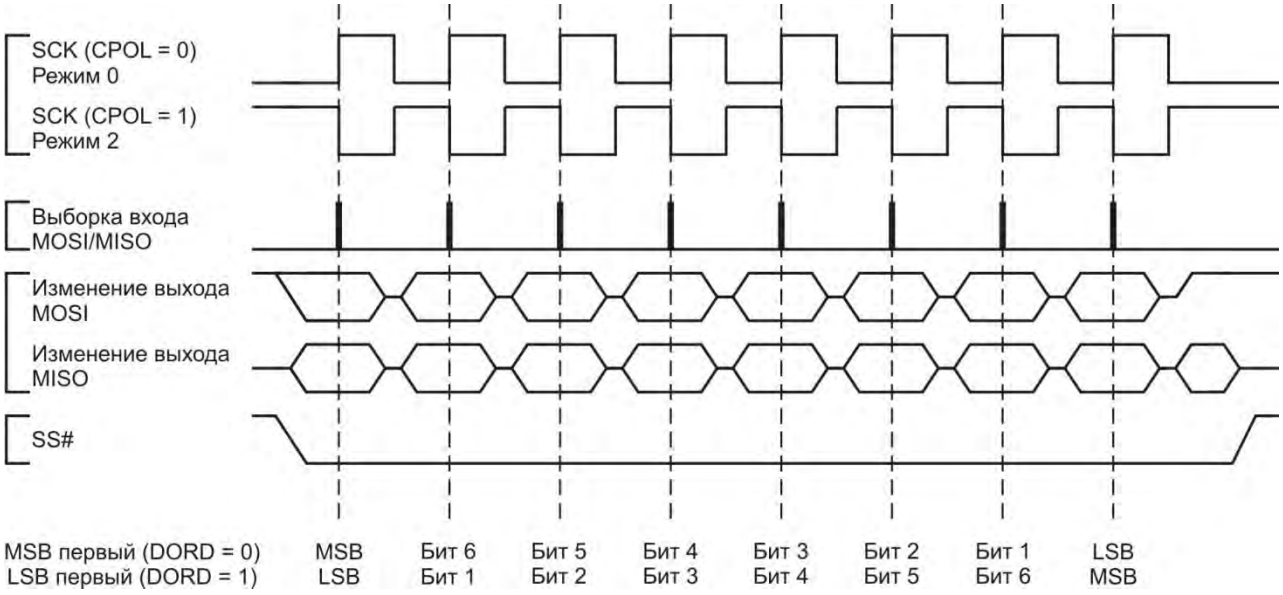


Рисунок 3.78 – Временная диаграмма передачи данных, когда CPHA = 0

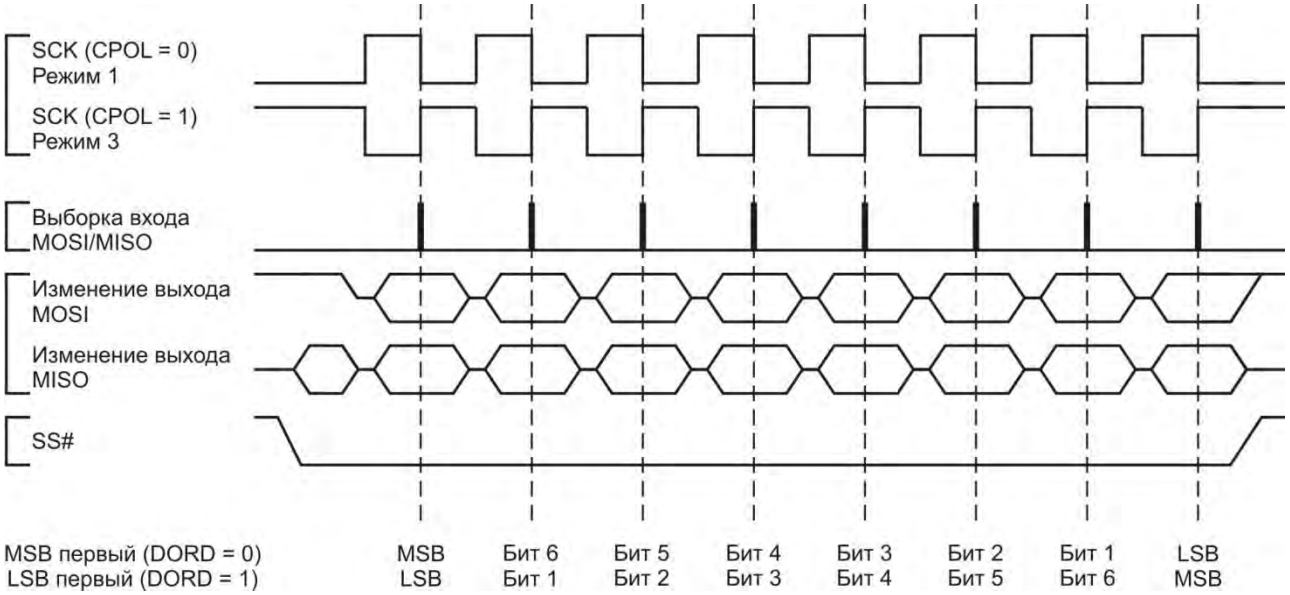


Рисунок 3.79 – Временная диаграмма передачи данных, когда CPHA = 1

3.15 Универсальный синхронно-асинхронный приемопередатчик

Универсальный синхронно-асинхронный приемопередатчик (USART) является гибким устройством последовательной связи. Основные отличительные особенности:

- Полнодуплексная работа (независимые последовательные регистры приема и передачи).
- Асинхронная или синхронная работа.
- Синхронная работа в ведущем или ведомом режиме.
- Контроллер скорости передачи с высокой разрешающей способностью.
- Поддержка последовательных кадров с 5, 6, 7, 8 или 9 информационными битами и 1 или 2 стоповыми битами.
- Формирование сигнала четности или нечетности и проверка на четность-нечетность, поддерживаемая аппаратным обеспечением.
- Детектирование переполнения данных.
- Детектирование ошибки в структуре кадра.

- Фильтрация шумов с детектированием ложного стартового бита и цифровым фильтром нижних частот.

- Три отдельных прерывания по завершении передачи, освобождении регистра передаваемых данных и завершении приема.

- Многопроцессорный режим связи.

- Режим асинхронной связи с удвоением скорости.

Микроконтроллер 1887BE7T содержит два модуля USART: USART0 и USART1. Функционирование обоих модулей описано ниже. Модули USART0 и USART1 имеют индивидуальные регистры ввода-вывода.

3.15.1 Краткий обзор

Упрощенная структурная схема универсального синхронно-асинхронного приемопередатчика представлена на рисунке 3.80. Регистры ввода-вывода и линии ввода-вывода, к которым имеет доступ ЦПУ, выделены жирным шрифтом.

На рисунке 3.80 пунктирной линией выделены три основных элемента USART: тактовый генератор, передатчик и приемник. Регистры управления являются общими для всех блоков. Логика генерации тактовых импульсов состоит из логики синхронизации, связанной с входом внешнего тактового генератора (используется при синхронной работе в ведомом режиме), и контроллера скорости передачи. Вывод XCK (синхронизация передачи) используется только в режиме синхронной передачи. Передатчик состоит из буфера одиночной записи, последовательного сдвигового регистра, блока формирования сигнала четности или нечетности и управляющей логики для обработки различных последовательных кадров. Буфер записи позволяет непрерывно передавать данные без каких-либо задержек между кадрами. Приемник является самым сложным элементом USART, поскольку в его состав входят блок восстановления синхронизации и блок восстановления данных. Блоки восстановления используются для асинхронного приема данных. Помимо названных блоков в состав приемника входят устройство проверки на четность-нечетность, управляющая логика, сдвиговый регистр и двухуровневый буфер приема (UDR). Приемник поддерживает те же форматы кадров, что и передатчик, а также может детектировать ошибку структуры кадра, переполнение данных и ошибки четности-нечетности.

3.15.2 Генерация тактовых импульсов

Логика генерации тактовых импульсов формирует основную синхронизацию для передатчика и приемника. USART поддерживает четыре режима работы синхронизации: нормальный асинхронный, асинхронный с удвоением скорости, синхронный ведущий и синхронный ведомый. Бит UMSEL в регистре C управления и статуса USART (UCSRC) позволяет выбирать асинхронную или синхронную работу. Удвоение скорости (только в асинхронном режиме) управляется битом U2X в регистре UCSRA. При использовании синхронного режима (UMSEL = 1) бит в регистре направления данных, соответствующий выводу XCK (DDR_XCK), определяет, будет ли источник синхронизации внутренним (ведущий режим) или внешним (ведомый режим). Вывод XCK активен только при использовании синхронного режима.

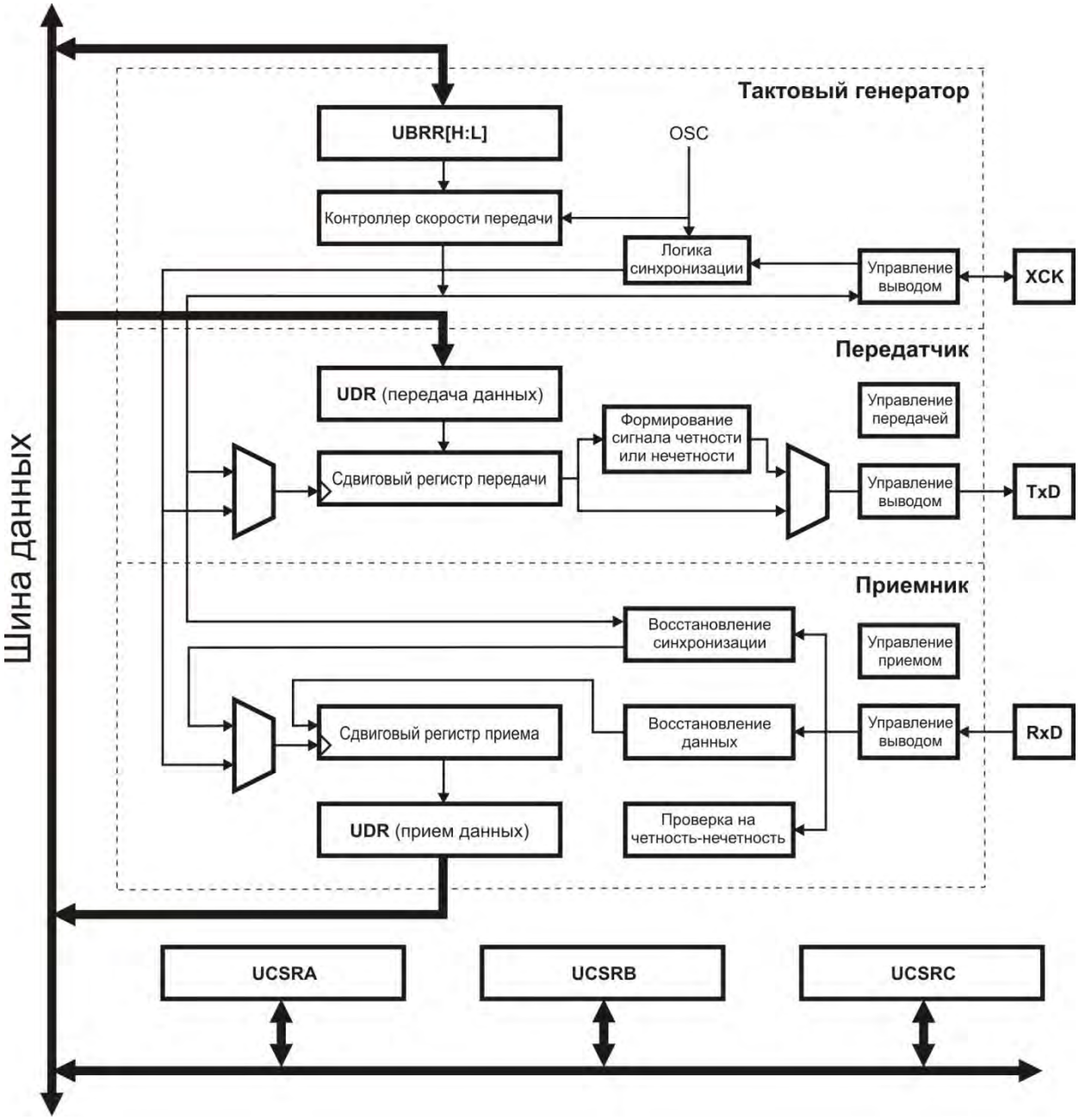


Рисунок 3.80 – Структурная схема USART

На рисунке 3.81 показана структурная схема логики генерации тактовых импульсов.

Описание сигналов:

txclk – синхронизация передатчика (внутренний сигнал).

rxclk – основная синхронизация приемника (внутренний сигнал).

xcki – вход XCK (внутренний сигнал). Используется для синхронной работы в ведомом режиме.

xcko – выход XCK (внутренний сигнал). Используется для синхронной работы в ведущем режиме.

fosc – частота вывода BQ (системная синхронизация).

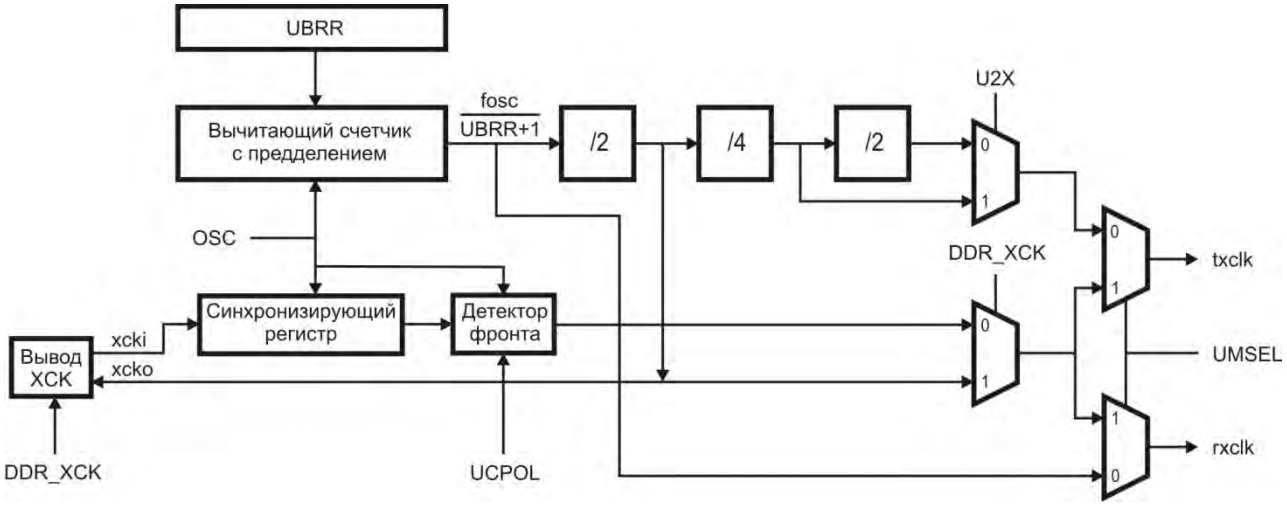


Рисунок 3.81 – Структурная схема логики генерации тактовых импульсов

Внутренняя генерация тактовых импульсов – контроллер скорости передачи

Внутренняя генерация тактовых импульсов используется для асинхронного и синхронного ведущего режима работы. Содержимое данного подпункта относится к рисунку 3.81.

Регистр скорости передачи USART (UBRR) и связанный с ним вычитающий счетчик функционируют как программируемый предделитель или контроллер скорости передачи. В вычитающий счетчик, тактируемый системной синхронизацией (fosc), загружается значение из регистра UBRR всякий раз, когда счетчик достигает нуля или когда производится запись в регистр UBRRL. Тактовый сигнал генерируется каждый раз, когда счетчик достигает нулевого значения. Данный тактовый сигнал является выходом синхронизации контроллера скорости передачи (fosc/(UBRR+1)). Передатчик делит частоту контроллера скорости передачи на 2, 8 или 16 в зависимости от режима работы. Выход контроллера скорости передачи используется непосредственно блоками восстановления синхронизации и данных, входящими в состав приемника. Однако блоки восстановления используют конечный автомат с 2, 8 или 16 состояниями в зависимости от режима, устанавливаемого битами UMSEL, U2X и DDR_XCK.

В таблице 3.74 содержатся уравнения для вычисления скорости передачи (в битах в секунду) и значения UBRR для каждого режима работы при использовании внутренне генерируемого тактового источника.

Таблица 3.74 – Вычисление установочных параметров для регистра скорости передачи

Режим работы	Формула для вычисления скорости передачи	Формула для вычисления значения UBRR
Асинхронный нормальный 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 ведущий режим	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$
Примечание – Скорость передачи представлена в битах в секунду (бит/с).		

Описание параметров:

- BAUD – скорость передачи (в битах в секунду, бит/с);
- fosc – тактовая частота системного генератора;

- UBRR – Содержимое регистров UBRRH и UBRRL, (0–4095).

Примеры значений UBRR для некоторых частот системной синхронизации представлены в таблице 3.82.

Работа с удвоением скорости (U2X)

Скорость передачи данных может быть удвоена, если установить в единицу бит U2X в регистре UCSRA. Установка данного бита оказывает эффект только в асинхронном режиме. При использовании синхронного режима необходимо сбрасывать бит U2X в ноль.

Установка данного бита приводит к уменьшению коэффициента деления для делителя скорости передачи с 16 до 8, фактически удваивая скорость асинхронной связи. Однако следует обратить внимание, что в этом случае приемник сокращает количество выборок с 16 до 8 для операций восстановления данных и синхронизации, поэтому при использовании данного режима необходимо применять более точные установки скорости передачи и более стабильный тактовый источник. Для передатчика удвоение скорости не связано с какими-либо ограничениями.

Внешняя синхронизация

Внешняя синхронизация используется в синхронном ведомом режиме работы. Содержимое данного подпункта относится к рисунку 3.81.

Для минимизации риска возникновения метастабильности на входе внешнего тактового генератора от вывода XCK производится выборка при помощи синхронизирующего регистра. Сигнал с выхода синхронизирующего регистра затем должен пройти через детектор фронта и только после этого может использоваться передатчиком и приемником. На данный процесс затрачивается два такта синхронизации ЦПУ, и поэтому максимальная частота внешней синхронизации на выводе XCK ограничивается следующим выражением:

$$f_{XCK} < \frac{f_{OSC}}{4}.$$

Следует обратить внимание, что частота f_{OSC} зависит от стабильности источника системной синхронизации. В связи с этим рекомендуется учесть некоторый запас для предотвращения возможной потери данных из-за колебаний частоты.

Синхронная работа

Когда выбран режим синхронной связи (UMSEL = 1), вывод XCK используется либо как вход синхронизации (режим ведомого), либо как выход синхронизации (режим ведущего). Зависимость между тактовыми фронтами и выборкой или изменением данных одна и та же. Принцип работы заключается в том, что выборка вводимых данных (на RxD) осуществляется на фронте XCK, противоположном фронту, по которому происходит изменение выходных данных (на TxD).

Бит UCPOL регистра UCSRC выбирает, какой фронт сигнала XCK используется для выборки данных, а какой для изменения данных. Как показано на рисунке 3.82, при UCPOL равном нулю изменение данных происходит по нарастающему фронту XCK, а выборка по спадающему фронту XCK. Когда UCPOL = 1, то изменение данных происходит по спадающему, а выборка по нарастающему фронту сигнала XCK.

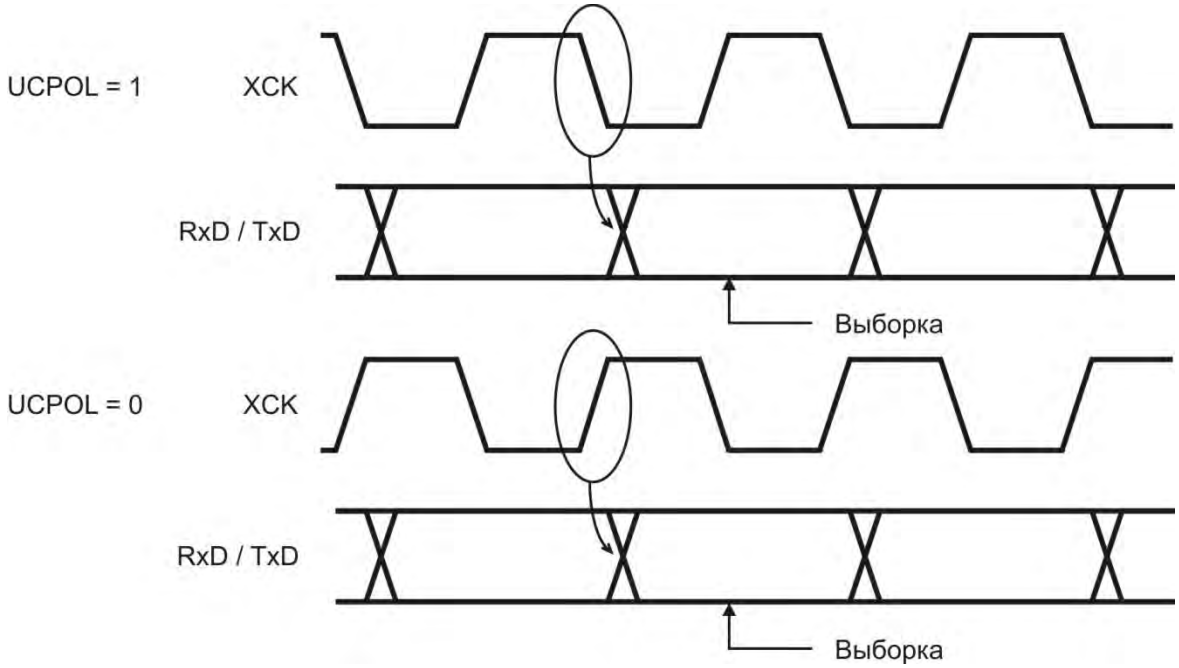


Рисунок 3.82 – Временная диаграмма для синхронного режима работы

3.15.3 Форматы кадров

Последовательный кадр представляет собой кодовую комбинацию из информационных битов, битов синхронизации (стартовые и стоповые биты), а также дополнительного бита контроля четности-нечетности для поиска ошибок. USART поддерживает все 30 комбинаций следующих битов как допустимые форматы кадров:

- 1 стартовый бит;
- 5, 6, 7, 8 или 9 информационных битов;
- без контроля, с битом контроля четности, с битом контроля нечетности;
- 1 или 2 стоповых бита.

Кадр начинается со стартового бита, за которым следует младший значащий информационный бит. Затем следует передача остальных информационных битов (максимальное число которых равно 9) и заканчивается старшим значащим битом. Если разрешена функция контроля четности-нечетности, то сразу после информационных битов передается бит контроля, а затем стоповые биты. После завершения передачи кадра можно начать передавать следующий кадр или же можно перевести линию связи в состояние ожидания (высокий уровень). На рисунке 3.83 иллюстрируются возможные комбинации форматов кадров. Биты в квадратных скобках являются необязательными.

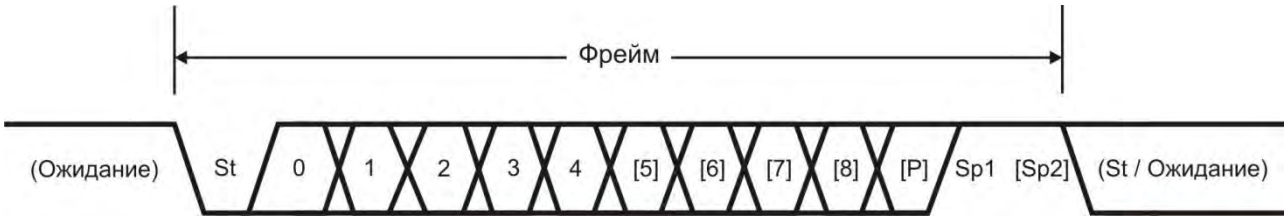


Рисунок 3.83 – Форматы кадров

St – стартовый бит, всегда имеет низкий уровень.
0–8 – информационные биты.
P – бит контроля четности-нечетности.
Sp1, Sp2 – стоповые биты, всегда имеют высокий уровень.

Ожидание – состояние ожидания, в котором приостановлена передача на RxD или TxD. В состоянии ожидания на линии должен быть высокий уровень.

Формат кадра, используемый приемопередатчиком USART, задается битами UCSZ2:0, UPM1:0 и USBS в регистрах UCSRB и UCSRC. Приемник и передатчик используют одни и те же установки. Обратите внимание, что изменение установок любого из этих битов приведет к повреждению текущего сеанса связи как для приемника, так и для передатчика.

Биты размера кодовой комбинации USART (UCSZ2:0) определяют число информационных битов в кадре. Биты режима контроля четности-нечетности USART (UPM1–0) разрешают и устанавливают тип бита контроля. Выбор количества стоповых битов осуществляется битом USBS. Приемник игнорирует второй стоповый бит. Поэтому ошибка кадра будет детектирована только в случаях, когда первый стоповый бит равен нулю.

Вычисление бита контроля четности-нечетности

Бит контроля четности-нечетности вычисляется путем выполнения логической операции исключающего ИЛИ над всеми информационными битами. Если используется контроль по нечетности, то результат этой операции инвертируется. Зависимость между битом контроля и информационными битами следующая:

$$P_{ЧЕТН} = d_{n-1} \oplus ... \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0,$$

$$P_{НЕЧЕТН} = d_{n-1} \oplus ... \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1,$$

где $P_{ЧЕТН}$ – бит контроля по четности;

$P_{НЕЧЕТН}$ – бит контроля по нечетности;

d_n – n-ый информационный бит в кодовой комбинации.

Если используется бит контроля четности-нечетности, то он передается между последним информационным битом и первым стоповым битом в последовательном кадре.

3.15.4 Инициализация USART

Перед началом сеанса связи необходимо выполнить инициализацию USART. Процесс инициализации обычно состоит из настройки скорости передачи, задания формата кадра и разрешения работы передатчика или приемника в зависимости от особенностей применения. При работе USART, управляемой прерываниями, следует сбрасывать флаг общего разрешения прерываний (т.е. необходимо запретить все прерывания), когда выполняется инициализация.

Перед выполнением повторной инициализации с измененными настройками скорости передачи или формата кадра, убедитесь, что в период изменения значений регистров не происходит передача данных. Флаг TXC может использоваться для проверки завершения работы передатчика, а флаг RXC – для проверки отсутствия непрочитанных данных в приемном буфере. Обратите внимание, что в этом случае флаг TXC должен сбрасываться программно перед началом каждой передачи (перед записью в UDR).

В следующих примерах показаны функции для простой инициализации USART на Ассемблере и Си. Примеры подразумевают асинхронную работу с использованием опроса (прерывания не разрешены) и фиксированного формата кадра. Скорость передачи выступает как параметр функции. В примере на Ассемблере предполагается, что параметр скорости передачи сохраняется в регистрах r17:r16.

Пример кода на Ассемблере:

```
USART_Init:
; Настройка скорости передачи
out UBRRH, r17
```

```
out UBRRL, r16
; Разрешение работы приемника и передатчика
ldi r16, (1<<RXEN)|(1<<TXEN)
out UCSRB,r16
; Установка формата фрейма: 8 информационных битов, 2 стоповых бита
ldi r16, (1<<USBS)|(3<<UCSZ0)
out UCSRC,r16
ret
```

Пример кода на Си:

```
#define FOSC 1843200// Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
...
USART_Init ( MYUBRR );
...
}
void USART_Init( unsigned int ubrr )
{
/* Настройка скорости передачи */
UBRRH = (unsigned char)(ubrr>>8);
UBRRL = (unsigned char)ubrr;
/* Разрешение работы приемника и передатчика */
UCSRB = (1<<RXEN)|(1<<TXEN);
/* Установка формата фрейма: 8 информационных битов, 2 стоповых бита */
UCSRC = (1<<USBS)|(3<<UCSZ0);
}
```

Примечание – При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

Более расширенные процедуры инициализации могут использовать интерфейс, где в качестве параметров выступают, например, формат кадра, отключение прерываний и т.д. Однако в большинстве прикладных программ используются фиксированные установки скорости передачи и управляющих регистров, поэтому для таких приложений код инициализации может быть непосредственно включен в основную программу или же может быть объединен с кодом инициализации для других модулей ввода-вывода.

3.15.5 Передача данных. Передатчик USART

Работа передатчика USART разрешается путем установки бита разрешения передачи (TXEN) в регистре UCSRB. Когда передатчик включен, функция нормальной работы вывода TxD заменяется альтернативной функцией последовательного выхода передатчика. Скорость передачи, режим работы и формат кадра должны быть однократно установлены перед началом любой передачи. Если используется синхронная работа, то функция вывода XCK также заменяется альтернативной функцией синхронизации передачи.

Передача кадров с количеством информационных битов от 5 до 8

Передача информации инициируется загрузкой данных, которые необходимо передать, в буфер передатчика. ЦПУ может загрузить буфер передатчика путем записи в ячейку ввода-вывода регистра UDR. Буферизованные данные из буфера передатчика будут перемещены в сдвиговый регистр, когда последний будет готов к отправке нового кадра. Запись новых данных в сдвиговый регистр происходит, если он находится в состоянии ожидания

(когда не выполняется передача данных) или сразу после завершения передачи последнего стопового бита предыдущего кадра. Когда в сдвиговый регистр загружены новые данные, начинается передача одного полного кадра на скорости, установленной регистром скорости передачи, битом U2X или сигналом XCK в зависимости от выбранного режима работы.

В следующих примерах представлена простая функция передачи через USART, основанная на опросе флага освобождения регистра данных (UDRE). При использовании кадров с числом информационных бит менее восьми старшие значащие биты, записанные в UDR, игнорируются. Перед вызовом данной функции должна быть выполнена инициализация USART. В программе на Ассемблере предполагается, что передаваемые данные сохраняются в регистре R16.

Пример кода на Ассемблере:

```
USART_Transmit:
; Ожидание освобождения буфера передатчика
sbis UCSRA,UDRE
rjmp USART_Transmit
; Помещение данных (r16) в буфер, пересылка данных
out UDR,r16
ret
```

Пример кода на Си:

```
void USART_Transmit( unsigned char data )
{
/* Ожидание освобождения буфера передатчика */
while ( !( UCSRA & (1<<UDRE)) )
;
/* Помещение данных в буфер, пересылка данных */
UDR = data;
}
```

Примечание – При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

Данная функция просто ожидает освобождения буфера передатчика, проверяя флаг UDRE перед загрузкой новых данных для передачи. Если используется прерывание по освобождению регистра данных, то программа обработки прерываний записывает данные в буфер.

Передача кадров с 9 информационными битами

Если необходимо передать 9 информационных битов (UCSZ = 7), то девятый бит данных должен быть записан в бит TXB8 регистра UCSRB, перед тем как младший байт кодовой комбинации будет записан в UDR. В следующих примерах показана функция для передачи 9 информационных битов. В программе на Ассемблере предполагается, что передаваемые данные сохраняются в регистрах R17 – R16.

Пример кода на Ассемблере:

```
USART_Transmit:
; Ожидание освобождения буфера передатчика
sbis UCSRA,UDRE
rjmp USART_Transmit
; Копирование девятого бита из r17 в TXB8
cbi UCSRB,TXB8
```

```
sbrc r17,0
sbi UCSRB,TXB8
; Помещение младшего байта данных (r16) в буфер, пересылка данных
out UDR,r16
ret
```

Пример кода на Си:

```
void USART_Transmit( unsigned int data )
{
/* Ожидание освобождения буфера передатчика */
while ( !( UCSRA & (1<<UDRE)) )
;
/* Копирование девятого бита в TXB8 */
UCSRB &= ~(1<<TXB8);
if ( data & 0x0100 )
UCSRB |= (1<<TXB8);
/* Помещение данных в буфер, пересылка данных */
UDR = data;
}
```

Примечания

1 Данная функция передачи записана как функция общего назначения. Ее можно оптимизировать, если содержимое регистра UCSRB будет статичным. То есть, когда после инициализации будет использоваться только бит TXB8 регистра UCSRB.

2 При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

Девятый бит можно использовать для обозначения адресного кадра, когда применяется многопроцессорный режим связи, или для других протоколов.

Флаги передатчика и прерывания

Передатчик USART имеет два флага, которые обозначают его состояние: флаг освобождения регистра данных USART (UDRE) и флаг завершения передачи (TXC). Оба флага могут использоваться для генерации прерываний.

Флаг освобождения регистра данных (UDRE) указывает на готовность буфера передатчика принять новые данные. Этот бит устанавливается при освобождении буфера передатчика и сбрасывается, когда буфер передатчика содержит данные для передачи, которые еще не были перемещены в сдвиговый регистр. При выполнении записи в регистр UCSRA рекомендуется записывать ноль в бит UDRE.

Если бит разрешения прерывания по освобождению регистра данных (UDRIE) в регистре UCSRB равен логической единице, то данное прерывание будет выполняться, пока будет установлен бит UDRE (при условии, что включено общее разрешение прерываний). Флаг UDRE сбрасывается при осуществлении записи в регистр UDR. Если используется передача данных с управлением по прерываниям, то программа обработки прерываний по освобождению регистра данных должна либо записывать новые данные в регистр UDR для сброса бита UDRE, либо отключать 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, когда целый кадр в сдвиговом регистре передатчика полностью сдвинут и в буфере передатчика отсутствуют новые данные для передачи. Флаг TXC сбрасывается автоматически при переходе на вектор обработки прерывания по завершению передачи, или же этот бит можно сбросить программно путем записи в него логической единицы. Флаг TXC полезно использовать при

организации полудуплексной связи (например, стандарт RS485), где прикладная программа для передачи данных должна переходить в режим приема и освобождать шину связи сразу после завершения передачи.

Если в регистре UCSRB установлен бит разрешения прерывания по завершению передачи (TXCIE), то данное прерывание будет выполнено, когда установится флаг TXC (при условии, что включено общее разрешение прерываний). При использовании прерывания по завершению передачи нет необходимости сбрасывать флаг TXC в программе обработки прерываний, поскольку это происходит автоматически при переходе на вектор обработки прерывания.

Генератор контроля четности-нечетности

Генератор контроля четности-нечетности вычисляет бит контроля для информации в последовательном кадре. Когда разрешено использование бита контроля четности-нечетности (UPM1 = 1), управляющая логика передатчика вставляет данный бит между последним информационным битом и первым стоповым битом в отправляемом фрейме.

Отключение передатчика

Отключение передатчика после сброса бита TXEN произойдет только тогда, когда завершатся текущая и ожидаемая передачи, то есть когда в сдвиговом регистре передатчика и буфере передатчика не будет данных для передачи. После отключения передатчика отменяется альтернативное назначение вывода TxD.

3.15.6 Прием данных. Приемник USART

Работа приемника USART разрешается путем установки бита разрешения приема (RXEN) в регистре UCSRB. Когда приемник включен, функция нормальной работы вывода RxD заменяется альтернативной функцией последовательного входа приемника. Скорость передачи, режим работы и формат кадра должны быть однократно установлены перед началом любого приема. Если используется синхронная работа, то функция вывода XCK также заменяется альтернативной функцией синхронизации передачи.

Прием кадров с количеством информационных битов от 5 до 8

Приемник начинает прием данных после обнаружения действительного стартового бита. Выборка каждого последующего информационного бита происходит с частотой скорости передачи или частотой сигнала XCK. Затем каждый информационный бит помещается в сдвиговый регистр приемника, пока не будет получен первый стоповый бит кадра. Приемник игнорирует второй стоповый бит. После получения первого стопового бита, то есть когда целый последовательный кадр будет находиться в сдвиговом регистре приемника, содержимое сдвигового регистра переместится в приемный буфер. Приемный буфер считывается при чтении регистра UDR.

В следующих примерах представлена простая функция приема данных, основанная на опросе флага завершения приема (RXC). При использовании кадров с числом информационных битов менее восьми старшие значащие биты, считанные из UDR, маскируются нулем. Перед вызовом данной функции должна быть выполнена инициализация USART.

Пример кода на Ассемблере:

```
USART_Receive:
; Ожидание данных для приема
sbis UCSRA, RXC
rjmp USART_Receive
; Получение и возврат полученных данных из буфера
in r16, UDR
ret
```

Пример кода на Си:

```
unsigned char USART_Receive( void )
{
/* Ожидание данных для приема */
while ( !(UCSRA & (1<<RXC)) )
;
/* Получение и возврат полученных данных из буфера */
return UDR;
}
```

Примечание – При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

Представленная функция ожидает появления данных в приемном буфере, проверяя флаг RXC перед считыванием буфера и возвратом значения.

Прием кадров с 9 информационными битами

Если необходимо выполнить прием 9 информационных битов (UCSZ = 7), то девятый бит данных должен быть считан из бита RXB8 регистра UCSRB перед считыванием младшего байта из UDR. Это правило также относится к флагам статуса FE, DOR и UPE: сначала опрашивается состояние UCSRA, а только затем считываются данные из UDR. Считывание регистра UDR приводит к изменению состояния приемного буфера FIFO, и, следовательно, биты TXB8, FE, DOR и UPE, хранящиеся в буфере FIFO, тоже изменятся.

В следующих примерах представлена простая функция приема данных, которая управляет и кодовыми комбинациями с 9 информационными битами, и битами состояния.

Пример кода на Ассемблере:

```
USART_Receive:
; Ожидание данных для приема
sbis UCSRA, RXC
rjmp USART_Receive
; Опрос состояния и получение девятого бита, а затем данных из буфера
in r18, UCSRA
in r17, UCSRB
in r16, UDR
; В случае ошибки возвращается -1
andi r18,(1<<FE)|(1<<DOR)|(1<<UPE)
breq USART_ReceiveNoError
ldi r17, HIGH(-1)
ldi r16, LOW(-1)
USART_ReceiveNoError:
; Фильтрация девятого бита, затем возврат
lsr r17
andi r17, 0x01
ret
```

Пример кода на Си:

```
unsigned int USART_Receive( void )
{
unsigned char status, resh, resl;
/* Ожидание данных для приема */
while ( !(UCSRA & (1<<RXC)) )
;
/* Опрос состояния и получение девятого бита, а затем данных из буфера */
status = UCSRA;
```

```
resh = UCSRB;
resl = UDR;
/* В случае ошибки возвращается -1 */
if ( status & (1<<FE)|(1<<DOR)|(1<<UPE) )
return -1;
/* Фильтрация девятого бита, затем возврат */
resh = (resh >> 1) & 0x01;
return ((resh << 8) | resl);
}
```

Примечание – При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

Представленная функция считывает значения всех регистров ввода-вывода в регистровый файл перед выполнением каких-либо вычислений. Это позволяет оптимально использовать приемный буфер, поскольку считанная буферная ячейка будет освобождаться для приема новых данных как можно раньше.

Флаг завершения приема и прерывание

Приемник USART имеет один флаг, который обозначает состояние приемника.

Флаг завершения приема (RXC) указывает на наличие несчитанных данных в приемном буфере. Этот флаг равен логической единице, когда в буфере имеются несчитанные данные. Когда приемный буфер пуст (то есть, когда он не содержит каких-либо несчитанных данных), значение флага RXC равно логическому нулю. При выключении приемника (RXEN = 0) приемный буфер очищается, и, следовательно, бит RXC принимает нулевое значение.

Если бит разрешения прерывания по завершению приема (RXCIE) в регистре UCSRB равен логической единице, то данное прерывание будет выполняться, пока будет установлен флаг RXC (при условии, что включено общее разрешение прерываний). Если используется прием данных с управлением по прерываниям, то программа обработки прерываний по завершению приема должна будет считывать полученные данные из UDR для сброса флага RXC, иначе при выходе из процедуры обработки прерывания сразу возникнет новое прерывание.

Флаги ошибок приемника

Приемник USART имеет три флага ошибок: флаг ошибки кадра (FE), флаг переполнения данных (DOR) и флаг ошибки контроля на четность-нечетность (UPE). Получить доступ ко всем этим флагам можно при чтении регистра UCSRA. Общим свойством данных флагов является то, что они располагаются в приемном буфере вместе с кадром, для которого они отображают состояние ошибки. Из-за буферизации флагов ошибок необходимо считывать сначала регистр UCSRA, а только затем данные из приемного буфера UDR, поскольку чтение регистра UDR приводит к изменению состояния считанной буферной ячейки. Другим сходством флагов ошибок является то, что они не могут быть изменены программно путем записи в соответствующий бит. Однако все же рекомендуется устанавливать нулевые значения для данных флагов при выполнении записи в регистр UCSRA. Ни один флаг ошибки не может генерировать прерывание.

Флаг ошибки кадра (FE) показывает состояние первого стопового бита для следующего читаемого кадра, хранящегося в приемном буфере. Флаг FE равен нулю, когда стоповый бит считывается корректно (как логическая единица). Флаг FE равен единице, когда стоповый бит некорректен (то есть равен нулю). Данный флаг может использоваться для выявления состояний рассинхронизации, обрыва связи и для управления протоколами. Установки бита USBS в регистре UCSRC не оказывают влияния на флаг FE, поскольку

приемник игнорирует все стоповые биты за исключением первого. При выполнении записи в регистр UCSRA рекомендуется устанавливать нулевое значение для бита FE.

Флаг переполнения данных (DOR) указывает на потерю данных из-за переполнения приемного буфера. Переполнение данных возникает, когда приемный буфер заполнен (то есть содержит два кадра), в сдвиговом регистре приемника ожидает считывания новый кадр и обнаружен новый стартовый бит. Если флаг DOR установлен, то это значит, что один последовательный кадр (или более) был потерян между кадром, считанным в последний раз из UDR, и следующим кадром, считываемым из UDR. При выполнении записи в регистр UCSRA рекомендуется устанавливать нулевое значение для бита DOR. Флаг DOR сбрасывается, когда принятый кадр был успешно перемещен из сдвигового регистра в приемный буфер.

Флаг ошибки контроля на четность-нечетность (UPE) показывает, что во время приема кадра была обнаружена ошибка контроля по четности-нечетности. Если контроль на четность-нечетность отключен, то флаг UPE всегда будет считываться как ноль. При выполнении записи в регистр UCSRA рекомендуется устанавливать нулевое значение для бита UPE. Для получения подробностей см. выше пункт 3.15.3 «Форматы кадров», а также следующий подпункт.

Устройство контроля четности-нечетности

Устройство контроля четности-нечетности активно, когда установлен старший бит режима контроля USART (UPM1). Тип контроля – по четности или по нечетности – задается битом UPM0. После активизации данное устройство вычисляет четность-нечетность информационных битов в принятом кадре и сравнивает полученное значение с битом контроля из того же последовательного кадра. Результат сравнения сохраняется в приемном буфере вместе с принятыми информационными и стоповыми битами. Флаг ошибки контроля на четность-нечетность (UPE) затем может быть считан программно для проверки, имела ли место ошибка контроля или нет.

Бит UPE устанавливается, если очередной кадр, который может быть считан из приемного буфера, имел ошибку контроля по четности-нечетности при приеме, и устройство контроля по четности-нечетности было включено (UPM1 = 1). Этот бит будет действителен до тех пор, пока не произойдет считывание приемного буфера (UDR).

Отключение приемника

В отличие от передатчика, отключение приемника происходит незамедлительно. Поэтому информация текущей операции приема будет потеряна. После отключения приемника (то есть когда RXEN = 0) отменяется альтернативное назначение вывода RxD и очищается приемный буфер FIFO. По этой причине данные, оставшиеся в буфере, будут потеряны.

Очистка приемного буфера

Приемный буфер FIFO сбрасывается после отключения приемника, то есть полностью освобождается от своего содержимого. Несчитанные данные будут потеряны. Если необходимо очистить буфер в процессе нормальной работы, например, из-за возникшей ошибки, то следует выполнить считывание содержимого регистра UDR, прежде чем сбросится флаг RXC. В следующем примере показано, как очистить приемный буфер.

Пример кода на Ассемблере:

```
USART_Flush:
sbis UCSRA, RXC
ret
in r16, UDR
rjmp USART_Flush
```

Пример кода на Си:

```
void USART_Flush( void )
{
unsigned char dummy;
while ( UCSRA & (1<<RXC) ) dummy = UDR;
}
```

Примечание – При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

Приемопередатчик USART содержит блок восстановления синхронизации и блок восстановления данных для управления асинхронным приемом данных. Логика восстановления синхронизации применяется для согласования внутренне генерируемой скорости передачи с поступающими асинхронными последовательными фреймами на выводе RxD. Логика восстановления данных осуществляет выборку и фильтрацию (ФНЧ) каждого входящего бита, повышая тем самым помехоустойчивость приемника. Рабочий диапазон асинхронного приема зависит от точности встроенного контроллера скорости передачи, скорости поступления кадров и размера кадров (в битах).

Асинхронное восстановление синхронизации

Логика восстановления синхронизации применяется для согласования работы внутреннего тактового генератора с поступающими последовательными кадрами. На рисунке 3.84 иллюстрируется процесс выборки стартового бита в поступающем кадре. Скорость выборки в 16 раз выше скорости передачи для нормального режима и в восемь раз выше для режима удвоения скорости. Горизонтальные стрелки иллюстрируют отклонение синхронизации в процессе выборки. Обратите внимание, что большее по времени отклонение возникает при использовании режима удвоения скорости (U2X = 1). Выборки, обозначенные номером 0, соответствуют состоянию ожидания на линии RxD (то есть, когда нет активного сеанса связи).

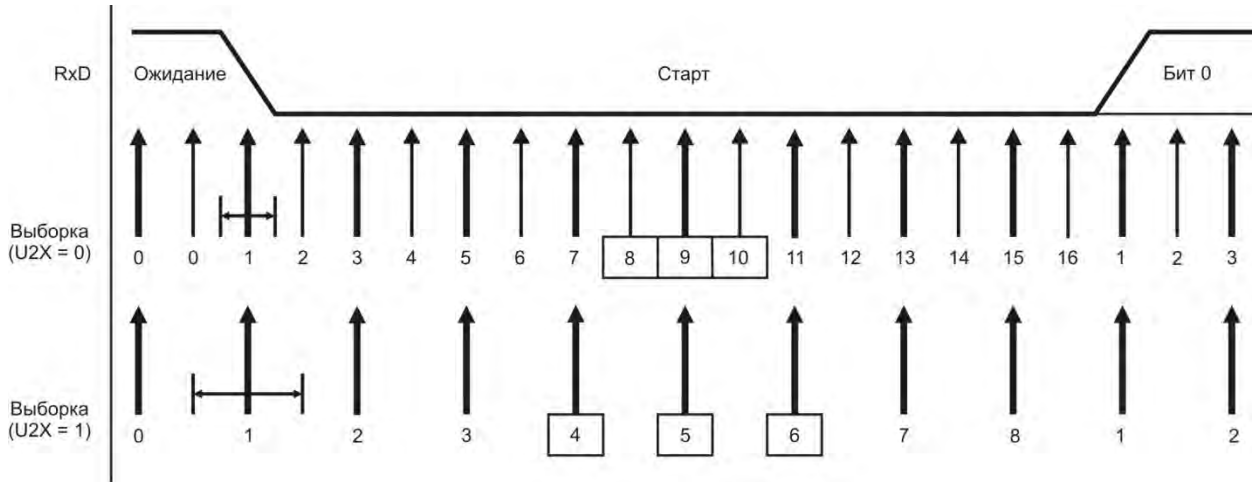


Рисунок 3.84 – Выборка стартового бита

Когда логика восстановления синхронизации детектирует переход из высокого состояния (ожидание) к низкому состоянию (старт) на линии RxD, инициируется последовательность действий по обнаружению стартового бита. Допустим, что выборка 1 обозначает первую выборку с нулевым значением, как показано на рисунке 3.84. Тогда логика восста-

новления синхронизации использует выборки 8, 9 и 10 в нормальном режиме и выборки 4, 5 и 6 в режиме удвоения скорости (на рисунке номера этих выборок помещены в рамку), чтобы решить, получен ли действительный стартовый бит. Если две или все три выборки имеют высокий логический уровень (принцип мажоритарного голосования), то стартовый бит отклоняется как импульсная помеха, а приемник начинает поиск следующего перехода сигнала от высокого уровня к низкому. Однако если обнаруживается действительный стартовый бит, то логика восстановления синхронизации оказывается согласованной по времени, и тогда может начинаться восстановление данных. Процесс синхронизации повторяется для каждого стартового бита.

Асинхронное восстановление данных

Когда синхронизация приемника согласована со стартовым битом, может начинаться восстановление данных. Блок восстановления данных использует автомат конечных состояний, который имеет 16 состояний для каждого бита в нормальном режиме работы и восемь состояний в режиме удвоения скорости. На рисунке 3.85 показана выборка информационных битов и бита контроля четности-нечетности. Для каждой выборки указан номер, который соответствует состоянию блока восстановления данных.

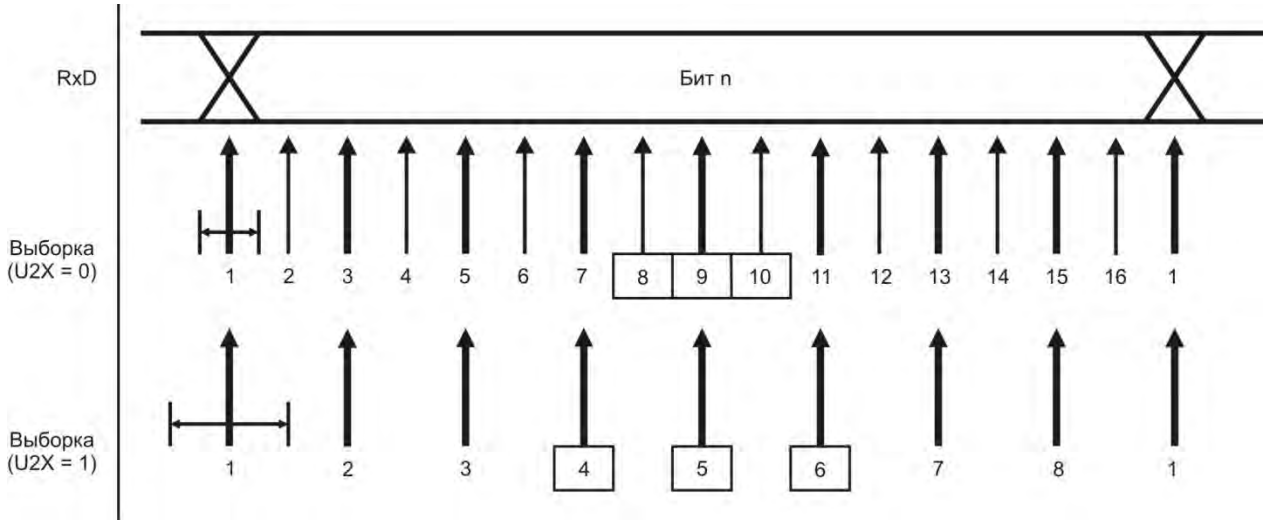


Рисунок 3.85 – Выборка информационных битов и бита контроля четности-нечетности

Выбор логического уровня принятого бита осуществляется с помощью мажоритарного голосования по трем выборкам, расположенным в центре полученного бита. На рисунке номера центральных выборок помещены в рамку. Процесс мажоритарного голосования заключается в следующем: если две или все три выборки имеют высокий уровень, то принятый бит фиксируется как логическая единица. Если две или все три выборки имеют низкий уровень, то принятый бит фиксируется как логический ноль. Процесс мажоритарного голосования, по сути, представляет собой фильтр нижних частот для сигнала, поступающего на вывод RxD. Далее процесс восстановления повторяется до тех пор, пока не будет принят целый кадр, включая первый стоповый бит. Обратите внимание, что приемник использует только первый стоповый бит кадра. На рисунке 3.86 показан процесс выборки стопового бита и начало стартового бита следующего кадра.

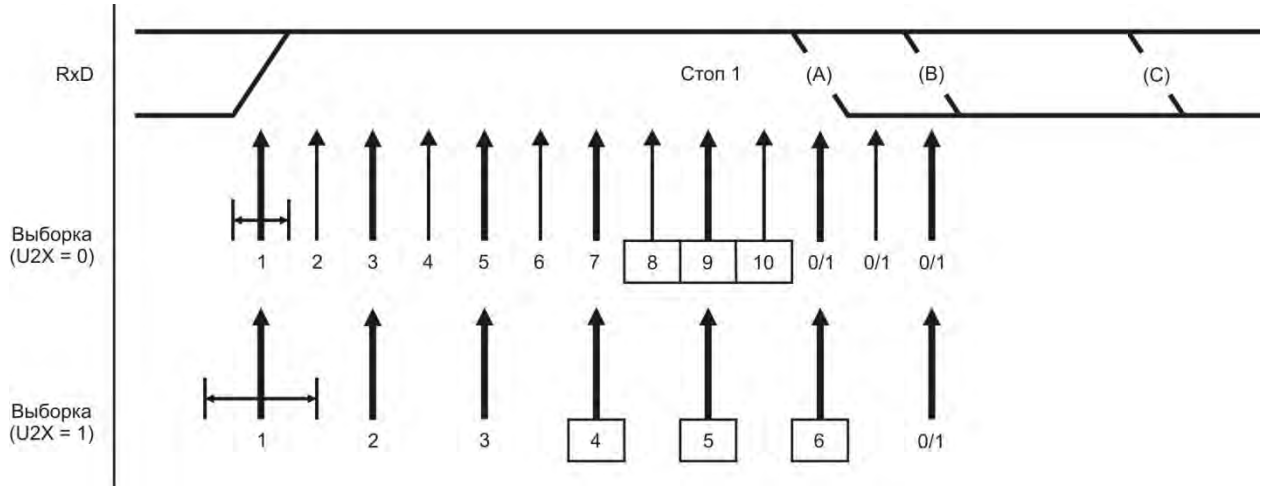


Рисунок 3.86 – Выборка стопового бита и следующего стартового бита

Тот же самый принцип мажоритарного голосования, который используется для других битов кадра, применяется и по отношению к стоповому биту. Когда значение стопового бита фиксируется как логический ноль, устанавливается флаг ошибки кадра (FE).

Новый переход сигнала от высокого уровня к низкому, указывающий на стартовый бит следующего кадра, может наступить сразу после появления последнего бита, используемого в процессе мажоритарного голосования. Для режима с нормальной скоростью первая выборка с низким уровнем может произойти в момент, обозначенный (A) на рисунке 3.86. Для режима с удвоением скорости появление первого низкого уровня должно быть отложено до момента (B). Точка (C) показывает полноразмерный стоповый бит. Раннее детектирование стартового бита оказывает влияние на рабочий диапазон приемника.

Рабочий диапазон при асинхронной работе

Рабочий диапазон приемника зависит от несоответствия между скоростью получения битов и внутренне генерируемой скоростью передачи. Если передатчик отправляет кадры на слишком быстрой или слишком медленной скорости, или внутренне генерируемая скорость передачи приемника не имеет подобной опорной частоты (см. таблицу 3.75), то приемник не сможет синхронизировать кадр по отношению к стартовому биту.

Следующие выражения можно использовать для вычисления соотношения скорости поступающих данных и внутренней скорости передачи приемника:

$$R_{медл} = \frac{(D+1)S}{S-1+D\cdot S+S_F},$$

$$R_{быстр} = \frac{(D+2)S}{(D+1)S+S_M},$$

где D – сумма количества передаваемых информационных битов и бита контроля четности-нечетности (D = 5–10);

S – количество выборок в секунду. Для режима с нормальной скоростью S = 16, а для режима с удвоением скорости S = 8;

S_F – Номер первой выборки, используемой для мажоритарного голосования. Для режима с нормальной скоростью S_F = 8, а для режима с удвоением скорости S_F = 4;

S_M – Номер центральной выборки, используемой для мажоритарного голосования. Для режима с нормальной скоростью S_M = 9, а для режима с удвоением скорости S_M = 5;

$R_{медл}$ – отношение наименьшей допустимой скорости поступления данных к скорости передачи приемника;

$R_{быстр}$ – отношение наибольшей допустимой скорости поступления данных к скорости передачи приемника.

В таблицах 3.75 и 3.76 приведен список максимально допустимых погрешностей для скорости передачи приемника. Обратите внимание, что режим нормальной скорости устойчив к более широким изменениям скорости передачи.

Таблица 3.75 – Рекомендуемая максимальная погрешность скорости передачи приемника в режиме нормальной скорости (U2X = 0)

D (сумма информационных битов и бита контроля четности/нечетности)	$R_{медл}$, %	$R_{быстр}$, %	Максимальная суммарная погрешность, %	Рекомендуемая максимальная погрешность приемника, %
5	93,20	106,67	+6,67/-6,8	± 3,0
6	94,12	105,79	+5,79/-5,88	± 2,5
7	94,81	105,11	+5,11/-5,19	± 2,0
8	95,36	104,58	+4,58/-4,54	± 2,0
9	95,81	104,14	+4,14/-4,19	± 1,5
10	96,17	103,78	+3,78/-3,83	± 1,5

Таблица 3.76 – Рекомендуемая максимальная погрешность скорости передачи приемника в режиме удвоения скорости (U2X = 1)

D (сумма информационных битов и бита контроля четности/нечетности)	$R_{медл}$, %	$R_{быстр}$, %	Максимальная суммарная погрешность, %	Рекомендуемая максимальная погрешность приемника, %
5	94,12	105,66	+5,66/-5,88	± 2,5
6	94,92	104,92	+4,92/-5,08	± 2,0
7	95,52	104,35	+4,35/-4,48	± 1,5
8	96,00	103,90	+3,90/-4,00	± 1,5
9	96,39	103,53	+3,53/-3,61	± 1,5
10	96,70	103,23	+3,23/-3,30	± 1,0

Рекомендации относительно погрешности скорости передачи приемника были сделаны исходя из того, что приемник и передатчик разделяют поровну максимальную суммарную погрешность.

Существует два возможных источника для возникновения погрешности скорости передачи приемника. Системная синхронизация приемника (BQ) всегда имеет некоторую нестабильность из-за колебаний напряжения питания и температуры. При использовании кварцевого резонатора для генерации системной синхронизации, как правило, не возникает таких проблем, но при использовании керамических резонаторов частота синхронизации может изменяться более чем на 2 % в зависимости от характеристик выбранного резонатора. Второй источник возникновения погрешности является более управляемым. Контроллер скорости передачи не может всегда выполнять точное деление системной частоты для получения желаемой скорости передачи. В этом случае, если возможно, следует использовать такое значение регистра UBRR, которое дает приемлемую низкую погрешность.

3.15.7 Многопроцессорный режим связи

Установка бита многопроцессорного режима связи (MPCM) в регистре UCSRA активизирует функцию фильтрации входящих кадров, полученных приемником USART. Кадры, которые не содержат адресную информацию, игнорируются и не помещаются в приемный буфер. Это позволяет существенно уменьшить количество поступающих кадров, подлежащих обработке ЦПУ, в многопроцессорных системах, где связь между процессорами организована через одну последовательную шину. Значение бита MPCM не оказывает ни-

какого влияния на работу передатчика, но при этом передатчик следует применять иначе, когда он является частью системы, использующей режим многопроцессорной связи.

Если приемник настроен на прием кадров с количеством информационных битов от 5 до 8, тогда первый стоповый бит обозначает, что именно содержится в кадре: данные или адресная информация. Если приемник настроен на прием кадров с девятью информационными битами, то девятый бит (RXB8) используется для идентификации адресных и информационных кадров. Когда идентификатор типа кадра (первый стоповый бит или девятый информационный бит) равен единице, то это значит, что в кадре содержится адрес. Если идентификатор равен нулю, значит, в кадре содержатся данные.

Многопроцессорный режим связи позволяет нескольким ведомым микроконтроллерам принимать данные от одного ведущего. Это осуществляется путем декодирования вначале адресного фрейма, чтобы выяснить, который из микроконтроллеров адресуется. Если один из ведомых микроконтроллеров обнаруживает свой адрес, то последующие информационные кадры он будет принимать в нормальном режиме, в то время как остальные ведомые микроконтроллеры будут игнорировать принятые кадры до тех пор, пока не будет получен очередной адресный кадр.

Использование MPCM

Если микроконтроллер действует как ведущий, то он может использовать 9-битный формат кадра (UCSZ = 7). Девятый информационный бит (TXB8) должен устанавливаться при передаче адресного кадра (TXB8 = 1) и сбрасываться при передаче информационного кадра (TXB8 = 0). В этом случае ведомые микроконтроллеры также должны быть настроены на использование 9-битного формата кадров.

Для обмена данными в многопроцессорном режиме связи необходимо применять следующую процедуру:

1 Все ведомые микроконтроллеры переводятся в многопроцессорный режим связи (MPCM = 1).

2 Ведущий микроконтроллер отправляет адресный кадр, а все ведомые принимают и считывают этот кадр. В ведомых микроконтроллерах флаг RXC в регистре UCSRA устанавливается как обычно.

3 Каждый ведомый микроконтроллер считывает значение регистра UDR и определяет, адресуется ли ведущий к нему или нет. Если да, то этот ведомый микроконтроллер сбрасывает бит MPCM в регистре UCSRA. Если же данный микроконтроллер не адресован, то он ожидает появления следующего адресного кадра и сохраняет установки бита MPCM.

4 Адресуемый микроконтроллер будет принимать все информационные кадры, пока не будет получен новый адресный кадр. Другие ведомые микроконтроллеры, у которых бит MPCM остался установленным, будут игнорировать эти информационные кадры.

5 После приема адресуемым устройством последнего информационного кадра, этот ведомый микроконтроллер устанавливает бит MPCM и ожидает получения нового адресного кадра от ведущего микроконтроллера. Затем процесс повторяется, начиная с пункта 2.

Использование форматов кадров с количеством информационных битов от 5 до 8 возможно, но непрактично, поскольку в этом случае приемник должен переключаться между форматами кадров с количеством информационных битов n и n+1. Это делает полнодуплексную работу затруднительной, так как передатчик и приемник используют одинаковые установки для размера кодовой комбинации. Если используются кадры с количеством информационных битов от 5 до 8, то передатчик должен быть настроен на использование двух стоповых битов (USBS = 1), поскольку первый стоповый бит применяется для идентификации типа кадра.

Не используйте 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.

3.15.8 Описание регистров USART

Регистр данных USART – UDRn

Бит	7	6	5	4	3	2	1	0	
	RXBn[7:0]								UDRn (Чтение)
	TXBn[7:0]								UDRn (Запись)
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Буферные регистры данных передатчика и приемника USARTn расположены по одному и тому же адресу в области ввода-вывода, обозначенной как регистр данных USARTn или UDRn. Буферный регистр данных передатчика (TXBn) является местом назначения для данных, записываемых в регистр UDRn. Считывание регистра UDRn возвращает содержимое буферного регистра данных приемника (RXBn).

При использовании 5-, 6- или 7-битных форматов кадров передатчик игнорирует старшие неиспользуемые биты, а приемник устанавливает их в ноль.

Запись в буфер передатчика можно выполнять только тогда, когда установлен флаг UDREn в регистре UCSRAn. Данные, записанные в UDRn при сброшенном флаге UDREn, будут проигнорированы передатчиком USARTn. Если была произведена запись в буфер передатчика при включенном передатчике, то передатчик загрузит данные в сдвиговый регистр передатчика, когда сдвиговый регистр освободится. Затем данные будут последовательно переданы на вывод TxDn.

Приемный буфер организован как двухуровневый буфер FIFO. FIFO изменяет свое состояние всякий раз, когда выполняется доступ к приемному буферу. Из-за такого поведения приемного буфера не следует применять инструкции «чтение-модификация-запись» (SBI и CBI). Следует быть осторожными при использовании инструкций тестирования битов (SBIC и SBIS), поскольку их выполнение также изменяет состояние буфера FIFO.

Регистр A управления и статуса USART – UCSRnA

Бит	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Чтение/ Запись	Ч	Ч/З	Ч	Ч	Ч	Ч	Ч/З	Ч/З	
Начальное значение	0	0	1	0	0	0	0	0	

Разряд 7 – RXCn: Флаг завершения приема USART

Этот флаг устанавливается, когда в приемном буфере содержатся несчитанные данные, и сбрасывается, когда приемный буфер свободен (то есть не содержит несчитанных данных). Если приемник отключить, то приемный буфер очистится и, следовательно, бит RXCn примет нулевое значение. Флаг RXCn можно использовать для генерации прерывания по завершению приема (см. описание бита RXCIEn в подпункте ниже).

Разряд 6 – TXCn: Флаг завершения передачи USART

Этот флаг устанавливается, когда целый кадр из сдвигового регистра передатчика полностью передан и в буфере передатчика (UDRn) нет новых данных для передачи. Флаг TXCn автоматически сбрасывается при переходе на вектор прерывания по завершению передачи, или же он может быть сброшен программно путем записи логической единицы в

соответствующий разряд регистра. Флаг TXCn можно использовать для генерации прерывания по завершению передачи (см. описание бита TXCIEn в подпункте ниже).

Разряд 5 – UDREn: Флаг освобождения регистра данных USART

Флаг UDREn указывает на готовность буфера передатчика (UDRn) к приему новых данных. Если UDREn = 1, значит буфер свободен и, следовательно, готов к записи. Флаг UDREn можно использовать для генерации прерывания по освобождению регистра данных (см. описание бита UDRIEn в подпункте ниже). Бит UDREn устанавливается после сброса, показывая, что передатчик готов.

Разряд 4 – FEn: Флаг ошибки кадра

Этот флаг устанавливается, если последующая кодовая комбинация в приемном буфере имела ошибку кадра при получении. То есть, когда первый стоповый бит очередной кодовой комбинации в приемном буфере равен нулю. Значение данного флага действительно до тех пор, пока не произойдет считывание приемного буфера (UDRn). Флаг FEn принимает нулевое значение, когда стоповый бит полученного кадра равен единице. Всегда устанавливайте бит FEn в ноль при выполнении записи в регистр UCSRnA.

Разряд 3 – DORn: Флаг переполнения данных

Этот флаг устанавливается при детектировании состояния переполнения данных. Переполнение данных возникает, когда приемный буфер заполнен (то есть содержит два кадра), в сдвиговом регистре приемника ожидает считывания новый кадр и обнаружен новый стартовый бит. Значение данного флага действительно до тех пор, пока не произойдет считывание приемного буфера (UDRn). Всегда устанавливайте бит DORn в ноль при выполнении записи в регистр UCSRnA.

Разряд 2 – UPEn: Флаг ошибки контроля по четности-нечетности

Этот флаг устанавливается, если очередной кадр, который может быть считан из приемного буфера, имел ошибку контроля по четности-нечетности при получении, и устройство контроля по четности-нечетности было включено в тот же момент (UPMn1 = 1). Значение данного флага действительно до тех пор, пока не произойдет считывание приемного буфера (UDRn). Всегда устанавливайте бит UPEn в ноль при выполнении записи в регистр UCSRnA.

Разряд 1 – U2Xn: Удвоение скорости передачи USART

Бит U2Xn оказывает влияние только при асинхронной работе. Этот бит необходимо устанавливать в ноль, когда используется синхронный режим. Запись логической единицы в данный бит уменьшает значение коэффициента деления для скорости связи с 16 до 8, удваивая тем самым скорость передачи данных в режиме асинхронной связи.

Разряд 0 – MPCMn: Многопроцессорный режим связи

Данный бит разрешает режим многопроцессорной связи. Когда MPCMn = 1, все входящие кадры, полученные приемником USART, будут проигнорированы, если они не содержат адресной информации. Установка бита MPCMn не влияет на работу передатчика. Для получения подробной информации см. 3.15.7 «Многопроцессорный режим связи».

Регистр B управления и статуса USART – UCSRnB

Бит	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – RXCIEn: Разрешение прерывания по завершению приема

Запись логической единицы в данный бит разрешает прерывание по флагу RXCn. Прерывание по завершению приема USART будет сгенерировано только при выполнении следующих условий: RXCIEn = 1, флаг общего разрешения прерываний I в регистре SREG равен единице, а также установлен бит RXCn в регистре UCSRnA.

Разряд 6 – TXCIEn: Разрешение прерывания по завершению передачи

Запись логической единицы в данный бит разрешает прерывание по флагу TXCn. Прерывание по завершению передачи USART будет сгенерировано только при выполнении следующих условий: TXCIEn = 1, флаг общего разрешения прерываний I в регистре SREG равен единице, а также установлен бит TXCn в регистре UCSRnA.

Разряд 5 – UDRIEn: Разрешение прерывания по освобождению регистра данных USART

Запись логической единицы в данный бит разрешает прерывание по флагу UDREn. Прерывание по освобождению регистра данных USART будет сгенерировано только при выполнении следующих условий: UDRIEn = 1, флаг общего разрешения прерываний I в регистре SREG равен единице, а также установлен бит UDREn в регистре UCSRnA.

Разряд 4 – RXENn: Разрешение работы приемника

Запись логической единицы в данный бит разрешает работу приемника USARTn. В этом случае приемник формирует сигнал, который разрешает альтернативную функцию вывода RxDn. Отключение приемника очищает приемный буфер, при этом значения флагов FEn, DORn и UPEn становяться недействительными.

Разряд 3 – TXENn: Разрешение работы передатчика

Запись логической единицы в данный бит разрешает работу передатчика USARTn. В этом случае передатчик формирует сигнал, который разрешает альтернативную функцию вывода TxDn. Отключение передатчика после сброса бита TXENn произойдет только тогда, когда завершатся текущая и ожидаемая передачи, то есть когда в сдвиговом регистре передатчика и буфере передатчика не будет данных для передачи. После отключения передатчика отменяется альтернативное назначение вывода TxDn.

Разряд 2 – UCSZn2: Размер кодовой комбинации

Бит UCSZn2 в сочетании с битами UCSZn1–0 в регистре UCSRnC задает число информационных битов (размер кодовой комбинации) в кадре, как для приемника, так и для передатчика.

Разряд 1 – RXB8n: Информационный бит 8 принимаемых данных

RXB8n является девятым информационным битом принимаемой кодовой комбинации, когда используется 9-битный формат кадров. Этот бит должен быть считан перед тем, как произойдет считывание младших битов из регистра UDRn.

Разряд 0 – TXB8n: Информационный бит 8 передаваемых данных

TXB8n является девятым информационным битом передаваемой кодовой комбинации, когда используется 9-битный формат кадров. Этот бит должен быть записан перед тем, как произойдет запись младших битов в регистр UDRn.

Регистр C управления и статуса USART – UCSRnC

Бит	7	6	5	4	3	2	1	0	
	-	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	1	1	0	

Разряд 7 – Резервный бит

Данный бит является резервным. При выполнении записи в регистр UCSRnC рекомендуется записывать ноль в этот бит.

Разряд 6 – UMSELn: Выбор режима работы USART

Данный бит позволяет переключаться между синхронным и асинхронным режимами последовательной связи.

Таблица 3.77 – Установки бита UMSELn

UMSELn	Режим
0	Асинхронная работа
1	Синхронная работа

Разряды 5–4: UPMn1–0: Режим контроля по четности-нечетности

Эти биты разрешают и устанавливают тип формирования сигнала четности-нечетности и тип контроля. После включения данного режима передатчик автоматически генерирует и передает бит контроля четности-нечетности в каждом фрейме. Приемник формирует значение бита контроля для поступающих данных и сравнивает его с установками битов UPMn1–0. При обнаружении несоответствия будет установлен флаг ошибки контроля на четность-нечетность (UPEn) в регистре UCSRnA.

Таблица 3.78 – Установки битов UPMn1–0

UPMn1	UPMn0	Режим контроля по четности-нечетности
0	0	Отключено
0	1	(Зарезервировано)
1	0	Включено, контроль по четности
1	1	Включено, контроль по нечетности

Разряд 3 – USBSn: Выбор количества стоповых битов

Данный бит определяет, сколько стоповых битов добавляет передатчик при формировании кадра. Приемник игнорирует эту настройку.

Таблица 3.79 – Установки бита USBSn

USBSn	Количество стоповых битов
0	Один бит
1	Два бита

Разряды 2–1: UCSZn1–0: Размер кодовой комбинации

Биты UCSZn1–0 в сочетании с битом UCSZn2 в регистре UCSRnB задают число информационных битов (размер кодовой комбинации) в кадре, как для приемника, так и для передатчика.

Таблица 3.80 – Установки битов UCSZn2–0

UCSZn2	UCSZn1	UCSZn0	Размер кодовой комбинации
0	0	0	5 битов
0	0	1	6 битов
0	1	0	7 битов
0	1	1	8 битов
1	0	0	(Зарезервировано)
1	0	1	(Зарезервировано)
1	1	0	(Зарезервировано)
1	1	1	9 битов

Разряд 0 – UCPOLn: Полярность синхронизации

Бит UCPOLn используется только в синхронном режиме. При асинхронной работе необходимо записывать ноль в данный бит. В синхронном режиме бит UCPOLn определяет зависимость между изменением выходных данных, выборкой входных данных и сигналом тактирования синхронной связи (XCKn).

Таблица 3.81 – Установки бита UCPOLn

UCPOLn	Изменение передаваемых данных на выходе TxDn	Выборка принимаемых данных на входе RxDn
0	Нарастающий фронт сигнала XCKn	Спадающий фронт сигнала XCKn
1	Спадающий фронт сигнала XCKn	Нарастающий фронт сигнала XCKn

Регистры скорости передачи USART – UBRRnL и UBRRnH

Бит	15	14	13	12	11	10	9	8	
	-	-	-	-	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	7	6	5	4	3	2	1	0	
Чтение/запись	Ч	Ч	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	
	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Разряды 15–12: Резервные биты

Данные биты являются резервными. При выполнении записи в регистр UBRRnH рекомендуется записывать ноль в эти биты.

Разряды 11–0: UBRRn11–0: Регистр скорости передачи USART

Это 12-разрядный регистр, который задает значение скорости передачи USART. Регистр UBRRnH содержит старшие четыре бита, а регистр UBRRnL – младшие восемь битов для значения скорости связи USARTn. Если во время передачи или приема изменить скорость передачи, то сеанс связи будет нарушен. Запись в регистр UBRRnL инициирует незамедлительное обновление предделителя скорости передачи.

3.15.9 Примеры установок для скорости передачи

Для стандартных частот кварцевых и керамических резонаторов самые широко применяемые значения скорости передачи при асинхронной работе можно сгенерировать, используя установки регистра UBRR в таблицах 3.82–3.85. Значения погрешностей скорости передачи при генерации вычисляются по формуле

$$Погрешность[\%] = \left(\frac{BAUD_{при\ точном\ совпадении}}{BAUD} - 1 \right) \cdot 100\%,$$

где BAUD – скорость передачи (в битах в секунду, бит/с).

Таблица 3.82 – Примеры установок UBRR для широко применяемых частот генератора

Скорость передачи, бит/с	f_{OSC} = 1,0 МГц				f_{OSC} = 1,8432 МГц				f_{OSC} = 2,0 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %
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	0,2
14,4K	3	8,5	8	-3,5	7	0,0	15	0,0	8	-3,5	16	2,1
19,2K	2	8,5	6	-7,0	5	0,0	11	0,0	6	-7,0	12	0,2
28,8K	1	8,5	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
57,6K	0	8,5	1	8,5	1	0,0	3	0,0	1	8,5	3	8,5
76,8K	-	-	1	-18,6	1	-25,0	2	0,0	1	-18,6	2	8,5
115,2K	-	-	0	8,5	0	0,0	1	0,0	0	8,5	1	8,5
230,4K	-	-	-	-	-	-	0	0,0	-	-	-	-
250K	-	-	-	-	-	-	-	-	-	-	0	0,0
Максимум *	62,5 Кбит/с		125 Кбит/с		115,2 Кбит/с		230,4 Кбит/с		125 Кбит/с		250 Кбит/с	

* UBRR = 0, погрешность равна 0 %.

Таблица 3.83 – Примеры установок UBRR для широко применяемых частот генератора

Скорость передачи, бит/с	f_{OSC} = 3,6864 МГц				f_{OSC} = 4,0 МГц				f_{OSC} = 7,3728 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %
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
14,4K	15	0,0	31	0,0	16	2,1	34	-0,8	31	0,0	63	0,0
19,2K	11	0,0	23	0,0	12	0,2	25	0,2	23	0,0	47	0,0
28,8K	7	0,0	15	0,0	8	-3,5	16	2,1	15	0,0	31	0,0
38,4K	5	0,0	11	0,0	6	-7,0	12	0,2	11	0,0	23	0,0
57,6K	3	0,0	7	0,0	3	8,5	8	-3,5	7	0,0	15	0,0
76,8K	2	0,0	5	0,0	2	8,5	6	-7,0	5	0,0	11	0,0
115,2K	1	0,0	3	0,0	1	8,5	3	8,5	3	0,0	7	0,0
230,4K	0	0,0	1	0,0	0	8,5	1	8,5	1	0,0	3	0,0
250K	0	-7,8	1	-7,8	0	0,0	1	0,0	1	-7,8	3	-7,8
0,5M	-	-	0	-7,8	-	-	0	0,0	0	-7,8	1	-7,8
1M	-	-	-	-	-	-	-	-	-	-	0	-7,8
Макси-мум *	230,4 Кбит/с		460,8 Кбит/с		250 Кбит/с		0,5 Мбит/с		460,8 Кбит/с		921,6 Кбит/с	

* UBRR = 0, погрешность равна 0 %.

Таблица 3.84 – Примеры установок UBRR для широко применяемых частот генератора

Скорость передачи, бит/с	f_{OSC} = 8,0 МГц				f_{OSC} = 11,0592 МГц				f_{OSC} = 14,7456 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %	UBRR	Погрешность, %
2400	207	0,2	416	-0,1	287	0,0	575	0,0	383	0,0	767	0,0
4800	103	0,2	207	0,2	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
14,4K	34	-0,8	68	0,6	47	0,0	95	0,0	63	0,0	127	0,0
19,2K	25	0,2	51	0,2	35	0,0	71	0,0	47	0,0	95	0,0
28,8K	16	2,1	34	-0,8	23	0,0	47	0,0	31	0,0	63	0,0
38,4K	12	0,2	25	0,2	17	0,0	35	0,0	23	0,0	47	0,0
57,6K	8	-3,5	16	2,1	11	0,0	23	0,0	15	0,0	31	0,0
76,8K	6	-7,0	12	0,2	8	0,0	17	0,0	11	0,0	23	0,0
115,2K	3	8,5	8	-3,5	5	0,0	11	0,0	7	0,0	15	0,0
230,4K	1	8,5	3	8,5	2	0,0	5	0,0	3	0,0	7	0,0
250K	1	0,0	3	0,0	2	-7,8	5	-7,8	3	-7,8	6	5,3
0,5M	0	0,0	1	0,0	-	-	2	-7,8	1	-7,8	3	-7,8
1M	-	-	0	0,0	-	-	-	-	0	-7,8	1	-7,8
Макси-мум *	0,5 Мбит/с		1 Мбит/с		691,2 Кбит/с		1,3824 Мбит/с		921,6 Кбит/с		1,8432 Мбит/с	

_____

* UBRR = 0, погрешность равна 0 %.

Таблица 3.85 – Примеры установок UBRR для широко применяемых частот генератора

Скорость передачи, бит/с	f_{OSC} = 16,0 МГц			
	U2X = 0		U2X = 1	
	UBRR	Погрешность, %	UBRR	Погрешность, %
2400	416	-0,1	832	0,0
4800	207	0,2	416	-0,1
9600	103	0,2	207	0,2
14,4K	68	0,6	138	-0,1
19,2K	51	0,2	103	0,2
28,8K	34	-0,8	68	0,6
38,4K	25	0,2	51	0,2
57,6K	16	2,1	34	-0,8
76,8K	12	0,2	25	0,2
115,2K	8	-3,5	16	2,1
230,4K	3	8,5	8	-3,5
250K	3	0,0	7	0,0
0,5M	1	0,0	3	0,0
1M	0	0,0	1	0,0
Максимум*	1 Мбит/с		2 Мбит/с	

_____

* UBRR = 0, погрешность равна 0 %.

3.16 Двухпроводной последовательный интерфейс

3.16.1 Отличительные особенности

Отличительные особенности последовательного интерфейса:

- Простой, но в то же время многофункциональный интерфейс связи, требующий только две шины.

- Поддержка как ведущего режима работы, так и ведомого.
- Устройство может работать в качестве передатчика или приемника.
- 7-разрядное адресное пространство позволяет подключать к шине до 128 ведомых устройств.
- Поддержка арбитража при многочисленных ведущих устройствах.
- Скорость передачи данных до 400 кГц.
- Выходные драйверы с ограниченной скоростью нарастания выходного напряжения.
- Схема подавления шумов сдерживает импульсные выбросы на шинах.
- Программируемый адрес для ведомого устройства с поддержкой общего вызова.
- Распознавание собственного адреса приводит к пробуждению микроконтроллера, если он находится в режиме сна.

3.16.2 Определение шины интерфейса TWI

Двухпроводной последовательный интерфейс TWI идеально подходит для типичных прикладных программ микроконтроллера. Протокол TWI позволяет проектировщику системы внутренне связать до 128 различных устройств, используя лишь две двунаправленные шины, одна из которых предназначена для синхронизации (SCL), а другая – для данных (SDA). Единственные внешние компоненты, необходимые для аппаратной реализации шины, – это подтягивающий к плюсу питания резистор для каждой из шин интерфейса TWI (см.рисунок 3.87). Все устройства, подключенные к шине, имеют свои индивидуальные адреса, а механизм для разрешения конфликтной ситуации при обращении к шине внутренне обеспечивается протоколом TWI.

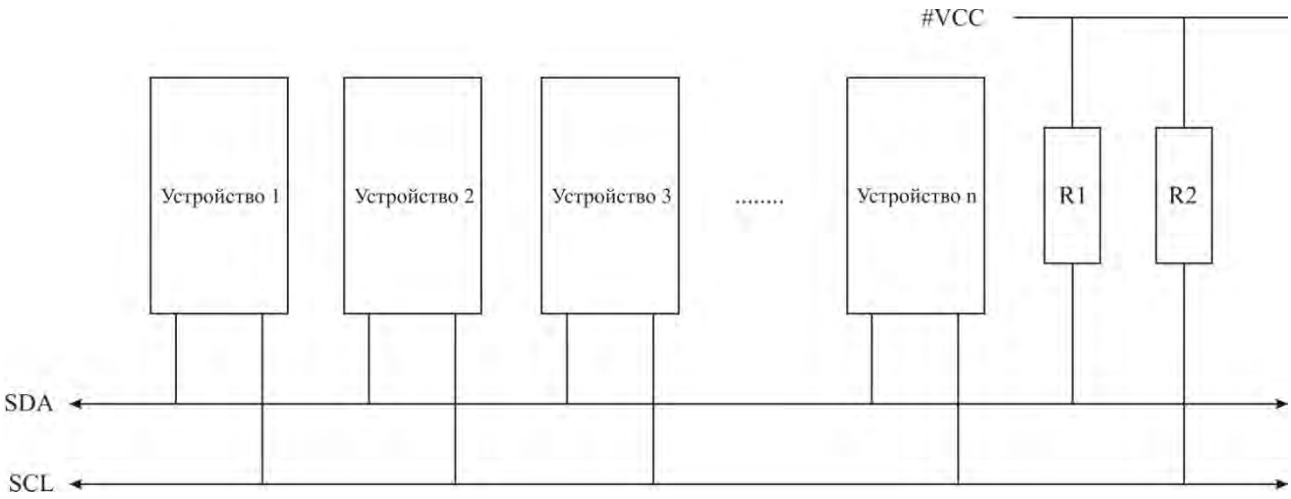


Рисунок 3.87 – Взаимосвязь шин интерфейса TWI

Терминология TWI

Приведенные в таблице 3.86 определения часто встречаются в данном подразделе.

Таблица 3.86 – Терминология TWI

Термин	Описание
Ведущее (главное) устройство	Это устройство, которое инициирует и останавливает передачу. Оно также формирует тактовый сигнал SCL
Ведомое (подчиненное) устройство	Адресация этого устройства выполняется с помощью ведущего
Передатчик	Данное устройство передает данные на шину
Приемник	Данное устройство считывает данные с шины

Электрические межсоединения

Как показано на рисунке 3.87, обе шины подключены к положительному напряжению питания через подтягивающие резисторы. Драйверы шин всех TWI-совместимых устройств являются транзисторами с открытым стоком или открытым коллектором. Благодаря этому реализована функция монтажного И, которая необходима для двунаправленной работы интерфейса. Низкий логический уровень на шине TWI генерируется, когда одно или более из TWI-устройств выводит логический ноль. Высокий уровень появляется, когда все TWI-устройства переводят свои выходы в третье состояние, позволяя подтягивающим резисторам задавать уровень логической единицы. Обратите внимание, что все устройства, подсоединенные к шине TWI, должны быть включены, чтобы сделать возможным выполнение какой бы то ни было операции на шине. Количество устройств, которые могут быть подключены к шине TWI, ограничивается только предельно допустимой емкостью шины (400 пФ) и 7-разрядным адресным пространством.

3.16.3 Передача данных и формат кадров

Передаваемые биты

Каждый информационный бит, передаваемый на шину TWI, сопровождается импульсом на линии синхронизации. Уровень сигнала на линии данных должен быть стабильным, когда на линии синхронизации присутствует логическая единица. Единственным исключением из этого правила является генерация условий начала (старта) и завершения (стопа) сеанса связи.

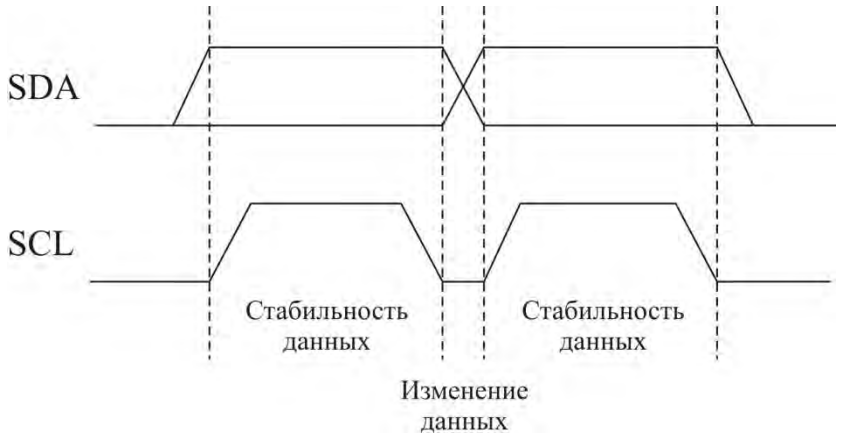


Рисунок 3.88 – Действительность данных

Условия СТАРТ и СТОП

Ведущее устройство начинает и заканчивает передачу данных. Передача начинается, когда главное устройство формирует условие СТАРТ на шине, и прекращается, когда оно формирует условие СТОП. Между условиями СТАРТ и СТОП шина считается занятой и никакое другое ведущее устройство не должно выполнять управляющие действия на шине. Существуют особые случаи, когда новое условие СТАРТа возникает между условиями СТАРТ и СТОП. Данный случай именуется как условие ПОВТОРНЫЙ СТАРТ и используется при необходимости инициировать главным устройством новый сеанс связи, не теряя при этом управление шиной. После ПОВТОРНОГО СТАРТа шина считается занятой до следующего СТОПа. Это идентично поведению после обычного СТАРТа, поэтому далее в документе описание условия СТАРТ относится и к условию СТАРТ и к условию ПОВТОРНЫЙ СТАРТ, если не указано иное. Как показано ниже, условия СТАРТ и СТОП представляют собой изменение логического уровня на линии SDA, когда на линии SCL присутствует логическая единица.

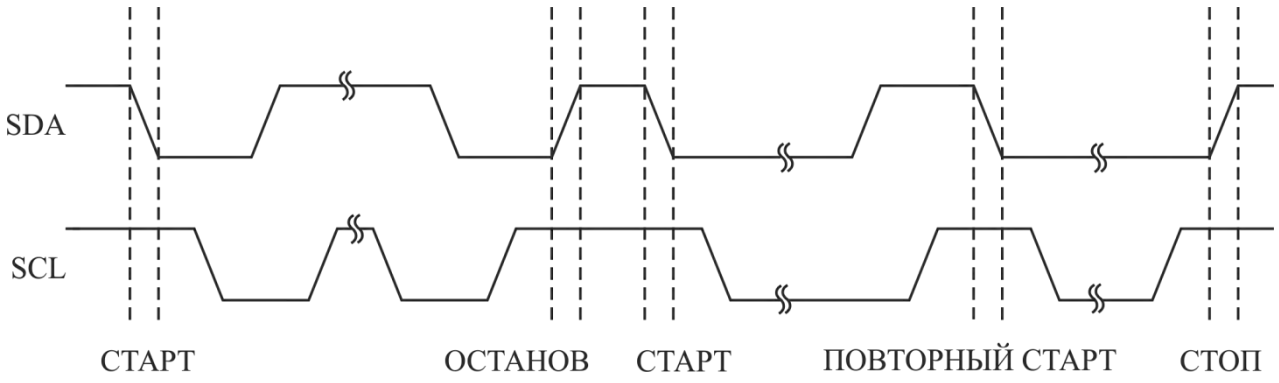


Рисунок 3.89 – Условия СТАРТ, ПОВТОРНЫЙ СТАРТ и СТОП

Формат адресного пакета

Длина всех адресных пакетов, передаваемых по шине TWI, составляет 9 битов: семь адресных битов, один бит управления для задания типа операции ЧТЕНИЕ/ЗАПИСЬ и один бит подтверждения. Если бит ЧТЕНИЕ/ЗАПИСЬ установлен, то должна выполниться операция чтения. Если этот бит сброшен, то следует произвести операцию записи. Когда ведомое устройство распознает, что происходит адресация, оно должно подтвердить это путем перевода линии SDA в низкий уровень на девятом такте SCL (формирование бита подтверждения ACK). Если адресуемое ведомое устройство занято или по каким-либо другим причинам не может обслужить ведущее устройство, то на линии SDA необходимо оставить высокий уровень во время такта подтверждения ACK. Ведущее устройство затем может передать условие СТОП или ПОВТОРНЫЙ СТАРТ для начала новой передачи. Адресный пакет, состоящий из адреса ведомого устройства и бита ЧТЕНИЕ или ЗАПИСЬ, обозначим как SLA+R и SLA+W, соответственно.

Старший значащий разряд адресного байта передается первым. Адреса ведомых устройств могут свободно назначаться проектировщиком, за исключением адреса 0000 000, который зарезервирован для общего вызова.

При определении общего вызова все ведомые устройства должны ответить переводом линии SDA в низкий уровень во время такта ACK. Общий вызов применяется, когда ведущее устройство желает передать одно и то же сообщение нескольким ведомым устройствам в системе. Когда адрес общего вызова, сопровождаемый битом ЗАПИСЬ, передается на шину, все ведомые устройства, готовящиеся к подтверждению общего вызова, переводят линию SDA в низкий уровень во время такта ACK. Последующие пакеты данных будут затем приниматься всеми ведомыми устройствами, которые подтвердили общий вызов. Обратите внимание, что передача адреса общего вызова, сопровождаемого битом ЧТЕНИЕ, не будет иметь смысла, поскольку в этом случае возникает конфликтная ситуация, если несколько ведомых устройств начинают передавать разные данные.

Все адреса с форматом 1111 xxx должны быть зарезервированы для будущего использования.

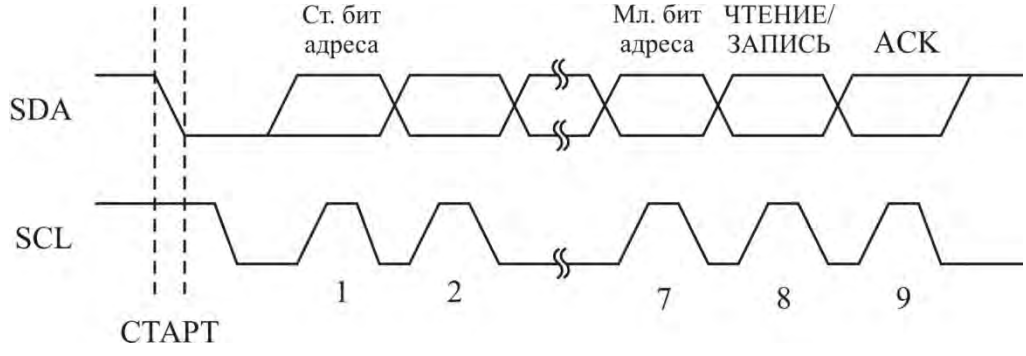


Рисунок 3.90 – Формат адресного пакета

Формат пакета данных

Длина всех пакетов данных, передаваемых по шине TWI, составляет 9 битов: один байт данных и один бит подтверждения. Во время передачи данных ведущее устройство генерирует тактовый импульс и условия СТАРТ и СТОП, при этом на приемник возлагается ответственность за подтверждение приема. Подтверждение (ACK) выполняется приемником путем перевода линии SDA в низкий уровень в течение девятого такта сигнала SCL. Если приемник оставляет высокий уровень на линии SDA, то это сигнализирует о том, что подтверждения не было (NACK). Когда приемник получил последний байт или по какой-то причине не может принять больше байтов, он должен информировать об этом передатчика путем посылки сигнала NACK после завершающего байта. Старший бит байта данных передается первым.

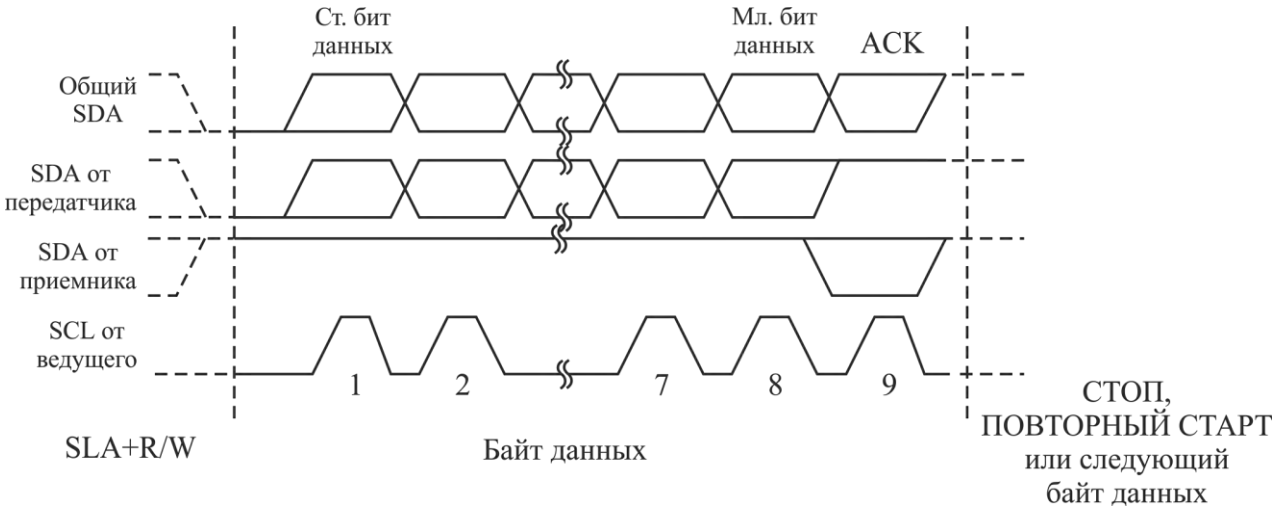


Рисунок 3.91 – Формат пакета данных

Совмещение пакетов адреса и данных во время сеанса связи

Сеанс связи по существу состоит из условия СТАРТ, адресного пакета SLA+R/W, одного или более пакетов данных и условия СТОП. Передача пустого сообщения, состоящего из условия СТАРТ, за которым сразу следует условие СТОП, является недопустимой. Обратите внимание, что монтажное И на линии SCL может использоваться для реализации подтверждения установления связи между ведущим и ведомым устройствами. Ведомое устройство может продлить период низкого уровня на линии SCL путем установки логического нуля на выводе SCL. Это полезно, когда скорость синхронизации, установленная ведущим устройством, слишком быстрая для ведомого устройства, или когда ведомое устройство нуждается в дополнительном времени для обработки информации между сеансами передачи данных. Ведомое устройство, продлевая период низкого уровня на линии SCL, не повлияет на длительность высокого уровня сигнала SCL, поскольку она определяется ведомым устройством. Как следствие, ведомое устройство может снизить скорость передачи данных, увеличив рабочий цикл SCL.

На рисунке 3.92 представлена типичная передача данных. Обратите внимание, что между адресным пакетом SLA+R/W и условием СТОП можно передать несколько байтов данных. Их число зависит от программного протокола, реализованного в прикладной программе.

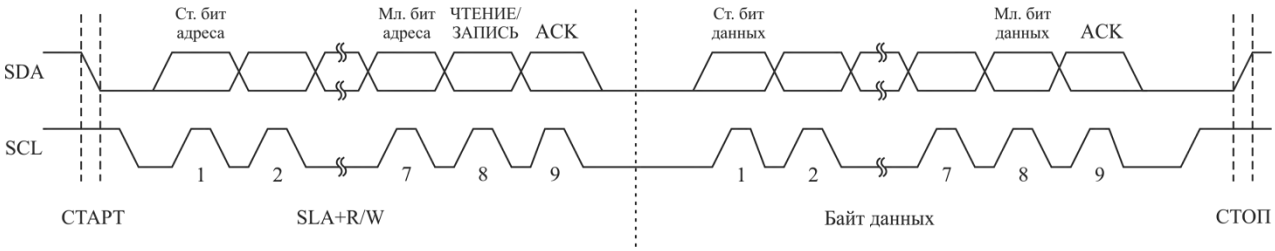


Рисунок 3.92 – Типичная передача данных

3.16.4 Системы шин с несколькими ведущими устройствами, арбитраж и синхронизация

Протокол TWI допускает использование систем шин с несколькими ведущими устройствами. Были приняты некоторые меры, чтобы гарантировать процесс передачи, даже если два или более ведущих устройства инициируют передачу в одно и то же время. Две проблемы возникают в системах шин с несколькими ведущими устройствами.

1 Алгоритм должен быть реализован так, чтобы только одно ведущее устройство могло завершить передачу. Все остальные ведущие устройства должны прекратить передачу, когда они обнаружат, что потеряли приоритет в процессе отбора. Этот процесс отбора называется арбитраж. Когда какое-либо конкурирующее ведущее устройство обнаруживает потерю приоритета, оно должно немедленно переключиться в ведомый режим, чтобы проверить, не к нему ли обращается ведущее устройство, которое выиграло процесс арбитража. То обстоятельство, что несколько ведущих устройств начали передачу в одно и то же время, не должно становиться доступным для обнаружения ведомыми устройствами, то есть передаваемые данные не должны быть повреждены.

2 Разные ведущие устройства могут использовать разные частоты сигнала SCL. Необходимо разработать схему, которая позволяла бы синхронизировать последовательные тактовые сигналы от всех ведущих устройств, чтобы дать возможность передаче продолжаться в неизменном режиме. Это облегчит процесс арбитража.

Использование принципа монтажного И для шин позволяет решить обе эти проблемы. Над последовательными тактовыми сигналами от всех ведущих устройств выполняется операция монтажного И, порождающая объединенный тактовый сигнал с периодом высокого уровня, равным аналогичному периоду от ведущего устройства с самой короткой длительностью высокого уровня. Период низкого уровня объединенного тактового сигнала равен аналогичному периоду от ведущего устройства с самой продолжительной длительностью низкого уровня. Обратите внимание, что все ведущие устройства ожидают сигнала на линии SCL, фактически начиная счет длительности своих периодов высокого и низкого уровня, когда объединенный сигнал SCL переходит в высокое или низкое состояние, соответственно.

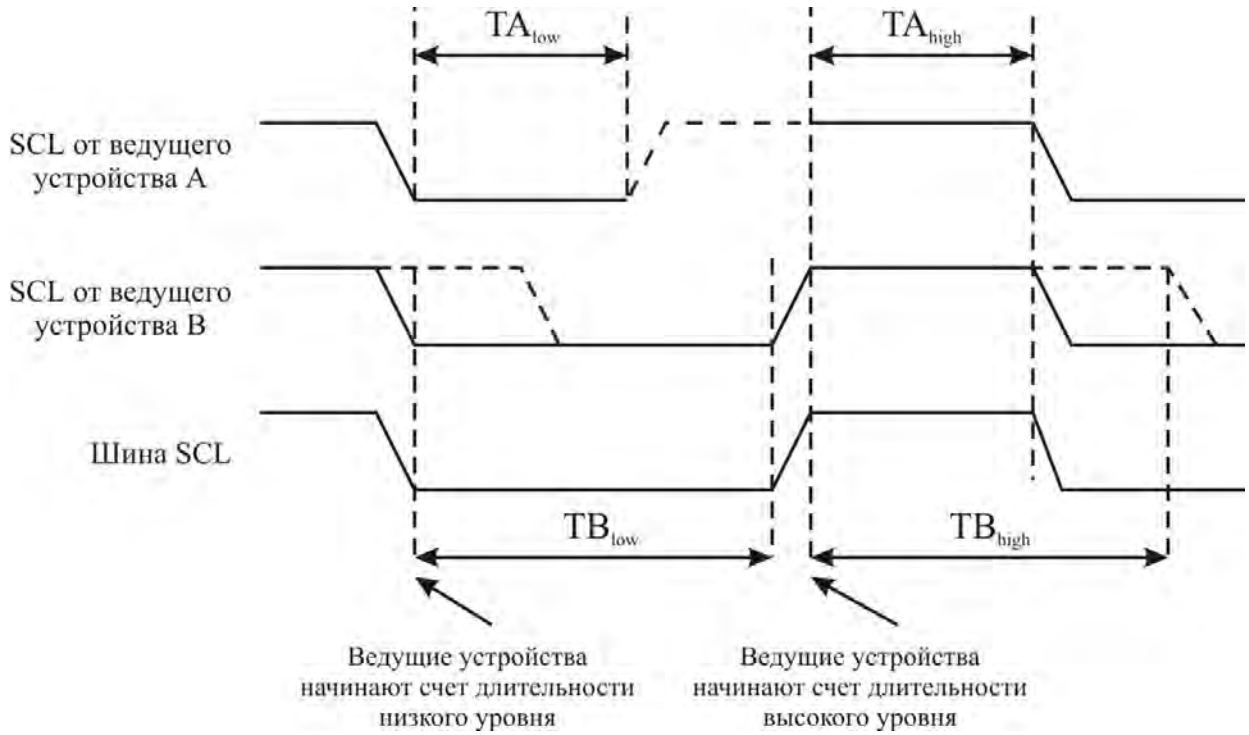


Рисунок 3.93 – Синхронизация на линии SCL между несколькими ведущими устройствами

Арбитраж осуществляется всеми ведущими устройствами, которые непрерывно отслеживают линию SDA после вывода данных. Если значение, считанное с линии SDA, не соответствует значению, которое выдало ведущее устройство, то данное устройство теряет приоритет. Обратите внимание, что приоритет теряется только в том случае, когда одно ведущее устройство выводит высокий уровень сигнала SDA, в то время как другое ведущее устройство выводит низкий логический уровень. Ведущее устройство, потерявшее приоритет, должно немедленно переключиться в ведомый режим, чтобы проверить, не к нему ли обращается ведущее устройство, которое выиграло процесс арбитража. Линия SDA должна быть оставлена в состоянии логической единицы, но ведущие устройства, потерявшие приоритет, могут продолжать генерировать тактовый сигнал до окончания передачи текущего пакета данных или адресного пакета. Процесс арбитража выполняется то тех пор, пока не останется активным только одно ведущее устройство, и для этого в ряде случаев может потребоваться передать много битов. Если несколько главных устройств пытается обратиться к одному и тому же подчиненному, то процесс арбитража переносится на пакет данных.

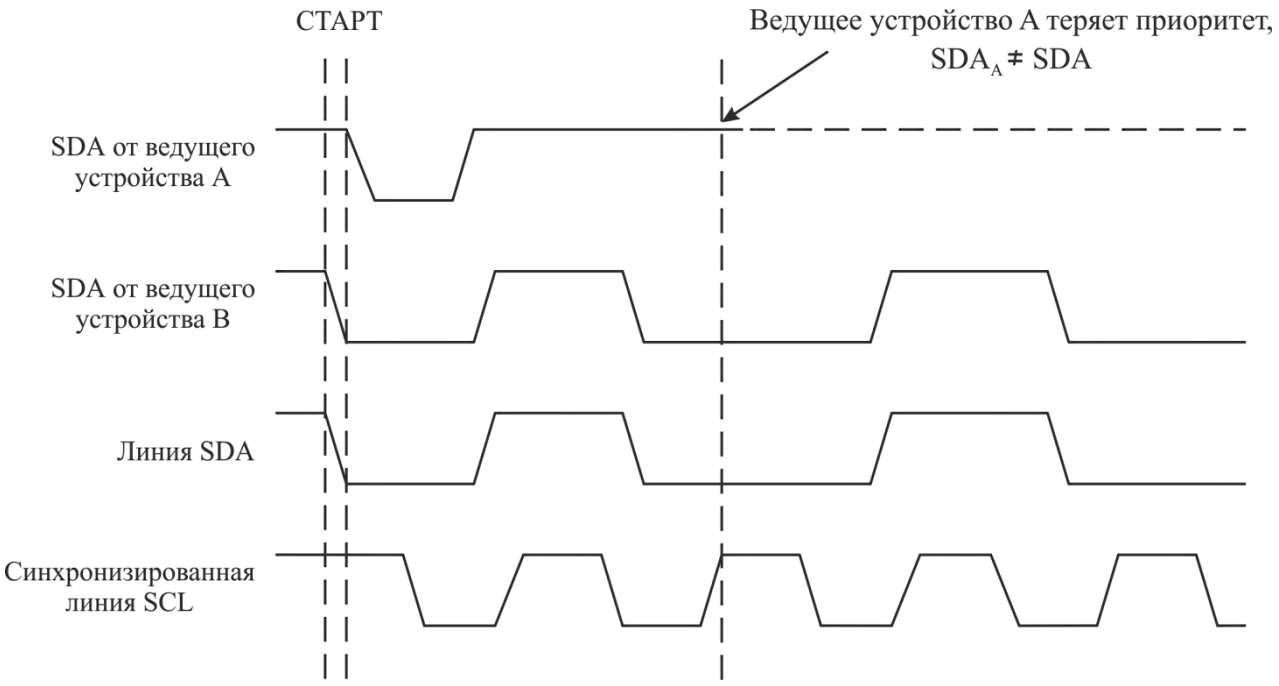


Рисунок 3.94 – Арбитраж между двумя ведущими устройствами

Обратите внимание, что арбитраж не разрешается в следующих случаях:
- между условием ПОВТОРНЫЙ СТАРТ и битом данных;
- между условием СТОП и битом данных;
- между условиями ПОВТОРНЫЙ СТАРТ и СТОП.

Программное обеспечение пользователя должно гарантировать, что эти недопустимые условия арбитража никогда не возникнут. Этим подразумевается, что в системах с несколькими ведущими устройствами во всех сеансах передачи данных должна использоваться одна и та же конфигурация адресного пакета SLA+R/W и пакетов данных. Другими словами, все сеансы передачи должны содержать одинаковое количество пакетов данных, в противном случае результат арбитража будет неопределенным.

3.16.5 Краткий обзор модуля TWI

Модуль TWI состоит из нескольких подмодулей, как показано на рисунке 3.95. Все регистры, выделенные жирной линией, доступны через шину данных микроконтроллера.

Выводы SCL и SDA

Эти выводы связывают двухпроводной последовательный интерфейс микроконтроллера с остальной частью системы. Выходные драйверы содержат ограничитель скорости нарастания выходного напряжения для согласования со спецификацией TWI. Входные каскады содержат блок подавления помех, задача которого состоит в устранении импульсных выбросов длительностью менее 50 наносекунд. Обратите внимание, что внутренние подтягивающие резисторы в контактных площадках микроконтроллера можно включить путем установки битов PORTD0 и PORTD1, которые соответствуют выводам SCL и SDA, как описано в подразделе 3.7 «Порты ввода-вывода». Использование встроенных подтягивающих резисторов в ряде случаев позволяет исключить необходимость применения внешних.

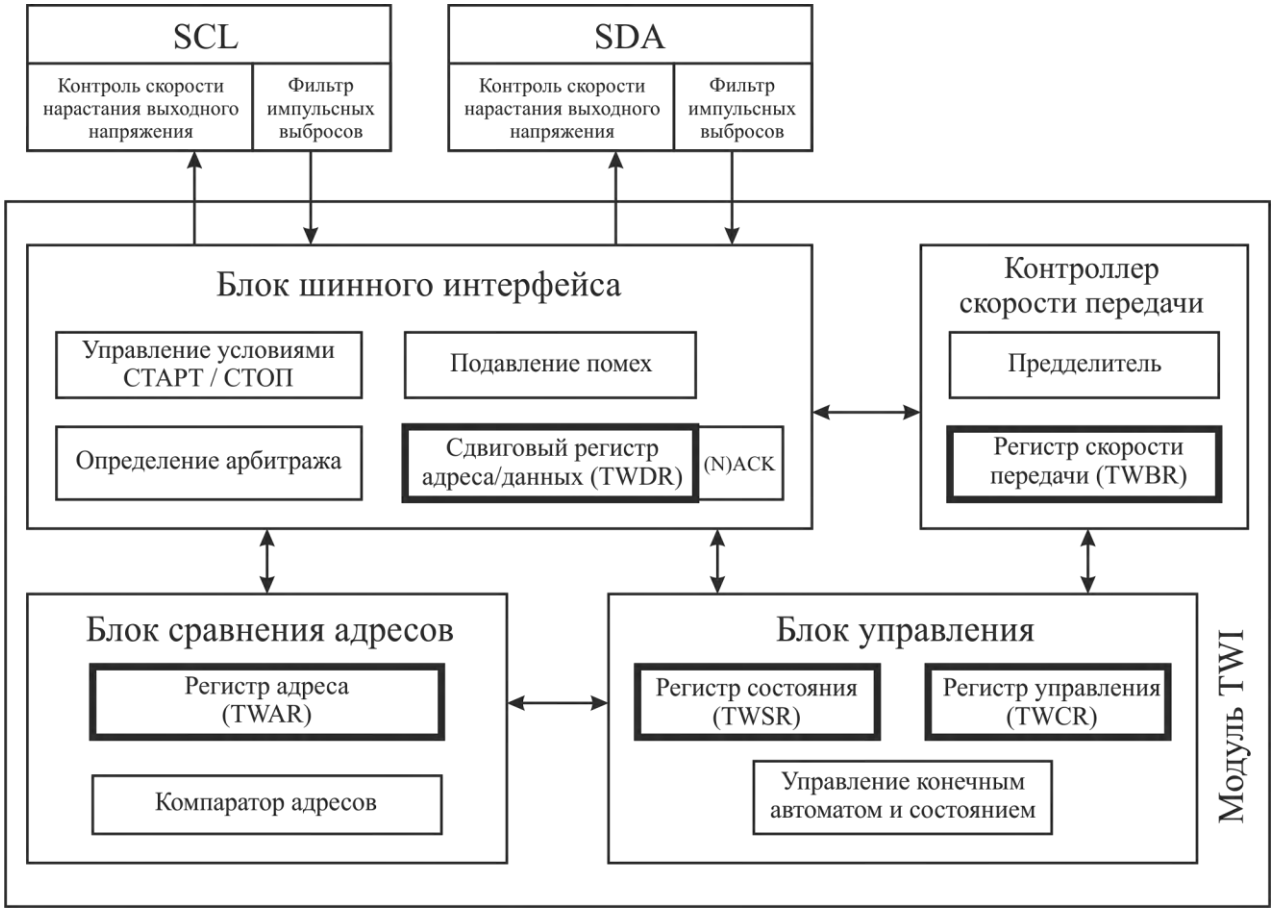


Рисунок 3.95 – Структурная схема модуля TWI

Контроллер скорости передачи

Данный блок контролирует период импульсов SCL в ведущем режиме работы. Период SCL задается регистром скорости передачи TWI (TWBR) и значением битов предделителя в регистре состояния TWI (TWSR). В ведомом режиме значения скорости передачи или установки предделителя не оказывают влияния на работу, но частота синхронизации ЦПУ ведомого устройства должна быть минимум в 16 раз выше частоты SCL. Обратите внимание, что ведомые устройства могут продлевать период низкого уровня на линии SCL, уменьшая тем самым период синхронизации шины TWI. Частота SCL формируется согласно следующему выражению

$$f_{SCL} = \frac{f_{CPU}}{16 + 2(TWBR) \cdot 4^{TWPS}},$$

где TWBR – значение регистра скорости передачи TWI;

TWPS – значение битов предделителя в регистре состояния TWI.

Примечание – Значение подтягивающего резистора следует выбирать в соответствии с частотой SCL и емкостной нагрузкой шины.

Блок шинного интерфейса

Этот блок содержит сдвиговый регистр данных и адреса (TWDR), контроллер СТАРТа/ СТОПа и аппаратные средства для арбитража. TWDR содержит адрес или байты данных, которые необходимо передать, или же принятый байт адреса или данных. Помимо 8-разрядного регистра TWDR в состав блока шинного интерфейса также входит регистр, хранящий значение передаваемого или принятого бита (N)ACK. 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 (N)ACK можно определить по значению регистра TWSR.

Контроллер СТАРТа/СТОПа отвечает за формирование и определение 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 устройства.

Если TWI инициировал передачу в качестве ведущего устройства, то схема арбитража непрерывно контролирует передачу, пытаясь определить, идет ли процесс арбитража. Если TWI теряет приоритет, то формируется соответствующий сигнал для блока управления. После этого могут выполняться корректные действия и генерироваться соответствующие коды состояний.

Блок сравнения адресов

Блок сравнения адресов проверяет, равен ли принятый адрес значению 7-разрядного адреса в регистре TWAR. Если установлен бит разрешения распознавания общего вызова TWGCE в регистре TWAR, то все входящие адресные биты будут дополнительно сравниваться с адресом общего вызова. При совпадении адресов подается сигнал блоку управления, что позволяет ему выполнить необходимые действия. В зависимости от установок регистра TWCR подтверждение адреса TWI может произойти, а может и не произойти. Блок сравнения адресов способен функционировать даже тогда, когда микроконтроллер находится в режиме сна. Это позволяет микроконтроллеру пробуждаться, когда к нему обращается ведущее устройство. Когда во время сравнения адресов в режиме хранения происходит другое прерывание (например, INT0), которое приводит к пробуждению ЦПУ, то TWI прекращает работу и возвращается в состояние неадресованного ведомого приемника. Если данное обстоятельство вызывает какие-либо проблемы, необходимо удостовериться, что прерывание при совпадении адресов является единственным разрешенным прерыванием при переходе в режим микропотребления.

Блок управления

Блок управления наблюдает за шиной TWI и генерирует отклики, соответствующие установкам в регистре управления TWI (TWCR). Если на шине TWI возникает событие, которое требует внимания со стороны программы, то устанавливается флаг прерывания TWI (TWINT). На следующем такте происходит обновление регистра состояния TWI (TWSR), то есть выполняется запись кода состояния, идентифицирующего возникшее событие. Регистр TWSR содержит соответствующую информацию о состоянии, только когда установлен флаг прерывания TWI. В остальное время в регистре TWSR содержится специальный код состояния, который показывает, что нет доступной информации о состоянии TWI. Пока флаг TWINT установлен, линия SCL удерживается в низком уровне. Это дает возможность прикладной программе завершить свои задачи перед тем, как разрешить продолжение сеанса связи.

Флаг TWINT устанавливается в следующих случаях:

- после передачи условия СТАРТ / ПОВТОРНЫЙ СТАРТ;
- после передачи адресного пакета SLA+R/W;
- после передачи адресного байта;
- после потери приоритета;
- после адресации TWI собственным адресом или общим вызовом;
- после приема байта данных;
- после приема условия СТОП или ПОВТОРНЫЙ СТАРТ в режиме адресации ведомого;
- после возникновения ошибки на шине из-за непредусмотренного условия СТАРТ или СТОП.

3.16.6 Описание регистров TWI

Регистр скорости передачи TWI – TWBR

Бит	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7–0: Биты регистра скорости передачи TWI

TWBR задает коэффициент деления частоты для контроллера скорости передачи. Последний представляет собой делитель частоты, который формирует сигнал синхронизации SCL в ведущих режимах работы. Выше в подпункте «Контроллер скорости передачи» показана методика вычисления скорости передачи.

Регистр управления TWI – TWCR

Бит	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	TWCR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч	Ч/З	Ч	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр TWCR предназначен для управления работой TWI. Он используется для выполнения таких операций, как:

- включение TWI;
- начало доступа для ведущего устройства путем подачи условия СТАРТ на шину;
- генерация подтверждения приема;
- генерация условия СТОП;
- управление остановом шины, пока данные, предназначенные для записи на шину, находятся в регистре TWDR.

Регистр TWCR также сигнализирует о конфликте при попытке записи в TWDR, когда доступ к нему запрещен.

Разряд 7 – TWINT: Флаг прерывания TWI

Данный бит устанавливается аппаратно, когда TWI завершил свое 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 должен сбрасываться программно путем записи в него логической единицы. Обратите внимание, что данный флаг не сбрасывается автоматически при выполнении программы обработки прерываний. Также нужно учесть, что сброс данного флага приводит к возобновлению работы TWI. Из этого следует, что программный сброс данного флага необходимо выполнять после завершения опроса регистров TWAR, TWSR и TWDR.

Разряд 6 – TWEA: Бит подтверждения включения TWI

Бит TWEA управляет формированием сигнала подтверждения. Когда в бит TWEA записана логическая единица, то на шине TWI генерируется сигнал ACK, если выполняются следующие условия:

1 Принят собственный адрес.

2 Принят общий вызов, когда установлен бит TWGCE в регистре TWAR.

3 Принят байт данных в режиме ведущего приемника или режиме ведомого приемника.

Запись логического нуля в бит TWEA позволяет временно отключить устройство от двухпроводной последовательной шины. Для возобновления распознавания адреса необходимо снова записать в данный бит логическую единицу.

Разряд 5 – TWSTA: Бит условия СТАРТ

Программа записывает единицу в данный бит, когда желает стать ведущим на двухпроводной последовательной шине. Аппаратное обеспечение TWI проверяет, доступна ли шина и генерирует условие СТАРТ на шине, если она свободна. Однако если шина занята, то TWI ожидает появления условия СТОП, а затем генерирует новое условие СТАРТ для утверждения ведущего статуса на шине. Бит TWSTA должен сбрасываться программно после передачи условия СТАРТ.

Разряд 4 – TWSTO: Бит условия СТОП

Запись логической единицы в бит TWSTO в ведущем режиме приводит к генерации условия СТОП на двухпроводной последовательной шине. Когда на шине выполняется условие СТОП, бит TWSTO сбрасывается автоматически. В ведомом режиме установка бита TWSTO в единицу может использоваться для возврата в исходное положение из состояния ошибки. В этом случае условие СТОП не генерируется, но интерфейс TWI возвращается к четко определенному режиму неадресованного ведомого и переводит линии SCL и SDA в высокоимпедансное состояние.

Разряд 3 – TWWC: Флаг конфликтной ситуации при попытке записи

Бит TWWC устанавливается при попытке записи в регистр данных TWDR, когда бит TWINT равен нулю. Данный флаг сбрасывается при выполнении записи в регистр TWDR, когда TWINT = 1.

Разряд 2 – TWEN: Бит разрешения работы TWI

Бит TWEN разрешает работу TWI и активирует интерфейс TWI. Когда TWEN = 1, TWI берет на себя функции управления линиями ввода-вывода, подключенными к SCL и SDA. При этом разрешается работа ограничителей скорости нарастания выходного напряжения и фильтров импульсных выбросов. Если бит TWEN равен нулю, то TWI отключается, и все сеансы передачи прекращаются независимо от выполняемой операции.

Разряд 1 – резервный бит

Данный бит является резервным и всегда считывается как ноль.

Разряд 0 – TWIE: Разрешение прерывания TWI

Когда TWIE = 1 и установлен бит I в регистре SREG, то запрос на прерывание TWI будет генерироваться до тех пор, пока будет установлен флаг TWINT.

Регистр состояния TWI – TWSR

Бит	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	TWSR
Чтение/ Запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч/З	Ч/З	
Начальное значение	1	1	1	1	1	0	0	0	

Разряды 7–3: TWS: Состояние TWI

Эти 5 битов отражают состояние логики TWI и двухпроводной последовательной шины. Различные коды состояний описаны далее в этом подразделе. Обратите внимание

что значение, считываемое из регистра TWSR, содержит как 5-разрядный код состояния, так и 2-разрядное значение предделителя. Программист должен маскировать биты предделителя значением «0» во время проверки битов состояния. В этом случае проверка состояния не будет зависеть от настроек предделителя. Этот принцип используется далее в данном документе, если только не указано иное.

Разряд 2 – Резервный бит

Данный бит является резервным и всегда считывается как ноль.

Разряды 1, 0 – TWPS: Биты предделителя TWI

Данные биты обладают полным доступом (чтение/запись) и позволяют управлять предделителем скорости передачи.

Таблица 3.87 – Предделитель скорости передачи TWI

TWPS1	TWPS0	Значение предделителя
0	0	1
0	1	4
1	0	16
1	1	64

Формула для вычисления скорости передачи представлена в подпункте выше «Контроллер скорости передачи». В уравнении используется значение битов TWPS1, 0.

Регистр данных TWI – TWDR

Бит	7	6	5	4	3	2	1	0	
	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	1	1	1	1	1	1	1	1	

В режиме передатчика регистр TWDR содержит следующий байт для передачи. В режиме приемника регистр TWDR содержит последний принятый байт. Данный регистр является перезаписываемым, пока TWI не находится в процессе передачи/приема данных. Такое состояние наступает, когда происходит аппаратная установка флага прерывания TWINT.

Обратите внимание, что регистр данных не может инициализироваться пользователем, пока не произойдет первое прерывание. Данные в регистре TWDR остаются неизменными, пока бит TWINT установлен. Во время сдвига данных из регистра одновременно выполняется подача новых данных с шины в регистр. TWDR всегда содержит последний байт, представленный на шине, за исключением ситуации, когда происходит пробуждение микроконтроллера из спящего режима по прерыванию TWI. В этом случае состояние TWDR является неопределенным. В случае потери приоритета шины не возникает потеря данных при переходе от ведущего устройства к ведомому. Управление битом ACK выполняется автоматически с помощью логики TWI, ЦПУ не может получать доступ к этому биту напрямую.

Разряды 7–0: TWD: Регистр данных TWI

Эти восемь битов составляют следующий байт данных, который необходимо передать по двухпроводной последовательной шине, или же последний принятый байт.

Регистр адреса TWI – TWAR

Бит	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	1	1	1	1	1	1	1	0	

В регистр TWAR следует загружать 7-разрядный адрес (его необходимо записывать в старшие семь битов), которому будет соответствовать TWI, когда он запрограммирован как ведомый передатчик или приемник. Этот 7-разрядный адрес не используется в ведущих режимах. В системах с несколькими ведущими устройствами регистр TWAR должен содержать действительный адрес в тех ведущих устройствах, которые могут быть адресованы как ведомые другими ведущими шинами.

Младший разряд регистра TWAR используется для разрешения распознавания адреса общего вызова ($00). Специальный компаратор выполняет сравнение собственного адреса (или адреса общего вызова) с принятым адресом. Если обнаруживается совпадение, то генерируется запрос на прерывание.

Разряды 7–1: TWA: Регистр адреса TWI

Эти семь битов составляют ведомый адрес блока TWI.

Разряд 0 – TWGCE: Бит разрешения обнаружения общего вызова по шине TWI

После установки данного бита разрешается работа схемы распознавания общего вызова, передаваемого по двухпроводной последовательной шине.

3.16.7 Рекомендации по использованию TWI

TWI имеет байтовую организацию и управляется прерываниями. Прерывания возникают после обнаружения любых событий на шине, например, после приема байта или передачи условия СТАРТ. Управление TWI по прерываниям позволяет освободить программное обеспечение для выполнения других задач во время передачи байта данных. Обратите внимание, что бит разрешения прерываний TWI (TWIE) в регистре TWCR наряду с битом общего разрешения прерываний (I) в регистре SREG позволяют прикладной программе определить, приведет ли установка флага TWINT к формированию запроса на прерывание или нет. Если бит TWIE сброшен, то программа должна опросить флаг TWINT, чтобы выявить действия на шине TWI.

После установки флага TWINT интерфейс TWI завершает работу и ожидает реакции программы. В этом случае регистр статуса TWSR содержит значение, которое показывает текущее состояние шины TWI. Затем программа может задать дальнейшее поведение интерфейса TWI, манипулируя регистрами TWCR и TWDR.

На рисунке 3.96 приведен простой пример того, как программа может взаимодействовать с аппаратным обеспечением TWI. В данном примере предполагается, что ведущее устройство желает передать ведомому устройству один байт данных. Это описание весьма абстрактное. Более подробное объяснение приводится далее в этом пункте. Ниже представлен также простой пример программы, реализующей желаемое поведение TWI.

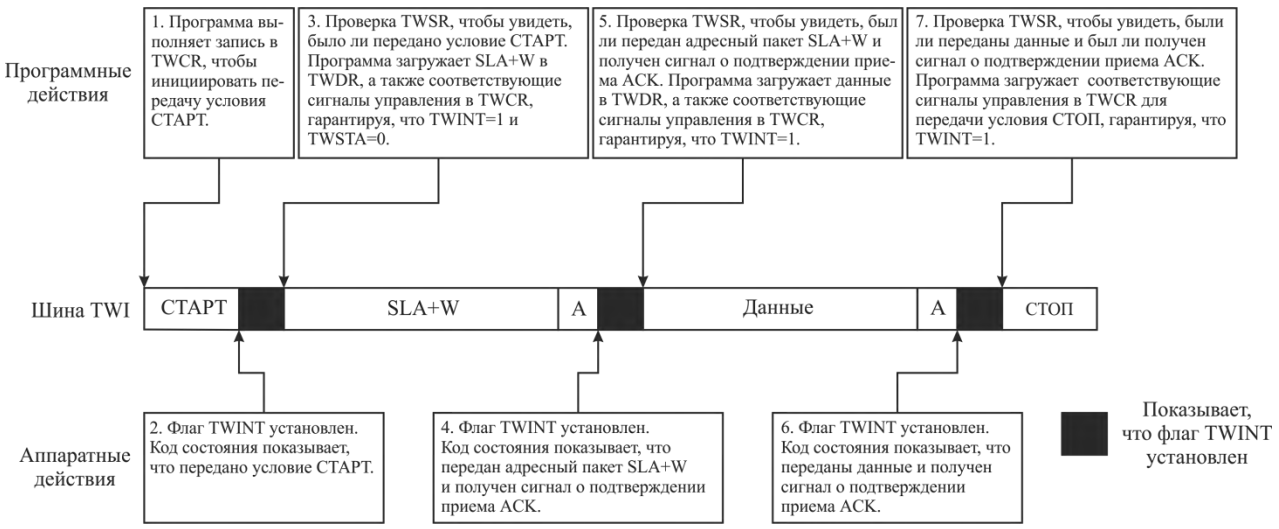


Рисунок 3.96 – Взаимодействие программы и аппаратного обеспечения TWI при типичной передаче данных

1 Первым шагом при передаче данных является передача условия СТАРТ. Она производится путем записи специфического значения в TWCR. О значении, которое следует записывать, будет сказано позже. Однако необходимо следить, чтобы в записываемом в регистр значении был установлен бит TWINT. Запись логической единицы в бит TWINT сбрасывает этот флаг. TWI не начнет выполнять какую бы то ни было операцию, пока будет установлен бит TWINT в регистре TWCR. Сразу после программного сброса флага TWINT начинается передача условия СТАРТ.

2 После передачи условия СТАРТ устанавливается флаг TWINT в регистре TWCR, а содержимое TWSR обновляется значением кода состояния, показывая, что условие СТАРТ было успешно передано.

3 Теперь программа должна выполнить проверку значения TWSR, чтобы убедиться в том, что условие СТАРТ было успешно передано. Если значение TWSR свидетельствует об иной ситуации, то программа может выполнить особое действие, например, вызов процедуры обработки ошибок. Если код состояния имеет ожидаемое значение, то программа должна загрузить адресный пакет SLA+W в TWDR. Следует помнить, что TWDR используется для хранения как адреса, так и данных. После того, как в TWDR было загружено желаемое значение SLA+W, в регистр TWCR необходимо записать специфическое значение, которое служит командой для передачи адресного пакета SLA+W, находящегося в TWDR. О том, какое именно значение следует записывать, будет сказано позже. Однако необходимо следить, чтобы в записываемом в регистр значении был установлен бит TWINT. Запись логической единицы в бит TWINT сбрасывает этот флаг. TWI не начнет выполнять операцию, пока будет установлен бит TWINT в регистре TWCR. Сразу после программного сброса флага TWINT начинается передача адресного пакета.

4 После передачи адресного пакета устанавливается флаг TWINT в регистре TWCR, а содержимое TWSR обновляется значением кода состояния, показывая, что адресный пакет был успешно передан. В коде состояния также отражается, было ли подтверждение приема адресного пакета со стороны подчиненного устройства или нет.

5 Теперь программа должна выполнить проверку значения TWSR, чтобы убедиться в том, что адресный пакет был успешно передан, и что бит подтверждения ACK имеет ожидаемое значение. Если значение TWSR свидетельствует об иной ситуации, то программа может выполнить особое действие, например, вызов процедуры обработки ошибок. Если код состояния имеет ожидаемое значение, то программа должна загрузить пакет данных в TWDR. Далее необходимо записать в регистр TWCR специфическое значение, которое служит командой для передачи пакета данных, находящегося в TWDR. О том, какое именно значение следует записывать, будет сказано позже. Однако необходимо следить, чтобы в

записываемом в регистр значении был установлен бит TWINT. Запись логической единицы в бит TWINT сбрасывает этот флаг. TWI не начнет выполнять операцию, пока будет установлен бит TWINT в регистре TWCR. Сразу после программного сброса флага TWINT начинается передача пакета данных.

6 После передачи пакета данных устанавливается флаг TWINT в регистре TWCR, а содержимое TWSR обновляется значением кода состояния, показывая, что пакет данных был успешно передан. В коде состояния также отражается, было ли подтверждение приема пакета данных со стороны подчиненного устройства или нет.

7 Теперь программа должна выполнить проверку значения TWSR, чтобы убедиться в том, что пакет данных был успешно передан, и что бит подтверждения ACK имеет ожидаемое значение. Если значение TWSR свидетельствует об иной ситуации, то программа может выполнить особое действие, например, вызов процедуры обработки ошибок. Если код состояния имеет ожидаемое значение, то программа должна записать в регистр TWCR специфическое значение, которое служит командой для передачи условия СТОП. О том, какое именно значение следует записывать, будет сказано позже. Однако необходимо следить, чтобы в записываемом в регистр значении был установлен бит TWINT. Запись логической единицы в бит TWINT сбрасывает этот флаг. TWI не начнет выполнять операцию, пока будет установлен бит TWINT в регистре TWCR. Сразу после программного сброса флага TWINT начинается передача условия СТОП. Обратите внимание, что флаг TWINT не устанавливается по завершении передачи условия СТОП.

Несмотря на простоту приведенного примера, он показывает принципы, лежащие в основе любой передачи через TWI. Из вышесказанного можно сделать следующие выводы:

- После того как TWI завершил работу и ожидает отклика программы, устанавливается флаг TWINT. Линия SCL будет иметь низкий уровень до тех пор, пока не сбросится флаг TWINT.

- Когда установлен флаг TWINT, пользователь должен обновить все регистры TWI значениями, которые соответствуют следующему этапу работы шины TWI. Например, в TWDR следует загрузить значение, которое необходимо передать в следующем цикле.

- После обновления всех регистров TWI и завершения других ожидающих программных задач осуществляется запись в TWCR. При выполнении записи в регистр TWCR необходимо, чтобы бит TWINT был установлен. Запись логической единицы в бит TWINT сбрасывает этот флаг. Тогда TWI приступает к выполнению операции, которая задана настройками в регистре TWCR.

Далее показан пример программы на языках Ассемблер и Си. В примере предполагается, что некоторые символьные обозначения определены в присоединенных файлах.

Примечание – При разработке примеров предполагалось, что подключен файл специфических заголовков. Если адресуемый регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить инструкциями доступа к расширенной памяти ввода-вывода «LDS» и «STS» в сочетании с командами «SBRS», «SBRC», «SBR» и «CBR».

Пример программы на Ассемблере	Пример программы на Си	Комментарии
`ldi r16, (1<<TWINT)|(1<<TWSTA)| (1<<TWEN)` `out TWCR, r16`	`TWCR = (1<<TWINT)|(1<<TWSTA)| (1<<TWEN)`	Передача условия СТАРТ

`wait1:`
`in r16,TWCR`
`sbrs r16,TWINT`
`rjmp wait1`
	`while (!(TWCR & (1<<TWINT)))`
	`;`
		Ожидание установки флага TWINT. Это означает, что было передано условие СТАРТ

`in r16,TWSR`
`andi r16, 0xF8`
`cpi r16, START`
`brne ERROR`
	`if ((TWSR & 0xF8) != START)`
	`ERROR();`
		Проверка значения регистра состояния TWI. Маскировка битов предделителя. Если код состояния не соответствует условию СТАРТ, то выполняется 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 SLA+W в регистр TWDR. Сброс бита TWINT в TWCR для начала передачи адреса

`wait2:`
`in r16,TWCR`
`sbrs r16,TWINT`
`rjmp wait2`
	`while (!(TWCR & (1<<TWINT)))`
	`;`
		Ожидание установки флага TWINT. Это означает, что был передан адресный пакет SLA+W, и был получен сигнал о подтверждении приема либо об отсутствии подтверждения (ACK/NACK)

`in r16,TWSR`
`andi r16, 0xF8`
`cpi r16, MT_SLA_ACK`
`brne ERROR`
	`if ((TWSR & 0xF8) != MT_SLA_ACK)`
	`ERROR();`
		Проверка значения регистра состояния TWI. Маскировка битов предделителя. Если код состояния не соответствует условию MT_SLA_ACK, то выполняется 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);`
		Загрузка данных в регистр TWDR. Сброс бита TWINT в TWCR для начала передачи данных

`wait3:`
`in r16,TWCR`
`sbrs r16,TWINT`
`rjmp wait3`
	`while (!(TWCR & (1<<TWINT)))`
	`;`
		Ожидание установки флага TWINT. Это означает, что данные были переданы, и был получен сигнал о подтверждении приема либо об отсутствии подтверждения (ACK/NACK)

`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. Маскировка битов предделителя. Если код состояния не соответствует условию MT_DATA_ACK, то выполняется переход на ERROR

`ldi r16, (1<<TWINT)|(1<<TWEN)| (1<<TWSTO)` `out TWCR, r16`	`TWCR = (1<<TWINT)|(1<<TWEN)| (1<<TWSTO);`	Передача условия СТОП

3.16.8 Режимы передачи

TWI может работать в одном из четырех основных режимов. Они называются: ведущий передатчик (MT), ведущий приемник (MR), ведомый передатчик (ST) и ведомый 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 ЭПД TWI, а режим MR – для считывания данных из ЭПД. Если в системе имеются другие ведущие устройства, то некоторые из них могут передавать данные TWI, а затем используется режим SR. Какие из режимов разрешены, определяет программа.

В следующих подпунктах описывается каждый из этих режимов. Возможные значения кодов состояния представлены вместе с рисунками, поясняя процесс передачи данных в каждом из режимов. На рисунках используются следующие аббревиатуры:

S – Условие СТАРТ;

RS – Условие ПОВТОРНЫЙ СТАРТ;

R – Бит «Чтение» (высокий уровень на линии SDA);

W – Бит «Запись» (низкий уровень на линии SDA);

A – Бит подтверждения (низкий уровень на линии SDA);

A# – Бит отсутствия подтверждения (высокий уровень на линии SDA);

DATA – 8-разрядный байт данных;

P – Условие СТОП;

SLA – Адрес.

На рисунках 3.98 – 3.104 окружности используются для того, чтобы обозначить, что флаг TWINT установлен. Число, записанное внутри окружности, показывает код состояния, хранящийся в регистре TWSR, где биты предделителя замаскированы нулем. На данном этапе ожидаются действия со стороны программы, чтобы продолжить или завершить передачу TWI. Передача TWI приостанавливается до тех пор, пока флаг TWINT не будет сброшен программно.

Когда установлен флаг TWINT, код состояния из регистра TWSR используется для определения, какое действие следует выполнить программе. В таблицах 3.88 – 3.91 представлена информация о том, какие программные действия должны быть предприняты при различных значениях кода состояния. Обратите внимание, что в данных таблицах биты предделителя замаскированы.

Режим ведущего передатчика

В режиме ведущего передатчика ряд байтов данных передается ведомому приемнику (см. рисунок 3.97). Для входа в ведущий режим необходимо передать условие СТАРТ. Формат следующего адресного пакета определяет в какой режим следует перейти: ведущий передатчик или ведущий приемник. Если передается SLA+W, то вводится режим MT (ведущий передатчик), а если SLA+R, то вводится режим MR (ведущий приемник). В данном подпункте предполагается, что для всех кодов состояний, упоминаемых здесь, биты предделителя равны нулю либо замаскированы.

Передача условия СТАРТ инициируется путем записи в регистр TWCR следующего значения:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

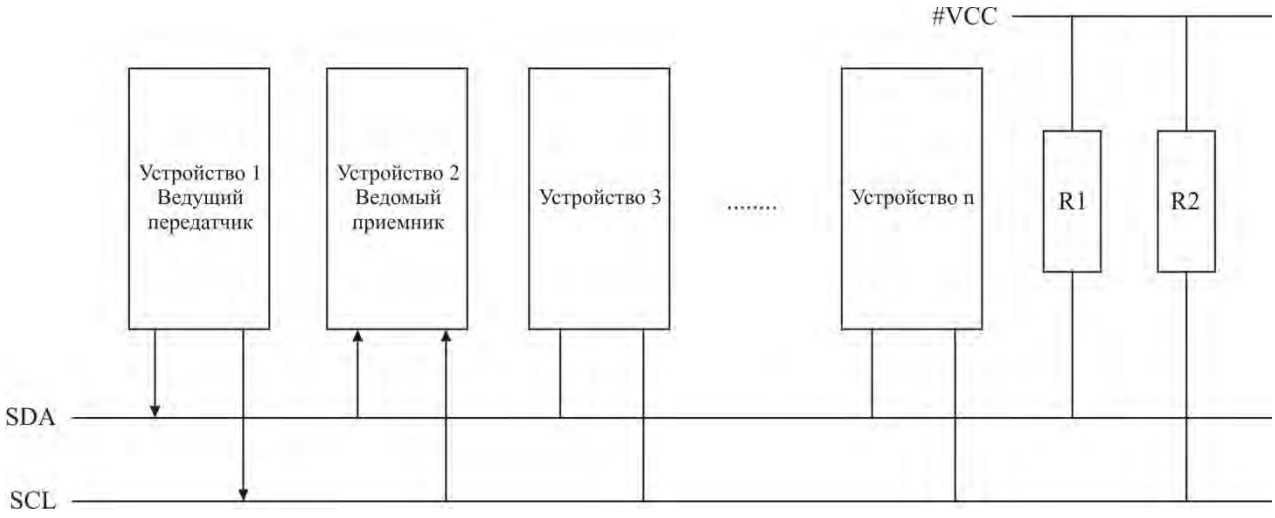


Рисунок 3.97 – Передача данных в режиме ведущего передатчика

Для разрешения работы двухпроводного последовательного интерфейса необходимо установить бит TWEN. Запись логической единицы в TWSTA инициирует передачу условия СТАРТ, а запись логической единицы в TWINT приводит к сбросу флага TWINT. После записи данного значения TWI тестирует двухпроводную последовательную шину и генерирует условие СТАРТ, как только шина освобождается. После передачи условия СТАРТ флаг TWINT устанавливается аппаратно, а в регистр TWSR помещается код состояния $08 (см. таблицу 3.88). Для перехода в режим ведущего передатчика необходимо передать адресный пакет SLA+W. Это осуществляется путем записи значения SLA+W в регистр TWDR. Затем следует сбросить флаг TWINT (путем записи логической единицы в соответствующий бит) для продолжения сеанса связи. Это 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 SLA+W и приема бита подтверждения флаг TWINT снова устанавливается, а в регистр TWSR помещается один из нескольких возможных кодов состояния. В ведущем режиме код состояния может быть равен $18, $20 или $38. Для каждого из этих кодов состояний необходимо выполнить соответствующие действия, которые подробно описаны в таблице 3.88.

После успешной передачи SLA+W должен быть передан пакет данных. Его передача инициируется записью байта данных в регистр TWDR. Запись в TWDR должна производиться только тогда, когда флаг TWINT установлен. В противном случае доступ блокируется и в регистре TWCR устанавливается флаг конфликтной ситуации (TWWC). После обновления TWDR необходимо сбросить флаг TWINT (путем записи логической единицы в соответствующий бит) для продолжения сеанса связи. Это выполняется путем записи следующего значения в TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	0	x	1	0	x

Данная последовательность операций 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 в 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 ПОВТОРНЫЙ СТАРТ (состояние $10) двухпроводной последовательный интерфейс может обращаться к тому же ведомому устройству или же к новому, при этом не требуется передача условия СТОП. Таким образом, ПОВТОРНЫЙ СТАРТ позволяет ведущему устройству переключаться между ведомыми устройствами, а также между режимами ведущий передатчик и ведущий приемник без потери управления шиной.

Таблица 3.88 – Коды состояния для режима ведущего передатчика

Код состояния (TWSR)	Состояние шины TWI и аппаратного обеспечения интерфейса TWI	Отклик программы						Следующее действие, выполняемое аппаратным обеспечением
		в/из TWDR	в TWCR					
			STA	STO	TWINT	TWEA		
1	2	3	4	5	6	7		8
$08	Передано условие СТАРТ	Загрузка SLA+W	0	0	1	x		SLA+W будет передан; ACK или NACK будет получен
$10	Передано условие ПОВТОРНЫЙ СТАРТ	Загрузка SLA+W или	0	0	1	x		SLA+W будет передан; ACK или NACK будет получен
		Загрузка SLA+R	0	0	1	x		SLA+R будет передан; Логика переключится в режим ведущего приемника
$18	Передан SLA+W, получен ACK	Загрузка байта данных или	0	0	1	x		Байт данных будет передан, и ACK или NACK будет получен
		Нет операции над TWDR или	1	0	1	x		ПОВТОРНЫЙ СТАРТ будет передан
		Нет операции над TWDR или	0	1	1	x		Условие СТОП будет передано, и флаг TWSTO будет сброшен
		Нет операции над TWDR	1	1	1	x		Будет передано условие СТАРТ, следующее за условием СТОП, и флаг TWSTO будет сброшен
$20	Передан SLA+W, получен NACK	Загрузка байта данных или	0	0	1	x		Байт данных будет передан, и ACK или NACK будет получен
		Нет операции над TWDR или	1	0	1	x		ПОВТОРНЫЙ СТАРТ будет передан
		Нет операции над TWDR или	0	1	1	x		Условие СТОП будет передано, и флаг TWSTO будет сброшен
		Нет операции над TWDR	1	1	1	x		Будет передано условие СТАРТ, следующее за условием СТОП, и флаг TWSTO будет сброшен
$28	Передан байт данных, получен ACK	Загрузка байта данных или	0	0	1	x		Байт данных будет передан, и ACK или NACK будет получен
		Нет операции над TWDR или	1	0	1	x		ПОВТОРНЫЙ СТАРТ будет передан
		Нет операции над TWDR или	0	1	1	x		Условие СТОП будет передано, и флаг TWSTO будет сброшен
		Нет операции над TWDR	1	1	1	x		Будет передано условие СТАРТ, следующее за условием СТОП, и флаг TWSTO будет сброшен

Продолжение таблицы 3.88

1	2	3	4	5	6	7	8
$30	Передан байт данных, получен NACK	Загрузка байта данных или	0	0	1	x	Байт данных будет передан, и ACK или NACK будет получен
		Нет операции над TWDR или	1	0	1	x	ПОВТОРНЫЙ СТАРТ будет передан
		Нет операции над TWDR или	0	1	1	x	Условие СТОП будет передано, и флаг TWSTO будет сброшен
		Нет операции над TWDR	1	1	1	x	Будет передано условие СТАРТ, следующее за условием СТОП, и флаг TWSTO будет сброшен
$38	Потерян приоритет в SLA+W или байтах данных	Нет операции над TWDR или	0	0	1	x	Шина TWI освободится, устройство переключится в неадресованный ведомый режим
		Нет операции над TWDR	1	0	1	x	Условие СТАРТ будет передано, когда шина освободится

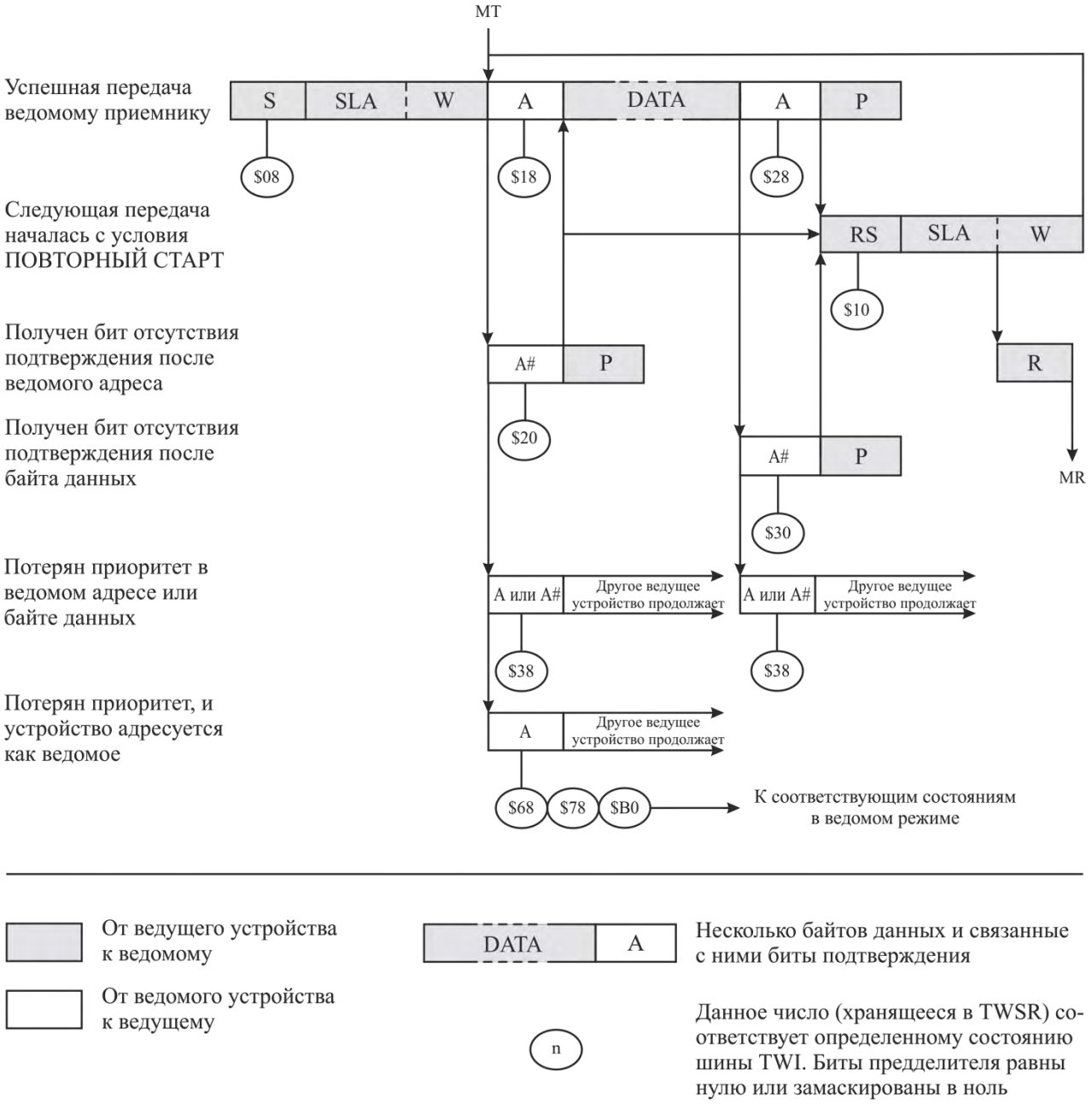


Рисунок 3.98 – Форматы и состояния в режиме ведущего передатчика

Режим ведущего приемника

В режиме ведущего приемника выполняется прием ряда байтов данных от ведомого передатчика (см. рисунок 3.99). Для входа в ведущий режим необходимо передать условие СТАРТ. Формат следующего адресного пакета определяет, в какой режим следует перейти: ведущий передатчик или ведущий приемник. Если передается SLA+W, то вводится режим MT (ведущий передатчик), а если SLA+R, то вводится режим MR (ведущий приемник). В данном подпункте предполагается, что для всех кодов состояний, упоминаемых здесь, биты предделителя равны нулю либо замаскированы.

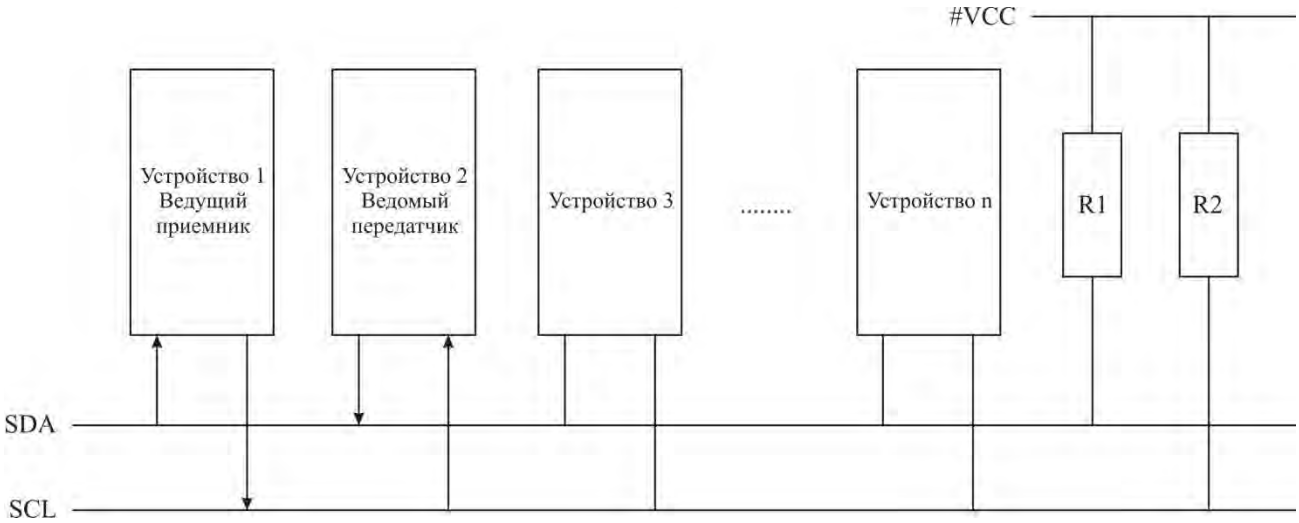


Рисунок 3.99 – Передача данных в режиме ведущего приемника

Передача условия СТАРТ инициируется путем записи в регистр 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 единицы в TWSTA инициирует передачу условия СТАРТ, а запись логической единицы в TWINT приводит к сбросу флага TWINT. После записи данного значения TWI тестирует двухпроводную последовательную шину и генерирует условие СТАРТ, как только шина освобождается. После передачи условия СТАРТ флаг TWINT устанавливается аппаратно, а в регистр TWSR помещается код состояния $08 (см. таблицу 3.89). Для перехода в режим ведущего приемника необходимо передать адресный пакет SLA+R. Это осуществляется путем записи значения SLA+R в регистр TWDR. Затем следует сбросить флаг TWINT (путем записи логической единицы в соответствующий бит) для продолжения сеанса связи. Это 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 SLA+R и приема бита подтверждения флаг TWINT снова устанавливается, а в регистр TWSR помещается один из нескольких возможных кодов состояния. В ведущем режиме код состояния может быть равен $38, $40 или $48. Для каждого из этих кодов состояний необходимо выполнить соответствующие действия, которые подробно описаны в таблице 3.89. Полученные данные могут быть считаны из регистра TWDR, когда флаг TWINT установится аппаратно. Данная последовательность операций повторяется до

тех пор, пока не будет получен последний байт. После приема последнего байта ведущий приемник должен информировать ведомого передатчика с помощью подачи сигнала об отсутствии подтверждения (NACK) после последнего полученного байта данных. Передача завершается генерацией условия СТОП или ПОВТОРНЫЙ СТАРТ. Условие СТОП генерируется путем записи следующего значения в 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 ПОВТОРНЫЙ СТАРТ (состояние $10) двухпроводной последовательный интерфейс может обращаться к тому же ведомому устройству или же к новому, при этом не требуется передача условия СТОП. Таким образом, ПОВТОРНЫЙ СТАРТ позволяет ведущему устройству переключаться между ведомыми устройствами, а также между режимами ведущий передатчик и ведущий приемник без потери управления шиной.

Таблица 3.89 – Коды состояния для режима ведущего приемника

Код состояния (TWSR)	Состояние шины TWI и аппаратного обеспечения интерфейса TWI	Отклик программы					Следующее действие, выполняемое аппаратным обеспечением
		в / из TWDR	в TWCR				
			STA	STO	TWINT	TWEA	
1	2	3	4	5	6	7	8
$08	Передано условие СТАРТ	Загрузка SLA+R	0	0	1	x	SLA+R будет передан; ACK или NACK будет получен
$10	Передано условие ПОВТОРНЫЙ СТАРТ	Загрузка SLA+R или	0	0	1	x	SLA+R будет передан; ACK или NACK будет получен
		Загрузка SLA+W	0	0	1	x	SLA+W будет передан; Логика переключится в режим ведущего передатчика
$38	Потерян приоритет в SLA+R или бите NACK	Нет операции над TWDR или	0	0	1	x	Шина TWI освободится, устройство переключиться в неадресованный ведомый режим
		Нет операции над TWDR	1	0	1	x	Условие СТАРТ будет передано, когда шина освободится
$40	Передан SLA+R, принят ACK	Нет операции над TWDR или	0	0	1	0	Байт данных будет принят, и будет возвращен NACK
		Нет операции над TWDR	0	0	1	1	Байт данных будет принят, и будет возвращен ACK
$48	Передан SLA+R, принят NACK	Нет операции над TWDR или	1	0	1	x	ПОВТОРНЫЙ СТАРТ будет передан
		Нет операции над TWDR или	0	1	1	x	Условие СТОП будет передано, и флаг TWSTO будет сброшен
		Нет операции над TWDR	1	1	1	x	Будет передано условие СТАРТ, следующее за условием СТОП, и флаг TWSTO будет сброшен

Продолжение таблицы 3.89

1	2	3	4	5	6	7	8
$50	Принят байт данных, возвращен ACK	Чтение байта данных или	0	0	1	0	Байт данных будет принят, и будет возвращен NACK
		Чтение байта данных	0	0	1	1	Байт данных будет принят, и будет возвращен ACK
$58	Принят байт данных, возвращен NACK	Чтение байта данных или	1	0	1	x	ПОВТОРНЫЙ СТАРТ будет передан
		Чтение байта данных или	0	1	1	x	Условие СТОП будет передано, и флаг TWSTO будет сброшен
		Чтение байта данных	1	1	1	x	Будет передано условие СТАРТ, следующее за условием СТОП, и флаг TWSTO будет сброшен

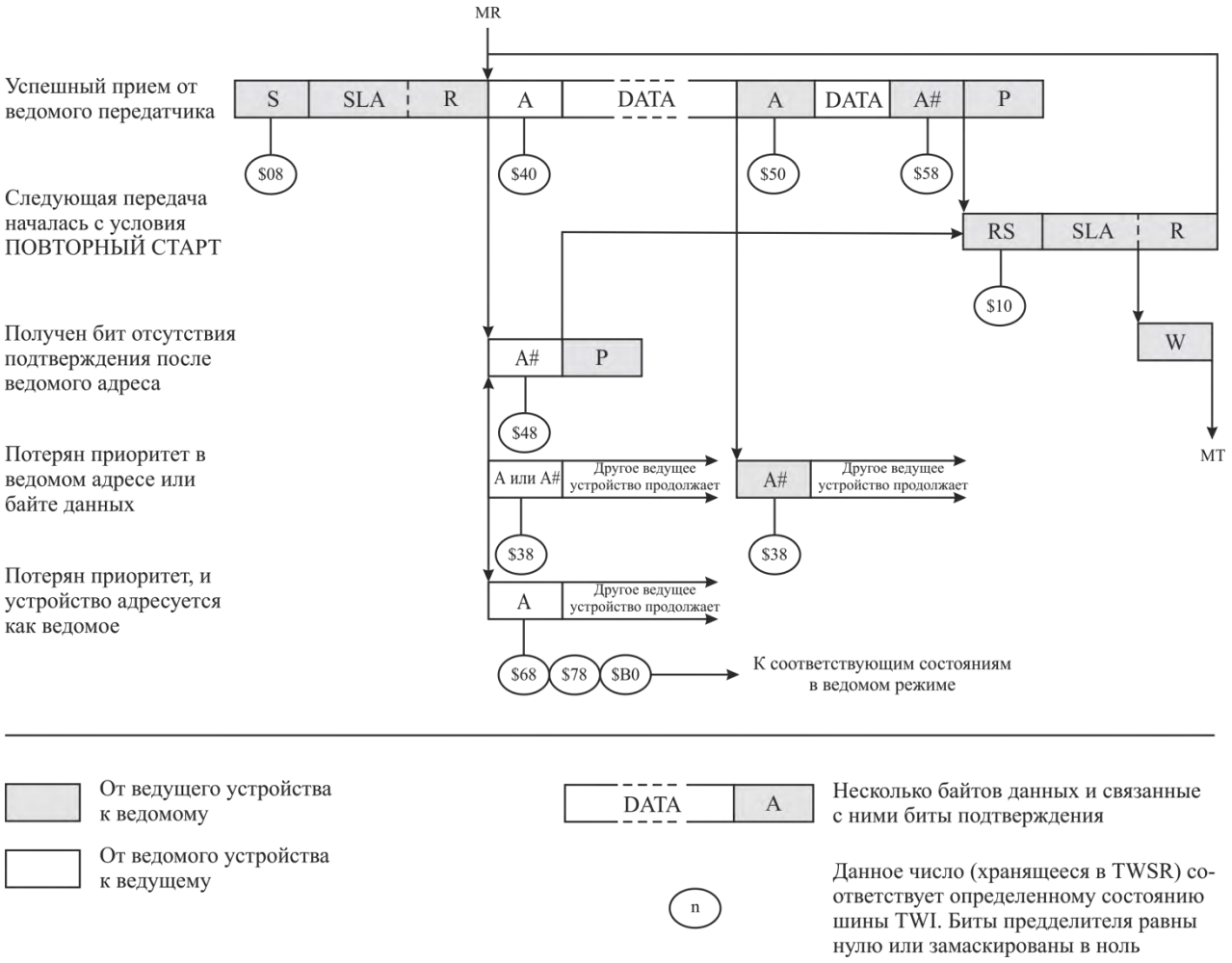


Рисунок 3.100 – Форматы и состояния в режиме ведущего приемника

Режим ведомого приемника

В режиме ведомого приемника выполняется прием ряда байтов данных от ведущего передатчика (см. рисунок 3.101). В данном подпункте предполагается, что для всех кодов состояний, упоминаемых здесь, биты предделителя равны нулю либо замаскированы.

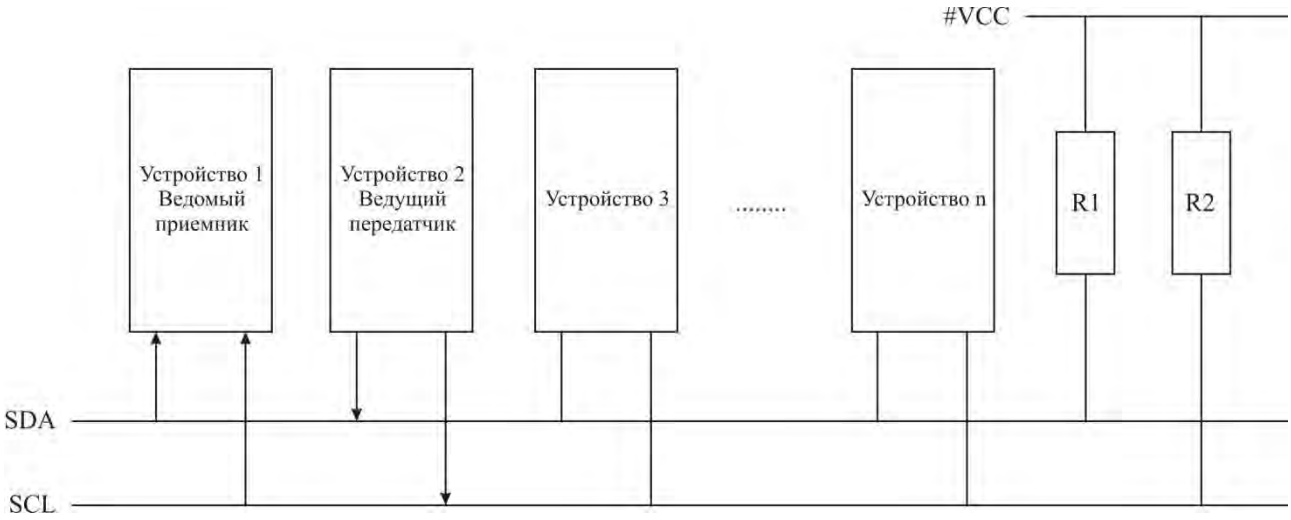


Рисунок 3.101 – Передача данных в режиме ведомого приемника

Для инициации режима ведомого приемника содержимое регистров TWAR и TWCR должно выглядеть следующим образом:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Значение	Собственный адрес устройства							

Старшие семь битов представляют собой адрес, на который будет реагировать интерфейс TWI, когда к нему обращается ведущее устройство. Если младший значащий разряд установлен, то TWI будет отзываться на адрес общего вызова ($00). В противном случае он будет игнорировать 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 единицу в TWEN. Для разрешения подтверждения собственного ведомого адреса устройства или адреса общего вызова в бит TWEA должна быть записана логическая единица. Битам TWSTA и TWSTO следует присвоить нулевое значение.

После инициализации TWAR и TWCR схема TWI ожидает обращения к своему собственному ведомому адресу (или, если разрешено, к адресу общего вызова), а вслед за этим ожидает получения бита направления данных. Если бит направления равен нулю (запись), то TWI будет работать в режиме ведомого приемника, в противном случае вводится режим ведомого передатчика. После приема собственного адреса и бита записи устанавливается флаг TWINT, а из регистра TWSR может быть считан действительный код состояния. Код состояния используется, чтобы определить, какие действия следует выполнить программе. В таблице 3.90 подробно описаны соответствующие действия для каждого из возможных значений кода состояния. Режим ведомого приемника также вводится, если происходит потеря приоритета, когда TWI работает в ведущем режиме (см. состояния $68 и $78).

Если сбросить бит TWEA во время передачи, то TWI возвратит NACK («1», нет подтверждения) на линию SDA после следующего принятого байта данных. Это свойство можно использовать для оповещения о том, что ведомое устройство больше не может принимать байты. Пока бит TWEA равняется нулю, TWI не подтверждает свой собственный ведомый адрес. Однако шина TWI продолжает оставаться под наблюдением, и распознавание адреса может возобновиться в любой момент с помощью установки бита TWEA. Это означает, что бит TWEA можно применять для временного отключения TWI от двухпроводной последовательной шины.

Во всех режимах сна, за исключением режима холостого хода, синхронизация TWI отключается. Если бит TWEA установлен, то интерфейс может продолжить подтверждение приема своего собственного ведомого адреса или адреса общего вызова, используя сигнал синхронизации шины TWI в качестве тактового источника. Затем микроконтроллер выходит из режима сна, при этом TWI удерживает линию SCL в низком уровне в процессе пробуждения и до сброса флага TWINT (путем записи логической единицы в соответствующий бит). Далее прием данных выполняется обычным образом при обычной системной синхронизации. Учтите, что если для микроконтроллера выбрано большое время запуска, то линия SCL может оказаться удерживаемой в низком уровне долгое время, блокируя другие сеансы передачи данных.

Обратите внимание, что после пробуждения регистр данных TWI (TWDR) не отражает последний байт, присутствовавший на шине во время выхода из указанных выше режимов сна.

Таблица 3.90 – Коды состояния для режима ведомого приемника

Код состояния (TWSR)	Состояние шины TWI и аппаратного обеспечения интерфейса TWI	Отклик программы					Следующее действие, выполняемое аппаратным обеспечением
		в / из TWDR	в TWCR				
			STA	STO	TWINT	TWEA	
1	2	3	4	5	6	7	8
$60	Получен собственный SLA+W; возвращен ACK	Нет операции над TWDR или Нет операции над TWDR	x x	0 0	1 1	0 1	Байт данных будет принят, и будет возвращен NACK Байт данных будет принят, и будет возвращен ACK
$68	Потерян приоритет в SLA+R/W в качестве ведущего; получен собственный SLA+W; возвращен ACK	Нет операции над TWDR или Нет операции над TWDR	x x	0 0	1 1	0 1	Байт данных будет принят, и будет возвращен NACK Байт данных будет принят, и будет возвращен ACK
$70	Получен адрес общего вызова; возвращен ACK	Нет операции над TWDR или Нет операции над TWDR	x x	0 0	1 1	0 1	Байт данных будет принят, и будет возвращен NACK Байт данных будет принят, и будет возвращен ACK
$78	Потерян приоритет в SLA+R/W в качестве ведущего; получен адрес общего вызова; возвращен ACK	Нет операции над TWDR или Нет операции над TWDR	x x	0 0	1 1	0 1	Байт данных будет принят, и будет возвращен NACK Байт данных будет принят, и будет возвращен ACK
$80	Предварительно адресован собственным SLA+W; данные получены; возвращен ACK	Чтение байта данных или Чтение байта данных	x x	0 0	1 1	0 1	Байт данных будет принят, и будет возвращен NACK Байт данных будет принят, и будет возвращен ACK

Продолжение таблицы 3.90

1	2	3	4	5	6	7	8
$88	Предварительно адресован собственным SLA+W; данные получены; возвращен NACK	Чтение байта данных или	0	0	1	0	Переключение в неадресованный ведомый режим; нет распознавания собственного SLA или GCA
		Чтение байта данных или	0	0	1	1	Переключение в неадресованный ведомый режим; собственный SLA распознается; GCA распознается, если TWGCE = 1
		Чтение байта данных или	1	0	1	0	Переключение в неадресованный ведомый режим; нет распознавания собственного SLA или GCA; условие СТАРТ будет передано, когда шина освободится
		Чтение байта данных	1	0	1	1	Переключение в неадресованный ведомый режим; собственный SLA распознается; GCA распознается, если TWGCE = 1; условие СТАРТ будет передано, когда шина освободится
$90	Предварительно адресован общим вызовом; данные получены; возвращен ACK	Чтение байта данных или	x	0	1	0	Байт данных будет принят, и будет возвращен NACK
		Чтение байта данных	x	0	1	1	Байт данных будет принят, и будет возвращен ACK
$98	Предварительно адресован общим вызовом; данные получены; возвращен NACK	Чтение байта данных или	0	0	1	0	Переключение в неадресованный ведомый режим; нет распознавания собственного SLA или GCA
		Чтение байта данных или	0	0	1	1	Переключение в неадресованный ведомый режим; собственный SLA распознается; GCA распознается, если TWGCE = 1
		Чтение байта данных или	1	0	1	0	Переключение в неадресованный ведомый режим; нет распознавания собственного SLA или GCA; условие СТАРТ будет передано, когда шина освободится
		Чтение байта данных	1	0	1	1	Переключение в неадресованный ведомый режим; собственный SLA распознается; GCA распознается, если TWGCE = 1; условие СТАРТ будет передано, когда шина освободится
$A0	Получено условие ОСТАНОВ или ПОВТОРНЫЙ СТАРТ во время адресации в качестве ведомого	Нет операции	0	0	1	0	Переключение в неадресованный ведомый режим; нет распознавания собственного SLA или GCA
			0	0	1	1	Переключение в неадресованный ведомый режим; собственный SLA распознается; GCA распознается, если TWGCE = 1
			1	0	1	0	Переключение в неадресованный ведомый режим; нет распознавания собственного SLA или GCA; условие СТАРТ будет передано, когда шина освободится
			1	0	1	1	Переключение в неадресованный ведомый режим; собственный SLA распознается; GCA распознается, если TWGCE = 1; условие СТАРТ будет передано, когда шина освободится

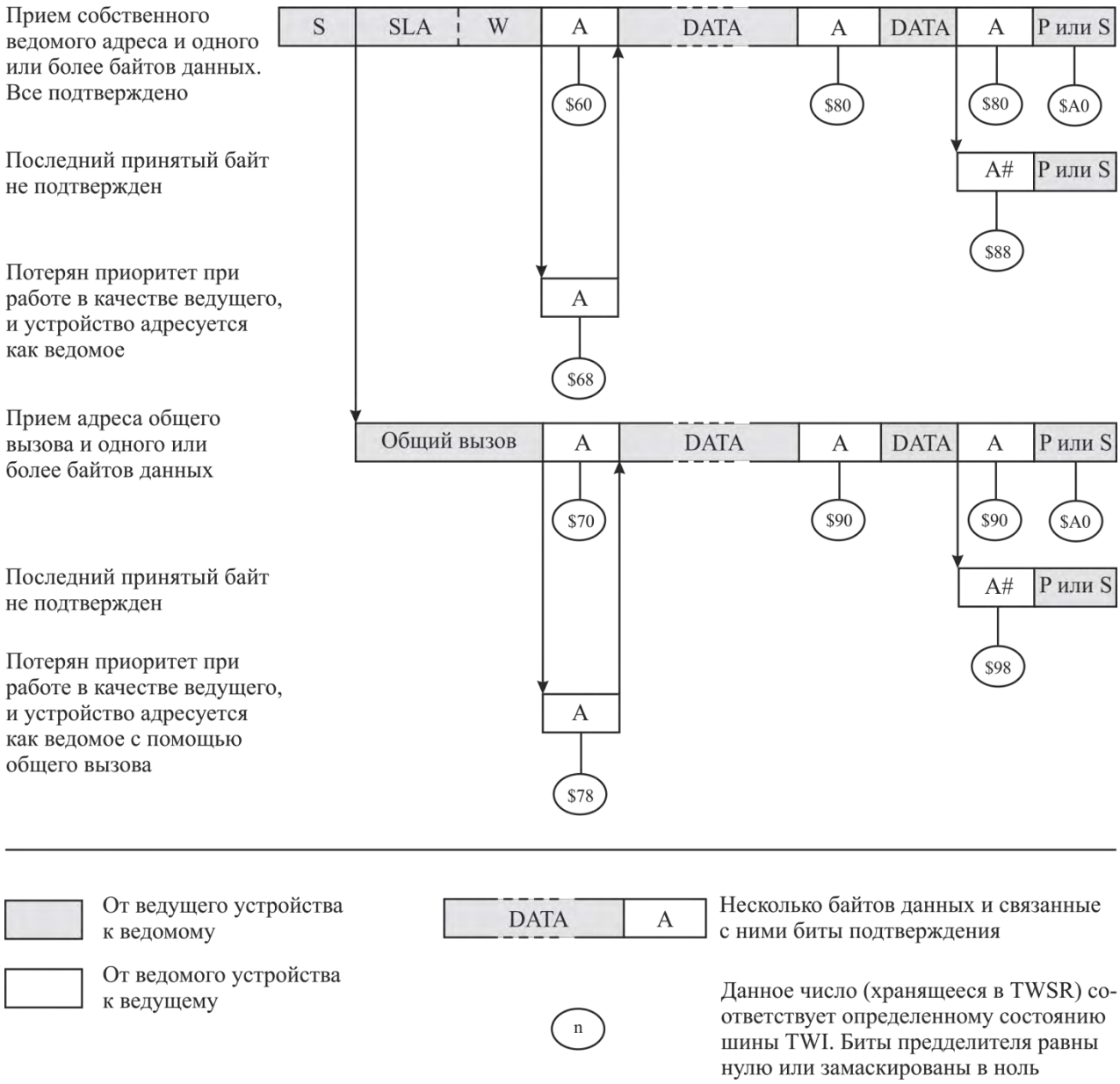


Рисунок 3.102 – Форматы и состояния в режиме ведомого приемника

Режим ведомого передатчика

В режиме ведомого передатчика выполняется передача ряда байтов данных ведущему приемнику (см. рисунок 3.103). В данном подпункте предполагается, что для всех кодов состояний, упоминаемых здесь, биты предделителя равны нулю либо замаскированы в ноль.

Для инициации режима ведомого передатчика содержимое регистров TWAR и TWCR должно выглядеть следующим образом:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Значение	Собственный адрес устройства							

Старшие семь битов представляют собой адрес, на который будет реагировать интерфейс TWI, когда к нему обращается ведущее устройство. Если младший значащий разряд установлен, то TWI будет отзываться на адрес общего вызова ($00). В противном случае он будет игнорировать адрес общего вызова.

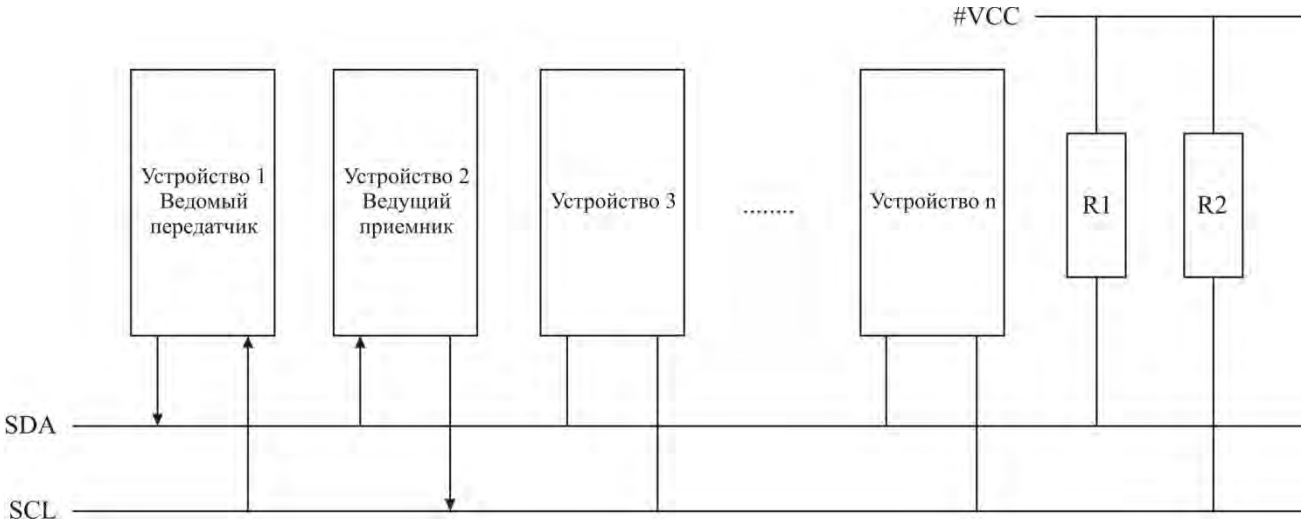


Рисунок 3.103 – Передача данных в режиме ведомого передатчика

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	0	1	0	0	0	1	0	x

Для разрешения работы TWI необходимо записать логическую единицу в TWEN. Для разрешения подтверждения собственного ведомого адреса устройства или адреса общего вызова в бит TWEA должна быть записана логическая единица. Битам TWSTA и TWSTO следует присвоить нулевое значение.

После инициализации TWAR и TWCR схема TWI ожидает обращения к своему собственному адресу (или, если разрешено, к адресу общего вызова), а вслед за этим ожидает получения бита направления данных. Если бит направления равен логической единице (чтение), то TWI будет работать в режиме ведомого передатчика, в противном случае вводится режим ведомого приемника. После приема собственного адреса и бита записи устанавливается флаг TWINT, а из регистра TWSR может быть считан действительный код состояния. Код состояния используется, чтобы определить, какие действия следует выполнить программе. В таблице 3.91 подробно описаны соответствующие действия для каждого из возможных значений кода состояния (см. также рисунок 3.104). Режим ведомого передатчика также вводится, если происходит потеря приоритета, когда TWI работает в ведущем режиме (см. состояние $B0).

Если сбросить бит TWEA во время передачи, то TWI передаст последний байт. Будет введено состояние $C0 либо состояние $C8. Это зависит от того, какой бит передает ведущий приемник после завершающего байта – NACK или ACK. TWI переходит в неадресованный ведомый режим и далее игнорирует ведущее устройство, если оно продолжает передачу. Таким образом, ведущий приемник принимает все «1» как последовательные данные. Состояние $C8 вводится, если ведущее устройство требует передачи дополнительных байтов данных (путем передачи бита ACK), даже если ведомое устройство передало последний байт (TWEA равен нулю и ожидается прием NACK от ведущего устройства).

Пока бит TWEA равняется нулю, TWI не реагирует на свой собственный адрес. Однако шина TWI продолжает оставаться под наблюдением, и распознавание адреса может возобновиться в любой момент с помощью установки бита TWEA. Это означает, что бит TWEA можно применять для временного отключения TWI от двухпроводной последовательной шины.

Во всех режимах сна, за исключением режима холостого хода, синхронизация TWI отключается. Если бит TWEA установлен, то интерфейс может продолжить подтверждение приема своего собственного адреса или адреса общего вызова, используя сигнал синхронизации шины TWI в качестве тактового источника. Затем микроконтроллер выходит из

режима сна, при этом TWI удерживает линию SCL в низком уровне в процессе пробуждения и до сброса флага TWINT (путем записи логической единицы в соответствующий бит).

Таблица 3.91 – Коды состояния для режима ведомого передатчика

Код состояния (TWSR)	Состояние шины TWI и аппаратного обеспечения интерфейса TWI	Отклик программы					Следующее действие, выполняемое аппаратным обеспечением
		в/из TWDR	в TWCR				
			STA	STO	TWINT	TWEA	
$A8	Получен собственный SLA+R; возвращен ACK	Загрузка байта данных или	x	0	1	0	Будет передан последний байт данных, и будет получен NACK
		Загрузка байта данных	x	0	1	1	Будет передан байт данных, и будет получен ACK
$B0	Потерян приоритет в SLA+R/W в качестве ведущего; получен собственный SLA+R; возвращен ACK	Загрузка байта данных или	x	0	1	0	Будет передан последний байт данных, и будет получен NACK
		Загрузка байта данных	x	0	1	1	Будет передан байт данных, и будет получен ACK
$B8	Передан байт данных из TWDR; получен ACK	Загрузка байта данных или	x	0	1	0	Будет передан последний байт данных, и будет получен NACK
		Загрузка байта данных	x	0	1	1	Будет передан байт данных, и будет получен ACK
$C0	Передан байт данных из TWDR; получен NACK	Нет операции над TWDR или	0	0	1	0	Переключение в неадресованный ведомый режим; нет распознавания собственного SLA или GCA
		Нет операции над TWDR или	0	0	1	1	Переключение в неадресованный ведомый режим; собственный SLA распознается; GCA распознается, если TWGCE = 1
		Нет операции над TWDR или	1	0	1	0	Переключение в неадресованный ведомый режим; нет распознавания собственного SLA или GCA; условие СТАРТ будет передано, когда шина освободится
		Нет операции над TWDR	1	0	1	1	Переключение в неадресованный ведомый режим; собственный SLA распознается; GCA распознается, если TWGCE = 1; условие СТАРТ будет передано, когда шина освободится
$C8	Передан последний байт данных из TWDR (TWEA = 0); получен ACK	Нет операции над TWDR или	0	0	1	0	Переключение в неадресованный ведомый режим; нет распознавания собственного SLA или GCA
		Нет операции над TWDR или	0	0	1	1	Переключение в неадресованный ведомый режим; собственный SLA распознается; GCA распознается, если TWGCE = 1
		Нет операции над TWDR или	1	0	1	0	Переключение в неадресованный ведомый режим; нет распознавания собственного SLA или GCA; условие СТАРТ будет передано, когда шина освободится
		Нет операции над TWDR	1	0	1	1	Переключение в неадресованный ведомый режим; собственный SLA распознается; GCA распознается, если TWGCE = 1; условие СТАРТ будет передано, когда шина освободится

Далее прием данных выполняется обычным образом при обычной системной синхронизации. Учтите, что если для микроконтроллера выбрано большое время запуска, то линия SCL может оказаться удерживаемой в низком уровне долгое время, блокируя другие сеансы передачи данных. Обратите внимание, что после пробуждения регистр данных TWI (TWDR) не отражает последний байт, присутствовавший на шине во время выхода из указанных выше режимов сна.

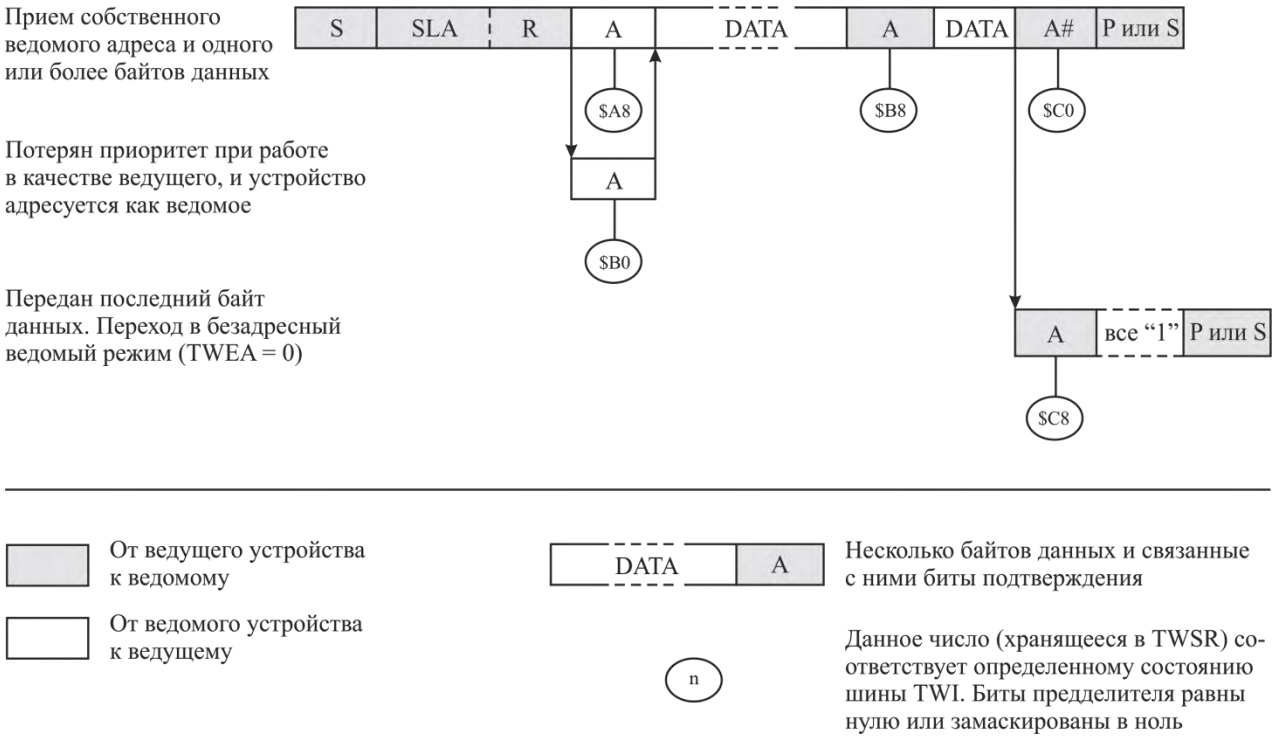


Рисунок 3.104 – Форматы и состояния в режиме ведомого передатчика

Неопределенные состояния

Существует два кода состояния, которые не соответствуют какому-либо определенному состоянию TWI (см. таблицу 3.92).

Таблица 3.92 – Неопределенные состояния

Код состояния (TWSR)	Состояние шины TWI и аппаратного обеспечения интерфейса TWI	Отклик программы						Следующее действие, выполняемое аппаратным обеспечением
		в / из TWDR	в TWCR					
			STA	STO	TWINT	TWEA		
$F8	Существенная информация о состоянии не доступна; TWINT = 0	Нет операции над TWDR	Нет операции над TWCR					Ожидание или возобновление текущей передачи
$00	Ошибка на шине из-за недопустимого условия СТАРТ или ОСТАНОВ	Нет операции над TWDR	0	1	1	x		Воздействие оказывается только на внутреннее аппаратное обеспечение, условие СТОП не передается на шину. Во всех случаях шина освобождается, и флаг TWSTO сбрасывается

Состояние $F8 показывает, что информация о статусе недоступна, поскольку флаг TWINT не установлен. Такое случается в промежутках между другими состояниями, и когда TWI не участвует в процессе последовательной передачи.

Состояние $00 показывает, что произошла ошибка на шине во время сеанса передачи. Это случается, когда условие СТАРТ или СТОП возникает в непредусмотренном месте в кадре, например во время последовательной передачи адресного байта, байта данных или бита подтверждения. Когда происходит ошибка на шине, флаг TWINT устанавливается. Для выхода из ошибочного состояния необходимо установить флаг TWSTO и сбросить флаг TWINT путем записи логической единицы в соответствующий бит. Это заставляет TWI перейти в неадресованный ведомый режим и сбросить флаг TWSTO (данная операция не оказывает влияния на другие биты в регистре TWCR). Линии SDA и SCL освобождаются, и условие СТОП не передается.

Комбинирование нескольких режимов TWI

В некоторых случаях необходимо объединить несколько режимов TWI для того, чтобы выполнить желаемое действие. В качестве примера можно рассматривать считывание информации из последовательной ЭПД. Как правило, такой сеанс передачи данных состоит из следующих этапов:

1 Инициация передачи данных.

2 Информирование ЭПД о том, какая ячейка памяти должна быть считана.

3 Процедура считывания.

4 Завершение передачи.

Следует обратить внимание, что данные передаются как от ведущего устройства к ведомому, так и наоборот. Ведущее устройство обязано сообщить ведомому, какая ячейка должна быть считана, требуя использования режима ведущего передатчика. Далее информация должна быть считана от ведомого устройства, предполагая использование режима ведущего приемника. Таким образом, направление передачи будет меняться. Ведущее устройство должно сохранять контроль над шиной в течение всех вышеперечисленных этапов, и данные этапы должны выполняться как неделимая операция. Если этот принцип нарушается в системе с несколькими ведущими устройствами, то другое ведущее устройство может изменить указатель данных в ЭПД между этапами 2 и 3, и тогда ведущее устройство будет считывать несоответствующую ячейку памяти. Изменение в направлении передачи информации осуществляется путем посылки условия ПОВТОРНЫЙ СТАРТ в период между передачей адресного байта и приемом данных. После ПОВТОРНОГО СТАРТА ведущее устройство продолжает контролировать шину. На рисунке 3.105 иллюстрируется процесс данного сеанса связи.

Рисунок 3.105 – Комбинирование нескольких режимов TWI для получения доступа к последовательной ЭПД

3.16.9 Системы с несколькими ведущими устройствами и арбитраж

Если несколько ведущих устройств подключено к одной и той же шине, то сеансы передачи могут инициироваться одновременно одним или несколькими ведущими устройствами. Стандарт TWI гарантирует, что подобные ситуации обрабатываются следующим образом: одному из ведущих устройств разрешается продолжить передачу, и при этом не происходит потери данных. Пример ситуации арбитража показан на рисунке 3.106, где два ведущих устройства пытаются передать информацию ведомому приемнику.

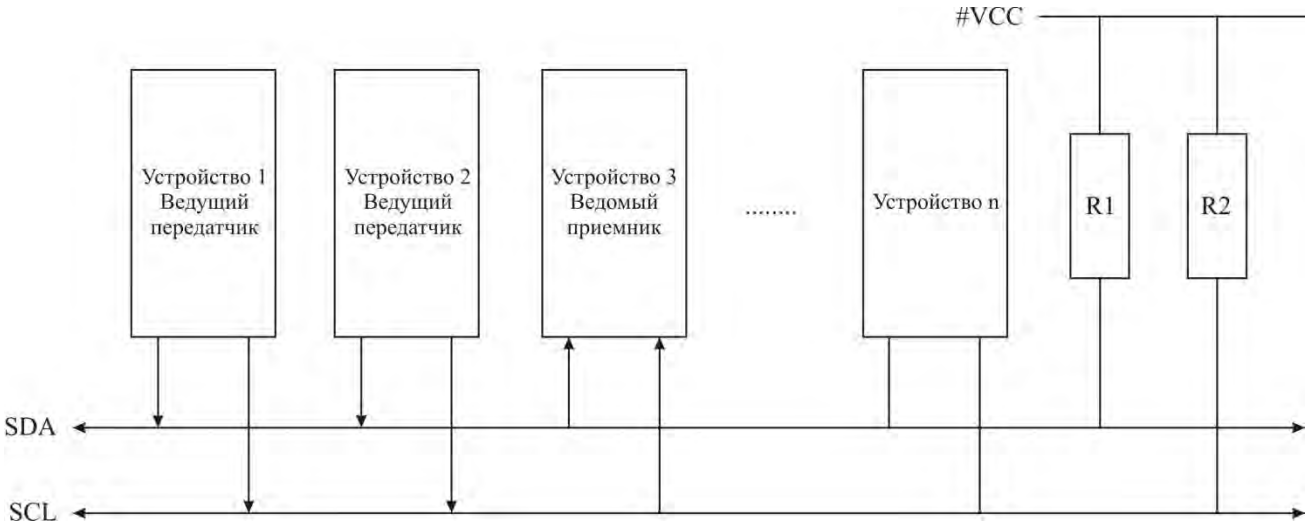


Рисунок 3.106 – Пример арбитража

В процессе арбитража может возникнуть несколько различных сценариев, как описано ниже:

- Два или более ведущих устройств выполняют идентичные сеансы передачи данных с одним и тем же ведомым устройством. В этом случае ни ведомое устройство, ни какое-либо из ведущих устройств не будет знать о конфликтной ситуации на шине.

- Два или более ведущих устройств с различающимися данными или битом направления имеют доступ к одному и тому же ведомому устройству. В этом случае арбитраж произойдет либо в бите ЧТЕНИЕ/ЗАПИСЬ, либо в битах данных. Ведущие устройства, пытающиеся передать логическую единицу на линию SDA, в то время как другое ведущее устройство передает логический ноль, потеряют приоритет. Проигравшие ведущие устройства перейдут в неадресованный ведомый режим или будут ждать, пока шина не освободится и не будет передано новое условие СТАРТ. Их поведение зависит от операции, которую выполняет программа.

- Два или более ведущих устройств получают доступ к разным ведомым устройствам. В этом случае арбитраж произойдет в битах SLA. Ведущие устройства, пытающиеся передать логическую единицу на линию SDA, в то время как другое ведущее устройство передает логический ноль, потеряют приоритет. Проигравшие ведущие устройства перейдут в ведомый режим, чтобы проверить, не к ним ли обращается ведущее устройство, выигравшее процесс арбитража. Если они адресованы выигравшим ведущим устройством, то они переключатся в режим ведомого приемника или ведомого передатчика, в зависимости от значения бита ЧТЕНИЕ/ЗАПИСЬ. Если они не адресованы, они перейдут в неадресованный ведомый режим или будут ждать, пока шина не освободится и не будет передано новое условие СТАРТ. Их поведение зависит от операции, которую выполняет программа.

Сказанное выше отражено на рисунке 3.107. Числа, записанные внутри окружностей, показывают возможные коды состояний.

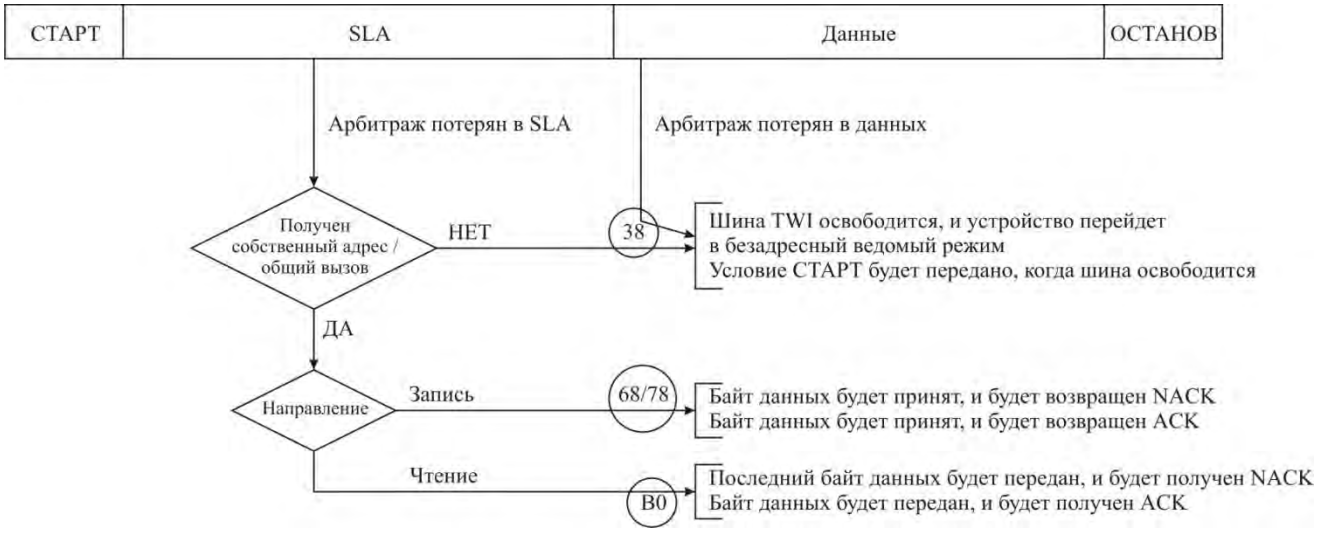


Рисунок 3.107 – Возможные коды состояния в процессе арбитража

3.17 Аналоговый компаратор

Аналоговый компаратор сравнивает уровни напряжений на положительном входе AIN0 и отрицательном входе AIN1. Когда напряжение на положительном входе AIN0 превышает напряжение на отрицательном входе AIN1, выход аналогового компаратора ACO принимает единичное состояние. Выход компаратора можно настроить на запуск функции входного захвата таймера/счетчика 1. Кроме того, компаратор может генерировать собственный запрос на обработку прерывания. Пользователь может выбрать режим запуска прерывания по нарастающему фронту, по спадающему фронту или по переключению фронта сигнала на выходе компаратора. Структурная схема компаратора и связанной с ним логики представлена на рисунке 3.108.

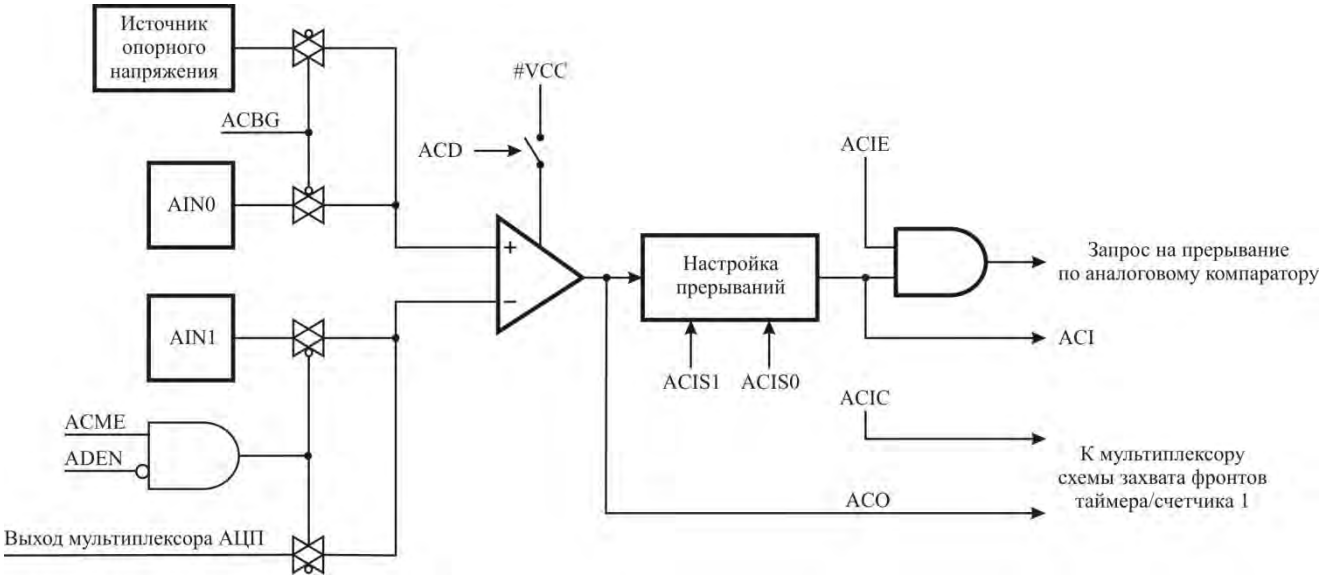


Рисунок 3.108 – Структурная схема аналогового компаратора

3.17.1 Регистр специальных функций ввода-вывода – SFIOR

Бит	7	6	5	4	3	2	1	0	
	TSM	-	-	-	ACME	PUD	PSR0	PSR321	SFIOR
Чтение/ Запись	Ч/З	Ч	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 3 – ACME: Включение мультиплексора аналогового компаратора

Когда в данный разряд записана логическая единица и АЦП выключен (бит ADEN в регистре ADCSRA равен нулю), мультиплексор АЦП выбирает отрицательный вход аналогового компаратора. Когда в данный разряд записан логический ноль, на отрицательный вход аналогового компаратора подается сигнал AIN1. Более подробное описание бита ACME приведено в 3.17.3 «Мультиплексированный вход аналогового компаратора».

3.17.2 Регистр управления и состояния аналогового компаратора – ACSR

Бит	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Чтение/ Запись	Ч/З	Ч/З	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	-	0	0	0	0	0	

Разряд 7 – ACD: Отключение аналогового компаратора

Запись логической единицы в разряд ACD приводит к снятию питания с аналогового компаратора. Данный разряд можно устанавливать в любой момент при необходимости отключить аналоговый компаратор. Это позволяет снизить энергопотребление в активном режиме и режиме холостого хода. Перед изменением бита ACD необходимо запретить прерывания по аналоговому компаратору путем сброса бита ACIE в регистре ACSR. В противном случае прерывание может возникнуть во время изменения бита ACD.

Разряд 6 – ACBG: Подключение источника опорного напряжения к аналоговому компаратору

Когда данный разряд установлен, источник опорного напряжения 1,23В подключается к положительному входу компаратора. Когда данный разряд сброшен, на положительный вход компаратора подается сигнал AIN0. См. также 3.5.8 «Встроенный источник опорного напряжения».

Разряд 5 – ACO: Выход аналогового компаратора

Выход аналогового компаратора синхронизирован и напрямую связан с битом ACO. Синхронизация вносит временную задержку на один или два такта.

Разряд 4 – ACI: Флаг прерывания аналогового компаратора

Данный разряд устанавливается аппаратно, когда событие на выходе компаратора запускает режим прерываний, заданный битами ACIS1 и ACIS0. Программа обработки прерываний аналогового компаратора выполняется, если установлены бит ACIE в регистре ACSR и бит I в регистре SREG. ACI сбрасывается аппаратно при переходе на соответствующий вектор обработки прерывания. Альтернативно, флаг ACI можно сбросить программно путем записи логической единицы в соответствующий разряд.

Разряд 3 – ACIE: Разрешение прерывания по аналоговому компаратору

Если в данный разряд записана логическая единица и установлен бит I в регистре статуса, то активируется прерывание по аналоговому компаратору. Запись логического нуля в бит ACIE приводит к блокировке 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 подключается к входному каскаду схемы захвата фронтов. Это заставляет компаратор использовать функции подавления шумов и настройки фронтов для прерывания по входному захвату таймера/счетчика 1. Когда бит ACIC сброшен, то связи между аналоговым компаратором и схемой захвата фронтов не существует. Чтобы побудить компаратор инициировать прерывание по входному захвату таймера/счетчика 1, необходимо установить бит TICIE1 в регистре масок прерываний таймеров (TIMSK).

Разряды 1, 0 – ACIS1, ACIS0: Выбор режима прерываний аналогового компаратора

Данные разряды определяют, какие события приводят к генерации запроса на прерывание аналогового компаратора. Варианты установок этих разрядов и их назначение представлены в таблице 3.93.

Таблица 3.93 – Установки разрядов ACIS1/ACIS0

ACIS1	ACIS0	Режим прерываний
0	0	Прерывание компаратора по выходному переключению
0	1	Зарезервировано
1	0	Прерывание компаратора по спадающему выходному фронту
1	1	Прерывание компаратора по нарастающему выходному фронту

Перед изменением битов ACIS1/ACIS0 необходимо запретить прерывания по аналоговому компаратору путем сброса бита ACIE в регистре ACSR. В противном случае прерывание может возникнуть во время изменения этих битов.

3.17.3 Мультиплексированный вход аналогового компаратора

Имеется возможность выбрать любой из выводов ADC7–0 для подключения к отрицательному входу аналогового компаратора. Для этого используется мультиплексор АЦП и, следовательно, в данном случае сам преобразователь должен быть отключен. Если установлен бит включения мультиплексора аналогового компаратора (бит ACME в регистре SFIOR) и АЦП выключен (бит ADEN в регистре ADCSRA равен нулю), то состояние разрядов MUX2–0 регистра ADMUX определяет, какой вывод микроконтроллера будет подключен к отрицательному входу аналогового компаратора (см. таблицу 3.94). Если ACME сброшен или установлен ADEN, то на отрицательный вход аналогового компаратора подается сигнал AIN1.

Таблица 3.94 – Мультиплексированный вход аналогового компаратора

ACME	ADEN	MUX2–0	Отрицательный 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

3.18 Аналого-цифровой преобразователь

3.18.1 Отличительные особенности

- 10-разрядная разрешающая способность.
- Интегральная нелинейность 0,5 младшего значащего разряда.
- Абсолютная погрешность ± 2 младших значащих разряда.
- Время преобразования от 13 до 260 мкс.
- Частота преобразования до 76,9 тысяч преобразований в секунду (до 15 тысяч преобразований в секунду при максимальном разрешении).
- Восемь мультиплексированных несимметричных входных каналов.
- Семь дифференциальных входных каналов.
- Два дифференциальных входных канала с дополнительным усилением 10х и 200х.
- Дополнительное левостороннее выравнивание для вывода результатов преобразования.
- Диапазон входного напряжения АЦП от 0 В до U_{CC}.
- Выборочный внутренний источник опорного напряжения на 2,56 В.
- Режим холостого хода или режим одиночного преобразования.
- Прерывание по завершении преобразования АЦП.
- Механизм подавления шумов в режиме сна.

Микроконтроллер содержит 10-разрядный АЦП последовательного приближения. На входе модуля АЦП имеется 8-канальный аналоговый мультиплексор, представляющий в распоряжение пользователя восемь каналов с несимметричными входами, которые связаны с выходами порта F. Общий вывод входных сигналов должен иметь потенциал 0В (т.е. связан с ∩0V).

АЦП также поддерживает 16 входных комбинаций для дифференциального напряжения. Два дифференциальных входа (ADC1, ADC0 и ADC3, ADC2) содержат каскад со ступенчатым программируемым усилением (перед выполнением преобразования): 0 дБ (1х), 20 дБ (10х) или 46 дБ (200х). Семь дифференциальных аналоговых входных каналов используют общий отрицательный вывод (ADC1), а все остальные входы АЦП выполняют функцию положительных входов. При коэффициентах усиления 1х и 10х действительная разрушающая способность АЦП по этим каналам составляет 8 разрядов, а при коэффициенте усиления 200х –7 разрядов.

АЦП содержит схему выборки и хранения, которая поддерживает на постоянном уровне напряжение на входе АЦП во время преобразования. Структурная схема АЦП представлена на рисунке 3.109.

АЦП имеет отдельный вывод питания ∩VCC (аналоговое питание). Напряжение ∩VCC не должно отличаться более чем на ± 0,3 В от #VCC. О том, как подключать данный вывод, смотрите ниже в 3.18.6 «Подавитель шумов АЦП».

Внутри кристалла расположены встроенные источники опорного напряжения номиналом 2,56 В или номиналом, равным ∩VCC. Поскольку эти источники опорного напряжения соединяются с выводом AREF микроконтроллера, при их использовании к выводу AREF желательно подключить внешний фильтрующий конденсатор для повышения помехозащищенности.

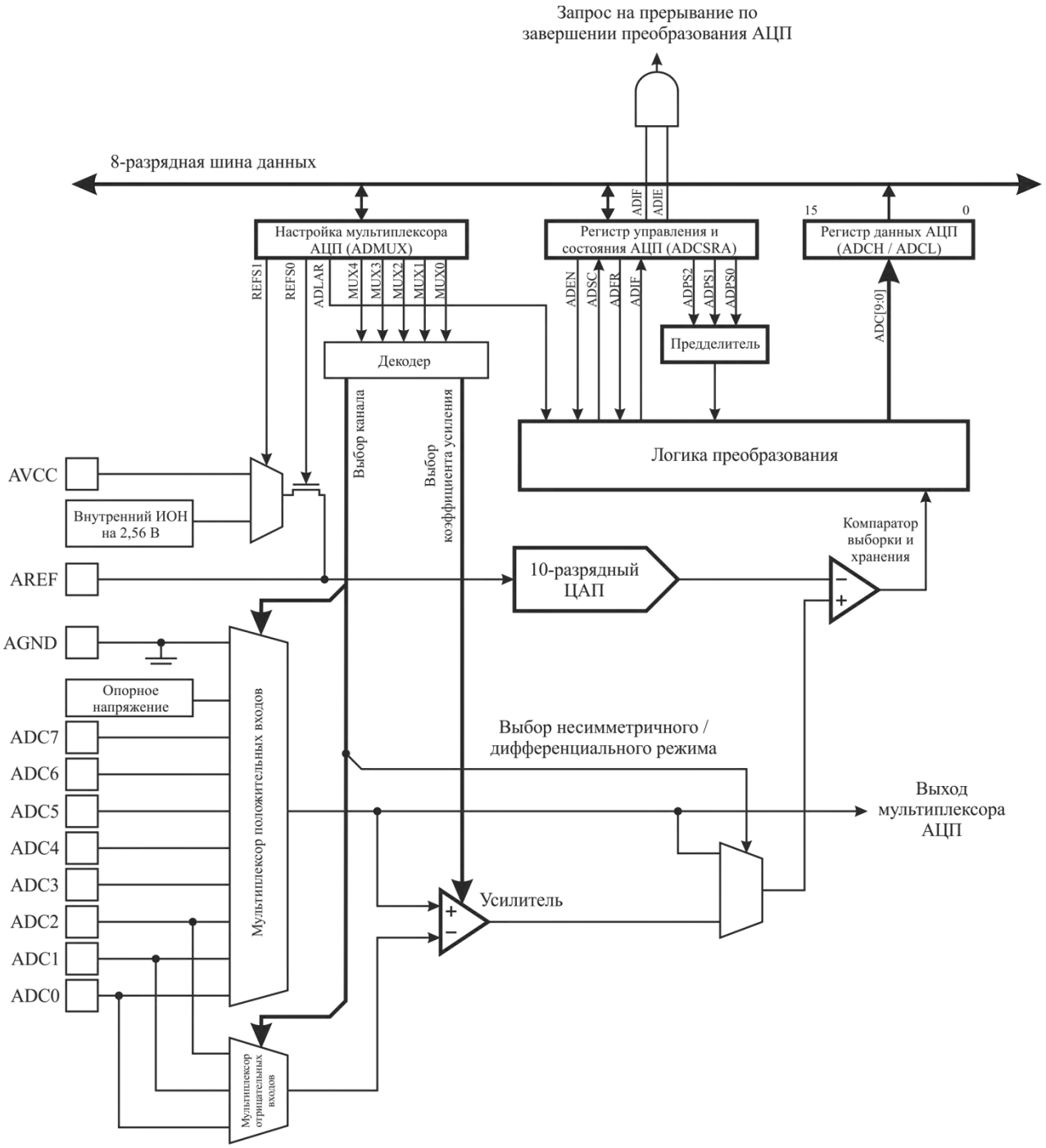


Рисунок 3.109 – Структурная схема АЦП

3.18.2 Принцип действия АЦП

АЦП преобразовывает входное аналоговое напряжение в 10-разрядный код методом последовательных приближений. Минимальное значение соответствует уровню ∩0V, а максимальное – уровню AREF минус один младший значащий разряд. К выводу AREF дополнительно может быть подключено напряжение ∩VCC или внутренний ИОН на 2,56 В путем записи соответствующих значений в биты REFSn в регистре 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 в биты MUX регистра ADMUX. В качестве несимметричного входа АЦП может

быть выбран любой из выводов ADC0 – ADC7, а также ∩0V и выход фиксированного источника опорного напряжения. Для выводов ADC0 – ADC7 предусмотрена возможность выбора положительных и отрицательных входов к дифференциальному усилителю.

Если выбраны дифференциальные каналы, то дифференциальный каскад будет усиливать разность напряжений между выбранной парой входов на заданный коэффициент усиления. Усиленное таким образом значение поступает на аналоговый вход АЦП. Если используются однополярные каналы, то каскад усиления игнорируется.

Работа АЦП разрешается путем установки бита ADEN в регистре ADCSRA. Выбор ИОН и входного канала невозможно выполнить до установки ADEN. Если ADEN = 0, то АЦП не потребляет ток, поэтому рекомендуется предварительно отключать АЦП при переходе в энергосберегающие режимы сна.

АЦП формирует 10-разрядный результат, который помещается в пару регистров данных АЦП – ADCH и ADCL. По умолчанию результат преобразования размещается в младших 10 разрядах 16-разрядного слова (выравнивание вправо), но по желанию его можно разместить в старших 10 разрядах (выравнивание влево) путем установки бита ADLAR в регистре ADMUX.

Если результат представлен с выравниванием влево и требуется точность не более 8 битов, то достаточно выполнить считывание только регистра ADCH. В противном случае необходимо сначала считать данные регистра ADCL, а затем ADCH, чтобы удостовериться, что содержимое регистров данных является результатом одного и того же преобразования. Как только произойдет считывание ADCL, доступ к регистрам данных для АЦП будет блокирован. Это означает, что если считаны данные регистра ADCL и преобразование завершается до выполнения считывания ADCH, то ни один из регистров не обновляется, и результат преобразования теряется. После считывания ADCH доступ к регистрам ADCH и ADCL со стороны АЦП снова разрешается.

АЦП генерирует собственный запрос на прерывание по завершении преобразования. Если доступ к регистрам данных для АЦП запрещается в момент времени между считыванием ADCH и ADCL, то прерывание возникнет, даже если результат преобразования будет потерян.

3.18.3 Запуск преобразования

Режим одиночного преобразования запускается путем записи логической единицы в бит запуска преобразования АЦП (ADSC). Этот бит остается установленным, пока продолжается преобразование, и сбрасывается аппаратно, когда преобразование завершается. Если в процессе преобразования будет выбран другой канал данных, то АЦП сначала завершит текущее преобразование и только затем переключит канал.

В режиме непрерывного преобразования АЦП постоянно оцифровывает аналоговый сигнал и обновляет свой регистр данных. Режим непрерывного преобразования выбирается путем записи логической единицы в бит ADFR регистра ADCSRA. Первое преобразование должно инициироваться записью логической единицы в бит ADSC регистра ADCSRA. В данном режиме АЦП выполняет последовательные преобразования независимо от того, сброшен флаг прерывания АЦП (ADIF) или нет.

3.18.4 Предделитель АЦП и временные диаграммы преобразования

Если требуется максимальная разрешающая способность (10 разрядов), то частота на входе схемы последовательного приближения должна быть в диапазоне от 50 до 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 (см. рисунок 3.110), который формирует допустимую частоту свыше 100 кГц по отношению к частоте синхронизации ЦПУ. Коэффициент деления устанавливается с помощью битов ADPS в регистре ADCSRA. Предделитель

начинает счет с момента включения АЦП путем записи логической единицы в бит ADEN регистра ADCSRA. Предделитель продолжает работать, пока бит ADEN установлен, и сбрасывается, когда бит ADEN равен нулю.

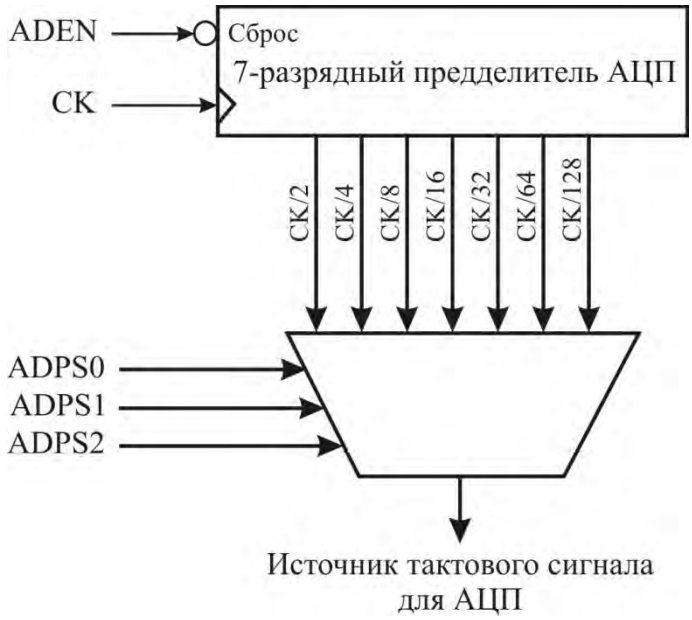


Рисунок 3.110 – Предделитель АЦП

При старте преобразования с несимметричным входом путем установки бита ADSC в регистре ADCSRA преобразование начинается со следующего нарастающего фронта тактового сигнала АЦП. См. ниже подпункт «Каналы дифференциального усиления» для изучения временных диаграмм дифференциального преобразования.

Нормальное преобразование требует 13 тактов синхронизации АЦП. Первое преобразование после включения АЦП (установка бита ADEN в регистре ADCSRA) требует 25 тактов синхронизации АЦП из-за необходимости инициализации аналоговой схемы.

Фактическая операция выборки и хранения занимает 1,5 такта АЦП после начала нормального преобразования и 13,5 тактов после запуска первого преобразования. По завершении преобразования результат помещается в регистры данных АЦП, и устанавливается флаг прерывания ADIF. В режиме одиночного преобразования одновременно сбрасывается бит ADSC. Бит ADSC можно снова установить программно, и тогда новое преобразование будет инициировано первым нарастающим фронтом тактового сигнала АЦП.

В режиме непрерывного преобразования новое преобразование начинается сразу по завершении предыдущего, пока бит ADSC остается в высоком состоянии. В таблице 3.95 приведена краткая информация о времени преобразования АЦП, на рисунках 3.111 – 3.113 – временные диаграммы работы АЦП.

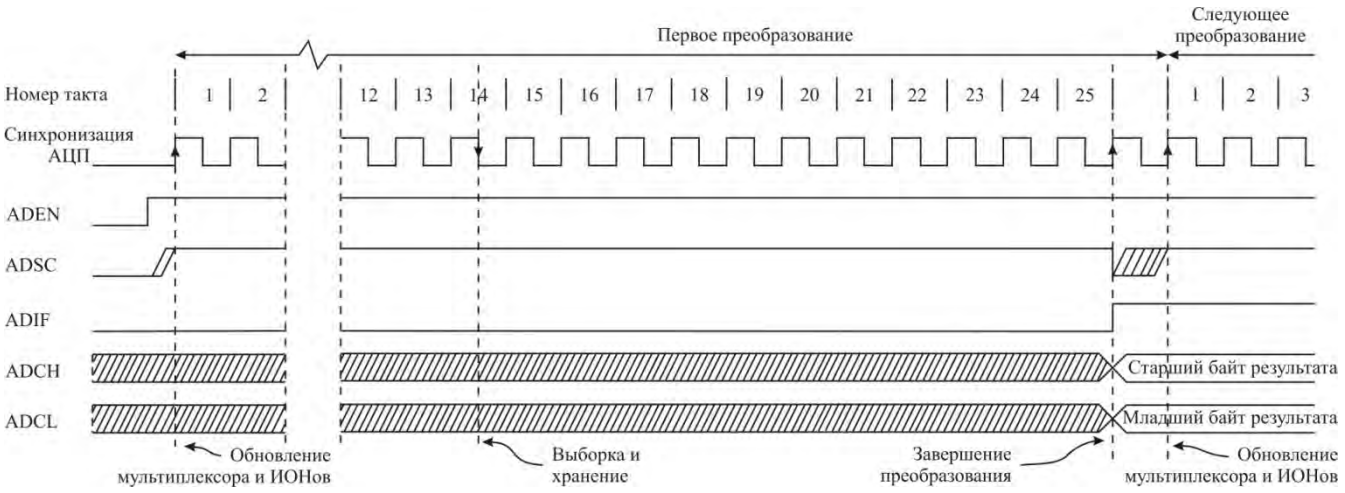


Рисунок 3.111 – Временная диаграмма работы АЦП при первом преобразовании в режиме одиночного преобразования

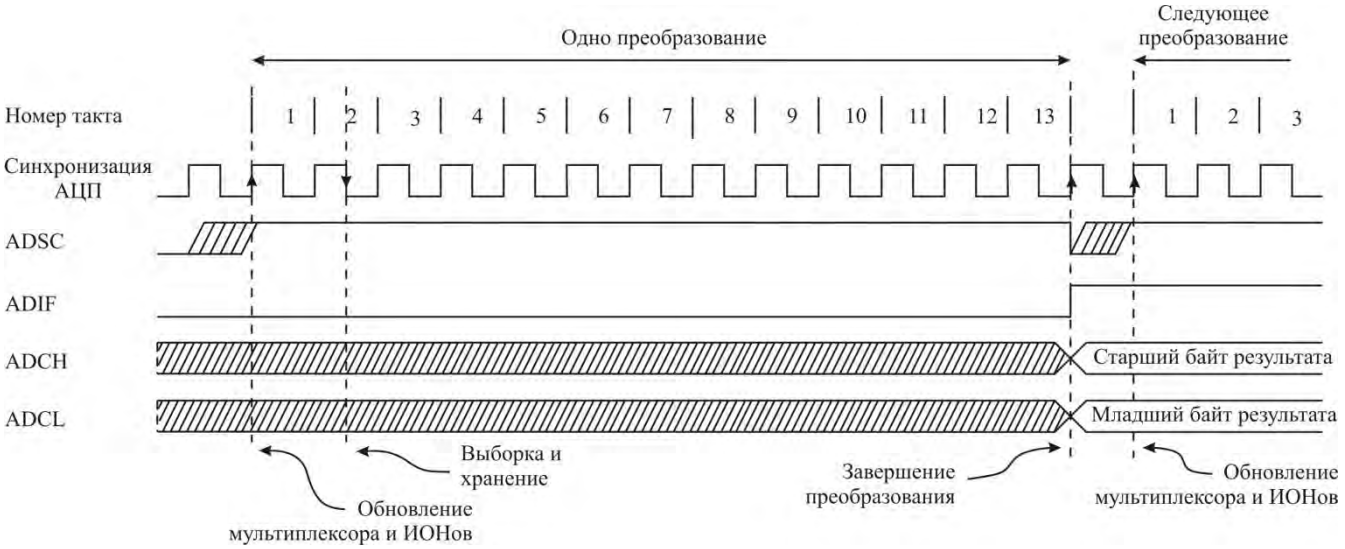


Рисунок 3.112 – Временная диаграмма работы АЦП в режиме одиночного преобразования

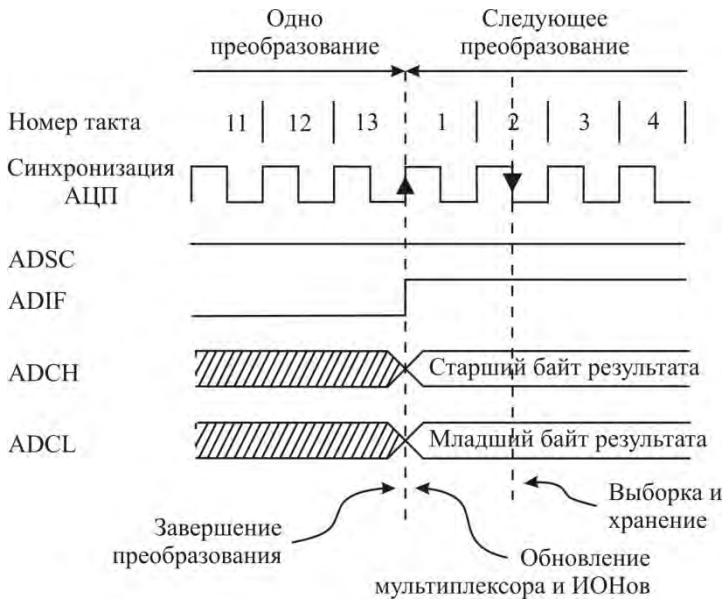


Рисунок 3.113 – Временная диаграмма работы АЦП в режиме непрерывного преобразования

Таблица 3.95 – Время преобразования АЦП

Состояние	Выборка и хранение (такты с момента запуска преобразования)	Время преобразования (такты)
Первое преобразование	13,5	25
Нормальные преобразования, несимметричный вход	1,5	13
Нормальные преобразования, дифференциальный вход	1,5/2,5	13/14

Каналы дифференциального усиления

При использовании каналов дифференциального усиления необходимо принять во внимание некоторые особенности.

Дифференциальные преобразования синхронизированы по отношению к внутренней синхронизации CK_{ADC2}, частота которой равна половине частоты синхронизации АЦП. Данная синхронизация выполняется автоматически интерфейсом АЦП таким образом, что операция выборки и хранения начинается определенным фронтом сигнала CK_{ADC2}. Преобразование, инициированное пользователем (то есть все одиночные преобразования и первое преобразование в непрерывном режиме), когда CK_{ADC2} находится в низком уровне, занимает то же самое количество времени, что и несимметричное преобразование (13 тактов синхронизации АЦП). Преобразование, инициированное пользователем когда CK_{ADC2} находится на высоком уровне, требует 14 тактов из-за работы механизма синхронизации. В непрерывном режиме новое преобразование стартует сразу после завершения предыдущего, и поскольку в этот момент CK_{ADC2} находится в высоком уровне, то все автоматически запущенные преобразования (то есть все, кроме первого) займут 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 полосы пропускания усилительного каскада, то необходимо установить внешний фильтр нижних частот. Следует обратить внимание, что частота синхронизации АЦП не зависит от ограничений частотного диапазона усилительного каскада. Например, период синхронизации АЦП может быть равен 6 мкс, позволяя частоте преобразования канала быть равной 12 тысячам преобразований в секунду, независимо от полосы пропускания этого канала.

3.18.5 Изменение канала или выбор источника опорного сигнала

Биты MUXn и REFS1–0 в регистре ADMUX поддерживают одноступенчатую буферизацию через временный регистр, к которому ЦПУ имеет произвольный доступ. Этим гарантируется, что новые настройки канала и опорного источника вступят в силу только в безопасный момент в процессе преобразования. До начала преобразования любые изменения канала и источника опорного сигнала вступают в силу сразу после их модификации. Как только начинается процесс преобразования, доступ к изменению канала и опорного источника блокируется, чтобы обеспечить достаточное время выборки для АЦП. Возможность модификации возобновляется на последнем такте АЦП перед завершением преобразования (перед установкой флага ADIF в регистре ADCSRA). Следует обратить внимание, что преобразование начинается на следующем нарастающем фронте тактового сигнала АЦП после записи ADSC. Таким образом, пользователю рекомендуется не записывать новое значение канала или источника опорного сигнала в регистр ADMUX до окончания первого такта синхронизации АЦП после записи ADSC.

Особые меры необходимо предпринять при изменении дифференциального канала. Как только осуществлен выбор дифференциального канала, усилительному каскаду может потребоваться 125 мкс для стабилизации нового значения. Таким образом, не следует начинать преобразования в течение первых 125 мкс после выбора нового дифференциального

канала. Если же в этот период времени преобразования все-таки выполнялись, то их результат необходимо игнорировать.

Такую же задержку необходимо ввести при первом дифференциальном преобразовании после изменения источника опорного сигнала АЦП (путем изменения битов REFS1–0 в регистре ADMUX).

Если разрешена работа интерфейса JTAG, то функции каналов АЦП PORTF7–4 блокируются. См. таблицу 3.42 в подпункте «Альтернативные функции порта F».

Входные каналы АЦП

При переключении канала пользователю следует учесть некоторые рекомендации для обеспечения корректности выбора канала.

В режиме одиночного преобразования переключение канала необходимо выполнять перед началом преобразования. Выбор канала можно осуществить в течение одного такта синхронизации АЦП после записи логической единицы в ADSC. Однако самым простым методом является ожидание завершения преобразования перед выбором нового канала.

В режиме непрерывного преобразования переключение канала необходимо выполнять перед началом первого преобразования. Выбор канала можно осуществить в течение одного такта синхронизации АЦП после записи логической единицы в ADSC. Однако самым простым методом является ожидание завершения первого преобразования перед выбором нового канала. Поскольку очередное преобразование уже запущено автоматически, то следующий результат будет соответствовать предыдущему каналу. Последующие преобразования будут отражать результат уже для нового канала.

При переключении на дифференциальный канал результат первого преобразования может иметь низкую точность из-за переходного процесса в схеме автоматической регулировки смещения. Поэтому результат первого преобразования рекомендуется игнорировать.

Источник опорного напряжения АЦП

Источник опорного напряжения для АЦП (U_{REF}) определяет диапазон преобразования АЦП. Если уровень однополярного сигнала выше U_{REF}, то результат преобразования будет равен коду, близкому к 0x3FF. В качестве U_{REF} могут выступать ∩VCC, внутренний ИОН на 2,56 В или внешний ИОН, подключенный к выводу AREF.

∩VCC подключается к АЦП через пассивный ключ. Внутреннее опорное напряжение 2,56 В генерируется внутренним эталонным источником (U_{BG}) через внутренний усилитель. В любом случае внешний вывод AREF связан с АЦП напрямую, и поэтому можно снизить влияние шумов на опорный источник за счет подключения конденсатора между выводом AREF и ∩0V. Напряжение U_{REF} также можно измерить на выводе AREF с помощью высокоомного вольтметра. Обратите внимание, что U_{REF} является высокоомным источником и поэтому внешне к нему может быть подключена только емкостная нагрузка.

Если пользователь имеет постоянный опорный источник, подключенный к выводу AREF, то пользователю не разрешается применять другие варианты для источника опорного сигнала, так как это приведет к шунтированию внешнего опорного напряжения. Если к выводу AREF не приложено напряжение, то для опорного источника пользователь может выбирать между ∩VCC и ИОН на 2,56 В. Результат первого преобразования после переключения опорного источника может быть неточным, и поэтому пользователю рекомендуется его игнорировать.

Если используются дифференциальные каналы, то напряжение выбранного опорного источника должно находиться в пределах от 2 В до (∩VCC – 0,5) В.

3.18.6 Подавитель шумов АЦП

АЦП характеризуется наличием подавителя шумов, который позволяет выполнять преобразования в режиме сна с целью уменьшения помех, вызванных работой ядра ЦПУ и периферийных устройств ввода-вывода. Подавитель шумов можно применять в режиме

снижения шумов АЦП и в режиме холостого хода. При использовании данной функции необходимо придерживаться следующей процедуры:

1 Убедитесь, что работа АЦП разрешена, и что он не выполняет какие-либо преобразования. Затем необходимо выбрать режим одиночного преобразования и разрешить прерывание по завершении преобразования АЦП.

2 Переведите микроконтроллер в режим снижения шумов АЦП (или режим холостого хода). АЦП запустит преобразование, как только остановится ЦПУ.

3 Если до завершения преобразования не возникают какие-либо другие прерывания, то прерывание АЦП выведет микроконтроллер из спящего режима, и программа перейдет на вектор обработки прерывания по завершении преобразования АЦП. Если до завершения преобразования другое прерывание пробуждает микроконтроллер, то это прерывание будет реализовано, а по завершении преобразования будет сгенерирован соответствующий запрос на прерывание. АЦП будет оставаться в активном режиме до тех пор, пока не произойдет выполнение очередной команды sleep.

Следует обратить внимание, что АЦП не отключается автоматически при переходе в другие спящие режимы, кроме режима холостого хода и снижения шумов АЦП. Поэтому пользователю рекомендуется записывать логический ноль в бит ADEN перед входом в такие режимы сна во избежание чрезмерного энергопотребления. Если же работа АЦП была разрешена в таких режимах сна, и пользователь желает выполнить дифференциальное преобразование, то рекомендуется сначала выключить, а затем снова включить АЦП после пробуждения, чтобы инициировать расширенное преобразование для получения действительного результата.

Схема аналогового входа

Схема аналогового входа для несимметричных каналов представлена на рисунке 3.114. Аналоговый сигнал, приложенный к выводу ADCn, нагружается емкостью вывода и входным сопротивлением утечки, независимо от того, выбран ли этот канал в качестве входа к АЦП. После подключения канала к АЦП аналоговый сигнал будет связан с конденсатором выборки и хранения через последовательный резистор, сопротивление которого эквивалентно всей входной цепи.

АЦП оптимизирован под аналоговые сигналы с выходным сопротивлением порядка 10 кОм или меньше. Если используется такой источник сигнала, то время выборки будет незначительным. Если же используется источник с более высоким сопротивлением, то время выборки будет определяться периодом времени, который требуется для зарядки конденсатора выборки и хранения, и длительность этого периода может колебаться в широких пределах. Поэтому рекомендуется использовать только источники с малым выходным сопротивлением и медленно изменяющимися сигналами, т. к. в этом случае заряд конденсатора выборки и хранения будет достаточно быстрым.

По отношению к каналам с дифференциальным усилением рекомендуется использовать сигналы с сопротивлением порядка нескольких сотен килоом или меньше.

Следует предусмотреть, чтобы в предварительных каскадах формирования аналогового сигнала к входу АЦП не вносились частоты выше $f_{АЦП}/2$, в противном случае результат преобразования может быть некорректным. Поэтому перед подачей сигналов на входы АЦП следует удалять высокочастотные составляющие с помощью фильтра нижних частот.

Рекомендации по снижению влияния шумов на результат преобразования

Работа цифровых узлов внутри и снаружи микроконтроллера связана с генерацией электромагнитных помех, которые могут негативно сказаться на точности измерения аналогового сигнала. Если точность преобразования является критическим параметром, то уровень шумов можно снизить, придерживаясь следующих рекомендаций:

- Поддерживайте путь аналоговых сигналов как можно более коротким. Следите, чтобы аналоговые сигналы проходили над плоскостью (слоем) с аналоговой землей (экраном) и далеко от проводников, передающих высокочастотные цифровые сигналы.

- Вывод ∩VCC необходимо связать с цифровым питанием #VCC через LC-цепь, как показано на рисунке 3.115.

- Используйте функцию подавления шумов АЦП для уменьшения помех, связанных с работой ядра ЦПУ.

- Если какой-либо из выводов АЦП используется как цифровой выход, то чрезвычайно важно следить за тем, чтобы не произошло переключение состояния этого выхода в процессе преобразования.

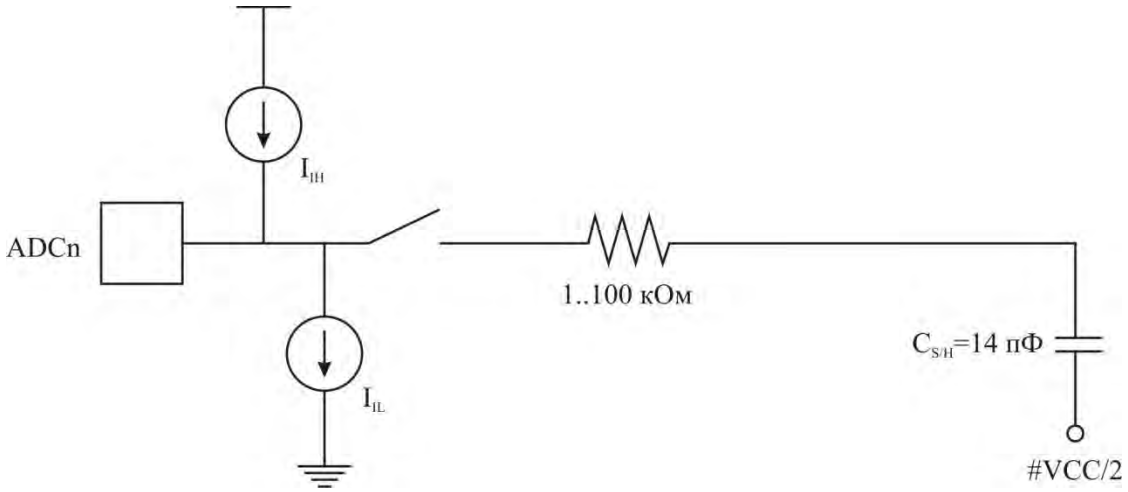


Рисунок 3.114 – Схема аналогового входа

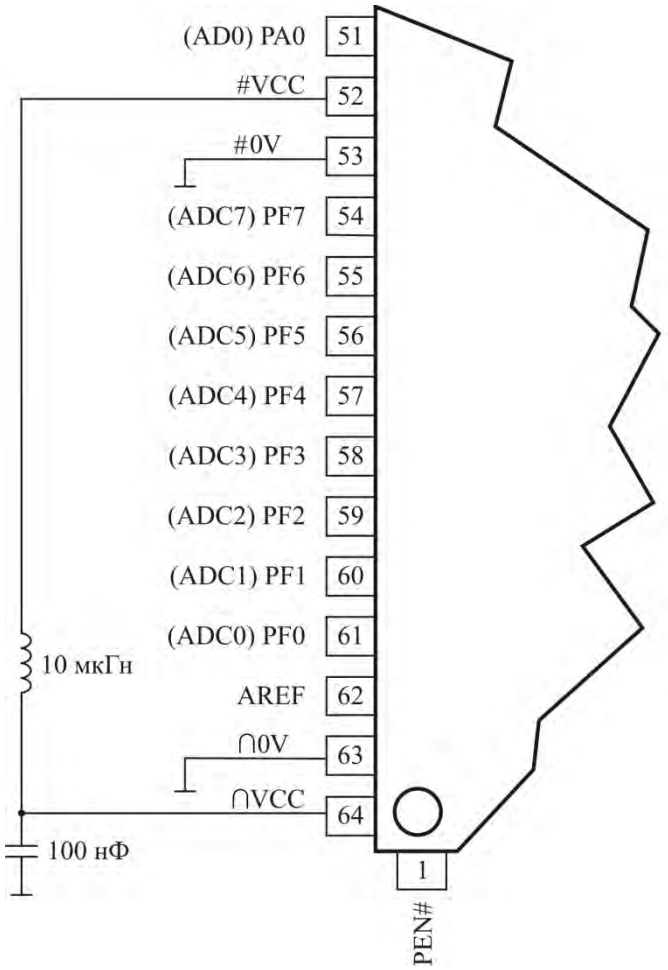


Рисунок 3.115 – Подключение питания АЦП

Методы компенсации смещения

Усилительный каскад имеет встроенную схему компенсации смещения, которая стремится максимально приблизить к нулю смещение дифференциального измерения. Оставшееся смещение на пути аналогового сигнала можно измерить напрямую, выбрав один и тот же канал для обоих дифференциальных входов. Это остаточное смещение можно затем программно вычесть из результата измерения. Использование программного алгоритма коррекции позволяет достичь уменьшения смещения менее одного младшего разряда для любого канала.

Определение погрешности АЦП

N-разрядный несимметричный АЦП преобразовывает напряжение с несимметричных входов линейно между ∩0V и V_{REF} с количеством шагов 2^n (младших разрядов). Минимальный код читается как 0, а максимальный - как (2^n-1).

Несколько параметров характеризуют отклонение от идеального результата. Основные погрешности преобразования являются отклонением реальной функции преобразования от идеальной. К ним относятся:

1. Смещение (рисунок 3.116) – отклонение первого перехода (с 0x000 на 0x001) по сравнению с идеальным переходом (т. е. при 0,5 младшего значащего разряда). Идеальное значение: ноль.

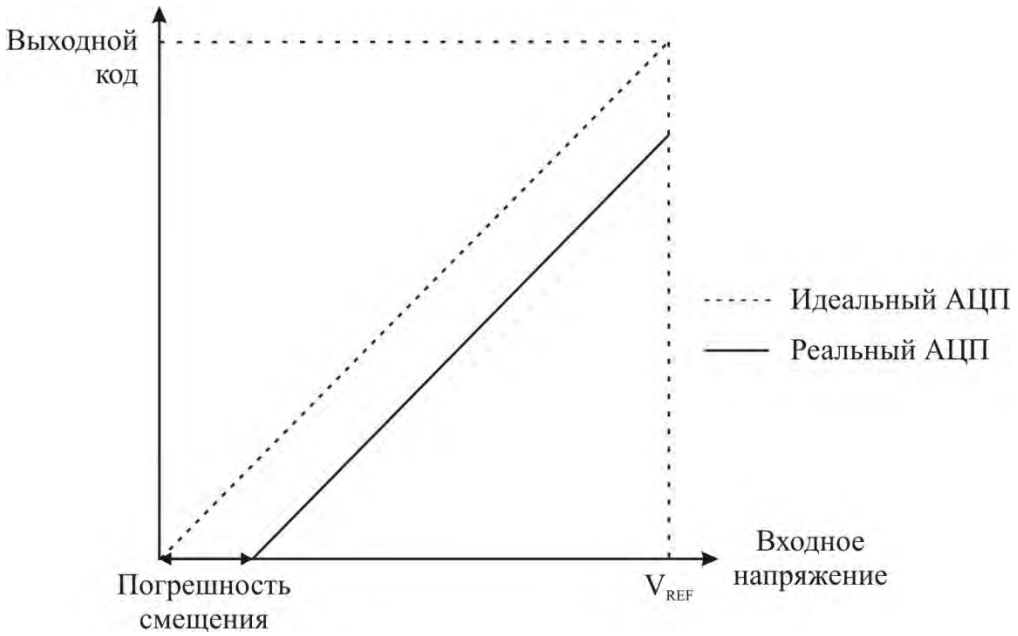


Рисунок 3.116 – Погрешность смещения

2. Погрешность усиления (рисунок 3.117). После корректировки смещения погрешность усиления определяется как отклонение последнего перехода (с 0x3FE на 0x3FF) от идеального перехода (т. е. отклонение при максимальном значении 1,5 младшего значащего разряда ниже максимума). Идеальное значение: ноль.

3. Интегральная нелинейность (INL). После корректировки смещения и погрешности усиления INL представляет собой максимальное отклонение реальной функции преобразования от идеальной функции для любого кода. Идеальное значение: ноль (рисунок 3.118).

4. Дифференциальная нелинейность (DNL) – максимальное отклонение между шириной фактического кода (интервал между двумя смежными переходами) от ширины идеального кода (1 младший значащий разряд). Идеальное значение: ноль (рисунок 3.119).

5. Погрешность квантования. Возникает из-за преобразования входного напряжения в конечное число кодов. Погрешность квантования – это интервал входного напряжения протяженностью один младший значащий разряд (шаг квантования по напряжению), кото-

рый характеризуется одним и тем же кодом. Погрешность квантования всегда равна ±0,5 младшего значащего разряда.

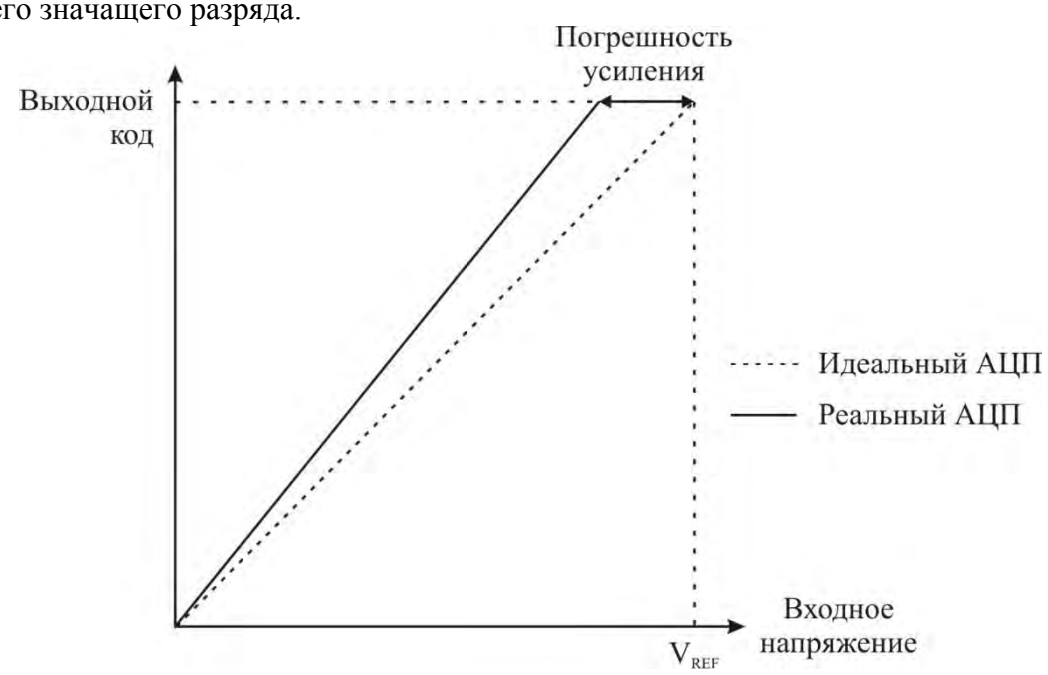


Рисунок 3.117 – Погрешность усиления

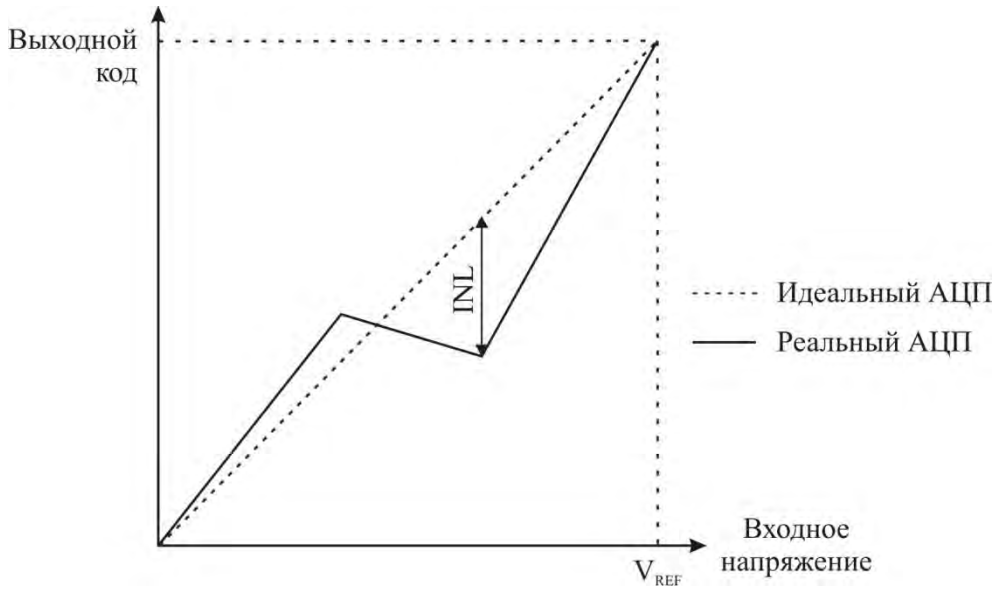


Рисунок 3.118 – Интегральная нелинейность (INL)

6. Абсолютная погрешность – максимальное отклонение реальной (без подстройки) функции преобразования от идеальной функции для любого кода. Она представляет собой результат действия нескольких эффектов: погрешностей смещения и усиления, дифференциальной погрешности, нелинейности и погрешности квантования. Идеальное значение: ±0,5 младшего значащего разряда.

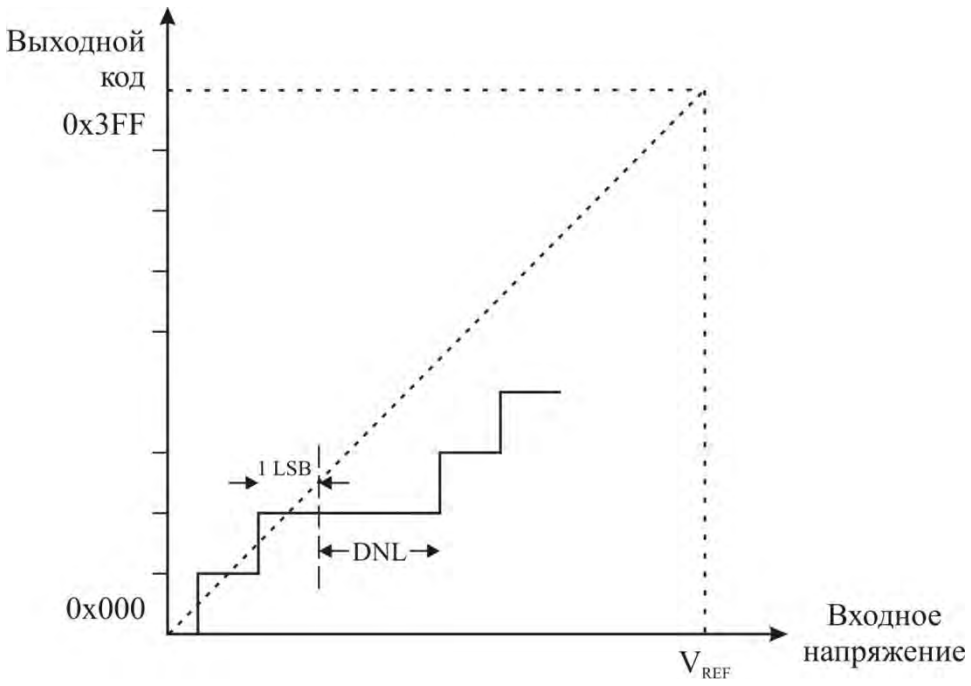


Рисунок 3.119 – Дифференциальная нелинейность (DNL)

3.18.7 Результат преобразования АЦП

По завершении преобразования (ADIF = 1) результат может быть считан из пары регистров данных АЦП (ADCL, ADCH).

При использовании несимметричных каналов результат преобразования равен

$$ADC = \frac{U_{IN} \cdot 1024}{U_{REF}},$$

где U_{IN} – уровень напряжения на выбранном входе АЦП;

U_{REF} – напряжение выбранного ИОН (см. таблицы 3.97 и 3.98).

Код 0x000 соответствует уровню аналоговой земли, а 0x3FF – уровню напряжения опорного источника минус один младший значащий разряд.

При использовании дифференциальных каналов результат будет равен

$$ADC = \frac{(U_{POS} - U_{NEG}) \cdot GAIN \cdot 512}{U_{REF}},$$

где U_{POS} – напряжение на положительном входе;

U_{NEG} – напряжение на отрицательном входе;

GAIN – выбранный коэффициент усиления;

U_{REF} – напряжение выбранного ИОН.

Результат представляется в коде двоичного дополнения, начиная с 0x200 (-512d) до 0x1FF (+511d). Обратите внимание, что при необходимости быстро определить полярность результата достаточно опросить старший бит результата преобразования (ADC9 в регистре ADCH). Если данный бит равен логической единице, то результат отрицательный, если же он равен логическому нулю, то результат положительный. На рисунке 3.120 представлена функция преобразования АЦП в дифференциальном режиме.

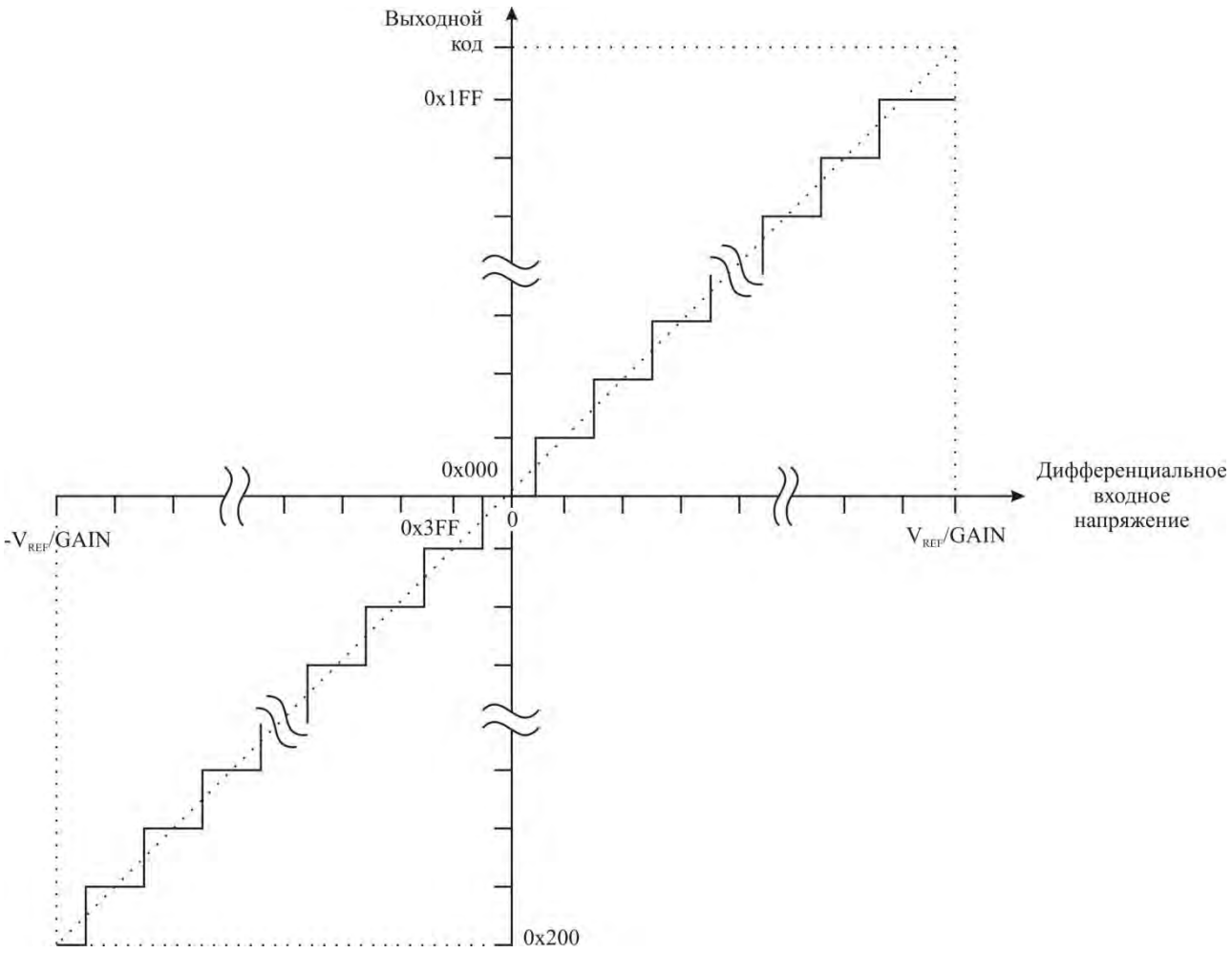


Рисунок 3.120 – Функция преобразования АЦП при измерении дифференциального сигнала

В таблице 3.96 представлены результирующие выходные коды для дифференциальной пары каналов (ADCn – ADCm) с коэффициентом усиления GAIN и опорным напряжением U_{REF}.

Таблица 3.96 – Связь между входным напряжением и выходными кодами

V_{ADCn}	Считываемый код	Соответствующее десятичное значение
$V_{ADCm} + U_{REF}$ /GAIN	0x1FF	511
$V_{ADCm} + 511/512\ U_{REF}$ /GAIN	0x1FF	511
$V_{ADCm} + 511/512\ U_{REF}$ /GAIN	0x1FE	510
…	…	…
$V_{ADCm} + 1/512\ U_{REF}$ /GAIN	0x001	1
V_{ADCm}	0x000	0
$V_{ADCm} - 1/512\ U_{REF}$ /GAIN	0x3FF	-1
…	…	…
$V_{ADCm} - 511/512\ U_{REF}$ /GAIN	0x201	-511
$V_{ADCm} - U_{REF}$ /GAIN	0x200	-512

Пример:

Пусть ADMUX = 0xED. Используется пара входов ADC3 – ADC2, коэффициент усиления 10x, U_{REF} = 2,56 В, напряжение на входах ADC3 и ADC2 равно 300 мВ и 500 мВ, соответственно. Результат представлен с левосторонним выравниванием. Тогда:

ADCR = 512 × 10 × (300 – 500) / 2560 = -400 = 0x270.

С учетом выбранного левостороннего формата размещения результата ADCL = 0x00, а ADCH = 0x9C. Если же выбран правосторонний формат (ADLAR = 0), то ADCL = 0x70, ADCH = 0x02.

Регистр управления мультиплексором АЦП – ADMUX

Бит	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7–6: REFS1–0: Биты выбора источника опорного напряжения

Данные биты определяют, какое напряжение будет использоваться в качестве опорного для АЦП (см. таблицу 3.97). Если изменить значения данных битов в процессе преобразования, то новые установки вступят в силу только после завершения текущего преобразования (т.е. когда установится бит ADIF в регистре ADCSRA). Не разрешается использовать варианты внутреннего опорного напряжения, если к выводу AREF подключен внешний опорный источник.

Таблица 3.97 – Выбор опорного источника для АЦП

REFS1	REFS0	Выбор источника опорного напряжения
0	0	AREF, внутреннее опорное напряжение отключено
0	1	∩VCC с внешним конденсатором на выводе AREF
1	0	Зарезервировано
1	1	Внутренний ИОН на 2,56 В с внешним конденсатором на выводе AREF

Разряд 5 – ADLAR: Бит управления представлением результата преобразования

Бит ADLAR влияет на представление результата преобразования в регистре данных АЦП. Если ADLAR = 1, то результат преобразования будет иметь левосторонний формат (выравнивание влево), если ADLAR = 0, то правосторонний (выравнивание вправо). Действие бита ADLAR вступает в силу сразу после изменения, независимо от выполняемого в данный момент преобразования. Полное описание действия данного бита представлено ниже в подпункте «Регистры данных АЦП – ADCL и ADCH».

Разряд 4–0: MUX4–0: Биты выбора аналогового канала и коэффициента усиления

Данные биты определяют, какие из имеющихся аналоговых входов подключаются к АЦП. Кроме того, с их помощью можно выбрать коэффициент усиления для дифференциальных каналов (см. таблицу 3.98). Если изменить значения данных битов в процессе преобразования, то новые установки вступят в силу только после завершения текущего преобразования (т.е. когда установится бит ADIF в регистре ADCSRA).

Таблица 3.98 – Выбор входного канала и коэффициента усиления

MUX4...0	Несимметричный вход	Положительный дифференциальный вход	Отрицательный дифференциальный вход	Коэффициент усиления
00000	ADC0	Не доступно		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			

Продолжение таблицы 3.98

MUX4...0	Несимметричный вход	Положительный дифференциальный вход	Отрицательный дифференциальный вход	Коэффициент усиления
01000*	Не доступно	ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010*		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x
11101		ADC5	ADC2	1x
11110	1,23 В (V_{BG})	Не доступно		
11111	0 В (∩OV)			

* Можно использовать для точного определения смещения.

Регистр А управления и состояния АЦП – ADCSRA

Бит	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – ADEN: Разрешение работы АЦП

Запись логической единицы в бит ADEN разрешает работу АЦП. Если в данный бит записать логический ноль, то АЦП отключается. Отключение АЦП в момент выполнения преобразования приводит к прекращению данного преобразования.

Разряд 6 – ADSC: Запуск преобразования АЦП

В режиме одиночного преобразования записывайте логическую единицу в данный бит для запуска каждого преобразования. В режиме непрерывного преобразования записывайте логическую единицу в данный бит для запуска первого преобразования. Первое преобразования после установки бита ADSC с момента включения АЦП, или если установка бита ADSC и включение АЦП происходят в один и тот же момент, займет 25 тактов синхронизации АЦП вместо нормальных 13 тактов. Это первое преобразование служит для инициализации АЦП.

Бит ADSC будет читаться как логическая единица, пока выполняется преобразование. По завершении преобразования данный бит читается как ноль. Запись логического нуля в бит ADSC не предусмотрена и не оказывает никакого воздействия.

Разряд 5 – ADFR: Выбор режима непрерывного преобразования АЦП

Если в данный бит записать логическую единицу, то АЦП перейдет в режим непрерывного преобразования. В этом режиме АЦП непрерывно выполняет преобразования и обновляет регистры данных. Запись логического нуля в этот бит прекращает работу в данном режиме.

Разряд 4 – ADIF: Флаг прерывания АЦП

Данный флаг устанавливается после завершения преобразования АЦП и обновления регистров данных. Прерывание по завершении преобразования АЦП выполняется, если установлены бит ADIE и бит I в регистре SREG. Флаг ADIF сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно флаг ADIF сбрасывается путем записи в него логической единицы. Имейте в виду, что при выполнении операции «чтение-модификация-запись» с регистром ADCSRA ожидаемое прерывание может быть блокировано. Это также следует учитывать при использовании команд SBI и CBI.

Разряд 3 – ADIE: Разрешение прерывания АЦП

После записи логической единицы в этот бит при условии, что установлен бит I в регистре SREG, активируется прерывание по завершении преобразования АЦП.

Разряды 2–0: ADPS2–0: Биты управления предделителем АЦП

Данные биты определяют коэффициент деления между частотой ЦПУ и частотой входной синхронизации АЦП.

Таблица 3.99 – Управление предделителем АЦП

ADPS2	ADPS1	ADPS0	Коэффициент деления
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Регистры данных АЦП – ADCL и ADCH

ADLAR = 0:

Бит	15	14	13	12	11	10	9	8	
	-	-	-	-	-	-	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Чтение/Запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч	
	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч	
Начальное значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1:

Бит	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	-	-	-	-	-	-	ADCL
	7	6	5	4	3	2	1	0	
Чтение/Запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч	
	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч	
Начальное значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

По завершении преобразования результат помещается в эти два регистра. При использовании дифференциального режима преобразования результат представляется в коде двоичного дополнения.

Если выполнено чтение ADCL, то обновление регистров данных АЦП не производится до тех пор, пока не будет считан регистр ADCH. Следовательно, если результат представлен в формате выравнивание влево и требуется точность не более 8 разрядов, то достаточно выполнить считывание только регистра ADCH. В противном случае необходимо сначала считать данные регистра ADCL и только затем ADCH.

Бит ADLAR, а также биты MUXn в регистре ADMUX, влияют на способ считывания результата из регистров данных АЦП. Если бит ADLAR установлен, то результат будет представлен в формате выравнивание влево. Если бит ADLAR сброшен (по умолчанию), то результат будет представлен в формате выравнивание вправо.

ADC9–0: Результат преобразования АЦП

Данные биты представляют результат преобразования.

3.19 Интерфейс JTAG и встроенная система отладки

3.19.1 Особенности

Особенности интерфейса JTAG и встроенной системы отладки:

- Интерфейс JTAG (совместимый со стандартом IEEE 1149.1).
- Возможность периферийного сканирования в соответствии со стандартом IEEE 1149.1.
- Отладочная программа имеет доступ к следующим блокам микроконтроллера:
  - Все внутренние периферийные модули.
  - Внутреннее и внешнее ОЗУ.
  - Внутренний регистровый файл.
  - Программный счетчик.
  - ЭПД и память программ.
- Обширная встроенная поддержка отладки для условий прерываний, включая:
  - Прерывание по инструкции микроконтроллера.
  - Прерывание по изменению потока памяти программ.
  - Пошаговое прерывание.
  - Точки прерывания памяти программ по одиночному адресу или адресному диапазону.
  - Точки прерывания памяти данных по одиночному адресу или адресному диапазону.
- Программирование памяти программ, ЭПД, конфигурационных битов и битов защиты программ через интерфейс JTAG.
- Встроенная система отладки поддерживается AVR Studio.

3.19.2 Краткий обзор

JTAG-интерфейс микроконтроллера, совместимый со стандартом IEEE 1149.1, может применяться для:

- тестирования печатных плат благодаря возможности периферийного сканирования;
- программирования энергонезависимой памяти, конфигурационных битов и битов защиты памяти программ;
- встроенной отладки.

Краткое описание представлено в следующих пунктах документа. Подробное описание программирования через интерфейс JTAG и использования цепи периферийного сканирования можно найти в 3.21.9 «Программирование через интерфейс JTAG» и 3.19.8 «Периферийное сканирование. Стандарт IEEE 1149.1», соответственно.

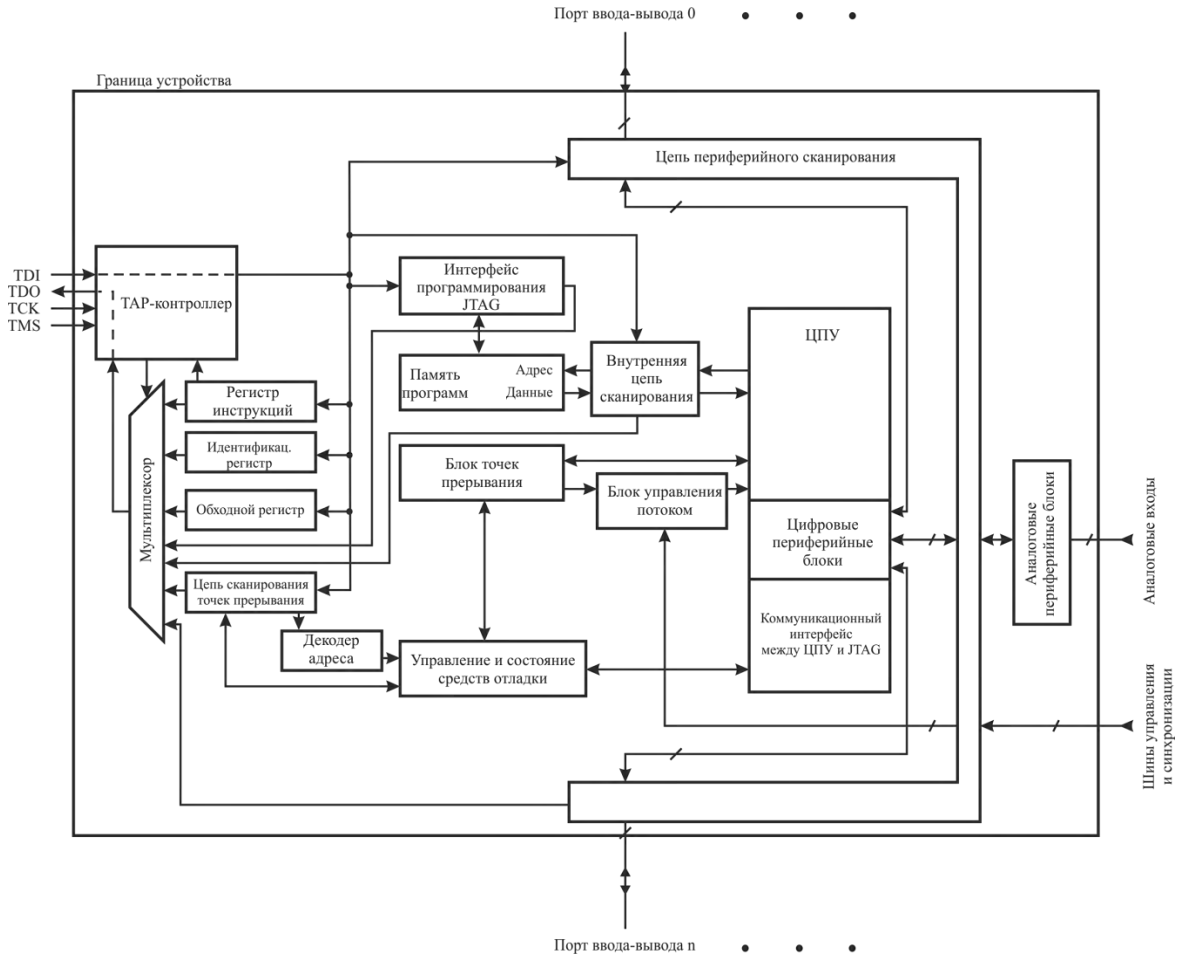


Рисунок 3.121 – Структурная схема интерфейса JTAG и встроенной системы отладки

На рисунке 3.121 изображена структурная схема JTAG-интерфейса и встроенной отладочной системы. TAP-контроллер представляет собой автомат конечных состояний, управляемый сигналами TCK и TMS. TAP-контроллер выбирает либо регистр инструкций JTAG (РИ), либо один из нескольких регистров данных (РД) в качестве цепи сканирования (сдвигового регистра) между входом TDI и выходом TDO. Регистр инструкций содержит JTAG-инструкции, управляющие поведением регистра данных.

Регистр идентификации устройства, обходной регистр и цепь периферийного сканирования являются регистрами данных, применяемыми для тестирования на уровне плат. Интерфейс программирования JTAG (фактически состоящий из нескольких физических и виртуальных регистров данных) используется для последовательного программирования

через JTAG-интерфейс. Внутренняя цепь сканирования и цепь сканирования точек прерывания применяются только для встроенной отладки.

3.19.3 Порт доступа к функциям тестирования

Доступ к JTAG-интерфейсу осуществляется через четыре вывода микроконтроллера. В терминологии JTAG эти выводы в совокупности составляют порт доступа к функциям тестирования – TAP. Данными выводами являются:

- TMS – выбор режима тестирования. Этот вывод применяется для навигации по автомату конечных состояний TAP-контроллера.
- TCK – синхронизация тестирования. Работа JTAG-интерфейса синхронизируется в соответствии с сигналом TCK.
- TDI – ввод данных при тестировании. Последовательные входные данные, которые должны быть сдвинуты в регистр инструкций или регистр данных.
- TDO – вывод данных при тестировании. Последовательные выходные данные из регистра инструкций или регистра данных.

Стандарт IEEE 1149.1 также определяет один дополнительный TAP-сигнал, TRST (сброс тестирования), который в данном микроконтроллере не предусмотрен.

Если конфигурационный бит JTAGEN не запрограммирован, то эти четыре TAP-вывода являются нормальными выводами порта, а TAP-контроллер находится в состоянии сброса. Если конфигурационный бит JTAGEN запрограммирован, а бит JTD в регистре MCUCSR сброшен, то разрешена работа JTAG-интерфейса для периферийного сканирования и программирования. В данном случае выходной TAP-вывод (TDO) находится в высокоимпедансном состоянии, пока TAP-контроллер не начнет выдавать данные, и поэтому должен быть подключен к подтягивающему резистору или к другой конструкции, имеющей подтягивающие резисторы (например, вход TDI следующего устройства в цепи сканирования). Микроконтроллер поставляется с запрограммированным конфигурационным битом JTAGEN.

Помимо выводов JTAG-интерфейса отладочной программой отслеживается вывод RESET# для детектирования источников внешнего сброса. Отладчик также может подать на этот вывод низкий уровень, чтобы сбросить всю систему, при условии, что в приложении используются схема с открытым стоком на линии сброса.

3.19.4 TAP-контроллер

TAP-контроллер представляет собой автомат конечных состояний с 16 состояниями, который управляет работой схемы периферийного сканирования, программированием JTAG или встроенной отладочной системой. Переходы состояний, изображенные на рисунке 3.122, зависят от уровня сигнала, присутствующего на выводе TMS (его значение показывается рядом с каждым переходом состояний) в момент нарастающего фронта TCK. Исходное значение после сброса при подаче питания – это «Сброс тестовой логики».

В данном руководстве пользователя, по определению, младший значащий бит LSB принимается и выдается первым во всех сдвиговых регистрах.

Если состояние «Запуск теста/Ожидание» является текущим состоянием, то типичным сценарием для использования JTAG-интерфейса будет следующий алгоритм:

1. На вход TMS подается последовательность 1, 1, 0, 0 в моменты нарастающих фронтов сигнала TCK для входа в сдвиговый регистр инструкций – состояние «Сдвиг РИ». В этом состоянии подается четыре бита JTAG-инструкций в регистр инструкций с входа TDI в момент нарастающего фронта TCK. На входе TMS должен удерживаться низкий уровень во время ввода трех младших значащих битов, чтобы оставаться в состоянии «Сдвиг РИ». Старший значащий бит (MSB) инструкции подается одновременно с установкой сигнала TMS в «1» для выхода из данного состояния. Пока инструкция подается на вывод TDI, захваченное состояние регистра инструкций 0x01 выдается на вывод TDO. JTAG-

инструкция выбирает отдельный регистр данных в качестве канала между TDI и TDO и управляет схемой, окружающей выбранный регистр данных.

2. На вывод TMS подается последовательность 1, 1, 0, чтобы повторно войти в состояние «Запуск теста/Ожидание». Инструкция защелкивается на параллельном выходе сдвигового регистра при переходе в состояние «Обновление РИ». Состояния «Выход 1 РИ», «Пауза РИ» и «Выход 2 РИ» применяются только для навигации по автомату конечных состояний.

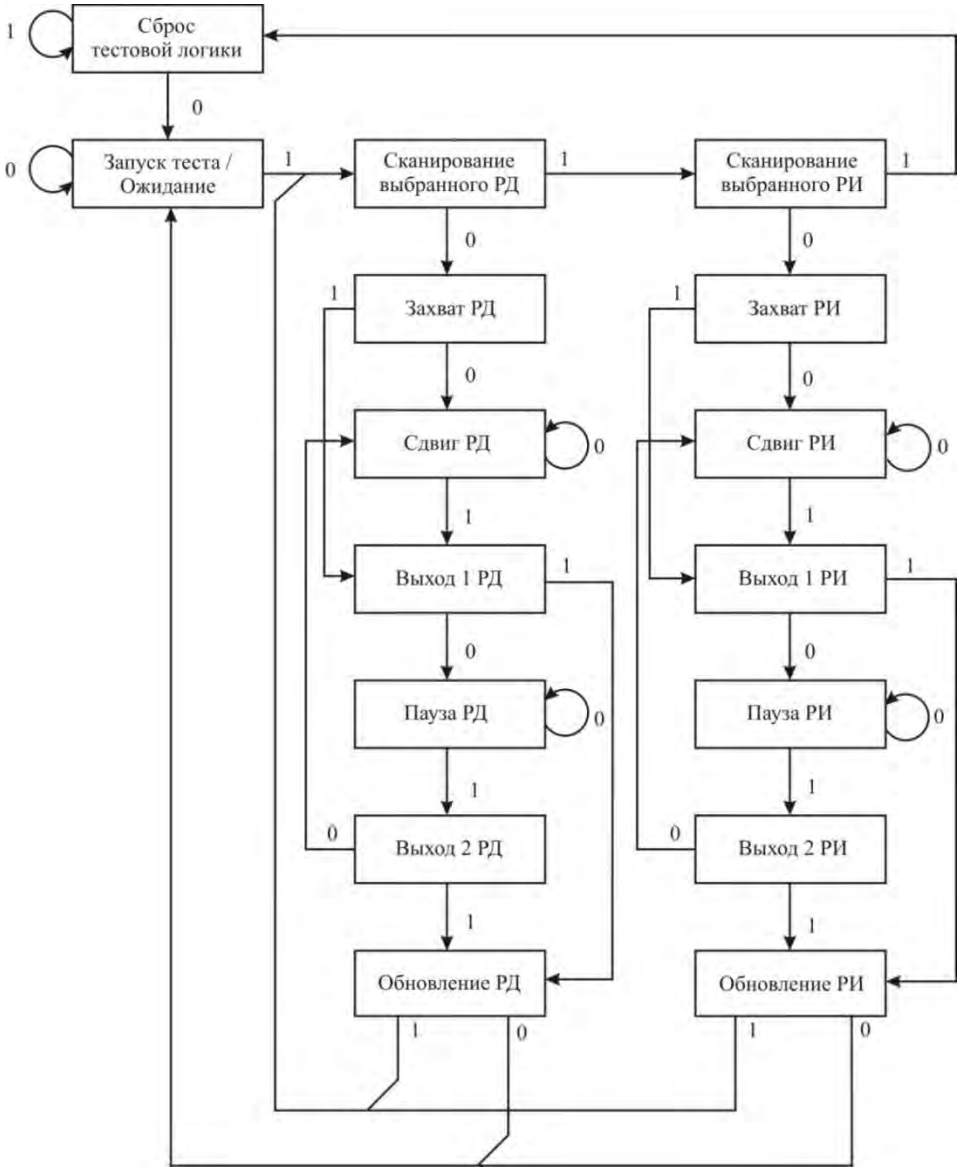


Рисунок 3.122 – Диаграмма состояний TAP-контроллера

3. На вход TMS подаем последовательность 1, 0, 0 в моменты нарастающих фронтов сигнала TCK для входа в сдвиговый регистр данных – состояние «Сдвиг РД». Пока находимся в этом состоянии, загружаем выбранный регистр данных (регистр, выбранный JTAG-инструкцией, представленной в регистре инструкций) с входа TDI в момент нарастающего фронта TCK. Для того чтобы оставаться в состоянии «Сдвиг РД», на входе TMS должен удерживаться низкий уровень во время ввода всех битов за исключением старшего значащего бита (MSB). Старший значащий бит (MSB) данных подается одновременно с установкой сигнала TMS в «1» для выхода из данного состояния. Пока регистр данных принимает данные с вывода TDI, хранимые данные регистра, захваченные в состоянии «Захват РД», выдаются на вывод TDO.

4. Подаем на вывод TMS последовательность 1, 1, 0, чтобы повторно войти в состояние «Запуск теста/Ожидание». Данные защелкивается на параллельном выходе сдвигового регистра при переходе в состояние «Обновление РД». Состояния «Выход 1 РД», «Пауза РД» и «Выход 2 РД» применяются только для навигации по автомату конечных состояний.

Как показано на диаграмме состояний, не требуется входить в состояние «Запуск теста/Ожидание» между выбором JTAG-инструкции и использованием регистров данных. Также, некоторые JTAG-инструкции могут выполнять определенные функции в состоянии «Запуск теста/Ожидание», делая его непригодным в качестве состояния ожидания.

Независимо от исходного состояния TAP-контроллера, в состояние «Сброс тестовой логики» всегда можно войти путем удержания сигнала TMS в высоком уровне в течение пяти периодов синхронизации тестирования TCK.

3.19.5 Использование цепи периферийного сканирования

Полное описание возможностей периферийного сканирования приведено далее в 3.19.8 «Периферийное сканирование. Стандарт IEEE 1149.1».

3.19.6 Использование встроенной системы отладки

Как показано на рисунке 3.121, аппаратное обеспечение для встроенной системы отладки состоит главным образом из следующих элементов:

- цепи сканирования между внутренним ЦПУ и внутренними периферийными модулями;
- блока точек прерывания;
- коммуникационного интерфейса между ЦПУ и системой JTAG.

Все операции чтения или модификации/записи, необходимые для реализации отладочной программы, выполнены путем подачи AVR-инструкций через внутреннюю цепь сканирования ЦПУ. Центральный процессор посылает результат в отображаемую ячейку памяти ввода-вывода, которая является частью коммуникационного интерфейса между ЦПУ и системой JTAG.

Блок точек прерывания реализует прерывание по изменению программного потока, пошаговое прерывание, две точки прерывания памяти программ и две комбинированные точки прерывания. Совместно эти точки прерывания могут быть сконфигурированы как:

- четыре отдельных точки прерывания памяти программ;
- три отдельных точки прерывания памяти программ плюс одна отдельная точка прерывания памяти данных;
- две отдельных точки прерывания памяти программ плюс две отдельных точки прерывания памяти данных;
- две отдельных точки прерывания памяти программ плюс одна точка прерывания памяти программ с маской («точка прерывания в диапазоне»);
- две отдельных точки прерывания памяти программ плюс одна точка прерывания памяти данных с маской («точка прерывания в диапазоне»).

Для включения порта доступа к функциям тестирования JTAG должен быть запрограммирован конфигурационный бит JTAGEN и сброшен бит JTD в регистре MCUCSR. Помимо этого, для того чтобы встроенная система отладки могла работать, должен быть запрограммирован конфигурационный бит OCDEN, а биты защиты программ не должны быть установлены. В целях безопасности работа отладочной системы блокируется, если установлен любой из битов защиты программ. В противном случае, встроенная отладочная система могла бы служить недокументированным способом входа, лазейкой в защищенное устройство.

Все команды, необходимые для выполнения, доступны в AVR Studio как на уровне исходного кода, так и на дизассемблированном уровне. Пользователь может выполнять программу пошагово либо путем входа в блок (построчная трассировка), либо путем пере-

шагивания через блок (выходим из функции, помещаем курсор на какой-либо оператор и выполняем программу до тех пор, пока не будет достигнут этот оператор, затем останавливаем программу и переустанавливаем цель выполнения). Кроме того, пользователь может иметь неограниченное число точек прерывания кода (используя команду BREAK) и до двух точек прерывания памяти данных, альтернативно скомбинированные как точки прерывания с маской.

3.19.7 Возможности программирования памяти через JTAG-интерфейс

Программирование через JTAG-интерфейс выполняется с помощью 4-выводного порта JTAG, линий TCK, TMS, TDI и TDO. Это единственные выводы (помимо выводов питания), за которыми необходимо следить/управлять, чтобы осуществить программирование через JTAG. Приложение внешнего напряжения 12 В не требуется. Конфигурационный бит JTAGEN должен быть запрограммирован, а бит JTD в регистре MCUCSR должен быть сброшен, чтобы включить порт доступа к функциям тестирования JTAG.

Интерфейс JTAG обеспечивает:

- программирование и верификацию памяти программ;
- программирование и верификацию ЭПД;
- программирование и верификацию конфигурационных битов;
- программирование и верификацию битов защиты программ.

Надежность битов защиты точно такая же, что и в режиме параллельного программирования. Когда запрограммированы биты защиты LB1 или LB2, то конфигурационный бит OCDEN не получится запрограммировать, если не произвести вначале стирание информации в кристалле.

Подробности о программировании через JTAG-интерфейс и специфических командах приведены в 3.21.9 «Программирование через интерфейс JTAG».

3.19.8 Периферийное сканирование. Стандарт IEEE 1149.1

Особенности

- JTAG-интерфейс (совместимый со стандартом IEEE 1149.1).
- Возможности периферийного сканирования в соответствии со стандартом JTAG.
- Полное сканирование всех функций порта, а также аналоговых схем, имеющих внешние подключения.
- Поддержка дополнительной инструкции IDCODE.
- Дополнительная открытая инструкция AVR_RESET для сброса микроконтроллера.

Обзор системы

Цепь периферийного сканирования позволяет управлять и наблюдать логические уровни на цифровых линиях ввода-вывода, а также интерфейс между цифровой и аналоговой логикой для аналоговых схем, имеющих внешние подключения. На системном уровне все интегральные схемы, имеющие возможности JTAG, подключаются последовательно в соответствии с сигналами TDI/TDO, чтобы образовать сдвиговый регистр с сохранением выдаваемых разрядов. Внешний контроллер настраивает устройства, чтобы они выводили значения на выходные линии и наблюдали за входными значениями, полученными от других устройств. Контроллер сравнивает полученные данные с ожидаемым результатом. Таким образом, периферийное сканирование обеспечивает механизм для тестирования внутренних соединений и целостности компонентов на печатной плате, используя лишь четыре TAP-сигнала.

Четыре обязательных JTAG-инструкции, определенные стандартом IEEE 1149.1 (IDCODE, BYPASS, SAMPLE/PRELOAD и EXTEST), а также открытая инструкция AVR_RESET могут использоваться для тестирования печатной платы. Начальное сканирование канала регистра данных покажет идентификационный код устройства, так как

IDCODE является JTAG-инструкцией по умолчанию. Во время режима тестирования желательно, чтобы устройство находилось в состоянии сброса. Если микроконтроллер не будет приведен в состояние сброса, то входы устройства, возможно, определятся операциями сканирования, но внутреннее программное обеспечение может пребывать в неопределенном состоянии при выходе из режима тестирования. При входе в состояние сброса выходы любого порта мгновенно перейдут в высокоимпедансное состояние, делая выполнение команды HIGHZ излишним. Если необходимо, можно выполнить инструкцию BYPASS, чтобы провести наикратчайшую цепь сканирования через устройство. Устройство может быть установлено в состояние сброса либо с помощью подачи на внешний вывод RESET# низкого уровня, либо путем вызова инструкции AVR_RESET с соответствующей установкой в регистре сброса.

Команда EXTEST применяется для выборки внешних выводов и загрузки данных на выходные линии. Данные из выходной защелки поступят на эти линии, как только команда EXTEST загрузится в регистр инструкций JTAG. Поэтому команду SAMPLE/PRELOAD также следует использовать для установки начальных значений в сканируемом кольце, чтобы избежать возможных проблем с печатной платой при вызове команды EXTEST в первый раз. Инструкция SAMPLE/PRELOAD также может применяться для просмотра состояния внешних выводов во время нормальной работы микроконтроллера.

Для включения порта доступа к функциям тестирования JTAG конфигурационный бит JTAGEN должен быть запрограммирован, а бит JTD в регистре ввода-вывода MCUCSR должен быть сброшен.

При использовании JTAG-интерфейса для периферийного сканирования возможно устанавливать частоту синхронизации тестирования выше внутренней тактовой частоты. Запускать синхронизацию кристалла не требуется.

3.19.9 Регистры данных

Регистрами данных, имеющими отношение к операциям периферийного сканирования, являются:

- обходной регистр;
- регистр идентификации устройства;
- регистр сброса;
- цепь периферийного сканирования.

Обходной регистр

Если обходной регистр выбран в качестве канала между TDI и TDO, то этот регистр сбрасывается в ноль при выходе из состояния TAP-контроллера «Захват РД». Обходной регистр можно использовать для шунтирования цепи сканирования в системе, когда должны тестироваться другие устройства.

Регистр идентификации устройства

На рисунке 3.123 показана структура регистра идентификации устройства.

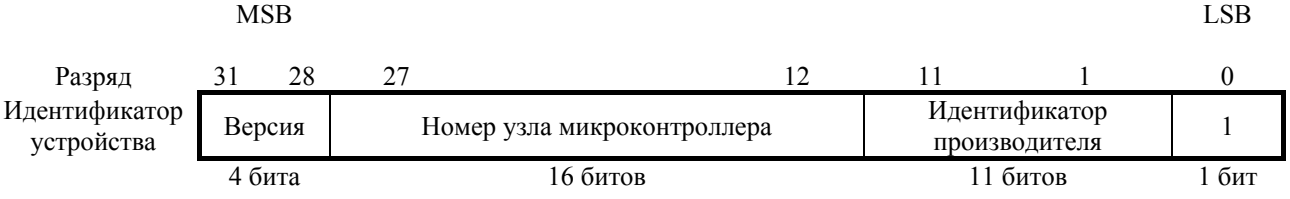


Рисунок 3.123 – Формат регистра идентификации устройства

Версия

Версия – это 4-разрядное значение, идентифицирующее версию устройства.

Номер устройства

Номер устройства представляет собой 16-разрядный код, идентифицирующий устройство. Номер устройства JTAG для ИС 1887BE7T приведен в таблице 3.100.

Таблица 3.100 – Номер устройства JTAG микроконтроллера

Номер узла	Номер узла JTAG (шестнадцатеричный)
ИС 1887BE7T	0x9702

Идентификатор

Идентификатор – 11-разрядный код. Значение кода для ИС 1887BE7T представлено в таблице 3.101.

Таблица 3.101 – Идентификатор

Идентификатор	Шестнадцатеричный код
ИС 1887BE7T	0x01F

Регистр сброса

Регистр сброса – это тестовый 1-битный регистр данных, применяемый для сброса микроконтроллера. Поскольку при сбросе микроконтроллер переводит выводы порта в третье состояние, то регистр сброса также может заменить функцию нереализованной дополнительной JTAG-инструкции HIGHZ.

Логическая единица в регистре сброса соответствует подаче на вывод внешнего сброса низкого логического уровня. Микроконтроллер будет пребывать в состоянии сброса до тех пор, пока в регистре сброса будет находиться логическая единица. В зависимости от установок конфигурационных битов для опций синхронизации устройство будет оставаться сброшенным в течение периода ожидания сброса (см. пункт 3.3.2 «Источники синхронизации») после освобождения регистра сброса. Выход из этого регистра данных не защелкивается, поэтому сброс произойдет немедленно, как показано на рисунке 3.124.

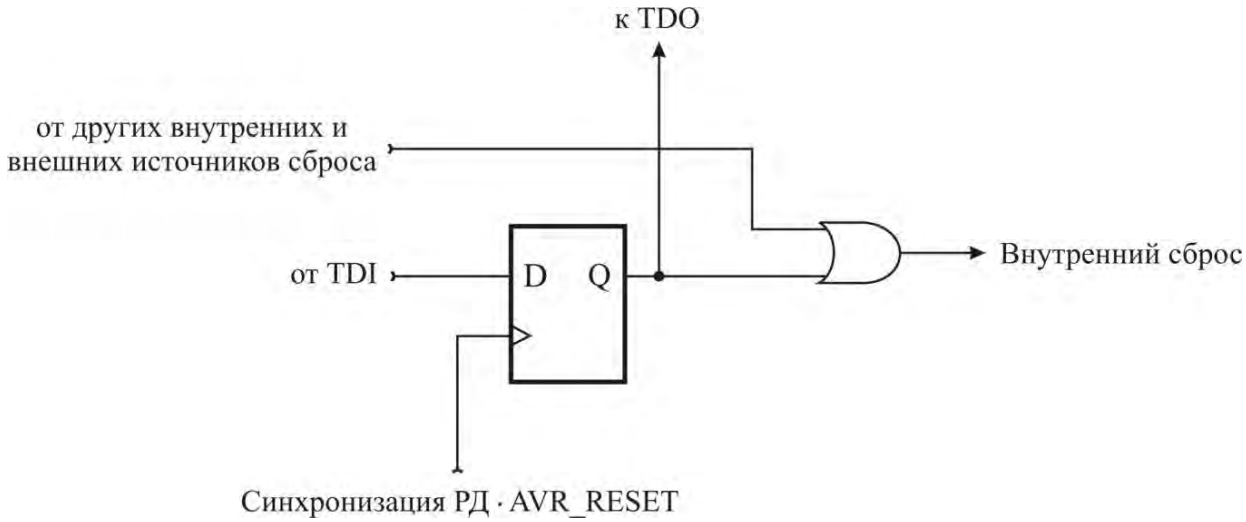


Рисунок 3.124 – Регистр сброса

Цепь периферийного сканирования

Цепь периферийного сканирования позволяет задавать и наблюдать логические уровни на цифровых линиях ввода-вывода, а также интерфейсе между цифровой и аналоговой логикой для аналоговых схем, имеющих внешние подключения.

Полное описание см. в 3.19.12 «Цепь периферийного сканирования».

3.19.10 Специфические JTAG-инструкции для периферийного сканирования

Регистр инструкций является 4-разрядным и поддерживает до 16 команд. Ниже приведены JTAG-инструкции, используемые в процессе периферийного сканирования. Обратите внимание, что дополнительная инструкция HIGHZ не реализована, но все выходы с тремя состояниями можно установить в высокоимпедансное состояние с помощью команды AVR_RESET, поскольку исходное положение для всех выводов порта – это третье состояние.

В данном руководстве пользователя, по определению, младший значащий бит LSB принимается и выдается первым во всех сдвиговых регистрах.

Код операции для каждой инструкции указывается в конце названия команды в шестнадцатеричном формате. Текст описания команды характеризует, который из регистров данных выбирается в качестве канала между TDI и TDO для каждой инструкции.

EXTEST; $0

Обязательная JTAG-инструкция для выбора цепи периферийного сканирования в качестве регистра данных, чтобы тестировать схемы, внешние по отношению к микроконтроллеру. Выводы порта, отключение подтягивающих резисторов, контроль выхода, выходные и входные данные, – все это является доступным в цепи сканирования. Для аналоговых схем, имеющих внешние подключения, в цепи сканирования расположен интерфейс между аналоговой и цифровой логикой. Содержимое фиксированных в защелке значений выходов цепи периферийного сканирования появится, как только в регистр инструкций JTAG загрузится команда EXTEST.

Активными состояниями являются:

- Захват РД: данные на внешних выводах отбираются в цепь периферийного сканирования.
- Сдвиг РД: внутренняя цепь сканирования сдвигается в соответствии с входом TCK.
- Обновление РД: данные из цепи сканирования выводятся на линии выхода.

IDCODE; $1

Дополнительная JTAG-инструкция для выбора 32-разрядного регистра идентификации устройства в качестве регистра данных. Регистр идентификации устройства состоит из номера версии, номера устройства и кода производителя. IDCODE является инструкцией, установленной по умолчанию после включения питания.

Активными состояниями являются:

- Захват РД: данные в регистре IDCODE отбираются в цепь периферийного сканирования.
- Сдвиг РД: цепь сканирования IDCODE сдвигается в соответствии с входом TCK.

SAMPLE_PRELOAD; $2

Обязательная JTAG-инструкция для предварительной загрузки выходных триггеров и просмотра мгновенного состояния линий ввода-вывода, не затрагивая работу системы. Однако выходные триггеры не подключены к этим линиям. Цепь периферийного сканирования выбирается в качестве регистра данных.

Активными состояниями являются:

- Захват РД: данные на внешних выводах отбираются в цепь периферийного сканирования.
- Сдвиг РД: цепь периферийного сканирования сдвигается в соответствии с входом TCK.
- Обновление РД: данные из цепи периферийного сканирования подаются на выходные триггеры. Однако выходные триггеры не подключены к линиям ввода-вывода.

AVR_RESET; $C

Специфическая открытая инструкция JTAG для принудительного перевода устройства в режим сброса. TAP-контроллер не сбрасывается этой командой. Одноразрядный регистр сброса выбирается в качестве регистра данных. Обратите внимание, что сброс будет активен до тех пор, пока в цепи сброса будет присутствовать логическая единица. Выход из этой цепи не защелкивается.

Активным состоянием является:

- Сдвиг РД: данные в регистр сброса подаются при тактировании TCK.

BYPASS; $F

Обязательная JTAG-инструкция для выбора обходного регистра в качестве регистра данных.

Активными состояниями являются:

- Захват РД: загружает логический ноль в обходной регистр.
- Сдвиг РД: ячейка обходного регистра между TDI и TDO смещается.

3.19.11 Регистр ввода-вывода, связанный с периферийным сканированием

Регистр управления и состояния микроконтроллера – MCUCSR

Регистр MCUCSR содержит биты управления общими функциями микроконтроллера и предоставляет информацию о том, какой из источников сброса вызвал сброс микроконтроллера.

Бит	7	6	5	4	3	2	1	0	
	JTD	-	-	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Чтение/ Запись	Ч/З	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	См. описание разрядов					

Разряд 7 – JTD: Включение интерфейса JTAG

Если значение этого бита равно нулю, то интерфейс JTAG включен, когда конфигурационный бит JTAGEN запрограммирован. Если значение бита JTD равно единице, интерфейс JTAG выключен. Чтобы избежать непреднамеренного выключения или включения JTAG-интерфейса, при изменении этого бита должна быть соблюдена временная последовательность: для изменения значения бита JTD прикладная программа должна дважды записывать в этот бит желаемое значение в течение четырех тактов.

Если интерфейс JTAG оставлен неподключенным к другим JTAG-схемам, бит JTD следует установить в единицу. Это необходимо для того, чтобы избежать статического тока на выводе TDO в JTAG-интерфейсе.

Разряд 4 – JTRF: Флаг сброса JTAG

Этот бит устанавливается, если сброс вызывается логической единицей в регистре сброса JTAG, записываемой с помощью JTAG-инструкции AVR_RESET. Данный бит сбрасывается во время сброса при подаче питания или путем записи логического нуля в этот флаг.

3.19.12 Цепь периферийного сканирования

Цепь периферийного сканирования позволяет задавать и наблюдать логические уровни на цифровых линиях ввода-вывода, а также на интерфейсе между цифровой и аналоговой логикой для аналоговых схем, имеющих внешние подключения.

Сканирование цифровых линий порта

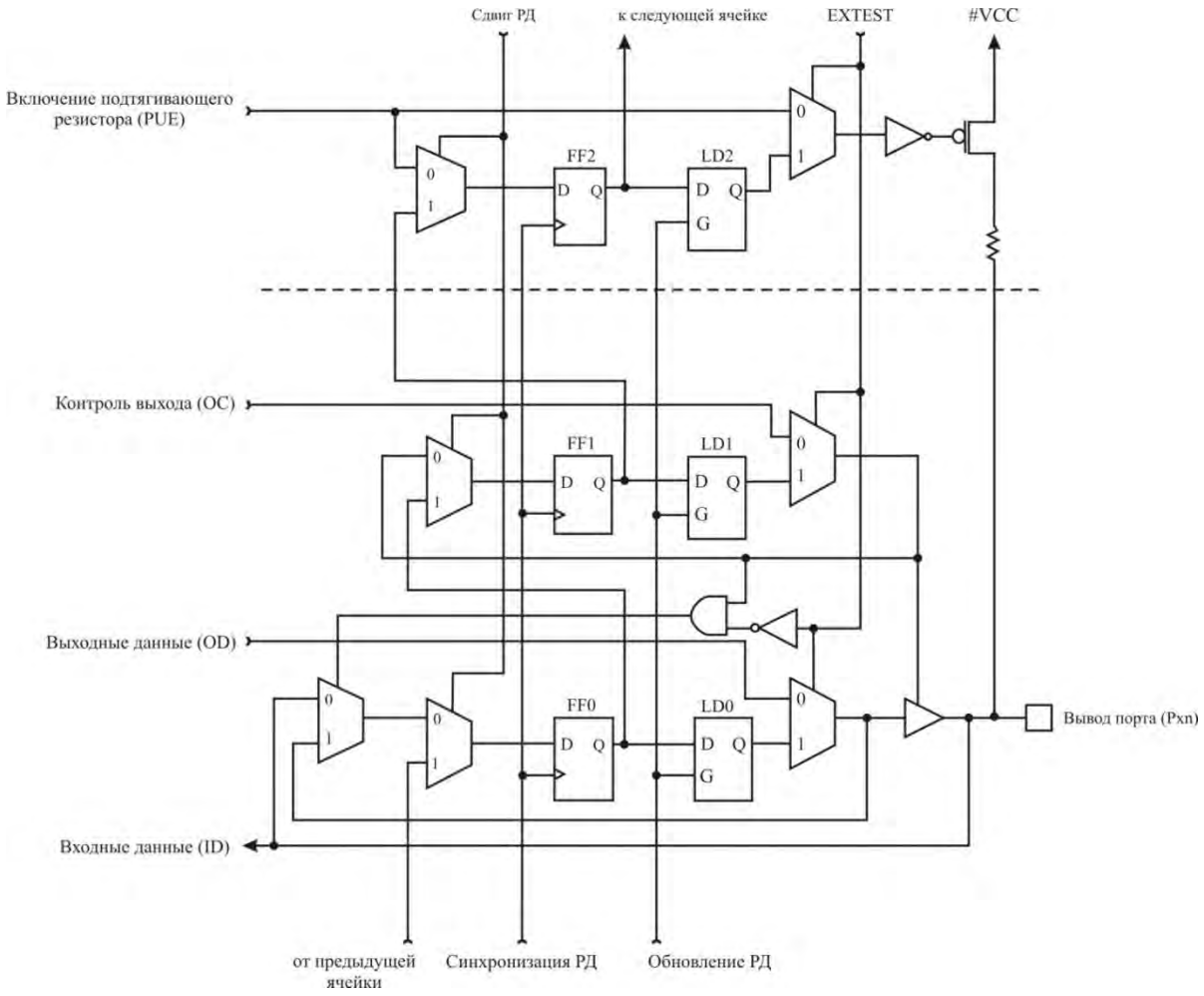


Рисунок 3.125 – Ячейка периферийного сканирования для двунаправленной линии порта с функцией подтягивающего резистора

На рисунке 3.125 показывается ячейка периферийного сканирования для двунаправленной линии порта с функцией подтягивающего резистора. Данная ячейка состоит из стандартной ячейки периферийного сканирования для включения подтягивающего резистора с функцией PUExn, и ячейки двунаправленной линии, где скомбинированы три сигнала: контроль выхода OCxn, выходные данные ODxn и входные данные IDxn (только в двухкаскадном сдвиговом регистре). Индексы порта и линии не используются в последующем описании.

Логика периферийного сканирования не показана на рисунках в данном руководстве пользователя. На рисунке 3.126 показана простая цифровая линия порта, как описано в подразделе 3.7 «Порты ввода-вывода». Область на рисунке 3.126, ограниченная штриховой линией, соответствует рисунку 3.125.

Когда альтернативная функция порта не используется, входные данные ID соответствуют значению регистра PINxn (ID не имеет синхронизатора), выходные данные соответствуют регистру PORT, контроль выхода соответствует регистру направления данных, а сигнал включения подтягивающего резистора PUExn соответствует логическому выражению PUD# – DDxn# – PORTxn.

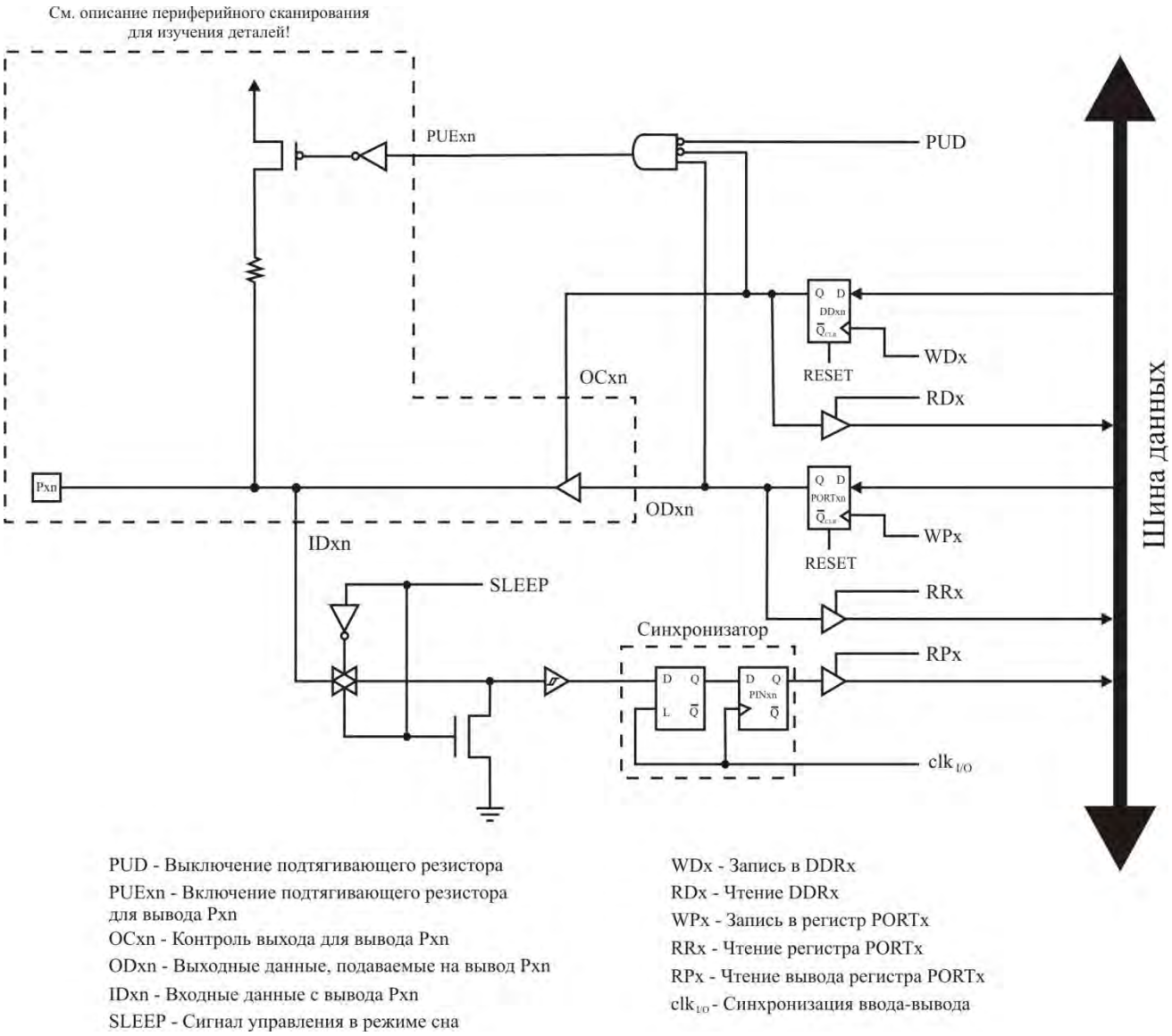


Рисунок 3.126 – Общая блок-схема линии порта

Цифровые альтернативные функции порта подключаются снаружи области на рисунке 3.126, ограниченной штриховой линией, для того чтобы заставить цепь сканирования считывать фактическое значение вывода. Для аналоговых функций существует прямое подключение от внешнего вывода к аналоговой схеме, а цепь сканирования помещается в интерфейс между цифровой логикой и аналоговой схемой.

Периферийное сканирование и двухпроводной интерфейс

Выводы двухпроводного интерфейса SCL и SDA имеют один дополнительный сигнал управления в цепи сканирования – сигнал включения двухпроводного интерфейса (TWIEN). Как показано на рисунке 3.127, сигнал TWIEN включает тристабильный буфер с управлением скоростью нарастания выходного напряжения, соединенный параллельно с обычными цифровыми выводами порта. Общая ячейка сканирования, как видно из рисунка 3.131, подключается к сигналу TWIEN.

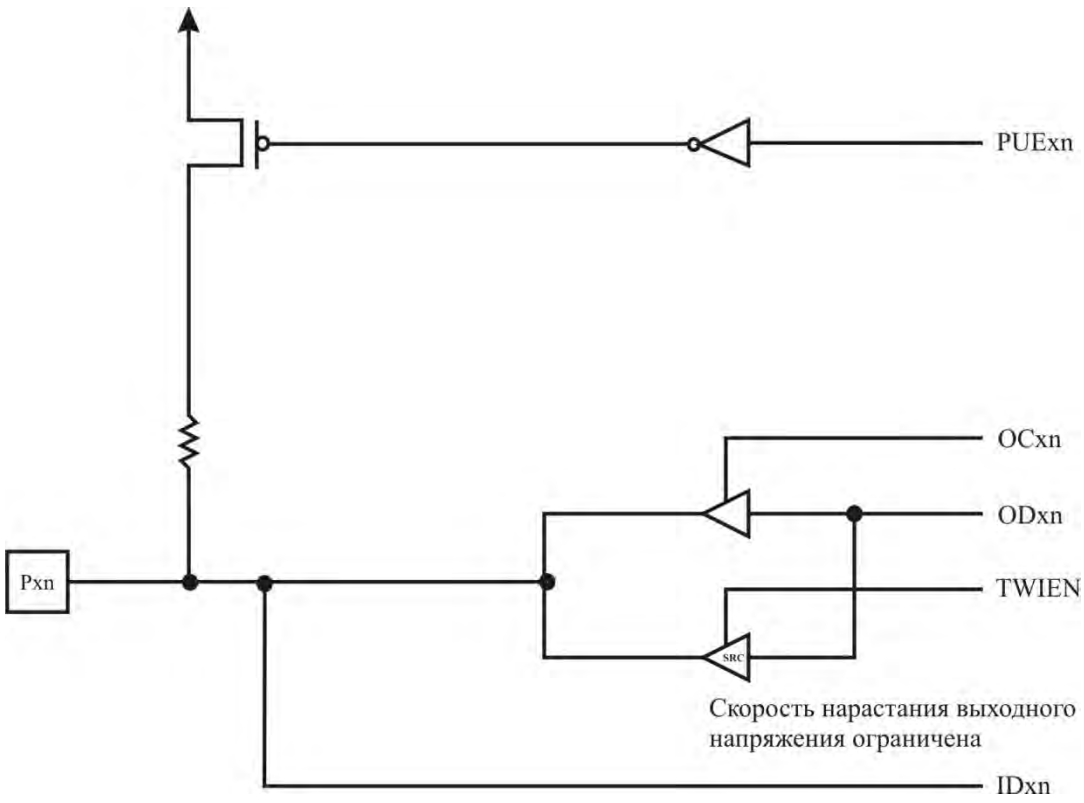


Рисунок 3.127 – Дополнительный сигнал сканирования для двухпроводного интерфейса

Примечания

1 Отдельная цепь сканирования для помехоподавляющего фильтра, подавляющего входные импульсы длительностью менее 50 нс, не предусмотрена. Для тестов коммутации достаточно обеспечения обычного сканирования цифровых выводов порта. Единственная причина для наличия сигнала TWIEN в канале сканирования – это возможность отключения буфера с управлением скоростью нарастания выходного напряжения во время выполнения периферийного сканирования.

2 Убедитесь, что сигналы OC и TWIEN не устанавливаются вместе, так как это приведет к конфликтной ситуации в управлении.

Сканирование вывода сброса

На вывод сброса может быть подано значение стандартной операции сброса или 12 В для высоковольтного параллельного программирования. Ячейка, предназначенная только для контроля, как показано на рисунке 3.128, предназначена как для сигнала сброса 5 В (RSTT), так и для сигнала сброса 12 В (RSTHV).

Сканирование выводов синхронизации

Микросхема имеет различные источники синхронизации, выбираемые конфигурационными битами. Источниками тактового сигнала являются: внутренний RC-генератор, внешний RC-генератор, внешний тактовый генератор, (высокочастотный) кварцевый генератор, низкочастотный кварцевый генератор и керамический резонатор.

На рисунке 3.129 показывается, каким образом каждый генератор с внешним подключением обслуживается в цепи сканирования. Сигнал включения обеспечивается общей ячейкой периферийного сканирования, в то время как выход генератора/синхронизатора подключается к ячейке, предназначенной только для наблюдения. Наряду с основной синхронизацией генератор таймера/счетчика сканируется таким же образом. Выход внутреннего RC-генератора не сканируется, так как этот генератор не имеет внешних подключений.

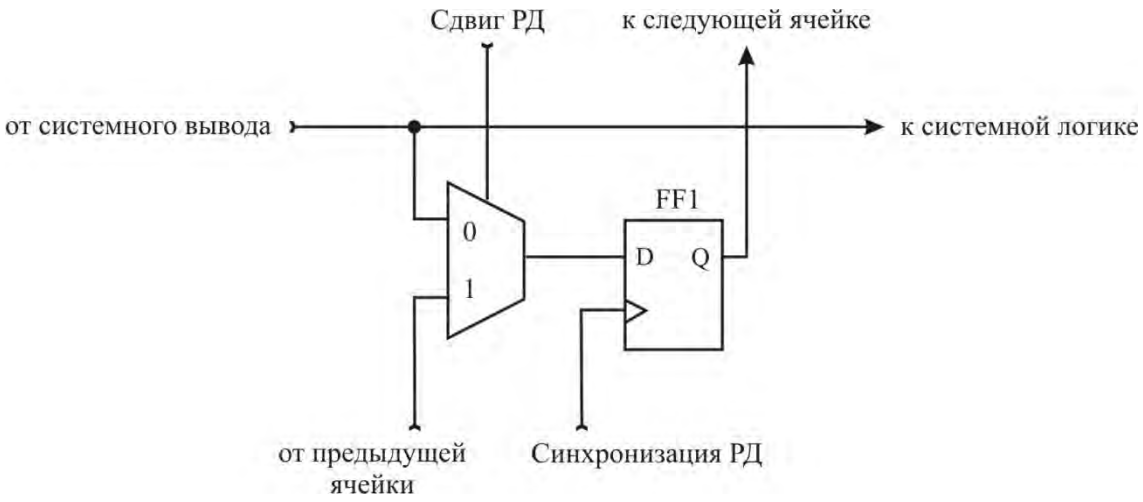


Рисунок 3.128 – Ячейка, предназначенная только для контроля

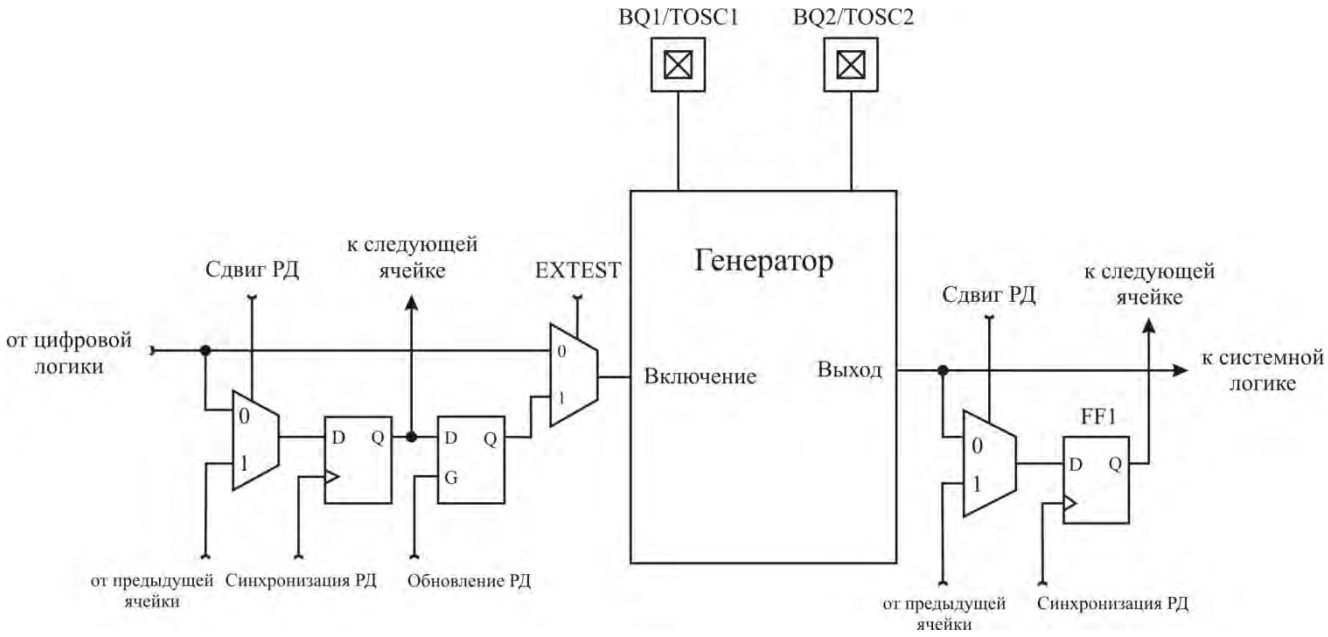


Рисунок 3.129 – Ячейки периферийного сканирования для генераторов и источников синхронизации

Данные в таблице 3.102 кратко характеризуют сканирующие регистры для вывода внешней синхронизации BQ1, генераторы с подключениями к BQ1/BQ2, а также генератор таймера/счетчика частотой 32 кГц.

Сканирование аналогового компаратора

Характерные сигналы компаратора, относящиеся к периферийному сканированию, отображены на рисунке 3.130. Ячейка периферийного сканирования с рисунка 3.131 подключена к каждому из этих сигналов. Данные сигналы описаны в таблице 3.103.

Нет необходимости использовать компаратор для тестирования коммутации, так как все аналоговые входы используются совместно с цифровым выводом порта.

Таблица 3.102 – Сигналы сканирования для генераторов

Сигнал включения	Сканируемая линия синхронизации	Источник синхронизации	Состояние сканируемой линии синхронизации, когда она не используется
EXTCLKEN	EXTCLK (BQ1)	Внешний тактовый генератор	0
OSCON	OSCCK	Внешний кварцевый генератор Внешний керамический резонатор	0
RCOSCEN	RCCK	Внешний RC-генератор	1
OSC32EN	OSC32CK	Низкочастотный внешний кварцевый генератор	0
TOSCON	TOSCK	Генератор таймера/счетчика частотой 32 кГц	0

Примечания

1 Не включайте за один раз более одного источника тактового сигнала в качестве основной синхронизации.

2 Сканирование выхода генератора дает непредсказуемые результаты, так как существует частотный дрейф между внутренним генератором и синхронизацией тестирования JTAG. По возможности предпочтительнее сканирование внешнего тактового генератора.

3 Параметры настройки синхронизации программируются конфигурационными битами. Поскольку конфигурационный бит не изменяется во время выполнения программы, настройка синхронизации считается фиксированной для заданного приложения. Пользователю рекомендуется сканировать тот же самый источник синхронизации, который используется в конечной системе. Сигналы включения поддерживаются в цепи сканирования из-за того, что системная логика может отключить источники синхронизации в спящих режимах, разъединяя тем самым выводы генератора и канал сканирования, если не обеспечить наличие этих сигналов. Конфигурационные биты INTCAP не поддерживаются в цепи периферийного сканирования, поэтому она не сможет просканировать генератор BQ, требующий внутренние конденсаторы для работы, если конфигурационный бит не будет правильно запрограммирован.

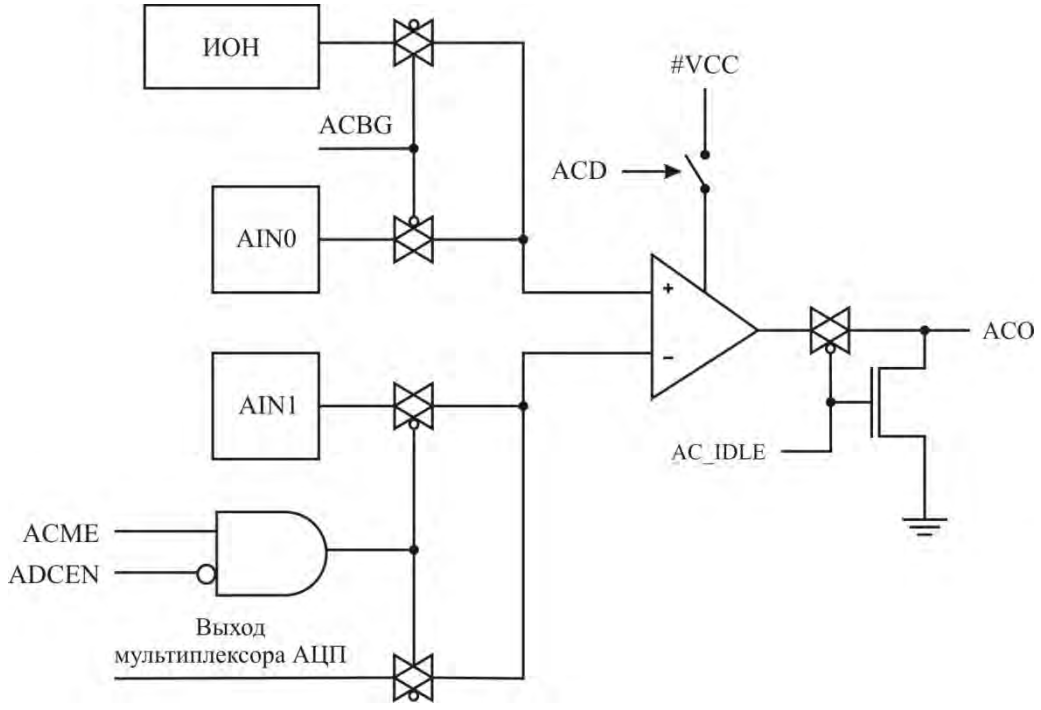


Рисунок 3.130 – Аналоговый компаратор

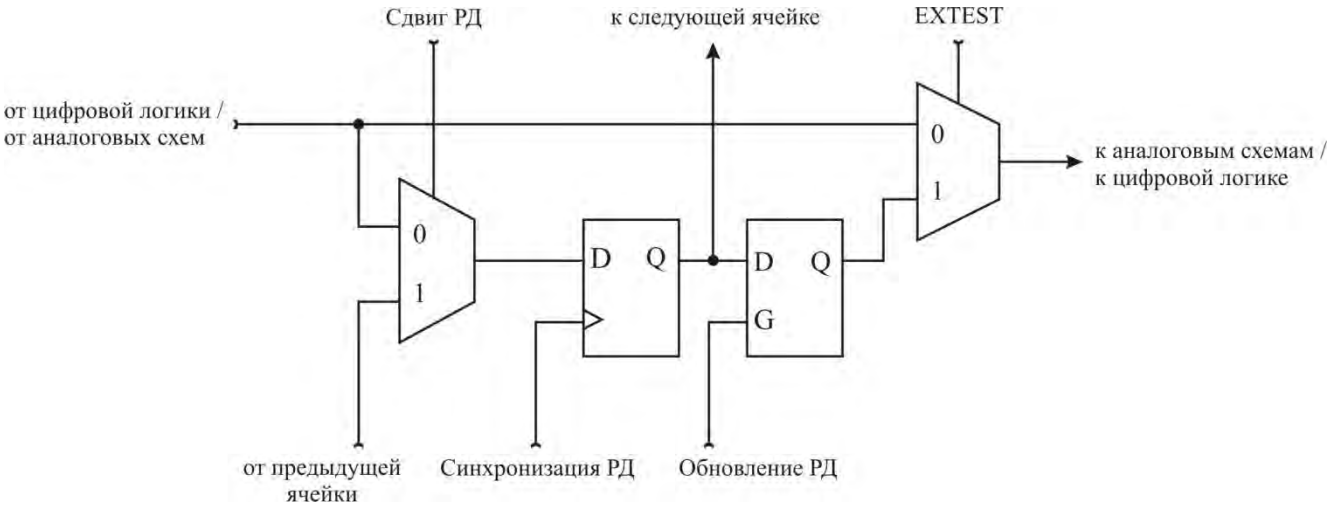


Рисунок 3.131 – Общая ячейка периферийного сканирования, используемая для сигналов компаратора и АЦП

Таблица 3.103 – Сигналы периферийного сканирования для аналогового компаратора

Название сигнала	Направление по отношению к компаратору	Описание	Рекомендованное состояние входа, когда устройство не используется	Выходные значения, когда используются рекомендованные входы
AC_IDLE	Вход	Выключает аналоговый компаратор, когда значение сигнала равно логической единице	1	Зависит от кода выполняемой команды
ACO	Выход	Выход аналогового компаратора	Станет входом для кода выполняемой команды	0
ACME	Вход	Использует выходной сигнал мультиплексора АЦП, когда значение сигнала ACME равно логической единице	0	Зависит от кода выполняемой команды
ACBG	Вход	Включение источника опорного напряжения	0	Зависит от кода выполняемой команды

Сканирование аналого-цифрового преобразователя

На рисунке 3.132 приведена структурная схема АЦП со всеми характерными сигналами управления и наблюдения. Ячейка периферийного сканирования с рисунка 3.128 подключена к каждому из этих сигналов. Нет необходимости использовать АЦП для тестирования коммутации, так как все аналоговые входы используются совместно с цифровым выводом порта.

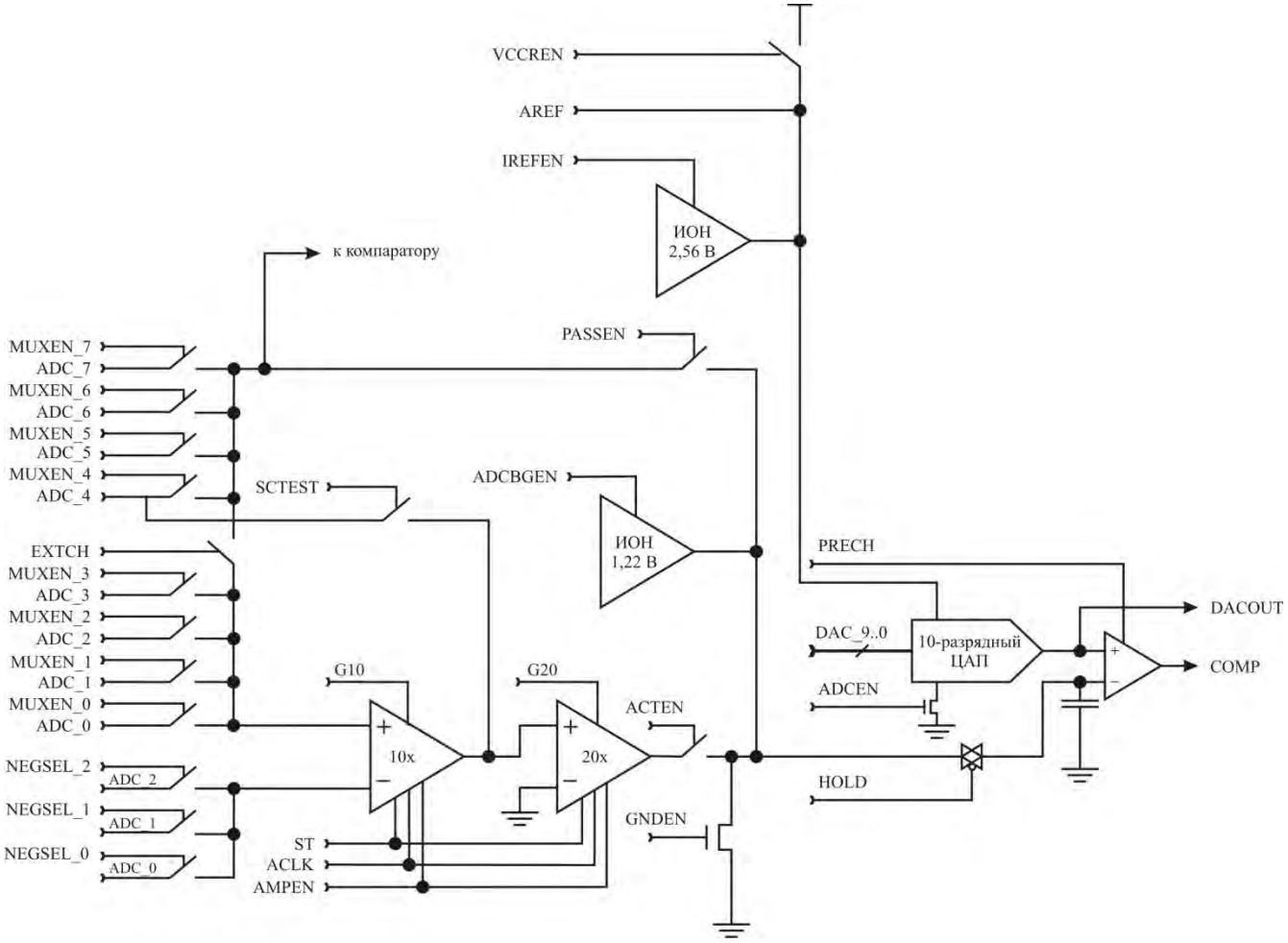


Рисунок 3.132 – Аналого-цифровой преобразователь

Сигналы периферийного сканирования для АЦП кратко описаны в таблице 3.104.

Таблица 3.104 – Сигналы периферийного сканирования для АЦП

Название сигнала	Направление по отношению к АЦП	Описание	Рекомендованное состояние входа, когда устройство не используется	Выходные значения, когда используются рекомендованные входы, а ЦПУ не использует АЦП
1	2	3	4	5
COMP	Выход	Выход компаратора	0	0
ACLK	Вход	Сигнал синхронизации для усилительных каскадов, реализованных в виде переключающих фильтров	0	0
ACTEN	Вход	Включение канала между усилительными каскадами и компаратором	0	0
ADCBGEN	Вход	Включение ИОН в качестве инвертирующего входа компаратора	0	0
ADCEN	Вход	Сигнал включения питания для АЦП	0	0

Продолжение таблицы 3.104

1	2	3	4	5
AMPEN	Вход	Сигнал включения питания для усилительных каскадов	0	0
DAC_9	Вход	Разряд 9 цифрового значения для ЦАП	1	1
DAC_8	Вход	Разряд 8 цифрового значения для ЦАП	0	0
DAC_7	Вход	Разряд 7 цифрового значения для ЦАП	0	0
DAC_6	Вход	Разряд 6 цифрового значения для ЦАП	0	0
DAC_5	Вход	Разряд 5 цифрового значения для ЦАП	0	0
DAC_4	Вход	Разряд 4 цифрового значения для ЦАП	0	0
DAC_3	Вход	Разряд 3 цифрового значения для ЦАП	0	0
DAC_2	Вход	Разряд 2 цифрового значения для ЦАП	0	0
DAC_1	Вход	Разряд 1 цифрового значения для ЦАП	0	0
DAC_0	Вход	Разряд 0 цифрового значения для ЦАП	0	0
EXTCH	Вход	Подключение каналов 0-3 АЦП к обходному пути вокруг усилительных каскадов	1	1
G10	Вход	Запуск десятикратного усилителя	0	0
G20	Вход	Запуск двадцатикратного усилителя	0	0
GNDEN	Вход	Заземление инвертирующего входа компаратора, когда значение данного сигнала равно логической единице	0	0
HOLD	Вход	Сигнал выборки и хранения. Аналоговый сигнал выборки при низком уровне. Сигнал хранения при высоком уровне. Если используются усилительные каскады, то данный сигнал должен активироваться, когда ACLK имеет высокий уровень	1	1
IREFEN	Вход	Включение ИОН в качестве сигнала опорного напряжения для ЦАП	0	0
MUXEN_7	Вход	Разряд 7 входного мультиплексора	0	0
MUXEN_6	Вход	Разряд 6 входного мультиплексора	0	0

Продолжение таблицы 3.104

1	2	3	4	5
MUXEN_5	Вход	Разряд 5 входного мультиплексора	0	0
MUXEN_4	Вход	Разряд 4 входного мультиплексора	0	0
MUXEN_3	Вход	Разряд 3 входного мультиплексора	0	0
MUXEN_2	Вход	Разряд 2 входного мультиплексора	0	0
MUXEN_1	Вход	Разряд 1 входного мультиплексора	0	0
MUXEN_0	Вход	Разряд 0 входного мультиплексора	1	1
NEGSEL_2	Вход	Входной мультиплексор для инвертирующего входа дифференциального сигнала, разряд 2	0	0
NEGSEL_1	Вход	Входной мультиплексор для инвертирующего входа дифференциального сигнала, разряд 1	0	0
NEGSEL_0	Вход	Входной мультиплексор для инвертирующего входа дифференциального сигнала, разряд 0	0	0
PASSEN	Вход	Включение обходного вентиля усилительных каскадов	1	1
PRECH	Вход	Предварительная зарядка выходной защелки компаратора (активным является низкий уровень)	1	1
SCTEST	Вход	Запуск тестирования переключающих фильтров. Выход десятикратного усилителя соединяется с выводом порта ADC_4	0	0
ST	Вход	Выход усилительных каскадов установится быстрее, если данный сигнал будет иметь высокий уровень первые два периода ACLK после того, как AMPEN примет значение, равное логической единице	0	0
VCCREN	Вход	Выбор #VCC в качестве аналогового напряжения питания ∩VCC	0	0
Примечание – Некорректные установки переключателей на рисунке 3.132 приведут к конфликту сигналов и могут повредить устройство. Существует несколько вариантов входа к схеме выборки и удержания на инвертирующем входе выходного компаратора. Убедитесь, что выбран только один канал: либо от вывода АЦП, либо от источника опорного напряжения, либо от вывода земли.				

Если не предполагается использование АЦП во время сканирования, то следует применять рекомендованные значения входов, указанные в таблице 3.104. Пользователю рекомендуется не использовать дифференциальные усилительные каскады во время сканирования. Усилительные каскады, основанные на переключающих фильтрах, требуют быстрой работы и точной синхронизации, чего трудно достичь, когда устройство используется в цепи сканирования. По этой причине подробности относительно работы дифференциальных усилительных каскадов не приведены.

Аналого-цифровой преобразователь основан на аналоговой схематике, отображенной на рисунке 3.132, с алгоритмом последовательных приближений, реализованным в цифровой логике. При использовании данного алгоритма в периферийном сканировании обычно существует проблема гарантировать, что приложенное аналоговое напряжение измеряется в некотором интервале. Это легко можно сделать, не запуская алгоритм последовательных приближений: подайте нижний предел на цифровые линии DAC[9–0], убедитесь, что выход компаратора имеет низкий уровень, затем подайте верхний предел на цифровые линии DAC[9–0] и проверьте, что выход компаратора имеет высокий уровень.

Нет необходимости использовать АЦП для тестирования коммутации, так как все аналоговые входы используются совместно с цифровым выводом порта.

При использовании АЦП следует помнить:

- Вывод порта для используемого канала АЦП должен быть сконфигурирован как вход с выключенным подтягивающим резистором во избежание конфликта сигналов.

- В нормальном режиме фиктивное преобразование (состоящее из десяти разрядов) выполняется при включении АЦП. Пользователю рекомендуется подождать по крайней мере 200 нс после включения АЦП перед управлением/наблюдением какого-либо сигнала АЦП или выполнить фиктивное преобразование перед использованием первого результата.

- Значения цифро-аналогового преобразователя должны быть стабильными в средней точке 0x200 при сигнале HOLD с низким уровнем (режим выборки).

В качестве примера можно рассмотреть задачу контроля входного сигнала 1,5 В ± 5 % в канале 3 АЦП, когда напряжение питания равно 5 В и сигнал AREF внешне подключен к #VCC.

Нижний предел равен: $\frac{1024 \cdot 1{,}5B \cdot 0{,}95}{5B} = 291 = 0x123$.

Верхний предел равен: $\frac{1024 \cdot 1{,}5B \cdot 1{,}05}{5B} = 323 = 0x143$.

Рекомендованные значения из таблицы 3.104 используются, если в алгоритме в таблице 3.105 не заданы другие значения. Следующая таблица показывает только значения ЦАП и вывода порта в цепи сканирования. Столбец «Действия» описывает, JTAG-инструкция какого рода должна использоваться перед заполнением регистра периферийного сканирования данными из последующих столбцов. Следует проводить верификацию просканированной информации, когда идет сканирование данных из того же ряда в таблице.

При использовании данного алгоритма временное лимитирование на сигнал HOLD ограничивает частоту синхронизации тестирования TCK. Поскольку алгоритм удерживает сигнал HOLD на высоком уровне в течение пяти шагов, частота синхронизации тестирования TCK должна быть, по крайней мере, в пять раз больше числа битов сканирования, деленных на максимальное время удержания $t_{hold,max}$.

Таблица 3.105 – Алгоритм для использования АЦП

Шаг	Действия	ADCEN	ЦАП	MUXEN	HOLD	PRECH	Данные с линии PA3	Управление линией PA3	Включение подтягивающего резистора на линии PA3
1	SAMPLE_PRELOAD	1	0x200	0x08	1	1	0	0	0
2	EXTEST	1	0x200	0x08	0	1	0	0	0
3		1	0x200	0x08	1	1	0	0	0
4		1	0x123	0x08	1	1	0	0	0
5		1	0x123	0x08	1	0	0	0	0
6	Проверьте бит COMP, он должен быть равен нулю	1	0x200	0x08	1	1	0	0	0
7		1	0x200	0x08	0	1	0	0	0
8		1	0x200	0x08	1	1	0	0	0
9		1	0x143	0x08	1	1	0	0	0
10		1	0x143	0x08	1	0	0	0	0
11	Проверьте бит COMP, он должен быть равен единице	1	0x200	0x08	1	1	0	0	0

3.19.13 Последовательность периферийного сканирования в ИС 1887ВЕ7Т

В таблице 3.106 поясняется последовательность сканирования между TDI и TDO, когда в качестве канала данных выбрана цепь периферийного сканирования. Бит 0 – это младший значащий бит, LSB, который первым сдвигается влево и вправо. Последовательность сканирования соответствует порядку расположения выводов насколько возможно. Поэтому, биты порта А сканируются в порядке, противоположном по отношению к битам других портов. Исключениями из правил являются цепи сканирования для аналоговых схем, которые являются старшими значащими битами цепи сканирования независимо от того, к какому физическому выводу они подключены. Рассмотрим рисунок 3.125. Данные соответствуют FF0, Pxn. Управление соответствует FF1 и Pxn. Включение подтягивающего резистора соответствует FF2. Биты 2 ,3, 4 и 5 порта С не расположены в цепи сканирования, поскольку эти линии служат выводами TAP, когда включен JTAG-интерфейс.

Таблица 3.106 – Последовательность периферийного сканирования в ИС 1887ВЕ7Т

Номер бита	Название сигнала	Модуль микроконтроллера
204	AC_IDLE	Компаратор
203	ACO	
202	ACME	
201	AINBG	
200	COMP	Аналого-цифровой преобразователь
199	PRIVATE_SIGNAL1*	
198	ACLK	
167	ACTEN	
196	PRIVATE_SIGNAL2*	
195	ADCBGEN	
194	ADCEN	
193	AMPEN	
192	DAC_9	
191	DAC_8	
190	DAC_7	
189	DAC_6	
188	DAC_5	

Продолжение таблицы 3.106

Номер бита	Название сигнала	Модуль микроконтроллера
187	DAC_4	Аналого-цифровой преобразователь
186	DAC_3	
185	DAC_2	
184	DAC_1	
183	DAC_0	
182	EXTCH	
181	G10	
180	G20	
179	GNDEN	
178	HOLD	
177	IREFEN	
176	MUXEN_7	
175	MUXEN_6	
174	MUXEN_5	
173	MUXEN_4	
172	MUXEN_3	
171	MUXEN_2	
170	MUXEN_1	
169	MUXEN_0	
168	NEGSEL_2	
167	NEGSEL_1	
166	NEGSEL_0	
165	PASSEN	
164	PRECH	
163	SCTEST	
162	ST	
161	VCCREN	
160	PEN	Разрешение программирования (только наблюдение)
159	PE0.Data	Порт E
158	PE0.Control	
157	PE0.Pullup_Enable	
156	PE1.Data	
155	PE1.Control	
154	PE1.Pullup_Enable	
153	PE2.Data	
152	PE2.Control	
151	PE2.Pullup_Enable	
150	PE3.Data	
149	PE3.Control	
148	PE3.Pullup_Enable	
147	PE4.Data	
146	PE4.Control	
145	PE4.Pullup_Enable	
144	PE5.Data	
143	PE5.Control	
142	PE5.Pullup_Enable	
141	PE6.Data	
140	PE6.Control	
139	PE6.Pullup_Enable	
138	PE7.Data	
137	PE7.Control	
136	PE7.Pullup_Enable	

Продолжение таблицы 3.106

Номер бита	Название сигнала	Модуль микроконтроллера
135	PB0.Data	Порт B
134	PB0.Control	
133	PB0.Pullup_Enable	
132	PB1.Data	
131	PB1.Control	
130	PB1.Pullup_Enable	
129	PB2.Data	
128	PB2.Control	
127	PB2.Pullup_Enable	
126	PB3.Data	
125	PB3.Control	
124	PB3.Pullup_Enable	
123	PB4.Data	
122	PB4.Control	
121	PB4.Pullup_Enable	
120	PB5.Data	
119	PB5.Control	
118	PB5.Pullup_Enable	
117	PB6.Data	
116	PB6.Control	
115	PB6.Pullup_Enable	
114	PB7.Data	
113	PB7.Control	
112	PB7.Pullup_Enable	
111	PG3.Data	Порт G
110	PG3.Control	
109	PG3.Pullup_Enable	
108	PG4.Data	
107	PG4.Control	
106	PG4.Pullup_Enable	
105	TOSC	Генератор таймера/счетчика с частотой 32 кГц
104	TOSCON	
103	RSTT	Логика сброса (только наблюдение)
102	RSTHV	
101	EXTCLKEN	Разрешение сигналов для основной синхронизации/осцилляторов
100	OSCON	
99	RCOSCEN	
98	OSC32EN	
97	EXTCLK (BQ1)	Вход синхронизации и осцилляторов для основного тактового сигнала (только для наблюдения)
96	OSCCK	
95	RCCK	
94	OSC32CK	
93	TWIEN	Двухпроводной интерфейс
92	PD0.Data	Порт D
91	PD0.Control	
90	PD0.Pullup_Enable	
89	PD1.Data	
88	PD1.Control	
87	PD1.Pullup_Enable	
86	PD2.Data	
85	PD2.Control	
84	PD2.Pullup_Enable	
83	PD3.Data	

Продолжение таблицы 3.106

Номер бита	Название сигнала	Модуль микроконтроллера
82	PD3.Control	
81	PD3.Pullup_Enable	
80	PD4.Data	
79	PD4.Control	
78	PD4.Pullup_Enable	
77	PD5.Data	
76	PD5.Control	
75	PD5.Pullup_Enable	
74	PD6.Data	
73	PD6.Control	
72	PD6.Pullup_Enable	
71	PD7.Data	
70	PD7.Control	
69	PD7.Pullup_Enable	
68	PG0.Data	Порт G
67	PG0.Control	
66	PG0.Pullup_Enable	
65	PG1.Data	
64	PG1.Control	
63	PG1.Pullup_Enable	
62	PC0.Data	Порт C
61	PC0.Control	
60	PC0.Pullup_Enable	
59	PC1.Data	
58	PC1.Control	
57	PC1.Pullup_Enable	
56	PC2.Data	
55	PC2.Control	
54	PC2.Pullup_Enable	
53	PC3.Data	
52	PC3.Control	
51	PC3.Pullup_Enable	
50	PC4.Data	
49	PC4.Control	
48	PC4.Pullup_Enable	
47	PC5.Data	
46	PC5.Control	
45	PC5.Pullup_Enable	
44	PC6.Data	
43	PC6.Control	
42	PC6.Pullup_Enable	
41	PC7.Data	
40	PC7.Control	
39	PC7.Pullup_Enable	
38	PG2.Data	Порт G
37	PG2.Control	
36	PG2.Pullup_Enable	
35	PA7.Data	Порт A
34	PA7.Control	
33	PA7.Pullup_Enable	
32	PA6.Data	
31	PA6.Control	
30	PA6.Pullup_Enable	

Продолжение таблицы 3.106

Номер бита	Название сигнала	Модуль микроконтроллера
29	PA5.Data	Порт A
28	PA5.Control	
27	PA5.Pullup_Enable	
26	PA4.Data	
25	PA4.Control	
24	PA4.Pullup_Enable	
23	PA3.Data	
22	PA3.Control	
21	PA3.Pullup_Enable	
20	PA2.Data	
19	PA2.Control	
18	PA2.Pullup_Enable	
17	PA1.Data	
16	PA1.Control	
15	PA1.Pullup_Enable	
14	PA0.Data	
13	PA0.Control	
12	PA0.Pullup_Enable	
11	PF3.Data	Порт F
10	PF3.Control	
9	PF3.Pullup_Enable	
8	PF2.Data	
7	PF2.Control	
6	PF2.Pullup_Enable	
5	PF1.Data	
4	PF1.Control	
3	PF1.Pullup_Enable	
2	PF0.Data	
1	PF0.Control	
0	PF0.Pullup_Enable	

_____

* PRIVATE_SIGNAL1, PRIVATE_SIGNAL2 всегда должны сканироваться как ноль.

3.19.14 Файлы языкового описания периферийного сканирования

Файлы языкового описания периферийного сканирования (BSDL) характеризуют устройства с возможностью периферийного сканирования в стандартном формате, используемом автоматизированными программами генерации тестовых кодов. Последовательность и функция битов в регистре данных периферийного сканирования включена в данное описание.

3.20 Самопрограммирование из секции загрузчика с поддержкой чтения во время записи

Область секции загрузчика обеспечивает механизм самопрограммирования с поддержкой чтения во время записи, применяемый для загрузки программного кода с удаленного компьютера и в удаленный компьютер самим микроконтроллером. Эта особенность позволяет выполнять обновление памяти программ с помощью программы-загрузчика, расположенной в одноименной секции. Программа-загрузчик может использовать любой доступный интерфейс передачи данных и связанный с ним протокол, чтобы считывать код и записывать его в память программ или считывать код из памяти программ. Программа-

загрузчик может изменять содержимое любой ячейки памяти программ, включая область секции загрузчика. Таким образом, программа-загрузчик может модифицироваться и даже стирать свой код, если она больше не требуется. Размер области секции загрузчика конфигурируется с помощью специальных битов. Загрузчик имеет два отдельных набора битов защиты загрузки, которые могут быть установлены независимо друг от друга. Это предоставляет пользователю гибкость в выборе различных уровней безопасности.

3.20.1 Особенности загрузчика

- Самопрограммирование с поддержкой чтения во время записи.
- Гибкий размер памяти начального загрузчика.
- Высокая безопасность (индивидуальные биты защиты загрузки для гибкой настройки уровня безопасности).
- Специальный конфигурационный бит для выбора вектора сброса.
- Оптимизированный размер страницы.
- Эффективный программный алгоритм.
- Эффективная поддержка операции «Чтение-Модификация-Запись».

Примечание – Страница представляет собой сегмент в памяти программ, состоящий из нескольких байтов (см. таблицу 3.124), используемых во время программирования. Структура страницы не влияет на нормальную работу.

3.20.2 Секция прикладной программы и секция загрузчика в памяти программ

Память программ делится на две основные части: секция прикладной программы и секция загрузчика (см. рисунок 3.134). Размер этих секций определяется установкой конфигурационных битов BOOTSZ, как показано в таблице 3.112 и на рисунке 3.134. Данные секции могут иметь различный уровень безопасности, поскольку у каждой из них свой набор битов защиты.

Секция прикладной программы

Секция прикладной программы представляет собой раздел памяти программ, который используется для хранения кода приложения. Уровень безопасности для данного раздела может быть выбран с помощью соответствующих битов защиты BLB02 и BLB01 (см. таблицу 3.108). Секция прикладной программы не может хранить код загрузчика, так как выполнение команды SPM из секции приложения запрещено.

Секция загрузчика (BLS)

В то время как секция прикладной программы используется для хранения кода прикладной программы, программа-загрузчик должна размещаться в секции BLS, поскольку команда SPM может запускать программирование, только когда она выполняется из секции BLS. Команда SPM может получать доступ ко всему пространству памяти программ, включая сам раздел BLS. Уровень безопасности для секции загрузчика может быть выбран с помощью соответствующих битов защиты BLB12 и BLB11 (см. таблицу 3.109).

3.20.3 Области памяти программ с поддержкой чтения во время записи и без поддержки данной функции

Поддерживает ли ЦПУ функцию чтения во время записи или останавливается в период обновления загрузчика, зависит от адреса программируемой ячейки. Помимо двух секций, конфигурируемых битами BOOTSZ, память программ также делится на две фиксированные области: область с поддержкой чтения во время записи (RWW) и область без поддержки данной функции (NRWW). Граница между данными областями обозначена в таблице 3.113 и на рисунке 3.134. Основные отличия между областями RWW и NRWW:

- При стирании информации или выполнении записи в страницу, расположенную внутри области RWW, область NRWW доступна для чтения.

- При стирании информации или выполнении записи в страницу, расположенную внутри области NRWW, ЦПУ останавливается до окончания этой операции.

Обратите внимание, что во время изменения содержимого страницы памяти программ, расположенной в области RWW, чтение этой области запрещено. Определение «область с поддержкой чтения во время записи» (RWW) означает, что при записи в данную область памяти программ, процессор может выполнять чтение кода программы из другой области (NRWW).

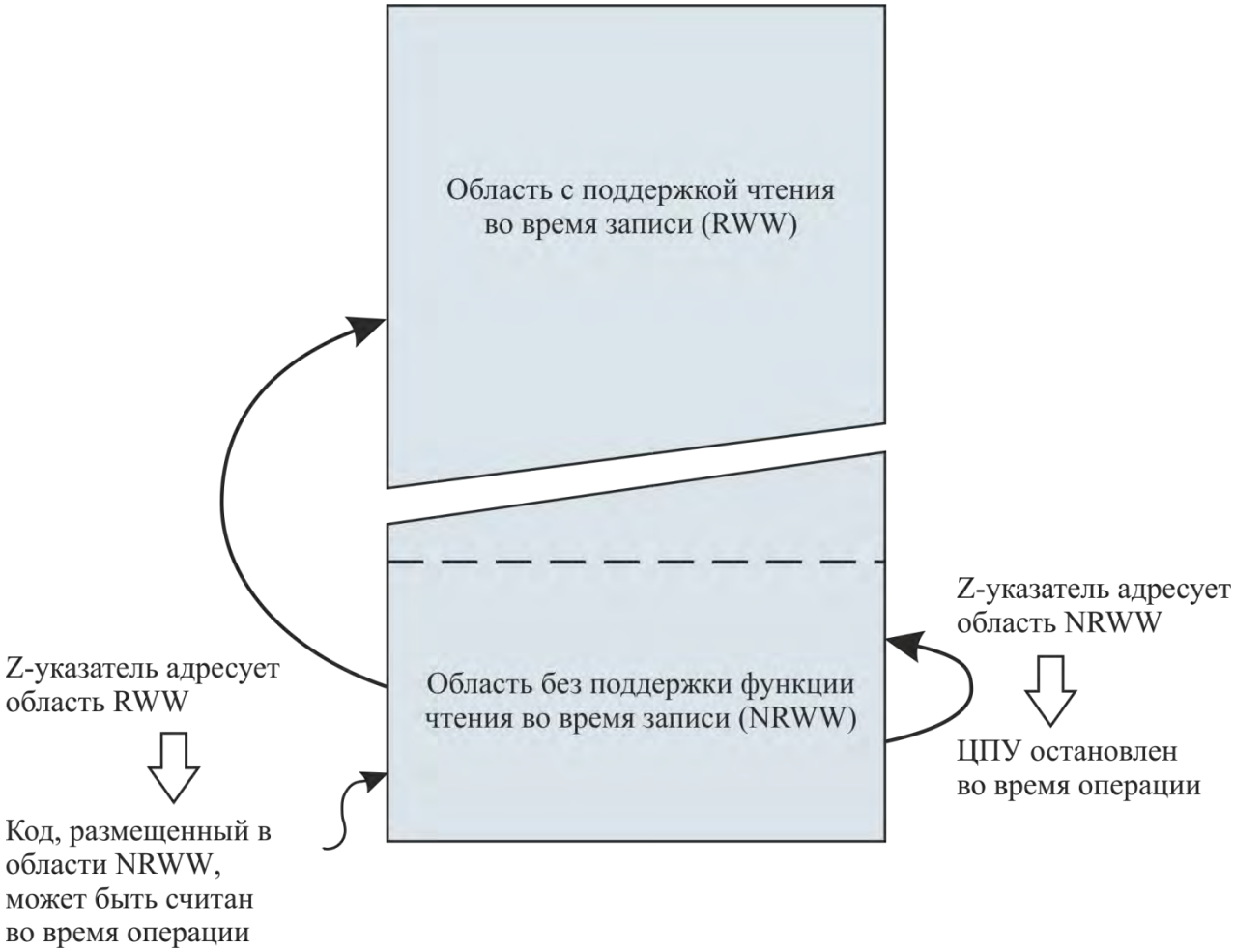


Рисунок 3.133 – Области памяти программ с поддержкой чтения во время записи и без поддержки данной функции

Область с поддержкой чтения во время записи – RWW

Когда идет программирование страницы памяти программ, расположенной внутри области RWW, разрешено считывать код памяти программ расположенный в области NRWW. Попытка обратиться в процессе программирования к коду, находящемуся в области RWW (в результате выполнения команд CALL/JMP/LPM или в результате прерывания), может привести к непредсказуемым последствиям. Во избежание этого следует либо запретить прерывания, либо перенести таблицу векторов прерываний в секцию загрузчика, который всегда находится в области NRWW. Флаг занятости области RWW (RWWSB) в регистре состояния и управления записью в память программ (SPMCSR) будет считываться как логическая единица, пока область RWW запрещена для чтения. После окончания программирования бит RWWSB должен быть сброшен программно перед операцией чтения кода из области RWW. См. ниже подпункт «Регистр состояния и управления записью в па-

мять программ – SPMCSR» для изучения деталей относительно того, как сбрасывать бит RWWSB.

Область без поддержки чтения во время записи – NRWW

Код, расположенный в области NRWW, доступен для чтения, когда идет изменение страницы памяти программ, расположенной в области RWW. При изменении содержимого памяти программ в области NRWW процессор останавливается до завершения операции (см. таблицу 3.107).

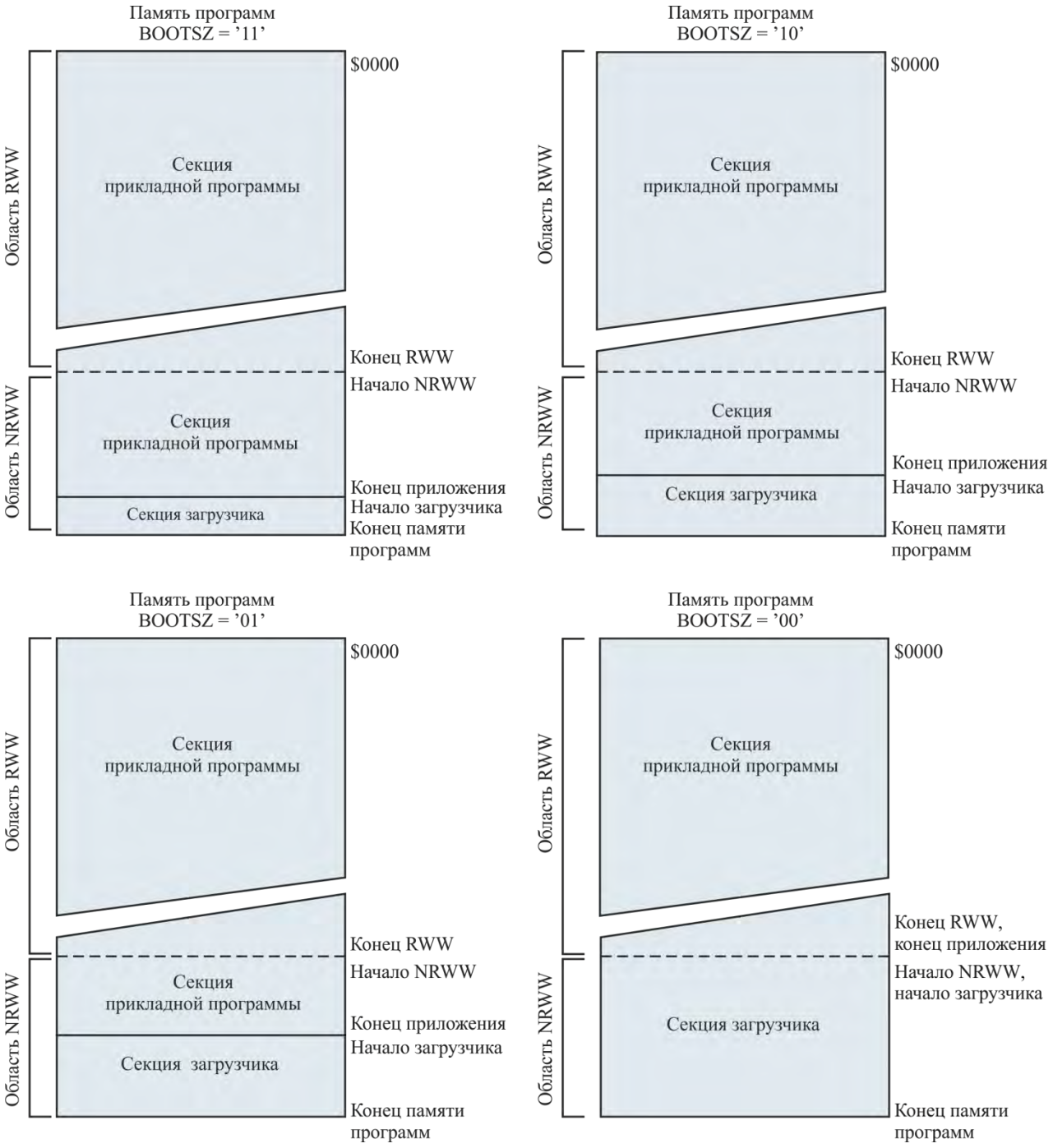


Рисунок 3.134 – Области памяти

Примечание – Параметры, показанные на данном рисунке, представлены в таблице 3.112.

Таблица 3.107 – Особенности функции чтения во время записи

Какая область имеет адрес с Z-указателем во время программирования?	Какая область доступна для чтения во время программирования?	Останавливается ли ЦПУ?	Поддерживается ли функция чтения во время записи?
Область RWW	Область NRWW	Нет	Да
Область NRWW	Никакой	Да	Нет

3.20.4 Биты защиты загрузчика

Когда в загрузчике нет необходимости, все пространство памяти программ доступно для кода прикладной программы. Загрузчик имеет два отдельных набора битов защиты, которые могут быть установлены независимо друг от друга. Это предоставляет пользователю гибкость в выборе различных уровней безопасности.

Пользователь может выбрать следующие варианты:

- Защита всего пространства памяти программ от изменения микроконтроллером.
- Защита только секции загрузчика от изменения микроконтроллером.
- Защита только секции прикладной программы от изменения микроконтроллером.
- Разрешить полный доступ ко всей памяти программ.

Подробности относительно установки того или иного уровня безопасности см. в таблицах 3.108 и 3.109. Биты защиты можно установить программно или в режиме последовательного или параллельного программирования, но сбросить их можно только с помощью команды стирания информации во всем кристалле.

Таблица 3.108 – Режимы защиты секции прикладной программы, выбираемые через биты BLB0

Режим BLB0	BLB02	BLB01	Уровень безопасности
1	1	1	Нет никаких ограничений для доступа к коду, расположенному в секции прикладной программы с помощью инструкций SPM или LPM
2	1	0	Запись в секцию прикладной программы с помощью команды SPM запрещена
3	0	0	Запись в секцию прикладной программы командой SPM запрещена. Команда LPM, вызываемая из секции загрузчика, не может выполнять чтение из секции прикладной программы. Если таблица векторов прерываний расположена в секции загрузчика, то при выполнении кода из секции прикладной программы прерывания запрещены.
4	0	1	Чтение из секции прикладной программы командой LPM, вызываемой из секции загрузчика, запрещено. Если таблица векторов прерываний расположена в секции загрузчика, то при выполнении кода из секции прикладной программы прерывания запрещены.
Примечание – «1» означает незапрограммированное состояние, «0» означает запрограммированное состояние.			

Таблица 3.109 – Режимы защиты секции загрузчика, выбираемые через биты BLB1

Режим BLB1	BLB12	BLB11	Уровень безопасности
1	1	1	Нет никаких ограничений для доступа к коду, расположенному в секции загрузчика, с помощью инструкций SPM или LPM
2	1	0	Запись в секцию загрузчика с помощью команды SPM запрещена
3	0	0	Запись в секцию загрузчика командой SPM запрещена. Команда LPM, вызываемая из секции прикладной программы, не может выполнять чтение из секции загрузчика. Если таблица векторов прерываний расположена в секции прикладной программы, то при выполнении кода из секции загрузчика прерывания запрещены.
4	0	1	Чтение из секции загрузчика командой LPM, вызываемой из секции прикладной программы, запрещено. Если таблица векторов прерываний расположена в секции прикладной программы, то при выполнении кода из секции загрузчика прерывания запрещены.
Примечание – «1» означает незапрограммированное состояние, «0» означает запрограммированное состояние.			

3.20.5 Вход в программу загрузчика

Вход в программу загрузчика происходит с помощью команды перехода или команды вызова из прикладной программы. Вход может быть инициирован сигналом запуска, таким как команда, полученная через USART или интерфейс SPI. Альтернативно, конфигурационный бит BOOTRST можно запрограммировать так, что вектор сброса будет указывать на адрес начала секции загрузчика после выполнения сброса. В этом случае код загрузчика запускается после сброса. После того как загрузится код приложения, программа загрузчика может начать выполнение этого кода. Обратите внимание, что конфигурационные биты не могут быть изменены самим микроконтроллером. Это значит, что после однократного программирования конфигурационного бита BOOTRST вектор сброса всегда будет указывать на адрес загрузчика, а конфигурационный бит может быть изменен только через последовательный или параллельный интерфейс программирования.

Таблица 3.110 – Конфигурационный бит BOOTRST

BOOTRST	Адрес сброса
1	Вектор сброса указывает на адрес начала области прикладной программы (адрес $0000)
0	Вектор сброса указывает на адрес начала секции загрузчика (см. таблицу 3.112)
Примечание – «1» означает незапрограммированное состояние, «0» означает запрограммированное состояние.	

Регистр состояния и управления записью в память программ – SPMCSR

Регистр состояния и управления записью в память программ содержит управляющие биты, отвечающие за выполнение тех или иных операций при вызове команды SPM.

Бит	7	6	5	4	3	2	1	0	
	SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	SPMCSR
Чтение/ Запись	Ч/З	Ч	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – SPMIE: Разрешение прерывания по готовности SPM

Если SPMIE = 1 и установлен бит I в регистре статуса, то прерывание по готовности SPM будет разрешено. Оно будет выполняться до тех пор, пока не сбросится бит SPMEN в регистре SPMCSR.

Разряд 6 – RWWSB: Бит занятости сегмента RWW

При запуске операции самопрограммирования (запись в страницу или стирание информации в странице) области RWW бит RWWSB устанавливается аппаратно. Когда бит RWWSB установлен, нельзя получить доступ к данным в области RWW. Сброс флага осуществляется либо программно, записью логической единицей в бит RWWSRE по окончании операции, либо аппаратно, при запуске операции загрузки страницы.

Разряд 5 – Резервный бит

Данный бит является резервным и всегда считывается как ноль.

Разряд 4 – RWWSRE: Разрешение чтения области RWW

При программировании (записи в страницу или стирании информации в странице) области RWW данный бит заблокирован для чтения (бит RWWSB устанавливается аппаратно). Чтобы отменить запрет на доступ к области RWW, пользовательская программа должна дождаться завершения программирования (будет сброшен бит SPMEN). Затем одновременно установить биты SPMEN и RWWSRE и в течение четырех тактов после установки этих битов запуститькоманду SPM. Разрешение доступа к области RWW может осуществляться только после завершения операции программирования (после сброса флага SPMEN). Если выполнить запись в бит RWWSRE во время загрузки памяти программ, то операция загрузки прервется, а загруженные данные будут утеряны.

Разряд 3 – BLBSET: Изменение ячеек защиты загрузчика

При одновременной установке этого бита и бита SPMEN, команда SPM, запущенная в течение последующих четырех тактов, выполнит программирование защитных ячеек загрузчика в соответствии с содержимым регистра R0. Данные в регистре R1 и адрес в Z-указателе не учитываются. Бит BLBSET будет сброшен автоматически по окончании программирования битов защиты, или если в течение четырех тактов не выполнится команда SPM.

Команда LPM, запущенная в течение четырех тактов после установки битов BLBSET и SPMEN, выполнит чтение либо конфигурационных битов, либо ячеек защиты (в зависимости от значения бита Z0 регистра Z) См. ниже подпункт «Считывание конфигурационных битов и битов защиты из программы» для изучения деталей.

Разряд 2 – PGWRT: Запись страницы

При одновременной установке этого бита и бита SPMEN команда SPM, запущенная в течение последующих четырех тактов, выполнит запись страницы памяти программ из временного буфера. Данные в регистрах R1 и R0 не учитываются. Сброс бита PGWRT осуществляется аппаратно по окончании записи страницы, либо если в течение четырех тактов после установки битов PGWRT и SPMEN не будет выполнена команда SPM. При записи страницы в область NRWW центральный процессор останавливается на время выполнения операции.

Разряд 1 – PGERS: Стирание страницы

При одновременной установке этого бита и бита SPMEN команда SPM, запущенная в течение последующих четырех тактов, выполнит стирание страницы памяти программ. Адрес страницы должен быть загружен в старший байт регистра Z (R31). Данные в регистрах R1 и R0 не учитываются. Сброс бита PGERS осуществляется аппаратно по окончании стирания страницы либо, если в течение четырех тактов после установки битов PGERS и

SPMEN не будет выполнена команда SPM. При стирании страницы в области NRWW центральный процессор останавливается на время выполнения операции.

Разряд 0 – SPMEN: Разрешение выполнения команды SPM

Установка этого бита разрешает запуск команды SPM в течение последующих четырех тактов. Если бит SPMEN устанавливается одновременно с одним из битов RWWSRE, BLBSET, PGWRT или RGERS выполняется операция, определяемая этим битом (см. описание выше). Если устанавливается только бит SPMEN, выполняется сохранение содержимого регистров R1 и R0 во временном буфере по адресу, находящемуся в регистре Z (младший значащий разряд регистра игнорируется). Сброс бита SPMEN осуществляется аппаратно после завершения операции либо, если в течение четырех тактов установки бита SPMEN не будет выполнена команда SPM. Во время записи или стирания страницы памяти программ бит SPMEN остается равным логической единице до завершения операции.

Запись любой другой комбинации кроме «10001», «01001», «00101», «00011» или «00001» в младшие пять битов не будет иметь никакого эффекта.

3.20.6 Адресация памяти программ во время самопрограммирования

Z-указатель наряду с регистром RAMPZ используется для адресации SPM-команд. Подробности относительно того, как использовать RAMPZ, см. в подпункте «Регистр выбора Z-страницы ОЗУ – RAMPZ» в подразделе 3.1 «Ядро микроконтроллера».

Бит	15	14	13	12	11	10	9	8
ZH (R31)	Z15	Z14	Z13	Z12	Z11	Z10	Z9	Z8
ZL (R30)	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0
	7	6	5	4	3	2	1	0

Поскольку память программ организована постранично (см. таблицу 3.123), программный счетчик можно рассматривать как счетчик, имеющий две различные секции. Одна секция, состоящая из младших значащих битов, адресует слова в пределах страницы, в то время как старшие значащие биты адресуют сами страницы. Это показано на рисунке 3.135. Обратите внимание, что операции стирания и записи в страницу адресуются независимо друг от друга. Поэтому чрезвычайно важным является то, что программа начальной загрузки адресует одну и ту же страницу и во время операции стирания, и во время операции записи. После однократного запуска операции программирования адрес фиксируется, и Z-указатель/RAMPZ может применяться для других операций.

Единственной операцией SPM, которая не использует Z-указатель/RAMPZ, является установка битов защиты начального загрузчика. Содержимое Z-указателя/RAMPZ игнорируется и не оказывает влияния на операцию. Инструкция (E)LPM использует Z-указатель/RAMPZ для хранения адреса. Поскольку данная команда адресует память программ побайтно, используется также и младший значащий бит (Z0) Z-указателя.

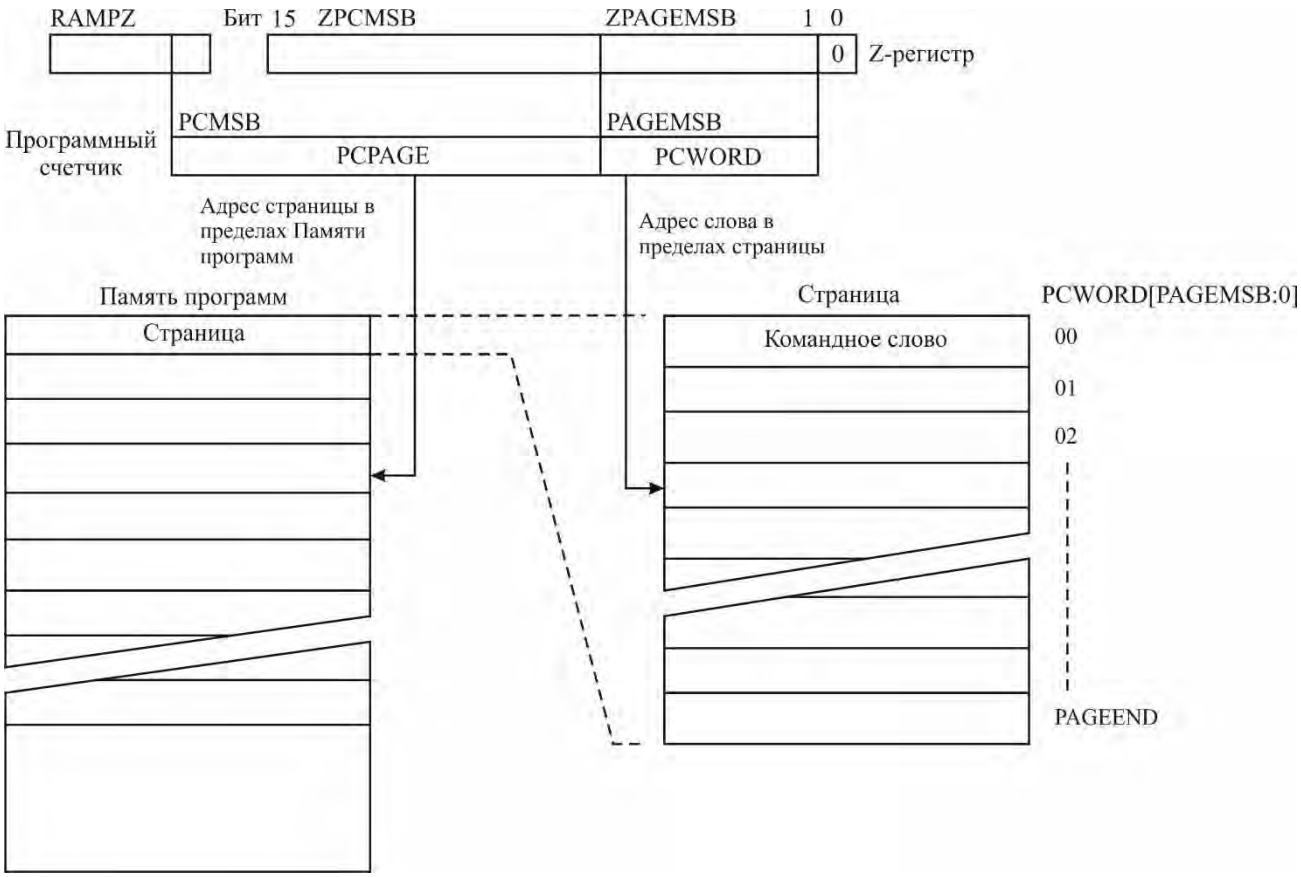


Рисунок 3.135 – Адресация памяти программ при выполнении команды SPM

Примечания

1 Различные параметры, используемые на рисунке 3.135, представлены в таблице 3.114.

2 Параметры PCPAGE и PCWORD представлены в таблице 3.124.

3.20.7 Самопрограммирование памяти программ

Содержимое памяти программ возможно стирать и записывать новыми данными как постранично, так и побайтно. Если необходимо записать новые данные в страницу памяти программ, то процедуру стирания можно пропустить. Временный буфер страницы за один раз заполняется одним словом при помощи инструкции SPM. Буфер необходимо заполнять перед выполнением операции стирания или записи. Буфер страницы можно заполнять как полностью, так и частично, изменяться будут только заполненные во временном буфере адреса слова. Возможно несколько вариантов изменения памяти программ.

Вариант 1. Изменение данных во всей странице:

- Заполните временный буфер страницы.
- Выполните стирание или запись информации в странице.

Вариант 2. Изменение части слова в странице:

- Заполните необходимые адреса слов во временном буфере страницы.
- Выполните запись или стирание данных.

При использовании варианта 1 загрузчик модифицирует всю страницу памяти программ. При использовании варианта 2 загрузчик изменит данные только выбранных слов в странице памяти программ. К временному буферу страницы можно получать доступ в произвольной последовательности. Существенным является то, что адрес страницы, используемый как в операции стирания, так и в операции записи, указывает на одну и ту же страницу (см. ниже подпункт «Пример простого ассемблированного кода для загрузчика»).

Выполнение стирания данных в странице с помощью команды SPM

Для выполнения стирания данных в странице установите адрес в Z-указателе и регистре RAMPZ, заполните буфер, запишите комбинацию «X0000011» в SPMCSR и выполните команду SPM в течение четырех тактов после записи в регистр SPMCSR. Данные в R1 и R0 не учитываются. Адрес страницы должен быть записан в PCPAGE в Z-регистре. Другие биты Z-указателя должны быть равны логическому нулю во время данной операции.

- Стирание данных в странице сегмента RWW: сегмент NRWW доступен для чтения во время операции стирания.

- Стирание данных в странице сегмента NRWW: ЦПУ останавливается на весь период операции стирания.

Заполнение временного буфера (загрузка страницы)

Для записи командного слова установите адрес в Z-указателе и данные в регистрах R1 и R0, запишите комбинацию «00000001» в SPMCSR и выполните команду SPM в течение четырех тактов после записи в регистр SPMCSR. Содержимое PCWORD в Z-регистре используется для адресации данных во временном буфере. Информация во временном буфере автоматически сотрется после выполнения операции записи или при записи бита RWWSRE в регистре SPMCSR. Эта информация также удаляется после системного сброса. Обратите внимание, данные в ячейках памяти можно обновлять без предварительного стирания.

Примечание – Если запись в ЭПД произведена в середине операции загрузки страницы SPM, то все загруженные данные будут потеряны.

Выполнение записи в страницу

Для выполнения записи установите адрес в Z-указателе и регистре RAMPZ, запишите комбинацию «X0000101» в SPMCSR и выполните команду SPM в течение четырех тактов после записи в регистр SPMCSR. Данные в R1 и R0 не учитываются. Адрес страницы должен быть записан в PCPAGE. Другие биты Z-указателя должны быть равны логическому нулю во время данной операции.

- Запись данных в страницу сегмента RWW: сегмент NRWW доступен для чтения во время операции записи.

- Запись данных в страницу сегмента NRWW: ЦПУ останавливается на весь период операции записи.

Использование прерывания по готовности SPM

Если разрешено прерывание по готовности SPM, то будет генерироваться непрерывное прерывание, когда бит SPMEN в регистре SRMCSR сброшен. Это значит, что данное прерывание можно использовать вместо программного опроса регистра SPMCSR. При использовании прерывания по готовности SPM векторы прерываний следует переместить в секцию загрузчика (BLS), чтобы избежать возможности получения доступа прерыванием к области RWW, когда она заблокирована для чтения. Подробности относительно перемещения прерываний см. в подразделе 3.6 «Прерывания».

Учет некоторых особенностей при обновлении секции загрузчика

Необходимо быть особенно осторожным, если пользователь разрешает обновление секции загрузчика, оставив бит BLB11 незапрограммированным. Случайная запись в сам загрузчик может повредить всю программу загрузчика, а дальнейшие обновления программ могут стать невозможными. Если нет необходимости изменять саму программу загрузчика, рекомендуется программировать бит BLB11, чтобы защитить загрузчик от любых внутренних программных изменений.

Предотвращение чтения области RWW во время самопрограммирования

Во время самопрограммирования (либо при стирании, либо при записи данных в страницу) запрещено обращаться к области RWW. Пользовательская программа сама должна предотвратить возможность адресации данной области в течение операции самопрограммирования. Бит RWWSB в регистре SPMCSR будет установлен, пока область RWW занята. Во время самопрограммирования таблицу векторов прерываний следует переместить в сектор загрузчика (BLS), как описано в подразделе 3.6 «Прерывания», либо прерывания должны быть запрещены. Перед адресацией области RWW после завершения программирования пользовательская программа должна сбросить бит RWWSB путем записи бита RWWSRE. См. ниже подпункт «Пример простого ассемблированного кода для загрузчика».

Установка битов защиты загрузчика с помощью команды SPM

Для установки битов защиты загрузчика внесите желаемые данные в регистр R0, запишите комбинацию «X0001001» в SPMCSR и выполните команду SPM в течение четырех тактов после записи в регистр SPMCSR. Единственными доступными битами защиты являются биты защиты загрузки BLB, которые могут предохранить секцию прикладной программы и секцию загрузчика от любого обновления программ микроконтроллером.

Бит	7	6	5	4	3	2	1	0
R0	1	1	BLB12	BLB11	BLB02	BLB01	1	1

См. таблицы 3.108 и 3.109, чтобы изучить влияние различных установок битов защиты на доступ к памяти программ.

Если какие-либо биты 5–2 в регистре R0 сброшены, то соответствующие биты защиты будут запрограммированы запуском команды SPM в течение четырех тактов после установки битов BLBSET и SPMEN в SPMCSR. Значение Z-указателя не столь важно во время данной операции, но все же рекомендуется загружать в него адрес $0001 (адрес, используемый для чтения битов защиты). Также рекомендуется устанавливать логическую единицу в битах 7, 6, 1 и 0 регистра R0 при записи битов защиты. При программировании битов защиты все пространство памяти программ доступно для чтения во время данной операции.

Предотвращение записи в регистр SPMCSR путем записи в ЭПД

Обратите внимание, что операция записи в ЭПД блокирует программирование памяти программ. Чтение конфигурационных битов и битов защиты из программы также не будет разрешено во время операции записи в ЭПД. Пользователю рекомендуется проверять бит статуса (EEWE) в регистре EECR и контролировать, что этот бит сбрасывается перед выполнением операции записи в регистр SPMCSR.

Чтение конфигурационных битов и битов защиты из программы

Имеется возможность читать как конфигурационные биты, так и биты защиты из программы. Для чтения битов защиты загрузите адрес $0001 в Z-указатель и установите биты BLBSET и SPMEN в регистре SPMCSR. Когда выполнится команда LPM в течение трех тактов после установки BLBSET и SPMEN, значение битов защиты будет загружено в заданный регистр. Биты BLBSET и SPMEN будут сброшены автоматически по окончании считывания битов защиты, или если ни одна инструкция LPM не выполнится в течение трех тактов, или ни одна команда SPM не выполнится в течение четырех тактов. Когда BLBSET и SPMEN сбросятся, LPM будет работать, как описано в приложении A.

Бит	7	6	5	4	3	2	1	0
Rd	-	-	BLB12	BLB11	BLB02	BLB01	LB2	LB1

Алгоритм для чтения младших конфигурационных битов подобен алгоритму для чтения битов защиты, описанному выше. Чтобы считать младшие конфигурационные биты, загрузите в Z-указатель адрес $0000 и установите биты BLBSET и SPMEN в SPMCSR. Когда выполнится команда LPM в течение трех тактов после установки BLBSET и SPMEN, значение младших конфигурационных битов (FLB) будет загружено в заданный регистр, как показано ниже. См. таблицу 3.119 для получения подробного описания и отображения младших конфигурационных битов.

Бит	7	6	5	4	3	2	1	0
Rd	FLB7	FLB6	FLB5	FLB4	FLB3	FLB2	FLB1	FLB0

Подобным образом, при считывании старших конфигурационных битов, загрузите адрес $0003 в Z-указатель. Когда выполнится команда LPM в течение трех тактов после установки BLBSET и SPMEN в регистре SPMCSR, значение старших конфигурационных битов (FHB) будет загружено в заданный регистр, как показано ниже. См. таблицу 3.118 для получения подробного описания и отображения старших конфигурационных битов.

Бит	7	6	5	4	3	2	1	0
Rd	FHB7	FHB6	FHB5	FHB4	FHB3	FHB2	FHB1	FHB0

При считывании расширенных конфигурационных битов загрузите в Z-указатель адрес $0002. Когда выполнится команда LPM в течение трех тактов после установки BLBSET и SPMEN в регистре SPMCSR, значение расширенных конфигурационных битов (EFB) будет загружено в заданный регистр, как показано ниже. См. таблицу 3.117 для получения подробного описания и отображения расширенных конфигурационных битов.

Бит	7	6	5	4	3	2	1	0
Rd	-	-	-	-	-	-	EFB1	EFB0

Запрограммированные конфигурационные биты и биты защиты будут считываться как ноль. Незапрограммированные конфигурационные биты и биты защиты будут считываться как единица.

Предотвращение повреждения памяти программ

В периоды низкого напряжения #VCC данные в памяти программ могут быть повреждены, так как напряжение питания слишком низкое для того, чтобы ЦПУ и память программ могли работать должным образом. В данном случае возникают точно такие же проблемы, что и в системах на уровне плат, которые используют память программ, и поэтому следует применять те же самые проектные решения.

Повреждение программы в памяти программ может быть вызвано двумя ситуациями, когда напряжение слишком низкое. Во-первых, правильная последовательность записи в память программ требует наличия определенного минимально необходимого напряжения для корректной работы. Во-вторых, само ЦПУ может выполнять команды неправильно, если напряжения питания слишком низкое для выполнения инструкций.

Повреждения памяти программ можно легко избежать, если придерживаться следующих проектных рекомендаций (одной будет достаточно):

1 Если нет необходимости в системном обновлении начального загрузчика, программируйте биты защиты для предотвращения любых обновлений загрузчика.

2 Удерживайте сигнал сброса микроконтроллера активным (в низком уровне) в периоды недостаточного напряжения питания. Это можно сделать путем включения внутреннего детектора питания BOD, или же использовать внешний монитор питания с требуемым уровнем детектирования. Если сброс происходит во время выполнения операции записи, то

данная операция будет завершена, при условии, что напряжение питания будет достаточным.

3 Переводите микроконтроллер в режим микропотребления в периоды низкого напряжения #VCC. Это предотвратит ЦПУ от попытки декодирования и выполнения инструкций, эффективно защищая регистр SPMCSR и, соответственно, память программ от непреднамеренной записи.

Время программирования памяти программ при использовании команды SPM

Калиброванный RC-генератор применяется для измерения времени доступа к памяти программ. В таблице 3.111 показывается типичное время программирования для доступа к памяти программ из ЦПУ.

Таблица 3.111 – Время программирования при использовании команды SPM

Обозначение	Минимальное время программирования, мс	Максимальное время программирования, мс
Запись памяти программ (стирание данных в странице, запись в страницу, запись битов защиты с помощью команды SPM)	3,7	4,5

Пример простого ассемблированного кода для загрузчика

```
; Программа записывает одну страницу данных из ОЗУ в память программ.
; Первая ячейка данных в ОЗУ адресуется Y-указателем.
; Первая ячейка данных в памяти программ адресуется Z-указателем.
; Обработка ошибок не включена в программу.
; Программа должна размещаться внутри пространства загрузки
; (по крайней мере, подпрограмма Do_spm). Только код внутри сегмента NRWW
; доступен для чтения во время самопрограммирования
; (при стирании и записи данных в страницу).
; Используемые регистры: r0, r1, temp1 (r16), temp2 (r17), looplo (r24),
; loophi (r25), spmcsrval (r20).
; Хранение и восстановление регистров не включено в программу.
; Использование регистров можно оптимизировать за счет размера кода.
; Предполагается, либо что таблица прерываний перемещена в сегмент начального
; загрузчика BLS, либо что прерывания запрещены.

.equ PAGESIZEB = PAGESIZE*2
; PAGESIZEB определяет размер страницы в байтах, а не в словах
.org SMALLBOOTSTART
Write_page:
; Стирание информации в странице
ldi spmcsrval, (1<<PGERS) | (1<<SPMEN)
call Do_spm

; Отмена запрета на доступ к сегменту RWW
ldi spmcsrval, (1<<RWWSRE) | (1<<SPMEN)
call Do_spm

; Перемещение данных из ОЗУ в буфер страницы памяти программ
ldi looplo, low(PAGESIZEB)  ;Инициализация цикличного параметра
ldi loophi, high(PAGESIZEB) ;Не требуется для PAGESIZEB<=256
Wrloop:
ld r0, Y+
ld r1, Y+
ldi spmcsrval, (1<<SPMEN)
call Do_spm
adiw ZH:ZL, 2
sbiw loophi:looplo, 2 ;Используйте команду subi для PAGESIZEB<=256
brne Wrloop
```

```
; Выполнение записи в страницу
subi ZL, low(PAGESIZEB)  ;Восстановление указателя
sbci ZH, high(PAGESIZEB) ;Не требуется для PAGESIZEB<=256
ldi spmcsrval, (1<<PGWRT) | (1<<SPMEN)
call Do_spm

; Отмена запрета на доступ к сегменту RWW
ldi spmcsrval, (1<<RWWSRE) | (1<<SPMEN)
call Do_spm

; Считывание в обратном направлении и проверка (дополнительно)
ldi looplo, low(PAGESIZEB)  ;Инициализация цикличного параметра
ldi loophi, high(PAGESIZEB) ;Не требуется для PAGESIZEB<=256
subi YL, low(PAGESIZEB)     ;Восстановление указателя
sbci YH, high(PAGESIZEB)
Rdloop:
lpm r0, Z+
ld r1, Y+
cpse r0, r1
jmp Error
sbiw loophi:looplo, 1 ;Используйте команду subi для PAGESIZEB<=256
brne Rdloop

; Возврат в сегмент RWW
; Проверка, что сегмент RWW можно безопасно считывать
Return:
lds temp1, SPMCSR
sbrs temp1, RWWSB ;Если бит RWWSB установлен, сегмент RWW еще занят
ret
; Отмена запрета на доступ к сегменту RWW
ldi spmcsrval, (1<<RWWSRE) | (1<<SPMEN)
call Do_spm
rjmp Return


Do_spm:
; Проверка предыдущего выполнения команды SPM
Wait_spm:
lds temp1, SPMCSR
sbrc temp1, SPMEN
rjmp Wait_spm
; spmcsrval определяет действие команды SPM
; Запрет прерываний, если они разрешены, запоминание состояния
in temp2, SREG
cli
; Проверка, что нет доступа к записи ЭПД
Wait_ee:
sbic EECR, EEWE
rjmp Wait_ee
; Временная последовательность SPM
sts SPMCSR, spmcsrval
spm
; Восстановление регистра SREG (для разрешения прерываний, если изначально
; они были разрешены)
out SREG, temp2
ret
```

Параметры загрузчика в ИС 1887ВЕ7Т

Размер страницы секции загрузчика составляет 32 шестнадцатиразрядных слова. Размер страницы секции прикладной программы составляет 128 шестнадцатиразрядных слов. В таблицах 3.112 – 3.114 представлены параметры, используемые в описании самопрограммирования.

Таблица 3.112 – Конфигурация размера секций памяти программ

BOOTSZ1	BOOTSZ0	Размер секции, слов	Число страниц	Секция прикладной программы	Секция загрузчика	Конец секции прикладной программы	Вектор сброса в секции загрузчика (начало секции загрузчика)
1	1	512	16	$0000 - $FDFF	$FE00 - $FFFF	$FDFF	$FE00
1	0	1024	32	$0000 - $FBFF	$FC00 - $FFFF	$FBFF	$FC00
0	1	2048	64	$0000 - $F7FF	$F800 - $FFFF	$F7FF	$F800
0	0	4096	128	$0000 - $EFFF	$F000 - $FFFF	$EFFF	$F000

Примечание – Различные настройки конфигурационных битов BOOTSZ показаны на рисунке 3.134.

Таблица 3.113 – Максимальный размер областей RWW и NRWW

Область	Число страниц	Адрес
RWW	480	$0000 - $EFFF
NRWW	128	$F000 - $FFFF

Примечание – Подробности, касающиеся сегментов RWW и NRWW, см. выше в подпунктах «Сегмент с поддержкой чтения во время записи – RWW» и «Сегмент без поддержки функции чтения во время записи – NRWW». Обратите внимание, что размер страницы области RWW 128 слов, а размер области NRWW 32 слова.

Таблица 3.114 – Пояснение различных параметров, используемых на рисунке 3.135, и отображение Z-указателя*

Параметр	Номер бита	Соответствующее Z-значение	Описание**
PCMSB	15		Старший значащий бит в программном счетчике. (Программный счетчик состоит из 16 битов PC[15–0])
PAGEMSB	6		Старший значащий бит, используемый для адресации слов в пределах одной страницы (для 128 слов в странице необходимы 7 битов PC[6–0])
ZPCMSB	-	Z16***	Бит в Z-регистре, который отображает PCMSB. Поскольку Z0 не используется, ZPCMSB равняется PCMSB + 1
ZPAGEMSB	-	Z7	Бит в Z-регистре, который отображает PAGEMSB. Поскольку Z0 не используется, ZPAGEMSB равняется PAGEMSB + 1
PCPAGE	PC[15–7]	Z16***–Z8	Адрес страницы программного счетчика: выбор страницы для стирания данных и записи
PCWORD	PC[6–0]	Z7–Z1	Адрес слова программного счетчика: выбор слова для заполнения временного буфера (должен быть равен логическому нулю во время операции записи в страницу)

* См. выше пункт 3.20.6 «Адресация памяти программ во время самопрограммирования» для получения подробностей об использовании Z-указателя во время самопрограммирования.

** Z0 должен равняться нулю для всех команд SPM, а для инструкций (E)LPM он выбирает байт.

*** Z-регистр состоит только из 16 битов Z[15–0]. Бит Z16 расположен в регистре RAMPZ в памяти ввода-вывода.

3.21 Программирование памяти

3.21.1 Биты защиты памяти программ и данных

Микроконтроллер 1887ВЕ7Т имеет 6 битов защиты, которые можно оставить в незапрограммированном состоянии («1») или же запрограммировать («0») для активации дополнительных функций, представленных в таблице 3.116. Стирание битов защиты (установка в «1», см. таблицу 3.115) может быть выполнено только при помощи команды стирания (Chip Erase).

Таблица 3.115 – Байт с битами защиты

Байт битов защиты	Номер бита	Описание	Значение по умолчанию
-	7	-	1 (незапрограммировано)
-	6	-	1 (незапрограммировано)
BLB12	5	Бит защиты загрузочного сектора	1 (незапрограммировано)
BLB11	4	Бит защиты загрузочного сектора	1 (незапрограммировано)
BLB02	3	Бит защиты загрузочного сектора	1 (незапрограммировано)
BLB01	2	Бит защиты загрузочного сектора	1 (незапрограммировано)
LB2	1	Бит защиты	1 (незапрограммировано)
LB1	0	Бит защиты	1 (незапрограммировано)
Примечание – «1» означает незапрограммированное состояние, «0» означает запрограммированное состояние.			

Таблица 3.116 – Режимы защиты

Биты защиты памяти			Тип защиты
Режим LB	LB2	LB1	
1	1	1	Защита памяти не активирована
2	1	0	Дальнейшее программирование памяти программ и ЭПД в параллельном режиме и режиме последовательного программирования через интерфейсы SPI/JTAG запрещено. Конфигурационные биты блокированы как при последовательном, так и при параллельном программировании
3	0	0	Дальнейшее программирование и верификация памяти программ и ЭПД в параллельном режиме и режиме последовательного программирования через интерфейсы SPI/JTAG запрещено. Конфигурационные биты блокированы как при последовательном, так и при параллельном программировании
Режим BLB0	BLB02	BLB01	
1	1	1	Нет никаких ограничений для доступа к коду, расположенному в секции прикладной программы, с помощью инструкций SPM или (E)LPM
2	1	0	Запись в секцию прикладной программы с помощью команды SPM запрещена
3	0	0	Запись в секцию прикладной программы командой SPM запрещена. Команда (E)LPM, вызываемая из секции загрузчика, не может выполнять чтение из секции прикладной программы. Если таблица векторов прерываний расположена в секции загрузчика, то, при выполнении кода из секции прикладной программы, прерывания запрещены.
4	0	1	Чтение из секции прикладной программы командой (E)LPM, вызываемой из секции загрузчика, запрещено. Если таблица векторов прерываний расположена в секции загрузчика, то, при выполнении кода из секции прикладной программы, прерывания запрещены.

Продолжение таблицы 3.116

Биты защиты памяти			Тип защиты
Режим BLB1	BLB12	BLB11	
1	1	1	Нет никаких ограничений для доступа к коду, расположенному в секции загрузчика, с помощью инструкций SPM или (E)LPM
2	1	0	Запись в секцию загрузчика с помощью команды SPM запрещена
3	0	0	Запись в секцию загрузчика командой SPM запрещена. Команда (E)LPM, вызываемая из секции прикладной программы, не может выполнять чтение из секции загрузчика. Если таблица векторов прерываний расположена в секции пркладной программы, то, при выполнении кода из секции загрузчика, прерывания запрещены
4	0	1	Чтение из секции загрузчика командой (E)LPM, вызываемой из секции прикладной программы, запрещено. Если таблица векторов прерываний расположена в секции прикладной программы, то, при выполнении кода из секции загрузчика, прерывания запрещены

Примечания

1 Конфигурационные биты необходимо запрограммировать перед программированием битов защиты.

2 «1» – означает незапрограммированное состояние, «0» – означает запрограммированное состояние.

3.21.2 Конфигурационные биты

Микроконтроллеры 1887ВЕ7Т имеют три конфигурационных байта. В таблицах 3.117 – 3.119 кратко описываются функционирование и расположение всех конфигурационных битов. Обратите внимание, что конфигурационные биты читаются как логический ноль, когда они запрограммированы.

На состояние конфигурационных битов не оказывает влияния команда стирания (Chip Erase). Обратите внимание, что доступ к конфигурационным битам блокирован, если запрограммирован бит защиты LB1. Поэтому конфигурационные биты необходимо программировать перед программированием битов защиты.

Таблица 3.117 – Расширенный конфигурационный байт

Расширенный конфигурационный байт	Номер бита	Описание	Значение по умолчанию
-	7	-	1 (незапрограммировано)
-	6	-	1 (незапрограммировано)
-	5	-	1 (незапрограммировано)
-	4	-	1 (незапрограммировано)
-	3	-	1 (незапрограммировано)
-	2	-	1 (незапрограммировано)
M103C*	1	Режим совместимости	0 (запрограммировано)
WDTON**	0	Сторожевой таймер всегда активен	1 (незапрограммировано)

* См. Приложение Б для изучения деталей.

** См. 3.5.9 «Сторожевой таймер» для изучения деталей.

Таблица 3.118 – Старший конфигурационный байт

Старший конфигурационный байт	Номер бита	Описание	Значение по умолчанию
OCDEN [4]	7	Разрешение работы средств отладки	1 (незапрограммировано, встроенная система отладки отключена)
JTAGEN [5]	6	Включение JTAG	0 (запрограммировано)
SPIEN [1]	5	Разрешение последовательного программирования	0 (запрограммировано)
CKOPT [2]	4	Опции генератора	1 (незапрограммировано)
EESAVE	3	ЭПД защищено от действия команды стирания (Chip Erase)	1 (незапрограммировано)
BOOTSZ1	2	Выбор размера секции памяти программ (см. таблицу 3.112)	0 (запрограммировано) [3]
BOOTSZ0	1	Выбор размера секции памяти программ (см. таблицу 3.112)	0 (запрограммировано) [3]
BOOTRST	0	Выбор вектора сброса	1 (незапрограммировано)

[1] **Внимание!** Конфигурационный бит SPIEN доступен для записи в режиме последовательного программирования через SPI. Необходимо быть внимательным при программировании данного бита, так как при неправильной установке данного бита доступ к микроконтроллеру в режиме внутрисистемного программирования будет заблокирован.

[2] Функционирование конфигурационного бита CKOPT зависит от установок битов CKSEL. См. 3.3.2 «Источники синхронизации» для изучения деталей.

[3] Исходное значение битов BOOTSZ1–0 соответствует выбору максимального размера загрузочного сектора. См. таблицу 3.112.

[4] Не забудьте деактивировать бит OCDEN перед поставкой готового изделия заказчику, независимо от того какие установки имеют биты защиты и конфигурационный бит JTAGEN. Если бит OCDEN будет запрограммирован, то некоторые части системы синхронизации останутся в работе при переводе микроконтроллера в любой из спящих режимов. Это может привести к повышенному энергопотреблению.

[5] Если интерфейс JTAG оставлен неподключенным, то конфигурационный бит JTAGEN должен быть по возможности отключен. Это позволит избежать возникновения статического тока на выводе TDO интерфейса JTAG.

Таблица 3.119 – Младший конфигурационный байт

Младший конфигурационный байт	Номер бита	Описание	Значение по умолчанию
BODLEVEL	7	Уровень срабатывания детектора питания	1 (незапрограммировано)
BODEN	6	Включение детектора питания	1 (незапрограммировано, BOD отключен)
SUT1	5	Выбор времени запуска	1 (незапрограммировано) [1]
SUT0	4	Выбор времени запуска	0 (запрограммировано) [1]
CKSEL3	3	Выбор источника синхронизации	0 (запрограммировано) [2]
CKSEL2	2	Выбор источника синхронизации	0 (запрограммировано) [2]
CKSEL1	1	Выбор источника синхронизации	0 (запрограммировано) [2]
CKSEL0	0	Выбор источника синхронизации	1 (незапрограммировано) [2]

[1] Исходные установки SUT1–0 соответствуют выбору максимального времени запуска. Подробности представлены в таблице 3.14.

[2] Исходные установки CKSEL3–0 соответствуют выбору внутреннего RC-генератора частотой 1 МГц. Подробности представлены в таблице 3.6.

Защита конфигурационных битов

Значение конфигурационных битов запоминается, когда микроконтроллер переходит в режим программирования. Изменение значений конфигурационных битов не будет иметь никакого эффекта до тех пор, пока устройство не выйдет из режима программирования. Это не распространяется на бит EESAVE, и он может быть запрограммирован в любой момент. Доступ к конфигурационным битам блокируется также при подаче питания в нормальном режиме работы.

3.21.3 Сигнатурные байты

Микроконтроллер 1887ВЕ7Т имеет трехбайтный сигнатурный код, который позволяет идентифицировать устройство. Этот код можно считать как в параллельном, так и в последовательном режиме программирования, в том числе, когда устройство защищено. Для ИС 1887ВЕ7Т сигнатурными байтами являются:

1. $000: $1E
2. $001: $97
3. $002: $02

3.21.4 Калибровочный байт

Микроконтроллер 1887ВЕ7Т хранит четыре различных калибровочных значения для внутреннего RC-генератора. Они находятся в адресных ячейках 0x0000, 0x0001, 0x0002 и 0x0003 и соответствуют частотам генератора 1, 2, 4 и 8 МГц. В процессе сброса в регистр OSCCAL автоматически загружается значение калибровочного байта для выбранных частот.

3.21.5 Параметры параллельного программирования, расположение выводов и команды

В данном пункте описывается, как выполнять параллельное программирование и верификацию памяти программ, ЭПД памяти данных, битов защиты памяти и конфигурационных битов в ИС 1887ВЕ7Т. Предполагается, что длительность импульсов – не менее 250 нс, если не указано иное.

Названия сигналов

В данном подпункте отдельные выводы микроконтроллера обозначаются названиями сигналов, которые описывают их функционирование в процессе параллельного программирования (см. рисунок 3.136 и таблицу 3.120). Выводы, не представленные в таблице 3.121, обозначаются нормальными названиями выводов.

Выводы XA1/XA0 определяют действие, выполняемое при положительном фронте на выводе BQ1 (см. таблицу 3.122).

При подаче импульсов WR# или OE# загруженная команда определяет выполняемое действие (см. таблицу 3.123).

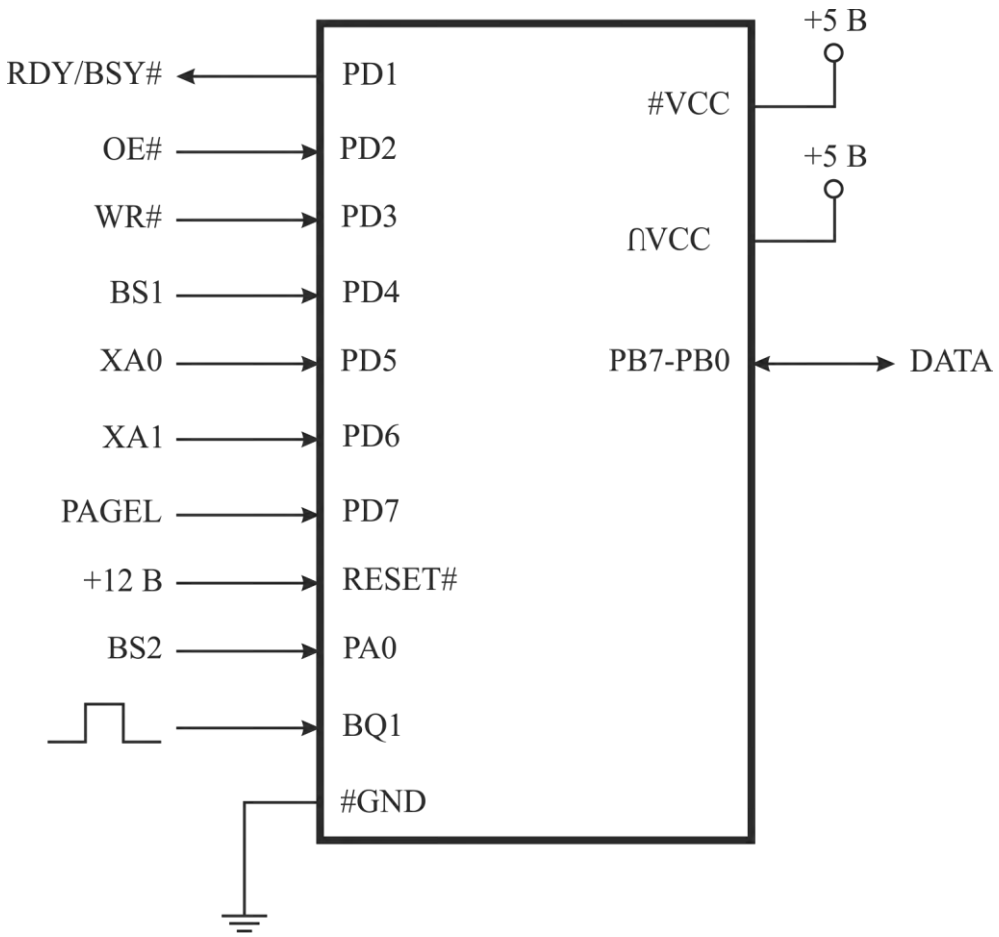


Рисунок 3.136 – Сигналы параллельного программирования

Таблица 3.120 – Расположение и назначение выводов программирования

Название сигнала в режиме программирования	Название вывода	Вход/выход (I/O, соответственно)	Функция
RDY/BSY#	PD1	O	0: Устройство занято программированием 1: Устройство готово к новой команде
OE#	PD2	I	Разрешение вывода (активный низкий уровень)
WR#	PD3	I	Импульс записи (активный низкий уровень)
BS1	PD4	I	Выбор байта 1 («0» выбирает младший байт, «1» выбирает старший байт)
XA0	PD5	I	Бит 0 выбора действия при активности на выводе BQ1
XA1	PD6	I	Бит 1 выбора действия при активности на выводе BQ1
PAGEL	PD7	I	Загрузка страницы данных в память программ и ЭПД
BS2	PA0	I	Выбор байта 2 («0» выбирает младший байт, «1» выбирает второй старший байт)
DATA	PB7-0	I/O	Двунаправленная шина данных (вывод данных, когда OE# имеет низкий уровень)

Таблица 3.121 – Состояния выводов, которые используются для входа в режим программирования

Вывод	Обозначение	Значение
PAGEL	Prog_enable[3]	0
XA1	Prog_enable[2]	0
XA0	Prog_enable[1]	0
BS1	Prog_enable[0]	0

Таблица 3.122 – Назначение кодов XA1 и XA0

XA1	XA0	Действие, выполняемое во время импульса BQ1
0	0	Загрузка адреса памяти программ или ЭПД (старший или младший байт адреса, определяемый BS1)
0	1	Загрузка данных (старший или младший байт данных, определяемый BS1)
1	0	Загрузка команды
1	1	Нет никаких действий (холостой ход)

Таблица 3.123 – Коды команд

Командный байт	Выполняемая команда
1000 0000	Стирание (Chip Erase)
0100 0000	Запись конфигурационных битов
0010 0000	Запись битов защиты
0001 0000	Запись в память программ
0001 0001	Запись в ЭПД
0000 1000	Чтение сигнатурных байтов и калибровочного байта
0000 0100	Чтение конфигурационных битов и битов защиты
0000 0010	Чтение памяти программ
0000 0011	Чтение ЭПД

Таблица 3.124 – Количество слов в странице и количество страниц в памяти программ

Область памяти программ	Размер страницы	PCWORD	Число страниц	PCPAGE	PCMSB
RWW (60К слов)	128 слов	PC[6:0]	480	PC[15:7]	15
NRWW (4К слов)	32 слов	PC[4:0]	128	PC[11:5]	11

Таблица 3.125 – Количество слов в странице и количество страниц в ЭПД

Размер ЭПД	Размер страницы	PCWORD	Число страниц	PCPAGE	EEAMSB
4 Кбайт	8 байт	EEA[2:0]	512	EEA[11:3]	8

3.21.6 Параллельное программирование

Параллельный режим программирования можно использовать только при напряжении питания микроконтроллера 5,0 В ± 10 %.

Вход в режим программирования

Выполнение следующего алгоритма переводит микроконтроллер в режим параллельного программирования:

1. Подайте напряжение 4,5–5,5 В между выводами #VCC и #OV (#GND) и подождите минимум 100 мкс.

2. Установите логический ноль на выводе RESET# и переключайте BQ1 по крайней мере шесть раз.

3. Установите комбинацию «0000» для выводов Prog_enable (см. таблицу 3.121) и подождите минимум 100 нс.

4. Подайте напряжение 11,5–12,5 В на вывод RESET#. Любые действия на выводах Prog_enable в течение 100 нс после подачи +12 В на вывод RESET# приведут к сбою при входе в режим программирования.

Обратите внимание, что если используется внешний кварцевый резонатор или внешняя RC-цепь, то нет возможности приложить импульсы к BQ1. В данном случае необходимо придерживаться следующего алгоритма:

1. Установите комбинацию «0000» для выводов Prog_enable (см. таблицу 3.121).

2. Подайте напряжение 4,5–5,5 В между выводами #VCC и #OV одновременно с подачей 11,5–12,5 В на вывод RESET#.

3. Подождите 100 мкс.

4. Перепрограммируйте конфигурационные биты, чтобы убедиться, что внешний тактовый генератор выбран в качестве источника синхронизации (CKSEL3–0 = 0b0000). Если биты защиты запрограммированы, то перед изменением значений конфигурационных битов необходимо выполнить команду стирания (Chip Erase).

5. Осуществите выход из режима программирования путем отключения питания или путем установки логического нуля на выводе RESET#.

6. Выполните вход в режим программирования, используя первоначальный алгоритм, описанный выше.

Рекомендации по повышению эффективности программирования

Загружаемые команды и адрес запоминаются в микроконтроллере в процессе программирования. Для повышения эффективности программирования необходимо учитывать следующее:

- Если выполняется чтение или запись многочисленных ячеек памяти, то команду нужно загружать только один раз.

- Старший байт адреса необходимо загружать лишь перед началом программирования или чтения новых 256 слов памяти программ или 256 байтов ЭПД. Это распространяется также на чтение сигнатурных байтов.

Стирание (Chip Erase)

Выполнение команды стирания (Chip Erase) приводит к очистке содержимого памяти программ и ЭПД, а также битов защиты. Очистка битов защиты происходит только после полного стирания памяти программ. Конфигурационные биты при этом не изменяются. Стирание кристалла необходимо выполнять перед перепрограммированием памяти программ и/или ЭПД.

Примечание – Память ЭПД защищена от действия команды стирания кристалла, если запрограммирован конфигурационный бит EESAVE.

Загрузка команды «Стирание кристалла»:

1. Установите кодовую комбинацию «10» для XA1 и XA0. Этим разрешается загрузка команды.

2. Установите BS1 = 0.

3. Установите DATA = «1000 0000». Это код команды стирания.

4. Подайте положительный импульс на BQ1. Это инициирует загрузку команды.

5. Подайте отрицательный импульс на WR#. Это запускает механизм стирания. RDY/BSY# переходит в низкое состояние.

6. Дождитесь, когда RDY/BSY# перейдет в высокое состояние, прежде чем загружать новую команду.

Программирование памяти программ

Память программ имеет постраничную организацию (см. таблицу 3.123). Во время программирования памяти программ данные помещаются в страничный буфер. Это позволяет за один подход запрограммировать целую страницу. Ниже описана процедура программирования всей памяти программ:

А. Загрузка команды «Запись памяти программ»

1. Установите кодовую комбинацию «10» для XA1 и XA0. Этим разрешается загрузка команды.

2. Установите BS1 = 0.

3. Установите DATA = «0001 0000». Это команда записи памяти программ.

4. Подайте положительный импульс на BQ1. Это инициирует загрузку команды.

B. Загрузка младшего адресного байта

1. Установите кодовую комбинацию «00» для XA1 и XA0. Этим разрешается загрузка адреса.

2. Установите BS1 = 0. Это позволяет выбрать младший адрес.

3. Установите DATA = младший адресный байт ($00 – $FF).

4. Подайте положительный импульс на BQ1. Это загружает младший байт адреса.

C. Загрузка младшего байта данных

1. Установите кодовую комбинацию «01» для XA1 и XA0. Этим разрешается загрузка данных.

2. Установите DATA = младший байт данных ($00 – $FF).

3. Подайте положительный импульс на BQ1. Это загружает байт данных.

D. Загрузка старшего байта данных

1. Установите BS1 = 1. Это позволяет выбрать старший байт данных.

2. Установите кодовую комбинацию «01» для XA1 и XA0. Этим разрешается загрузка данных.

3. Установите DATA = старший байт данных ($00 – $FF).

4. Подайте положительный импульс на BQ1. Это загружает байт данных.

E. Защелкивание данных

1. Установите BS1 = 1. Это позволяет выбрать старший байт данных.

2. Подайте положительный импульс на PAGEL. Это приводит к защелкиванию байтов данных (см. форму сигналов на рисунке 3.138).

F. Повторяйте этапы B – E до тех пор, пока не заполнится весь буфер или пока не загрузятся все данные в пределах страницы.

В то время как младшие адресные разряды указывают на слова в пределах страницы, старшие разряды адресуют страницы в пределах памяти программ. Это иллюстрируется на рисунке 3.137. Обратите внимание, что если требуется менее восьми разрядов для адресации слов в странице (размер страницы менее 256), то старшие разряды в младшем адресном байте используются для указания на страницу при выполнении записи страницы.

G. Загрузка старшего адресного байта

1. Установите кодовую комбинацию «00» для XA1 и XA0. Этим разрешается загрузка адреса.

2. Установите BS1 = 1. Это позволяет выбрать старший адрес.

3. Установите DATA = старший адресный байт ($00 – $FF).

4. Подайте положительный импульс на BQ1. Это загружает старший байт адреса.

H. Программирование страницы

1. Установите BS1 = 0.

2. Подайте отрицательный импульс на WR#. Это запускает операцию программирования целой страницы данных. RDY/BSY# переходит в низкое состояние.

3. Дождитесь, когда RDY/BSY# перейдет в высокое состояние (см. форму сигналов на рисунке 3.138).

I. Повторяйте этапы B – H до тех пор, пока не будет запрограммирована вся память программ или пока не будут запрограммированы все данные.

J. Завершение программирования страницы

1. Установите кодовую комбинацию «10» для XA1 и XA0. Этим разрешается загрузка команды.

2. Установите DATA = «0000 0000». Это команда «нет операции».

3. Подайте положительный импульс на BQ1. Это инициирует загрузку команды, и внутренние сигналы записи сбрасываются.

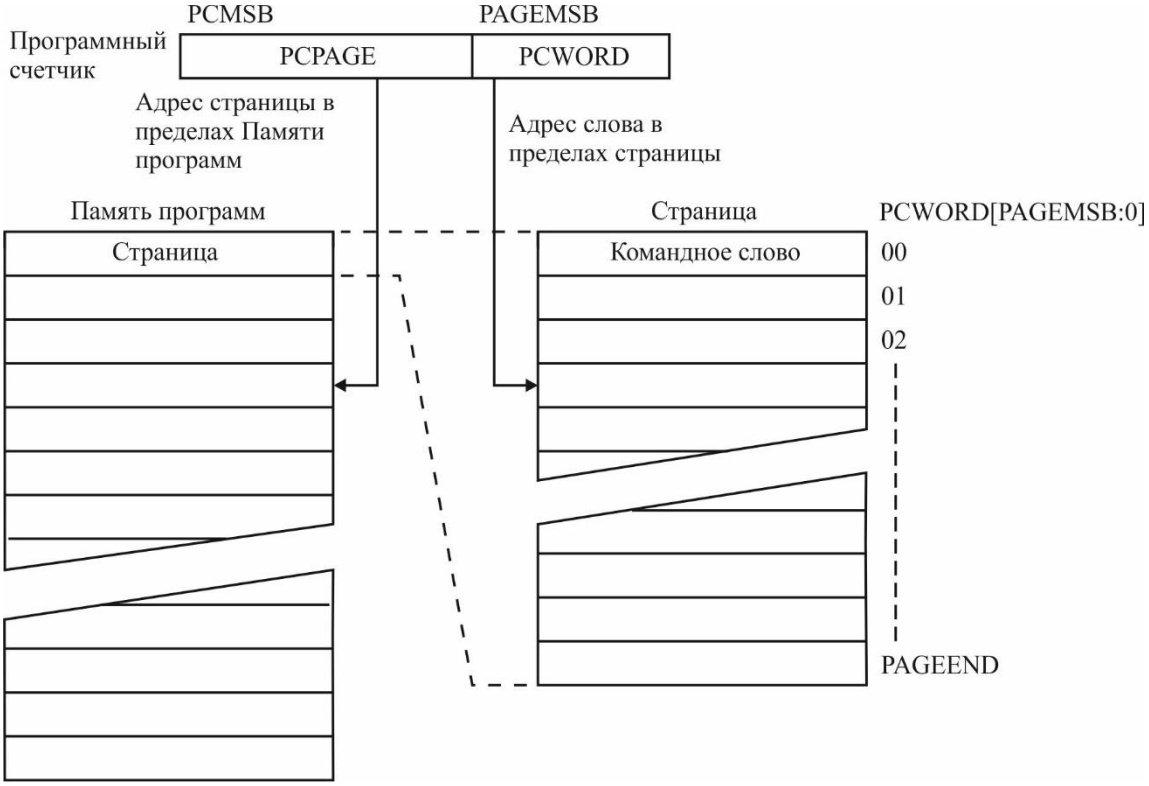


Рисунок 3.137 – Адресация памяти программ, имеющей постраничную организацию

Примечание – PCPAGE и PCWORD приведены в таблице 3.124.

Программирование энергонезависимой памяти данных (ЭПД)

ЭПД имеет постраничную организацию (см. таблицу 3.125). Во время программирования ЭПД данные помещаются в страничный буфер. Это позволяет за один подход запрограммировать целую страницу. Ниже описана процедура программирования ЭПД (см. предыдущий подпункт «Программирование памяти программ» для получения подробностей относительно загрузки команды, адреса и данных):

1. A: Загрузка команды «0001 0001».
2. G: Загрузка старшего адресного байта ($00 – $FF).
3. B: Загрузка младшего адресного байта ($00 – $FF).
4. C: Загрузка данных ($00-$FF).
5. E: Защелкивание данных (подача положительного импульса на PAGEL).

K: Повторяйте этапы 3 – 5 до тех пор, пока не будет заполнен весь буфер.

L: Программирование страницы ЭПД:

1. Установите BS1 = 0.
2. Подайте отрицательный импульс на WR#. Это запускает операцию программирования страницы ЭПД. RDY/BSY# переходит в низкое состояние.
3. Дождитесь, когда RDY/BSY# перейдет в высокое состояние, прежде чем программировать следующую страницу (см. временную диаграмму на рисунке 3.139).

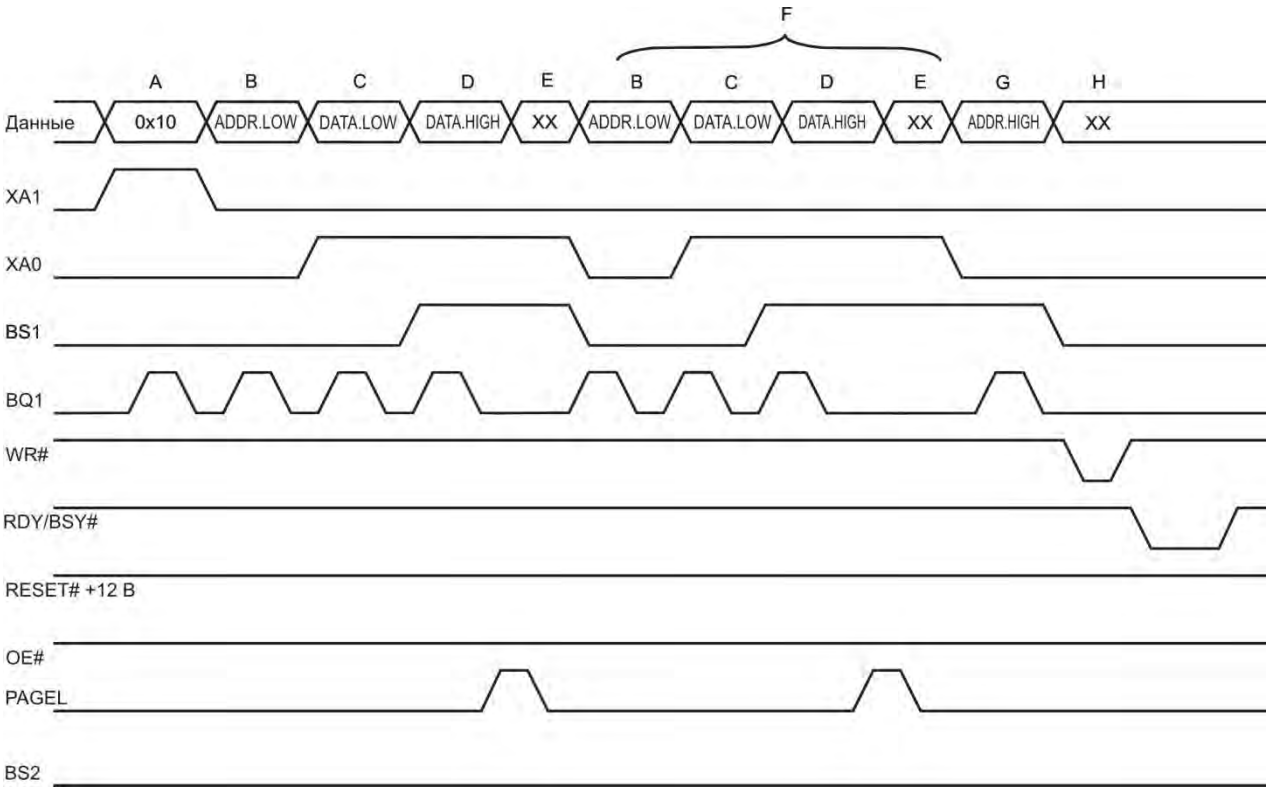


Рисунок 3.138 – Временная диаграмма программирования памяти программ

Примечание – «XX» символизирует, что не имеет значения, какие данные будут присутствовать. Буквы на рисунке соответствуют этапам программирования, описанным выше.

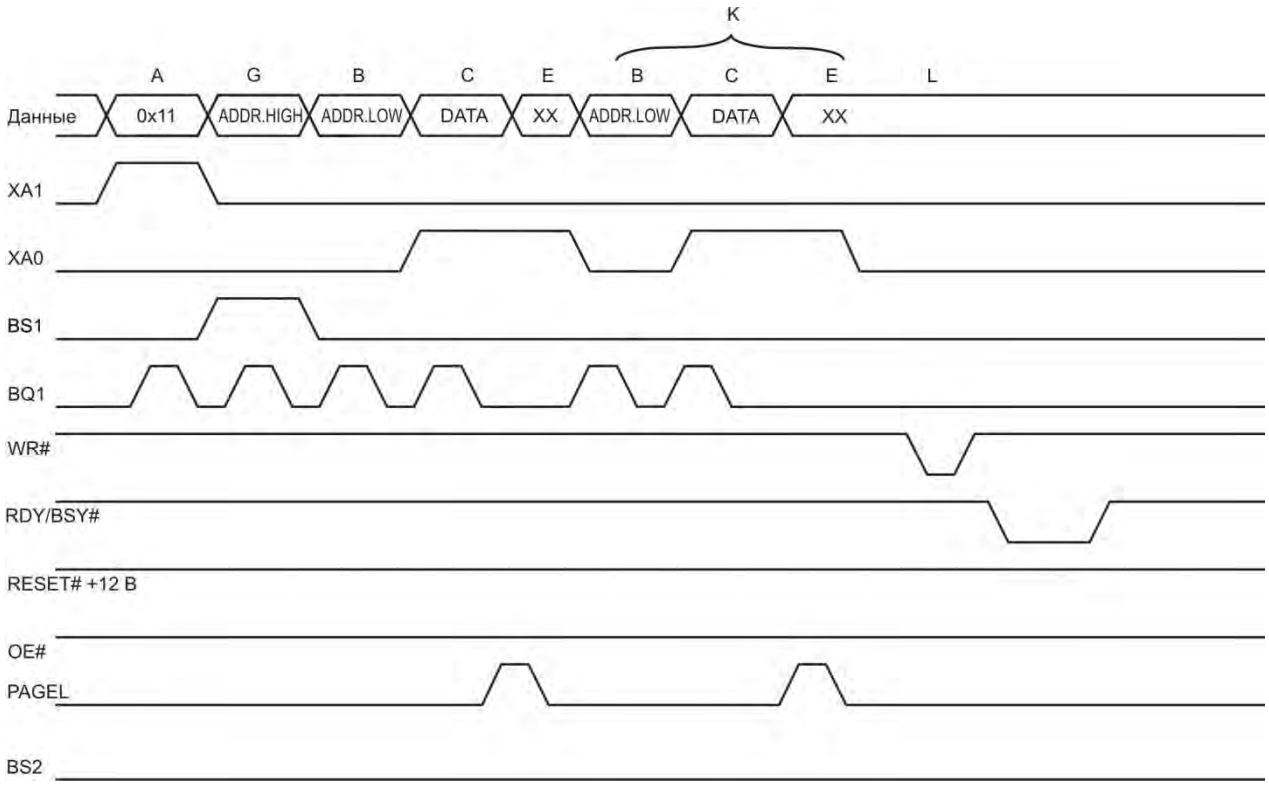


Рисунок 3.139 – Временная диаграмма программирования ЭПД

Чтение памяти программ

Алгоритм чтения памяти программ следующий (см. выше подпункт «Программирование памяти программ» для получения подробностей относительно загрузки команды и адреса):

1. A: Загрузка команды «0000 0010».
2. G: Загрузка старшего адресного байта ($00 – $FF).
3. B: Загрузка младшего адресного байта ($00 – $FF).
4. Установите OE# = 0 и BS1 = 0. Теперь с шины DATA может быть считан младший байт слова.
5. Установите BS1 = 1. Теперь с шины DATA может быть считан старший байт слова.
6. Установите OE# = 1.

Чтение ЭПД

Алгоритм чтения ЭПД следующий (см. выше подпункт «Программирование памяти программ» для получения подробностей относительно загрузки команды и адреса):

1. A: Загрузка команды «0000 0011».
2. G: Загрузка старшего адресного байта ($00 – $FF).
3. B: Загрузка младшего адресного байта ($00 – $FF).
4. Установите OE# = 0 и BS1 = 0. Теперь с шины DATA может быть считан байт данных ЭПД.
5. Установите OE# = 1.

Программирование младших конфигурационных битов

Алгоритм программирования младших конфигурационных битов следующий (см. выше подпункт «Программирование памяти программ» для получения подробностей относительно загрузки команды и данных):

1. A: Загрузка команды «0100 0000».
2. C: Загрузка младшего байта данных. Значение бита n = 0/1 соответствует программированию/стиранию конфигурационного бита.
3. Установите BS1 = 0 и BS2 = 0.
4. Подайте отрицательный импульс на WR# и дождитесь появления высокого уровня на RDY/BSY#.

Программирование старших конфигурационных битов

Алгоритм программирования старших конфигурационных битов следующий (см. выше подпункт «Программирование памяти программ» для получения подробностей относительно загрузки команды и данных):

1. A: Загрузка команды «0100 0000».
2. C: Загрузка младшего байта данных. Значение бита n = 0/1 соответствует программированию/стиранию конфигурационного бита.
3. Установите BS1 = 1 и BS2 = 0. Это позволяет выбрать старший байт данных.
4. Подайте отрицательный импульс на WR# и дождитесь появления высокого уровня на RDY/BSY#.
5. Установите BS1 = 0. Это позволяет выбрать младший байт данных.

Программирование расширенных конфигурационных битов

Алгоритм программирования расширенных конфигурационных битов следующий (см. выше подпункт «Программирование памяти программ» для получения подробностей относительно загрузки команды и данных):

1. A: Загрузка команды «0100 0000».
2. C: Загрузка младшего байта данных. Значение бита n = 0/1 соответствует программированию/стиранию конфигурационного бита.

3. Установите BS2 = 1 и BS1 = 0. Это позволяет выбрать расширенный байт данных.

4. Подайте отрицательный импульс на WR# и дождитесь появления высокого уровня на RDY/BSY#.

5. Установите BS2 = 0. Это позволяет выбрать младший байт данных.

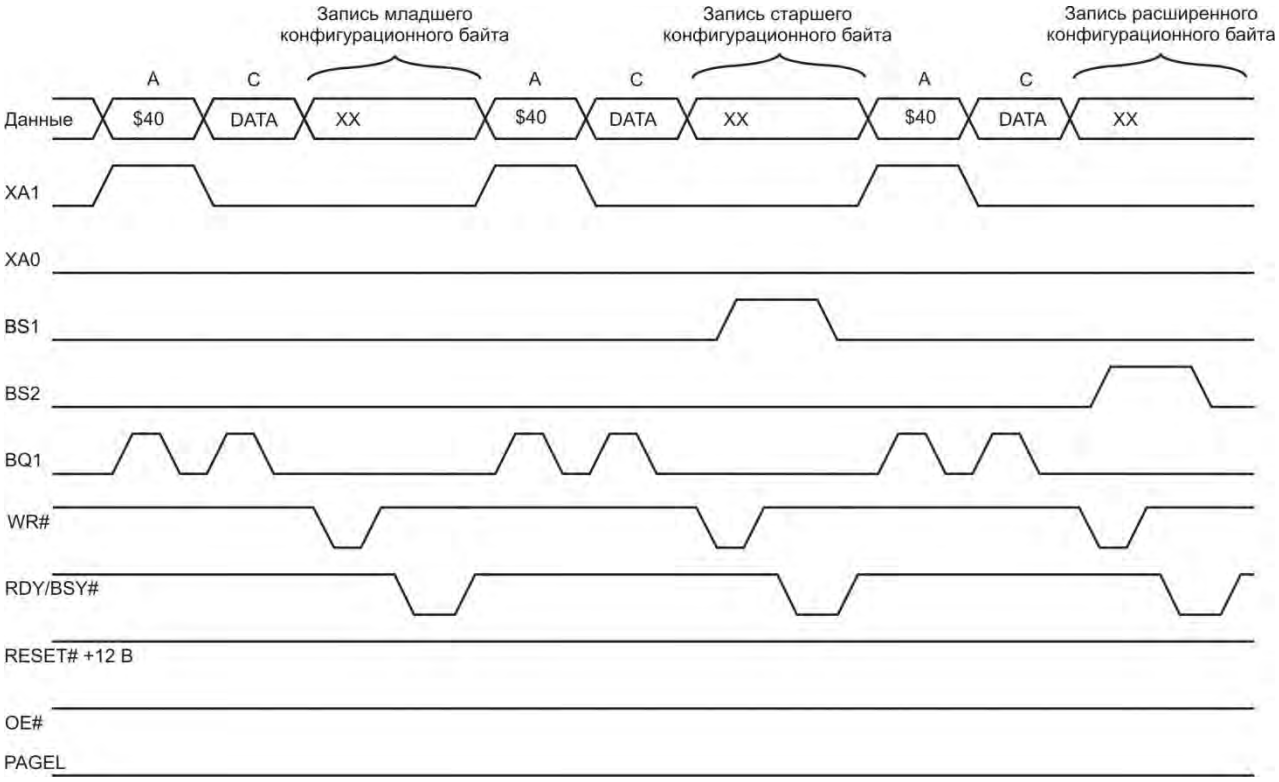


Рисунок 3.140 – Программирование конфигурационных битов

Программирование битов защиты

Алгоритм программирования битов защиты следующий (см. выше подпункт «Программирование памяти программ» для получения подробностей относительно загрузки команды и данных):

1. A: Загрузка команды «0010 0000».

2. C: Загрузка младшего байта данных. Значение бита n = 0 соответствует программированию бита защиты.

3. Подайте отрицательный импульс на WR# и дождитесь появления высокого уровня на RDY/BSY#.

Биты защиты можно очистить только с помощью команды стирания (Chip Erase).

Чтение конфигурационных битов и битов защиты

Алгоритм чтения конфигурационных битов и битов защиты следующий (см. выше подпункт «Программирование памяти программ» для получения подробностей относительно загрузки команды):

1. A: Загрузка команды «0000 0100».

2. Установите OE# = 0, BS2 = 0 и BS1 = 0. Теперь с шины DATA может быть считано состояние младших конфигурационных битов («0» означает запрограммированное состояние).

3. Установите OE# = 0, BS2 = 1 и BS1 = 1. Теперь с шины DATA может быть считано состояние старших конфигурационных битов («0» означает запрограммированное состояние).

4. Установите OE# = 0, BS2 = 1 и BS1 = 0. Теперь с шины DATA может быть считано состояние расширенных конфигурационных битов («0» означает запрограммированное состояние).

5. Установите OE# = 0, BS2 = 0 и BS1 = 1. Теперь с шины DATA может быть считано состояние битов защиты («0» означает запрограммированное состояние).

6. Установите OE# = 1.

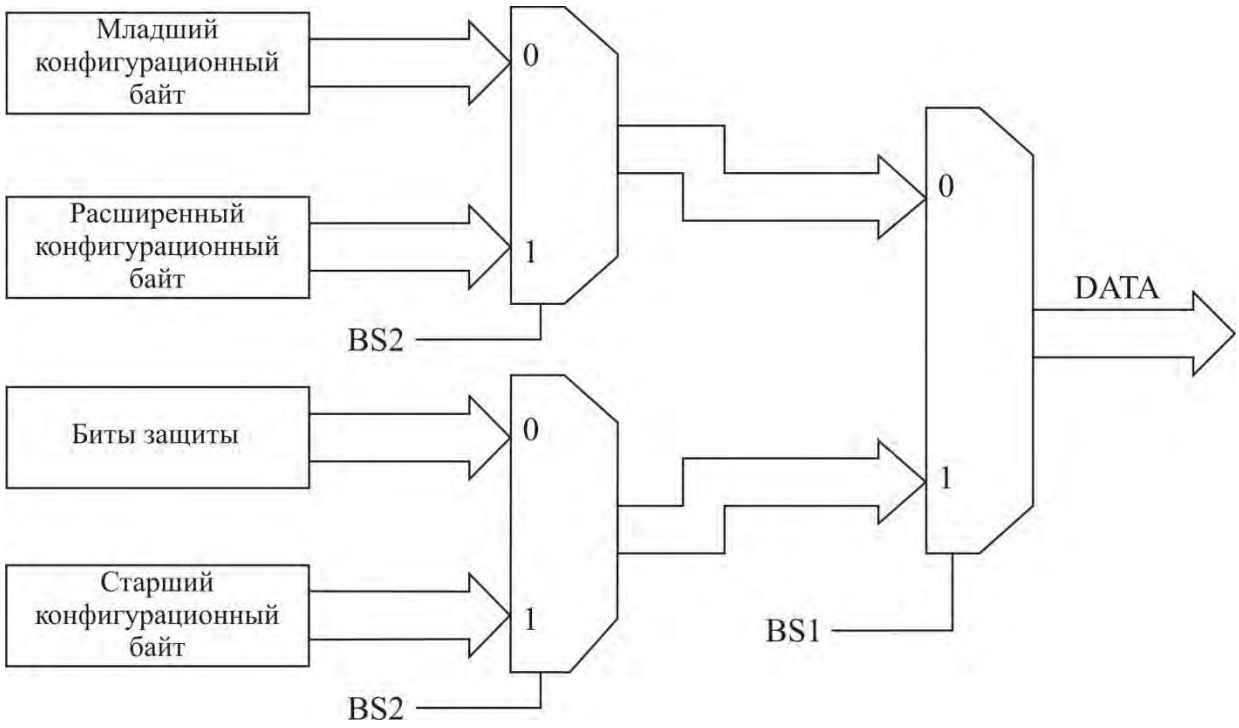


Рисунок 3.141 – Схема считывания конфигурационных битов и битов защиты под управлением сигналов BS1 и BS2

Чтение сигнатурных байтов

Алгоритм чтения сигнатурных байтов следующий (см. выше подпункт «Программирование памяти программ» для получения подробностей относительно загрузки команды и адреса):

1. A: Загрузка команды «0000 1000».
2. B: Загрузка младшего адресного байта ($00 – $02).
3. Установите OE# = 0 и BS1 = 0. Теперь выбранный сигнатурный байт может быть считан с шины DATA.
4. Установите OE# = 1.

Чтение калибровочного байта

Алгоритм чтения калибровочного байта следующий (см. выше подпункт «Программирование памяти программ» для получения подробностей относительно загрузки команды и адреса):

1. A: Загрузка команды «0000 1000».
2. B: Загрузка младшего адресного байта.
3. Установите OE# = 0 и BS1 = 1. Теперь калибровочный байт может быть считан с шины DATA.
4. Установите OE# = 1.

Характеристики параллельного программирования

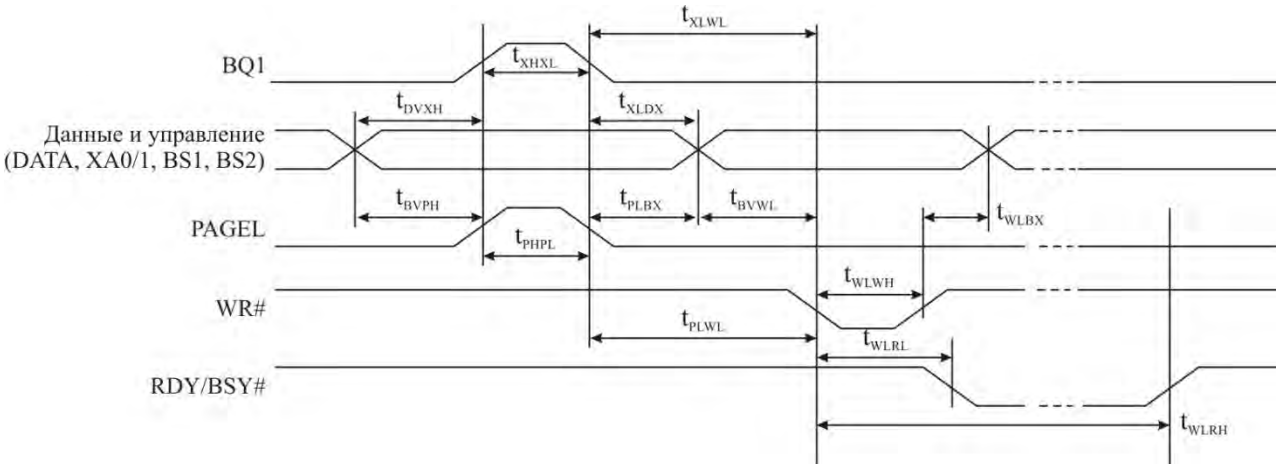


Рисунок 3.142 – Временная диаграмма параллельного программирования: общие требования к временной диаграмме

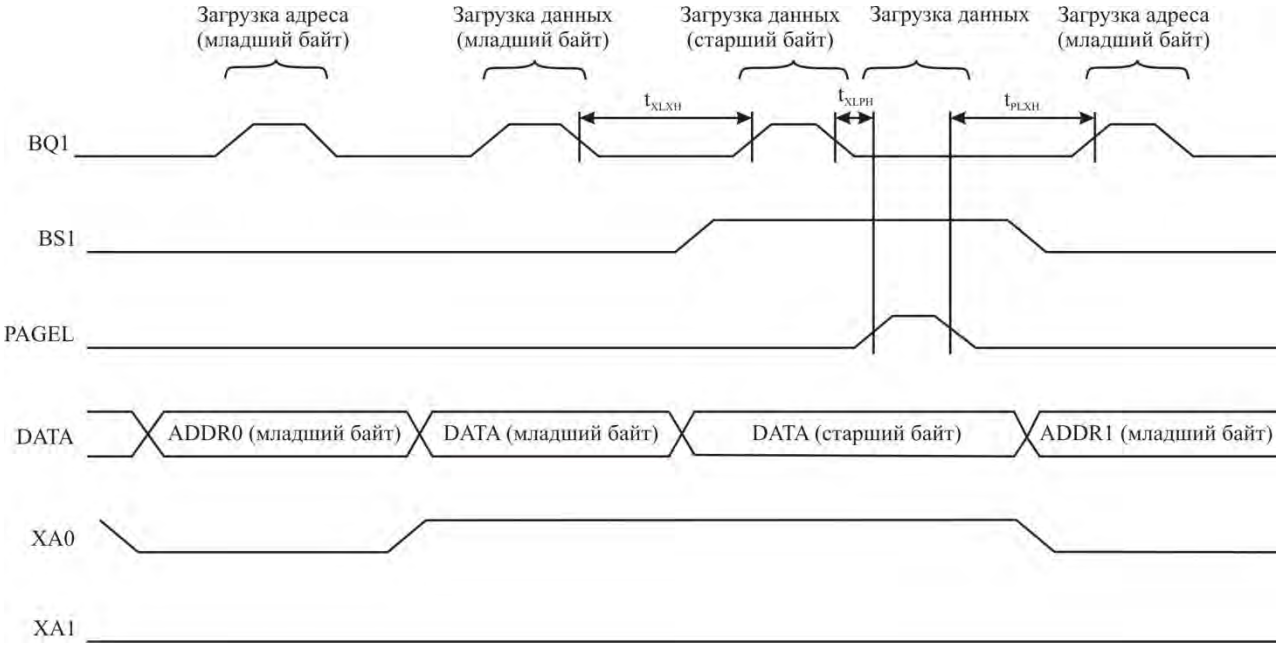


Рисунок 3.143 – Временная диаграмма параллельного программирования: последовательность загрузки и необходимые условия

Примечание – Требования к временной диаграмме, показанные на рисунке 3.142 (то есть t_{DVXH}, t_{XHXL} и t_{XLDX}), также применимы и к операции загрузки.

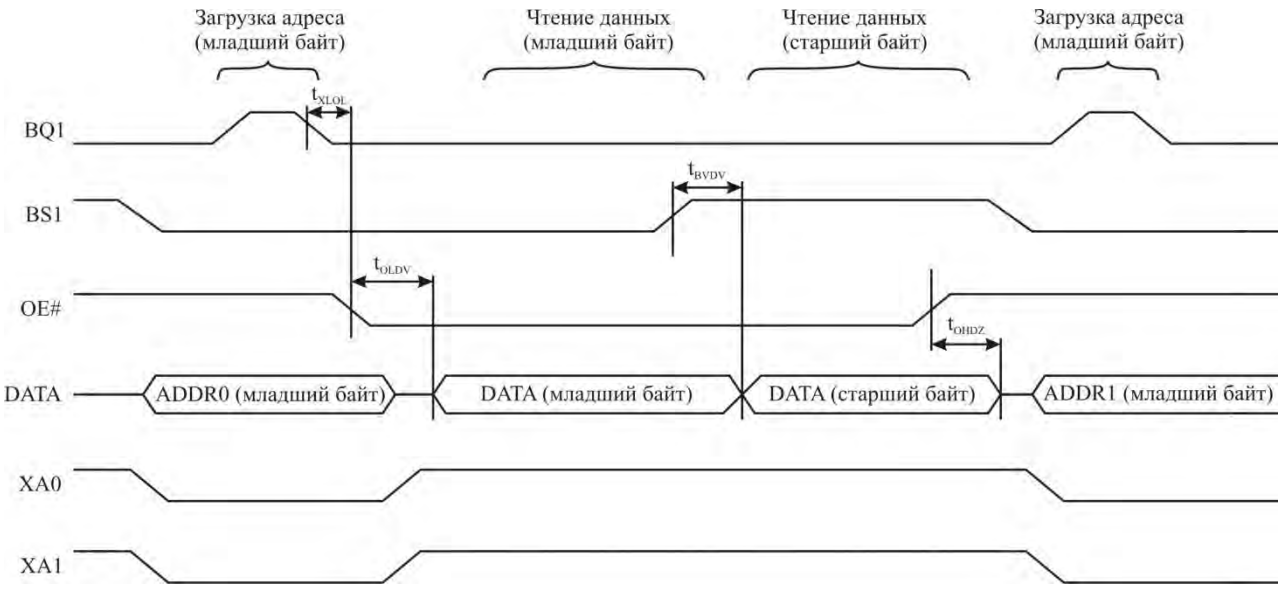


Рисунок 3.144 – Временная диаграмма параллельного программирования: последовательность чтения (в пределах одной страницы) и необходимые условия

Примечание – Требования к временной диаграмме, показанные на рисунке 3.142 (то есть t_{DVXH}, t_{XHXL} и t_{XLDX}), также применимы и к операции чтения.

Таблица 3.126 – Характеристики параллельного программирования при $U_{\#VCC}$ = 5,0 В ± 10 %

Обозначение	Параметр	Мин. значение	Макс. значение	Ед. изм.
V_{PP}	Напряжение разрешения программирования	11,5	12,5	В
I_{PP}	Ток по выводу RESET# при программировании	–	5,0	мА
t_{DVXH}	Действительность данных и управления до появления логической единицы на BQ1	67,0	–	нс
t_{XLXH}	Длительность низкого уровня на BQ1	200	–	нс
t_{XHXL}	Длительность высокого уровня на BQ1	150	–	нс
t_{XLDX}	Хранение данных и управления после того, как на BQ1 появится низкий уровень	67,0	–	нс
t_{XLWL}	Время между появлением логического нуля на BQ1 и логического нуля на WR#	0	–	нс
t_{XLPH}	Время между появлением логического нуля на BQ1 и логической единицы на PAGEL	0	–	нс
t_{PLXH}	Время между появлением логического нуля на PAGEL и логической единицы на BQ1	150	–	нс
t_{BVPH}	Действительность сигнала BS1 до появления логической единицы на PAGEL	67,0	–	нс
t_{PHPL}	Длительность высокого уровня на PAGEL	150	–	нс
t_{PLBX}	Хранение сигнала BS1 после того, как на PAGEL появится низкий уровень	67,0	–	нс
t_{WLBX}	Хранение сигналов BS2/1 после того, как на WR# появится низкий уровень	67,0	–	нс
t_{PLWL}	Время между появлением логического нуля на PAGEL и логического нуля на WR#	67,0	–	нс
t_{BVWL}	Действительность сигнала BS1 до появления логического нуля на WR#	67,0	–	нс

Продолжение таблицы 3.126

Обозначение	Параметр	Мин. значение	Макс. значение	Ед. изм.
t_{WLWH}	Длительность низкого уровня на WR#	150		нс
t_{WLRL}	Время между появлением логического нуля на WR# и логического нуля на RDY/BSY#	0	1,0	мкс
t_{WLRH}	Время между появлением логического нуля на WR# и логической единицы на RDY/BSY#	3,7	5,0	мс
t_{WLRH_CE}	Время между появлением логического нуля на WR# и логической единицы на RDY/BSY# для выполнения команды стирания кристалла Chip Erase	7,5	10	мс
t_{XLOL}	Время между появлением логического нуля на BQ1 и логического нуля на OE#	0	–	нс
t_{BVDV}	Период времени от установления действительности сигнала BS1 до установления действительности сигнала DATA	0	250	нс
t_{OLDV}	Период времени от появления логического нуля на OE# до установления действительности сигнала DATA	–	250	нс
t_{OHDZ}	Время между появлением логической единицы на OE# и переходом сигнала DATA в третье состояние	–	250	нс

3.21.7 Последовательное программирование

Как память программ, так и ЭПД могут быть запрограммированы через последовательную шину интерфейса SPI, когда вывод RESET# имеет низкий логический уровень. Последовательный интерфейс состоит из выводов SCK, MOSI (вход) и MISO (выход). После подачи низкого уровня на вход RESET# необходимо сначала выполнить инструкцию разрешения программирования, чтобы затем можно было выполнять операции программирования/стирания/чтения. В таблице 3.127 представлено описание сигналов программирования. Обратите внимание, что не все выводы последовательного программирования совпадают с выводами внутреннего интерфейса SPI. Также следует отметить, что везде при описании последовательного программирования используются наименования MOSI и MISO для описания последовательного ввода и вывода данных, соответственно. Для ИС 1887ВЕ7Т соответствующие выводы программирования именуются PDI и PDO.

3.21.8 Расположение выводов при последовательном программировании через интерфейс SPI

Несмотря на то что при последовательном программировании используется тот же модуль SPI, что и при обычной работе микроконтроллера, имеется одно важное отличие: выводы MOSI/MISO, которые соответствуют PB2 и PB3 в модуле ввода-вывода, не используются при программировании. Вместо них используются PE0 и PE1 для ввода и вывода данных при последовательном программировании (см. таблицу 3.127).

Таблица 3.127 – Выводы интерфейса SPI при последовательном программировании

Обозначение	Вывод	Вход/выход (I/O)	Описание
MOSI (PDI)	PE0	I	Последовательный ввод данных
MISO (PDO)	PE1	O	Последовательный вывод данных
SCK	PB1	I	Последовательная синхронизация

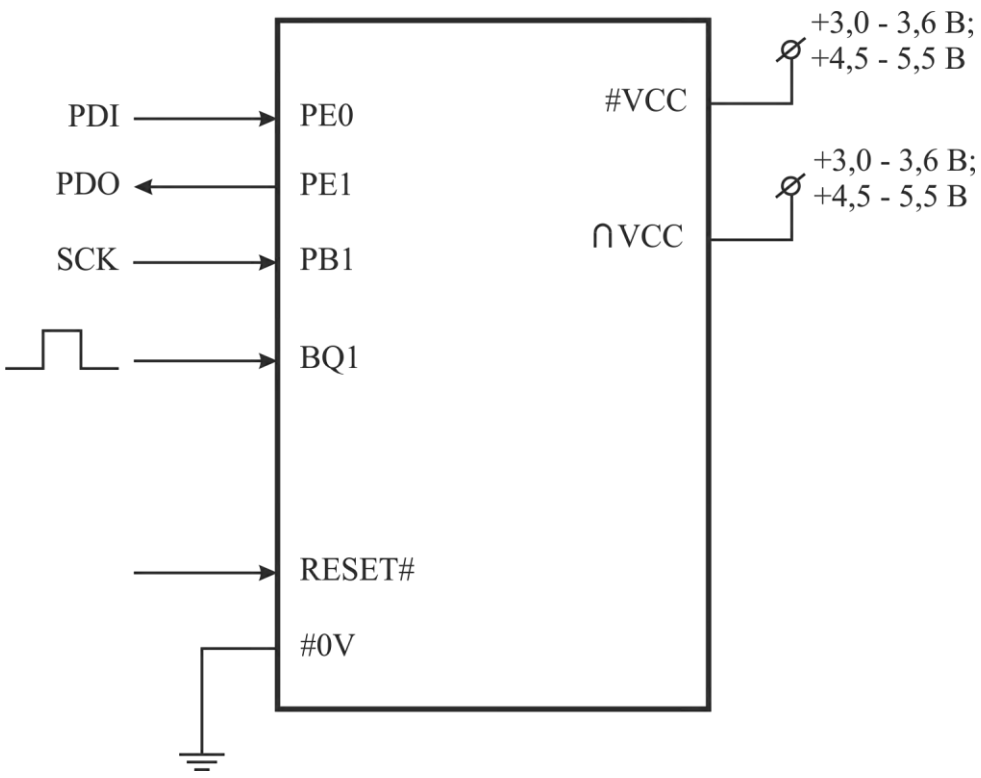


Рисунок 3.145 – Сигналы последовательного программирования

Примечания

1 Если микроконтроллер тактируется внутренним генератором, то нет необходимости подключать источник синхронизации к выводу BQ1.

2 $U_{\#VCC} - 0{,}3\ В < U_{\cap VCC} < U_{\#VCC} + 0{,}3\ В$. Однако напряжение $U_{\cap VCC}$ всегда должно находиться в пределах от 4,5 до 5,5 В.

При программировании ЭПД цикл автоматического стирания является неотъемлемой частью самосинхронизирующейся операции программирования (только в режиме последовательного программирования), и поэтому нет необходимости выполнять сначала команду стирания (Chip Erase). Выполнение команды «Стирание» приводит к заполнению каждой ячейки памяти программ и ЭПД кодом 0x00.

Параметры тактового сигнала зависят от настроек синхронизации микроконтроллера с помощью битов CKSEL. Минимальная длительность периодов низкого и высокого уровней сигнала последовательной синхронизации (SCK) должна соответствовать следующим далее значениям.

Длительность низкого уровня:

- больше двух тактов ЦПУ, если $f_{ck} < 12$ МГц;
- больше трех тактов ЦПУ, если $f_{ck} \geq 12$ МГц.

Длительность высокого уровня:

- больше двух тактов ЦПУ, если $f_{ck} < 12$ МГц;
- больше трех тактов ЦПУ, если $f_{ck} \geq 12$ МГц.

Алгоритм последовательного программирования через SPI

В процессе последовательной записи в ИС 1887ВЕ7Т данные синхронизируются нарастающим фронтом SCK. При выполнении чтения из микроконтроллера данные синхронизируются спадающим фронтом SCK. Временная диаграмма – на рисунке 3.146.

Для программирования и верификации ИС 1887ВЕ7Т в режиме последовательного программирования через SPI необходимо придерживаться следующей последовательности.

1. Последовательность включения питания: подать напряжение между выводами #VCC и GND, когда на входах RESET# и SCK присутствует логический ноль. В некоторых системах программатор не может гарантировать, что SCK удерживается в низком уровне во время подачи питания. В этом случае на RESET# необходимо подать положительный импульс длительностью не менее двух тактов ЦПУ после того, как сигнал SCK принял низкое состояние.

Альтернативно сигналу RESET# можно использовать вывод PEN#. В этом случае сигнал PEN# должен удерживаться в низком уровне во время сброса при подаче питания, пока SCK устанавливается в ноль. Здесь важно только значение PEN# при сбросе. Если программатор не может гарантировать, что SCK удерживается в низком уровне во время подачи питания, то применение метода с сигналом PEN# недопустимо. Необходимо отключить питание устройства, чтобы начать нормальную работу при использовании данного метода.

2. Ожидание не менее 20 мс и разрешение последовательного программирования путем подачи команды разрешения программирования на вход MOSI.

3. Инструкции последовательного программирования не выполняются, если отсутствует синхронизация связи. Наличие синхронизации сообщается путем выдачи микроконтроллером кода ($53) при записи третьего байта инструкции разрешения программирования. Несмотря на то, корректно полученное значение или нет, все четыре байта инструкции должны быть переданы. Если значение $53 не получено, то следует подать положительный импульс на вход RESET# и ввести новую команду разрешения программирования.

4. Память программ программируется постранично. Размер страницы указан в таблице 3.124. Страница памяти загружается побайтно путем подачи семи младших значащих разрядов адреса и данных вместе с инструкцией загрузки страницы памяти программ. Чтобы гарантировать корректность загрузки страницы, необходимо сначала записать младший байт данных, а затем старший байт по заданному адресу. Страница памяти программ сохраняется путем ввода инструкции для записи страницы памяти программ с девятью старшими значащими разрядами адреса. Если опрос не используется, то пользователь должен ожидать период времени не менее t_{WD_FLASH} перед загрузкой следующей страницы (см. таблицу 3.128).

Примечание – Если какая-либо другая команда кроме опроса (чтения) вводится перед завершением любой операции записи (память программ, ЭПД, биты защиты, конфигурационные биты), то программирование может закончиться некорректно.

5. Массив памяти ЭПД программируется побайтно путем подачи адреса и данных вместе с соответствующей инструкцией записи. Перед записью новых данных вначале выполняется автоматическое стирание ячейки памяти ЭПД. Если опрос не используется, то пользователь должен ожидать период времени не менее t_{WD_EEPROM} перед вводом следующего байта (см. таблицу 3.128).

6. Любую ячейку памяти можно проверить, используя инструкцию чтения, которая возвращает содержимое ячейки по указанному адресу путем последовательной передачи на выходе MISO.

7. По завершении программирования вход RESET# можно перевести в высокое состояние для начала нормальной работы.

8. Последовательность выключения питания (при необходимости):

- установите RESET# равным 1;
- отключите питание #VCC.

Время программирования памяти программ, ЭПД, длительность стирания

В таблице 3.128 приведены длительности циклов записи памяти программ, ЭПД, конфигурационных байт, а также длительность стирания (Chip Erase).

Таблица 3.128 – Минимальные длительности задержек перед записью очередной ячейки памяти программ или ЭПД, $U_{\#VCC}$ = 5,0 В ± 10 %

Обозначение	Минимальная задержка, мс
t_{WD_FUSE}	4,5
t_{WD_FLASH}	5
t_{WD_EEPROM}	10
t_{WD_ERASE}	1800

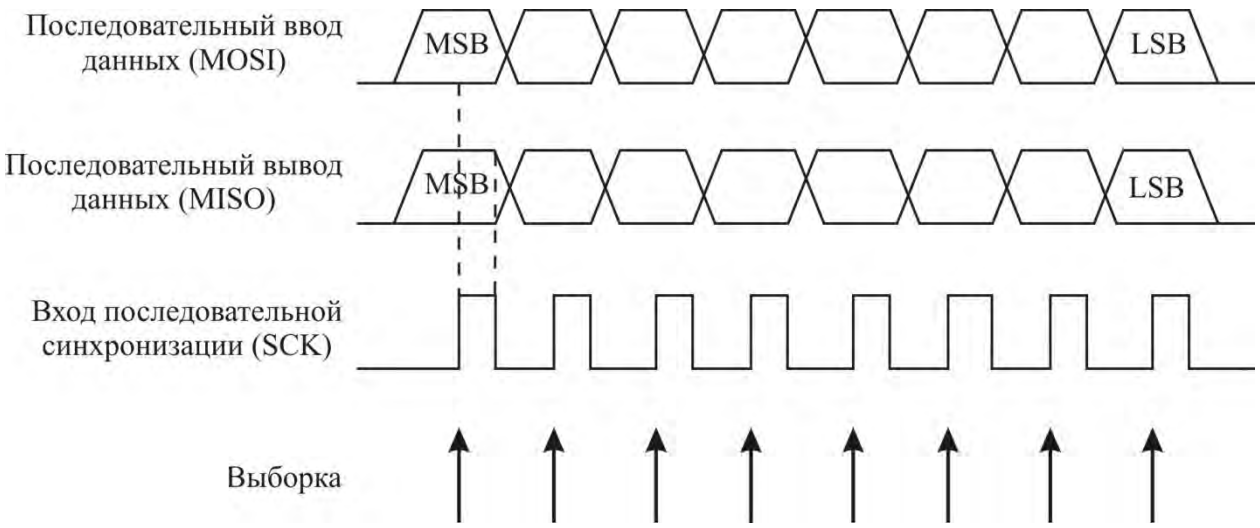


Рисунок 3.146 – Временная диаграмма сигналов при последовательном программировании через SPI

Таблица 3.129 – Набор команд для последовательного программирования через SPI

Команда	Формат команды				Операция
	Байт 1	Байт 2	Байт 3	Байт 4	
Разрешение программирования	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Разрешение последовательного программирования после того, как RESET# примет нулевое состояние
Стирание (Chip Erase)	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Стирание ЭПД и памяти программ
Чтение памяти программ	0010 H000	aaaa aaaa	bbbb bbbb	oooo oooo	Считывание старшего или младшего байта (H) данных o из памяти программ по адресу a:b
Загрузка страницы памяти программ	0100 H000	xxxx xxxx	xbbb bbbb	iiii iiii	Запись старшего или младшего байта (H) данных i в страницу памяти программ по адресу b.
Запись страницы памяти программ	0100 1100	aaaa aaaa	bxxx xxxx	xxxx xxxx	Запись страницы памяти программ по адресу a:b
Чтение памяти ЭПД	1010 0000	xxxx aaaa	bbbb bbbb	oooo oooo	Чтение данных o из памяти ЭПД по адресу a:b
Запись памяти ЭПД	1100 0000	xxxx aaaa	bbbb bbbb	iiii iiii	Запись данных i в память ЭПД по адресу a:b

Продолжение таблицы 3.129

Команда	Формат команды				Операция
	Байт 1	Байт 2	Байт 3	Байт 4	
Чтение битов защиты	0101 1000	0000 0000	xxxx xxxx	xxoo oooo	Чтение битов защиты. «0» означает запрограммированное, «1» - незапрограммированное состояние (см. таблицу 3.115)
Запись битов защиты	1010 1100	111x xxxx	xxxx xxxx	11ii iiii	Запись битов защиты. Чтобы запрограммировать эти биты, установите их в ноль (см. таблицу 3.115)
Чтение сигнатурного байта	0011 0000	xxxx xxxx	xxxx xxbb	oooo oooo	Чтение сигнатурного байта «o» по адресу b
Запись конфигурационных битов	1010 1100	1010 0000	xxxx xxxx	iiii iiii	Запись конфигурационных битов. Чтобы запрограммировать эти биты, установите их в ноль. Чтобы перевести их в незапрограммированное состояние – установите в единицу (см. таблицу 3.119)
Запись старших конфигурационных битов	1010 1100	1010 1000	xxxx xxxx	iiii iiii	Запись старших конфигурационных битов. Чтобы запрограммировать эти биты, установите их в ноль. Чтобы перевести их в незапрограммированное состояние – установите в единицу (см. таблицу 3.118)
Запись расширенных конфигурационных битов	1010 1100	1010 0100	xxxx xxxx	xxxx xxii	Запись расширенных конфигурационных битов. Чтобы запрограммировать эти биты, установите их в ноль. Чтобы перевести их в незапрограммированное состояние – установите в единицу (см. таблицу 3.119)
Чтение конфигурационных битов	0101 0000	0000 0000	xxxx xxxx	oooo oooo	Чтение конфигурационных битов. «0» означает запрограммированное, «1» - незапрограммированное состояние (см. таблицу 3.119)
Чтение расширенных конфигурационных битов	0101 0000	0000 1000	xxxx xxxx	oooo oooo	Чтение расширенных конфигурационных битов. «0» означает запрограммированное, «1» - незапрограммированное состояние (см. таблицу 3.119)

Продолжение таблицы 3.129

Команда	Формат команды				Операция
	Байт 1	Байт 2	Байт 3	Байт 4	
Чтение старших конфигурационных битов	0101 1000	0000 1000	xxxx xxxx	oooo oooo	Чтение старших конфигурационных битов. «0» означает запрограммированное, «1» - незапрограммированное состояние (см. таблицу 3.118)
Чтение калибровочного байта	0011 1000	xxxx xxxx	0000 00bb	oooo oooo	Чтение калибровочного байта o по адресу b.

Примечания
a – старшие биты адреса;
b – младшие биты адреса;
H: 0 – младший байт, 1 – старший байт;
o – выходные данные;
i – входные данные;
x – может иметь произвольное значение.

3.21.9 Программирование через интерфейс JTAG

При программировании через интерфейс JTAG необходимо контролировать четыре специфических вывода JTAG-интерфейса: TCK, TMS, TDI и TDO. Управление выводами сброса и тактирования микроконтроллера не требуется.

Для активизации JTAG-интерфейса необходимо запрограммировать конфигурационный бит JTAGEN. Устройство поставляется заказчику с установленным данным битом по умолчанию. Помимо этого необходимо сбросить бит JTD в регистре MCUCSR. Альтернативно, если бит JTD установлен, можно принудительно перевести внешний вход сброса в низкое состояние. Тогда спустя два такта синхронизации бит JTD будет очищен, а выводы JTAG-интерфейса станут доступными для программирования. Этим обеспечивается возможность использования выводов JTAG-интерфейса в качестве обычных линий ввода-вывода в процессе работы, в то время как все еще разрешено внутрисистемное программирование через JTAG-интерфейс. Обратите внимание, что данный метод нельзя применять при использовании выводов JTAG для периферийного сканирования или внутренней отладки. В данном случае выводы JTAG-интерфейса должны использоваться только для этих целей.

Также необходимо выделить, что как при последовательной передаче, так и при последовательном приеме, первым передается младший разряд всех сдвиговых регистров.

Специфические JTAG-инструкции программирования

Регистр инструкций является четырехразрядным и поддерживает 16 инструкций. JTAG-инструкции, используемые для программирования, представлены ниже.

Код операции для каждой инструкции показывается следом за наименованием инструкции в шестнадцатеричном формате. Текст описывает, который из регистров данных выбран в качестве канала между TDI и TDO для каждой инструкции.

Состояние «Запуск теста / Ожидание» TAP-контроллера используется для генерации внутренней синхронизации. Оно также может использоваться как промежуточное состояние простоя между последовательностями JTAG. Последовательность автомата конечных состояний для изменения слова инструкции представлена на рисунке 3.147.

AVR_RESET ($C)

Специфическая инструкция для принудительного ввода микроконтроллера в режим задания начальных условий или вывода его из данного состояния. TAP-контроллер не сбра-

сывается при выполнении данной инструкции. В качестве регистра данных выступает одноразрядный регистр сброса. Обратите внимание, что микроконтроллер будет непрерывно находиться в состоянии сброса, пока в цепи сброса присутствует логическая единица. Выход данной цепи не защелкивается. Единственным активным состоянием является:

- Сдвиг РД: регистр сброса сдвигается под управлением входа TCK.

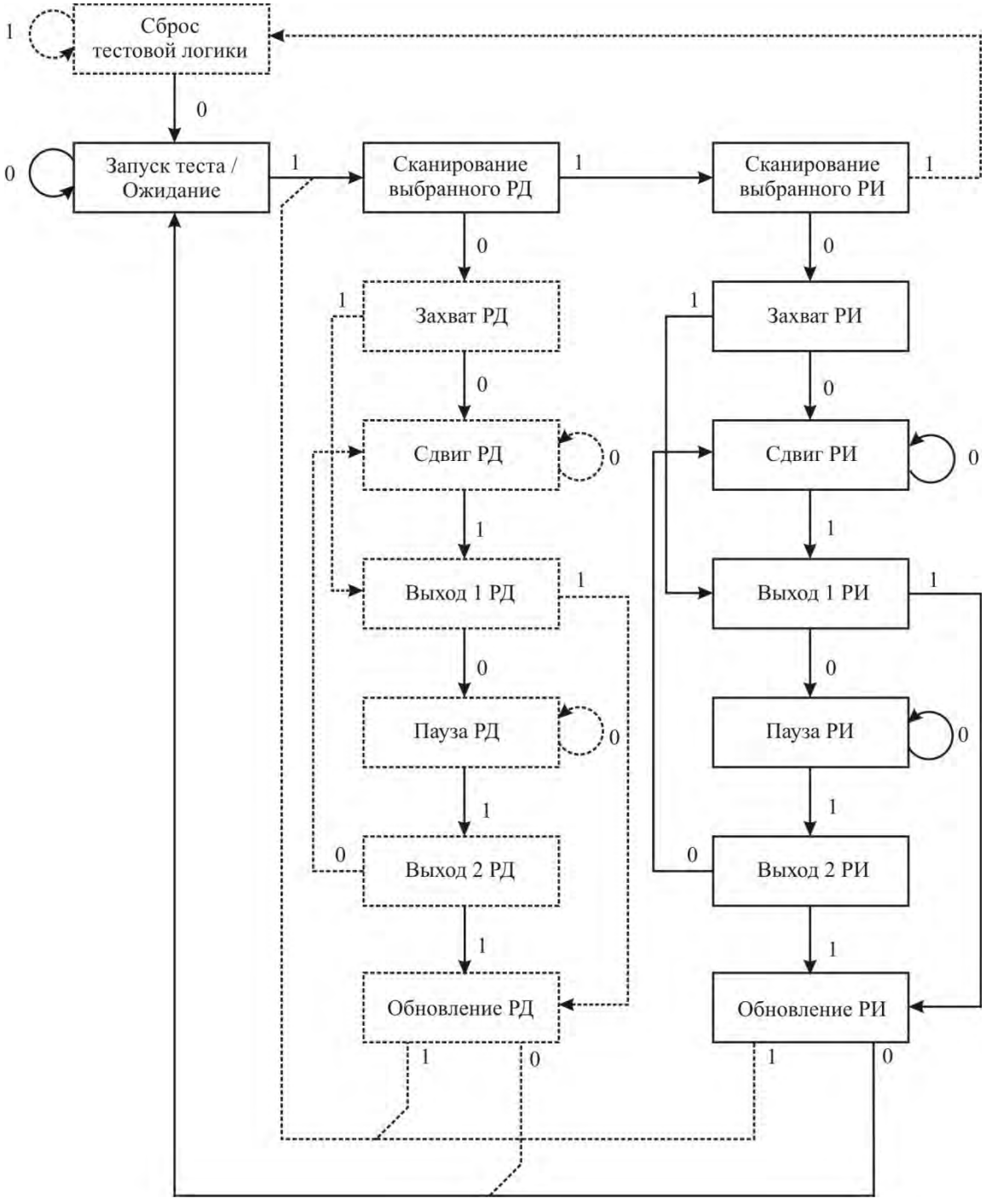


Рисунок 3.147 – Последовательность автомата конечных состояний для изменения слова инструкции

PROG_ENABLE ($4)

Специфическая инструкция для разрешения программирования через JTAG-порт. В качестве регистра данных выбирается 16-разрядный регистр разрешения программирования. Активными состояниями являются:

- Сдвиг РД: сигнатурный код разрешения программирования загружается в регистр данных.

- Обновление РД: сигнатурный код разрешения программирования сравнивается с корректным значением, и если он действителен, то осуществляется переход в режим программирования.

PROG_COMMANDS ($5)

Специфическая инструкция для ввода команд программирования через порт JTAG. В качестве регистра данных выбирается 15-разрядный регистр команд программирования. Активными состояниями являются:

- Захват РД: результат предыдущей команды загружается в регистр данных.

- Сдвиг РД: регистр данных сдвигается под управлением тактового входа TCK, сдвигом выводится результат предыдущей команды и вводится новая команда.

- Обновление РД: команда программирования поступает на входы памяти программ.

- Запуск теста / Ожидание: генерируется один такт синхронизации, выполняется загруженная команда.

PROG_PAGELOAD ($6)

Специфическая инструкция для непосредственной загрузки страницы данных памяти программ через JTAG-порт. В качестве регистра данных выбирается 2048-битный виртуальный регистр загрузки страницы памяти программ. Он представляет собой виртуальную цепь сканирования с длиной, равной количеству битов в одной странице памяти программ. Сдвиговый регистр является 8-разрядным. В отличие от большинства JTAG-инструкций состояние «Обновление РД» не используется для передачи данных из сдвигового регистра. Данные автоматически передаются в буфер страницы памяти программ байт за байтом в состоянии «Сдвиг РД» с помощью внутреннего автомата конечных состояний. Единственным активным состоянием является сдвиг РД: данные вводятся через TDI под управлением TCK и автоматически загружаются в страницу памяти программ побайтно.

Примечание – JTAG-инструкция PROG_PAGELOAD может использоваться, только если микроконтроллер является первым устройством в цепи сканирования. Если микроконтроллер не может быть первым устройством в цепи сканирования, то необходимо применять алгоритм побайтного программирования.

PROG_PAGEREAD ($7)

Специфическая инструкция для полного считывания одной страницы памяти программ через порт JTAG. В качестве регистра данных выбирается 2056-битный виртуальный регистр чтения страницы памяти программ. Он представляет собой виртуальную цепь сканирования с длиной, равной количеству битов в одной странице памяти программ плюс восемь. Изнутри сдвиговый регистр является 8-разрядным. В отличие от большинства JTAG-инструкций состояние «Захват РД» не используется для передачи данных в сдвиговый регистр. Данные автоматически передаются из буфера страницы памяти программ байт за байтом в состоянии «Сдвиг РД» с помощью внутреннего автомата конечных состояний. Единственным активным состоянием является сдвиг РД: данные памяти программ автоматически считываются побайтно и передаются сдвигом на TDO под управлением входа TCK. Вход TDI игнорируется.

Примечание – JTAG-инструкция PROG_PAGEREAD может использоваться, только если микроконтроллер является первым устройством в цепи сканирования. Если микроконтроллер не может быть первым устройством в цепи сканирования, то необходимо применять алгоритм побайтного программирования.

Регистры данных

Регистры данных выбираются с помощью регистров JTAG-инструкций, как описано выше в подпункте «Специфические JTAG-инструкции программирования». Ниже перечислены регистры данных, которые относятся к операциям программирования:

- Регистр сброса.
- Регистр разрешения программирования.
- Регистр команд программирования.
- Виртуальный регистр загрузки страницы памяти программ.
- Виртуальный регистр чтения страницы памяти программ.

Регистр сброса

Регистр сброса является тестовым регистром данных и используется для сброса микроконтроллера в процессе программирования. Необходимо выполнять сброс перед входом в режим программирования.

Логическая единица в регистре сброса соответствует установке низкого уровня на внешнем входе сброса. Микроконтроллер будет находиться в состоянии сброса, пока в регистре сброса присутствует логическая единица. В зависимости от настроек конфигурационных битов для синхронизации микроконтроллер будет оставаться в состоянии сброса в течение периода простоя (см. 3.3.2 «Источники синхронизации») после записи логического нуля в регистр сброса. Выход данного регистра не синхронизирован, поэтому сброс наступает незамедлительно, как показано на рисунке 3.124.

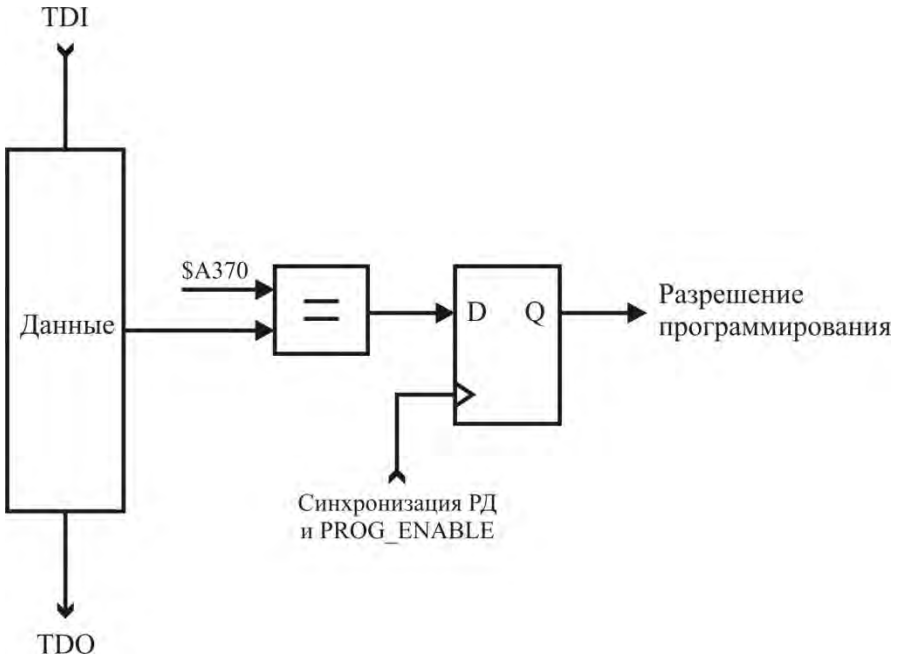


Рисунок 3.148 – Регистр разрешения программирования

Регистр разрешения программирования

Регистр разрешения программирования является 16-разрядным. Содержимое этого регистра сравнивается с сигнатурным кодом разрешения программирования «1010_0011_0111_0000». Когда содержимое регистра совпадает с сигнатурным кодом, программирование через порт JTAG разрешается. Регистр принимает нулевое состояние после сброса при подаче питания и всегда должен сбрасываться при выходе из режима программирования.

Регистр команд программирования

Регистр команд программирования является 15-разрядным регистром. Данный регистр используется для последовательного ввода команд программирования и последова-

тельного вывода результата предыдущей команды, если таковая имелась (рисунок 3.149). Набор команд для программирования через JTAG-интерфейс приведен в таблице 3.130. Последовательность состояний при загрузке команд программирования представлена на рисунке 3.150.

Рисунок 3.149 – Регистр команд программирования

Таблица 3.130 – Набор команд для программирования через JTAG-интерфейс

Команда	Последовательность TDI	Последовательность TDO	Примечания
1a. Стирание (Chip Erase)	0100011_10000000 0110001_10000000 0110011_10000000 0110011_10000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
1b. Опрос завершения стирания [2)]	0110011_10000000	xxxxxox_xxxxxxxx	
2a. Переход к записи памяти программ	0100011_00010000	xxxxxxx_xxxxxxxx	
2b. Загрузка старшего байта адреса [9)]	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	
2c. Загрузка младшего байта адреса	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
2d. Загрузка младшего байта данных	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
2e. Загрузка старшего байта данных	0010111_iiiiiiii	xxxxxxx_xxxxxxxx	
2f. Защелкивание данных [1)]	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
2g. Запись страницы памяти программ [1)]	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	

Продолжение таблицы 3.130

Команда	Последовательность TDI	Последовательность TDO	Примечания
2h. Опрос завершения записи страницы памяти программ [2)]	0110111_00000000	xxxxxox_xxxxxxxx	
3a. Переход к чтению памяти программ	0100011_00000010	xxxxxxx_xxxxxxxx	
3b. Загрузка старшего байта адреса [9)]	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	
3c. Загрузка младшего байта адреса	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
3d. Чтение младшего и старшего байта данных	0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo xxxxxxx_oooooooo	 млад. байт стар. байт
4a. Переход к записи ЭПД	0100011_00010001	xxxxxxx_xxxxxxxx	
4b. Загрузка старшего байта адреса [9)]	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	
4c. Загрузка младшего байта адреса	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
4d. Загрузка байта данных	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
4e. Защелкивание данных [1)]	0110111_00000000 1110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
4f. Запись страницы ЭПД [1)]	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
4g. Опрос завершения записи страницы ЭПД [2)]	0110011_00000000	xxxxxox_xxxxxxxx	
5a. Переход к чтению ЭПД	0100011_00000011	xxxxxxx_xxxxxxxx	
5b. Загрузка старшего байта адреса [9)]	0000111_aaaaaaaa	xxxxxxx_xxxxxxxx	
5c. Загрузка младшего байта адреса	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
5d. Чтение байта данных	0110011_bbbbbbbb 0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
6a. Переход к записи конфигурационных битов	0100011_01000000	xxxxxxx_xxxxxxxx	
6b. Загрузка младшего байта данных [3),6)]	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
6c. Запись расширенного конфигурационного байта [1)]	0111011_00000000 0111001_00000000 0111011_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
6d. Опрос завершения записи конфигурационных битов [2)]	0110111_00000000	xxxxxox_xxxxxxxx	
6e. Загрузка младшего байта данных [3),7)]	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
6f. Запись старшего конфигурационного байта [1)]	0110111_00000000 0110101_00000000 0110111_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
6g. Опрос завершения записи конфигурационных битов [2)]	0110111_00000000	xxxxxox_xxxxxxxx	
6h. Загрузка младшего байта данных [3),7)]	0010011_iiiiiiii	xxxxxxx_xxxxxxxx	
6i. Запись младшего конфигурационного байта [1)]	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	

Продолжение таблицы 3.130

Команда	Последовательность TDI	Последовательность TDO	Примечания
6j. Опрос завершения записи конфигурационных битов [2)]	0110011_00000000	xxxxxox_xxxxxxxx	
7a. Переход к записи битов защиты	0100011_00100000	xxxxxxx_xxxxxxxx	
7b. Загрузка байта данных [4),9)]	0010011_11iiiiii	xxxxxxx_xxxxxxxx	
7c. Запись битов защиты [1)]	0110011_00000000 0110001_00000000 0110011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	
7d. Опрос завершения записи битов защиты [2)]	0110011_00000000	xxxxxox_xxxxxxxx	
8a. Переход к чтению конфигурационных битов / битов защиты	0100011_00000100	xxxxxxx_xxxxxxxx	
8b. Чтение расширенного конфигурационного байта [6)]	0111010_00000000 0111011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
8c. Чтение старшего конфигурационного байта [7)]	0111110_00000000 0111111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
8d. Чтение младшего конфигурационного байта [8)]	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
8e. Чтение битов защиты [5),9)]	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxoooooo	
8f. Чтение конфигурационных битов и битов защиты [5)]	0111010_00000000 0111110_00000000 0110010_00000000 0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo xxxxxxx_oooooooo xxxxxxx_oooooooo xxxxxxx_oooooooo	расш. конф. байт стар. конф. байт млад. конф. байт биты защиты
9a. Переход к чтению сигнатурного байта	0100011_00001000	xxxxxxx_xxxxxxxx	
9b. Загрузка адресного байта	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
9c. Чтение сигнатурного байта	0110010_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
10a. Переход к чтению калибр. байта	0100011_00001000	xxxxxxx_xxxxxxxx	
10b. Загрузка адресного байта	0000011_bbbbbbbb	xxxxxxx_xxxxxxxx	
10c. Чтение калибровочного байта	0110110_00000000 0110111_00000000	xxxxxxx_xxxxxxxx xxxxxxx_oooooooo	
11a. Загрузка команды «Нет операции»	0100011_00000000 0110011_00000000	xxxxxxx_xxxxxxxx xxxxxxx_xxxxxxxx	

[1)] Эта последовательность команд не требуется, если семь старших значащих разрядов корректно установлены с помощью предыдущей последовательности команд (что обычно имеет место).

[2)] Повторяйте до тех пор, пока ноль не станет равным единице.

[3)] Чтобы запрограммировать конфигурационные биты, установите соответствующие разряды в ноль. Чтобы перевести в незапрограммированное состояние – установите их в единицу.

[4)] Чтобы запрограммировать биты защиты, установите соответствующие разряды в ноль.

[5)] «0» означает запрограммированное, «1» - незапрограммированное состояние.

[6)] Информация по расположению расширенного конфигурационного байта представлена в таблице 3.117.

[7)] Информация по расположению старшего конфигурационного байта представлена в таблице 3.118.

[8)] Информация по расположению младшего конфигурационного байта представлена в таблице 3.119.

Продолжение таблицы 3.130

[9] Информация по расположению битов защиты представлена в таблице 3.115.

Примечание – Обозначения в таблице: a – старшие биты адреса; b – младшие биты адреса; 0 – младший байт, 1 – старший байт; o – выходные данные; i – входные данные; x – может иметь произвольное значение.

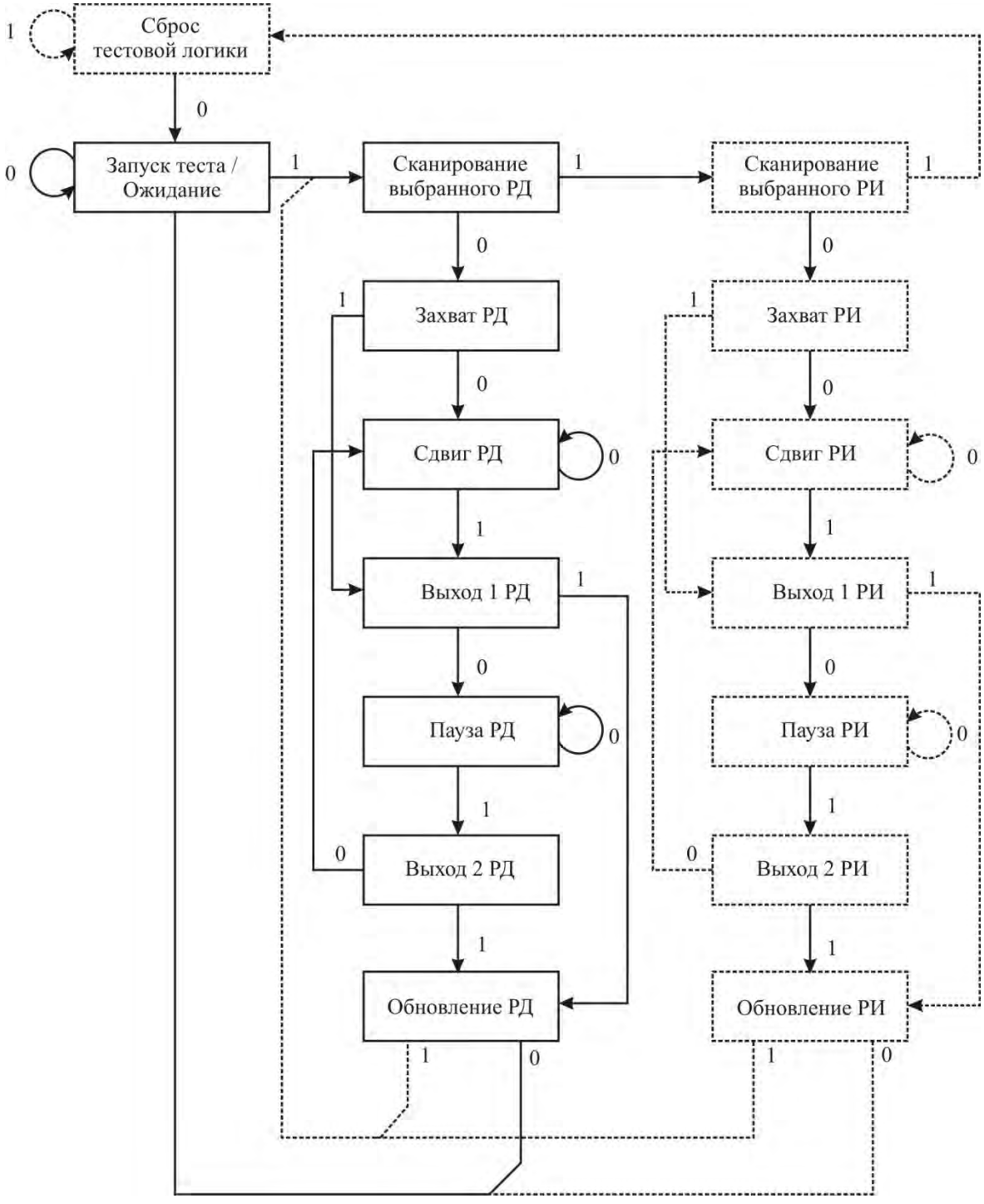


Рисунок 3.150 – Последовательность автомата конечных состояний для изменения/чтения слова данных

Виртуальный регистр загрузки страницы памяти программ

Виртуальный регистр загрузки страницы памяти программ представляет собой виртуальную цепь сканирования с длиной равной количеству битов в одной странице памяти программ. Изнутри сдвиговый регистр является 8-разрядным. Данные автоматически пере-

даются в буфер страницы памяти программ байт за байтом. Слова инструкций вводятся сдвигом, начиная с младшего разряда первой инструкции и заканчивая старшим разрядом последней инструкции в пределах одной и той же страницы. Этим обеспечивается эффективный способ загрузки всего буфера страницы памяти программ перед выполнением записи страницы.

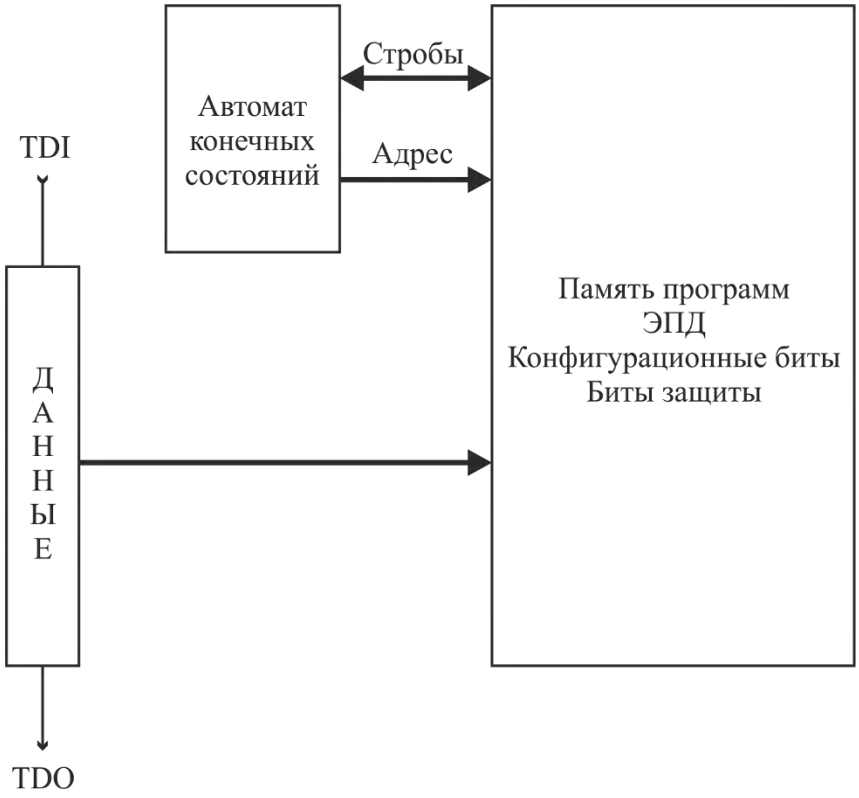


Рисунок 3.151 – Виртуальный регистр загрузки страницы памяти программ

Виртуальный регистр чтения страницы памяти программ

Виртуальный регистр чтения страницы памяти программ представляет собой виртуальную цепь сканирования с длиной, равной количеству битов в одной странице памяти программ плюс восемь. Изнутри сдвиговый регистр является 8-разрядным. Данные автоматически передаются из страницы памяти программ байт за байтом. Первые восемь тактов используются для передачи первого байта во внутренний сдвиговый регистр, и биты, выводимые сдвигом в течение этих восьми тактов, следует игнорировать. После такой инициализации данные выводятся, начиная с младшего разряда первой инструкции текущей страницы и заканчивая старшим разрядом последней инструкции этой же страницы. Этим обеспечивается эффективный способ чтения целой страницы памяти программ для проверки результата программирования.

Алгоритм программирования

Ниже используются ссылки на коды инструкций «1a», «1b» и т.п. Информация о них представлена в таблице 3.130.

Переход в режим программирования

1. Введите JTAG-инструкцию AVR_RESET и загрузите значение «1» в регистр сброса.

2. Введите инструкцию PROG_ENABLE и загрузите значение «1010_0011_0111_0000» в регистр разрешения программирования.

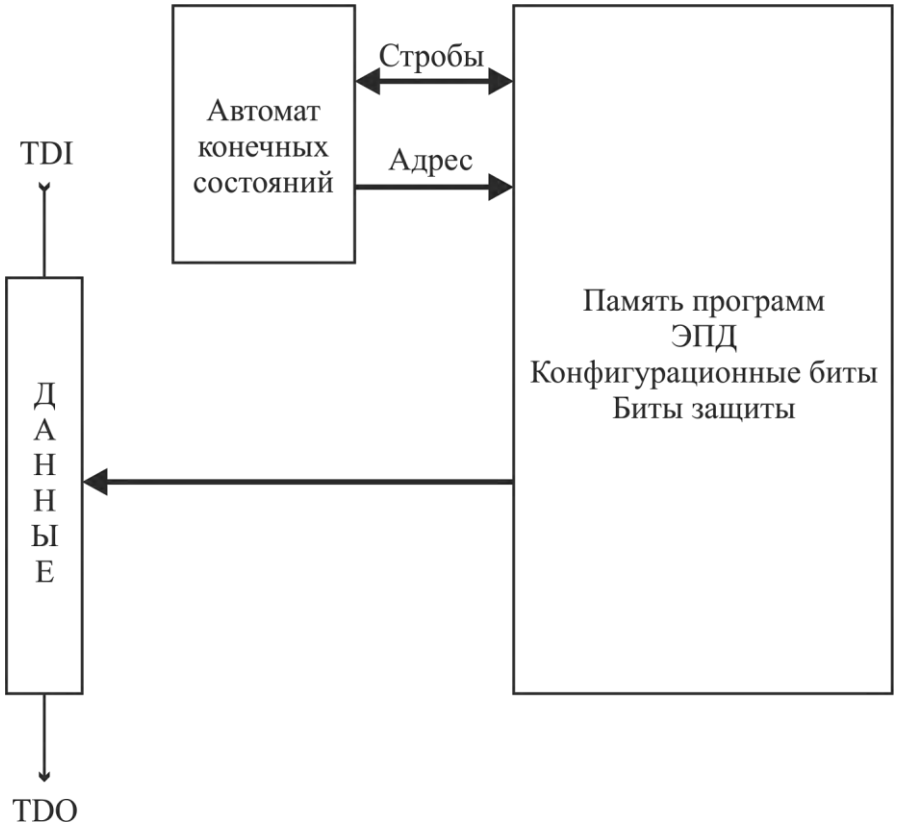


Рисунок 3.152 – Виртуальный регистр чтения страницы памяти программ

Выход из режима программирования

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Отключите все инструкции программирования с помощью команды «нет операции» 11a.
3. Введите инструкцию PROG_ENABLE и загрузите значение «0000_0000_0000_0000» в регистр разрешения программирования.
4. Введите JTAG-инструкцию AVR_RESET и загрузите значение «0» в регистр сброса.

Стирание (Chip Erase)

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Запустите стирание с помощью инструкции программирования 1a.
3. Убедитесь в завершении стирания с помощью инструкции программирования 1b или подождите период времени t_{WLRH_CE} (см. таблицу 3.126).

Программирование памяти программ

Перед программированием памяти программ можно как выполнять, так и не выполнять операцию «Стирание (Chip Erase)» (см. подпункт выше).

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Разрешите запись памяти программ, используя инструкцию программирования 2a.
3. Загрузите старший байт адреса с помощью инструкции программирования 2b.
4. Загрузите младший байт адреса с помощью инструкции программирования 2c.
5. Загрузите данные с помощью инструкций программирования 2d, 2e и 2f.
6. Повторите шаги 4 и 5 для всех командных слов в странице.
7. Выполните запись страницы с помощью инструкции программирования 2g.
8. Убедитесь в завершении записи памяти программ, используя инструкцию программирования 2h, или подождите период времени t_{WLRH} (см. таблицу 3.126).
9. Повторяйте шаги 3 – 7 до тех пор, пока не будут запрограммированы все данные.

Более эффективная передача данных может быть достигнута при использовании инструкции PROG_PAGELOAD:

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Разрешите запись памяти программ, используя инструкцию программирования 2a.
3. Загрузите адрес страницы, используя инструкции программирования 2b и 2c. PCWORD (см. таблицу 3.124) используется для адресации в пределах одной страницы, и в него необходимо записать ноль.
4. Введите JTAG-инструкцию PROG_PAGELOAD.
5. Загрузите целую страницу путем сдвига всех командных слов в странице, начиная с младшего разряда первой инструкции текущей страницы и заканчивая старшим разрядом последней инструкции этой же страницы.
6. Введите JTAG-инструкцию PROG_COMMANDS.
7. Выполните запись страницы с помощью инструкции программирования 2g.
8. Убедитесь в завершении записи памяти программ, используя инструкцию программирования 2h, или подождите период времени t_{WLRH} (см. таблицу 3.126).
9. Повторяйте шаги 3 – 8 до тех пор, пока не будут запрограммированы все данные.

Чтение памяти программ

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Разрешите чтение памяти программ, используя инструкцию программирования 3a.
3. Загрузите адрес с помощью инструкций программирования 3b и 3c.
4. Выполните чтение данных с помощью инструкции программирования 3d.
5. Повторяйте шаги 3 и 4 до тех пор, пока не будут считаны все данные.

Более эффективная передача данных может быть достигнута при использовании инструкции PROG_PAGEREAD:

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Разрешите чтение памяти программ, используя инструкцию программирования 3a.
3. Загрузите адрес страницы, используя инструкции программирования 3b и 3c. PCWORD (см. таблицу 3.123) используется для адресации в пределах одной страницы, и в него необходимо записать ноль.
4. Введите JTAG-инструкцию PROG_PAGEREAD.
5. Выполните чтение целой страницы путем сдвига всех командных слов в странице, начиная с младшего разряда первой инструкции текущей страницы и заканчивая старшим разрядом последней инструкции этой же страницы. Помните, что первые 8 сдвигаемых битов следует игнорировать.
6. Введите JTAG-инструкцию PROG_COMMANDS.
7. Повторяйте шаги 3 – 6 до тех пор, пока не будут считаны все данные.

Программирование ЭПД

Перед программированием ЭПД можно как выполнять, так и не выполнять операцию «Стирание».

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Разрешите запись в ЭПД, используя инструкцию программирования 4a.
3. Загрузите старший байт адреса с помощью инструкции программирования 4b.
4. Загрузите младший байт адреса с помощью инструкции программирования 4c.
5. Загрузите данные с помощью инструкций программирования 4d и 4e.
6. Повторите шаги 4 и 5 для всех байтов данных в странице.
7. Выполните запись данных с помощью инструкции программирования 4f.
8. Убедитесь в завершении записи ЭПД, используя инструкцию программирования 4g, или подождите период времени t_{WLRH} (см. таблицу 3.126).
9. Повторяйте шаги 3 – 8 до тех пор, пока не будут запрограммированы все данные.

Обратите внимание! Инструкция PROG_PAGELOAD не может быть использована для программирования ЭПД.

Чтение ЭПД

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Разрешите чтение ЭПД, используя инструкцию программирования 5a.
3. Загрузите адрес с помощью инструкций программирования 5b и 5c.
4. Выполните чтение данных с помощью инструкции программирования 5d.
5. Повторяйте шаги 3 и 4 до тех пор, пока не будут считаны все данные.

Внимание! Инструкция PROG_PAGEREAD не может быть использована для чтения ЭПД.

Программирование конфигурационных битов

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Разрешите запись конфигурационных битов, используя инструкцию 6a.
3. Загрузите байт данных с помощью инструкции программирования 6b. Запись «0» приводит к программированию соответствующего конфигурационного бита, а запись «1» приводит его в незапрограммированное состояние.
4. Выполните запись расширенного конфигурационного байта с помощью инструкции программирования 6c.
5. Убедитесь в завершении записи конфигурационных битов, используя инструкцию программирования 6d, или подождите период времени t_{WLRH} (см. таблицу 3.126).
6. Загрузите байт данных с помощью инструкции программирования 6e. Запись «0» приводит к программированию соответствующего конфигурационного бита, а запись «1» приводит его в незапрограммированное состояние.
7. Выполните запись старшего конфигурационного байта с помощью инструкции программирования 6f.
8. Убедитесь в завершении записи конфигурационных битов, используя инструкцию программирования 6g, или подождите период времени t_{WLRH} (см. таблицу 3.126).
9. Загрузите байт данных с помощью инструкции программирования 6h. Запись «0» приводит к программированию соответствующего конфигурационного бита, а запись «1» приводит его в незапрограммированное состояние.
10. Выполните запись младшего конфигурационного байта с помощью инструкции программирования 6i.
11. Убедитесь в завершении записи конфигурационных битов, используя инструкцию программирования 6j, или подождите период времени t_{WLRH} (см. таблицу 3.126).

Программирование битов защиты

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Разрешите запись битов защиты, используя инструкцию программирования 7a.
3. Загрузите данные с помощью инструкции программирования 7b. Запись «0» приводит к программированию соответствующего бита защиты, а запись «1» оставляет этот бит в прежнем состоянии.
4. Выполните запись битов защиты с помощью инструкции программирования 7c.
5. Убедитесь в завершении записи битов защиты, используя инструкцию программирования 7d, или подождите период времени t_{WLRH} (см. таблицу 3.126).

Чтение конфигурационных битов и битов защиты

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Разрешите чтение конфигурационного бита/бита защиты, используя инструкцию программирования 8a.
3. Для чтения всех конфигурационных битов и битов защиты используйте инструкцию программирования 8f.

Для чтения только расширенного конфигурационного байта используйте инструкцию программирования 8b.

Для чтения только старшего конфигурационного байта используйте инструкцию программирования 8c.

Для чтения только младшего конфигурационного байта используйте инструкцию программирования 8d.

Для чтения только битов защиты используйте инструкцию программирования 8e.

Чтение сигнатурных байтов

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Разрешите чтение сигнатурного байта, используя инструкцию программирования 9a.
3. Загрузите адрес $00 с помощью инструкции программирования 9b.
4. Выполните чтение первого сигнатурного байта, используя инструкцию программирования 9c.
5. Повторите шаги 3 и 4 с адресами $01 и $02 для чтения 2-го и 3-го сигнатурного байта, соответственно.

Чтение калибровочного байта

1. Введите JTAG-инструкцию PROG_COMMANDS.
2. Разрешите чтение калибровочного байта, используя инструкцию программирования 10a.
3. Загрузите адрес $00 с помощью инструкции программирования 10b.
4. Выполните чтение калибровочного байта, используя инструкцию программирования 10c.

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, относится 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 восьмого по четвертый). 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, У 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 шесть команд (по две 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 байт 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.

Микроконтроллер поддерживает четыре 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 с преддекрементом содержимое индексного регистра сначала уменьшается на единицу, а затем производится обращение по полученному адресу.

Микроконтроллер поддерживает шесть команд (по две 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».

Микроконтроллер поддерживает шесть команд (по две 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 ИС 1887ВЕ7Т

Система команд микроконтроллера насчитывает 133 различных инструкции. 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 символических 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 два или три 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, в 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
…                  ; текст основной программы
sp_test            ; метка подпрограммы
push r2            ; сохранить r2 в стеке
…                  ; 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 три команды:
- NOP – нет операции;
- SLEEP - перевод микроконтроллера в режим пониженного энергопотребления;
- WDR - сброс сторожевого таймера.

Все команды этой группы выполняются за один машинный цикл.

6 Отладочные средства

6.1 Отладочное устройство КФДЛ.301411.243

Отладочное устройство для микроконтроллера 1887ВЕ7Т предназначено для отладки и программирования систем, разрабатываемых на его основе, с использованием внутрисистемного программирования через последовательный интерфейс, параллельного высоковольтного программирования, а также программирования и отладки через интерфейс JTAG.

Отладочное устройство (см. рисунок 6.1) представляет собой завершенный стартовый набор и систему для проектирования на основе микроконтроллера 1887ВЕ7Т. С его помощью разработчик сможет оперативно приступить к разработке программного кода и использовать широкие возможности стартового набора по макетированию и проверке новых решений.

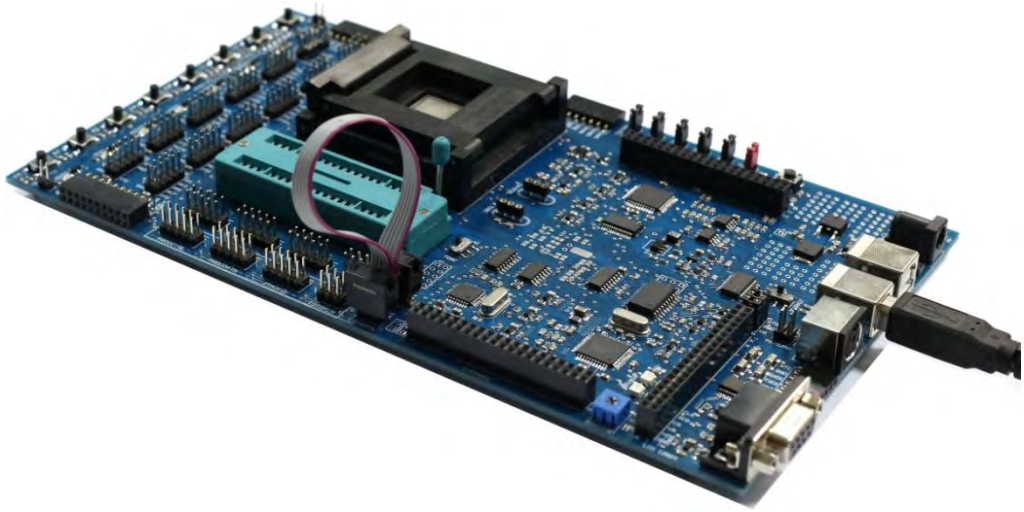


Рисунок 6.1 – Отладочное устройство КФДЛ.301411.243

Для подключения к ПК, с установленным на нем программным пакетом AVR Studio, на плате отладочного устройства предусмотрен порт USB. Для поддержки всего набора микроконтроллеров на плате предусмотрен стабилизатор напряжения VTG, который может регулироваться в пределах от 0,9 до 5,5 В с шагом 0,1 В.

Для задания опорного уровня в аналоговом тракте целевого микроконтроллера на плате устройства предусмотрен аналоговый ИОН. ИОН (AREF) поддерживает возможность регулировки в пределах от 0 до 5,5 В с шагом 100 мВ.

С помощью программируемого тактового генератора разработчик может выбирать тактовые частоты микроконтроллера в пределах от 1,1 кГц до 3,686 МГц. Установленный на плате кварцевый генератор способен работать совместно как с керамическими резонаторами, так и с кварцевыми резонаторами, частота которых лежит в пределах от 4 до 24 МГц.

Все линии ввода-вывода целевого микроконтроллера выведены на разъемы отладочной платы. Для облегчения макетирования и отладки предусмотрена возможность подключения к линиям ввода-вывода восьми светодиодов и кнопок.

Отладочное устройство имеет следующие функциональные элементы и характеристики:

- внутрисистемный последовательный интерфейс программирования ISP;
- высоковольтный параллельный интерфейс программирования;
- интерфейс программирования и отладки JTAG;
- регулируемый стабилизированный источник питания (0,7 – 6,0) В;
- регулируемый источник опорного напряжения;
- внутренний кварцевый генератор;

- генератор тактового сигнала в диапазоне от 0 до 3,686 МГц;
- разъем для подключения часового кварцевого резонатора;
- контактное устройство для подключения 64-выводной ИС 1887ВЕ7Т;
- контактное устройство с нулевым усилием, позволяющее подключать ИС в 40-выводном DIP корпусе и микроконтроллер 1887ВЕ4У через дополнительный переходник 48-DIP40 ТР, выпускаемый АО «НИИЭТ» специально для данной микросхемы;
- доступ ко всем портам ввода-вывода ИС;
- расширительные разъемы для подключения внешних устройств и плат макетирования;
- индикаторные светодиоды, которые можно подключать к отлаживаемому микроконтроллеру;
- кнопочные переключатели, подключаемые к портам ИС;
- порт USB для связи с персональным компьютером;
- коммуникационный порт RS232;
- совместимость с программной средой AVR Studio фирмы ATMEL;
- совместимость с программатором КФДЛ.301411.247 АО «НИИЭТ»;
- совместимость со сторонними AVR программаторами.

Дополнительную информацию по отладочному устройству КФДЛ.301411.243 можно найти на сайте АО «НИИЭТ» в разделе «Средства для программирования и отладки».

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).

6.2 Средства программирования

Для программирования ИС 1887ВЕ7Т можно рекомендовать USB-программатор КФДЛ.301411.247, выпускаемый АО «НИИЭТ» (дополнительная информация – на сайте в разделе «Средства для программирования и отладки»).

Также для программирования и отладки микроконтроллера 1887ВЕ7Т можно использовать другие программно-отладочные комплексы, такие как STK500, STK600, JTAGICE MKII, JTAGICE3, Atmel-ICE, AVRISP MKII и др.

Заключение

В настоящем руководстве пользователя приведено подробное описание архитектуры, функционального построения, системы команд и особенностей применения ИС 1887ВЕ7Т, которая представляет собой 8-разрядный микроконтроллер с AVR RISC-архитектурой, тактовой частотой 16 МГц, содержащий 128 Кбайт памяти программ, 4 Кбайт ЭПД данных, 4 Кбайт ОЗУ, 8-канальный 10-разрядный АЦП, последовательный периферийный интерфейс SPI, двухпроводной последовательный интерфейс TWI, программируемые последовательные порты USART.

Микроконтроллер предназначен для применения в системах управления робототехническими комплексами, системах автоматизации технологических процессов, а также в системах автоматизированного управления электроприводом, телекоммуникационной технике и т. д.

Все значения электрических параметров микросхем приведены в технических условиях АЕЯР.431280.910ТУ. Значения параметров, приведенные в настоящем РП, являются справочными.

Руководство пользователя может служить практическим руководством по применению микроконтроллера для разработчиков систем на основе ИС 1887ВЕ7Т и программистов.

Приложение А
(обязательное)

Описание системы команд

В таблице А.1 приведена группа команд арифметических и логических операций.
В таблице А.2 приведена группа команд операций перехода.
В таблице А.3 приведена группа команд передачи данных.
В таблице А.4 приведена группа команд битовых операций и операций контроля битов.
В таблице А.5 приведена группа команд управления системой.

Таблица А.1 – Группа команд арифметических и логических операций

Мнемокод	Операнды	Описание	Действие	Флаги	Количество машинных циклов
ADD	Rd, Rr	Сложить два регистра	Rd ← Rd + Rr	Z,C,N, V, S H	1
ADC	Rd, Rr	Сложить два регистра с переносом	Rd ← Rd + Rr + C	Z,C,N, V,S H	1
ADIW	Rdl, K	Сложить слово с константой	Rd+1:Rd ← Rd+1:Rd + K	Z,C,N, V,S	2
SUB	Rd, Rr	Вычесть два регистра	Rd ← Rd – Rr	Z,C,N, V, SH	1
SUBI	Rd, K	Вычесть константу из регистра	Rd ← Rd - K	Z,C,N, V, SH	1
SBC	Rd, Rr	Вычесть два регистра с учетом переноса	Rd ← Rd - Rr - C	Z,C,N, V, SH	1
SBCI	Rd, K	Вычесть константу из регистра с учетом переноса	Rd ← Rd - K - C	Z,C,N, V, S H	1
SBIW	Rdl, K	Вычесть константу из слова	Rd+1:Rd ← Rd+1:Rd - K	Z,C,N, V,S	2
AND	Rd, Rr	Логическое И между регистрами	Rd ← Rd * Rr	Z,N,V	1
ANDI	Rd, K	Логическое И между регистром и константой	Rd ← Rd * K	Z,N,V	1
OR	Rd, Rr	Логическое ИЛИ между регистрами	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Логическое ИЛИ между регистром и константой	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Искл. ИЛИ между регистрами	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	Дополнение до 0b11111111 ($FF), инверсия	Rd ← $FF - Rd	Z,C,N, V	1
NEG	Rd	Дополнение до 0b00000000 ($00)	Rd ← $00 - Rd	Z,C,N, V,H	1
SBR	Rd, K	Установка бит (бита) в регистре	Rd ← Rd v K	Z,N,V	1
CBR	Rd, K	Сброс бит (бита) в регистре	Rd ← Rd · ($FF - K)	Z,N,V	1
INC	Rd	Инкремент	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Декремент	Rd ← Rd - 1	Z,N,V	1
TST	Rd	Проверка на ноль или минус	Rd ← Rd A* Rd	Z,N,V	1

Продолжение таблицы А.1

Мнемокод	Операнды	Описание	Действие	Флаги	Количество машинных циклов
CLR	Rd	Сброс регистра	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Установка регистра	Rd ← $FF	Нет	1
MUL	Rd, Rr	Умножение без знака	R1:R0 ← Rd x Rr(UU)	Z, C	2
MULS	Rd, Rr	Умножение со знаком	R1:R0 ← Rd x Rr(SS)	Z, C	2
MULSU	Rd, Rr	Умножение знакового с беззнаковым числом	R1:R0 ← Rd x Rr(SU)	Z, C	2
FMUL	Rd. Rr	Дробное умножение без знака	R1:R0 ← (Rd x Rr) << 1(UU)	Z, C	2
FMULS	Rd, Rr	Дробное умножение со знаком	R1:R0 ← (Rd x Rr) << 1(SS)	Z, C	2
FMULSU	Rd, Rr	Дробное умножение знакового с беззнаковым числом	R1:R0 ← (Rd x Rr) << 1(SU)	Z, C	2

Таблица А.2 – Группа команд операций перехода

Мнемокод	Операнды	Описание	Действие	Флаги	Количество машинных циклов
RJMP	k	Относительный переход	PC ← PC + k +1	Нет	2
IJMP		Косвенный переход по указателю (Z)	PC ← Z	Нет	2
JMP	k	Безусловный переход	PC ← k	Нет	3
RCALL	k	Относительный вызов процедуры	PC ← PC + k + 1	Нет	3
ICALL		Косвенный вызов процедуры по указателю (Z)	PC ← Z	Нет	3
CALL	k	Безусловный вызов процедуры	PC ← k	Нет	4
RET		Возврат из подпрограммы	PC ← STACK	Нет	4
RETI		Возврат из прерывания	PC ← STACK	I	4
CPSE	Rd, Rr	Сравнение и пропуск, если равно	if (Rd = Rr) PC ← PC + 2 или 3	Нет	1 / 2 / 3
CP	Rd, Rr	Сравнение	Rd - Rr	Z,N,V, C,H	1
CPC	Rd, Rr	Сравнение с учетом переноса	Rd - Rr - C	Z,N,V, C,H	1
CPI	Rd, K	Сравнение регистра с константой	Rd - K	Z,N,V, C,H	1
SBRC	Rr, b	Пропуск, если бит в регистре сброшен	if (Rr(b) = 0) PC ← PC + 2 или 3	Нет	1 / 2 / 3
SBRS	Rr, b	Пропуск, если бит в регистре установлен	if (Rr(b) = 1) PC ← PC + 2 или 3	Нет	1 / 2 / 3
SBIC	P, b	Пропуск, если бит в регистре ввода-вывода сброшен	if (I/O (P(b) = 0) PC ← PC + 2 или 3	Нет	1 / 2 / 3
SBIS	P, b	Пропуск, если бит в регистре ввода-вывода установлен	if (I/O (P(b) = 1) PC ← PC + 2 или 3	Нет	1 / 2 / 3
BRBS	s, k	Переход, если флаг состояния установлен	if (SREG(s) = 1) then PC ← PC + k + 1	Нет	1 / 2

Продолжение таблицы А.2

Мнемокод	Операнды	Описание	Действие	Флаги	Количество машинных циклов
BRBC	s, k	Переход, если флаг состояния сброшен	if (SREG(s) = 0) then PC ← PC + k + 1	Нет	1 / 2
BREQ	k	Переход, если равно	if (Z = 1) then PC ← PC + k + 1	Нет	1 / 2
BRNE	k	Переход, если не равно	if (Z = 0) then PC ← PC + k + 1	Нет	1 / 2
BRCS	k	Переход, если перенос установлен	if (C = 1) then PC ← PC + k + 1	Нет	1 / 2
BRCC	k	Переход, если перенос сброшен	if (C = 0) then PC ← PC + k + 1	Нет	1 / 2
BRSH	k	Переход, если больше или равно	if (C = 0) then PC ← PC + k + 1	Нет	1 / 2
BRLO	k	Переход, если меньше	if (C = 1) then PC ← PC + k + 1	Нет	1 / 2
BRMI	k	Переход, если минус	if (N = 1) then PC ← PC + k + 1	Нет	1 / 2
BRPL	k	Переход, если плюс	if (N = 0) then PC ← PC + k + 1	Нет	1 / 2
BRGE	k	Переход, если больше или равно с учетом знака	if (N ⊕ V = 0) then PC ← PC + k + 1	Нет	1 / 2
BRLT	k	Переход, если меньше нуля с учетом знака	if (N ⊕ V = 1) then PC ← PC + k + 1	Нет	1 / 2
BRHS	k	Переход, если флаг H установлен	if (H = 1) then PC ← PC + k + 1	Нет	1 / 2
BRHC	k	Переход, если флаг H сброшен	if (H = 0) then PC ← PC + k + 1	Нет	1 / 2
BRTS	k	Переход, если флаг T установлен	if (T = 1) then PC ← PC + k +1	Нет	1 / 2
BRTC	k	Переход, если флаг T сброшен	if (T = 0) then PC ← PC + k + 1	Нет	1 / 2
BRVS	k	Переход, если флаг V установлен	if (V = 1) then PC ← PC + k + 1	Нет	1 / 2
BRVC	k	Переход, если флаг V сброшен	if (V = 0) then PC ← PC + k + 1	Нет	1 / 2
BRIE	k	Переход, если прерывания разрешены	if (I = 1) then PC ← PC + k + 1	Нет	1 / 2
BRID	k	Переход, если прерывания запрещены	if (I = 0) then PC ← PC + k + 1	Нет	1 / 2

Таблица А.3 – Группа команд передачи данных

Мнемокод	Операнды	Описание	Действие	Флаги	Количество машинных циклов
MOV	Rd, Rr	Запись из регистра в регистр	Rd ← Rr	Нет	1
MOVW	Rd, Rr	Перезапись слова между регистрами	Rd + 1:Rd ← Rr + 1:Rr	Нет	1
LDI	Rd, K	Запись константы в регистр	Rd ← K	Нет	1
LD	Rd, X	Косвенное считывание из памяти в регистр	Rd ← (X)	Нет	2

Продолжение таблицы А.3

Мнемокод	Операнды	Описание	Действие	Флаги	Количество машинных циклов
LD	Rd, X+	Косвенное считывание из памяти в регистр и инкр.	Rd ← (X), X ← X + 1	Нет	2
LD	Rd, -X	Предварительный декремент, а затем косвенное считывание из памяти в регистр	X ← X - 1, Rd ← (X)	Нет	2
LD	Rd, Y	Косвенное считывание из памяти в регистр	Rd ← (Y)	Нет	2
LD	Rd, Y+	Косвенное считывание из памяти в регистр и инкр.	Rd ← (Y), Y ← Y + 1	Нет	2
LD	Rd, -Y	Предварительный декремент, а затем косвенное считывание из памяти в регистр	Y ← Y - 1, Rd ← (Y)	Нет	2
LDD	Rd, Y+q	Косвенное считывание из памяти в регистр со смещением	Rd ← (Y + q)	Нет	2
LD	Rd, Z	Косвенное считывание из памяти в регистр	Rd ← (Z)	Нет	2
LD	Rd, Z+	Косвенное считывание из памяти в регистр и инкр.	Rd ← (Z), Z ← Z + 1	Нет	2
LD	Rd, -Z	Предварительный декремент, а затем косвенное считывание из памяти в регистр	Z ← Z - 1, Rd ← (Z)	Нет	2
LDD	Rd, Z+q	Косвенное считывание из памяти в регистр со смещением	Rd ← (Z + q)	Нет	2
LDS	Rd, k	Непосредственное чтение из ОЗУ в регистр	Rd ← (k)	Нет	2
ST	X, Rr	Косвенная запись	(X) ← Rr	Нет	2
ST	X+, Rr	Косвенная запись и постинкремент	(X) ← Rr, X ← X + 1	Нет	2
ST	-X, Rr	Прединкремент и косвенная запись	X ← X - 1, (X) ← Rr	Нет	2
ST	Y, Rr	Косвенная запись	(Y) ← Rr	Нет	2
ST	Y+, Rr	Косвенная запись и постинкремент	(Y) ← Rr, Y ← Y + 1	Нет	2
ST	-Y, Rr	Прединкремент и косвенная запись	Y ← Y - 1, (Y) ← Rr	Нет	2
STD	Y+q, Rr	Косвенная запись со смещением	(Y + q) ← Rr	Нет	2
ST	Z, Rr	Косвенная запись	(Z) ← Rr	Нет	2
ST	Z+, Rr	Косвенная запись и постинкремент	(Z) ← Rr, Z ← Z + 1	Нет	2
ST	-Z, Rr	Прединкремент и косвенная запись	Z ← Z - 1, (Z) ← Rr	Нет	2
STD	Z+q, Rr	Косвенная запись со смещением	(Z + q) ← Rr	Нет	2

Продолжение таблицы А.3

Мнемокод	Операнды	Описание	Действие	Флаги	Количество машинных циклов
STS	k, Rr	Непосредственная запись в ОЗУ	(k) ← Rr	Нет	2
LPM		Чтение из памяти программ	R0 ← (Z)	Нет	3
LPM	Rd, Z	Чтение из памяти программ	Rd ← (Z)	Нет	3
LPM	Rd, Z+	Чтение из памяти программ и последующий инкремент	Rd ← (Z), Z ← Z + 1	Нет	3
ELPM		Расширенное чтение из памяти программ	R0 ← (RAMPZ:Z)	Нет	3
ELPM	Rd, Z	Расширенное чтение из памяти программ	Rd ← (RAMPZ:Z)	Нет	3
ELPM	Rd, Z+	Расширенное чтение из памяти программ и последующий инкремент	Rd ← (RAMPZ:Z), Z ← Z + 1	Нет	3
SPM		Запись в память программ	(Z) ← R1:R0	Нет	-
IN	Rd, P	Считывание из порта ввода-вывода в регистр	Rd ← I/O (P)	Нет	1
OUT	P, Rr	Запись из регистра в порт ввода-вывода	I/O (P) ← Rr	Нет	1
PUSH	Rr	Помещение содержимого регистра в стек	STACK ← Rr	Нет	2
POP	Rd	Извлечение из стека в регистр	Rd ← STACK	Нет	2

Таблица А.4 – Группа команд битовых операций и операций контроля битов

Мнемокод	Операнды	Описание	Действие	Флаги	Количество машинных циклов
SBI	P, b	Установка бита в регистре ввода-вывода	I/O(P,b) ← 1	Нет	2
CBI	P, b	Сброс бита в регистре ввода-вывода	I/O(P,b) ← 0	Нет	2
LSL	Rd	Логический сдвиг влево	Rd(n+1) ← Rd(n), Rd(0) ← 0, C ← Rd(7)	Z,C,N, V	1
LSR	Rd	Логический сдвиг вправо	Rd(n) ← Rd(n+1), Rd(7) ← 0, C ← Rd(0)	Z,C,N, V	1
ROL	Rd	Вращение влево через перенос	Rd(0) ← C, Rd(n+1) ← Rd(n), C ← Rd(7)	Z,C,N, V	1
ROR	Rd	Вращение вправо через перенос	Rd(7) ← C, Rd(n) ← Rd(n+1), C ← Rd(0)	Z,C,N, V	1
ASR	Rd	Арифметический сдвиг вправо	Rd(n) ← Rd(n+1), n=0..6	Z,C,N, V	1
SWAP	Rd	Обмен полубайтами	Rd(3..0) ↔ Rd(7..4), Rd(7..4) ← Rd(3..0)	Нет	1
BSET	s	Установка флага регистра SREG	SREG(s) ← 1	SREG (s)	1
BCLR	s	Сброс флага регистра SREG	SREG(s) ← 0	SREG (s)	1

Продолжение таблицы А.4

Мнемокод	Операнды	Описание	Действие	Флаги	Количество машинных циклов
BST	Rr, b	Запись бита регистра в T	T ← Rr(b)	T	1
BLD	Rd, b	Чтение из T в бит регистра	Rd(b) ← T	Нет	1
SEC		Установка переноса	C ← 1	C	1
CLC		Сброс переноса	C ← 0	C	1
SEN		Установка флага N	N ← 1	N	1
CLN		Сброс флага N	N ← 0	N	1
SEZ		Установка флага нуля Z	Z ← 1	Z	1
CLZ		Сброс флага нуля Z	Z ← 0	Z	1
SEI		Общее разрешение прерываний	I ← 1	I	1
CLI		Общий запрет прерываний	I ← 0	I	1
SES		Установка флага S	S ← 1	S	1
CLS		Сброс флага S	S ← 0	S	1
SEV		Установка флага V в регистре SREG	V ← 1	V	1
CLV		Сброс флага V в регистре SREG	V ← 0	V	1
SET		Установка флага T в регистре SREG	T ← 1	T	1
CLT		Сброс флага T в регистре SREG	T ← 0	T	1
SEH		Установка флага H в регистре SREG	H ← 1	H	1
CLH		Сброс флага H в регистре SREG	H ← 0	H	1

Таблица А.5 – Группа команд управления системой

Мнемокод	Операнды	Описание	Действие	Флаги	Количество машинных циклов
NOP		Нет операции		Нет	1
SLEEP		Переход в режим сна	(см. подробное описание режима сна)	Нет	1
WDR		Сброс сторожевого таймера	(см. подробное описание сторожевого таймера)	Нет	1
BREAK		Прерывание	Только для встроенной отладки	Нет	-

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⊕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 64 Кбайт
	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 ← Rd + Rr + C

	Синтаксис	Операнды:	Счетчик программ:
(i)	ADC Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16-разрядный код операции:

0001	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: 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#
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 ← Rd + Rr

	Синтаксис	Операнды:	Счетчик программ:
(i)	ADD Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	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: 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#
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) Rd+1:Rd ← Rd+1:Rd + K

	Синтаксис	Операнды:	Счетчик программ:
(i)	ADIW Rd+l:Rd,K	d ϵ {24,26,28,30}, 0 ≤ K ≤ 63	PC < PC + 1

16-разрядный код операции:

1001	0110	KKdd	KKKK

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	↔	↔	↔	↔	↔

S: N⊕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 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 r25 : 24, 1  ; Сложить 1 с r25:r24
adiw ZH : ZL, 63 ; Сложить 63 с Z указателем (r31 : r30)
```

Слов: 1 (2 байта)

Циклов: 2

Команда AND – выполнить логическое И

Описание:

Выполнение логического И между содержимым регистров Rd и Rr, и помещение результата в регистр назначения Rd.

Операция:

(i) Rd ← Rd·Rr

	Синтаксис	Операнды:	Счетчик программ:
(i)	AND Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← 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: 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:

```
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 И с непосредственным значением

Описание:

Выполнение логического И между содержимым регистра Rd и константой и помещение результата в регистр назначения Rd.

Операция:

(i) Rd ← Rd·K

	Синтаксис	Операнды:	Счетчик программ:
(i)	ANDI Rd,K	$16 \le d \le 31$, $0 \le K \le 255$	PC ← PC + 1

16-разрядный код операции:

0111	KKKK	dddd	KKKK

Булевы 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: 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 ниббл 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.

	Синтаксис	Операнды:	Счетчик программ:
(i)	ASR Rd	$0 \leq d \leq 31$,	PC ← PC + 1

16-разрядный код операции:

1001	010d	dddd	0101

Булевы 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: 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

Булевы 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 T флага регистра статуса (SREG) в бит регистра

Описание:
Копирование содержимого T флага регистра статуса в бит b регистра Rd

Операция:

(i) Rd(b) ← T

	Синтаксис	Операнды:	Счетчик программ:
(i)	BLD Rd,b	$0 \le d \le 31$, $0 \le b \le 7$	PC ← PC + 1

16-разрядный код операции:

1111	100d	dddd	0bbb

Булевы 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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRBC s, k	$0 \le s \le 7$, $-64 \le k \le +63$	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	ksss

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
        cpi 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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRBS s, k	$0 \le s \le 7$, $-64 \le k \le +63$	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	ksss

Булевы 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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRCC k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k000

Булевы 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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRCS 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:

```
        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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BREQ k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k001

Булевы 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, 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 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 (N⊕V = 0) then PC ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRGE k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k100

Булевы 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 (H) 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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRHC k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k101

Булевы 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)
```

Слов: 1 (2 байта)

Циклов: 1, если условия не соблюдены,
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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRHS k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k101

Булевы 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 << 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRID k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k111

Булевы 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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRIE k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k111

Булевы выражения регистра статуса (SREG)

I	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 < 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 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, меньше двоичного числа со знаком, представленного 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 (N⊕V = 1) then PC ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRLT k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k100

Булевы 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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRMI k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k010

Булевы 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 < назначение < PC+63). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

(i) If Rd ≠ Rr (Z = 0) then PC ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRNE k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k001

Булевы 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
        nop     ; 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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRPL k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k010

Булевы 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRSH k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k000

Булевы 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:   nop   ; 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 < назначение < PC+63). Параметр k является смещением относительно состояния счетчика программ и представлен в форме дополнения до двух.

Операция:

(i) If T = 0 then PC ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRTC k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← 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:  nop        ; 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 T 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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRTS k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	00kk	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
        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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRVC k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	01kk	kkkk	k011

Булевы 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 < 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 ← PC + k +1, else PC ← PC +1

	Синтаксис	Операнды:	Счетчик программ:
(i)	BRVS k	-64 ≤ k ≤+63	PC ← PC + k + 1 PC ← PC + 1, если условия не соблюдены

16-разрядный код операции:

1111	00kk	kkkk	k011

Булевы 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 $0 \le s \le 7$ PC ← PC + 1

16-разрядный код операции:

1001	0100	0sss	1000

Булевы 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 T
bset 7 ; Разрешить прерывание
```

Слов: 1 (2 байта)

Циклов: 1

Команда BST – переписать бит из регистра в флаг T регистра статуса (SREG)

Описание:
Перезапись бита b из регистра Rd в флаг T регистра статуса (SREG).

Операция:

(i) T ← Rd(b)

	Синтаксис	Операнды:	Счетчик программ:
(i)	BST Rd,b	$0 \le d \le 31, 0 \le b \le 7$	PC ← PC + 1

16-разрядный код операции:

1111	101d	dddd	0bbb

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 4регистра 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 ← k Приборы с 16-разрядным счетчиком программ, максимальный объем памяти программ 128K.

	Синтаксис	Операнды:	Счетчик программ:
(i)	CALL k	0 ≤ k ≤ 64K	PC ← k STACK ← PC + 2 SP ← SP-2, (2 байта, 16 битов)

16-разрядный код операции:

1001	010k	kkkk	111k
kkkk	kkkk	kkkk	kkkk

Булевы 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	$0 \le P \le 31, 0 \le b \le 7$	PC ← PC + 1

16-разрядный код операции:

1001	1000	PPPP	Pbbb

Булевы 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 ← Rd · ($FF - K)

	Синтаксис	Операнды:	Счетчик программ:
(i)	CBR Rd, K	$16 \le d \le 31, 0 \le K \le 255$	PC ← PC + 1

16-разрядный код операции:

Смотри команду ANDI с комплементом K.

Булевы 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: 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 ниббл 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

Булевы 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

Булевы 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 ← 0

	Синтаксис	Операнды:	Счетчик программ:
(i)	CLI	None	PC ← PC + 1

16-разрядный код операции:

1001	0100	1111	1000

Булевы 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 B
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 ← 0

	Синтаксис	Операнды:	Счетчик программ:
(i)	CLN	None	PC ← PC + 1

16-разрядный код операции:

1001	0100	1010	1000

Булевы 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 ← Rd ⊕ Rd

Синтаксис Операнды: Счетчик программ:

(i) CLR Rd $0 \le d \le 31$ PC ← PC + 1

16-разрядный код операции:

0010	01dd	dddd	dddd

Булевы 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

Булевы 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 T флаг

Описание:
Очистка флага пересылки (T) 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

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	0	-	-	-	-	-	-

T: 0
Очищен

Пример:

```
clt ; Очистить T флаг
```

Слов: 1 (2 байта)

Циклов: 1

Команда CLV – очистить флаг переполнения

Описание:
Очистка флага переполнения (V) в регистре статуса (SREG).

Операция:

(i) V ← 0

	Синтаксис	Операнды:	Счетчик программ:
(i)	CLV	None	PC ← PC + 1

16-разрядный код операции:

1001	0100	1011	1000

Булевы 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 ← 0

Синтаксис	Операнды:	Счетчик программ:
(i) CLZ	None	PC ← PC + 1

16-разрядный код операции:

1001	0100	1001	1000

Булевы 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 COM – выполнить дополнение до единицы

Описание:

Команда выполняет дополнение до единицы (реализует обратный код) содержимого регистра Rd.

Операция:

(i) Rd ← $FF - Rd

Синтаксис Операнды: Счетчик программ:

(i) COM Rd $0 \le d \le 31$ PC ← PC + 1

16-разрядный код операции:

1001	010d	dddd	0000

Булевы 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 CP – 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 $0 \le d \le 31$, $0 \le r \le 31$ PC ← PC + 1

16-разрядный код операции:

0001	01rd	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 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#
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:

```
        cp r4, r19 ; Сравнить r4 с r19
        brne noteq ; Перейти, если r4 <> r19
                   . . .
noteq:  nop        ; 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 $\begin{matrix}0 \le d \le 31,\\ 0 \le r \le 31\end{matrix}$ PC ← PC + 1

16-разрядный код операции:

0000	01rd	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 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
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:  nop      ; 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 $\begin{matrix}16 \le d \le 31,\\ 0 \le K \le 255\end{matrix}$ PC ← PC + 1

16-разрядный код операции:

0011	KKKK	dddd	KKKK

Булевы 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:  nop        ; 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 ← PC + 2 (or 3), else PC ← 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

Булевы 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, если условие ложное
2, если условие истинное и пропускаемая инструкция занимает 1 слово
3, если условие истинное и пропускаемая инструкция занимает 2 слова

Команда DEC – декрементировать

Описание:

Вычитание единицы 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 работе с беззнаковыми числами для выполнения перехода в соответствии с результатом команды могут использоваться только команды BREQ и BRNE. При обработке значений в форме дополнения до двух допустимы все учитывающие знак переходы.

Операция:

(i) Rd← Rd - 1

Синтаксис Операнды: Счетчик программ:

(i) DEC Rd $0 \le d \le 31$ PC ← PC + 1

16-разрядный код операции:

1001	010d	dddd	1010

Булевы 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 ИЛИ

Описание:

Выполнение логического исключающего ИЛИ между содержимым регистра Rd и регистром Rr и помещение результата в регистр назначения Rd.

Операция:

(i) Rd← Rd⊕Rr

Синтаксис Операнды: Счетчик программ:

(i) EOR Rd,Rr $0 \le d \le 31$, $0 \le r \le 31$ PC ← PC + 1

16-разрядный код операции:

0010	01rd	dddd	rrrr

Булевы 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: 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 вызвать 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

	Синтаксис	Операнды:	Счетчик программ:	Стек
(i)	ICALL	None	См. Операция	STACK← PC + 1 SP← SP-2, (2 байта, 16 битов)

16-разрядный код операции:

1001	0101	0000	1001

Булевы 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 128К.

	Синтаксис	Операнды:	Счетчик программ:	Стек
(i)	IJMP	None	См. Операция	Не задействуется

16-разрядный код операции:

1001	0100	0000	1001

Булевы 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 регистрового файла.

Операция:

(i) Pd← I/O(P)

Синтаксис	Операнды:	Счетчик программ:
(i) IN Rd,P	$0 \le d \le 31, 0 \le P \le 63$	PC← PC + 1

16-разрядный код операции:

1011	0PPd	dddd	PPPP

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
        in r25, $16 ; Считать Порт B
        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← Rd + 1

Синтаксис Операнды: Счетчик программ:

(i) INC Rd $0 \le d \le 31$ PC ← PC + 1

16-разрядный код операции:

1001	010d	dddd	0011

Булевы 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 (4M слов) памяти программ. См. также команду RJMP.

Операция:

(i) PC← 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

Булевы 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: 3

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 64 Кбайта. Для обращения к другой странице СОЗУ необходимо изменить регистр RAMPX в I/O области. Регистр-указатель X может остаться неизменным после выполнения команды, но может быть инкрементирован или декрементирован. Использование регистра-указателя X обеспечивает удобную возможность обращения к матрицам, таблицам, указателю стека.

```
LD r26, X+
LD r27, X+
LD r26, -X
LD r27, -X
```

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	Операнды:	Счетчик программ:
(i)	LD Rd,X	0 ≤ d ≤ 31	PC← PC + 1
(ii)	LD Rd,X+	0 ≤ d ≤ 31	PC← PC + 1
(iii)	LD Rd,-X	0 ≤ d ≤ 31	PC← PC + 1

16-разрядный код операции:

(i)	1001	000d	dddd	1100
(ii)	1001	000d	dddd	1101
(iii)	1001	000d	dddd	1110

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
    clr   r27     ;Очистить старший   X
    ldi   r26, $20 ;Установить $20 в младший байт X
    ld    r0, X+   ;Загрузить в r0 содержимое SRAM по адресу $20 (X постинкрементируется)
    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 преддекрементируется)
```

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 64 Кбайта. Для обращения к другой странице СОЗУ необходимо изменить регистр RAMPY в I/O области. Регистр-указатель Y может остаться неизменным после выполнения команды, но может быть инкрементирован или декрементирован. Использование регистра-указателя Y обеспечивает удобную возможность обращения к матрицам, таблицам, указателю стека.

```
LD r28, Y+
LD r29, Y+
LD r28, -Y
LD r29, -Y
```

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	$0 \le d \le 31$	PC← PC + 1
(ii)	LD Rd,Y+	$0 \le d \le 31$	PC← PC + 1
(iii)	LD Rd,-Y	$0 \le d \le 31$	PC← PC + 1
(iv)	LDD Rd, Y + q	$0 \le d \le 31$ $0 \le q \le 63$	PC← PC + 1

16-разрядный код операции:

(i)	1000	000d	dddd	1000
(ii)	1001	000d	dddd	1001
(iii)	1001	000d	dddd	1010
(iiii)	10q0	qq0d	dddd	1qqq

Булевы 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 64 Кбайта. 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.

```
LD r30, Z+
LD r31, Z+
LD r30, -Z
LD r31, -Z
```

Использование Z-указателя:

	Операция:	Комментарий:	
(i)	Rd ← (Z)		Z: Неизменен
(ii)	Rd ← (Z)	Z ← Z + 1	Z: Инкрементирован впоследствии
(iii)	Z ← Z - 1	Rd ← (Z)	Z: Предварительно декрементирован
(iv)	Rd ← (Z + q)		Z: Неизменен, q: смещение

	Синтаксис	Операнды:	Счетчик программ:
(i)	LD Rd, Z	$0 \le d \le 31$	PC← PC + 1
(ii)	LD Rd, Z+	$0 \le d \le 31$	PC← PC + 1
(iii)	LD Rd, -Z	$0 \le d \le 31$	PC← PC + 1
(iv)	LDD Rd, Z + q	$0 \le d \le 31$ $0 \le q \le 63$	PC← PC + 1

16-разрядный код операции:

(i)	1000	000d	dddd	0000
(ii)	1001	000d	dddd	0001

(iii)	1001	000d	dddd	0010
(iiii)	10q0	qq0d	dddd	0qqq

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
clr  r29      ;Очистить старший байт Z
ldi  r28, $20  ;Установить $20 в младший байт Z
ld   r0, Z+    ;Загрузить в r0 содерж. SRAM по адресу $20 (Z постинкрементируется)
ld   r1, Z     ;Загрузить в r1 содержимое SRAM по адресу $21
ldi  r28, $23  ;Установить $23 в младший байт Y
ld   r2, Z     ;Загрузить в r2 содержимое SRAM по адресу $23
ld   r3, -Z    ;Загрузить в r3 содерж. SRAM по адресу $22 (Z преддекрементируется)
ldd  r4, Z+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	$16 \le d \le 31$, $0 \le K \le 255$	PC← PC + 1

16-разрядный код операции:

1110	KKKK	dddd	KKKK

Булевы 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 64 Кбайта. Команда LDS использует для обращения к памяти выше 64 Кбайт регистр RAMPZ.

Операция:

(i) Rd ← (k)

	Синтаксис	Операнды:	Счетчик программ:
(i)	LDS Rd,k	$0 \le d \le 31, 0 \le k \le 65535$	PC← PC + 2

32-разрядный код операции:

1001	000d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Булевы 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: 2

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 64 Кбайта (32К слов) памяти программ.

```
LPM r30, Z+
LPM r31, Z+
```

	Операция:		Комментарий:
(i)	R0← (Z)	Z ← Z + 1	Z: не изменяется, R0 – регистр назначения
(ii)	Rd← (Z)		Z: не изменяется
(iii)	Rd← (Z)		Z: увеличение на 1 после выполнения инструкции

	Синтаксис	Операнды:	Счетчик программ:
(i)	LPM	None	PC← PC + 1
(ii)	LPM Rd, Z	0 ≤ d ≤ 31	PC← PC + 1
(iii)	LPM Rd, Z+	0 ≤ d ≤ 31	PC← PC + 1

16-разрядный код операции:

(i)	1001	0101	1100	1000
(ii)	1001	000d	dddd	0100
(iii)	1001	000d	dddd	0101

Булевы 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:

	Синтаксис	Операнды:	Счетчик программ:
(i)	LSL Rd	$0 \le d \le 31$	PC← PC + 1

16-разрядный код операции:

0000	11dd	dddd	dddd

Булевы 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.

Операция:

	Синтаксис	Операнды:	Счетчик программ:
(i)	LSR Rd	$0 \le d \le 31$	PC← PC + 1

16-разрядный код операции:

1001	010d	dddd	0110

Булевы 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.

Операция:

(i) Rd ← Rr

	Синтаксис	Операнды:	Счетчик программ:
(i)	MOV Rd, Rr	$0 \le d \le 31, 0 \le r \le 31$	PC← PC + 1

16-разрядный код операции:

0010	11rd	dddd	rrrr

Булевы 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 ← Rd x Rr

	Синтаксис	Операнды:	Счетчик программ:
(i)	MUL Rd, Rr	$0 \le d \le 31, 0 \le r \le 31$	PC ← PC + 1

16-разрядный код операции:

1001	11rd	dddd	rrrr

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	↔	↔

C: R15
Устанавливается, если установлен бит 15 результата, в ином случае очищается

Z: R15#·R14#·R13#·R12#·R11#·R10#·R9#·R8#·R7#·R6#·R5#·R4#·R3#·R2#·R1#·R0#
Устанавливается, если результат равен $0000, 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 ← $00 - Rd

	Синтаксис	Операнды:	Счетчик программ:
(i)	NEG Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-разрядный код операции:

1001	010d	dddd	0001

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	↔	↔	↔	↔	↔	↔

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: R7#·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 $00

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← PC+ 1

16-разрядный код операции:

0000	0000	0000	0000

Булевы 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 B
nop           ; Ожидать (пустая операция)
out  $18, r17 ; Записать 1 в Порт B
```

Слов: 1 (2 байта)

Циклов: 1

Команда OR – выполнить логическое ИЛИ

Описание:

Команда выполняет логическое ИЛИ содержимого регистров Rd и Rr и размещает результат в регистре назначения Rd.

Операция:

(i) Rd ← Rd v Rr

	Синтаксис	Операнды:	Счетчик программ:
(i)	OR Rd,Rr	$0 \le d \le 31, 0 \le r \le 31$	PC ← PC + 1

16-разрядный код операции:

0010	10rd	dddd	rrrr

Булевы 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: R7#·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 ИЛИ с непосредственным значением

Описание:

Команда выполняет логическое ИЛИ между содержимым регистра Rd и константой и размещает результат в регистре назначения Rd.

Операция:

(i) Rd ← Rd v K

	Синтаксис	Операнды:	Счетчик программ:
(i)	ORI Rd,K	$16 \le d \le 31, 0 \le K \le 255$	PC ← PC + 1

16-разрядный код операции:

0110	KKKK	dddd	KKKK

Булевы 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: R7#·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 ниббл 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) I/O(P) ← Rr

	Синтаксис	Операнды:	Счетчик программ:
(i)	OUT P, Rr	$0 \le r \le 31$, $0 \le P \le 63$	PC← PC + 1

16-разрядный код операции:

1011	1PPr	rrrr	PPPP

Булевы 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 B
nop          ; Ожидать (пустая операция)
out $18, r17 ; Записать единицы в Порт B
```

Слов: 1 (2 байта)

Циклов: 1

Команда POP – записать регистр из стека

Описание:
Команда загружает регистр Rd байтом содержимого стека.

Операция:
(i) Rd ← STACK

	Синтаксис	Операнды:	Счетчик программ:
(i)	POP Rd	0 ≤ d ≤31	PC← PC + 1 SP← SP + 1

16-разрядный код операции:

1001	000d	dddd	1111

Булевы 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 PUSH – поместить регистр в стек

Описание:
Команда помещает содержимое регистра Rr в стек.

Операция:
(i) STACK ← Rr

	Синтаксис	Операнды:	Счетчик программ:
(i)	PUSH Rr	$0 \leq r \leq 31$	PC← PC + 1 SP← SP - 1

16-разрядный код операции:

1001	001r	rrrr	1111

Булевы 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 – относительный вызов подпрограммы

Описание:

Команда вызывает подпрограмму в пределах 2К слов (4К 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

Булевы 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 128K байт памяти программ

(ii) PC(21-0) ← STACK Приборы с 22-разрядным счетчиком команд, максимум 8M 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	0000	1000

Булевы 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	0001	1000

Булевы 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 – относительный переход

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 ← PC + k + 1

	Синтаксис	Операнды:	Счетчик программ:	Стек
(i)	RJMP k	-2K ≤ k < 2K	PC ← PC + k + 1	Стек не меняется

16-разрядный код операции:

1100	kkkk	kkkk	kkkk

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

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 rjmp (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:

	Синтаксис	Операнды:	Счетчик программ:
(i)	ROL Rd	$0 \le d \le 31$	PC ← PC + 1

16-разрядный код операции:

0001	11dd	dddd	dddd

Булевы 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: **R**7#·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:

```
        lsl   r18    ; Умножение регистров r19, r18 на два
        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:

	Синтаксис	Операнды:	Счетчик программ:
(i)	ROR Rd	$0 \le d \le 31$	PC ← PC + 1

16-разрядный код операции:

1001	010d	dddd	0111

Булевы 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: 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:

```
    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 ← Rd - Rr - C

	Синтаксис	Операнды:	Счетчик программ:
(i)	SBC Rd, Rr	$0 \le d \le 31, 0 \le r \le 31$	PC ← PC + 1

16-разрядный код операции:

0000	10rd	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 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 флага переноса (C) из содержимого регистра, размещение результата в регистре назначения Rd.

Операция:

(i) Rd ← Rd - K - C

	Синтаксис	Операнды:	Счетчик программ:
(i)	SBCI Rd, K	$16 \le d \le 31, 0 \le K \le 255$	PC ← PC + 1

16-разрядный код операции:

0100	KKKK	dddd	KKKK

Булевы 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#·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	$0 \le P \le 31, 0 \le b \le 7$	PC ← PC + 1

16-разрядный код операции:

1001	1010	PPPP	Pbbb

Булевы 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 ← PC + 2 (or 3) else PC ← PC + 1

	Синтаксис	Операнды:	Счетчик программ:
(i)	SBIC P,b	$0 \le P \le 31, 0 \le b \le 7$	PC ← PC + 1, если условия не соблюдены, нет пропуска PC ← PC + 2, если следующая команда длиной в 1слово PC ← PC + 3, если следующая команда длиной в 2 слова

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
         nop          ; Продолжать (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1, если условие не выполняется (нет пропуска)
2, если условие выполняется, и длина пропускаемой инструкции составляет 1 слово
3, если условие выполняется, и длина пропускаемой инструкции составляет 2 слова

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 ← PC + 2 (or 3) else PC ← PC + 1

Синтаксис	Операнды:	Счетчик программ:
(i) SBIS P, b	$0 \le P \le 31, 0 \le b \le 7$	PC ← PC + 1, если условия не соблюдены, нет пропуска PC ← PC + 2, если следующая команда длиной в 1слово PC ← PC + 3, если следующая команда длиной в 2слова

16-разрядный код операции:

1001	1011	PPPP	Pbbb

Булевы 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 условие не выполняется (нет пропуска)

2, если условие выполняется, и длина пропускаемой инструкции составляет 1 слово

3, если условие выполняется, и длина пропускаемой инструкции составляет 2 слова

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) Rd + 1:Rd ← Rd +1:Rd - K

	Синтаксис	Операнды:	Счетчик программ:
(i)	SBIW Rd +1: Rd,K	d ∈ {24,26,28,30}, 0 ≤ K ≤63	PC ← PC + 1

16-разрядный код операции:

1001	0111	KKdd	KKKK

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	↔	↔	↔	↔	↔

S: N⊕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  r25 : r24, 1  ; Вычесть 1 из r25:r24
sbiw  YH: YL, 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 ← Rd v K

	Синтаксис	Операнды:	Счетчик программ:
(i)	SBR Rd, K	$16 \le d \le 31, 0 \le K \le 255$	PC ← PC + 1

16-разрядный код операции:

0110	KKKK	dddd	KKKK

Булевы 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: 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, 3 ; 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	$0 \le r \le 31, 0 \le b \le 7$	PC ← PC + 1, если условия не соблюдены, нет пропуска PC ← PC + 2, если следующая команда длиной в 1 слово PC ← PC + 3, если следующая команда длиной в 2 слова

16-разрядный код операции:

1111	110r	rrrr	0bbb

Булевы 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
nop        ; Продолжать (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1, если условие не выполняется (нет пропуска)
2, если условие выполняется, и длина пропускаемой инструкции составляет 1 слово
3, если условие выполняется, и длина пропускаемой инструкции составляет 2 слова

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	$0 \le r \le 31, 0 \le b \le 7$	PC ← PC + 1, если условия не соблюдены, нет пропуска PC ← PC + 2, если следующая команда длиной в 1 слово PC ← PC + 3, если следующая команда длиной в 2слова

16-разрядный код операции:

1111	111r	rrrr	0bbb

Булевы 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 условие не выполняется (нет пропуска)
2, если условие выполняется, и длина пропускаемой инструкции составляет 1 слово
3, если условие выполняется, и длина пропускаемой инструкции составляет 2 слова

Команда SEC –- установить флаг переноса

Описание:
Команда устанавливает флаг переноса (C) в регистре статуса (SREG).

Операция:
(i) C ← 1

	Синтаксис	Операнды:	Счетчик программ:
(i)	SEC	None	PC ← PC + 1

16-разрядный код операции:

1001	0100	0000	1000

Булевы 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

Булевы 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 ← 1

	Синтаксис	Операнды:	Счетчик программ:
(i)	SEI	None	PC ← PC + 1

16-разрядный код операции:

1001	0100	0111	1000

Булевы 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 B
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

Булевы 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 ← $FF

	Синтаксис	Операнды:	Счетчик программ:
(i)	SER Rd	16 ≤ d ≤ 31	PC ← PC + 1

16-разрядный код операции:

1110	1111	dddd	1111

Булевы 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 B
nop           ; Задержка (пустая операция)
out  #18, r17 ; Записать единицы в Порт B
```

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

Булевы 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 ← 1

	Синтаксис	Операнды:	Счетчик программ:
(i)	SET	None	PC ← PC + 1

16-разрядный код операции:

1001	0100	0110	1000

Булевы 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 ← 1

	Синтаксис	Операнды:	Счетчик программ:
(i)	SEV	None	PC ← PC + 1

16-разрядный код операции:

1001	0100	0011	1000

Булевы 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 ← 1

	Синтаксис	Операнды:	Счетчик программ:
(i)	SEZ	None	PC ← PC + 1

16-разрядный код операции:

1001	0100	0001	1000

Булевы 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	1000	1000

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
  mov r0, r11 ; Копировать r11 в r0
…ldi r16, (1<<SE) ; Разрешение спящего режима
…out MCUCR, r16
  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 64 Кбайта. Для обращения к другой странице СОЗУ необходимо изменить регистр RAMPX в I/O области. Регистр-указатель X может остаться неизменным после выполнения команды, но может быть инкрементирован или декрементирован. Эта особенность очень удобна при использовании регистра-указателя X в качестве указателя стека.

```
ST X+, r26
ST X+, r27
ST -X, r26
ST -X, r27
```

Использование X-указателя:

	Операция:		Комментарий:
(i)	(X) ← Rr		X: Неизменен
(ii)	(X) ← Rr	X ← X + 1	X: Инкрементирован впоследствии
(iii)	X ← X - 1	(X) ← Rr	X: Предварительно декрементирован

	Синтаксис	Операнды:	Счетчик программ:
(i)	ST X, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST X+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -X, Rr	$0 \le r \le 31$	PC ← PC + 1

16-разрядный код операции:

(i)	1001	001r	rrrr	1100
(ii)	1001	001r	rrrr	1101
(iii)	1001	001r	rrrr	1110

Булевы 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    X, r2,    ; Сохранить в r2 содержимое SRAM по адресу $23
  st    -X, r3,   ; 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 64 Кбайта. Для обращения к другой странице СОЗУ необходимо изменить регистр RAMPY в I/O области. Регистр-указатель Y может остаться неизменным после выполнения команды, но может быть инкрементирован или декрементирован. Эта особенность очень удобна при использовании регистра-указателя Y в качестве указателя стека.

```
ST Y+, r28
ST Y+, r29
ST -Y, r28
ST -Y, r29
```

Использование Y-указателя:

	Операция:		Комментарий:
(i)	(Y) ← Rr		Y: Неизменен
(ii)	(Y) ← Rr	Y ← Y + 1	Y: Инкрементирован впоследствии
(iii)	Y ← Y - 1	(Y) ← Rr	Y: Предварительно декрементирован
(iv)	(Y + q) ← Rr		Y: Неизменен, q: смещение

	Синтаксис	Операнды:	Счетчик программ:
(i)	ST Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Y+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(iiii)	STD Y+q, Rr	$0 \le r \le 31$, $0 \le q \le 63$	PC ← PC + 1

16-разрядный код операции:

(i)	1000	001r	rrrr	1000
(ii)	1001	001r	rrrr	1001
(iii)	1001	001r	rrrr	1010
(iiii)	10q0	qq1r	rrrr	1qqq

Булевы 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 постинкрементируется)
        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 преддекрементируется)
        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 64 Кбайта. 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.

```
ST Z+, r30
ST Z+, r31
ST -Z, r30
ST -Z, r31
```

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	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Z+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(iv)	STD Z+q, Rr	$0 \le r \le 31$, $0 \le q \le 63$	PC ← PC + 1

16-разрядный код операции:

(i)	1000	001r	rrrr	0000
(ii)	1001	001r	rrrr	0001

(iii)	1001	001r	rrrr	0010
(iiii)	10q0	qq1r	rrrr	0qqq

Булевы 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 адр. $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 адр. $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 64 Кбайт регистр RAMPZ.

Операция:
(i) (k) ← Rr

	Синтаксис	Операнды:	Счетчик программ:
(i)	STS k, Rr	$0 \le r \le 31, 0 \le k \le 65535$	PC ← PC + 2

32-разрядный код операции:

1001	001d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Булевы 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: 2

Команда SUB – вычесть без переноса

Описание:
Вычитание содержимого регистра-источника Rr из содержимого регистра Rd, размещение результата в регистре назначения Rd.

Операция:
(i) Rd ← Rd - Rr

	Синтаксис	Операнды:	Счетчик программ:
(i)	SUB Rd, Rr	$0 \le d \le 31, 0 \le r \le 31$	PC ← PC + 1

16-разрядный код операции:

0001	10rd	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 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#
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 ← Rd - K

	Синтаксис	Операнды:	Счетчик программ:
(i)	SUBI Rd, K	$16 \le d \le 31, 0 \le K \le 255$	PC ← PC + 1

16-разрядный код операции:

0101	KKKK	dddd	KKKK

Булевы 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 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 нибблы (полубайты) регистра.

Операция:
(i) R(7-4) ← Rd(3-0), R(3-0) ← Rd(7-4)

	Синтаксис	Операнды:	Счетчик программ:
(i)	SWAP Rd	$0 \le d \le 31$	PC ← PC + 1

16-разрядный код операции:

1001	010d	dddd	0010

Булевы 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 нибблы r1
inc  r1  ; Увеличить на 1 старший ниббл r1
swap r1  ; Снова поменять местами нибблы 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 ← Rd · Rd

	Синтаксис	Операнды:	Счетчик программ:
(i)	TST Rd	$0 \le d \le 31$	PC ← PC + 1

16-разрядный код операции:

0010	00dd	dddd	dddd

Булевы 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: 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	1010	1000

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
wdr ; Сбросить сторожевой таймер
```

Слов: 1 (2 байта)

Циклов: 1

Приложение Б
(справочное)

Режим совместимости с МК ATmega103

При программировании конфигурационного бита M103C микроконтроллер 1887ВЕ7Т становится совместим с МК ATmega103 относительно ОЗУ, выводов МК и векторов прерываний. Однако, некоторые особенности микроконтроллера недоступны в режиме совместимости. Ниже приведен перечень этих особенностей.

- Доступен только один USART вместо двух и только в асинхронном режиме. Доступны только восемь младших значащих бит регистра генератора частоты.
- Доступен один 16-разрядный таймер/счетчик с двумя регистрами сравнения вместо двух 16-разрядных таймеров/счетчиков с тремя регистрами сравнения.
- Двухпроводной последовательный интерфейс не поддерживается.
- Порт C имеет функцию только выхода.
- Порт G выполняет только альтернативную функцию.
- Порт F работает только как цифровой вход в дополнение к аналоговому входу АЦП.
- Область загрузчика не поддерживается.
- Отсутствует возможность настройки частоты внутреннего калиброванного RC-генератора.
- При использовании интерфейса внешней памяти количество выводов адресной шины фиксировано и не может освобождаться под функцию стандартного ввода-вывода.
- Возможность настройки разных секций адреса внешней памяти на работу только с одним значением состояния ожидания.
- В регистре MCUCSR доступны только флаги EXTRF и PORF.
- Для изменения тайм-аута сторожевого таймера не требуется временная последовательность.
- Выводы внешних прерываний 3-0 обслуживают только прерывание по уровню.
- USART не имеет FIFO буфера, поэтому переполнение наступает раньше.

Приложение В
(справочное)

Карта регистров микроконтроллера

Т а б л и ц а В.1 – Карта регистров микроконтроллера

Адрес	Имя	Бит7	Бит6	Бит5	Бит4	Бит3	Бит2	Бит1	Бит0
($FF)	–	–	–	–	–	–	–	–	–
..	–	–	–	–	–	–	–	–	–
($9E)	–	–	–	–	–	–	–	–	–
($9D)	UCSR1C	–	UMSEL1	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1
($9C)	UDR1	USART1 регистр ввода/вывода данных							
($9B)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1
($9A)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81
($99)	UBRR1L	USART1 – младший регистр скорости передачи							
($98)	UBRR1H	–	–	–	–	USART1 – старший регистр скорости передачи			
($97)	–	–	–	–	–	–	–	–	–
($96)	–	–	–	–	–	–	–	–	–
($95)	UCSR0C	–	UMSEL0	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0
($94)	–	–	–	–	–	–	–	–	–
($93)	–	–	–	–	–	–	–	–	–
($92)	–	–	–	–	–	–	–	–	–
($91)	–	–	–	–	–	–	–	–	–
($90)	UBRR0H	–	–	–	–	USART0 – старший регистр скорости передачи			
($8F)	–	–	–	–	–	–	–	–	–
($8E)	–	–	–	–	–	–	–	–	–
($8D)	–	–	–	–	–	–	–	–	–
($8C)	TCCR3C	FOC3A	FOC3B	FOC3C	–	–	–	–	–
($8B)	TCCR3A	COM3A1	COM3A0	COM3B1	COM3B0	COM3C1	COM3C0	WGM31	WGM30
($8A)	TCCR3B	ICNC3	ICES3	–	WGM33	WGM32	CS32	CS31	CS30
($89)	TCNT3H	Старший байт регистра счётчика							
($88)	TCNT3L	Младший байт регистра счётчика							
($87)	OCR3AH	Старший байт регистра сравнения							
($86)	OCR3AL	Младший байт регистра сравнения							
($85)	OCR3BH	Старший байт регистра сравнения							
($84)	OCR3BL	Младший байт регистра сравнения							
($83)	OCR3CH	Старший байт регистра сравнения							
($82)	OCR3CL	Младший байт регистра сравнения							
($81)	ICR3H	Старший байт регистра захвата							
($80)	ICR3L	Младший байт регистра захвата							
($7F)	–	–	–	–	–	–	–	–	–
($7E)	–	–	–	–	–	–	–	–	–
($7D)	ETIMSK	–	–	TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	OCIE1C
($7C)	ETIFR	–	–	ICF3	OCF3A	OCF3B	TOV3	OCF3C	OCF1C
($7B)	–	–	–	–	–	–	–	–	–
($7A)	TCCR1C	FOC1A	FOC1B	FOC1C	–	–	–	–	–
($79)	OCR1CH	Старший байт регистра сравнения							
($78)	OCR1CL	Младший байт регистра сравнения							
($77)	–	–	–	–	–	–	–	–	–
($76)	–	–	–	–	–	–	–	–	–
($75)	–	–	–	–	–	–	–	–	–
($74)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
($73)	TWDR	Регистр данных двухпроводного последовательного интерфейса							
($72)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
($71)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0
($70)	TWBR	Регистр скорости передачи двухпроводного последовательного интерфейса							
($6F)	OSCCAL	Регистр калибровки генератора							
($6E)	–	–	–	–	–	–	–	–	–
($6D)	XMCRA	–	SRL2	SRL1	SRL0	SRW01	SRW00	SRW11	
($6C)	XMCRB	XMBK	–	–	–	–	XMM2	XMM1	XMM0
($6B)	–	–	–	–	–	–	–	–	–
($6A)	EICRA	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00
($69)	–	–	–	–	–	–	–	–	–
($68)	SPMCSR	SPMIE	RWWSP	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN

Продолжение таблицы В.1

Адрес	Имя	Бит7	Бит6	Бит5	Бит4	Бит3	Бит2	Бит1	Бит0
($67)	–	–	–	–	–	–	–	–	–
($66)	–	–	–	–	–	–	–	–	–
($65)	PORTG	–	–	–	PORTG4	PORTG3	PORTG2	PORTG1	PORTG0
($64)	DDRG	–	–	–	DDG4	DDG3	DDG2	DDG1	DDG0
($63)	PING	–	–	–	PING4	PING3	PING2	PING1	PING0
($62)	PORTF	PORTF7	PORTF6	PORTF5	PORTF4	PORTF3	PORTF2	PORTF1	PORTF0
($61)	DDRF	DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
($60)	–	–	–	–	–	–	–	–	–
$3F($5F)	SREG	I	T	H	S	V	N	Z	C
$3E($5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8
$3D($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
$3C($5C)	XDIV	XDIVEN	XDIV6	XDIV5	XDIV4	XDIV3	XDIV2	XDIV1	XDIV0
$3B($5B)	RAMPZ	–	–	–	–	–	–	–	RAMPZ0
$3A($5A)	EICRB	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40
$39($59)	EIMSK	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
$38($58)	EIFR	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0
$37($57)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
$36($56)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
$35($55)	MCUCR	SRE	SRW10	SE	SM1	SM0	SM2	IVSEL	IVCE
$34($54)	MCUCSR	JTD	–	–	JTRF	WDRF	BORF	EXTRF	PORF
$33($53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
$32($52)	TCNT0	Регистр таймера/счетчика							
$31($51)	OCR0	Регистр сравнения таймера/счетчика							
$30($50)	ASSR	–	–	–	–	AS0	TCN0UB	OCR0UB	TCR0UB
$2F($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10
$2E($4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
$2D($4D)	TCNT1H	Старший байт регистра счётчика							
$2C($4C)	TCNT1L	Младший байт регистра счётчика							
$2B($4B)	OCR1AH	Старший байт регистра сравнения							
$2A($4A)	OCR1AL	Младший байт регистра сравнения							
$29($49)	OCR1BH	Старший байт регистра сравнения							
$28($48)	OCR1BL	Младший байт регистра сравнения							
$27($47)	ICR1H	Старший байт регистра захвата							
$26($46)	ICR1L	Младший байт регистра захвата							
$25($45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20
$24($44)	TCNT2	Регистр таймера/счетчика							
$23($43)	OCR2	Регистр сравнения таймера/счетчика							
$22($42)	OCDR	IDRD/ OCDR7	OCDR6	OCDR5	OCDR4	OCDR3	OCDR2	OCDR1	OCDR0
$21($41)	WDTCR	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0
$20($40)	SFIOR	TSM	–	–	–	ACME	PUD	PSR0	PSR321
$1F($3F)	EEARH	–	–	–	–	Старшие биты регистра адреса			
$1E($3E)	EEARL	Младший байт регистра адреса							
$1D($3D)	EEDR	Регистр данных							
$1C($3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE
$1B($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
$1A($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
$19($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
$18($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
$17($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
$16($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
$15($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
$14($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
$13($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
$12($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
$11($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
$10($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
$0F($2F)	SPDR	Регистр данных							
$0E($2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SP12X
$0D($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
$0C($2C)	UDR0	Регистр данных							
$0B($2B)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0
$0A($2A)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80

Продолжение таблицы В.1

Адрес	Имя	Бит7	Бит6	Бит5	Бит4	Бит3	Бит2	Бит1	Бит0
$09($29)	UBRR0L	Младший байт регистра скорости передачи							
$08($28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
$07($27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
$06($26)	ADCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
$05($25)	ADCH	Старший байт регистра данных							
$04($24)	ADCL	Младший байт регистра данных							
$03($23)	PORTE	PORTE7	PORTE6	PORTE5	PORTE4	PORTE3	PORTE2	PORTE1	PORTE0
$02($22)	DDRE	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
$01($21)	PINE	PINE7	PINE6	PINE5	PINE4	PINE3	PINE2	PINE1	PINE0
$00($20)	PINF	PINF7	PINF6	PINF5	PINF4	PINF3	PINF2	PINF1	PINF0

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Подп.	Дата
	изме-ненных	замененных	новых	анну-лиро-ванных				
9	-	все	-	-	455			10.12.19