МИКРОСХЕМА ИНТЕГРАЛЬНАЯ
1887ВЕ4У
Руководство пользователя

Содержание

Введение

В настоящем руководстве пользователя приведено описание архитектуры, функционального построения, технических характеристик, системы команд и особенностей применения ИС 1887ВЕ4У, которая представляет собой 8-разрядную микро-ЭВМ с AVR RISC архитектурой, тактовой частотой 8 МГц, содержащий 8 Кбайт Flash ЗУ программ, ЭСППЗУ (1К×8) бит, ОЗУ (512×8) бит, 8-канальный 10-разрядный АЦП, последовательный периферийный интерфейс SPI, двухпроводной последовательный интерфейс TWI, последовательный порт UART. Разработанные микросхемы будут служить основой для создания перспективных систем управления.

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

1 Назначение и область применения

В области промышленного производства микросхема 1887ВЕ4У может быть использована для управления робототехническими комплексами, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, оргтехнике, вычислительной технике, телекоммуникационной технике и т. п. Особенно перспективно применение микросхем 1887ВЕ4У в портативной носимой аппаратуре и приборах, имеющих жесткие ограничения по соотношению быстродействие/потребляемая мощность/стоимость.

2 Краткое техническое описание ИС 1887ВЕ4У

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

AVR ядро сочетает расширенный набор команд с 32 рабочими регистрами общего назначения. Все 32 регистра соединены с АЛУ, что обеспечивает доступ к двум независимым регистрам на время исполнения команды за один машинный такт. Благодаря выбранной архитектуре достигнута высокая производительность, в 10 раз превосходящая скорость соответствующего CISC микроконтроллера.

Микроконтроллер содержит 8 Кбайт внутрисистемной программируемой флэш-памяти (Flash) программ с возможностью чтения в процессе записи, 1 Кбайт ЭСППЗУ, 512 байтов СОЗУ, 32 входа/выхода общего назначения, 32 рабочих регистра, три гибких таймера/счётчика с режимом сравнения, внешние и внутренние прерывания, последовательный программируемый UART, 8-канальный 10-битный АЦП, программируемый сторожевой таймер с внутренним генератором, последовательный SPI порт и шесть выбираемых программно режимов сбережения энергии.

В режиме холостого хода ЦПУ не функционирует, в то время как функционируют СОЗУ, таймеры/счётчики, SPI порт и система прерываний. В микроконтроллере существу-

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

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

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

2.1 Технические характеристики и функциональные возможности

максимальная тактовая частота, МГц ($U_{\#VCC}$ = 5,0 В)	16
максимальная тактовая частота, МГц ($U_{\#VCC}$ = 3,3 В)	8
разрядность АЛУ	8
объём встроенного ОЗУ, бит	512 × 8
память программ (Flash типа), бит	8К × 8
ЭСППЗУ, бит	1К × 8
количество источников прерываний	20
количество параллельных 8-разрядных портов	4
число каналов аналого-цифрового преобразователя	8
число разрядов аналого-цифрового преобразователя	10
16-разрядных таймеров	1
8-разрядных таймеров	2
последовательный порт UART	1
последовательный периферийный интерфейс SPI	1
аналоговый компаратор	1
сторожевой таймер WDT	1
число режимов пониженного потребления мощности	6

Микросхемы 1887ВЕ4У разработаны в металлокерамическом корпусе Н16.48-2В.

Номинальные значения напряжения питания микросхем: плюс 5,0 В или плюс 3,3 В. Допустимое отклонение напряжения питания от номинального ±10 %. Амплитуда пульсаций напряжения питания не более 50 мВ.

Напряжение источника опорного напряжения от 2,0 В до $U_{\cap VCC}$.

Допустимое отклонение напряжения питания от крайних значений минус 1 % для напряжения 4,5 В и плюс 1 % для напряжения 5,5 В.

Допустимое отклонение напряжения питания от крайних значений минус 1 % для напряжения 3,0 В и плюс 1 % для напряжения 3,6 В.

2.2 Особенности архитектуры

Особенности архитектуры ИС 1887ВЕ4У:

- быстродействующая архитектура типа «регистр-регистр»;
- регистровое ОЗУ емкостью до 512 байт;
- последовательный периферийный интерфейс SPI;
- последовательный синхронно-асинхронный приемопередатчик UART;
- двухпроводной последовательный интерфейс TWI;
- 16-разрядный таймер/счетчик;
- два 8-разрядных таймера/счетчика;
- 8-разрядный сторожевой таймер;
- 8-канальный 10-разрядный аналого-цифровой преобразователь;
- аналоговый компаратор;
- 4-канальный ШИМ;
- четыре 8-разрядных порта ввода-вывода;
- режимы холостого хода IDLE и хранения POWERDOWN.

Структурная схема ИС 1887ВЕ4У приведена на рисунке 2.1.

Условное графическое обозначение микросхемы приведено на рисунке 2.2.

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

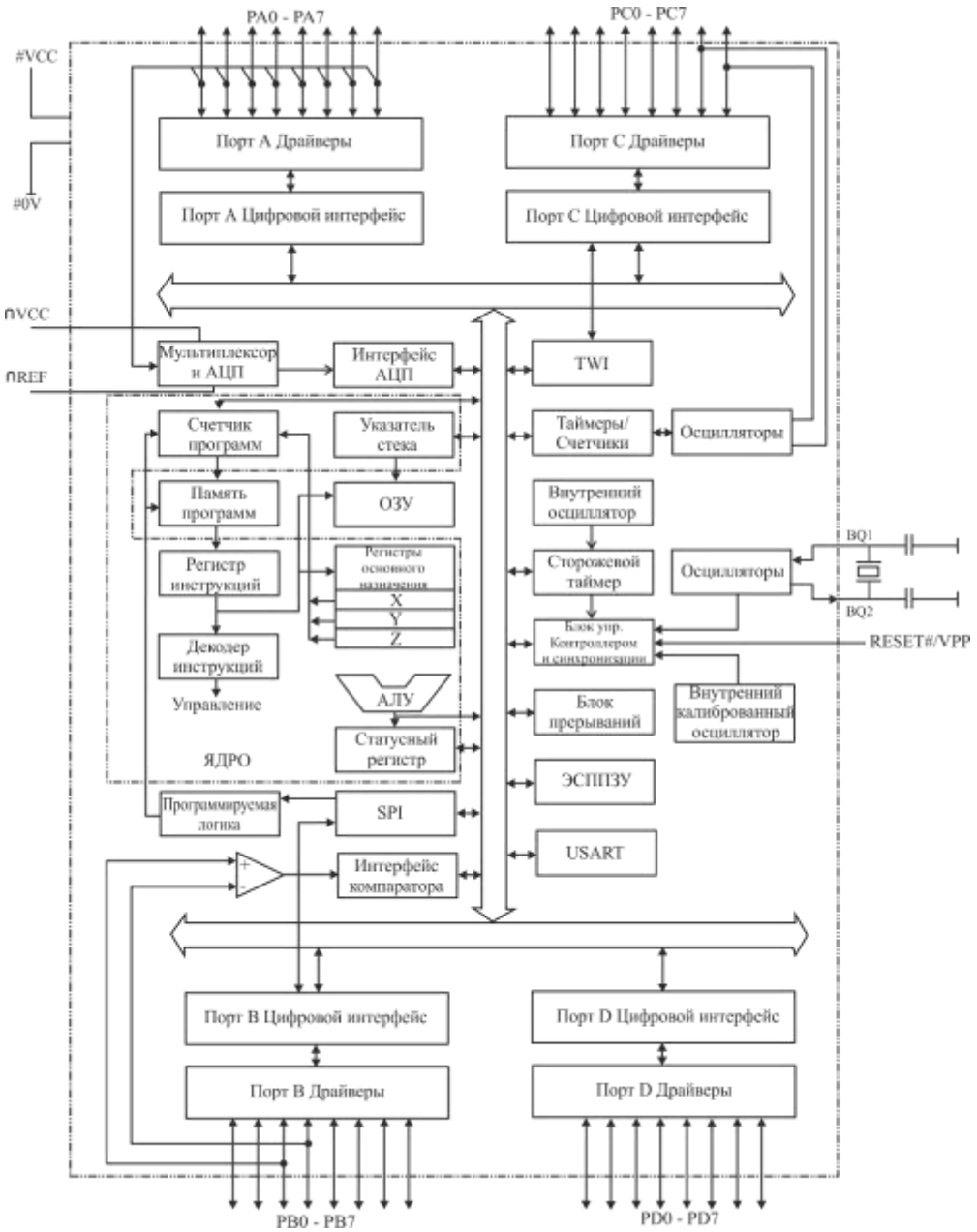


Рисунок 2.1 – Структурная схема ИС 1887ВЕ4У

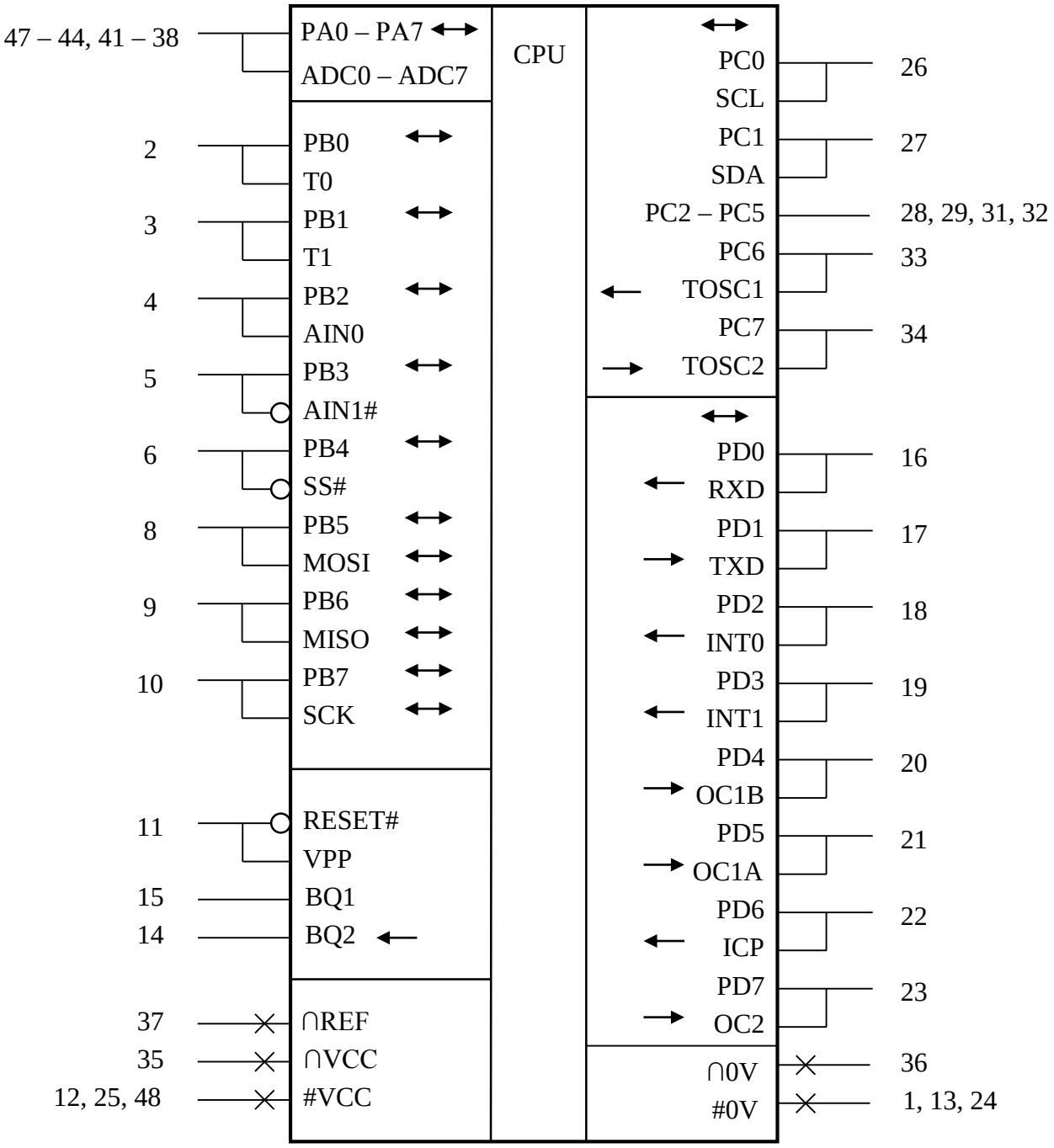


Рисунок 2.2 – Условное графическое обозначение ИС 1887ВЕ4У

Таблица 2.1 – Назначение выводов микросхемы

Номер вывода	Обозначение		Функциональное назначение вывода	Тип вывода
	вывода	альтернативной функции вывода		
47–44, 41–38	PA0–PA7	 ADC0–ADC7	Вход/выход 8-разрядного двунаправленного порта A Вход АЦП, каналы 0 – 7	I/O I
2, 3	PB0, PB1	 T0, T1	Вход/выход 8-разрядного двунаправленного порта B Вход таймеров/счетчиков 0, 1	I/O I
4, 5	PB2, PB3	 AIN0 AIN1#	Вход/выход 8-разрядного двунаправленного порта B Прямой вход аналогового компаратора Инверсный вход аналогового компаратора	I/O I I
6	PB4	 SS#	Вход/выход 8-разрядного двунаправленного порта B Инверсный вход выбора последовательного периферийного интерфейса	I/O I
8, 9	PB5, PB6	 MOSI, MISO	Вход/выход 8-разрядного двунаправленного порта B Вход/выход данных SPI	I/O I/O
10	PB7	 SCK	Вход/выход 8-разрядного двунаправленного порта B Вход/выход тактового сигнала SPI	I/O I/O
26	PC0	 SCL	Вход/выход 8-разрядного двунаправленного порта C Вход/выход тактового сигнала TWI	I/O I/O
27	PC1	 SDA	Вход/выход 8-разрядного двунаправленного порта C Вход/выход данных TWI	I/O I/O
28, 29, 31, 32	PC2–PC5		Вход/выход 8-разрядного двунаправленного порта C	I/O
33	PC6	 TOSC1	Вход/выход 8-разрядного двунаправленного порта C Вход подключения часового кварцевого резонатора	I/O I
34	PC7	 TOSC2	Вход/выход 8-разрядного двунаправленного порта C Выход подключения часового кварцевого резонатора	I/O O
16	PD0	 RXD	Вход/выход 8-разрядного двунаправленного порта D Вход последовательных данных передатчика	I/O I
17	PD1	 TXD	Вход/выход 8-разрядного двунаправленного порта D Выход последовательных данных передатчика	I/O O
18, 19	PD2, PD3	 INT0, INT1	Вход/выход 8-разрядного двунаправленного порта D Вход внешнего прерывания 0, 1	I/O I
20, 21	PD4, PD5	 OC1B, OC1A	Вход/выход 8-разрядного двунаправленного порта D Компараторный выход B и A таймера/счетчика	I/O O
22	PD6	 ICP	Вход/выход 8-разрядного двунаправленного порта D Вход захвата таймера/счетчика 1	I/O I
23	PD7	 OC2	Вход/выход 8-разрядного двунаправленного порта D Компараторный выход таймера/счетчика 2	I/O O
37	∩REF		Вывод опорного напряжения	–
35	∩VCC		Вывод питания аналоговой части микросхемы	–
36	∩0V		Общий вывод аналоговой части микросхемы	–
14 15	BQ2 BQ1		Выводы для подключения кварцевого резонатора	O I
11	RESET#/ VPP		Вывод сигнала общего сброса Вывод программирования	I I
1, 13, 24	#0V		Общий вывод 0 В	–
12, 25, 48	#VCC		Выводы питания цифровой части микросхемы	–

Примечания

1 Выводы 7, 30, 42, 43 не задействованы.

2 Условные обозначения: I – вход, O – выход, I/O – вход/выход.

2.3 Система команд

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

2.4 Электрические параметры микросхем

Электрические параметры микросхем 1887ВЕ4У при приемке и поставке приведены в таблице 2.2

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

Таблица 2.2 – Электрические параметры микросхем при приемке и поставке

Наименование параметра, единица измерения, режим измерения		Буквенное обозначение параметра	Норма параметра		Температура среды, °С
			не менее	не более	
1		2	3	4	5
1 Выходное напряжение низкого уровня по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, В	I_{OL} = 20 мА, $U_{\#VCC} = U_{\cap VCC}$ = 5,0 В	U_{OL}	–	0,6	−60 ± 3 25 ± 10 85 ± 3
	I_{OL} = 12 мА, $U_{\#VCC} = U_{\cap VCC}$ = 3,3 В				
2 Выходное напряжение высокого уровня по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, В, I_{OH} = −3 мА	$U_{\#VCC} = U_{\cap VCC}$ = 5,0 В	U_{OH}	4,2	–	
	$U_{\#VCC} = U_{\cap VCC}$ = 3,3 В		2,4	–	
3 Токи утечки на входе по выводам PA0 – PA7, PB0 – PB7, PC0 – PC7, PD0 – PD7, мкА, $U_{\#VCC} = U_{\cap VCC}$ = 5,5 В; 0,45 В ≤ U_I ≤ U_{CC}		I_{ILL}, I_{ILH}	−8	8	25 ± 10 85 ± 3
4 Динамический ток потребления по выводам #VCC, ∩VCC в активном режиме, мА, f_{CI} = 4 МГц	$U_{\#VCC} = U_{\cap VCC}$ = 5,5 В	I_{OCC1}	–	30	−60 ± 3 25 ± 10 85 ± 3
	$U_{\#VCC} = U_{\cap VCC}$ = 3,6 В		–	10	
5 Динамический ток потребления по выводам #VCC, ∩VCC в режиме пониженного потребления, мА, f_{CI} = 4 МГц	$U_{\#VCC} = U_{\cap VCC}$ = 5,5 В	I_{OCC2}	–	8	
	$U_{\#VCC} = U_{\cap VCC}$ = 3,6 В		–	6	
6 Ток потребления по выводам #VCC, ∩VCC в режиме хранения, мкА	$U_{\#VCC} = U_{\cap VCC}$ = 5,5 В	I_{CCS}	–	200	
	$U_{\#VCC} = U_{\cap VCC}$ = 3,6 В		–	125	
7 Напряжение смещения компаратора, мВ, $U_I = U_{CC}/2$	$U_{\#VCC} = U_{\cap VCC}$ = 5,0 В	U_{IO}	–	40	
	$U_{\#VCC} = U_{\cap VCC}$ = 3,3 В				
8 Функциональный контроль	$U_{\#VCC} = U_{\cap VCC}$= (3,0 – 3,6) В, f_{CI}= 8 МГц	ФК	–	–	
	$U_{\#VCC}=U_{\cap VCC}$ = (4,5 – 5,5) В, f_{CI}= 16 МГц				

Таблица 2.3 – Значения предельно допустимых электрических режимов эксплуатации и предельных режимов в диапазоне рабочих температур от минус 60 ºС 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 Напряжение питания цифровой части ИС, В		$U_{\#VCC}$ [1)]	3,0 4,5	3,6 5,5	–0,3	7,0
2 Напряжение питания аналоговой части ИС, В		$U_{\cap VCC}$ [1)]	3,0 4,5	3,6 5,5	–0,3	7,0
3 Входное напряжение низкого уровня, В		U_{IL}	–0,5	$0,2U_{\#VCC}$	–0,6	–
4 Входное напряжение высокого уровня по выводам PA0–PA7, PB0–PB7, PC0–PC7, PD0–PD7, В		U_{IH1}	$0,6U_{\#VCC}$	$U_{\#VCC}+0,5$	–	$U_{\#VCC}+0,6$
5 Входное напряжение высокого уровня по выводу BQ1, В		U_{IH2}	$0,8U_{\#VCC}$	$U_{\#VCC}+0,5$	–	$U_{\#VCC}+0,6$
6 Входное напряжение высокого уровня по выводу RESET#/VPP, В		U_{IH3}	$0,9U_{\#VCC}$	$U_{\#VCC}+0,5$	–	$U_{\#VCC}+0,6$
7 Напряжение программирования на выводе RESET#/VPP, В		U_{PP}	11,5	12,5	–	13,0
8 Выходной ток низкого уровня по выводам PA0–PA7, PB0–PB7, PC0–PC7, PD0–PD7, мА, $U_{\#VCC} = U_{\cap VCC}$ = (4,5–5,5) В $U_{\#VCC} = U_{\cap VCC}$ = (3,0–3,6) В		I_{OL} [2)]	 – –	 20 12	 – –	 25 25
9 Выходной ток высокого уровня по выводам PA0–PA7, PB0–PB7, PC0–PC7, PD0–PD7, мА		I_{OH} [2)]	–3,0	–	–5,0	–
10 Длительность фронтов входных сигналов, нс, - для входа BQ1 - для остальных входов		t_{LH}, t_{HL}	 – –	 10 20	 – –	 – –
11 Тактовая частота, МГц	$U_{\#VCC} = U_{\cap VCC}$ = = (3,0–3,6) В	f_{CI}	0	8	–	–
	$U_{\#VCC} = U_{\cap VCC}$ = = (4,5–5,5) В		0	16		
12 Емкость нагрузки по выводам PA0–PA7, PB0–PB7, PC0–PC7, PD0–PD7, пФ		C_L	–	80	–	150

[1)] Между напряжениями питания аналоговой части и цифровой части микросхемы должно сохраняться соотношение $|U_{\#VCC} - U_{\cap VCC}| \le 0,3$ В.

[2)] Суммарный предельно допустимый нагрузочный ток по выводам PA0–PA7, PB0–PB7, PC0–PC7, PD0–PD7 не должен превышать 400 мА. Конкретные значения для различных выводов зависят от режима работы и приведены в настоящем техническом описании.

3 Функциональное описание ИС 1887ВЕ4У

ИС 1887ВЕ4У представляет собой маломощный КМОП 8-разрядный микроконтроллер, построенный на расширенной архитектуре RISC. Благодаря выполнению команд в течение одного тактового цикла, ИС 1887ВЕ4У достигает производительности в 1 миллион операций в секунду, что позволяет разработчику систем оптимизировать потребляемую мощность с учетом скорости обработки данных.

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

Микроконтроллер 1887ВЕ4У содержит 8 Кбайт внутрисистемной программируемой флэш-памяти с возможностью считывания во время записи, ЭСППЗУ емкостью 1 Кбайт, 512 байт СОЗУ, 32 линии ввода-вывода общего назначения, 32 рабочих регистра общего назначения, три гибких таймера/счетчика с режимами сравнения, внутренние и внешние прерывания, последовательный программируемый универсальный синхронно-асинхронный последовательный порт UART, побайтно ориентированный двухпроводной последовательный интерфейс, 8-канальный 10-разрядный АЦП с дополнительным дифференциальным входным каскадом с программируемым усилением. Микроконтроллер имеет также программируемый сторожевой таймер с внутренним генератором, последовательный периферийный интерфейс SPI и шесть энергосберегающих режимов (sleep), выбираемых программно. Режим холостого хода останавливает ЦПУ, в то время как СОЗУ, таймеры/счетчики, SPI интерфейс и система прерывания продолжают работать. Режим микропотребления сохраняет содержимое регистров, но приостанавливает генератор, отключая все остальные функции кристалла до наступления следующего прерывания или общего сброса. В режиме хранения продолжает работать асинхронный таймер, в то время как остальная часть устройства находится в спящем режиме. Режим пониженного шума АЦП останавливает ЦПУ и все периферийные модули для того, чтобы свести к минимуму шум при переключении во время АЦП преобразований. В режиме ожидания работает кварцевый генератор, в то время как остальная часть устройства находится в спящем режиме. Это позволяет обеспечить очень быстрый запуск при малой потребляемой мощности. В длительном режиме ожидания продолжают работать основной генератор и асинхронный таймер.

Микроконтроллер изготавливается с использованием технологии энергонезависимой перепрограммируемой памяти с высокой плотностью упаковки. Встроенная в кристалл флэш-память с внутрисистемным программированием позволяет перепрограммировать память с помощью последовательного SPI интерфейса или же с помощью также встроенной в кристалл памяти загрузки. Программа загрузки может использовать любой интерфейс для загрузки прикладных программ в флэш-память. Программное обеспечение (ПО) в секции загрузки флэш-памяти будет продолжать работать во время обновления секции приложений флэш-памяти, благодаря чему обеспечивается реальное считывание во время записи. Благодаря объединению 8-разрядного RISC ЦПУ со встроенной самопрограммируемой флэш-памятью на одном кристалле, контроллер 1887ВЕ4У является маломощным микроконтроллером, обеспечивающим чрезвычайно гибкое и экономичное решение для многих видов применения систем встроенного управления.

ИС 1887ВЕ4У поддерживает полный набор программ и средств системной разработки, включая С-компиляторы, макроассемблеры, средства отладки и моделирования, внутрисхемные эмуляторы и оценочные комплекты.

Описание выводов

Обозначение вывода	Описание вывода
#VCC	Напряжение питания
#0V	Общий вывод
Порт A (PA7–PA0)	Аналоговые входы АЦП. Порт A является также 8-разрядным двунаправленным портом ввода-вывода, если не используется АЦП. Выводы порта могут служить также в качестве внутренних нагрузочных резисторов (выбираются для каждого разряда). Выходные буферы порта A имеют характеристики с высокой нагрузочной способностью и емкостью источника. Когда выводы PA0–PA7 используются в качестве входов и извне переводятся в низкое логическое состояние, они становятся источниками тока, если активированы внутренние нагрузочные резисторы. Выводы порта A находятся в состоянии высокого импеданса в случае активации сброса, даже если отсутствует тактирование
Порт B (PB7–PB0)	Порт B является 8-разрядным двунаправленным портом ввода-вывода с внутренними нагрузочными резисторами (выбираемыми для каждого разряда). Буферы порта B имеют характеристики с высокой нагрузочной способностью и емкостью источника. Когда выводы PB0–PB7 используются в качестве входов и извне переводятся в низкое логическое состояние, они становятся источниками тока, если активированы внутренние нагрузочные резисторы. Выводы порта B находятся в состоянии высокого импеданса в случае активации сброса, даже если отсутствует тактирование. Порт B выполняет также различные специальные функции
Порт C (PC7–PC0)	Порт C является 8-разрядным двунаправленным портом ввода-вывода с внутренними нагрузочными резисторами (выбираемыми для каждого разряда). Буферы порта C имеют характеристики с высокой нагрузочной способностью и емкостью источника. Когда выводы PC0–PC7 используются в качестве входов и извне переводятся в низкое логическое состояние, они становятся источниками тока, если активированы внутренние нагрузочные резисторы. Выводы порта C находятся в состоянии высокого импеданса в случае активации сброса, даже если отсутствует тактирование
Порт D (PD7–PD0)	Порт D является 8-разрядным двунаправленным портом ввода-вывода с внутренними нагрузочными резисторами (выбираемыми для каждого разряда). Буферы порта D имеют характеристики с высокой нагрузочной способностью и емкостью источника. Когда выводы PD0–PD7 используются в качестве входов и извне переводятся в низкое логическое состояние, они становятся источниками тока, если активированы внутренние нагрузочные резисторы. Выводы порта D находятся в состоянии высокого импеданса в случае активации сброса, даже если отсутствует тактирование. Порт D выполняет также различные специальные функции
RESET#	Вывод для сигнала общего сброса. Состояние низкого уровня на этом выводе продолжительнее, чем минимальная длина импульса (смотри таблицу 3.15), генерирует сигнал сброса, даже если отсутствует тактирование. Не гарантируется, что более короткие импульсы осуществят сброс

Обозначение вывода	Описание вывода
BQ1	Вход инвертирующего усилителя осциллятора и вход схемы внутреннего тактирования
BQ2	Выход инвертирующего усилителя осциллятора
∩VCC	∩VCC представляет собой вывод напряжения питания для порта A и АЦП. Он должен быть внешне подсоединен к #VCC, даже если АЦП не используется. Если АЦП используется, то он должен быть подсоединен к #VCC через фильтр нижних частот
∩REF	∩REF является аналоговым выводом опорного напряжения АЦП

3.1 Ядро микроконтроллера

В данном подразделе описывается архитектура ядра микропроцессора в целом. Основной функцией ядра ЦПУ является обеспечение правильного исполнения программы. С учетом этого ЦПУ должен иметь возможность доступа к устройствам памяти, выполнения вычислений, управления периферийными устройствами и обращения с прерываниями.

3.1.1 Анализ архитектуры

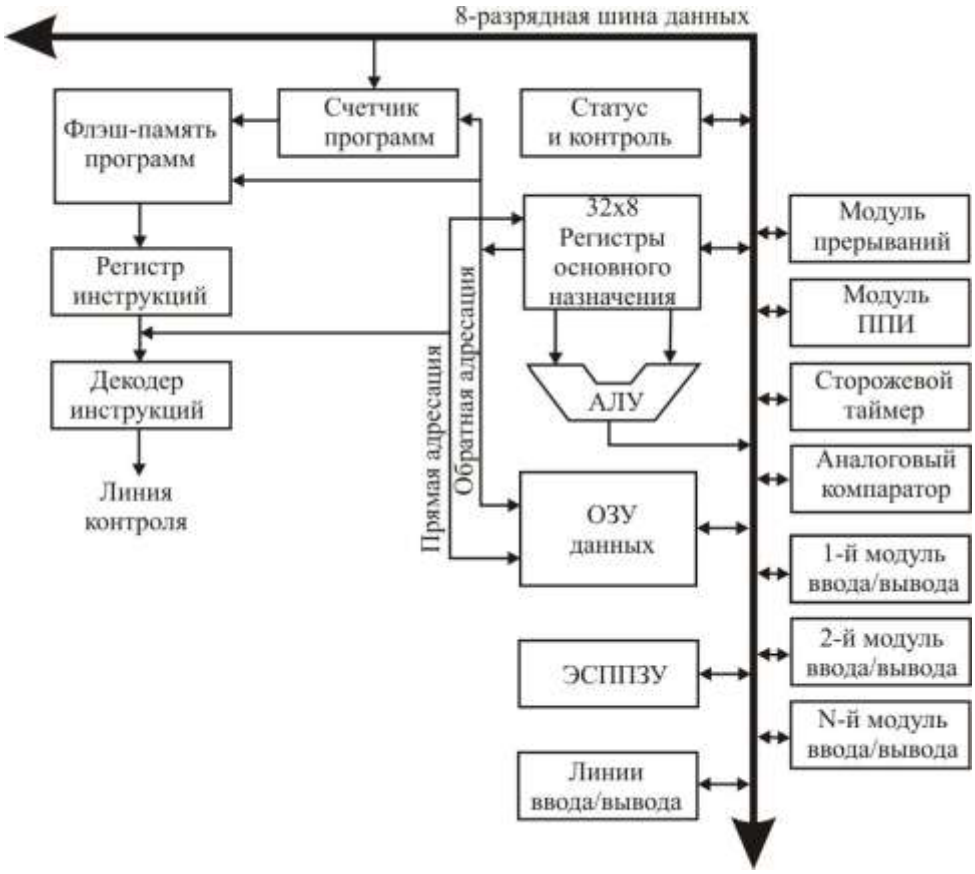


Рисунок 3.1 – Блок-схема архитектуры ядра микропрограммного управления

Для того чтобы максимально улучшить характеристики и параллелизм, в микропроцессоре используется Гарвардская архитектура с отдельными устройствами памяти и шинами для программ и данных. Команды в памяти программ выполняются на едином уровне конвейеризации. Во время исполнения одной команды следующая команда предвари-

тельно извлекается из памяти программ. Такой принцип позволяет исполнять команды за каждый тактовый цикл. Память программ представляет собой внутрипроцессорную перепрограммируемую флэш-память.

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

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

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

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

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

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

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

Пространство памяти ввода-вывода содержит 64 адреса для регистров периферийных компонентов микроконтроллера. Доступ к памяти ввода-вывода может осуществляться напрямую или адресацией пространства данных, следующих за аналогичными адресами файла регистров, 0x20–0x5F.

3.1.2 Арифметико-логическое устройство АЛУ

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

Регистр состояний содержит информацию о результатах самых последних выполненных арифметических команд. Данная информация может быть использована для изме-

нения потока программ при выполнении условных операций. Необходимо обратить внимание на то, что регистр состояний обновляется после выполнения всех операций АЛУ. В подобной ситуации, во многих случаях, это устраняет необходимость в использовании специальных команд сравнения, что обеспечивает ускоренный и более компактный код. Регистр состояния не сохраняется автоматически при входе в процедуру прерывания и не сохраняется при возвращении из прерывания. Подобная ситуация должна обрабатываться программно.

Определение регистра состояния SREG – выглядит следующим образом:

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	

Разряд 7: I – разрешение общего прерывания

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

Разряд 6: T – хранение копии разряда

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

Разряд 5: H – флаг половинного переноса

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

Разряд 4: S – знаковый разряд, S = N ^ V

Разряд S всегда означает «исключающее или» между отрицательным флагом N и флагом V дополнения до двух.

Разряд 3: V – флаг двойного дополнительного переполнения

Флаг V дополнения до двух поддерживает арифметику дополнения до двух. Подробно приведено в приложении A «Описание системы команд».

Разряд 2: N – отрицательный флаг

Отрицательный флаг N означает отрицательный результат при арифметической или логической операции. Для подробного рассмотрения см. приложение A «Описание системы команд».

Разряд 1: Z – нулевой флаг

Нулевой флаг Z означает нулевой результат арифметической или логической операции. Для подробного рассмотрения см.приложение A «Описание системы команд».

Разряд 0: C – флаг переноса

Флаг переноса C обозначает перенос в логической или арифметической операции.

3.1.3 Файл регистров общего назначения

Файл регистров оптимизирован для расширенного набора команд. Для достижения требуемых характеристик и гибкости, файлом регистров поддерживаются следующие варианты ввода-вывода:

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

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

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

Примечание – Принятые условные обозначения: мл. – младший; ст. – старший.

Рисунок 3.2 – Рабочие регистры общего назначения ЦПУ

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

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

Регистр X, регистр Y и регистр Z

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

Рисунок 3.3 – X-, Y-, Z-регистры

В различных режимах адресации эти адресные регистры имеют такие функции, как фиксированное смещение, автоматический инкремент и автоматический декремент.

3.1.4 Указатель стека

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

Указатель стека указывает на ту часть стека данных СОЗУ, в которой расположены стеки подпрограмм и прерываний. Эта область в СОЗУ должна определяться программой до того, как исполняются любые вызовы подпрограмм или разрешаются прерывания. Указатель стека должен иметь значение выше 0×60.

Указатель стека уменьшается на единицу по мере того, как данные перемещаются в стек с помощью команды PUSH, и он уменьшается на два, когда обратный адрес перемещается в стек с помощью вызова подпрограммы или прерывания. Указатель стека увеличивается на единицу, когда данные выходят из стека с помощью команды POP, и увеличивается на два, когда данные выходят из стека, возвращаясь из подпрограммы RET или возвращаясь из прерывания RETI.

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

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	
Нач. значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Временные диаграммы исполнения команд

Ниже описываются общие концепции временных диаграмм доступа для выполнения команд. Центральное процессорное устройство работает от тактового сигнала clk_{CPU}, сформированного непосредственно от выбранного для микроконтроллера источника тактовых импульсов. При этом не используется внутреннее деление тактовой частоты.

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

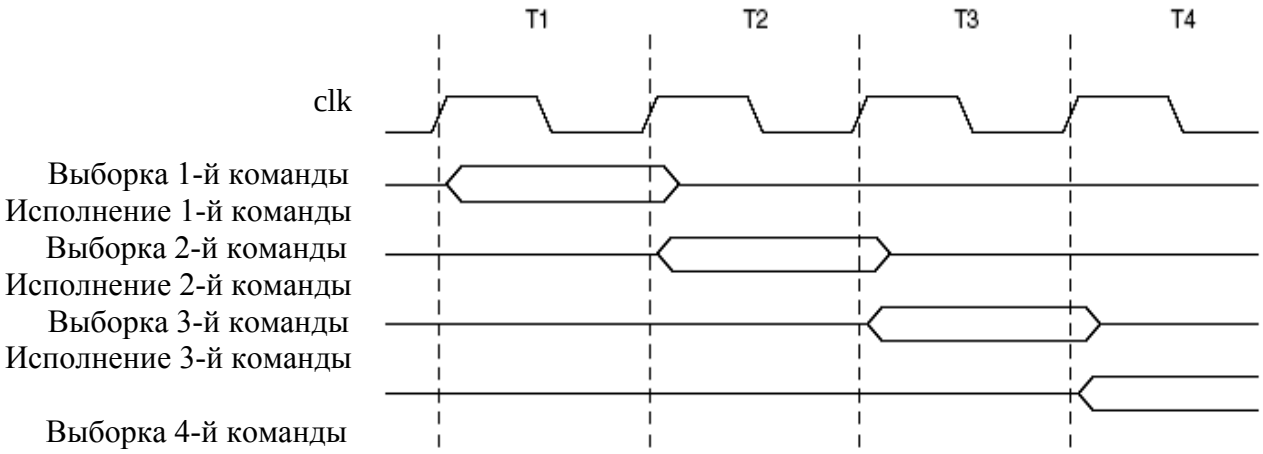


Рисунок 3.4 – Параллельная выборка и исполнение команд

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

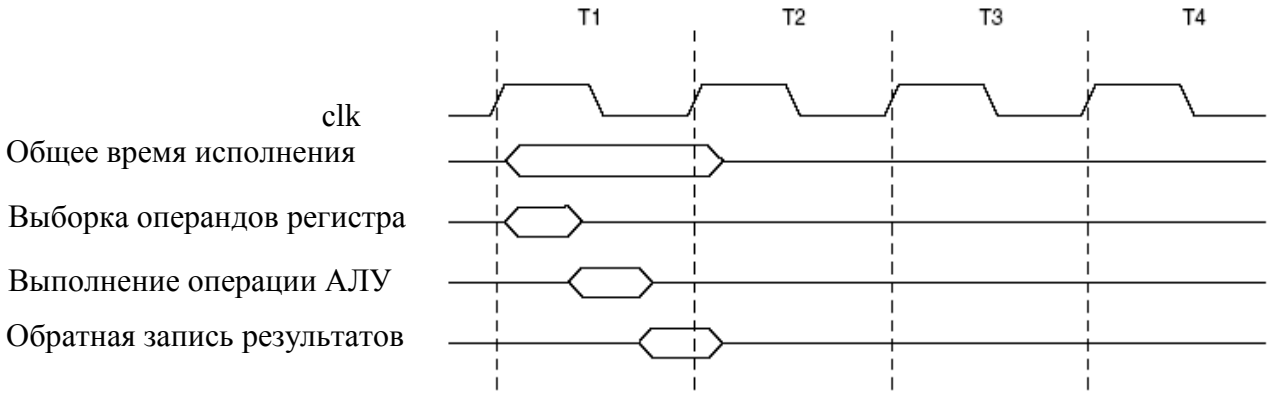


Рисунок 3.5 – Работа АЛУ в одиночном цикле

3.1.5 Сброс и обработка прерываний

ИС 1887ВЕ4У содержит несколько различных источников прерывания. Каждое из этих прерываний и отдельный вектор сброса имеют отдельный программный вектор в пространстве памяти программ. Все прерывания имеют свои отдельные разряды разрешения, которые должны быть установлены вместе с установкой разряда разрешения глобального прерывания в регистре состояний для того, чтобы разрешить прерывание. В зависимости от значения программного счетчика, прерывания могут быть автоматически запрещены, когда программируются биты блокировки BLB02 или BLB12. Подобная особенность повышает безопасность программного обеспечения. Для получения подробной информации см. подраздел 3.20 «Программирование памяти».

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 имеет наивысший приоритет, а следующим является INT0 – запрос 0 внешнего прерывания. Векторы прерывания могут быть перемещены в начало секции загрузки флэш-памяти путем установки разряда IVSEL в регистре управления общим прерыванием (GICR). Подробно вопрос по «прерываниям» рассмотрен в подразделе 3.7. Вектор сброса может быть также перемещен в начало секции загрузки флэш-памяти с помощью программирования бита BOOTRST, смотри подраздел 3.19 «Поддержка загрузчика – самопрограммирование».

При возникновении прерывания бит I разрешения глобального прерывания очищается и все прерывания блокируются. Пользовательское ПО может записывать логическую единицу в бит I для разрешения вложенных прерываний. Все разрешенные прерывания могут прерывать текущее прерывание. Разряд I устанавливается автоматически при исполнении возврата из команды прерывания RETI.

Существует два основных типа прерываний. Первый тип запускается событием, которое устанавливает флаг прерывания. Для этого типа прерываний программный счетчик переходит на вектор прерываний для исполнения подпрограммы обработки прерываниями, при этом аппаратно очищается соответствующий флаг прерывания. Флаги прерывания могут также очищаться путем записи логической единицы в позицию разряда соответствующего флага. Если возникает условие прерывания в то время, как соответствующий разряд 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 разряде глобального разрешения прерывания I, соответствующий флаг(и) будет установлен и внесен в память до тех пор, пока 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 прерывания. Если условие прерывания исчезает до разрешения прерывания, то прерывание не будет запущено.

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

Обратите внимание на то, что регистр статуса не сохраняется автоматически при возврате из процедуры прерывания. 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 прерываний во время запланированной последовательности записи ЭСППЗУ.

Пример ассемблерного кода	
in r16, SREG ;	запись значения в SREG
cli ;	блокировка прерываний в течение времени цикла
sbi EECR, EEMWE ;	запуск записи в EEPROM
sbi EECR, EEWE	
out SREG, r16 ;	восстановление значения SREG (I-bit)

Пример C-кода
char cSREG; cSREG = SREG; /* запись значения в SREG */ /* блокировка прерываний в течение времени цикла */ _CLI(); EECR |= (1<<EEMWE); /* запуск записи в EEPROM */ EECR |= (1<<EEWE); SREG = cSREG; /* восстановление значения SREG (I-bit)*/

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

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

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

Время отклика прерывания

Отклик выполнения 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 прерывания, 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 выполнения прерывания составляет сумму времени выхода из спящего режима и длительности четырех тактовых циклов.

Возвращение из режима обработки прерывания обычно занимает четыре тактовых цикла. Во время этих четырех циклов значение программного счетчика (два байта) восстанавливается из стека, указатель стека увеличивается на два, и устанавливается разряд «I» в SREG.

3.2 Устройства памяти микроконтроллера 1887ВЕ4У

В этом подразделе описываются различные устройства памяти ИС 1887ВЕ4У. Архитектура микроконтроллера имеет два основных пространства памяти: память данных и память программ. Кроме того, ИС 1887ВЕ4У отличается наличием ЭСППЗУ для хранения данных. Все три пространства памяти являются линейными и регулярными.

3.2.1 Внутрисистемная перепрограммируемая флэш-память программ

ИС 1887ВЕ4У содержит 8 Кбайт встроенной внутрисистемной перепрограммируемой флэш-памяти для хранения программ. Поскольку все команды процессора имеют ширину 16 или 32 разряда, флэш-память организована в виде 4К × 16. Для повышения безопасности ПО пространство флэш-памяти разделено на две секции: секцию загрузчика и секцию прикладных программ.

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

Программный счетчик PC для ИС 1887ВЕ4У имеет разрядность, равную 12 бит, что позволяет адресовать 4К адресов памяти программ. Работа области загрузчика и связанных с ней разрядов блокировки области загрузчика подробно описаны в подразделе 3.19 «Поддержка загрузчика – самопрограммирование».

При записи данных в массив памяти, необходимость в предварительном стирании отсутствует. Состояние ячеек памяти после команд стирания – 0000. Запись производится постранично (размер страницы – 32 16-разрядных слова).

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

На рисунке 3.6 показана организация памяти программ для ИС 1887ВЕ4У.

Рисунок 3.6 – Карта памяти программ

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

Пять различных режимов адресации для памяти данных включают: прямой, косвенный со смещением, косвенный, косвенный с преддекрементом и косвенный с постинкрементом. В регистровом файле регистры R26–R31 имеют альтернативную функцию регистров-указателей с косвенной адресацией.

Непосредственная адресация охватывает все пространство данных.

Режим косвенный со смещением охватывает 63 адреса от основного адреса, выданного регистром Y или Z.

При использовании режимов с косвенной адресацией, регистров с автоматическим преддекрементом и постинкрементом, значения адресных регистров X, Y и Z уменьшаются или увеличиваются.

32 рабочих регистра общего назначения, 64 регистра ввода-вывода и 512 байт СОЗУ внутренних данных в ИС 1887ВЕ4У доступны с помощью всех упомянутых режимов адресации.

Регистровый файл	Пространство адресов данных
R0	$0000
R1	$0001
R2	$0002
…	…
R29	$001D
R30	$001E
R31	$001F
Регистры ввода-вывода	
$00	$0020
$01	$0021
$02	$0022
…	…
$3D	$005D
$3E	$005E
$3F	$005F
	Внутреннее СОЗУ
	$0060
	$0061
	…
	$025E
	$025F

Рисунок 3.7 – Карта памяти данных

Времена доступа к памяти данных

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

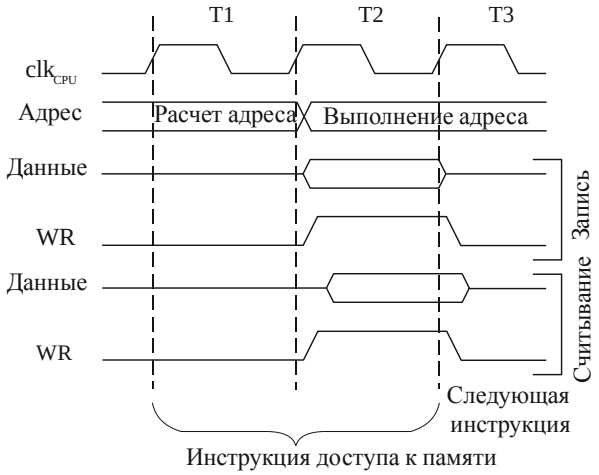


Рисунок 3.8 – Циклы доступа к данным СОЗУ на кристалле

3.2.2 Память данных ЭСППЗУ

ИС 1887ВЕ4У содержит 1 Кбайт памяти данных ЭСППЗУ. Она организована в виде отдельного пространства данных, в котором могут считываться и записываться отдельные байты. Доступ между ЭСППЗУ и ЦПУ описан далее с указанием регистров адреса, регистров данных и регистра управления ЭСППЗУ.

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

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

Доступ считывания/записи ЭСППЗУ

Регистры доступа к ЭСППЗУ доступны в пространстве регистров ввода-вывода.

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

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

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

Адрес ЭСППЗУ
Регистр – EEARH и EEARL

Бит	7	6	5	4	3	2	1	0	
	-	-	-	-	-	-	EEAR9	EEAR8	EEARH
	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
Чтение/запись	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч/З	
	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное	0	0	0	0	0	0	0	X	
значение	X	X	X	X	X	X	X	X	

Разряды 15–9: Res – резервные разряды

Эти разряды в ИС 1887ВЕ4У являются резервными и всегда считываются как ноль.

Разряды 8–0: EEAR–EEAR0 – адрес ЭСППЗУ

Регистры адреса ЭСППЗУ – EEARH и EEARL – указывают адрес байта в 1 Кбайтном адресном пространстве ЭСППЗУ. Байты данных ЭСППЗУ адресуются линейным образом в пространстве от 0 до 1023. Начальное значение EEAR не определено. Надлежащее значение должно быть записано до того, как будет получен доступ к ЭСППЗУ.

Регистр данных ЭСППЗУ–EEDR

Бит	7	6	5	4	3	2	1	0	
								LSB	EEDR
Чтение/запись	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–EEDR0 – данные ЭСППЗУ

Для операции записи ЭСППЗУ регистр EEDR содержит данные, которые должны быть записаны в ЭСППЗУ в адрес, указанный регистром EEAR. Для операции считывания ЭСППЗУ EEDR содержит данные, считываемые из ЭСППЗУ по адресу, указаннному регистром EEAR.

Управление ЭСППЗУ. Регистр 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: Res – зарезервированные разряды

Для ИС 1887ВЕ4У эти разряды являются резервными и всегда будут считываться как ноль.

Разряд 3: EERIE – разрешение готовности к прерыванию ЭСППЗУ

Запись логической единицы в EERIE разрешает прерывание готовности ЭСППЗУ, если установлен разряд «I» в SREG. Сброс бита EERIE блокирует прерывание. Прерывание готовности ЭСППЗУ формирует постоянное прерывание, когда EEWE очищено.

Разряд 2: EEMWE – разрешение записи основного ЭСППЗУ

Разряд EEMWE определяет, будет ли установка бита EEWE вызывать запись в ЭСППЗУ. При установке бита EEMWE, установка бита EEWE в течение четырех тактовых циклов будет инициировать запись данных в ЭСППЗУ в выбранный адрес. Если EEMWE равно нулю, то установка EEWE не оказывает воздействия. После установления программно бита EEMWE, устройство сбрасывает разряд после четырех тактовых циклов.

Разряд 1: EEWE – разрешение записи ЭСППЗУ

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

1 Дождаться, пока EEWE не станет равным нулю.

2 Дождаться, пока SPMEN в SPMCR не станет равным нулю.

3 Записать новый адрес ЭСППЗУ в EEAR (опция).

4 Записать новые данные ЭСППЗУ в EEDR (опция).

5 Записать логическую единицу в разряд EEMWE при выполнении записи в EEWE в EECR.

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

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

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

Разряд 0: EERE – разрешение считывания ЭСППЗУ

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

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

Для синхронизации доступа к ЭСППЗУ используется калиброванный осциллятор. В таблице 3.1 приведено типичное время программирования для доступа к ЭСППЗУ из ЦПУ.

Таблица 3.1 – Время программирования ЭСППЗУ из ЦПУ

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

Далее приведен пример, показывающий одну функцию, реализованную на ассемблере и одну функцию, реализованную на С для записи в ЭСППЗУ. Этот пример означает, что прерывания контролируются (с помощью глобальной блокировки прерываний) таким образом, что во время выполнения этих функций не происходит прерываний. Также пример демонстрирует, что в программе не присутствует загрузчик флэш-памяти. Если имеет место подобный код, то функция записи ЭСППЗУ должна также дождаться завершения любой текущей команды SPM.

Пример ассемблерного кода

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

Пример С кода

```
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
(
/* Wait for completion of previous write */
while(EECR & (1<<EEWE))
;
/* Set up Address and Data Registers */
EEAR = uiAddress;
EEDR = ucData;
/* Write logical one to EEMWE */
EECR |= (1<<EEMWE);
/* Start eeprom write by setting EEWE */
EECR |= (1<<EEWE);
)
```

Следующий пример показывает функции, реализованные на ассемблере и на «С» для считывания ЭСППЗУ. В этом примере предполагается, что прерывания контролируются таким образом, что во время выполнения функций не происходит прерываний.

Пример ассемблерного кода

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

Пример С кода

```
unsigned char EEPROM_read(unsigned int uiAddress)
(
/* Wait for completion of previous write */
while(EECR & (1<<EEWE))
;
/* Set up Address Register */
EEAR = uiAddress;
/* Start eeprom read by writing EERE */
```

```
EECR |= (1<<EERE);
/* Return data from Data Register */
return EEDR;
)
```

3.2.3 Предотвращение ошибок в ЭСППЗУ

В периоды снижения напряжения данные в ЭСППЗУ могут искажаться, поскольку напряжение питания слишком мало для того, чтобы обеспечить правильную работу ЦПУ и ЭСППЗУ. Подобные проблемы аналогичны тем, с которыми сталкиваются на уровне системных плат, и в таких случаях должны использоваться аналогичные конструкторские решения.

Сбой данных ЭСППЗУ может происходить в двух случаях при слишком низком напряжении питания. Во-первых, регулярная последовательность записи в ЭСППЗУ требует минимального напряжения для корректного выполнения операции. Во-вторых, само ЦПУ может неправильно выполнять команды при слишком низком напряжении питания.

В то же время, можно избежать сбоя данных при работе ЭСППЗУ, если учитывать следующую рекомендацию: необходимо поддерживать общий сброс в активном (низком) состоянии во время низкого напряжения питания. Это можно выполнить путем включения внутреннего устройства обнаружения условий пониженного питания (Brown-out Detector) BOD. Если уровень обнаруженного внутреннего BOD не соответствует требуемому уровню обнаружения, то можно использовать внешнюю схему сброса для защиты от низкого $U_{\#VCC}$. Если сброс происходит в период, когда идет процесс записи, то процесс записи будет завершен, когда восстановится достаточный уровень напряжения питания.

3.2.4 Память ввода-вывода

В пространстве ввода-вывода ИС 1887ВЕ4У размещены все устройства ввода-вывода и периферийные устройства. Доступ к адресам ввода-вывода осуществляется с помощью команд IN и OUT путем передачи данных между 32 рабочими регистрами общего назначения и пространством ввода-вывода. Регистры ввода-вывода внутри диапазона адресов 0x00 – 0x1F обладают возможностью побитовой адресации, используя команды SBI и CBI. В указанных регистрах значение отдельных разрядов можно проверить, используя команды SBIS и SBIC. При использовании специфических команд ввода-вывода IN и OUT необходимо использовать адреса 0x00 – 0x3F регистров ввода-вывода. При адресации к регистрам ввода-вывода как к пространству данных, используя команды LD и ST, к значению адреса этих регистров необходимо добавлять 0x20.

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

Некоторые из флагов статуса очищаются путем записи в них логической единицы. Обратите внимание на то, что инструкции CBI и SBI должны работать для всех битов регистров ввода-вывода; запись единицы в любой флаг, считанный как установленный, приведет к сбросу флага. Команды CBI и SBI работают только с регистрами с 0x00 по 0x1F адрес.

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

3.3 Система синхронизации

На рисунке 3.9 представлены основные системы синхронизации и их распределение. Необязательно, чтобы в данный промежуток времени все тактовые импульсы были актив-

ными. Для снижения потребляемой мощности, подача тактовых импульсов к неиспользуемым модулям может быть приостановлена с помощью использования различных спящих режимов. Подробности в отношении систем синхронизации приведены ниже.

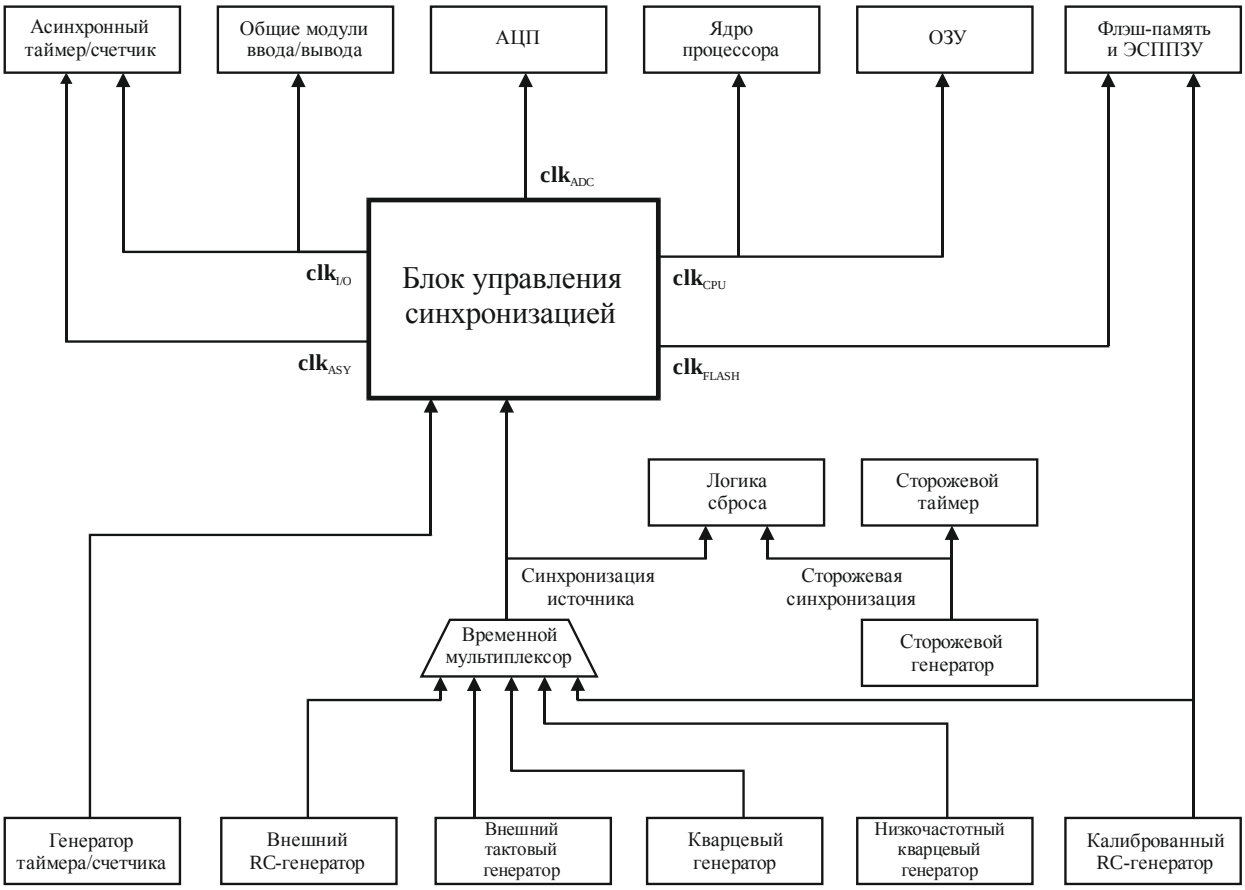


Рисунок 3.9 – Распределение тактовых импульсов

Тактовый сигнал ЦПУ – clk_{CPU}

Тактовый сигнал ЦПУ подсоединен к узлам системы, связанным с работой ядра. Примерами таких модулей являются регистровый файл 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 UART. 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 обратить также 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 флэш-памяти – clk_{FLASH}

Тактовый сигнал флэш-памяти clk_{FLASH} управляет работой интерфейса флэш-памяти. 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.

Источники тактового сигнала

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

Таблица 3.2 – Опции выбора синхронизирующего устройства

Опция выбора синхронизирующего устройства	CKSEL3–CKSEL0
Внешний керамический/кварцевый 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 битов конфигурации «1» означает отсутствие программирования, в то время как «0» означает программирование.	

Ниже представлены различные варианты выбора источника синхронизации. Когда ЦПУ начинает выходить из режима микропотребления или режима хранения, выбранный источник синхронизации используется до начала запуска, обеспечивая при этом стабильную работу генератора до начала исполнения команд. Когда ЦПУ запускается после сброса (RESET), возникает дополнительная задержка, позволяющая довести питание до стабильного уровня к началу нормальной работы. Генератор сторожевого таймера используется для синхронизации этого отрезка времени запуска. Количество циклов генератора WDT, используемых для каждого периода, показано в таблице 3.3. Частота работы генератора сторожевого таймера зависит от напряжения.

Таблица 3.3 – Количество циклов генератора сторожевого таймера WDT для генерации задержки сброса

Типовое время сброса, мс ($U_{\#VCC}$ = 5,0 В)	Типовое время сброса, мс ($U_{\#VCC}$ = 3,3 В)	Количество циклов
8,2	8,6	4K (4096)
130	138	64K (65536)

Источник тактовых импульсов по умолчанию

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

3.3.1 Тактовый генератор с внешним резонатором

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

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

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

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

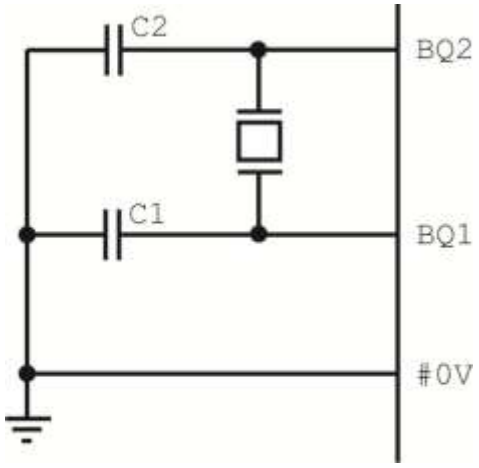


Рисунок 3.10 – Подключение кварцего или керамического резонатора

Генератор может работать в трех различных режимах, каждый из которых оптимизирован для конкретного диапазона частот. Рабочий режим выбирается с помощью ячеек конфигурации CKSEL3–CKSEL1, как показано в таблице 3.4.

Таблица 3.4 – Выбор режимов работы генератора

CKOPT	CKSEL3–CKSEL1	Частотный диапазон[1)], МГц	Рекомендуемые значения конденсаторов C1 и C2 для использования с кварцевыми резонаторами, пФ
1	101[2)]	0,4–0,9	–
1	110	0,9–3,0	12–22
1	111	3,0–8,0	12–22
0	101, 110, 111	1,0–8,0	12–22

[1)] Значения для диапазонов частот являются предварительными.

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

Таблица 3.5 – Времена запуска для выбора частоты кварцевого генератора

CKSEL0	SUT1, SUT0	Время запуска из режима пониженного питания и режима энергосбережения	Дополнительная задержка от сброса, мс ($U_{\#VCC}$ = 5,0 В) / ($U_{\#VCC}$ = 3,3 В)	Рекомендуемое применение
0	00	258 CK[1)]	8,2 / 8,6	Керамический резонатор, быстро нарастающее питание
0	01	258 CK[1)]	130 / 138	Керамический резонатор, медленно нарастающее питание
0	10	1K CK[2)]	–	Керамический резонатор, BOD разрешен
0	11	1K CK[2)]	8,2 / 8,6	Керамический резонатор, быстро нарастающее питание
1	00	1K CK[2)]	130 / 138	Керамический резонатор, медленно нарастающее питание
1	01	16K CK	–	Керамический резонатор, BOD разрешен
1	10	16K CK	8,2 / 8,6	Керамический резонатор, быстро нарастающее питание
1	11	16K CK	130 / 138	Керамический резонатор, медленно нарастающее питание

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

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

Конфигурационный бит CKSEL0 вместе с битами SUT1, SUT0 выбирает временные интервалы для запуска, как это показано в таблице 3.5.

3.3.2 Кварцевый генератор низкой частоты

Для использования часового кварца с частотой 32,768 кГц в качестве основного тактового генератора необходимо выбирать низкочастотный кварцевый генератор с помощью установки конфигурационных бит CKSEL в состояние «1001». Кварцевый генератор следует подсоединять в соответствии с рисунком 3.11. С помощью программирования бита CKOPT пользователь может подключить внутренние емкости к BQ1 и BQ2, устранив тем самым необходимость использования внешних конденсаторов. Внутренние конденсаторы имеют номинал 36 пФ.

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

Таблица 3.6 – Времена запуска для выбора низкочастотного кварцевого тактового генератора

SUT1, SUT0	Время запуска из режима микропотребления и режима хранения	Дополнительная задержка от сброса, мс ($U_{\#VCC}$ = 5,0 В) / ($U_{\#VCC}$ = 3,3 В)	Рекомендуемый вид применения
00	1K CK*	8,2 / 8,6	Быстро нарастающее питание или разрешенный BOD
01	1K CK*	130 / 138	Медленно нарастающее питание
10	32K CK	130 / 138	Стабильная частота при запуске
11	Зарезервировано		

* Эти опции следует использовать, только если стабильность по частоте при запуске не важна для данного вида применения.

3.3.3 Внешний RC генератор

Конфигурация внешнего RC генератора, показанная на рисунке 3.11, предназначена для применений, не требующих точно определенного значения частоты. Частоту можно приблизительно оценить по формуле f = 1/(3RC). Величина C должна быть, по меньшей мере, 22 пФ. Выполняя программирование с помощью перемычки CKOPT, пользователь может подключить внутреннюю емкость в 36 пФ между BQ1 и #0V, устраняя тем самым потребность во внешней емкости.

Генератор может работать в четырех различных режимах, каждый из которых оптимизирован для определенного диапазона частот. Рабочий режим выбирается с помощью бит CKSEL3–CKSEL0, как это показано в таблице 3.7.

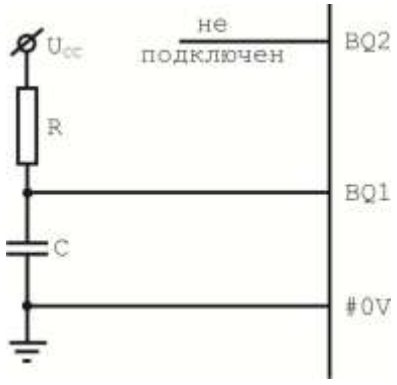


Рисунок 3.11 – Электрическая схема подключения внешнего RC генератора

Таблица 3.7 – Рабочие режимы внешнего RC генератора

CKSEL3–CKSEL0	Частотный диапазон, МГц
0101	0,4–0,9
0110	0,9–3,0
0111	3,0–8,0

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

Таблица 3.8 – Времена запуска для выбора внешнего RC генератора

SUT1, SUT0	Время запуска из режима микропотребления и режима хранения	Дополнительная задержка от сброса, мс ($U_{\#VCC}$ = 5,0 В) / ($U_{\#VCC}$ = 3,3 В)	Рекомендуемый вид применения
00	18CK	–	BOD разрешено
01	18CK	8,2 / 8,6	Быстро нарастающее питание
10	18CK	130 / 138	Медленно нарастающее питание
11	6CK [1)]	8,2 / 8,6	Быстро нарастающее питание или разрешенный BOD

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

3.3.4 Калиброванный внутренний RC генератор

Калиброванный внутренний RC генератор задает фиксированную тактовую частоту 1; 2; 4 или 8 МГц. Все частоты являются номинальными значениями при 5 В и 25 °С. Этот генератор может быть выбран в качестве системного с помощью программирования бит CKSEL, как это показано в таблице 3.9. Если он выбран, то может работать без внешних компонентов. При использовании этого источника бит CKOPT всегда должен быть не запрограммирован. Во время сброса аппаратное обеспечение загружает байт для калибровки в регистр OSCCAL и, таким образом, автоматически калибрует RC генератор. При использовании этого генератора в качестве генератора микроконтроллера генератор сторожевого таймера будет по-прежнему использоваться для сторожевого таймера и для определения истечения времени сброса.

Таблица 3.9 – Рабочие режимы внутреннего калиброванного RC генератора

CKSEL3–CKSEL0	Номинальная частота, МГц
0001*	1,0
0010	2,0
0011	4,0
0100	8,0

* Микросхема поставляется с активированной данной опцией.

Если выбирается этот генератор, то времена запуска определяются битами SUT, как это показано в таблице 3.10. BQ1 и BQ2 следует оставить неподключенными.

В приложении Б приведены значения частоты RC генератора в диапазоне температур и напряжений.

Таблица 3.10 – Времена запуска для выбора внутреннего калиброванного RC генератора

SUT1, SUT0	Время запуска из режима микропотребления и режима хранения	Дополнительная задержка от сброса, мс ($U_{\#VCC}$ = 5,0 В) / ($U_{\#VCC}$ = 3,3 В)	Рекомендуемый вид применения
00	6CK	–	BOD разрешено
01	6CK	8,2 / 8,6	Быстро нарастающее питание
10*	6CK	130 / 138	Медленно нарастающее питание
11	Зарезервировано		
* Микросхема поставляется с уже выбранной данной опцией.			

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	Значение калибровки для данного прибора								

Разряды 7–0: CAL7–CAL0 – значение калибровки генератора

Запись байта калибровки в этот адрес скорректирует внутренний генератор для устранения разброса частоты генератора. Во время сброса калибрационное значение 1 МГц автоматически загружается в регистр OSCCAL. Для работы внутреннего RC генератора на других частотах, значения калибровки должны загружаться вручную. Это можно реализовать, сохранив значения калибровки во флэш-памяти или ЭСППЗУ. Затем данное значение можно считать программным способом и загрузить в регистр OSCCAL.

Если OSCCAL находится в состоянии 0x00, то выбирается самая низкая из частот, имеющихся в наличии. Запись не нулевых значений в этот регистр увеличит частоту внутреннего генератора. Запись 0xFF в регистр даст самую высокую из имеющихся в наличии частот. Для синхронизации доступа в ЭСППЗУ и флэш-память используется калиброванный генератор. Если выполняется запись в ЭСППЗУ и флэш-память, то не следует выполнять калибровку при значениях частоты, более 10 % от номинальной. В противном случае может не произойти запись в ЭСППЗУ и флэш-память. Необходимо обратить внимание на то, что генератор предназначен для калибровки при 1; 2; 4 или 8 МГц. Как видно из таблицы 3.11, настройка для других значений не гарантируется.

Таблица 3.11 – Диапазон частот для внутреннего RC генератора

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

3.3.5 Внешний тактовый генератор

Для работы микроконтроллера от внешнего источника тактового сигнала, BQ1 должен быть управляем, как это показано на рисунке 3.12. Для работы микроконтроллера от внешнего тактового генератора необходимо запрограммировать конфигурационные биты

CKSEL в состояние «0000». С помощью программирования бита CKOPT пользователь может подключить внутреннюю емкость 36 пФ между BQ1 и #0V.

Рисунок 3.12 – Конфигурация при тактировании от внешнего генератора

Таблица 3.12 – Времена запуска для выбора внешнего синхронизирующего генератора

SUT1, SUT0	Время запуска из режима микропотребления и режима хранения	Дополнительная задержка от сброса, мс ($U_{\#VCC}$ = 5,0 В) / ($U_{\#VCC}$ = 3,3 В)	Рекомендуемый вид применения
00	6CK	–	BOD разрешено
01	6CK	8,2 / 8,6	быстро нарастающее питание
10	6CK	130 / 138	медленно нарастающее питание
11	Зарезервировано		

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

3.3.6 Генератор таймера/счетчика

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

3.4 Энергосберегающие режимы

Использование спящих режимов позволяет отключать неиспользуемые модули в микроконтроллере, благодаря чему экономится энергия. ИС 1887ВЕ4У обеспечивает различные спящие режимы, которые позволяют пользователю регулировать потребление мощности для различных видов применения.

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

Регистр управления МПУ – MCUCR

Регистр управления МПУ содержит разряды управления для регулирования энергопотребления.

Бит	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	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	

Разряды 7, 5, 4: SM2–SM0 – выбирают режим энергосбережения

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

Таблица 3.13 – Выбор спящего режима

SM2	SM1	SM0	Режим энергосбережения
0	0	0	Режим холостого хода
0	0	1	Снижение шума АЦП
0	1	0	Режим микропотребления
0	1	1	Режим хранения
1	0	0	Зарезервировано
1	0	1	Зарезервировано
1	1	0	Режим ожидания*
1	1	1	Расширенный режим ожидания*

* Режим ожидания и расширенный режим ожидания могут быть реализованы только с использованием внешних кварцевых генераторов или резонаторов.

Разряд 6: SE – разрешение спящего режима

В разряд SE должна быть записана логическая единица для того, чтобы МПУ вошло в спящий режим во время исполнения команды SLEEP. Чтобы избежать перехода МПУ в спящий режим до тех пор, пока это не предусмотрено программой, рекомендуется установить разряд Sleep Enable (SE) – разрешение на спящий режим непосредственно перед исполнением команды SLEEP и сразу же очистить ее после пробуждения.

Таблица 3.14 – Активные области тактирования и источники активации в различных режимах спящего состояния

	Активные области тактирования					Генераторы		Источники активации					
Режим ожидания	clk_{CPU}	clk_{FLASH}	$clk_{I/O}$	clk_{ADC}	clk_{ASY}	Основной источник синхронизации активен	Генератор таймера активен	INT2, INT1, INT0	TWI	TIMER	SPM	ADC	Other
Холостой ход			X	X	X	X	X[2)]	X	X	X	X	X	X
Снижение шума АЦП				X	X	X	X[2)]	X[3)]	X	X	X	X	
Микропотребление								X[3)]	X				
Хранение					X[2)]		X[2)]	X[3)]	X	X[2)]			
Ожидание[1)]						X		X[3)]	X				
Длительное ожидание[1)]					X[2)]	X	X[2)]	X[3)]	X	X[2)]			

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

[2)] Если установлен разряд AS2 в ASSR.

[3)] Только INT2 или прерывание уровня INT1 и INT0.

3.4.1 Режим холостого хода

Если в разряды SM2–SM0 записывается «000», то команда SLEEP заставляет микроконтроллер переходить в режим холостого хода, останавливая ЦПУ, но разрешая при этом, чтобы SPI, UART, аналоговый компаратор, АЦП, двухпроводной последовательный интерфейс, таймеры/счетчики, сторожевой таймер и система прерывания продолжали работать. Подобный спящий режим в основном останавливает clk_{CPU} и clk_{FLASH}, позволяя при этом работать другим тактовым сигналам.

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

3.4.2 Режим снижения шума АЦП

Если в разряды SM2–SM0 записывается «001», то команда SLEEP переводит микроконтроллер в режим снижения шума АЦП, останавливая при этом ЦПУ, но разрешая продолжение работы АЦП, внешних прерываний, слежения за адресом двухпроводного последовательного интерфейса, таймера/счетчика 2 и сторожевого таймера (при разрешении его работы). Этот спящий режим в основном останавливает $clk_{I/O}$, clk_{CPU} и clk_{FLASH}, позволяя при этом работать другим тактовым сигналам.

Это уменьшает внешние шумы АЦП, разрешая выполнять измерения с повышенной разрешающей способностью. Если АЦП включен, то преобразование начинается автоматически при вводе данного режима. Кроме прерывания от завершенного преобразования АЦП, только внешний сброс, сброс сторожевого таймера, сброс от схемы контроля за снижением питания (Brown-out Reset), прерывание при совпадении адреса двухпроводного последовательного интерфейса, прерывание таймера/счетчика 2, прерывание готовности SPM/EEPROM, внешние прерывания по уровню на INT0 или INT1, или внешнее прерывание на INT2 могут вывести МПУ из режима понижения шума АЦП.

3.4.3 Режим микропотребления

Если в разряды SM(2–0) записывается «010», то команда SLEEP переводит МК в режим микропотребления (Power Save). В этом режиме внешний генератор останавливается, в то время как внешние прерывания, слежение за совпадением адреса двухпроводного последовательного интерфейса и сторожевой таймер (если разрешена его работа) продолжают работать. Только внешний сброс, сброс сторожевого таймера, сброс от схемы контроля за снижением питания (Brown-out Reset), прерывание при совпадении адреса двухпроводного последовательного интерфейса, внешние прерывания по уровню INT0 или INT1 или внешнее прерывание для INT2 могут активировать МК. Этот спящий режим в основном блокирует все генерируемые синхронизирующие сигналы, разрешая работать только асинхронным модулям.

Обратите внимание на то, что если прерывание, срабатывающее по уровню, используется для пробуждения из режима микропотребления, то измененный уровень должен удерживаться в течение некоторого времени для того, чтобы вызвать активацию МК.

При пробуждении из режима пониженной мощности имеет место задержка, пока микроконтроллер перейдет в активный режим. Это позволяет перезапустить тактовый генератор и стабилизировать его работу после остановки. Период пробуждения определяется теми же конфигурационными битами CKSEL, которые определяют период сброса.

3.4.4 Режим хранения

Если «011» записывается в разряды SM2–SM0, то команда SLEEP переводит МК в режим хранения (Power Down). Этот режим аналогичен режиму микропотребления за одним исключением: если таймер/счетчик 2 тактируются асинхронно, т. е. устанавливается разряд AS2 в ASSR, то таймер/счетчик 2 будет продолжать работать во время «сна». Прибор может пробудиться или от события переполнения таймера или сравнения выхода от таймера/счетчика 2, если в TIMSK установлены соответствующие разряды разрешения прерывания, а в SREG установлен разряд разрешения глобального прерывания.

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

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

3.4.5 Режим ожидания

Если разряды SM2–SM0 содержат «110» и выбирается опция внешнего кварцевого генератора/резонатора, то команда SLEEP переводит МК в режим ожидания. Этот режим аналогичен режиму хранения, за исключением того, что генератор продолжает работать.

Устройство активируется из режима Standby в течение шести тактовых циклов.

3.4.6 Режим длительного ожидания

Если разряды SM2–SM0 содержат «111» и выбирается режим внешнего кварцевого генератора/резонатора, то команда SLEEP переводит МК в режим длительного ожидания. Этот режим аналогичен режиму хранения, за исключением того, что генератор продолжает работать.

Устройство активируется из режима длительного ожидания в течение шести тактовых циклов.

Минимизация потребления мощности

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

Аналогово-цифровой преобразователь

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

Аналоговый компаратор

Если аналоговый компаратор не используется, то при переходе в режим холостого хода он должен быть отключен. Если АЦП вводится в режим пониженного шума, то аналоговый компаратор должен быть отключен. В других спящих режимах аналоговый компаратор отключается автоматически. Однако если аналоговый компаратор настраивается для использования внутреннего опорного напряжения как входного, то аналоговый компаратор должен быть отключен во всех спящих режимах. В других случаях, внутренний источник опорного напряжения будет включен независимо от спящего режима.

Детектор отключения питания

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

Внутреннее опорное напряжение

Внутреннее опорное напряжение будет включено, когда это необходимо для детектора отключения питания, аналогового компаратора или АЦП. Если эти модули отключаются, как это описано в разделах выше, то внутреннее опорное напряжение будет отключено и мощность не будет потребляться. При повторном включении пользователь должен позволить источнику опорного напряжения запуститься перед тем, как будет использовано его выходное значение. Если источник опорного напряжения поддерживается включенным в спящем режиме, то выход можно использовать сразу после пробуждения.

Сторожевой таймер

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

Выводы портов

При переходе в спящий режим все выводы портов должны получить такую конфигурацию, для которой потребление мощности минимально. В этом случае самым важным является гарантия, что выводы не имеют резистивную нагрузку. В спящих режимах, когда

оба тактовых сигнала $clk_{I/O}$ и clk_{ADC} останавливаются, входные буферные устройства прибора отключаются. Тем самым гарантируется отсутствие потребления мощности входной логикой, когда в этом нет необходимости. В некоторых случаях входная логика необходима для обнаружения условий пробуждения и в таком случае она будет разрешена. Если входной буфер разрешен и входной сигнал остается в плавающем состоянии или если уровень аналогового сигнала близок к $U_{\#VCC}/2$, то входной буфер будет потреблять избыточную мощность.

3.5 Сброс и управление системой

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

ИС 1887ВЕ4У имеет четыре источника сброса:

- сброс включения питания. МК сбрасывается, если напряжение питания ниже порога сброса включения питания (V_{POT});
- внешний сброс. МК сбрасывается, если присутствует низкий уровень на выводе RESET# в течение времени, превышающего минимальную длину импульса;
- сброс сторожевого таймера. МК сбрасывается по истечению периода сторожевого таймера и когда сторожевой таймер активируется;
- сброс отключения питания. МК сбрасывается, если напряжение питания $U_{\#VCC}$ ниже порогового значения (V_{BOT}), и устройство обнаружения отключения активировано.

Замечание

После окончания периода time-out задержки сброса напряжение питания микроконтроллера должно находиться в рабочем диапазоне: от 4,5 до 5,5 В либо от 3,0 до 3,6 В. 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.

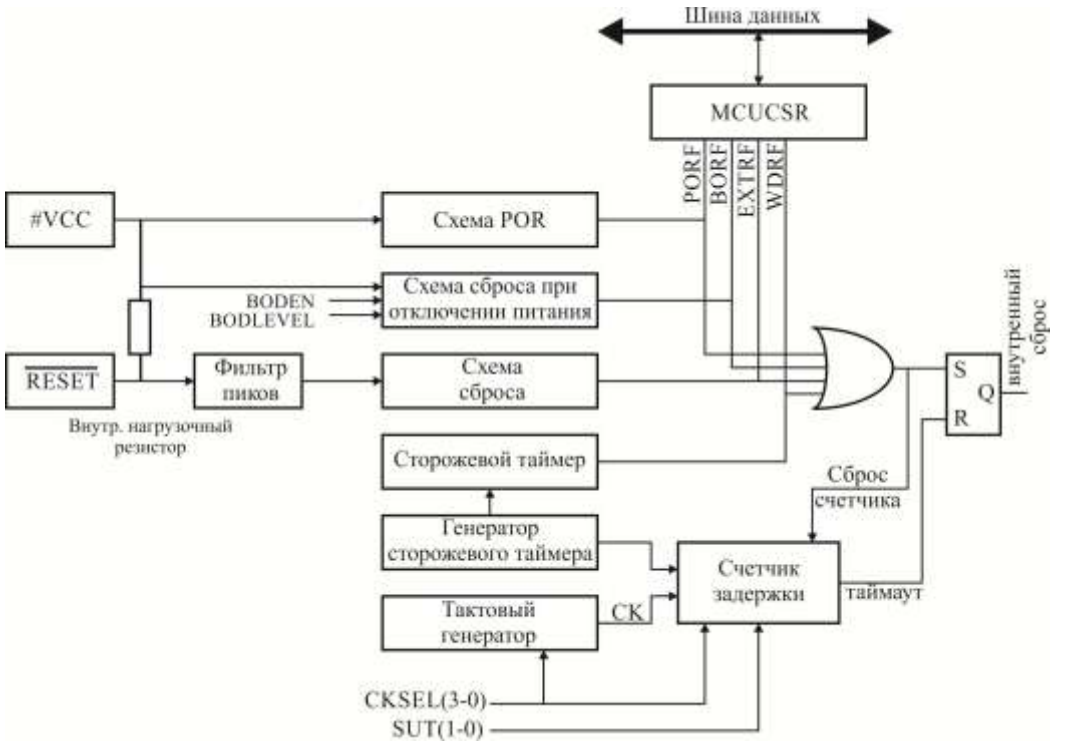


Рисунок 3.13 – Логика сброса

Таблица 3.15 – Характеристики переустановки *

Символ	Параметр	Условие	Мин.	Тип.	Макс.	Единица измерения
U_{POT}	Начальный сброс, пороговый уровень (передний фронт)		-	1,4	2,3	В
	Начальный сброс, пороговый уровень (задний фронт)		-	1,3	2,3	В
U_{RST}	Пороговое напряжение на выводе общего сброса		0,1		0,9	В
t_{RST}	Минимальная длина импульса сброса		-	50	-	нс
U_{BOT}	Пороговое напряжение выключения сброса	BODLEVEL = 1	2,5	2,7	3,2	В
		BODLEVEL = 0	3,7	4,0	4,2	
t_{BOD}	Минимальная длительность подачи низкого уровня для отключения	BODLEVEL = 1	-	2,0		мкс
		BODLEVEL = 0	-	2,0	-	мкс
U_{HYST}	Задержка отключения		-	130	-	мВ

Примечания

1 Сброс по включению питания не будет срабатывать, пока напряжение питания не снизится до величины ниже U_{POT} (спадающее).

2 Для некоторых приборов U_{BOT} может оказаться ниже минимального рабочего напряжения. Для тех приборов, для которых это имеет место, во время технологического контроля измерения производятся вплоть до $U_{\#VCC} = U_{BOT}$. Тем самым гарантируется, что сброс при внезапном отключении питания произойдет до того, как $U_{\#VCC}$ упадет до уровня напряжения, при котором правильная работа микроконтроллера не может быть больше гарантирована.

* Данные значения являются справочными.

3.5.1 Сброс по включению питания

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

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

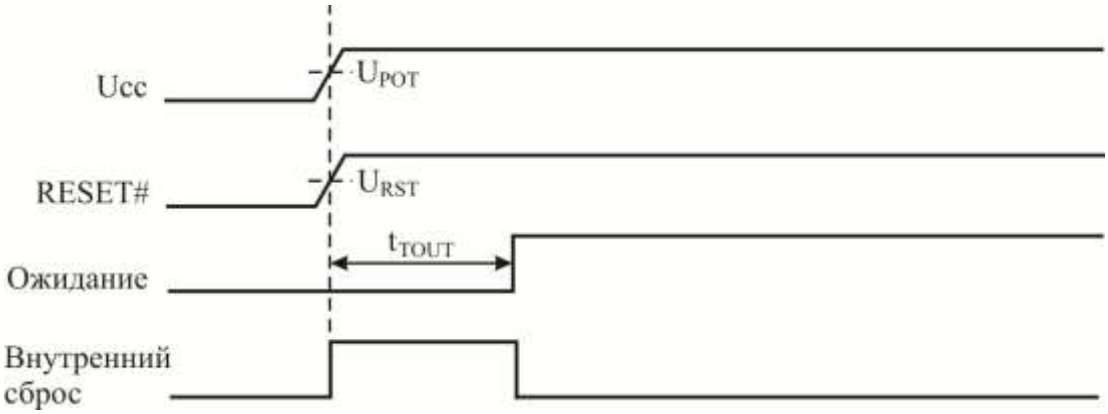


Рисунок 3.14 – Запуск МК (команда RESET связана с уровнем $U_{\#VCC}$)

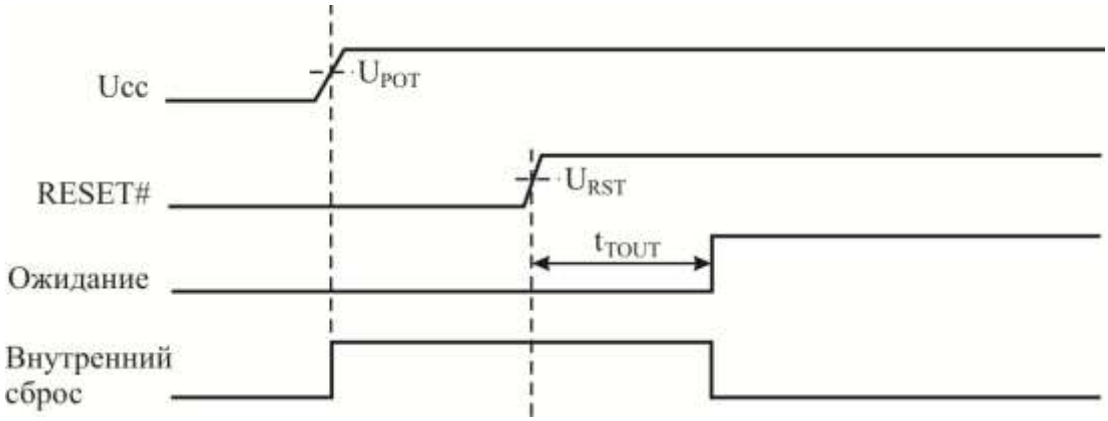


Рисунок 3.15 – Запуск МКУ (команда RESET продлена
с помощью внешнего воздействия)

3.5.2 Внешний сброс

Внешний сброс запускается при низком логическом уровне на выводе RESET#. Импульсы сброса, превышающие минимальную ширину импульса, см. таблицу 3.15, активируют сброс, если даже тактовый генератор не работает. При более коротких импульсах не гарантируется активация сброса. Когда приложенный сигнал достигает уровня порогового напряжения сброса – U_{RST} по своему положительному фронту, то счетчик задержки запускает микроконтроллер после окончания периода t_{TOUT}.

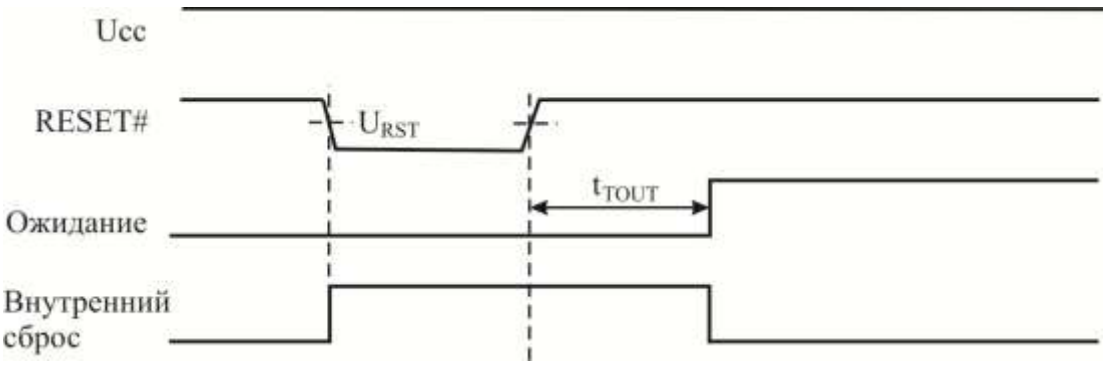


Рисунок 3.16 – Внешний сброс во время работы

3.5.3 Детектирование отключения питания

ИС 1887ВЕ4У имеет встроенную схему детектирования отключения питания (BOD) для контроля уровня $U_{\#VCC}$ во время работы, что реализуется путем его сравнения с фиксированным уровнем. Требуемый уровень для BOD можно выбрать с помощью бита BODLEVEL, чтобы он был равен 2,7 В (BODLEVEL не запрограммирован (unprogrammed) или 4,0 В (BODLEVEL запрограммирован (programmed). Уровень порога имеет гистерезис для обнаружения отключения питания без наличия скачков напряжения. Гистерезис для порогового уровня должен интерпретироваться как:

$U_{BOT+} = U_{BOT} + U_{HYST/2}$,
$U_{BOT-} = U_{BOT} - U_{HYST/2}$.

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

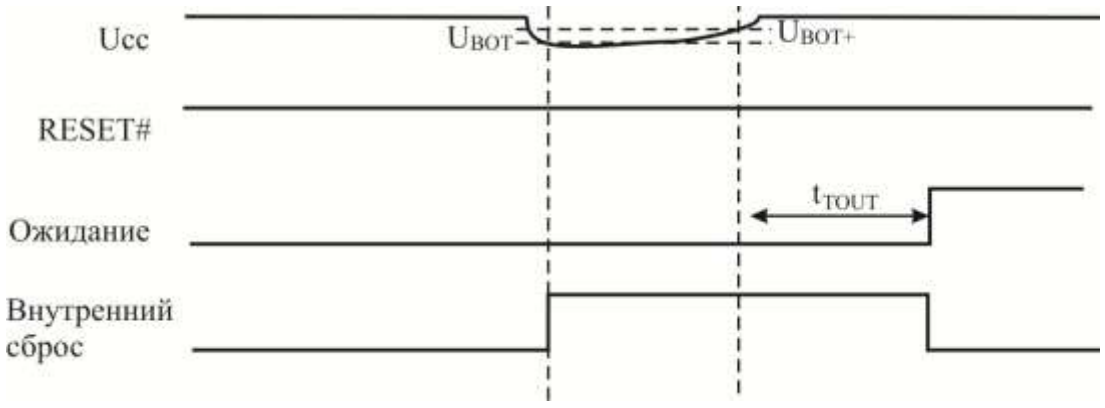


Рисунок 3.17 – Сброс по отключению питания во время работы

В диапазоне напряжения питания, равном 3,3 В ± 10 %, недопустимо использование внутренней схемы детектирования отключения питания (BOD). Для этих целей рекомендуется использовать внешний монитор питания.

3.5.4 Сброс сторожевого таймера

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

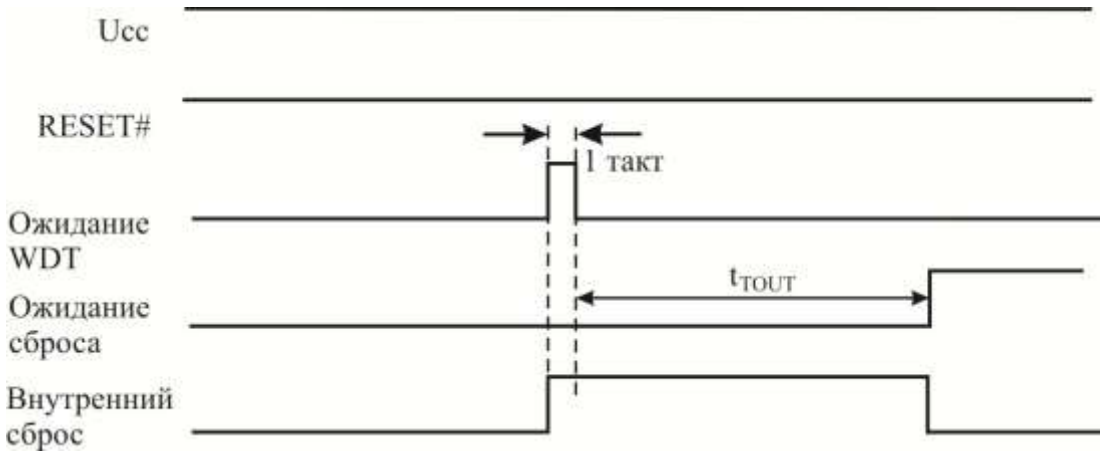


Рисунок 3.18 – Сброс от сторожевого таймера во время работы

Контроль и состояние МК

Регистр – MCUCSR

Регистр управления и статуса МК выдает информацию о том, какой из источников сброса вызвал сброс МК.

Бит	7	6	5	4	3	2	1	0	
	-	SE	-	-	WDRF	BORF	EXTRF	PORF	MCUCSR
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 3: WDRF – флаг сброса сторожевого таймера

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

Разряд 2: BORF – флаг сброса отключения питания

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

Разряд 1: EXTRF – флаг внешнего сброса

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

Разряд 0: PORF – флаг сброса по включению питания

Этот флаг устанавливается в том случае, если происходит сброс по включению питания. Флаг сбрасывается только с помощью записи логического нуля в разряд флага.

Для того, чтобы использовать флаги сброса для идентификации условия сброса, пользователь должен прочитать, а затем сбросить MCUCSR в программе как можно рань-

ше. Если регистр очищается до того, как происходит другой сброс, источник сброса можно найти путем анализа флагов сброса.

Внутреннее опорное напряжение

ИС 1887ВЕ4У содержит внутренний источник опорного напряжения. Уровень источника опорного напряжения используется для обнаружения отключения питания, и он может быть использован в качестве входа для аналогового компаратора или АЦП. Опорное напряжение 2,56 В для АЦП генерируется внутренним источником опорного напряжения.

Разрешающие сигналы опорного напряжения и время запуска

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

1 Когда запускается BOD (путем программирования перемычки BODEN).

2 Когда источник опорного напряжения подсоединен к аналоговому компаратору (с помощью установки разряда ACBG в ACSR).

3 Когда работает АЦП.

Если BOD не активирован, то после установки разряда ACBG или активации АЦП пользователь всегда должен разрешить запуск источника опорного напряжения до того, как будет использован выход аналогового компаратора или АЦП. Для снижения потребления мощности в режиме хранения, пользователь может исключить три вышеуказанных случая, в которых используется источник опорного напряжения, чтобы отключить опорное напряжение до того, как запустится режим хранения.

Т а б л и ц а 3.16 – Характеристики внутреннего опорного напряжения

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

3.6 Сторожевой таймер

Сторожевой таймер синхронизируется от отдельного встроенного генератора, работающего на частоте 500 кГц ± 10 %. Это типовое значение при $U_{\#VCC}$ = 5,0 В, $U_{\#VCC}$ = 3,3 В. Благодаря использованию предделителя сторожевого таймера можно регулировать интервал его сброса. Сторожевой таймер сбрасывается с помощью команды WDR – сброс сторожевого таймера. Сторожевой таймер сбрасывается также в случае его отключения и когда происходит общий сброс. Для определения периода сброса можно выбрать восемь различных периодов. Если период сброса истечет без повторного сброса сторожевого таймера, то ИС 1887ВЕ4У выполняет сброс и обеспечивает выполнение программы от вектора сброса.

Для предотвращения непреднамеренного отключения сторожевого таймера или непреднамеренного изменения периода прерывания с помощью бит конфигурации S8535C и WDTON выбираются три различных уровня безопасности, как это показано в таблице 3.17. WDT можно перевести в любой уровень безопасности.

Т а б л и ц а 3.17 – Конфигурация команды WDT в зависимости от установок для бит S8535C и WDTON

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

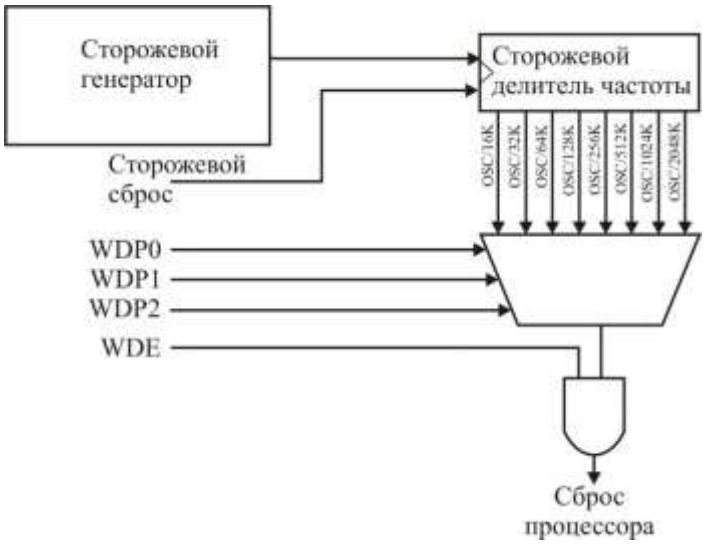


Рисунок 3.19 – Сторожевой таймер

Бит	7	6	5	4	3	2	1	0	
	-	-	-	WDCE	WDE	WDP2	WDP1	WDP0	MCUCR
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: резервные разряды

Эти разряды являются резервными разрядами для ИС 1887ВЕ4У, они всегда будут читаться как ноль.

Разряд 4: WDCE – разрешение изменения периода сторожевого таймера

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

Разряд 3: WDE – разрешение работы сторожевого таймера

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

1 Необходимо единовременно записать логическую единицу в WDCE и WDE. Логическая единица должна быть записана в разряд WDE, даже если он установлен до начала операции отключения.

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

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

Разряды 2–0: WDP2, WDP1, WDP0 – биты предделителя сторожевого таймера 2, 1 и 0

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

Т а б л и ц а 3.18 – Выбор предделителя сторожевого таймера

WDP2	WDP1	WDP0	Количество циклов осциллятора WDT	Типичный период WDT*
0	0	0	16K (16384)	32,6 мс
0	0	1	32K (32786)	65,0 мс
0	1	0	64K (65536)	130 мс
0	1	1	128K (131072)	0,26 с
1	0	0	256K (262144)	0,52 с
1	0	1	512K (524288)	1,04 с
1	1	0	1024K (1048576)	2,0 с
1	1	1	2048K (2097152)	4,2 с

* Данные значения являются ориентировочными. Разброс частоты встроенного генератора в диапазоне рабочих температур и частот составляет до 15 %.

Ниже приведена функция, реализованная на ассемблере, и функция, реализованная на С, для отключения WDT. Данный пример предполагает, что прерывания контролируются (путем глобального отключения прерываний) таким образом, чтобы они были блокированы во время исполнения этих функций.

Ассемблерный код:

```
WDT_off:
; Reset WDT
wdr
; Write logical one to WDCE and WDE
in r16, WDTCR
ori r16, (1<<WDCE)|(1<<WDE)
out WDTCR, r16
; Turn off WDT
ldi r16, (0<<WDE)
out WDTCR, r16
ret
```

С код:

```
void WDT_off(void)
(
/* Reset WDT */
_WDR()
/* Write logical one to WDCE and WDE */
WDTCR |= (1<<WDCE) | (1<<WDE);
/* Turn off WDT */
WDTCR = 0x00;
)
```

Временные последовательности для изменения конфигурации сторожевого таймера

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

Уровень безопасности 0

Таймер устройства первоначально не активирован, но его можно запустить, установив разряд WDE. Для отключения работающего таймера и/или изменения периода до сброса, необходимо соблюдать следующую последовательность:

1 Необходимо единовременно записать логическую единицу в разряд WDCE и WDE. Логическая единица должна быть записана в разряд WDE независимо от первоначального значения разряда WDE.

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

Уровень безопасности 1

В этом режиме сторожевой таймер сначала отключен, но можно его включить, установив разряд WDE. При изменении периода до сброса или отключении работающего таймера необходима соответствующая временная последовательность. Для отключения работающего таймера и/или изменения периода до сброса, необходимо выполнить следующие процедуры:

1 Необходимо единовременно записать логическую единицу в разряд WDCE и WDE. Логическая единица должна быть записана в WDE независимо от первоначального значения разряда WDE.

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

Уровень безопасности 2

В этом режиме таймер всегда включен, и разряд WDE будет всегда считываться как единица. Для изменения периода до сброса таймера необходимо выполнить следующие процедуры:

1 Необходимо единовременно записать логическую единицу в разряд WDCE и WDE. Логическая единица должна быть записана в разряд WDE независимо от первоначального значения разряда WDE. Хотя WDE всегда считается установленным, в разряд WDE необходимо записать логическую единицу для запуска временной последовательности.

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

3.7 Прерывания

В данном подразделе описываются особенности обращения с прерываниями, выполняемыми в микросхеме 1887ВЕ4У.

Векторы прерывания в ИС 1887ВЕ4У

Т а б л и ц а 3.19 – Векторы сброса и прерывания

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

П р и м е ч а н и я

1 При программировании бита BOOTRST МК при сбросе переходит по адресу, указанному в векторе сброса в секцию загрузчика.

2 При установке разряда IVSEL в GICR векторы прерываний будут перемещены в начало области загрузчика флэш-памяти (Boot Flash). Адрес каждого вектора прерывания станет затем адресом в данной таблице, просуммированным с исходным адресом раздела Boot Flash.

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

Таблица 3.20 – Размещение векторов сброса и прерываний

BOOTRST [1)]	IVSEL	Адрес сброса	Стартовый адрес векторов прерывания
1	0	0x0000	0x0000
1	1	0x0000	Адрес сброса загрузчика + 0x0001
0	0	Адрес сброса загрузчика	0x0000
0	1	Адрес сброса загрузчика	Адрес сброса загрузчика + 0x0001
[1)] Для бита BOOTRST значение «1» означает «не запрограммировано», а значение «0» означает «запрограммировано».			

Ниже представлена наиболее типичная и общая программная установка для векторов сброса и прерываний в ИС 1887ВЕ4У.

```
Адрес          Код      Источник              Комментарий
0x000          rjmp     RESET                 Обработчик сброса
0x001          rjmp     EXT_INT0              IRQ0 обработчик
0x002          rjmp     EXT_INT1              IRQ1 обработчик
0x003          rjmp     TIM2_COMP             Обработчик сравнения таймера 2
0x004          rjmp     TIM2_OVF              Обработчик переполнения таймера 2
0x005          rjmp     TIM1_CAPT             Обработчик захвата таймера 1
0x006          rjmp     TIM1_COMPA            Обработчик сравнения A таймера 1
0x007          rjmp     TIM1_COMPB            Обработчик сравнения B таймера 1
0x008          rjmp     TIM1_OVF              Обработчик переполнения таймера 2
0x009          rjmp     TIM0_OVF              Обработчик переполнения таймера 0
0x00A          rjmp     SPI_STC               Обработчик завершения переноса SPI
0x00B          rjmp     UART_RXC              Обработчик завершения UART RX
0x00C          rjmp     UART_UDRE             Обработчик освобождения UDR
0x00D          rjmp     UART_TXC              Обработчик завершения UART TX
0x00E          rjmp     ADC                   Обработчик завершения преобразования АЦП
0x00F          rjmp     EE_RDY                Обработчик готовности ЭСППЗУ
0x010          rjmp     ANA_COMP              Обработчик аналогового компаратора
0x011          rjmp     TWSI                  Обработчик двухпроводного последовательного
                                              интерфейса
0x012          rjmp     EXT_INT2              Обработчик IRQ2
0x013          rjmp     TIM0_COMP             Обработчик таймер 0
0x014          rjmp     SPM_RDY               Обработчик готовности SPM
;
0x015 RESET:   ldi      r16, high (RAMEND)    Запуск основной программы
0x016          out      SPH, r16              Установка указателя стека в конец области ОЗУ
0x017          ldi      r16, low (RAMEND)
0x018          out      SPL, r16
0x019          sei                            Разрешение прерываний
0x020          <instr>  xxx
...            ...      ...
```

Если бит BOOTRST не запрограммирован, то размер секции загрузчика устанавливается равным 2 Кбайта и разряд IVSEL в регистре GICR устанавливается до того, как будут разрешены любые прерывания, при этом наиболее типичная и общая программная установка для векторов прерываний и сброса выглядит следующим образом:

```
Адрес           Код                                 Комментарий
0x000 RESET:    ldi     r16, high (RAMEND)          Запуск основной программы
0x001           out     SPH, r16                    Установка указателя стека в конец области ОЗУ
0x002           ldi     r16, low (RAMEND)
0x003           out     SPL, r16
0x004           sei     TIM2_OVF                    Разрешение прерываний
0x005           <instr> xxx
;
.org 0xC01
0xC01           rjmp    EXT_INT0                    Обработчик IRQ0
0xC01           rjmp    EXT_INT1                    Обработчик IRQ1
…               …       …
0xC14           rjmp    SPM_RDY                     Обработчик готовности SPM
```

Если бит BOOTRST запрограммирован и размер секции загрузчика составляет 2 Кбайта, то наиболее типичная и общая программная установка для векторов прерываний и сброса выглядит следующим образом:

```
Адрес           Код                                 Комментарий
.org 0x001
0x001           rjmp    EXT_INT0                    Обработчик IRQ0
0x002           rjmp    EXT_INT1                    Обработчик IRQ1
..... .         ..
0x014           rjmp    SPM_RDY                     Обработчик готовности SPM
;
.org 0xC00
.org 0xC01
0xC00 RESET:    ldi     r16, high (RAMEND)          Обработчик готовности SPM
0xC01           out     SPH, r16                    Установка указателя стека в конец области ОЗУ
0xC02           ldi     r16, low(RAMEND)
0xC03           out     SPL, r16
0xC04           sei     Разрешение прерываний
0xC05           <instr> xxx
```

Если бит BOOTRST запрограммирован и размер секции загрузчика составляет 2 Кбайта и разряд IVSEL в регистре GICR устанавливается до того, как будут разрешены любые прерывания, то наиболее типичная и общая программная установка для векторов прерываний и сброса выглядит следующим образом:

```
Адрес           Код                                 Комментарий
.org 0xC00
0xC00           rjmp    RESET                       Обработчик Reset
0xC01           rjmp    EXT_INT0                    Обработчик IRQ0
0xC02           rjmp    EXT_INT1                    Обработчик IRQ1
... ... .       ..
0xC14           rjmp    SPM_RDY                     Обработчик готовности SPM
;
0xC15 RESET:    ldi     r16, high (RAMEND)          Запуск основной программы
0xC16           out     SPH, r16                    Установка указателя стека в конец области ОЗУ
0xC17           ldi     r16, high (RAMEND)
0xC18           out     SPH, r16
0xC19           ldi                                 Разрешение прерываний
0xC20           <instr> xxx
```

Перемещение прерываний между секциями приложения и загрузчика

Основной регистр управления прерываниями управляет размещением таблицы векторов прерываний.

3.7.1 Основной регистр управления прерываниями GICR

Разряд 1: IVSEL – Выбор вектора прерывания

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

Для того чтобы избежать непреднамеренных изменений в таблице векторов прерываний, необходимо соблюдать специальную процедуру записи для изменения разряда IVSEL:

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

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

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

П р и м е ч а н и е – Если векторы прерывания размещаются в секции автоматического загрузчика программ и программируется разряд блокировки загрузки BLB02, то прерывания запрещены при выполнении из секции приложений. Если векторы прерывания размещаются в секции приложений и программируется разряд блокировки загрузки BLB12, то прерывания запрещены при исполнении из секции автоматического загрузчика программ.

Разряд 0: IVCE – Разрешение изменения вектора прерывания

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

Ассемблерный код:

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

С код:

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

3.8 Порты ввода-вывода

Все порты микросхемы имеют истинную функциональность чтение-модификация-запись, если они используются в качестве цифровых портов ввода-вывода общего назначения. Это означает, что направление одного вывода порта может быть изменено без изменения направления любого другого вывода с помощью инструкций SBI и CBI. То же самое имеет место при изменении состояния порта (если вывод порта конфигурируется в качестве выхода) или подключения/отключения нагрузочных резисторов (если вывод порта используется в качестве входа). Нагрузочной способности достаточно для управления светодиодными индикаторами. Все выводы портов имеют выбираемые отдельно нагрузочные резисторы с сопротивлением, не зависящим от напряжения питания. Все выводы порта имеют защитные диоды как для питания $U_{\#VCC}$, так и для «земли», как это показано на рисунке 3.20.

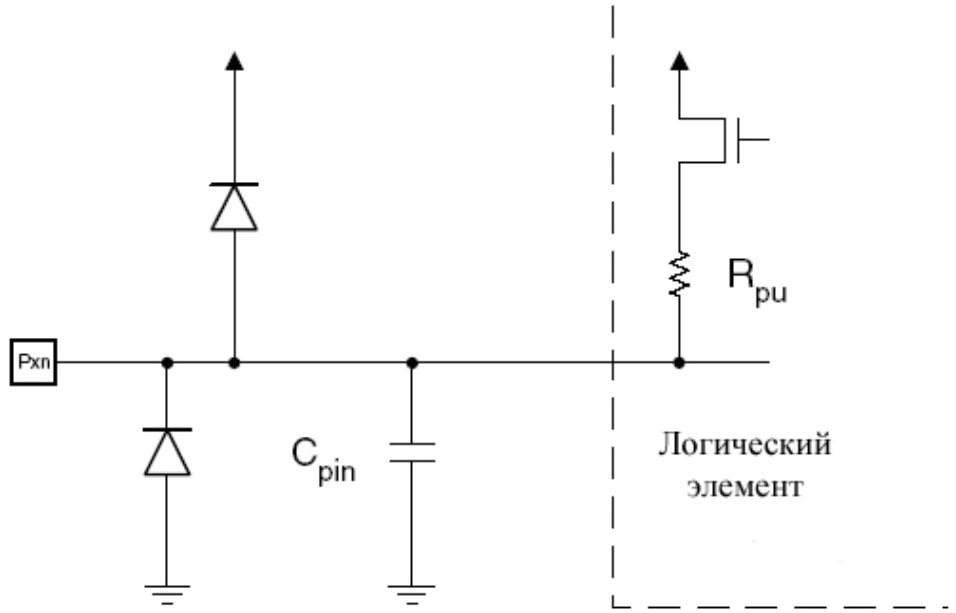


Рисунок 3.20 – Эквивалентная схема порта входа/выхода

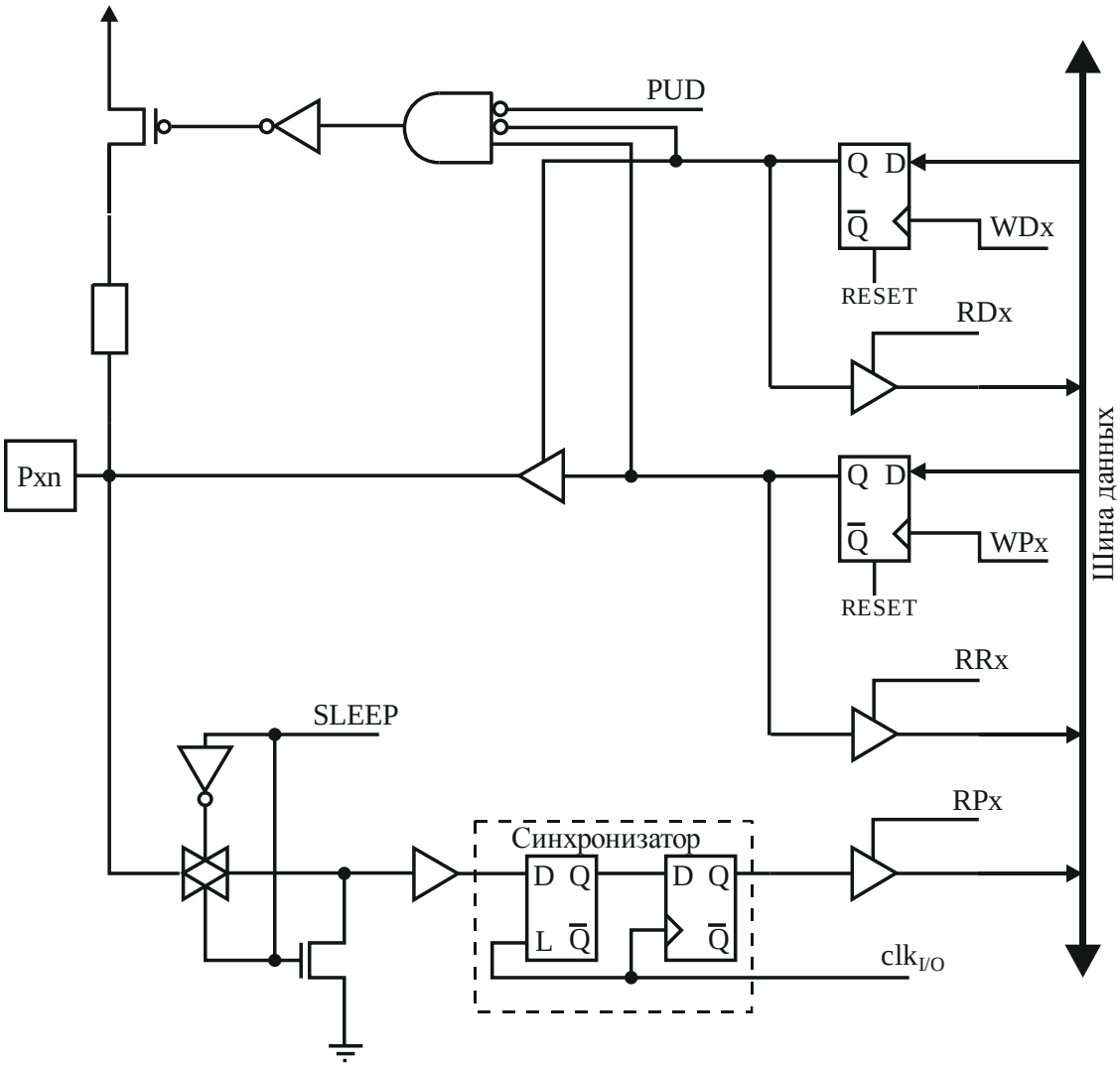
Все данные по регистрам и разрядам в данном подразделе написаны в общем виде. Буква «x» представляет собой обозначение порта, а буква «n» представляет собой номер разряда. Однако, при использовании определений регистра или разряда в программе, необходимо использовать точную форму. Например, PORTB3 для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

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

Большинство выводов портов мультиплексированы альтернативными функциями для периферийных функций устройства. Для ознакомления с полным описанием альтернативных функций необходимо обратиться к описаниям отдельных модулей и обратить внимание на то, что активация альтернативной функции какого-либо вывода порта не влияет на использование других выводов порта в качестве общего цифрового порта ввода-вывода.

3.8.1 Порты в качестве общих цифровых портов ввода-вывода

Порты являются двунаправленными портами ввода-вывода с дополнительной внутренней нагрузкой. На рисунке 3.21 показано функциональное описание одного вывода порта ввода-вывода, который в общем виде назван Pxn.



PUD – запрещение внутр. нагрузки
SLEEP – управление SLEEP
$clk_{I/O}$ – тактовый сигнал I/O

WDx – запись в DDRx
RDx – чтение DDRx
WPx – запись в порт
RRx – чтение регистра порта
RPx – чтение вывода порта

Рисунок 3.21 – Общий цифровой порт ввода-вывода

3.8.2 Конфигурация вывода

Каждый вывод порта состоит из трех элементов: DDxn, PORTxn и PINxn.
Разряд DDxn в регистре DDRx выбирает направление для данного вывода. Если в DDxn записывается логическая единица, то Pxn конфигурируется как выход. Если в DDxn записывается логический нуль, то Pxn конфигурируется как вход.

Если в PORTxn записывается логическая единица при конфигурировании порта в качестве входа, то нагрузочный резистор активируется. Для отключения нагрузочного рези-

стора в PORTxn должен быть записан логический нуль или же порт должен быть сконфигурирован как выход. Выводы порта переходят в состояние высокого импеданса, когда сброс становится активным, даже если не работают тактовые генераторы.

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

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

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

3.8.3 Считывание значения вывода

Независимо от разряда направления данных DDxn, вывод порта может считываться с помощью бита регистра PINxn. Как показано на рисунке 3.21, разряд регистра PINxn и предшествующая ему триггер-защелка образуют синхронизирующее устройство. Оно необходимо для того, чтобы избежать метастабильности, если физический вывод изменяет значение недалеко от фронта внутреннего синхронизирующего импульса, однако, это в свою очередь, вносит задержку. На рисунке 3.22 изображена временная диаграмма синхронизации при считывании значения вывода, прикладываемого извне. Максимальное и минимальное время задержки распространения обозначены, соответственно, $t_{pd,max}$ и $t_{pd,min}$.

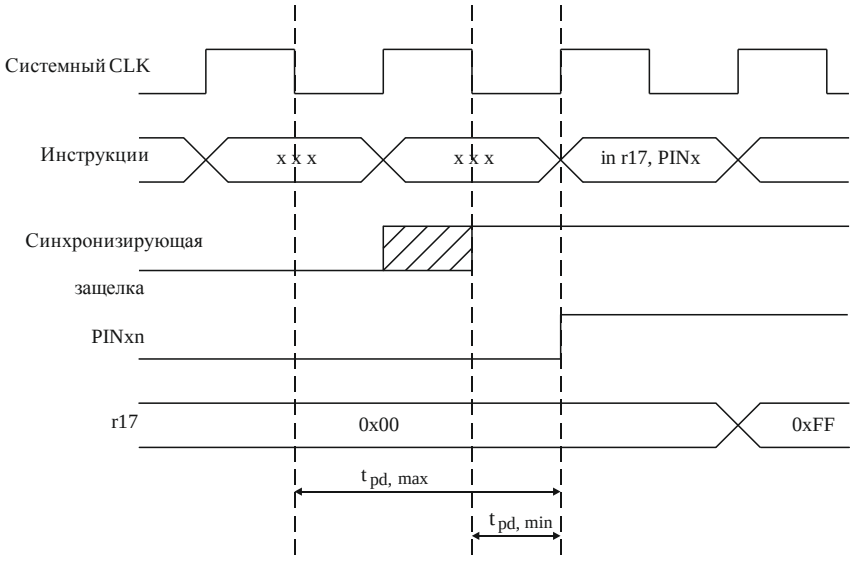


Рисунок 3.22 – Синхронизация при считывании значения вывода, определяемого извне

Рассмотрим тактовый период, начинающийся сразу же после заднего фронта импульса синхронизации системы. Защелка закрывается, когда тактовый сигнал находится в низком состоянии, и становится прозрачной, когда тактовый сигнал находится в высоком состоянии, как это показано в заштрихованной области сигнала «синхронизирующая защелка» на рисунке 3.22. Значение сигнала фиксируется, когда системный тактовый импульс переходит в низкое состояние. Он синхронизирован в регистр PINxn при последующем положительном фронте сигнала тактирования. Как указано двумя стрелками $t_{pd,max}$ и $t_{pd,min}$, переход одиночного сигнала на выводе будет задержан на величину от ½ до 1½ тактового периода в зависимости от времени установления. При считывании значения вывода, которое было определено программным способом, необходимо вставить команду «nop», как указано на рисунке 3.23. Команда «out» устанавливает сигнал «синхронизирующая защелка» к положительному фронту сигнала синхронизации. В этом случае задержка t_{pd} при прохождении через синхронизирующее устройство составляет один тактовый период.

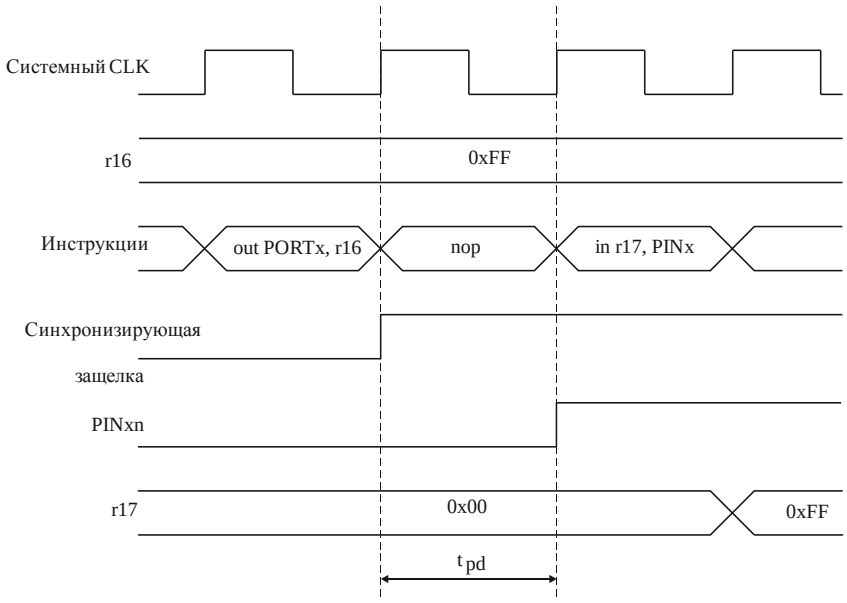


Рисунок 3.23 – Синхронизация при считывании значения вывода, определенного программно

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

Ассемблерный код:

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

С код:

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

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

3.8.4 Режим разрешения цифрового входа и спящий режим

Как показано на рисунке 3.21, входной цифровой сигнал может быть подсоединен к земле на входе триггера Шмитта. Сигнал, далее обозначенный SLEEP на рисунке 3.21, устанавливается контроллером спящего режима МК в режимах микропотребления, хранения, режиме ожидания и длительном режиме ожидания для того, чтобы избежать высокого потребления мощности, если некоторые входные сигналы оставлены как плавающие или имеют уровень аналогового сигнала близкий к $U_{\#VCC}/2$.

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

Если на выводе асинхронного внешнего прерывания, получившем конфигурацию «прерывание по переднему фронту, заднему фронту или по любому логическому изменению на выводе», присутствует высокий логический уровень («единица») в то время, когда внешнее прерывание не разрешено, то соответствующий флаг внешнего прерывания будет установлен при возобновлении работы после возвращения из вышеупомянутых спящих режимов, поскольку фиксирование в этих спящих режимах вызывает требуемое логическое изменение.

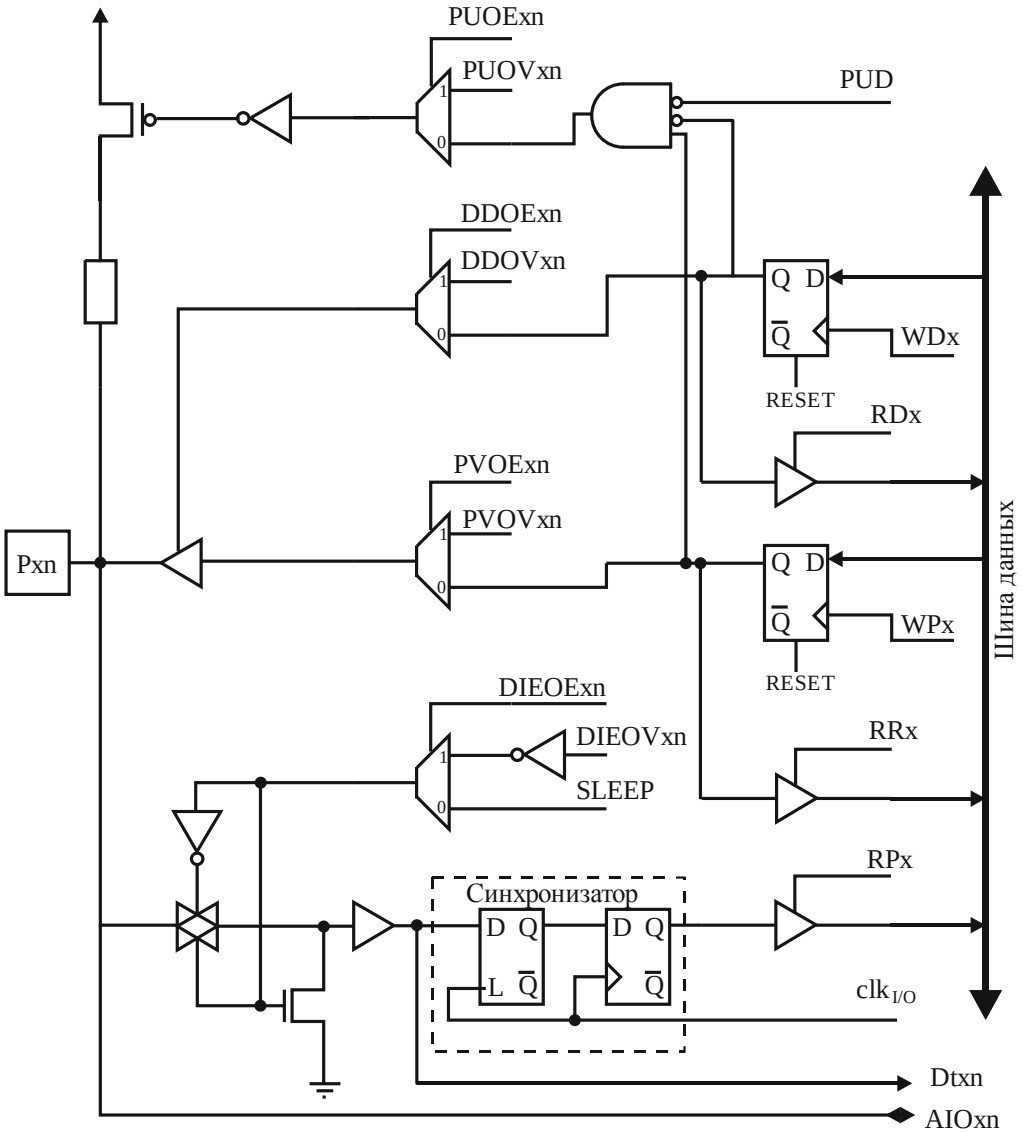
3.8.5 Неподсоединенные выводы

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

Самый простой метод обеспечения определенного уровня на неиспользуемых выводах заключается в подключении внутренней нагрузки. В этом случае внутренняя нагрузка будет отключаться во время сброса. Если существенным фактором является малая потребляемая мощность во время сброса, то рекомендуется использовать внешнее повышение или понижение напряжения. Не рекомендуется непосредственно подключать неиспользуемые входы к #VCC или #0V, поскольку это может вызвать повышенные токи, если случайно этот вывод будет сконфигурирован как выход.

3.8.6 Альтернативные функции портов

Большинство портов имеют альтернативные функции в дополнение к основным функциям цифровых входов/выходов. На рисунке 3.24 показано, каким образом сигналы управления выводом порта из упрощенной схемы на рисунке 3.21 могут быть переписаны альтернативными функциями. Переопределяемые сигналы могут не присутствовать на выводах всех портов, этот рисунок представляет собой общее описание.



PUOExn – разрешение переопределения внутр. нагрузки
PUOVxn – значение переопределения внутр. нагрузки
DDOExn – разрешение переопределения напрвления
DDOVxn – значение переопределения напрвления
PVOExn – разрешение переопределения значения
PVOVxn – переопределенное значение
DIEOExn – переопределение разрешения цифр. входа
DIEOVxn – переопределение значения цифр. входа
WDx – запись в DDRx
RDx – чтение DDRx
WPx – запись в порт
RRx – чтение регистра порта
RPx – чтение вывода порта
PUD – запрещение внутр. нагрузки
SLEEP – управление SLEEP
$clk_{I/O}$ – тактовый сигнал I/O

П р и м е ч а н и е – WPx, WDx, RRx, RPx и RDx являются общими для всех выводов внутри того же порта. $clk_{I/O}$, SLEEP и PUD являются общими для всех портов. Все остальные сигналы являются уникальными для каждого вывода.

Рисунок 3.24 – Альтернативные функции портов

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

Т а б л и ц а 3.21 – Общее описание переопределяемых сигналов для альтернативных функций

Наименование сигнала	Полное наименование	Описание
PUOE	Разрешение переопределения внутренней нагрузки	Если установлен этот сигнал, то разрешение внутренней нагрузки управляется с помощью сигнала PUOV. Если этот сигнал сбрасывается, то внутренняя нагрузка разрешается в том случае, когда (DDxn, PORTxn, PUD) = 0b010
PUOV	Переопределяемое значение внутренней нагрузки	Если устанавливается PUOE, то внутренняя нагрузка разрешается/запрещается, когда PUOV устанавливается /очищается независимо от установки DDxn, PORTxn и разрядов регистра PUD
DDOE	Разрешение переопределения направления данных	Если установлен этот сигнал, то разрешение выходного драйвера управляется сигналом DDOV. Если этот сигнал сбрасывается, то выходной драйвер активируется с помощью разряда регистра DDxn
DDOV	Переопределяемое значение направления данных	Если установлено значение DDOE, то выходной драйвер разрешается/запрещается, если устанавливается/сбрасывается DDOV независимо от установки разряда регистра DDxn
PVOE	Разрешение переопределения значения порта	Если установлен этот сигнал, и разрешен выходной драйвер, то значение порта управляется сигналом PVOV. Если PVOE сбрасывается, и разрешается выходной драйвер, то значение управляется разрядом выходного драйвера PORTxn
PVOV	Переопределяемое значение порта	Если установлено PVOE, то значение порта управляется сигналом PVOV, независимо от установки разряда регистра PORTxn
DIEOE	Разрешение переопределения функции разрешения цифрового входа	Если установлен этот разряд, то разрешение цифрового входа управляется сигналом DIEOV. Если этот сигнал сбрасывается, то разрешение цифрового входа определяется состоянием МК (нормальный режим, спящие режимы)
DIEOV	Переопределяемое значение разрешения цифрового входа	Если установлено DIEOE, то цифровой вход разрешается/ запрещается, если DIEOV устанавливается/сбрасывается независимо от состояния МК (нормальный режим, спящие режимы)
DI	Цифровой вход	Это цифровой вход для альтернативных функций. На схеме сигнал подводится к выходу триггера Шмитта, но перед синхронизирующим устройством. Если цифровой вход не используется в качестве цифрового входа, то модуль с альтернативной функцией будет использовать свое собственное синхронизирующее устройство
∩I/O	Аналоговый вход/выход	Это аналоговый вход/выход к альтернативным функциям и от них. Сигнал подсоединяется непосредственно к выводу и может быть использован двунаправленно

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

Регистр специальных функций – SFIOR

Бит	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	ADHSM	ACME	PUD	PSR2	PSR10	SFIOR
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 2: PUD – Отключение нагрузочных резисторов

Если в этот разряд записывается логическая единица, то нагрузочные сопротивления в портах входа/выхода отключаются, даже если регистры DDxn и PORTxn имеют конфигурацию, активирующую нагрузочные сопротивления ((DDxn, PORTxn) = 0b01).

3.8.7 Альтернативные функции порта A

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

Т а б л и ц а 3.22 – Альтернативные функции порта A

Разряды порта	Альтернативная функция
PA7	ADC7 (АЦП входной канал 7)
PA6	ADC6 (АЦП входной канал 6)
PA5	ADC5 (АЦП входной канал 5)
PA4	ADC4 (АЦП входной канал 4)
PA3	ADC3 (АЦП входной канал 3)
PA2	ADC2 (АЦП входной канал 2)
PA1	ADC1 (АЦП входной канал 1)
PA0	ADC0 (АЦП входной канал 0)

В таблицах 3.23 и 3.24 показана связь альтернативных функций порта A с сигналами корректировки.

Т а б л и ц а 3.23 – Переопределяющие сигналы для альтернативных функций в PA7–PA4

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

Т а б л и ц а 3.24 – Переопределяющие сигналы для альтернативных функций в PA3–PA0

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

3.8.8 Альтернативные функции порта B

Т а б л и ц а 3.25 – Альтернативные функции выводов порта B

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

Конфигурация альтернативных выводов приведена ниже.

SCK – порт B, бит 7

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

MISO – порт B, бит 6

MISO: вход данных ведущего, выход данных ведомого для канала SPI. При запуске SPI в качестве ведущего, этот вывод получает конфигурацию входа независимо от установок DDB6. При запуске SPI в качестве пого, направление данных для этого вывода контролируется DDB6. Если SPI переопределяет вывод на функцию входа, то нагрузочное сопротивление может по-прежнему управляться битом PORTB6.

MOSI – порт B, бит 5

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

SS# – порт B, бит 4

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

AIN1 / OC0 – порт B, бит 3

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

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

AIN0 / INT2 – порт B, бит 2

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

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

T1 – порт B, бит 1

T1: источник счетных импульсов таймера/счетчика 1.

T0/XCK – порт B, бит 0

T0: источник счетных импульсов таймера/счетчика 0.

XCK: внешний тактовый сигнал UART. Регистр направления данных (DDB0) контролирует является вывод тактового сигнала выходом (DDB0 установлен) или входом (DDB0 очищен). Вывод XCK является активным только, если UART работает в синхронном режиме.

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

Т а б л и ц а 3.26 – Переопределяющие сигналы для альтернативных функций в PB7–PB4

Сигнал	PB7 / SCK	PB6 / MISO	PB5 / MOSI	PB4 / SS#
PUOE	SPE × MSTR#	SPE × MSTR	SPE × MSTR#	SPE × MSTR#
PUOV	PORTB7 × PUD#	PORTB6 × PUD#	PORTB5 × PUD#	PORTB4×PUD#
DDOE	SPE × MSTR#	SPE × MSTR	SPE × MSTR#	SPE × MSTR#
DDOV	0	0	0	0
PVOE	SPE × MSTR	SPE × MSTR#	SPE × MSTR	0
PVOV	SCK OUTPUT	SPI SLAVE OUT	SPI MSTR OUT	0
DIEOE	0	0	0	0
DIEOV	0	0	0	0
DI	SCK INPUT	SPI MSTR IN	SPI SLAVE IN	SPI SS#
AIO	-	-	-	-

Т а б л и ц а 3.27 – Переопределяющие сигналы для альтернативных функций в PB3–PB0

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

3.8.9 Альтернативные функции порта C

Выводы порта C с альтернативными функциями показаны в таблице 3.28.

Т а б л и ц а 3.28 – Выводы альтернативных функций порта C

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

Альтернативная конфигурация выводов выглядит следующим образом:

TOSC2 – порт C, бит 7

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

TOSC1 – порт C, бит 6

TOSC1, вывод 1 генератора таймера: при установке бита AS2 в ASSR для запуска асинхронного тактирования таймера/счетчика 2, вывод PC6 отсоединяется от порта и становится входом инвертирующего усилителя генератора. В этом режиме кварцевый генера-

тор подсоединяется к этому выводу и он не может использоваться в качестве порта входа/выхода.

SDA – порт C, бит 1

SDA, шина данных двухпроводного последовательного интерфейса: когда в TWCR для запуска двухпроводного последовательного интерфейса устанавливается бит TWEN, то вывод PC1 отсоединяется от порта и становится выводом последовательного входа-выхода данных для двухпроводного последовательного интерфейса. В этом режиме на выводе присутствует фильтр пиковых напряжений для подавления выбросов короче 50 нс для входного сигнала, и вывод управлятся с помощью драйвера с открытым стоком с ограничением скорости нарастания. Когда этот вывод используется двухпроводным последовательным интерфейсом, нагрузочное сопротивление может контролироваться с помощью разряда PORTC1.

SCL – порт C, бит 0

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

Таблица 3.29 – Переопределяющие сигналы для альтернативных функций в PC7, PC6, PC1, PC0

Сигнал	PC7 / TOSC2	PC6 / TOSC1	PC1 / SDA	PC0 / SCL
PUOE	AS2	AS2	TWEN	TWEN
PUOV	0	0	PORTC1 × PUD#	PORTC0 × PUD#
DDOE	AS2	AS2	TWEN	TWEN
DDOV	0	0	SDA_OUT	SCL_OUT
PVOE	0	0	TWEN	TWEN
PVOV	0	0	0	0
DIEOE	AS2	AS2	0	0
DIEOV	0	0	0	0
DI	-	-	-	-
AIO	T / C2 OSC OUT	T / C2 OSC IN	SDA INPUT	SCL INPUT

Примечание – При активировании двухпроводной последовательный интерфейс запускает управление скоростью нарастания на выходных выводах PC0 и PC1. В дополнение к этому, фильтры выбросов подсоединены между выходами AIO, показанными на схеме порта, и цифровой логикой модуля TWI.

3.8.10 Альтернативные функции порта D

Выводы порта D с альтернативными функциями показаны в таблице 3.30.

Таблица 3.30 – Альтернативные функции выводов порта D

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

Конфигурация альтернативного вывода выглядит следующим образом:

OC2 – порт D, бит 7

OC2 – выход сравнения/совпадения таймера/счетчика 2. Вывод OC2 является также выводом выхода для функции таймера режима PWM.

ICP – порт D, бит 6

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

OC1A – порт D, бит 5

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

OC1B – порт D, бит 4

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

INT1 – порт D, бит 3

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

IgT0 – порт D, бит 2

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

TXD – порт D, бит 1

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

RXD – порт D, бит 0

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

В таблицах 3.31 и 3.32 показана связь альтернативных функций порта D с сигналами коррекции.

Т а б л и ц а 3.31 – Переопределяющие сигналы для альтернативных функций PD7–PD4

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

Т а б л и ц а 3.32 – Переопределяющие сигналы для альтернативных функций PD3–PD0

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

Описание регистров для портов входа-выхода.

Регистр данных порта A – PORTA

Бит	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Чт./Зап.	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	
	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0	DDRA
Чт./Зап.	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 выводов входа порта A – PINA

Бит	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Чт./Зап.	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 данных порта B – PORTB

Бит	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Чт./Зап.	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	
	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0	DDRB
Чт./Зап.	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 выводов входа порта B – PINB

Бит	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Чт./Зап.	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 данных порта C – PORTC

Бит	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Чт./Зап.	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	
	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0	DDRC
Чт./Зап.	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 выводов входа порта C – PINC

Бит	7	6	5	4	3	2	1	0	
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Чт./Зап.	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 данных порта D – PORTD

Бит	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Чт./Зап.	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	
	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0	DDRD
Чт./Зап.	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 выводов входа порта D – PIND

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

3.9 Внешние прерывания

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

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

Если уровень запускаемого прерывания используется для пробуждения из режима хранения, то измененный уровень должен некоторое время удерживаться для активации МПУ. Это делает МПУ менее чувствительным к шуму. Измененный уровень стробируется дважды с помощью тактового импульса генератора сторожевого таймера. Период генератора равен 2 мкс (номинальное значение) при $U_{\#VCC}$ = 5,0 В, $U_{\#VCC}$ = 3,3 В и T = 25 ºC. Частота генератора зависит от напряжения. МПУ будет запущено, если на входе присутствует требуемый уровень во время этой выборки, или если этот уровень сохраняется до окончания времени запуска. Время запуска определяется битами SUT. Если уровень дважды стробируется тактовым генератором сторожевого таймера, но он исчезает до окончания времени запуска, то МПУ все еще будет способно к активации, но прерывание при этом генерироваться не будет. Требуемый уровень должен удерживаться достаточно длительное время, чтобы МПУ завершило пробуждение для запуска прерывания уровня.

3.9.1 Регистр управления МПУ – MCUCR

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

Бит	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Чт./Зап.	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 3, 2: ISC11, ISC10 – управление типом прерывания 1, разряд 1 и разряд 0

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

Т а б л и ц а 3.33 – Управление обнаружением прерываний 1

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

Разряды 1, 0: ISC01, ISC00 – управление типом прерывания 0 разряд 1 и разряд 0

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

Т а б л и ц а 3.34 – Управление обнаружением прерывания 0

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

Регистр состояния и управления МПУ – MCUCSR

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

Разряд 6: ISC2 – Управление типом прерывания 2

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

Т а б л и ц а 3.35

Символ	Параметр	Мин.	Тип.	Макс.	Единица измерения
t_{INT}	Минимальная ширина импульса для асинхронного внешнего прерывания	TBD	50	TBD	нс

3.9.2 Регистр управления общим прерыванием – GICR

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

Разряд 7: INT1 – разрешение запроса на внешнее прерывание 1

Когда устанавливается разряд INT1 и устанавливается разряд «I» в регистре состояний (SREG), то разрешается прерывание для внешнего вывода. Биты контроля чувствительности прерывания 1 – ISC11 и ISC10 в регистре общего управления МПУ (MCUCR) определяют, будет ли внешнее прерывание активироваться передним и/или задним фронтом вывода INT1 или будет обнаруживаться по уровню. Активность на выводе вызовет запрос на прерывание даже в том случае, когда INT1 конфигурируется как выход. Соответствующее прерывание внешнего запроса на прерывание 1 выполняется по вектору прерывания INT1.

Разряд 6: INT0 – разрешение запроса на внешнее прерывание 0

Когда устанавливается разряд IpT0 и устанавливается разряд «I» в регистре состояний (SREG), то разрешается прерывание для внешнего вывода. Биты контроля чувствительности прерывания 0 – ISC01и ISC00 в регистре общего управления МПУ (MCUCR) определяют, будет ли внешнее прерывание активироваться передним и/или задним фронтом вывода INT1 или будет обнаруживаться по уровню. Активность на выводе вызовет запрос на

прерывание даже в том случае, когда INT0 конфигурируется как выход. Соответствующее прерывание внешнего запроса на прерывание 0 выполняется по вектору прерывания INT0.

Разряд 5: INT2 – разрешение запроса на внешнее прерывание 2

Когда устанавливается разряд INT2 и устанавливается разряд «I» в регистре состояний (SREG), то разрешается прерывание для внешнего вывода. Бит контроля чувствительности прерывания ISC2 в регистре общего управления МПУ (MCUCR) определяет: будет ли внешнее прерывание активироваться передним и/или задним фронтом вывода INT2. Активность на выводе вызовет запрос на прерывание даже в том случае, когда INT2 конфигурируется как выход. Соответствующее прерывание внешнего запроса на прерывание 2 выполняется по вектору прерывания INT2.

3.9.3 Регистр флага общего прерывания – GIFR

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

Разряд 7: INTF1 – флаг 1 внешнего прерывания

Когда фронт или логическое изменение на выводе INT1 активирует запрос на прерывание, то INTF1 устанавливается. Если разряд «I» в SREG и разряд INT1 в GICR установлены, то программа МК перейдет на соответствующий вектор прерывания. Флаг очистится после выполнения процедуры прерывания. В качестве альтернативы флаг может быть очищен путем записи в него логической единицы. Этот флаг всегда очищен, когда INT1 конфигурируется для срабатывания прерывания по уровню.

Разряд 6: INTF0 – флаг 0 внешнего прерывания

Когда фронт или логическое изменение на выводе INT0 активирует запрос на прерывание, то INTF0 устанавливается. Если разряд «I» в SREG и разряд INT0 в GICR установлены, то программа МК перейдет на соответствующий вектор прерывания. Флаг очистится после выполнения процедуры прерывания. В качестве альтернативы флаг может быть очищен путем записи в него логической единицы. Этот флаг всегда очищен, когда INT0 конфигурируется для срабатывания прерывания по уровню.

Разряд 5: INTF2 – флаг 2 внешнего прерывания

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

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

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

Функциональные особенности:

- одноканальный счетчик;
- сброс таймера при совпадении сравнения (автоматическая перезагрузка);
- ШИМ с фазовой коррекцией, свободный от сбоев;

- генератор частоты;
- счетчик внешних событий;
- 10-разрядный предварительный делитель частоты;
- источники прерывания при сравнении/совпадении и при переполнении (TOV0 и OCF0).

3.10.1 Обзор

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

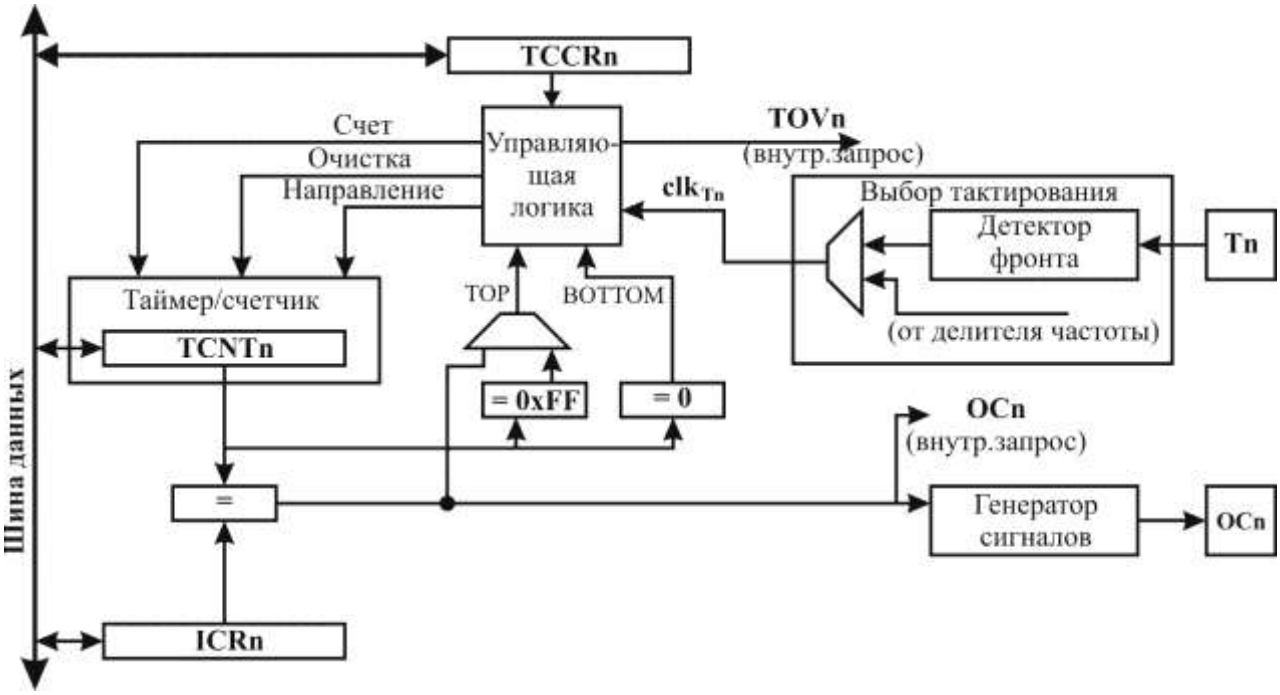


Рисунок 3.25 – Блок-схема 8-разрядного таймера/счетчика и его окружения

3.10.2 Регистры

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

Таймер/счетчик может синхронизироваться внутрисхемно, через предварительный делитель частоты, или с помощью источника внешнего тактового генератора на выводе T0. Логический блок управления синхронизацией выбирает, какой источник сигнала синхронизации и фронта использует таймер/счетчик для инкремента или декремента его значения. Таймер/счетчик не активен, если не выбран источник сигнала синхронизации. Выход логики выбора тактового сигнала обозначается как тактовый сигнал таймера (clk_{T0}). Регистр сравнения по выходу с двойным буфером (OCR0) постоянно сравнивается со значением таймера/счетчика. Результат сравнения может быть использован генератором сигнала для запуска ШИМ или выхода переменной частоты на выводе «выход сравнения» (OC0). Событие совпадения будет также устанавливать флаг сравнения (OCF0), который может использоваться для запроса прерывания выхода сравнения.

3.10.3 Определения

Многие справочные сведения относительно регистров и разрядов записаны в КФДЛ.431295.039ТО в общем виде. Буква «n» заменяет номер таймера/счетчика, в данном случае 0. Использование регистра или разряда определяется программой, необходимо использовать точную форму, например, TCNT0 для оценки значения таймера/счетчика 0 и т. д.

Определения из таблицы 3.36 также используются в подразделе 3.10.

Т а б л и ц а 3.36 – Определения

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

3.10.4 Источники тактового сигнала таймера/счетчика

Таймер/счетчик может синхронизироваться как внутренним, так и внешним источником синхронизации. Источник синхронизации выбирается с помощью логики выбора тактового генератора, который управляется с помощью разрядов выбора сигнала синхронизации CS0(2–0), расположенных в регистре управления таймера/счетчика (TCCR0).

3.10.5 Модуль счетчика

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

Рисунок 3.26 – Блок схема модуля счетчика

Описание сигналов (внутренние сигналы):

Счет	– инкремент или декремент TCNT0 на 1;
Направление	– выбор между инкрементом или декрементом;
Очистка	– очистка TCNT0 (значения всех разрядов устанавливаются в ноль);
clk_{Tn}	– тактовый сигнал таймера/счетчика, далее именуемый как clk_{T0};
top	– сигнализирует, что TCNT0 достигло верхнего значения;
bottom	– сигнализирует, что TCNT0 достигло нижнего значения (нуля).

В зависимости от используемого режима работы счетчик сбрасывается, инкрементируется или декрементируется при каждом тактовом сигнале таймера (clk_{T0}). clk_{T0} сможет

генерироваться от внешнего или внутреннего источника тактового сигнала, выборка при этом производится с помощью разрядов CS0(2–0). Если источник тактовых сигналов не выбирается, т. е. CS0(2–0) = 0, то таймер останавливается. В то же время величина TCNT0 может быть выбрана с помощью ЦПУ независимо от того, присутствует clk_{T0} или нет. Операция записи в ЦПУ отменяет (обладает большим приоритетом) все операции очистки счетчика или операции счета.

Последовательность счета определяется путем установки разрядов WGM01 и WGM00, расположенных в регистре управления таймером/счетчиком (TCCR0). Существует непосредственная связь между тем, как ведет себя счетчик (т. е. как он считает) и какие генерируются сигналы на выходе сравнения OC0.

Флаг переполнения таймера/счетчика (TOV0) устанавливается в соответствии с режимом работы, который выбирается с помощью разрядов WGM0(1–0). Для генерации прерывания ЦПУ можно использовать TOV0.

3.10.6 Модуль сравнения по выходу

8-разрядный компаратор непрерывно сравнивает TCNT0 с регистром сравнения по выходу (OCR0). Как только TCNT0 становится равным OCR0, компаратор сигнализирует о совпадении. Совпадение установит флаг выхода сравнения (OCF0) в следующем цикле тактового сигнала таймера. При активации (OCIE0 = 1 при установке флага общего прерывания в SREG) флаг выхода сравнения создает прерывание. Флаг OCF0 очищается автоматически при выполнении прерывания. В качестве альтернативы, флаг OCF0 можно очищать программно путем записи в него логической единицы. Генератор сигнала произвольной формы использует сигнал сравнения для генерации выходного сигнала согласно режиму работы, установленному разрядами WGM0(1–0), и режиму выхода сравнения, установленному разрядами COM0(1–0). Максимальное и нижнее значение сигнала используется генератором сигнала произвольной формы для решения специальных случаев с крайними значениями в некоторых режимах работы.

На рисунке 3.27 показана схема блока управления устройства сравнения по выходу.

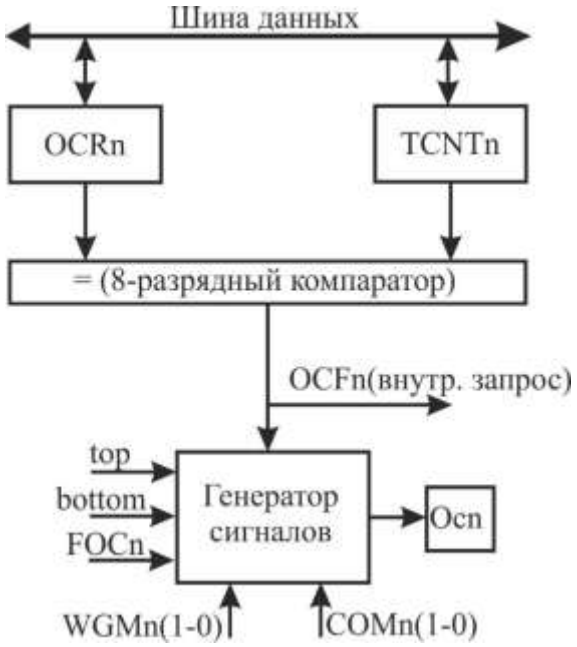


Рисунок 3.27 – Блок-схема модуля сравнения по выходу

При работе в любом режиме с использованием ШИМ (широтно-импульсная модуляция), регистр OCR0 имеет двойную буферизацию. При работе в обычном режиме и ре-

жиме очистки таймера при сравнении (CTC), двойная буферизация отключается. Двойная буферизация синхронизирует обновление регистра сравнения либо к верхнему, либо к нижнему значению последовательности счета. Синхронизация предотвращает возникновение импульсов избыточной длины, несимметричных импульсов ШИМ, обеспечивая таким образом бесперебойную работу выхода. Доступ к регистру может показаться достаточно сложным, но это не так. При активации двойной буферизации ЦПУ имеет доступ к регистру буфера OCR0, а если двойная буферизация отключена, то ЦПУ будет осуществлять прямую выборку OCR0.

3.10.7 Принудительное совпадение

В режимах генерации без использования режима ШИМ выход совпадения компаратора может быть принудительно установлен записью логической единицы в разряд принудительного сравнения (FOC0). Принудительное сравнение/совпадение не будет устанавливать флаг OCF0 или перезагружать/сбрасывать таймер, однако вывод OC0 будет обновляться, как будто произошло реальное совпадение при сравнении (установки разрядов COM0(1–0) определяют, является ли вывод OC0 установленным, сброшенным или переключенным).

3.10.8 Блокировка совпадения/сравнения с помощью записи TCNT0

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

3.10.9 Использование блока сравнения по выходу

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

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

Необходимо убедиться в том, что разряды COM0(1–0) не имеют двойной буферизации вместе со значением сравнения. Изменение разрядов COM0(1–0) дает немедленный эффект.

3.10.10 Блок совпадения при сравнении

Разряды режима сравнения на выходе COM0(1–0) несут две функции. Генератор формы сигнала использует разряды COM0(1–0) для оценки состояния сравнения на выходе при следующем совпадении при сравнении. Кроме этого, разряды COM0(1–0) управляют источником выходного сигнала для вывода OC0. На рисунке 3.28 представлена упрощенная блок-схема логики, используемой при установке разряда COM0(1–0), регистры входа/выхода, разряды входа/выхода и выводы входа/выхода. Показаны только части общих

регистров управления портом входа/выхода (DDR и PORT) на которые влияют разряды COM0(1–0). При ссылках на состояние OC0, речь идет о внутреннем регистре OC0, а не о выводе OC0. Если происходит сброс системы, то регистр OC0 сбрасывается в ноль.

Общая функция порта входа/выхода переписывается функцией выхода сравнения (OC0) из генератора сигнала, если устанавливается какой-либо из разрядов COM0(1–0). Однако, направление вывода OC0 (вход или выход) все еще контролируется регистром направления данных (DDR) для вывода порта. Разряд регистра направления данных для вывода OC0 (DDR_OC0) должен быть установлен в качестве выхода до того, как величина OC0 станет видимой на выводе. Функция переопределения порта не зависит от режима генерации формы сигнала.

Логическая схема сравнения разрешает инициализацию состояния OC0 до того, как активируется выход. Следует обратить внимание на то, что некоторые установки разряда COM0(1–0) сохраняются для определенных режимов работы.

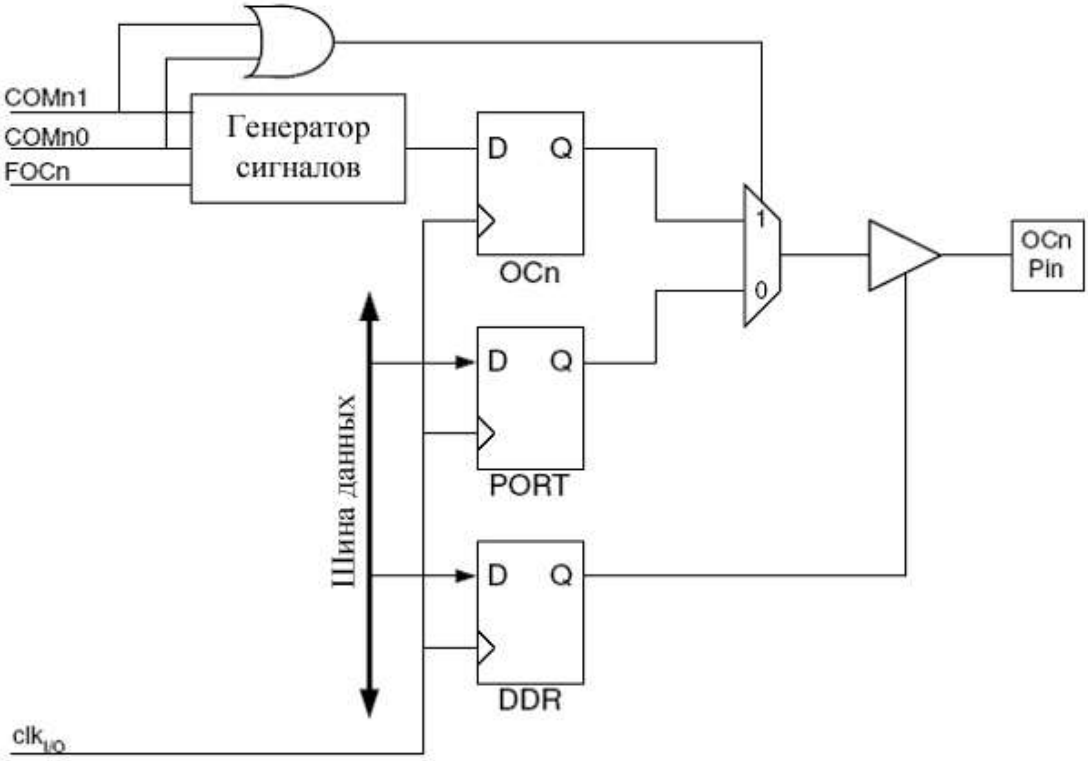


Рисунок 3.28 – Блок сравнения/совпадения на выходе

3.10.11 Режим сравнения по выходу и генерация сигнала произвольной формы

Генератор сигнала произвольной формы использует разряды COM0(1–0) по-разному для обычного режима, CTC и режима ШИМ. Для всех типов режимов установка COM0(1–0) = 0 сообщает генератору сигнала, что в регистре OC0 не должно выполняться никаких действий по следующему сравнению/совпадению.

Изменение состояния разрядов в COM0(1–0) повлияет на первое совпадение при сравнении после записи разрядов. Для режимов, отличающихся от ШИМ, может быть использовано действие, дающее немедленный эффект благодаря использованию разрядов стробирования FOC0.

3.10.12 Режимы работы

Режим работы, т. е. поведение таймера/счетчика и выводов «выхода сравнения», определяется комбинацией разрядов для режима генерации сигнала WGM0(1–0) и режима сравнения по выходу COM0(1–0). Разряды режима сравнения по выходу не влияют на последовательность счета, в то время как разряды режима генерации сигнала влияют на него. Разряды COM0(1–0) управляют тем, будет ли инвертирован или нет сгенерированный выход ШИМ (инвертированный или неинвертированный ШИМ). Для режимов, не использующих ШИМ, разряды COM0(1–0) контролируют должен ли выход быть установлен, очищен или переключен по совпадению при сравнении.

Нормальный режим

Самым простым режимом работы является нормальный режим WGM0(1–0) = 0. В этом режиме направление счета всегда следует вверх (значение инкрементируется) и сброс счетчика не выполняется. Счетчик просто переполняется, когда он проходит свое максимальное 8-разрядное значение (TOP = 0xFF) и затем перезапускается (0x00). В нормальном режиме работы флаг переполнения таймера/счетчика (TOV0) будет установлен во время того же цикла синхронизации таймера, когда TCNT0 становится равным нулю. Флаг TOV0 в этом случае ведет себя как девятый разряд, за исключением того, что он только устанавливается, а не сбрасывается. Однако, при объединении с прерыванием переполнения таймера, которое автоматически сбрасывает флаг TOV0, разрешение таймера можно повысить программным способом. В нормальном режиме нет необходимости рассматривать специальные случаи, при этом новое значение таймера может быть записано в любое время.

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

Сброс таймера в режиме сравнения/совпадения (CTC)

В режиме сброса таймера при сравнении или режиме CTC, WGM0(1–0) = 2, регистр OCR0 используется для управления разрешением счетчика. В режиме CTC счетчик очищается, когда значение счетчика (TCNT0) совпадает с OCR0. OCR0 определяет верхнее значение для счетчика, а следовательно, его разрешение. Данный режим обеспечивает большую степень управления выходной частотой совпадения при сравнении. Это упрощает также работу для подсчета внешних событий.

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

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

Для генерации сигнала в режиме CTC выход OC0 может быть установлен на переключение своего логического уровня при каждом совпадении при сравнении путем установки разрядов режима сравнения по выходу на режим переключения COM0(1–0) = 1.

Значение OC0 будет отсутствовать на выводе порта до тех пор, пока не будет определено направление данных на выход.

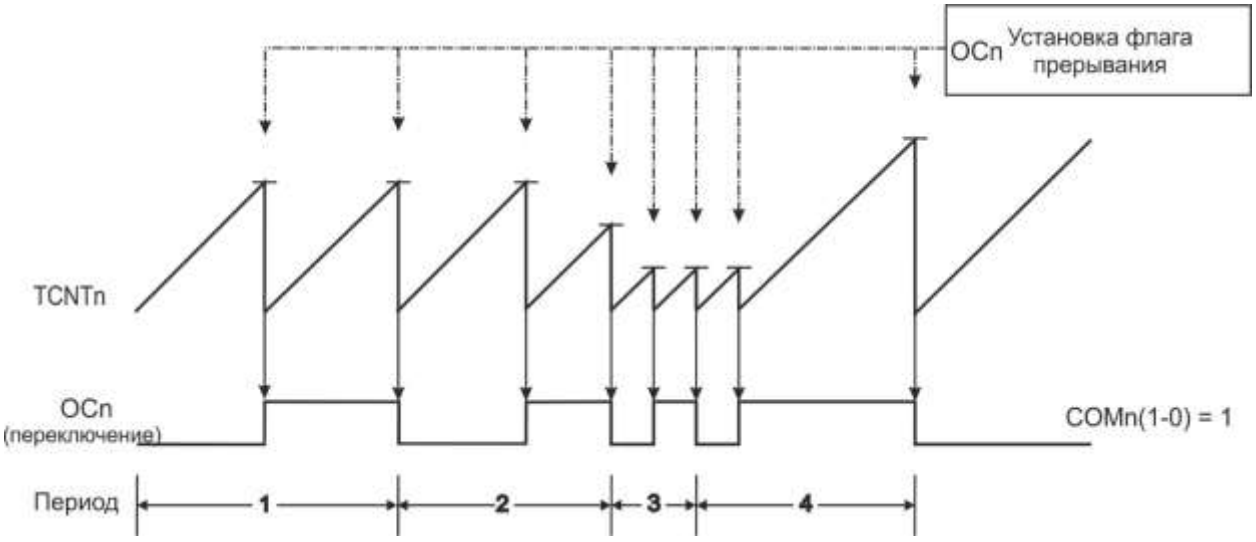


Рисунок 3.29 – Режим CTC, временная диаграмма

Генерируемый сигнал будет иметь максимальную частоту $f_{OC0} = f_{clk\_I/O}/2$ при установке OCR0 в ноль (0x00). Частота сигнала определяется следующим выражением:

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

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

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

Быстрый ШИМ режим

Быстрая широтно-импульсная модуляция или быстрый ШИМ режим WGM0(1–0) = 3 предоставляет опцию генерации сигнала ШИМ с высокой частотой. Быстродействующая ШИМ отличается от обычной ШИМ благодаря операциям с использованием одиночных фронтов. Счетчик выполняет счет из положения BOTTOM до MAX, затем вновь начинает работу с BOTTOM. В неинвертируемом режиме выхода сравнения OC0 очищается по совпадению при сравнении между TCNT0 и OCR0 и устанавливается как BOTTOM. В инвертируемом режиме сравнения по выходу, выход устанавливается по совпадению при сравнении и сбрасывается при BOTTOM. Благодаря работе с использованием одиночных фронтов, рабочая частота для быстрого ШИМ режима может в два раза превышать частоту ШИМ режима с фазовой коррекцией частоты, которая использует принцип работы с применением двух фронтов. Благодаря столь высокой частоте быстрый ШИМ режим очень хорошо подходит для регулировки мощности, выпрямления и применения в ЦАП. Высокая частота обуславливает малые физические размеры внешних компонентов (катушек, конденсаторов), благодаря чему снижается общая стоимость системы.

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

как инвертированные, так и неинвертированные ШИМ выходы. Небольшие горизонтальные метки на фронтах TCNT0 представляют совпадения между OCR0 и TCNT0.

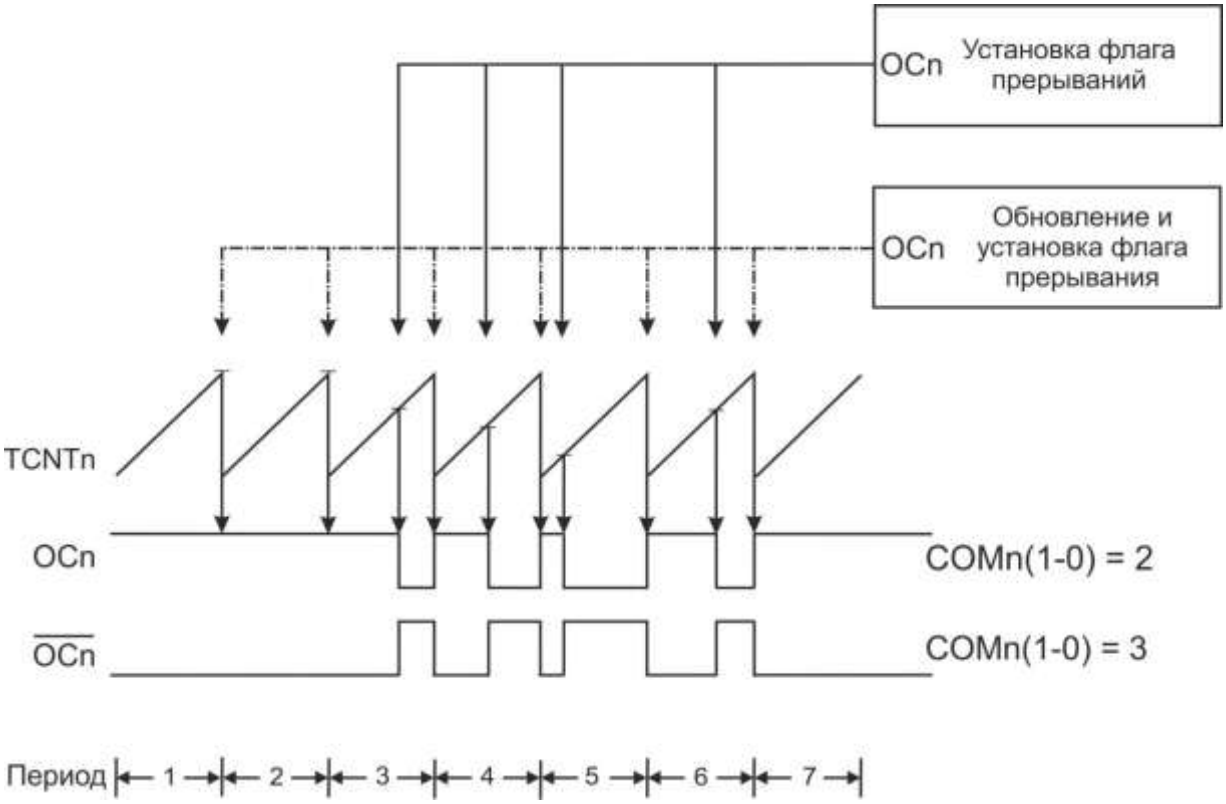


Рисунок 3.30 – Временная диаграмма для быстрого ШИМ режима

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

В быстром ШИМ режиме модуль сравнения разрешает генерацию ШИМ сигналов на выводе OC0. Установка разрядов COM0(1–0) в значение «10» дает неинвертированный ШИМ, а инвертированный ШИМ можно сформировать, установив COM0(1–0) в значение «11». Фактическое значение OC0 будет видно на выводе входа, если направление данных для разряда порта определено как выход. Форма сигнала ШИМ генерируется с помощью установки (или очистки) регистра OC0 по совпадению при сравнении между OCR0 и TCNT0 и очистки (или установки) регистра OC0, во время тактового периода, когда счетчик очищается (изменяется с MAX на BOTTOM).

Частоту ШИМ для выхода можно вычислить, используя следующее уравнение:

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

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

Крайние значения для регистра OCR0 представляют особые случаи при генерации формы сигнала ШИМ на выходе в быстром ШИМ режиме. Если OCR0 устанавливается равным BOTTOM, то выходной сигнал будет представлять собой узкий импульс для каждого тактового цикла таймера MAX + 1. Если OCR0 устанавливается равным MAX, то это приводит к постоянному высокому или низкому значению на выходе (в зависимости от полярности на выходе, которая установлена разрядами COM0(1–0).

Частота сигнала на выходе (при 50 % рабочем режиме) в быстром ШИМ режиме может быть получена путем установки OC0 на переключение своего логического уровня при каждом совпадении при сравнении COM0(1–0) = 1. Форма сигнала, генерируемая при этом, будет иметь максимальную частоту $f_{OC0} = f_{clk\_I/O}/2$ при установке значения OCR0 в ноль. Эта особенность аналогична переключению OC0 в режиме CTC, за исключением того, что двойная буферизация в модуле сравнения по выходу в быстром ШИМ режиме включена.

Режим фазовой коррекции ШИМ

Режим фазовой коррекции ШИМ WGM0(1–0) = 1 обеспечивает возможность генерации сигнала ШИМ с фазовой коррекцией и высокой разрешающей способностью. Режим фазовой коррекции ШИМ основан на работе с использованием двух фронтов. Счетчик ведет повторяемый подсчет от BOTTOM к MAX, а затем от MAX к BOTTOM. В неинвертированном режиме сравнения OC0 очищается при совпадении между TCNT0 и OCR0 при инкременте и устанавливается при совпадении при декременте. В инвертированном режиме сравнения по выходу операция инвертируется. Принцип работы с двойным фронтом имеет пониженную максимальную рабочую частоту по сравнению с одинарным фронтом. В то же время, благодаря симметричной особенности ШИМ режимов с двойным фронтом, подобные режимы предпочтительны для использования при управлении двигателями.

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

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

В режиме ШИМ с фазовой коррекцией модуль сравнения обеспечивает генерацию импульсов ШИМ на выводе OC0. Установка разрядов COM0(1–0) в значение «10» вызовет неинвертированный ШИМ режим. Инвертированный выход ШИМ может генерироваться путем записи в COM0(1–0) значения «11».

Фактическое значение OC0 будет присутствовать на выводе порта только в случае, когда направление данных вывода порта будет задано как «вход». Форма ШИМ сигнала генерируется путем сброса (или установки) регистра OC0 при совпадении между OCR0 и TCNT0, когда значение счетчика инкрементируется, и при установке (или сбросе) регистра OC0 при совпадении между OCR0 и TCNT0, когда значение счетчика декрементируется. Частота ШИМ для выхода может быть вычислена с помощью следующего уравнения:

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

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

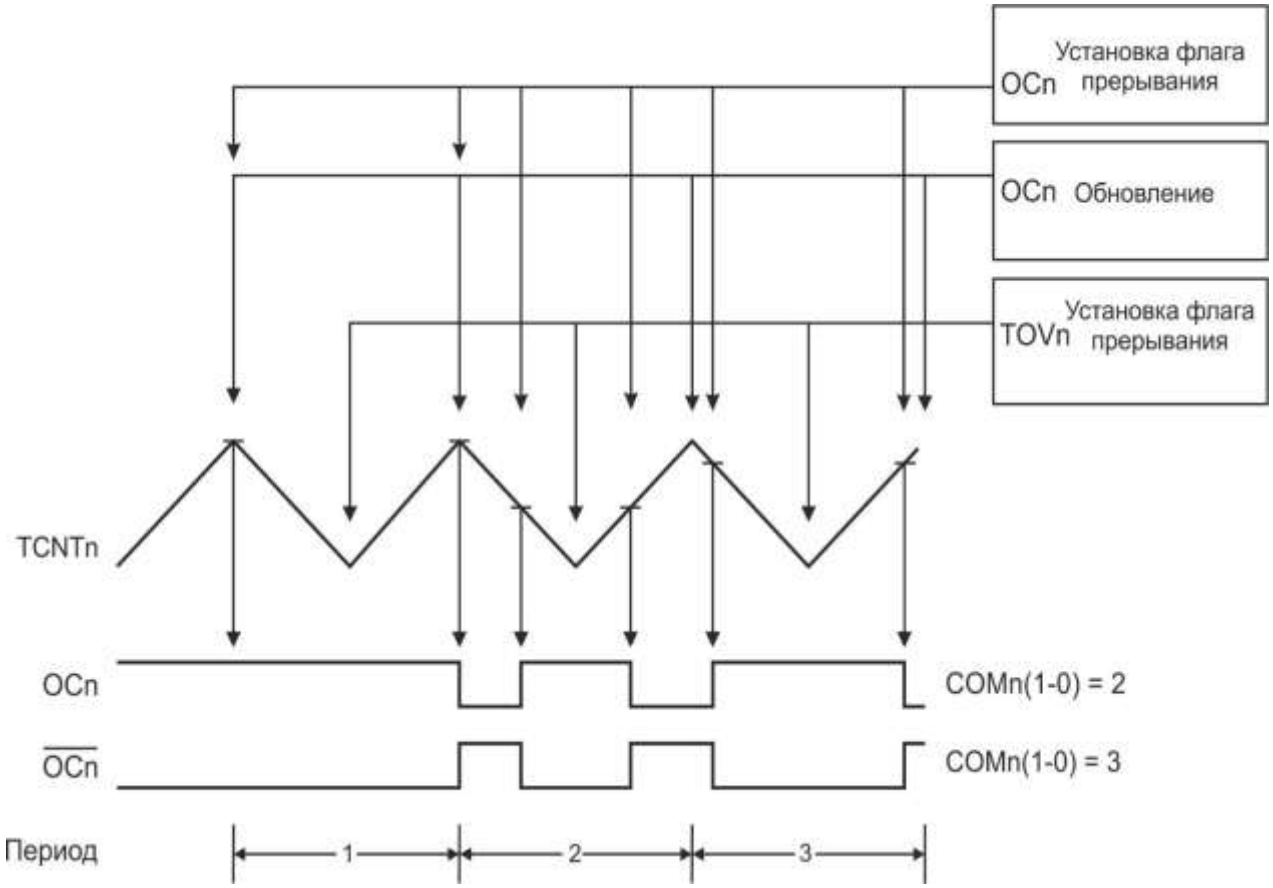


Рисунок 3.31 – Временная диаграмма, режим ШИМ с фазовой коррекцией

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

Временные диаграммы таймера/счетчика

Таймер/счетчик имеет синхронный принцип работы и тактовый сигнал таймера (clk_{T0}) с учетом этого показан на рисунках 3.32–3.35 как сигнал разрешения тактирования. На рисунках дана информация о том, когда устанавливаются флаги прерывания. На рисунке 3.32 приведены временные параметры по основному режиму работы таймера/счетчика, приведена последовательность счета, близкая к значению MAX во всех режимах, отличающихся от режима ШИМ с фазовой коррекцией.

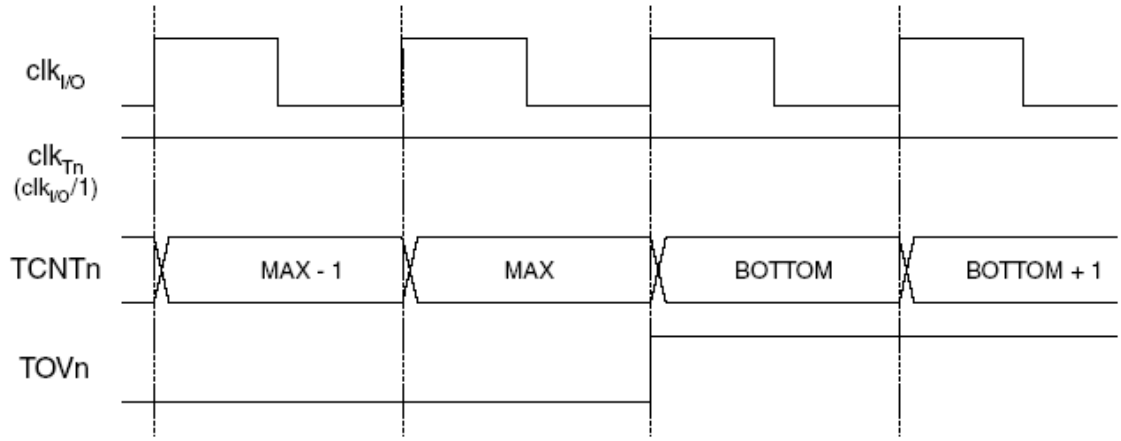


Рисунок 3.32 – Временная диаграмма таймера/счетчика без предделителя

На рисунке 3.33 показаны те же временные параметры, но с предделителем.

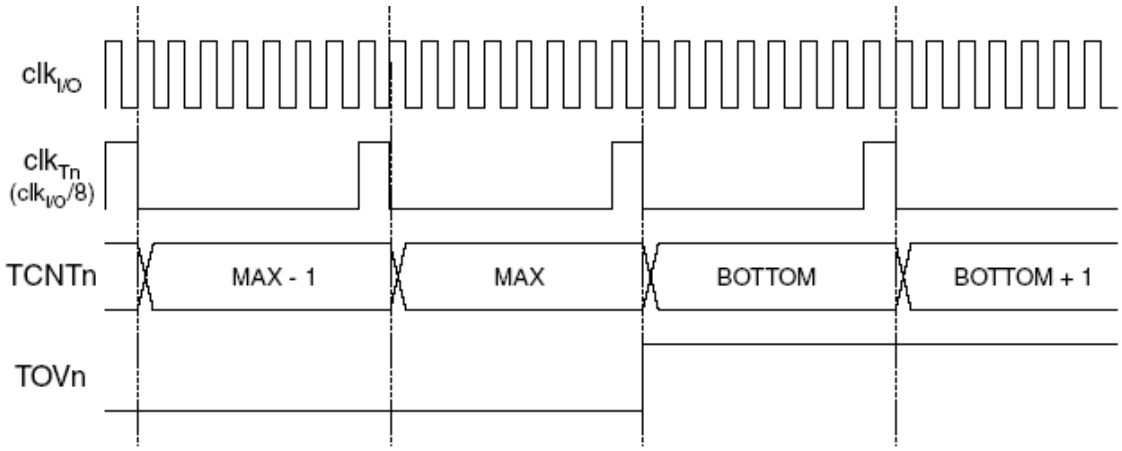


Рисунок 3.33 – Временная диаграмма таймера/счетчика с предделителем ($f_{clk\_I/O}/8$)

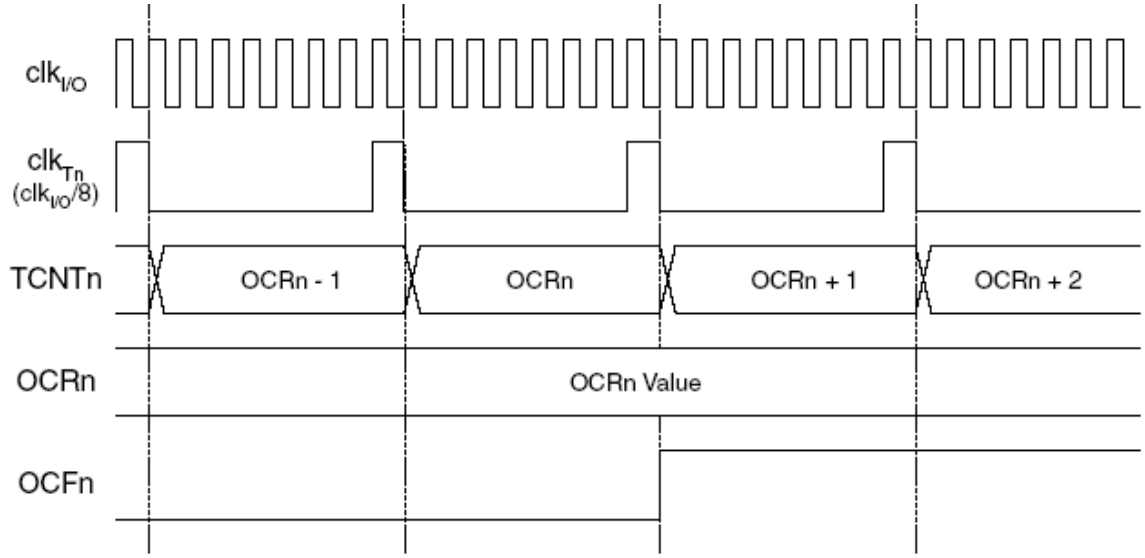


Рисунок 3.34 – Временная диаграмма таймера/счетчика с установкой OCF0
с предделителем ($f_{clk\_I/O}/8$)

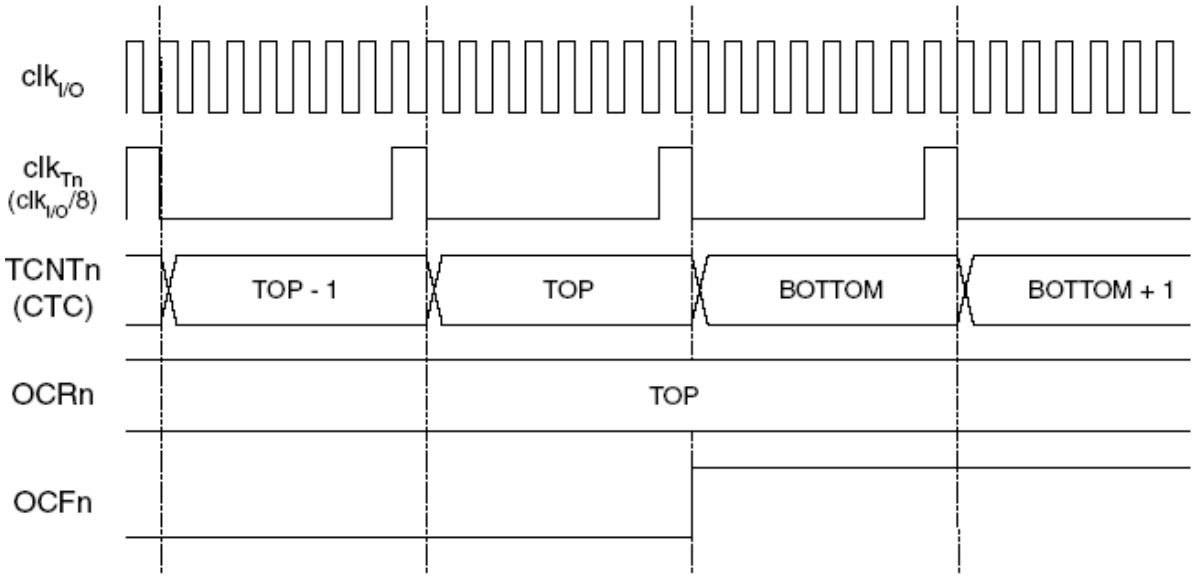


Рисунок 3.35 – Временная диаграмма таймера/счетчика, сброс таймера в режиме сравнения/совпадения, с предделителем ($f_{clk\_I/O}/8$)

3.10.13 Описание регистров 8-разрядного таймера/счетчика

Регистр управления таймера/счетчика – TCCR0

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

Разряд FOC0 активируется только тогда, когда разряд WGM00 задает режим без использования ШИМ. В то же время, для совместимости с будущими устройствами, этот разряд должен быть установлен в ноль, когда записывается TCCR0 при работе в режиме ШИМ. При записи логической единицы в разряд FOC0, на модуль генерации сигнала подается команда на немедленное сравнение. Выход OC0 изменяется в соответствии с установкой его разрядов COM0(1–0). Следует обратить внимание на то, что разряд FOC0 реализован в виде строба. С учетом этого значение, присутствующее в разрядах COM0(1–0), определяет результат принудительного сравнения.

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

Разряд FOC0 всегда считывается как ноль.

Разряды 6, 3 – режим генерации сигнала WGM0(1–0)

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

Таблица 3.37 – Описание разряда режима работы генерации формы сигнала *

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

* Наименования разрядов CTC0 и PWM0 в настоящее время считаются устаревшими. Используются определения WGM0(1–0), в то же время функциональность и расположение этих разрядов совместимы с прежними версиями таймера.

Разряды 5, 4: COM0(1–0) – режим выхода сравнения/совпадения

Эти разряды управляют поведением вывода «выход сравнения» (OC0). Если устанавливается один или оба разряда COM0(1–0), то выход меняет нормальную функциональность порта входа/выхода. В то же время следует обратить внимание на то, что разряд регистра направления (DDR), соответствующий выводу OC0, должен быть установлен так, чтобы активировать выходной драйвер.

Когда к выводу подсоединен OC0, то функция разрядов COM0(1–0) зависит от установки разряда WGM0(1–0). В таблице 3.38 показана функциональность разряда COM0(1–0), когда разряды WGM0(1–0) устанавливаются в нормальный режим работы или в режим CTC (не режим ШИМ).

Таблица 3.38 – Режим сравнения по выходу, режим без ШИМ

COM01	COM00	Описание
0	0	Нормальная работа порта, OC0 отсоединен
0	1	Переключение OC0 на совпадение при сравнении
1	0	Сброс OC0 по совпадению при сравнении
1	1	Установка OC0 по совпадению при сравнении

В таблице 3.39 показывается функциональность разряда COM0(1–0), когда разряды WGM0(1–0) устанавливаются в режим быстрого ШИМ.

Таблица 3.39 – Режим сравнения по выходу, быстрый режим ШИМ *

COM01	COM00	Описание
0	0	Нормальная работа порта, OC0 отсоединен
0	1	Зарезервировано
1	0	Сброс OC0 по совпадению при сравнении, установка OC0 при TOP
1	1	Установка OC0 по совпадению при сравнении, сброс OC0 при TOP

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

В таблице 3.40 показывается функциональность разрядов COM0(1–0), когда разряды WGM0(1–0) установлены в режим фазовой коррекции ШИМ.

Т а б л и ц а 3.40 – Режим сравнения по выходу, режим фазовой коррекции ШИМ

COM01	COM00	Описание
0	0	Нормальная работа порта, OC0 отсоединен
0	1	Зарезервировано
1	0	Сброс OC0 по совпадению при сравнении, при счете вверх. Установка OC0 по совпадению при сравнении при подсчете вниз
1	1	Установка OC0 по совпадению при сравнении, сброс OC0 по совпадению при сравнении при счете вниз
П р и м е ч а н и е – Особый случай имеет место, когда OCR0 равно TOP и установлен COM01. В этом случае совпадение при сравнении игнорируется, однако установка или сброс выполняются при TOP.		

Разряды 2–0: CS0(2–0) – выбор синхронизации

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

Т а б л и ц а 3.41 – Описание разряда выбора синхронизации

CS02	CS01	CS00	Описание
0	0	0	Нет источника синхронизации (таймер/счетчик остановлен)
0	0	1	$clk_{I/O}$ (без деления частоты)
0	1	0	$clk_{I/O}$ /8 (от устройства предварительного деления частоты)
0	1	1	$clk_{I/O}$ /64 (от устройства предварительного деления частоты)
1	0	0	$clk_{I/O}$ /256 (от устройства предварительного деления частоты)
1	0	1	$clk_{I/O}$ /1024 (от устройства предварительного деления частоты)
1	1	0	Внешний источник тактового сигнала на выводе T0. Тактирование по заднему фронту
1	1	1	Внешний источник тактового сигнала на выводе T0. Тактирование по переднему фронту

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

Регистр таймера/счетчика – TCNT0

Бит	7	6	5	4	3	2	1	0	
	TCNT0(7–0)								TCNT0
Чт./Зап.	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 обеспечивает прямой доступ как для операции считывания, так и записи в 8-разрядный счетчик блока таймера/счетчика. Запись в блоки регистра TCNT0 удаляет совпадение при сравнении на следующем сигнала синхронизации таймера. Изменение счетчика TCNT0 во время работы счетчика создает риск отсутствия совпадения при сравнении между TCNT0 и регистром OCR0.

Регистр сравнения по выходу – OCR0

Бит	7	6	5	4	3	2	1	0	
	OCR0(7-0)								OCR0
Чт./Зап.	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 по выходу содержит 8-разрядную величину, которая непрерывно сравнивается со значением счетчика (TCNT0). Совпадение может использоваться для генерации прерывания при сравнении или для генерации выходного сигнала на выводе OC0.

Регистр маски прерывания таймера/счетчика – TIMSK

Бит	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE1	TOIE0	TIMSK
Чт./Зап.	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 при сравнении на выходе таймера/счетчика

Если разряд OCIE0 и разряд «I» в регистре состояния установлен, то разрешается прерывание по совпадению при сравнении таймера/счетчика 0. Соответствующее прерывание выполняется, если происходит совпадение при сравнении в таймере/счетчике 0 (т. е. когда разряд OCF0 установлен в регистре флага прерывания таймера/счетчика – TIFR).

Разряд 0: TOIE0 – разрешение прерывания переполнения таймера/счетчика 0

Если разряд OCIE0 и разряд «I» в регистре состояния установлен, то разрешается прерывание переполнения таймера/счетчика 0. Соответствующее прерывание выполняется при появлении переполнения в таймере/счетчике 0 (т. е. когда разряд TOV0 устанавливается в регистре флага прерывания таймера/счетчика – TIFR)

Регистр флага прерывания таймера/счетчика – TIFR

Бит	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Чт./Зап.	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 0 выхода сравнения

Разряд OCF0 устанавливается, когда происходит совпадение при сравнении между таймером/счетчиком 0 и данными с OCR0 – регистре 0 сравнения по выходу. OCF0 очищается аппаратным способом при выполнении соответствующей процедуры обработки прирывания. В качестве альтернативы, OCF0 очищается путем записи логической единицы в разряд флага. Когда устанавливается разряд «I» в SREG, OCIE0 (разрешение прерывания по совпадению при сравнении таймера/счетчика 0), и OCF0, то выполняется прерывание по совпадению при сравнении таймера/счетчика 0.

Разряд 0: TOV0 – флаг переполнения таймера/счетчика 0

Разряд TOV0 устанавливается, когда происходит переполнение таймера/счетчика 0. TOV0 очищается аппаратным способом при выполнении соответствующей процедуры обработки прерывания. В качестве альтернативы, TOV0 очищается путем записи логической единицы в разряд флага. Когда устанавливаются разряды SREG «I», TOIE0 (разрешение прерывания переполнения таймера/счетчика 0) и разряд TOV0, то выполнятся прерывание при переполнении таймера/счетчика 0. В режиме фазовой коррекции ШИМ этот разряд устанавливается, когда таймер/счетчик 0 изменяет направление счета при 0x00.

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

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

3.11.1 Источник внутреннего синхроимпульса

Таймер/счетчик может синхронизироваться непосредственно системным тактовым генератором (с помощью установки CSn(2–0) = 1). Это обеспечивает более быструю работу с максимальной тактовой частотой таймера/счетчика, равной тактовой частоте системы ($f_{clk\_I/O}$). Альтернативно в качестве тактового генератора можно использовать один из четырех выводов от предварительного делителя частоты. Деление частоты позволяет получить одну из частот $f_{clk\_I/O}/8$, $f_{clk\_I/O}/64$, $f_{clk\_I/O}/256$ или $f_{clk\_I/O}/1024$.

3.11.2 Сброс предварительного деления частоты

Устройство предварительного деления частоты работает свободно (т. е. независимо от логики выбора частоты таймера счетчика) и используется совместно таймером/счетчиком 1 и таймером/счетчиком 0. Поскольку на устройство предварительного деления частоты не влияет выбор тактовой частоты таймера/счетчика, то состояние устройства предварительного деления частоты будет иметь последствия для ситуаций, когда используется предварительное деление частоты. Один из примеров случая предварительного деления происходит, когда таймер запускается и синхронизируется с помощью устройства предварительного деления частоты (6 > CSn(2–0) > 1). Число системных импульсов синхронизации с того момента, когда таймер запускается до момента, когда происходит первый подсчет, может составлять от одного до N+1 системных импульсов синхронизации, где N равно коэффициенту деления предварительного делителя частоты (8, 64, 256 или 1024).

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

3.11.3 Источник внешней синхронизации

Источник внешней синхронизации подключается к выводу T1/T0, может использоваться как устройство синхронизации таймера/счетчика (clk_{T1}/clk_{T0}). Выборка T1/T0 осу-

ществляется при каждом импульсе синхронизации системы с помощью логики синхронизации вывода. Синхронизированный (выбранный) сигнал затем проходит через устройство обнаружения фронта. На рисунке 3.36 показана функциональная эквивалентная блок-диаграмма логики синхронизации и обнаружения фронта T1/T0. Регистры синхронизируются при положительном фронте внутренней синхронизации системы ($clk_{I/O}$). Защелка прозрачна в период нахождения внутренней системы синхронизации в высоком логическом состоянии. Устройство обнаружения фронта генерирует один импульс clk_{T1}/clk_{T0} для каждого положительного CSn(2–0) = 7 или отрицательного CSn(2–0) = 6 фронта, который оно обнаруживает.

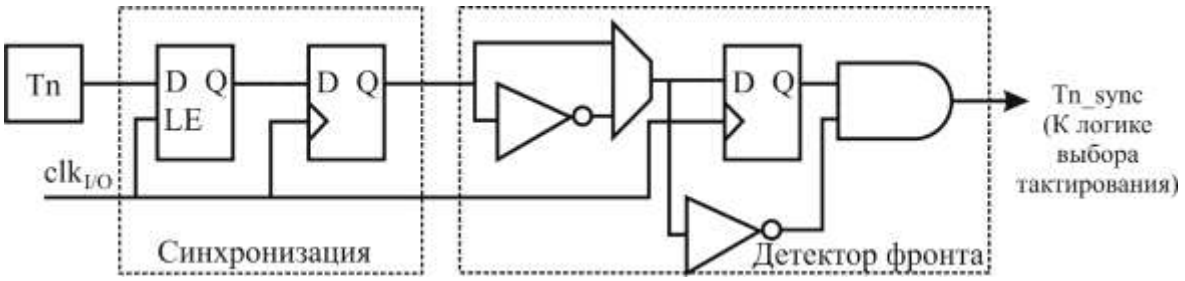


Рисунок 3.36 – Стробирование вывода T1/T0

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

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

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

Для источника внешнего сигнала синхронизации нельзя использовать предварительное деление частоты.

Регистр специальных функций входа/выхода – SFIOR

Бит	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	ADHSM	ACME	PUD	PSR2	PSR10	SFIOR
Чт./Зап.	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 0: PSR10 – переустановка устройства предварительного деления частоты таймера/счетчика 1 и таймера/счетчика 0

Когда разряд 0: PSR10 устанавливается, устройство предварительного деления частоты таймера/счетчика 1 и таймера/счетчика 0 будет сброшено. Разряд будет очищен аппаратным способом после завершения операции. Запись в этот разряд нуля не будет давать эффекта. Следует обратить внимание на то, что таймер/счетчик 1 и таймер/счетчик 0 одновременно используют тот же предварительный делитель частоты и переустановка этого устройства повлияет на оба таймера. Этот разряд будет всегда считываться как ноль.

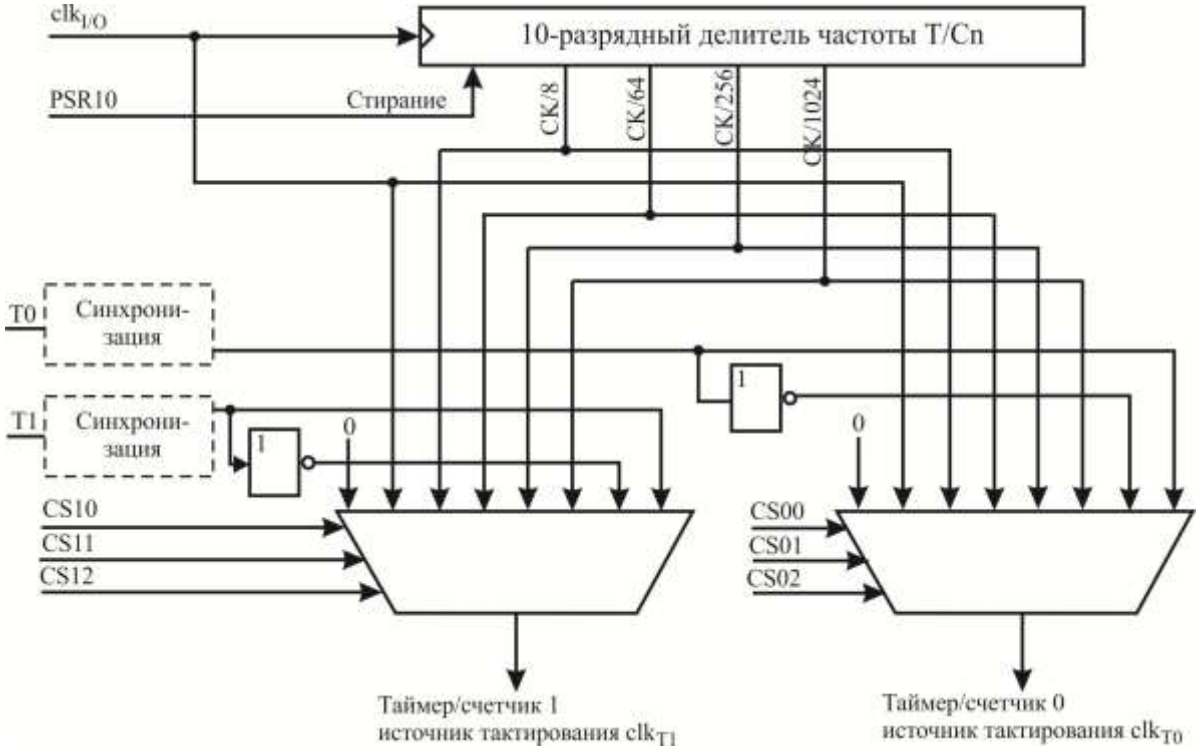


Рисунок 3.37 – Устройство предварительного деления частоты для таймера/счетчика 0 и таймера/счетчика 1

3.12 16-разрядный таймер/счетчик 1

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

- истинную 16-разрядную компоновку (т. е. допускает использование 16-раз-рядного ШИМ);
- два независимых блока сравнения по выходу;
- регистры сравнения по выходу с двойным буфером;
- один блок захвата по входу;
- устройство подавления шума захвата по выходу;
- сброс таймера по совпадению при сравнении (автоматическая перезагрузка);
- надежный ШИМ модулятор с фазовой коррекцией;
- регулируемый ШИМ период;
- генератор частоты;
- счетчик внешних событий;
- четыре независимых источника прерывания (TOV1, OCF1A, OCF1B и ICF1).

3.12.1 Обзор

Основные ссылки на регистры и разряды в подразделе 3.12 представлены в общем виде. Буква «n» заменяет номер таймера/счетчика, а буква «x» – номер канала сравнения по выходу. В то же время, когда использование регистра или разряда определено в программе, необходимо использовать точную форму, например, TCNT1 для значения доступа к таймеру/счетчику 1 и т. д.

3.12.2 Регистры

Таймер/счетчик (TCNT1), регистры выводов сравнения OCR1A, OCR1B и регистр захвата на входе (ICR1) – все являются 16-разрядными регистрами. При доступе к 16-разрядным регистрам необходимо соблюдать специальные процедуры. Регистры команд таймера/счетчика TCCR1A, TCCR1B являются 8-разрядными регистрами и не имеют никаких ограничений при доступе из ЦПУ. Запросы на прерывание (на рисунке 3.38 сокращенно «внутр. запрос») видны в регистре флагов прерывания таймера (TIFR). Все прерывания маскируются индивидуально с использованием регистра маскирования прерываний таймера (TIMSK). TIFR и TIMSK не указаны на рисунке, поскольку эти регистры используются одновременно другими модулями таймера.

Таймер/счетчик может быть синхронизирован внутри схемы через делитель частоты или с помощью источника внешнего тактового сигнала на выводе T1. Логическая схема выбора тактового генератора определяет, какой источник синхронизации и фронт использует таймер/счетчик, чтобы инкрементировать (или декрементировать) его значение. Таймер/счетчик не активен, если не выбран никакой источник синхронизации. Выход из логической схемы выбора источника тактирования определен как тактовый сигнал таймера (clk_{T1}).

Регистры сравнения по выходу с двойным буфером OCR1A, OCR1B постоянно сравниваются со значением таймера/счетчика. Результат сравнения может использоваться генератором сигнала для создания режима ШИМ или сигнала с переменной частотой на выводе OCR1A, OCR1B. Событие совпадения при сравнении также установит флаг соответствия при сравнении OCR1A, OCR1B который может использоваться для создания запроса на прерывание при сравнении.

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

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

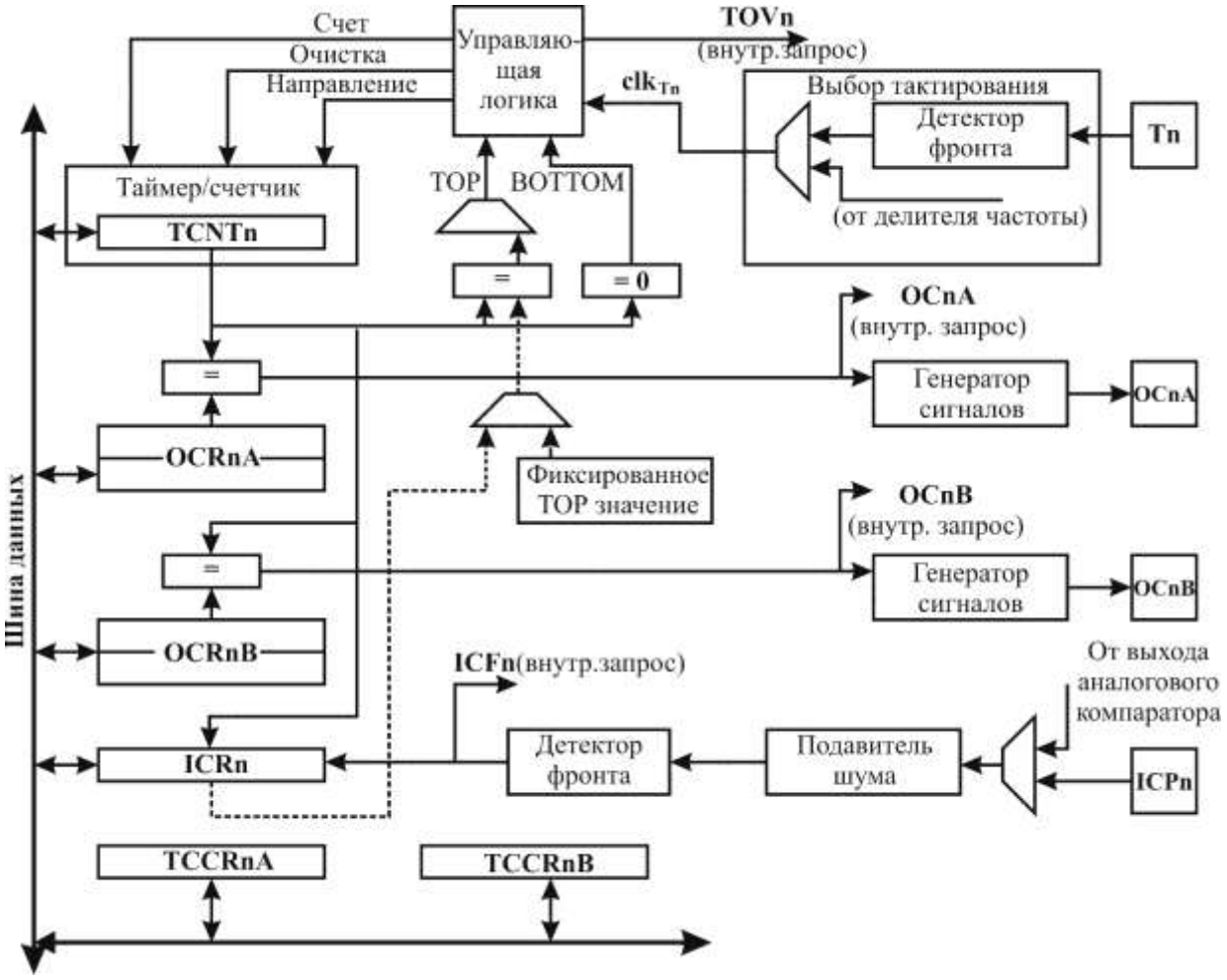


Рисунок 3.38 – Блок-схема 16-разрядного таймера/счетчика

3.12.3 Определения

В подразделе 3.12 достаточно часто используются определения, представленные в таблице 3.42.

Т а б л и ц а 3.42

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

Доступ к 16-разрядным регистрам

TCNT1, OCR1A, OCR1B и ICR1 – 16-разрядные регистры, к которым может обращаться ЦПУ через 8-битовую шину данных. Доступ к 16-разрядному регистру должен осуществляться по байтам, используя две операции считывания или записи. У каждого 16-разрядного таймера есть один 8-битовый регистр для временного хранения старшего байта 16-разрядного доступа. Тот же самый временный регистр используется совместно

всеми 16-разрядными регистрами для каждого 16-разрядного таймера. Доступ к младшему байту запускает операцию 16-разрядного считывания или записи. Когда младший байт 16-разрядного регистра записывается ЦПУ, то старший байт, сохраненный во временном регистре, и записанный младший байт оба копируются в 16-разрядный регистр в том же самом тактовом цикле. Когда младший байт 16-разрядного регистра считывается ЦПУ, то старший байт 16-разрядного регистра копируется во временный регистр в том же самом тактовом цикле, пока считывается младший байт.

Не все виды 16-разрядного доступа используют временный регистр для старшего байта. При считывании OCR1A, OCR1B 16-разрядные регистры не используют временный регистр.

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

Следующие примеры кода показывают как обратиться к 16-разрядным регистрам таймера, полагая при этом, что никакие прерывания не обновляют временный регистр. Тот же самый принцип может использоваться непосредственно для доступа к регистрам OCR1A, OCR1B и ICR1. Следует обратить внимание на то, что при использовании «С» кода, компилятор обрабатывает 16-разрядный доступ.

Ассемблерный код:

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

С код:

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

П р и м е ч а н и е – Данный пример кода предполагает, что включен характерный для устройства заголовочный файл.

Пример кода ассемблера возвращает значение TCNT1 в пару регистров r17–r16.

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

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

Ассемблерный код:

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

С код:

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

П р и м е ч а н и е – Данный пример кода предполагает, что включен характерный для устройства заголовочный файл.

Пример кода ассемблера возвращает значение TCNT1 в пару регистров r17–r16.

Следующие примеры кодов показывают, как выполнять поэлементную запись в содержимое регистров TCNT1. Запись любого из регистров OCR1A, OCR1B или ICR1 может быть выполнена с использованием того же принципа.

Ассемблерный код:

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

С код:

```
void TIM16_WriteTCNT1( unsigned int i )
(
unsigned char sreg;
/* Save Global Interrupt Flag */
sreg = SREG;
/* Disable interrupts */
```

```
_CLI();
/* Set TCNT1 to i */
TCNT1 = i;
/* Restore Global Interrupt Flag */
SREG = sreg;
)
```

П р и м е ч а н и е – Данный пример кода предполагает, что включен характерный для устройства заголовочный файл.

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

Повторное использование временного регистра старшего байта

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

3.12.4 Источники синхронизации таймера/счетчика

Таймер/счетчик можно синхронизировать с помощью внутреннего или внешнего источника синхронизации. Источник синхронизации выбирается с помощью логики выбора синхронизации, которая управляется разрядами выбора синхронизации CS(12–0), расположенными в регистре В управления таймером/счетчиком (TCCR1B).

3.12.5 Модуль счетчика

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

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

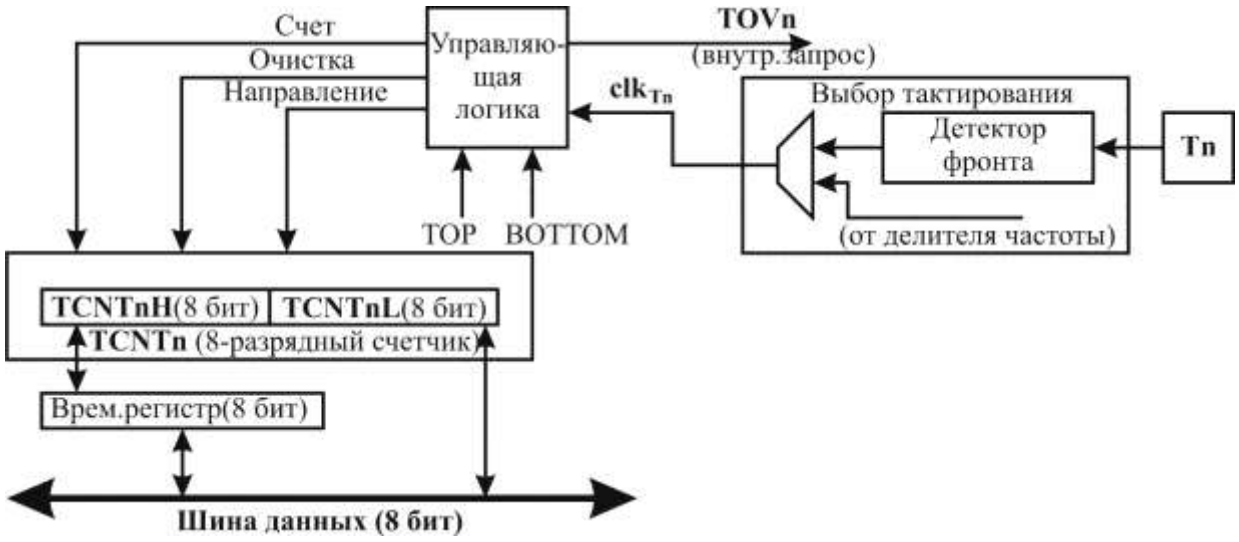


Рисунок 3.39 – Блок-схема модуля счетчика

Описание сигналов (внутренние сигналы):

Счет	- возрастание или уменьшение TCNT1 на ;
Направление	- выбор между возрастанием или убыванием;
Очистка	- сброс TCNT1 (установка всех разрядов на ноль);

clk_{T1}	- тактовый генератор таймера/счетчика;
TOP	- указывает на то, что TCNT1 достиг верхнего значения;
BOTTOM	- указывает на то, что TCNT1 достиг нижнего значения (ноль).

16-разрядный счетчик размещен в двух 8-битовых регистрах памяти ввода-вывода: счетчик старшего байта (TCNT1H), содержащий старшие восемь бит счетчика, и счетчик младшего байта (TCNT1L), содержащий младшие восемь бит. К регистру TCNT1H можно обратиться с помощью ЦПУ только косвенно. Когда центральный процессор осуществляет доступ к регистрам ввода-вывода TCNT1H, центральный процессор обращается к регистру временного хранения (TEMP) старшего байта. Содержимое регистра временного хранения переписывается значением из TCNT1H при считывании TCNT1L, содержимое TCNT1H обновляется значением регистра временного хранения, когда TCNT1L записывается. Это позволяет центральному процессору читать или записывать все значения 16-разрядного счетчика в пределах одного тактового цикла через 8-битовую шину данных. Важно обратить внимание на то, что особые случаи записи в регистр TCNT1, когда счетчик находится в процессе счета, дают непредсказуемые результаты. Особые случаи описаны в разделах, для которых они существенны.

В зависимости от используемого режима работы счетчик сбрасывается, инкрементирует или декрементирует свои значения при каждом такте синхронизации таймера (clk_{T1}). Такт clk_{T1} может быть сформирован либо от внешнего, либо от внутреннего источника тактового сигнала и выбирается битами выбора тактового сигнала CS1(2–0). Если не выбран никакой из источников CS1(2–0) = 0, таймер останавливается. Однако TCNT1 может быть доступен ЦПУ независимо от того, присутствует clk_{T1} или нет. Запись ЦПУ отменяет (имеет приоритет над ними) все операции сброса или счета. Последовательность счета определяется установкой разрядов режима работы генерации сигнала WGM1(3–0), расположенных в регистрах управления таймерами/счетчиками A и B (TCCR1A и TCCR1B). Существует выраженная взаимосвязь между тем, как счетчик ведет себя (считает) и тем, какие сигналы генерируются на выводах «выхода сравнения» OC1x. Флаг переполнения таймера/счетчика TOV1 устанавливается согласно режиму работы, который выбирается разрядами WGM1(3-0). TOV1 может использоваться для генерации прерывания ЦПУ.

Модуль захвата данных на входе

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

Модуль сбора данных на входе проиллюстрирован блок-схемой, показанной на рисунке 3.40. Буква «n» в названиях регистров и разрядов указывает номер таймера/ счетчика.

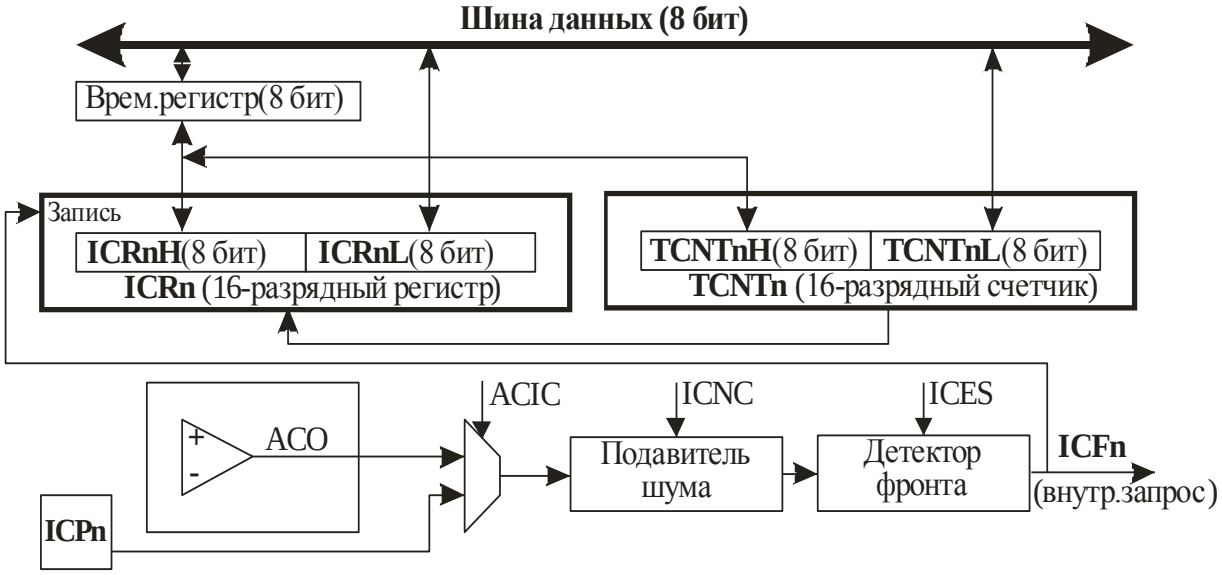


Рисунок 3.40 – Блок-схема модуля сбора данных на входе

Когда изменение логического уровня (событие) происходит на выводе «захват данных на входе» ICP1 или, альтернативно, на выходе аналогового компаратора ACO, и если это изменение подтверждает установку детектора фронта, то будет срабатывать захват данных. Когда захват срабатывает, то 16-разрядное значение счетчика (TCNT1) записывается в регистр захвата данных на входе (ICR1). Флаг захвата данных на входе (ICF1) устанавливается во время того же системного импульса синхронизации, поскольку значение TCNT1 копируется в регистр ICR1. Если получено разрешение (TICIE1 = 1), то флаг захвата данных на входе генерирует прерывание захвата. Флаг ICF1 автоматически сбрасывается при выполнении процедуры обработки прерывания. Также флаг ICF1 может быть очищен программным способом путем записи логической единицы в разряд флага.

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

Регистр ICR1 может записываться только когда применяется режим генерации сигнала, который использует регистр ICR1 для определения значения счетчика TOP. В этих случаях разряды для режима генерации формы сигнала WGM1(3–0) должны быть установлены до того, как значение TOP может быть записано в регистр ICR1. При записи регистра ICR1, старший байт должен быть записан в адрес ICR1H до того, как младший байт записывается в ICR1L.

Источник срабатывания сбора данных на входе

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

Оба входа – захвата данных на входе ICP1 и выхода аналогового компаратора ACO стробируются с использованием той же методики, что и для вывода T1. Устройство детектирования фронтов также идентично. В то же время, когда устройство шумоподавления запускается, перед устройством детектирования фронта встраивается дополнительная

логика, которая увеличивает задержку на четыре цикла системной синхронизации. Следует обратить внимание на то, что вход устройства шумоподавления и устройство обнаружения фронта всегда разрешены до тех пор, пока таймер/счетчик устанавливается в режим генератора формы сигнала, который использует ICR1 для определения состояния TOP.

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

Устройство шумоподавления

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

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

Использование модуля захвата данных на входе

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

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

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

Измерение рабочего цикла внешних сигналов требует, чтобы фронт срабатывания изменялся после каждого захвата данных. Изменение чувствительности фронта должно выполняться как можно раньше после считывания регистра ICR1. После изменения фронта флаг захвата данных на входе (ICF1) должен очищаться программным способом (с помощью записи логической единицы в адрес разряда флага). Если выполняется только измерение частоты, то сброс флага ICF1 не требуется (при условии использования устройства обращения с прерываниями).

3.12.6 Модули сравнения по выходу

16-разрядный компаратор постоянно сравнивает TCNT1 с регистром сравнения по выходу (OCR1x). Если TCNT равно OCR1x, то компаратор выдает сигнал о совпадении. Это совпадение установит флаг выхода сравнения (OCF1x) во время следующего цикла синхронизации таймера. При получении разрешения (OCIE1x = 1) флаг выхода сравнения генерирует прерывание. Флаг OCF1x автоматически сбрасывается при выполнении прерывания. В качестве альтернативы флаг OCF1x может быть сброшен программно путем записи логической единицы в адрес его разряда. Генератор сигнала специальной формы использует сигнал совпадения для генерации выхода в соответствии с режимом работы,

который установился соответствующими разрядами режима генерации сигнала WGM1(3–0) и разрядами режима сравнения по выходу COM1x(1–0). Сигналы TOP и BOTTOM используются генератором сигнала специальной формы для обработки особых случаев крайних значений для некоторых режимов работы.

Специальная функция модуля A выхода сравнения позволяет ему определять величину TOP таймера/счетчика (т. е. разрешающую способность). В дополнение к разрешающей способности величина TOP определяет период времени для сигнала, формируемого генератором сигнала специальной формы.

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

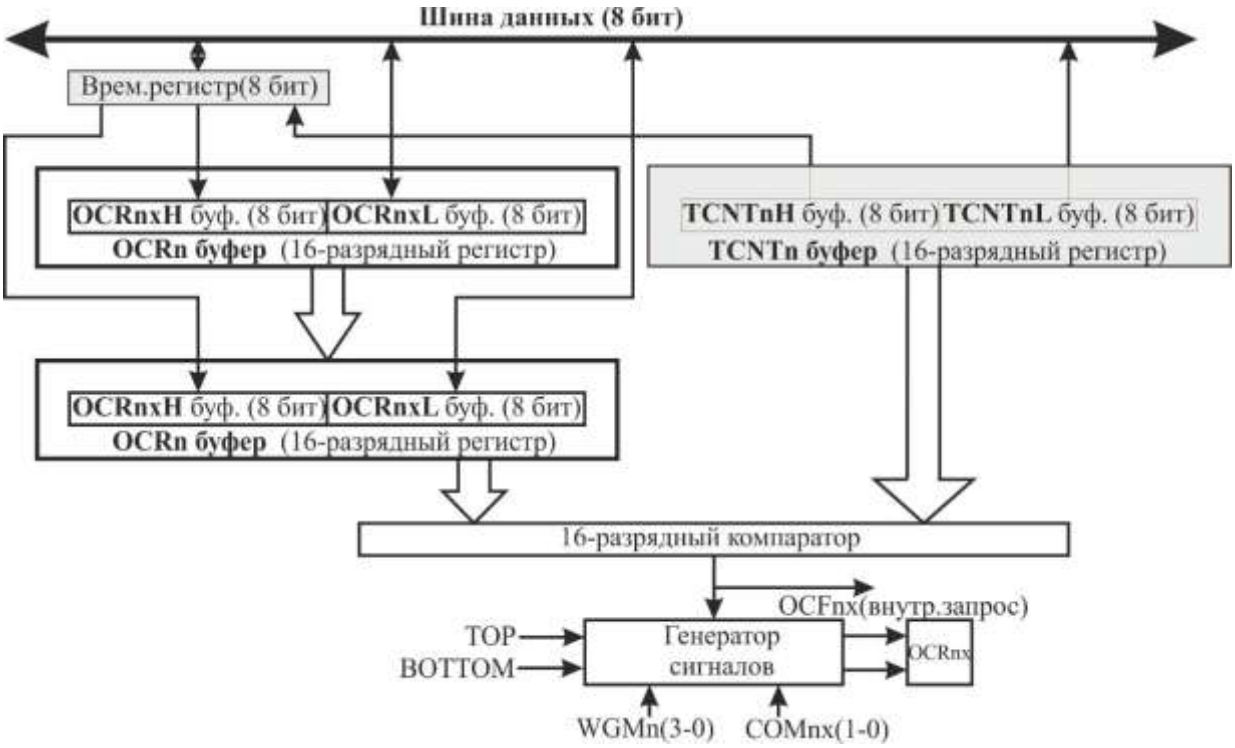


Рисунок 3.41 – Блок-схема модуля сравнения по выходу

Регистр OCR1x имеет двойную буферизацию при использовании любого из двенадцати режимов ШИМ. В нормальном режиме и режиме очистки таймера при сравнении (режим CTC) двойная буферизация отключается. Двойная буферизация синхронизирует обновление регистра сравнения до TOP или BOTTOM в последовательности считывания. Синхронизация предотвращает появление несимметричных импульсов избыточной длины, обеспечивая на выходе сигналы, свободные от помех.

Доступ к регистру OCR1x может показаться сложным, однако, это не так. При запуске двойной буферизации ЦПУ получает доступ к регистру буфера OCR1x, а если двойная буферизация будет отключена, то ЦПУ получит прямой доступ к OCR1x. Содержимое регистра OCR1x изменяется только при операции записи (таймер/счетчик не обновляет этот счетчик автоматически, так как это выполняет TCNT1 – и регистр ICR1). Вследствие этого OCR1x не считывается через регистр временного хранения старшего байта (TEMP). Однако, исходя из нормальной практики, следует сосчитать сначала младший байт, также как при доступе к 16-разрядным регистрам. Запись в регистры OCR1x должна выполняться через регистр TEMP, поскольку сравнение всех 16 разрядов выполняется непрерывно. Сначала должен записываться старший байт (OCR1xH). Когда ЦПУ записывает значение старшего байта, содержимое регистра TEMP будет обновлено на записанное значение. За-

тем, когда в младшие восемь разрядов записывается младший байт (OCR1xL), старший байт будет скопирован в старшие восемь разрядов либо буфера OCR1x, либо регистра сравнения OCR1x в течение того же цикла синхронизации.

3.12.7 Принудительное совпадение

При работе в режимах, не относящихся к ШИМ режимам, выход совпадения компаратора может быть принудительно установлен при записи единицы в разряд принудительного сравнения (FOC1x). Принудительное совпадение не установит флаг OCF1x и не перегрузит/очистит таймер, но вывод OC1x будет обновлен, как если бы произошло реальное совпадение при сравнении (состояние COM1(1–0) определяет: установлен вывод OC1x, очищен или переключен).

3.12.8 Блокировка совпадения при сравнении с помощью записи TCNT1

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

3.12.9 Использование модуля сравнения по выходу

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

Установка OC1x должна быть выполнена до настройки регистра направления данных вывода порта, как "выход данных". Самый простой способ установки значения OC1x состоит в том, чтобы использовать биты FOC1x в нормальном режиме. Регистр OC1x сохраняет свое значение, даже если он переходит в различные режимы генерации сигнала произвольной формы.

Необходимо убедиться в том, что биты COM1x(1–0) не имеют двойной буферизации совместно со значением сравнения. Изменение битов COM1x(1–0) немедленно вступит в силу.

3.12.10 Модуль сравнения/совпадения по выходу

Разряды режима выхода сравнения COM1x(1–0) имеют две функции. Генератор сигнала произвольной формы использует разряды COM1x(1–0) для определения состояния выхода сравнения (OC1x) при последующем совпадении при сравнении. Вторая функция состоит в том, что разряды COM1x(1–0) управляют источником выхода для вывода OC1x. На рисунке 3.42 показана упрощенная логическая схема, запускаемая с помощью установки разряда COM1x(1–0), показаны регистры, разряды и выводы входа/выхода. В данном случае показаны только те части общих регистров управления порта входа/выхода (DDR и PORT), на которые воздействуют разряды COM1x(1–0). При ссылке на состояние OC1x,

эта ссылка относится к внутреннему регистру OC1x, а не к выводу OC1x. Если происходит сброс системы, то регистр OC1x сбрасывается в ноль.

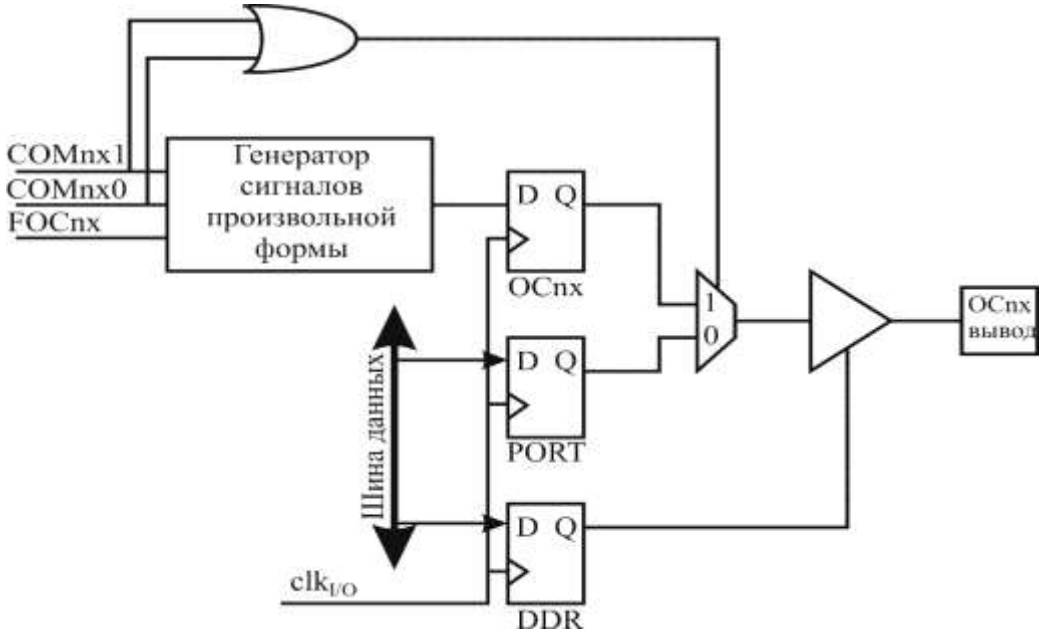


Рисунок 3.42 – Схема модуля сравнения/совпадения по выходу

Общая функция порта входа/выхода переписывается сигналом выхода сравнения OC1x из генератора сигнала произвольной формы, если установлен любой из разрядов COM1x(1–0). Однако для вывода порта направление вывода OC1x (вход или выход) по-прежнему контролируется регистром направления данных (DDR). Бит регистра направления данных для вывода OC1x (DDR_OC1x) должен быть установлен как выход до того, как значение OC1x станет присутствовать на выводе. Порт переопределяет функцию в общем виде независимо от режима генерации сигнала произвольной формы, но с некоторыми исключениями.

Логика для вывода сравнения позволяет выполнять инициализацию состояния OC1x до того, как вывод разрешен. Следует обратить внимание на то, что некоторые установки разрядов COM1x(1–0) резервируются для определенных режимов работы.

Биты COM1x(1–0) никак не влияют на модуль захвата данных на входе.

3.12.11 Режим сравнения по выходу и режим генерации сигнала произвольной формы

Генератор сигнала произвольной формы использует биты COM1x(1–0) по-разному в нормальном режиме, режиме CTC и режиме широтно-импульсной модуляции. Для всех режимов установка значений COM1x(1–0) = 0 сообщает генератору сигнала произвольной формы о том, что в отношении регистра OC1x не следует выполнять никаких действий при следующем совпадении при сравнении. Изменение состояния разрядов COM1x(1–0) повлияет на первое совпадение при сравнении после того, как биты записаны. Для режимов, не относящихся к ШИМ, совпадение может быть вынужденным путем использования разрядов FOC1x.

3.12.12 Режимы работы

Режим работы, то есть поведение на выводах таймера/счетчика и выводе «выход сравнения», определяется комбинацией разрядов для режима WGM1(3–0) генерации сигнала произвольной формы и режима сравнения по выходу COM1x(1–0). Разряды режима сравнения по выходу не затрагивают последовательность подсчета, в то время как разряды режима генерации сигнала произвольной формы влияют на него. Разряды COM1x(1–0) определяют, должен ли быть инвертирован или нет сгенерированный выходной ШИМ сигнал. Для режимов без ШИМ биты COM1x(1–0) управляют действиями в отношении того, должен ли выход быть установлен, очищен или переключен по совпадению при сравнении.

Обычный режим работы

Самым простым режимом работы является обычный режим WGM1(3–0) = 0. В этом режиме направление счета всегда следует по возрастанию, и при этом не выполняется очистка счетчика. Счетчик просто переполняется при прохождении своего максимального 16-битного значения (MAX = 0xFFFF), а затем перезапускается из BOTTOM (0x0000). В нормальном режиме работы флаг переполнения таймера/счетчика (TOV1) будет установлен в течение того же цикла синхронизации таймера, когда TCNT1 становится равным нулю.

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

Модуль захвата данных на входе можно легко использовать в нормальном режиме. При этом необходимо обратить внимание на то, что максимальный интервал между внешними событиями не должен превышать разрешающую способность счетчика. Если интервал между событиями слишком велик, то прерывание переполнения счетчика или предделитель должны быть использованы для повышения разрешающей способности модуля захвата данных.

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

Очистка таймера в режиме совпадения при сравнении (CTC)

В режиме очистки таймера при сравнении или режиме CTC (WGM1(3–0) = 4 или 12), регистры OCR1A или ICR1 используются для управления разрешающей способностью счетчика. В режиме CTC счетчик сбрасывается до нуля, когда значение счетчика (TCNT1) совпадает либо с OCR1A WGM1(3–0) = 4, либо с ICR1 WGM1(3–0) = 12. OCR1A или ICR1 определяют верхнее значение счетчика, а следовательно, его разрешающую способность. Этот режим обеспечивает лучшее управление выходной частоты совпадения при сравнении. Он также упрощает операцию счета внешних событий.

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

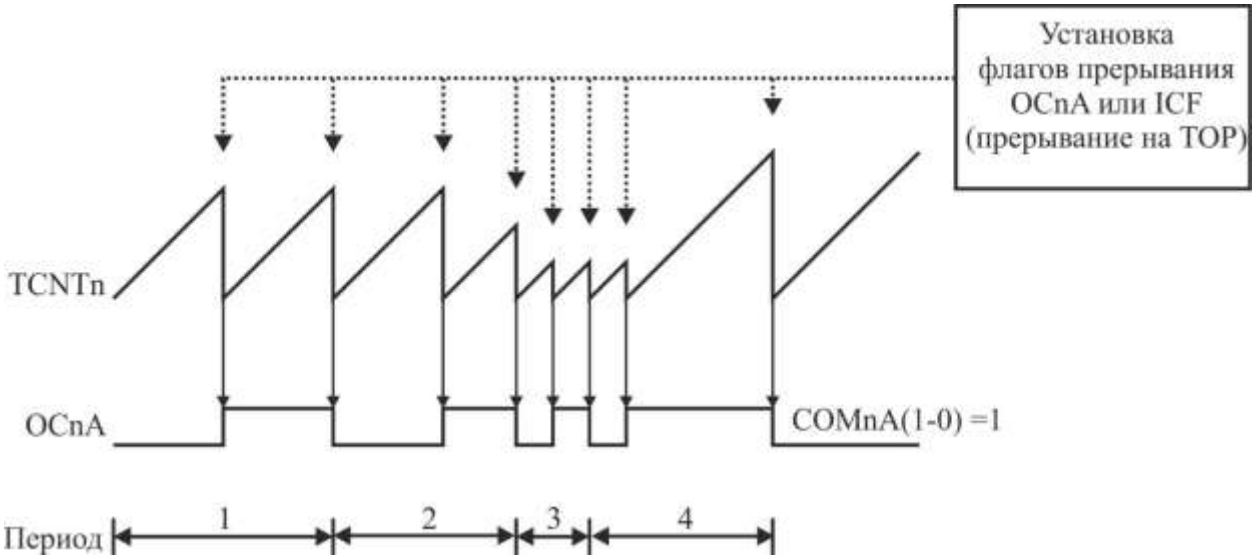


Рисунок 3.43 – Режим CTC, временная диаграмма

Прерывание может быть сгенерировано каждый раз, когда значение счетчика достигает TOP благодаря использованию флага OCF1A или ICF1 в соответствии с используемым регистром для определения значения TOP. Если прерывание разрешено, то подпрограмма обработчика прерываний может использоваться для того, чтобы обновить значение TOP. При этом изменение значения величины TOP на величину, близкую к BOTTOM, когда счетчик работает с малым значением предделителя или без него, должно выполняться очень осторожно, поскольку режим CTC не имеет возможностей для двойной буферизации. Если новое значение, записанное в OCR1A или ICR1, ниже, чем текущее значение TCNT1, то счетчик пропустит совпадение при сравнении. Счетчик должен будет тогда выполнить счет до его максимального значения (0xFFFF) и сделать сброс, начав со значения 0x0000 прежде, чем сможет произойти совпадение при сравнении. Во многих случаях эта особенность не желательна. В качестве альтернативы тогда необходимо будет использовать быстродействующий режим ШИМ, используя OCR1A для того, чтобы определить TOP WGM1(3–0) = 15, поскольку OCR1A тогда будет иметь двойную буферизацию.

Для генерации сигнала произвольной формы в режиме CTC, выход OC1A можно установить для переключения его логического уровня при каждом совпадении путем установки разрядов режима сравнения по выходу для режима переключения COM1A(1–0) = 1. Величина OC1A не будет видна на выводе порта до тех пор, пока направление данных для вывода не будет установлено как выход (DDR_OC1A = 1). Генерируемая форма сигнала будет иметь максимальную частоту $f_{OC1A} = f_{clk\_I/O}/2$, когда OCR1A установится в ноль (0x0000). Частота формы сигнала определяется уравнением

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

В данном случае переменная «N» представляет собой коэффициент предварительного деления частоты (1, 8, 64, 256 или 1024).

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

Быстрый режим ШИМ

Быстрый режим ШИМ (WGM1(3–0) = 5, 6, 7, 14 или 15) предоставляет возможность генерации сигнала ШИМ с высокой частотой. Быстрый режим ШИМ отличается от других вариантов режимом работы по одному фронту. Счетчик выполняет счет от BOTTOM до TOP, затем вновь запускается с BOTTOM. В неинвертированном режиме сравнения по выходу (OC1x) вывод устанавливается по совпадению при сравнении TCNT1 и OCR1x и затем сбрасывается при TOP. В режиме инвертированного сравнения по выходу вывод сбрасывается по совпадению при сравнении и затем устанавливается при TOP. Благодаря режиму работы по одному фронту, рабочая частота в быстродействующем режиме ШИМ может быть в два раза выше, чем для режимов ШИМ с фазовой или фазо-частотной коррекцией, которые используют режим работы по двум фронтам. Такая высокая частота делает быстрый ШИМ режим очень приемлемым для регулировки мощности, выпрямления и применений для ЦАП. Высокая частота позволяет использовать внешние компоненты с малыми физическими размерами (катушки, конденсаторы), благодаря чему снижается общая стоимость системы.

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

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

В быстром ШИМ режиме значение счетчика возрастает до тех пор, пока не совпадет с одним из фиксированных значений 0x00FF, 0x01FF или 0x03FF (WGM1(3–0) = 5, 6 или 7), значением в ICR1 WGM1(3–0) = 14 или значением в OCR1A WGM1(3–0) = 15. Затем при следующем цикле синхронизации таймера счетчик сбрасывается. Временная диаграмма для быстрого режима ШИМ показана на рисунке 3.44. На рисунке показан быстрый режим ШИМ для случая, когда OCR1A или ICR1 используются для определения TOP. Значение TCNT1 на временной диаграмме показано как гистограмма для иллюстрации режима работы по одному фронту. На схеме показаны как неинвертированные, так и инвертированные выходы ШИМ. Небольшие горизонтальные линейные метки на наклонных участках TCNT1 представляют совпадение между OCR1x и TCNT1. Флаг прерывания OC1x установится, когда произойдет совпадение при сравнении.

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

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

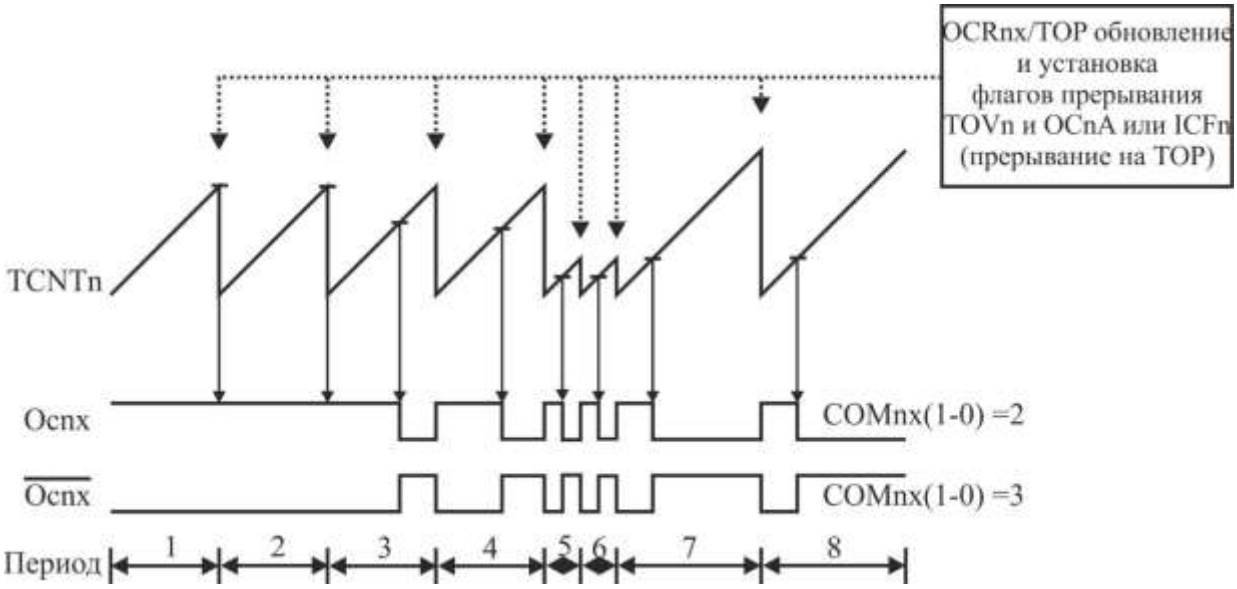


Рисунок 3.44 – Временная диаграмма, быстрый режим ШИМ

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

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

В быстром режиме ШИМ модули сравнения позволяют обеспечить генерацию сигнала ШИМ на выводах OC1x. Установка разрядов COM1x(1-0) в значение «10» создаст неинвертированный режим ШИМ и инвертированный выход ШИМ может генерироваться путем установки COM1x(1–0) в значение «11». Фактическое значение OC1x будет видимым на выводе порта, если направление данных для вывода порта установлено как выход (DDR_OC1x). Форма сигнала ШИМ генерируется путем установки (или очистки) регистра OC1x при совпадении OCR1x и TCNT1, и очистки (или установки) регистра OC1x каждый раз, когда счетчик обнуляется (изменения с TOP на BOTTOM).

Частоту ШИМ для выхода можно вычислить с помощью уравнения

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

Переменная «N» представляет собой значение предделителя (1, 8, 64, 256 или 1024). Крайние значения OCR1x регистра представляют специальные случаи, когда генерируется ШИМ сигнал в быстром режиме. Если OCR1x установлен равным BOTTOM (0x0000), выходной сигнал будет представлять узкий импульс для каждого TOP + 1 тактового импульса таймера. Установка OCR1x равным TOP приведет к постоянному высокому или низкому логическому уровню на выходе (в зависимости от полярности выхода, установленного битами COM1x(1–0). Частота (с 50 % скважностью) выходного сигнала в быстром ШИМ режиме может быть достигнута установкой бита OC1A, чтобы переключить его логический уровень при каждом совпадении COM1A(1–0) = 1. Сгенерированный сигнал будет иметь максимальную частоту $f_{OC1A} = f_{clk\_I/O}/2$, когда OCR1A будет равно нулю (0x0000). Эта особенность напоминает ситуацию, когда OC1A переключается в CTC режим, кроме двойной буферизации выхода модуля сравнения, разрешенного в быстром ШИМ режиме.

Режим фазовой коррекции

Режим фазовой коррекции ШИМ (WGM1(3–0) = 1, 2, 3, 10 или 11) обеспечивает высокое разрешение фазовой коррекции ШИМ сигнала. Фазовая коррекция ШИМ, подобно фазовой и частотной коррекции ШИМ режима, основывается на операции по двум фронтам. В неинвертирующем режиме сравнения, выход результата сравнения (OC1x) очищается при обнаружении совпадения между TCNT1 и OCR1x при инкременте и устанавливается, когда обнаруживается совпадение при декременте. В инвертирующем режиме сравнения производится операция инверсии. Операция по двум фронтам имеет максимальную частоту ниже, чем у операции по одному фронту. Однако, благодаря симметричным особенностям режима ШИМ по двум фронтам, эти режимы предпочтительней для приложений управления электродвигателями.

Разрешение ШИМ для режима фазовой коррекции может быть фиксированным 8-, 9- или 10-битным или определено либо ICR1, либо OCR1A. Минимальное разрешение – 2 бита (ICR1 или OCR1A устанавливаются в значение 0x0003), максимальное разрешение – 16 бит (ICR1 или OCR1A устанавливаются в значение MAX). Разрешение ШИМ может быть вычислено, используя выражение

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

В режиме фазовой коррекции ШИМ счетчик инкрементируется до тех пор, пока его значение на станет равно 0x00FF, 0x01FF или 0x03FF (WGM1(3–0) = 1, 2 или 3), значению в ICR1 WGM1(3–0) = 10 или значению в OCR1A WGM1(3–0) = 11. Счетчик достигает значения TOP и изменяет направление счета. Значение TCNT1 будет равно значению TOP в течение одного периода тактового сигнала таймера. Временная диаграмма режима фазовой коррекции представлена на рисунке 3.45. Рисунок представлен в виде гистограммы для иллюстрации операции по двум фронтам. Диаграмма содержит неинвертированный и инвертированный выходы ШИМ. Небольшая горизонтальная линия отмечает на TCNT1 положения, которые представляют совпадения при сравнении между OCR1x и TCNT1. Флаг прерывания OC1x будет установлен, когда произойдет совпадение.

Флаг переполнения таймера/счетчика (TOV1) устанавливается каждый раз, когда счетчик достигает значения BOTTOM. Если для определения значения TOP используется OCR1A или ICR1, то флаг OC1A или ICF1 устанавливается в тот же самый цикл тактирования, когда регистры OCR1x обновляются значениями двойной буферизации (на TOP). Флаги прерывания затем могут использоваться для генерации прерывания каждый раз, когда счетчик достигает значение TOP или BOTTOM.

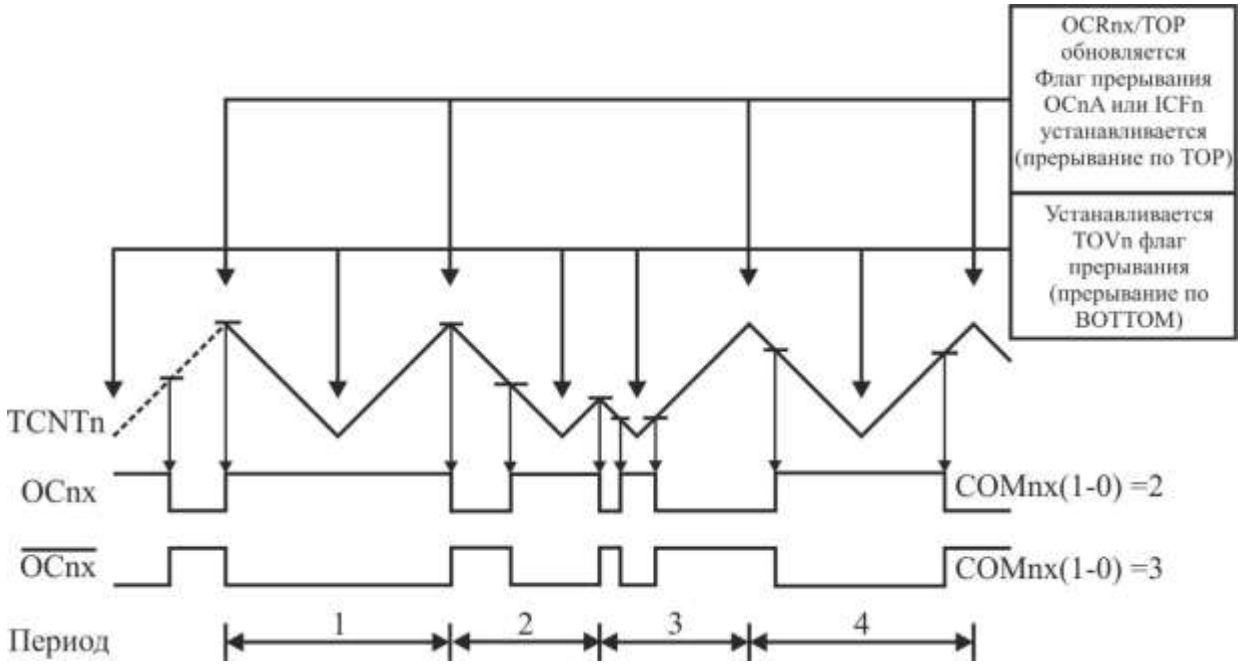


Рисунок 3.45 – Временная диаграмма режима фазовой коррекции ШИМ

При смене значения TOP программа должна обеспечить, чтобы новое значение TOP было больше или равно значению всех регистров сравнения. Если значение TOP меньше, чем любое значение регистров сравнения, то совпадение при сравнении никогда не произойдет между значениями TCNT1 и OCR1x. Необходимо заметить, что когда используется фиксированное TOP значение, неиспользуемые биты маскируются нулями, когда производится запись в любой из OCR1x регистров. Как показано в третьем периоде на рисунке 3.45, изменение значения TOP в то время, когда таймер/счетчик работает в режиме фазовой коррекции, может привести к формированию несимметричного сигнала на выходе. Причина этого может быть найдена во время обновления регистра OCR1x. Поскольку обновление регистра OCR1x происходит на TOP, период ШИМ начинается и заканчивается на TOP. Это подразумевает, что длина спадающего фронта определяется предыдущим значением TOP, в то время, как длина нарастающего фронта определяется новым значением TOP. Когда эти два значения различны, длина двух фронтов периода будут отличаться. Различие длин даст несимметричный результат на выходе.

Рекомендуется использовать режим коррекции фазы и частоты вместо режима фазовой коррекции, когда требуется изменять значение TOP во время работы таймера/счетчика. Когда используется статическое TOP значение, практически отсутствует разница между вышеуказанными режимами работы.

В режиме ШИМ с фазовой коррекцией модули сравнения позволяют обеспечить генерацию сигнала ШИМ на выводах OC1x. Установка разрядов COM1x(1–0) в значение «10» создаст неинвертированный сигнал ШИМ, при установке COM1x(1–0) в значение «11» будет сгенерирован инвертированный сигнал ШИМ (таблица 3.47). Фактическое значение OC1x будет присутствовать на выводе порта, если направление данных для вывода порта устанавливается как выход (DDR_OC1x). Форма сигнала ШИМ генерируется путем установки (или сброса) регистра OC1x при совпадении между OCR1x и TCNT1, когда значение счетчика инкрементируется, и сброса (или установки) регистра OC1x при совпадении между OCR1x и TCNT1, когда значение счетчика декрементируется. Частоту ШИМ для выхода при использовании ШИМ режима с фазовой коррекцией можно вычислить с помощью формулы

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

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

Режимы коррекции фазы и частоты ШИМ

Фазовая и частотная коррекция ШИМ (WGM1(3–0) = 8 или 9) обеспечивает генерации сигнала ШИМ с фазовой и частотной коррекцией, обладающей высокой разрешающей способностью. Режим фазовой и частотной коррекции подобен режиму фазовой коррекции ШИМ, который базируется на операции по двум фронтам. Счетчик считает сначала от значения BOTTOM до значения TOP, после чего меняет направление счета и движется от значения TOP к значению BOTTOM. В неинвертирующем режиме сравнения выход компаратора сбрасывается при обнаружении совпадений между TCNT1 и OCR1x при инкременте и устанавливается при обнаружении совпадения при декрементировании. В инвертирующем режиме сравнения результат будет представлен в инвертированном виде. Операция по двум фронтам обеспечивает более низкую максимальную частоту операции по сравнению с операцией по одному фронту. Однако, благодаря симметричным особенностям режима ШИМ по двум фронтам, эти режимы более предпочтительны для приложений управления электродвигателями.

Основным отличием между режимами фазовой коррекции и фазочастотной коррекции является время обновления регистра OCR1x буферным регистром.

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

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

В фазочастотном режиме коррекции ШИМ счетчик инкрементируется до тех пор, пока его значение не станет равным значению в ICR1 WGM1(3–0) = 8 или значению в OCR1A WGM1(3–0) = 9. Счетчик достигает значения TOP и изменяет направление счета. Значение TCNT1 будет равным TOP в течение одного периода тактового сигнала таймера. Временная диаграмма фазовой коррекции и частотной коррекции представлена на рисунке 3.46. На рисунке показан режим фазочастотной коррекции ШИМ, когда OCR1A или ICR1 используются для определения TOP. Значение TCNT1 на временной диаграмме показано в виде гистограммы для иллюстрации операции по двум фронтам. Небольшая горизонтальная линия отмечает на TCNT1 моменты, которые представляют операцию сравнения между OCR1x и TCNT1. Флаг прерывания OC1x будет установлен, когда произойдет совпадение.

Флаг переполнения T/C (TOV1) устанавливается в том же цикле тактового сигнала таймера, когда OCR1x регистры обновляются значением двойного буфера (BOTTOM). Когда либо OCR1A, либо ICR1 используются для определения значения TOP, OC1A или ICF1, флаг установится, когда TCNT1 достигнет TOP. Флаги прерывания могут быть ис-

пользованы для формирования прерываний каждый раз, когда счетчик достигнет значения TOP или BOTTOM.

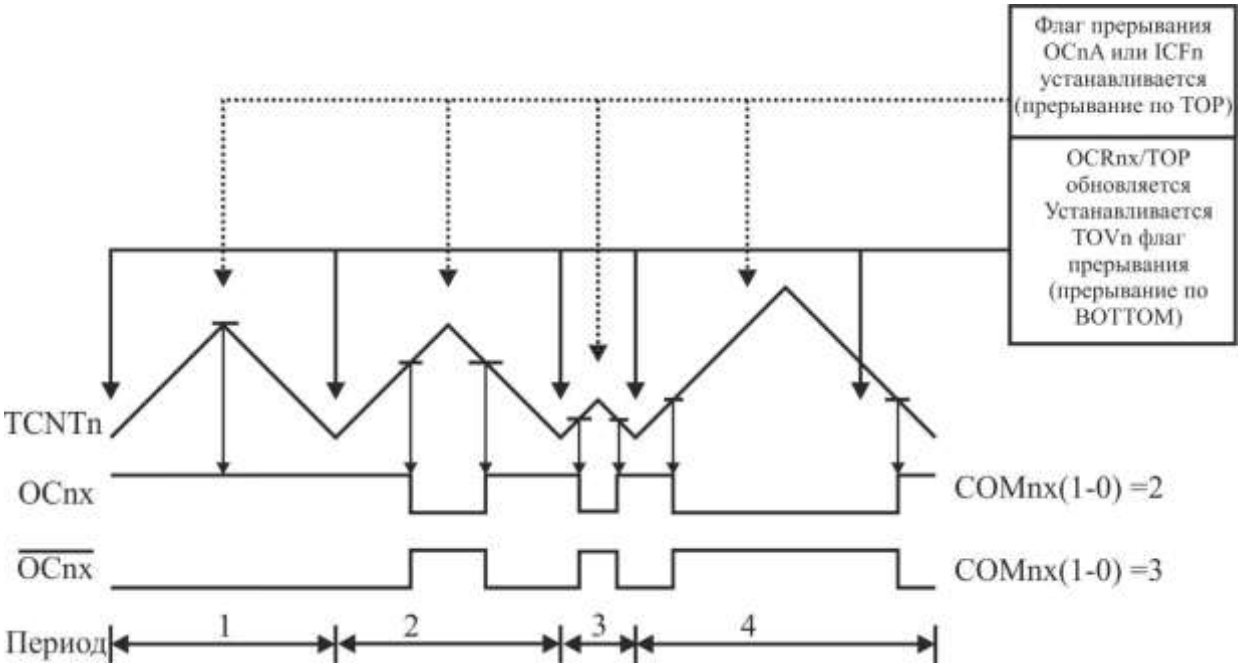


Рисунок 3.46 – Временная диаграмма режима фазовой и частотной коррекции

Когда изменяется значение TOP, программа должна гарантировать, что новое значение TOP будет больше или равно значениям всех регистров сравнения. Если значение TOP ниже, чем значение любого регистра сравнения, совпадение никогда не произойдет между TCNT1 и OCR1x.

На рисунке 3.46 показан сгенерированный выходной сигнал, который в противоположность режиму фазовой коррекции, симметричен весь период. С момента, когда содержимое OCR1x регистра обновляется значением BOTTOM, длина нарастающего и спадающего фронтов всегда будет одинакова. Это приводит к симметричности выходных импульсов с корректной частотой.

Использование ICR1 регистра для определения TOP хорошо работает при использовании фиксированных значений TOP. Использованием регистра ICR1 обеспечивается освобождение регистра OCR1A для генерации ШИМ выходного сигнала на OC1A. Однако, если базовая частота ШИМ активно изменяется сменой значения TOP, использование OCR1A как TOP является несомненно лучшим выбором благодаря особенности двойной буферизации. В фазочастотном режиме коррекции ШИМ модули сравнения позволяют генерировать ШИМ сигнал на OC1x выводах. Установка COM1x(1–0) битов в значение «10» будет давать на выходе неинвертированный ШИМ сигнал, установка же COM1x(1–0) битов в значение «11» приведет к генерации инверсного ШИМ сигнала. Действительное значение OC1x будет доступно только на выводах порта, если порт будет сконфигурирован на вывод данных. ШИМ сигнал генерируется установкой (или очисткой) OC1x регистра при обнаружении совпадения между регистрами OCR1x и TCNT1 при инкрементировании счетчика, и очисткой (или установкой) OC1x регистра при обнаружении совпадения между регистрами OCR1x и TCNT1 при декрементировании счетчика. Частота сигнала ШИМ при использовании режима фазочастотной коррекции может быть вычислена, используя выражение

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

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

Временные диаграммы таймера/счетчика

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

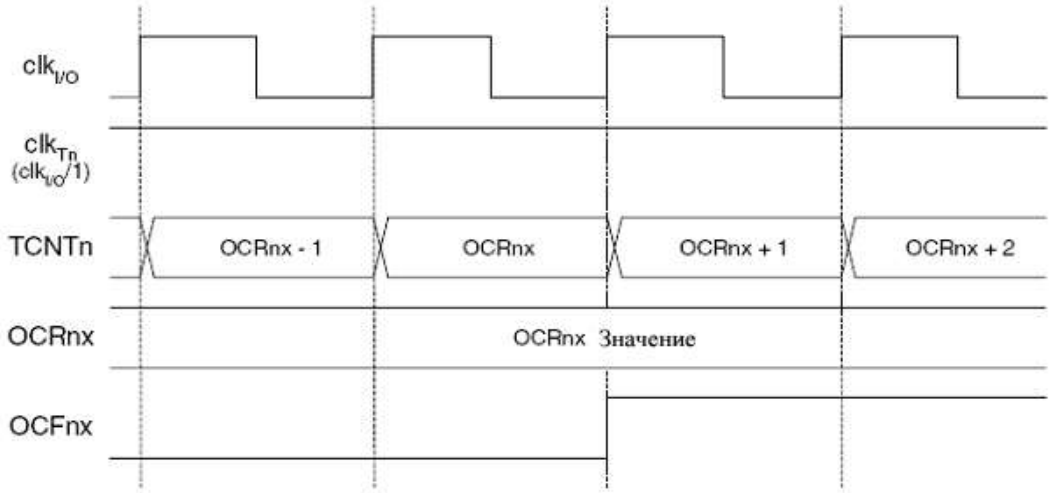


Рисунок 3.47 – Временная диаграмма таймера/счетчика, установка OCF1x без предварительного деления частоты

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

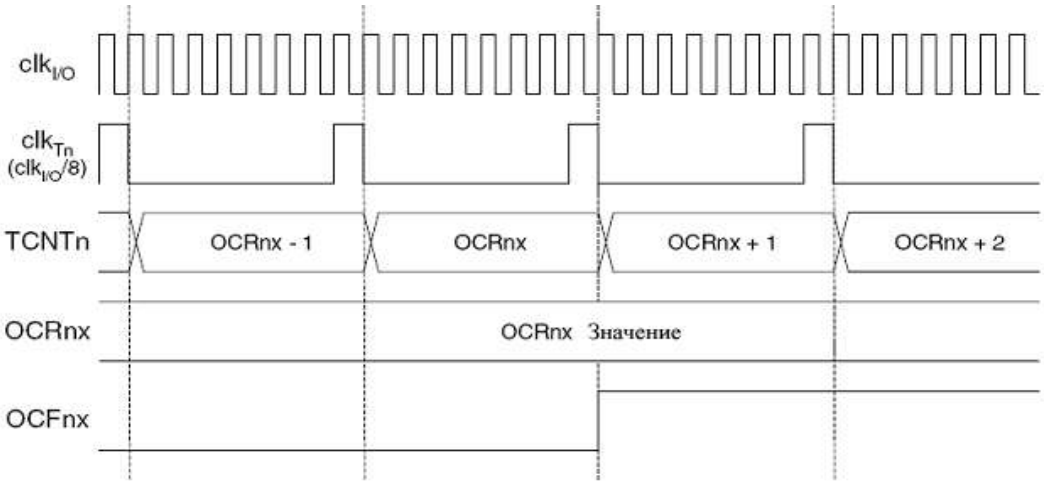


Рисунок 3.48 – Временная диаграмма таймера/счетчика, установка OCF1x с предварительным делением частоты

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

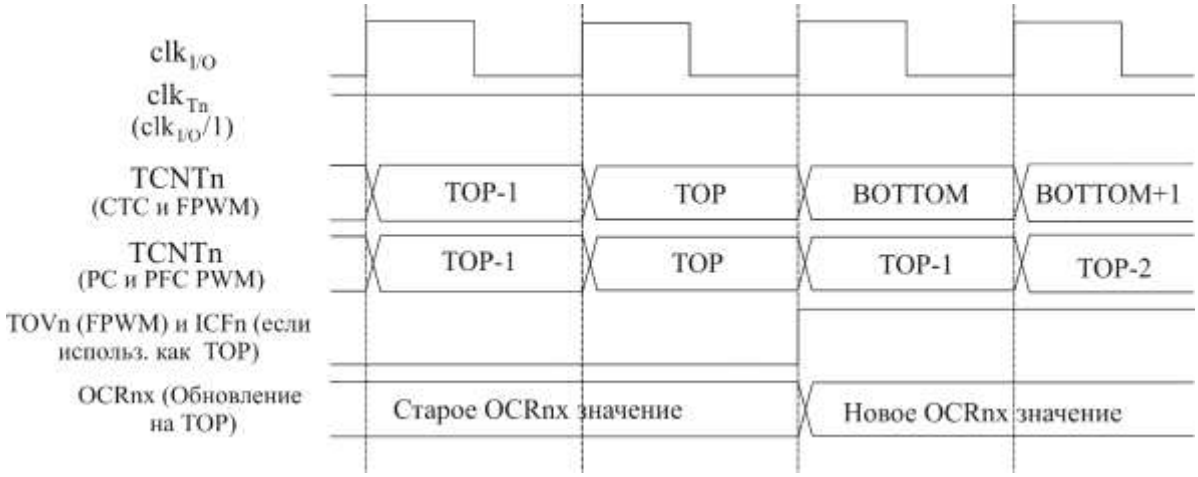


Рисунок 3.49 – Временная диаграмма таймера/счетчика без устройства предварительного деления частоты

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

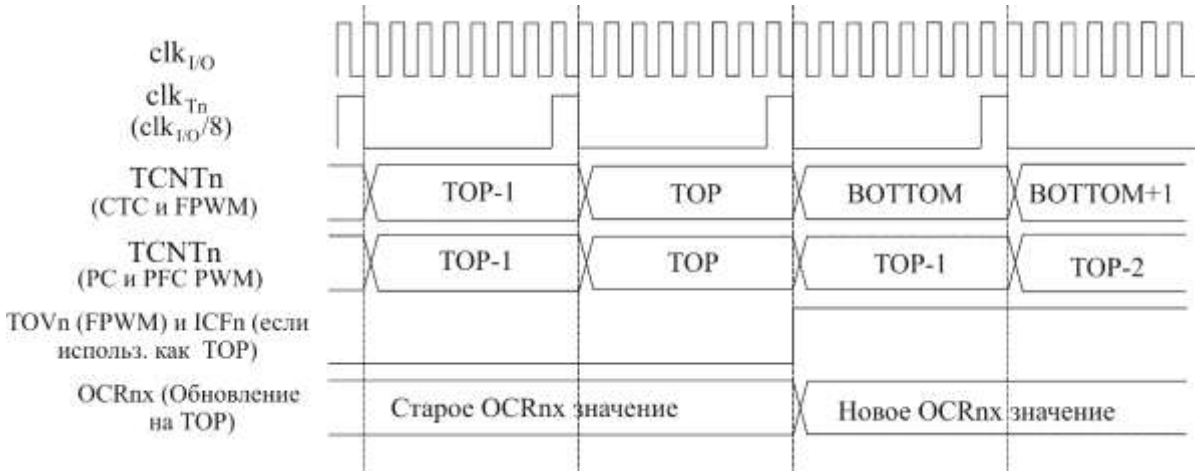


Рисунок 3.50 – Временная диаграмма таймера/счетчика с устройством предварительного деления частоты ($f_{clk\_I/O}/8$)

3.12.13 Описание регистров 16-разрядного таймера/счетчика

Регистр A управления таймера/счетчика 1 – TCCR1A

Бит	**7**	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Чт./Зап.	**З**	Ч/З	Ч/З	Ч/З	З	З	Ч/З	Ч/З	
Начальное значение	**0**	0	0	0	0	0	0	0	

Разряды 7, 6: COM1A(1–0) – режим сравнения по выходу для канала A
Разряды 5, 4: COM1B(1–0) –Режим сравнения по выходу для канала B

COM1A(1–0) и COM1B(1–0) управляют поведением выводов сравнения по выходу (соответственно, OC1A и OC1B). Если один или оба бита COM1A(1–0) устанавливаются, то выход OC1A меняет нормальное функциональное назначение порта входа/выхода, к которому он подсоединен. При этом следует обратить внимание на то, что разряд регистра направления данных (DDR), соответствующий выводу OC1A или OC1B, должен быть установлен таким образом, чтобы разрешить драйвер выхода. Если OC1A или OC1B подсоединены к выводу, то функция разрядов COM1x(1–0) зависит от установки разрядов WGM1(3–0). В таблице 3.43 показываются функции разрядов COM1x(1–0), когда разряды WGM1(3–0) установлены в нормальный режим или режим CTC (не режим ШИМ).

Т а б л и ц а 3.43 – Режим сравнения по выходу, не режим ШИМ

COM1A1, COM1B1	COM1A0, COM1B0	Описание
0	0	Нормальная работа порта, OC1A, OC1B отключены
0	1	Переключение OC1A, OC1B по совпадению при сравнении
1	0	Сброс OC1A, OC1B по совпадению при сравнении (выход устанавливается в низкий логический уровень)
1	1	Установка OC1A, OC1B по совпадению при сравнении (выход устанавливается в высокий логический уровень)

В таблице 3.44 показывается функциональность разряда COM1x(1–0) при установке разрядов WGM1(3–0) в быстрый режим ШИМ.

Т а б л и ц а 3.44 – Режим сравнения по выходу, быстрый режим ШИМ [1)]

COM1A1, COM1B1	COM1A0, COM1B0	Описание
0	0	Нормальная работа порта, OC1A/OC1B отключены
0	1	WGM1(3-0) = 15 Переключение OC1A по совпадению при сравнении, OC1B отключен (нормальный режим работы порта). Для других значений WGM1 нормальные операции портов OC1A/OC1B отключены
1	0	Очистка OC1A/OC1B по совпадению при сравнении, установка OC1A/OC1B на TOP
1	1	Установка OC1A/OC1B по совпадению при сравнении, очистка OC1A/OC1B на TOP
[1)] Особое место имеет случай, когда OCR1A/OCR1B равняется TOP и установлено COM1A1/COM1B1. В этом случае совпадение при сравнении игнорируется, однако установка или сброс выполняются на TOP.		

В таблице 3.45 показаны функции разряда COM1x(1–0), когда разряды WGM1(3–0) установлены в режим ШИМ с фазовой коррекцией или же с фазочастотной коррекцией.

Таблица 3.45 – Режим сравнения по выходу, фазовая и фазочастотная коррекция ШИМ

COM1A1, COM1B1	COM1A0, COM1B0	Описание
0	0	Общая функция порта, OC1A/OC1B отключены
0	1	WGM13=0: Общая функция порта, OC1A/OC1B отключены WGM13=1: Включение OC1A в режим сравнения, OC1B зарезервирован
1	0	Сброс OC1A/OC1B при совпадении (инкремент счетчика) Установка OC1A/OC1B при совпадении (декремент счетчика)
1	1	Установка OC1A/OC1B при совпадении (инкремент счетчика) Сброс OC1A/OC1B при совпадении (декремент счетчика)
Примечание – Особое место имеет случай, когда OCR1A/OCR1B равняется TOP и установлено COM1A1/COM1B1.		

Разряд 3: FOC1A – принудительное сравнение по выходу для канала A
Разряд 2: FOC1B – принудительное сравнение по выходу для канала B

Биты FOC1A/FOC1B активны только тогда, когда биты WGM1(3–0) определяют режим без ШИМ. В то же время для обеспечения совместимости с будущими устройствами, эти биты должны быть установлены в ноль, если TCCR1A записывается при работе в режиме ШИМ. При записи логической единицы в разряд FOC1A/FOC1B, в блок генерации сигнала произвольной формы принудительно устанавливается совпадение. Выход OC1A/OC1B изменяется согласно своим установкам разряда COM1x(1–0). Следует обратить также внимание на то, что биты FOC1A/FOC1B реализованы как стробы. С учетом этого именно значение, присутствующее в разрядах COM1x(1–0), определяет влияние принудительного совпадения. Стробирующий сигнал FOC1A/FOC1B не будет создавать какого-либо прерывания и не будет очищать таймер в режиме сброса таймера при совпадении при использовании OCR1A как TOP. Биты FOC1A/FOC1B всегда считываются как ноль.

Разряды 1, 0: WGM1(1–0) –режим генерации сигнала произвольной формы

Вместе с битами WGM1(3–2), расположенными в регистре TCCR1B, данные разряды могут управлять последовательностью счета счетчика. В отношении источника для верхнего значения счетчика (TOP), а также относительно того, какая форма сигнала должна использоваться, смотри таблицу 3.46. Режимы работы, поддерживаемые блоком таймера/счетчика, следующие: нормальный режим (счетчик), режим сброса таймера по совпадению при сравнении (CTC) и три режима ШИМ.

Таблица 3.46 – Описание битов для режимов генератора сигнала произвольной формы [1)]

Режим	WGM3	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Режим операции T/C	TOP	Обновление OCR1x в:	TOV1 флаг установлен в:
1	2	3	4	5	6	7	8	9
0	0	0	0	0	Нормальный	0xFFFF	Непосредств.	MAX
1	0	0	0	1	ШИМ, фазовая коррекция, 8 бит	0x00FF	TOP	BOTTOM
2	0	0	1	0	ШИМ, фазовая коррекция, 9 бит	0x01FF	TOP	BOTTOM
3	0	0	1	1	ШИМ, фазовая коррекция, 10 бит	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Непосредств.	MAX

Окончание таблицы 3.46

1	2	3	4	5	6	7	8	9
5	0	1	0	1	Быстрый ШИМ, 8 бит	0x00FF	TOP	TOP
6	0	1	1	0	Быстрый ШИМ, 9 бит	0x01FF	TOP	TOP
7	0	1	1	1	Быстрый ШИМ, 10 бит	0x03FF	TOP	TOP
8	1	0	0	0	ШИМ, фазовая и частотная коррекция	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	ШИМ, фазовая и частотная коррекция	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	ШИМ, фазовая коррекция	ICR1	TOP	BOTTOM
11	1	0	1	1	ШИМ, фазовая коррекция	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Непосредств.	MAX
13	1	1	0	1	Резервирован	-	-	-
14	1	1	1	0	Быстрый ШИМ	ICR1	TOP	TOP
15	1	1	1	1	Быстрый ШИМ	OCR1A	TOP	TOP

[1)] Наименования бит CTC1 и PWM1(1–0) являются устаревшими. Необходимо использовать определения WGM1(2–0), в то же время функциональное назначение и расположение этих битов совместимо с предыдущими версиями таймера.

Управление таймером/счетчиком 1

Регистр B – TCCR1B

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

Бит 7: ICNC1 – подавитель шума при захвате сигналов на входе

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

Бит 6: ICES1 – выбор фронта захвата события на входе

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

Когда ICR1 используется как значение TOP (см. описание разрядов WGM1(3–0), расположенных в регистре TCCR1A и TCCR1B), то ICP1 отсоединяется и затем отключается функция захвата события на входе.

Бит 5 – зарезервированный бит

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

Биты 4, 3: WGM1(3–2) – режим генерации сигнала произвольной формы

Смотри описание регистра TCCR1A.

Биты 2–0: CS1(2–0) – выбор тактового сигнала

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

Т а б л и ц а 3.47 – Описание бита выбора тактовой частоты

CS12	CS11	CS10	Описание
0	0	0	Без источника тактовой частоты (таймер/счетчик остановлен)
0	0	1	$clk_{I/O}$/1 (без предделения)
0	1	0	$clk_{I/O}$/8 (из предделителя)
0	1	1	$clk_{I/O}$/64 (из предделителя)
1	0	0	$clk_{I/O}$/256 (из предделителя)
1	0	1	$clk_{I/O}$/1024 (из предделителя)
1	1	0	Источник внешней синхронизации на выводе T1. Тактирование по спадающему фронту
1	1	1	Источник внешней синхронизации на выводе T1. Тактирование по нарастающему фронту

Если режимы на внешних выводах используются для таймера/счетчика 1, то переходы на выводе T1 будут тактировать счетчик даже в том случае, если вывод имеет конфигурацию в качестве выхода. Эта особенность позволяет обеспечить программное управление функцией счета.

Таймер/счетчик 1 – TCNT1H и TCNT1L

Бит	7	6	5	4	3	2	1	0	
	TCNT1(15–8)								TCNT1H
	TCNT1(7–0)								TCNT1L
Чт./Зап.	З	Ч/З	Ч	Ч/З	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Два регистра таймера/счетчика (TCNT1H и TCNT1L, объединенного TCNT1) обеспечены прямым доступом как для операций считывания, так и записи для блока 16-разрядного таймера/счетчика. Чтобы оба, старший и младший байты, считывались и записывались одновременно при обращении ЦПУ к этим регистрам, доступ осуществляется с использованием 8-разрядного временного регистра старшего байта (TEMP). Этот временный регистр одновременно используется всеми остальными 16-разрядными регистрами. Если в счетчик вносятся изменения (TCNT1), когда он работает, то это вызывает риск пропуска совпадения при сравнении между TCNT1 и одним из регистров OCR1x.

Запись в регистр TCNT1 блокирует совпадение при сравнении на следующий цикл тактового сигнала таймера для всех блоков сравнения.

Регистр сравнения по выходу – OCR1AH и OCR1AL

Бит	7	6	5	4	3	2	1	0	
	OCR1A(15–8)								OCR1AH
	OCR1A (7–0)								OCR1AL
Чт./Зап.	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 по выходу – OCR1BH и OCR1BL

Бит	7	6	5	4	3	2	1	0	
	OCR1B(15–8)								OCR1BH
	OCR1B(7–0)								OCR1BL
Чт./Зап.	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	

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

Регистры сравнения по выходу имеют размерность 16 бит. Чтобы оба, старший и младший байты, считывались и записывались одновременно в случае, если ЦПУ выполняет запись в эти регистры, доступ осуществляется с использованием 8-разрядного временного регистра старшего байта (TEMP). Этот временный регистр одновременно используется всеми 16-разрядными регистрами.

Бит	7	6	5	4	3	2	1	0	
	ICR1(15–8)								ICR1H
	ICR1(7–0)								ICR1L
Чт./Зап.	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 входа обновляется со значением счетчика каждый раз, когда происходит событие на выводе ICP1 (или на выходе аналогового компаратора для T/C1). Захват входа может быть использован для определения значения TOP.

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

Регистр маски прерываний T/C – TIMSK

Бит	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Чт./Зап.	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 – Регистр маски прерываний T/C – TIMSK содержит биты управления прерываниями для различных T/C, но только биты таймера 1 описаны в этом разделе. Остальные биты описаны в соответствующих разделах таймеров.

Бит 5: TICIE1 – разрешение прерывания по захвату данных

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

Бит 4: OCIE1A – разрешение прерывания по сравнению/совпадению A T/C1

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

Бит 3: OCIE1B – разрешение прерывания по сравнению/совпадению B T/C1

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

Бит 2: TOIE1 – разрешение прерывания по переполнению T/C1

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

Регистр флагов прерываний T/C – TIFR

Бит	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Чт./Зап.	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 – Регистр флагов прерываний T/C – TIFR содержит биты управления прерываниями для различных T/C, но только биты таймера 1 описаны в этом разделе. Остальные биты описаны в соответствующих разделах таймеров.

Бит 5: ICF1 – флаг захвата данных

Этот флаг устанавливается, когда происходит событие захвата на ICP выводе. Когда регистр захвата входа ICR1 устанавливается WGM1(3–0) для использования его значения как TOP, флаг ICF1 устанавливается, когда счетчик достигает значения TOP.

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

Бит 4: OCF1A – флаг сравнения/совпадения A T/C1

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

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

Бит 3: OCF1B – флаг сравнения/совпадения B T/C1

Флаг устанавливается в следующем цикле тактового импульса таймера, после того как произойдет совпадение регистра OCR1B.

Заметим, что строб вынужденного совпадения не будет устанавливать OCF1B флаг.

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

Бит 2: TOV1 – флаг переполнения T/C1

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

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

3.13 8-битный таймер/счетчик 2 с ШИМ и асинхронными операциями

Таймер/счетчик 2 (T/C2) является одноканальным модулем 8-разрядного T/C общего назначения.

Отличительные особенности:

- одноканальный счетчик;
- сброс таймера при совпадении сравнения (автоматическая перезагрузка);
- ШИМ с фазовой коррекцией, свободной от сбоев;
- генератор частоты;
- счетчик внешних событий;
- 10-разрядный предварительный делитель частоты;
- источники прерывания при совпадении, при сравнении и при переполнении (TOV2 и OCF2);
- возможность тактирования от внешнего часового кварца частотой 32 768 Гц.

3.13.1 Обзор

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

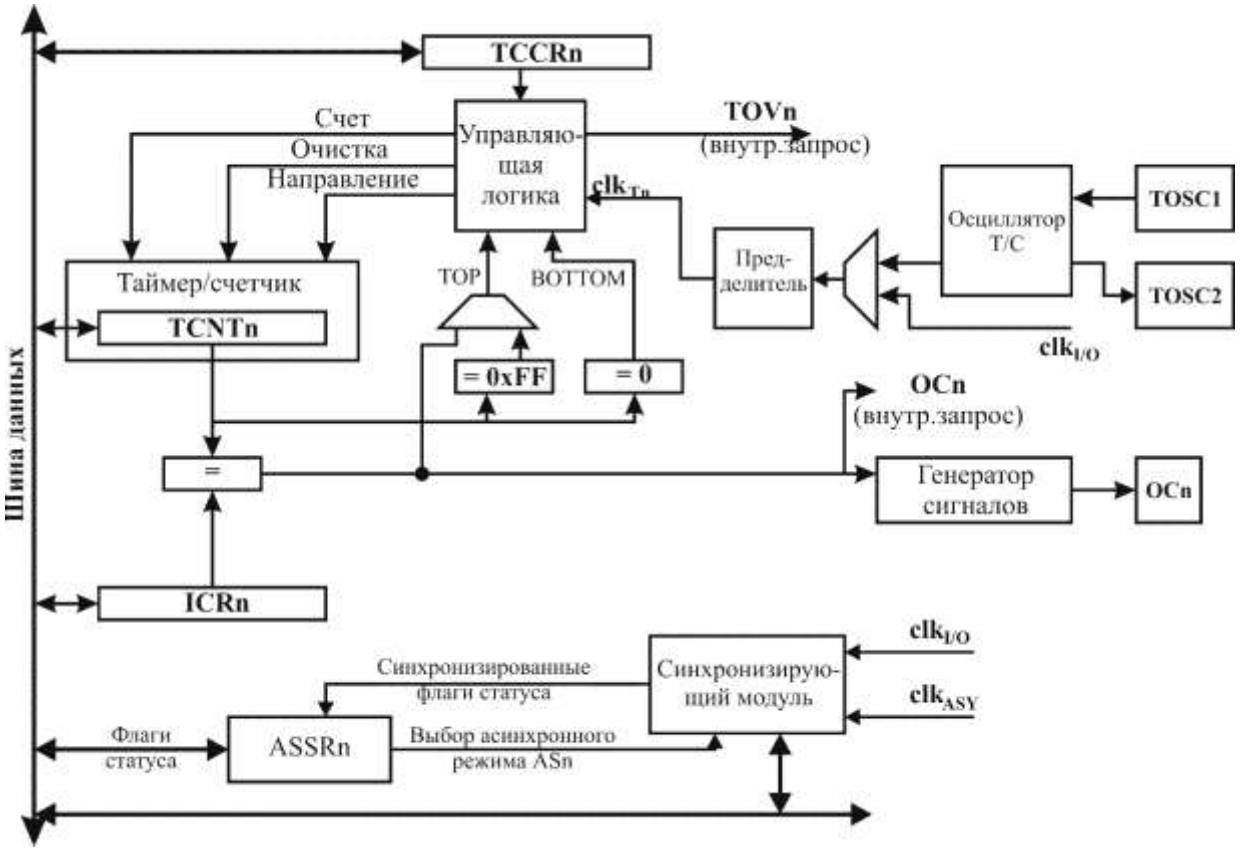


Рисунок 3.51 – Блок-схема 8-разрядного таймера/счетчика

3.13.2 Регистры

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

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

3.13.3 Определения

Многие справочные сведения относительно регистров и разрядов записаны в подразделе 3.13 в общем виде. Буква «n» в обозначении указывает номер таймера/счетчика, в данном случае 2. Однако, когда использование регистра или разряда определено программой, необходимо использовать точную форму, например, TCNT2 для оценки значения счетчика таймера/счетчика 2 и т. д.

Определения из таблицы 3.48 также интенсивно используются в подразделе 3.13.

Таблица 3.48 – Определения

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

3.13.4 Источники тактового сигнала таймера/счетчика

Таймер/счетчик может синхронизироваться как внутренним, так и внешним источником синхронизации. Источник синхронизации выбирается с помощью логики выбора тактового сигнала, который управляется с помощью разрядов выбора сигнала синхронизации CS2(2–0), расположенных в регистре управления таймера/счетчика (TCCR2).

3.13.5 Модуль счетчика

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

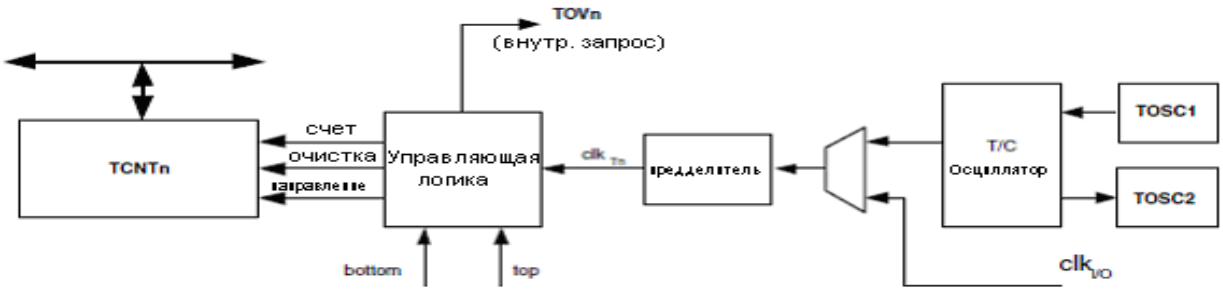


Рисунок 3.52 – Блок-схема модуля счетчика

Описание сигналов (внутренние сигналы):

- счет – инкремент или декремент TCNT2 на 1;
- направление – выбор между инкрементом или декрементом;
- очистка – очистка TCNT2 (все разряды устанавливаются в ноль);
- clk_{Tn} – тактовый генератор таймера/счетчика, далее именуемый как clk_{T2};
- top – сигнализирует, что TCNT2 достигло верхнего значения;
- bottom – сигнализирует, что TCNT2 достигло нижнего значения (нуля).

В зависимости от используемого режима работы счетчик сбрасывается, инкрементируется или декрементируется в каждом цикле тактового сигнала таймера (clk_{T2}). clk_{T2} сможет генерироваться от внешнего или внутреннего источника тактового сигнала, выборка при этом производится с помощью разрядов CS2(2–0). Если источник тактовых сигналов не выбирается, т. е. CS2(2–0) = 0, то таймер останавливается. В то же время, величина TCNT2 может быть выбрана с помощью ЦПУ независимо от того, присутствует clk_{T2} или нет. Операция записи в ЦПУ отменяет (обладает большим приоритетом) все операции очистки счетчика или операции счета.

Последовательность счета определяется путем установки разрядов WGM21 и WGM20, расположенных в регистре управления таймером/счетчиком TCCR2. Существует непосредственная связь между тем, как ведет себя счетчик (т. е. как он считает) и какие сигналы формируются на выходе OC2.

Флаг переполнения таймера/счетчика TOV2 устанавливается в соответствии с режимом работы, который выбирается с помощью разрядов WGM2(1–0). Для генерации прерывания ЦПУ можно использовать TOV2.

3.13.6 Модуль сравнения по выходу

8-разрядный компаратор непрерывно сравнивает TCNT2 с регистром сравнения выхода (OCR2). Как только TCNT2 становится равным OCR2, компаратор сигнализирует о совпадении. Совпадение установит флаг OCF2 при следующем цикле тактового сигнала таймера. При активации (OCIE2 = 1 при установке флага общего прерывания в SREG) флаг OCF2 создает прерывание выхода сравнения. Флаг OCF2 очищается автоматически при выполнении процедуры обработки прерывания. В качестве альтернативы флаг OCF2 можно очищать программно путем записи логической единицы в адрес разряда флага. Генератор сигнала произвольной формы использует сигнал сравнения для генерации выходного сигнала согласно режиму работы, установленному разрядами WGM2(1–0), и режиму сравнения выхода, установленному разрядами COM2(1–0). Максимальное и нижнее значения сигнала используются генератором сигнала произвольной формы для решения специальных случаев с крайними значениями в некоторых режимах работы.

На рисунке 3.53 показана схема блока управления устройством сравнения выхода.

При работе в любом режиме с использованием ШИМ (широтно-импульсной модуляции), регистр OCR2 имеет двойную буферизацию. При работе в обычном режиме и режиме очистки таймера при совпадении (CTC) двойная буферизация отключается. Двойная буферизация синхронизирует обновление регистра сравнения либо вверх, либо вниз относительно последовательности счета. Синхронизация предотвращает возникновение импульсов избыточной длины, несимметричных импульсов ШИМ, обеспечивая таким образом бесперебойную работу выхода. Доступ к регистру может показаться достаточно сложным, но это не так. При активации двойной буферизации ЦПУ имеет доступ к регистру буфера OCR2, а если двойная буферизация отключена, то ЦПУ будет осуществлять прямую выборку OCR2.

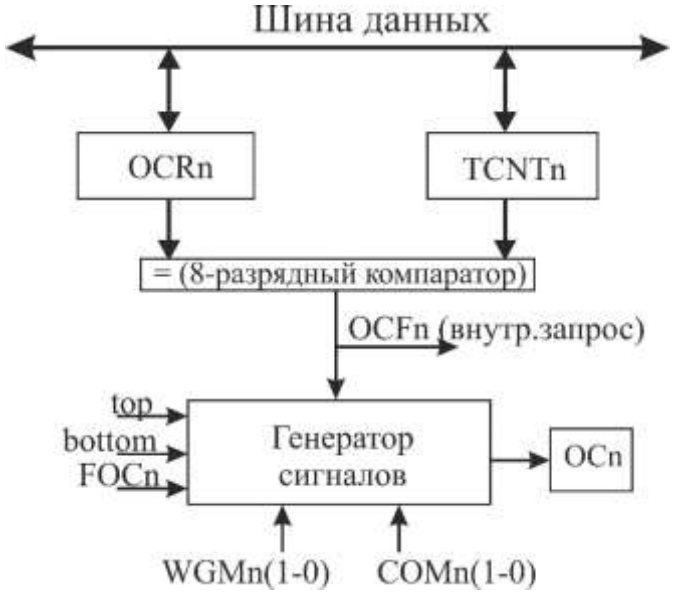


Рисунок 3.53 – Блок-схема модуля сравнения выхода

3.13.7 Принудительное совпадение

В режимах генерации без использования ШИМ принципа, выход сравнения для компаратора может быть установлен записью логической единицы в разряд FOC2. Принудительное сравнение/совпадение не будет устанавливать флаг OCF2 или перезагружать/сбрасывать таймер, однако вывод OC2 будет обновляться, как будто произошло реальное совпадение (установки разрядов COM2(1–0) определяют, что произойдет с выводом OC2, будет ли он установлен, сброшен или переключен).

3.13.8 Блокировка совпадения сравнения с помощью записи TCNT2

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

3.13.9 Использование блока сравнения по выходу

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

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

Следует убедиться в том, что разряды COM2(1–0) не имеют двойной буферизации вместе с величиной сравнения. Изменение разрядов COM2(1–0) сразу же даст эффект.

3.13.10 Блок сравнения совпадений на выходе

Разряды режима сравнения на выходе COM2(1–0) несут две функции. Генератор сигнала произвольной формы использует разряды COM2(1–0) для оценки состояния сравнения на выходе при следующем совпадении. Кроме этого, разряды COM2(1–0) управляют источником выходного сигнала для вывода OC2. На рисунке 3.54 представлена упрощенная блок-схема логики, используемой при установке разряда COM2(1–0). Показаны только части общих регистров управления портом входа/выхода DDR и PORT на которые влияют разряды COM2(1–0). При ссылках на состояние OC2 речь идет о внутреннем регистре OC2, а не о выводе OC2. Если происходит сброс системы, то регистр OC2 сбрасывается в «0».

Общая функция порта входа/выхода переопределяется с помощью выхода сравнения (OC2) из генератора сигнала произвольной формы, если устанавливается какой-либо из разрядов COM2(1–0). Однако, направление вывода OC2 (вход или выход) все еще контролируется регистром направления данных DDR для вывода порта. Разряд регистра направления данных для вывода OC2 (DDR_OC2) должен быть установлен в качестве выхода до того, как величина OC2 станет доступной на выводе. Функция переопределения порта не зависит от режима генерации сигнала.

Логическая схема сравнения выхода разрешает инициализацию состояния OC2 до того, как активируется выход. Следует обратить внимание на то, что некоторые установки разряда COM2(1–0) сохраняются для определенных режимов работы.

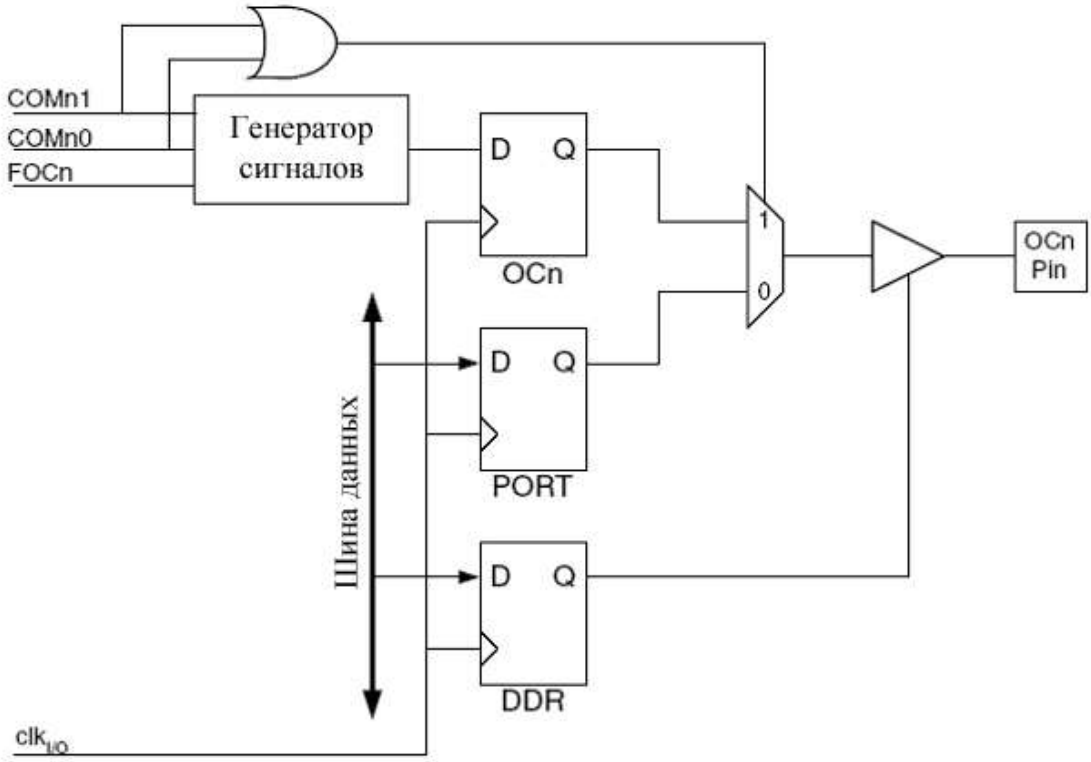


Рисунок 3.54 – Блок сравнения/совпадения на выходе

Логическая схема вывода «выход сравнения» позволяет инициализацию состояния OC2 до того, как активируется выход. Обратите внимание на то, что некоторые установки разряда COM2(1–0) сохраняются для определенных режимов работы.

3.13.11 Режим сравнения по выходу и генерация сигнала произвольной формы

Генератор сигнала произвольной формы использует разряды COM2(1–0) различным образом для обычного режима, режима условных переходов (CTC) и режима широтно-импульсной модуляции. Для всех типов режимов установка COM2(1–0) = 0 сообщает генератору, что в регистре OC2 не должно выполняться никаких действий по следующему сравнению/совпадению.

Изменение состояния разрядов в COM2(1–0) повлияет на первое совпадение после записи разрядов. Для режимов, отличающихся от ШИМ, может быть использовано действие, дающее немедленный эффект благодаря использованию разрядов принудительного совпадения FOC2.

3.13.12 Режимы работы

Режим работы, т. е. поведение таймера/счетчика и выводов сравнения, определяется комбинацией разрядов для режима генерации сигнала произвольной формы WGM2(1–0) и режима сравнения по выходу COM2(1–0). Разряды режима сравнения выхода не влияют на последовательность счета, в то время как разряды режима генерации сигнала влияют. Разряды COM2(1–0) управляют тем, будет инвертирован или нет сгенерированный выход ШИМ (инвертированный или неинвертированный ШИМ). Для режимов, не использующих

ШИМ, разряды COM2(1–0) контролируют: должен ли выход быть установлен, очищен или переключен при совпадении.

Нормальный режим

Самым простым режимом работы является нормальный режим WGM2(1–0) = 0. В этом режиме направление счета всегда возрастающее и сброс счетчика не выполняется. Счетчик просто переполняется, когда он проходит свое максимальное 8-разрядное значение (TOP = 0xFF) и затем перезапускается (0x00). В нормальном режиме работы флаг переполнения таймера/счетчика (TOV2) будет установлен во время того же цикла синхронизации таймера, когда TCNT2 становится равным нулю. Флаг TOV2 в этом случае ведет себя как 9-й разряд, за исключением того, что он только устанавливается, а не сбрасывается. Однако, при объединении с прерыванием переполнения таймера, которое автоматически сбрасывает флаг TOV2, разрешение таймера можно повысить программным способом. В нормальном режиме нет необходимости рассматривать специальные случаи, при этом новое значение таймера может быть записано в любое время.

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

Сброс таймера в режиме сравнения/совпадения (CTC)

В режиме сброса таймера при сравнении или режиме CTC, WGM2(1–0) = 2, регистр OCR2 используется для управления разрешением счетчика. В режиме CTC счетчик сбрасывается до нуля, когда значение счетчика (TCNT2) совпадает с OCR2. OCR2 определяет верхнее значение счетчика, а следовательно, его разрешение. Данный режим обеспечивает большую степень управления выходной частотой сравнения/совпадения. Это упрощает также работу для счета внешних событий.

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

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

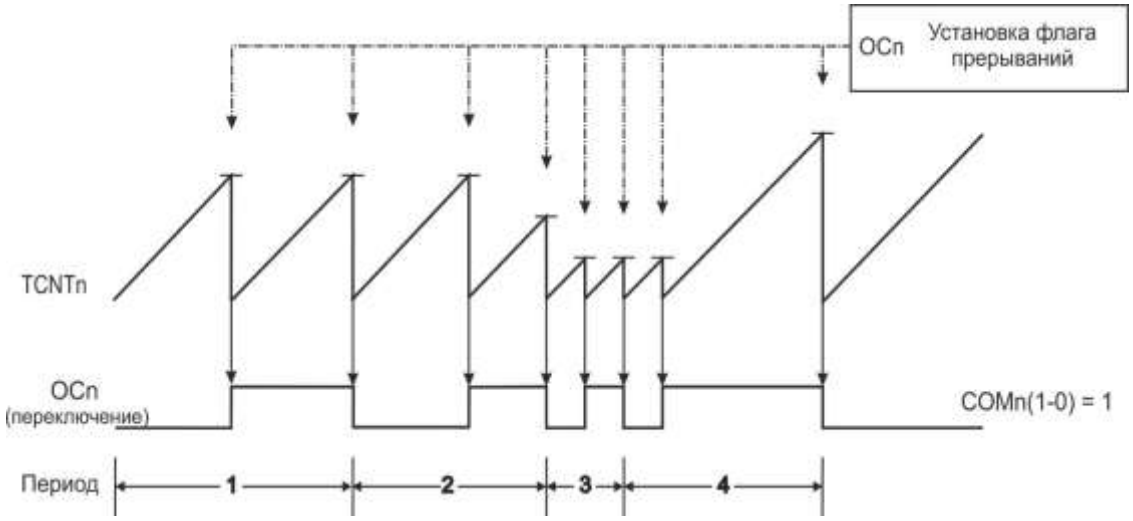


Рисунок 3.55 – Режим CTC, временная диаграмма

Для генерации выхода сигнала произвольной формы в режиме CTC выход OC2 может быть установлен на переключение своего логического уровня при каждом совпадении путем установки разрядов режима сравнения выхода в режим переключения COM2(1–0) = 1. Значение OC2 не будет доступно на выводе порта до тех пор, пока направление данных не установится, как «выход». Генерируемый сигнал будет иметь максимальную частоту $f_{OC2} = f_{clk\_I/O}/2$ при установке OCR2 в ноль (0x00). Частота сигнала определяется выражением

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

Переменная «N» представляет собой коэффициент предварительного деления частоты (1, 8, 32, 64, 128, 256 или 1024).

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

Быстрый ШИМ режим

Быстрая широтно-импульсная модуляция или быстрый ШИМ режим WGM2(1–0) = 3 предоставляет возможность генерации сигнала ШИМ с высокой частотой. Быстродействующая ШИМ отличается от другой ШИМ благодаря ее работе по одному фронту. Счетчик выполняет счет из положения BOTTOM до MAX, затем вновь начинает работу с BOTTOM. В неинвертируемом режиме сравнения по выходу, OC2 очищается при совпадении сравнения между TCNT2 и OCR2 и устанавливается как BOTTOM. В инвертируемом режиме сравнения выхода, выход устанавливается при совпадении и сбрасывается при BOTTOM. Благодаря работе с использованием одного фронта, рабочая частота для быстрого ШИМ режима может в два раза превышать частоту ШИМ режима с фазовой коррекцией частоты, которая использует принцип работы с применением двух фронтов. Благодаря столь высокой частоте, быстрый ШИМ режим очень хорошо подходит для регулировки мощности, выпрямления и применения в ЦАП. Высокая частота обуславливает малые физические размеры внешних компонентов (катушек, конденсаторов), благодаря чему снижается общая стоимость системы.

В быстром ШИМ режиме счетчик возрастает до тех пор, пока его значение не достигнет значения MAX. Затем счетчик сбрасывается при следующем цикле синхронизации таймера. Временная диаграмма для быстрого ШИМ режима показана на рисунке 3.56. Значение величины TCNT2 на временной диаграмме показано в виде гистограммы для ил-

люстрации работы в режиме с использованием одного фронта. На рисунке показаны как инвертированные, так и неинвертированные ШИМ выходы. Небольшие горизонтальные метки на фронтах TCNT2 представляют совпадения между OCR2 и TCNT2.

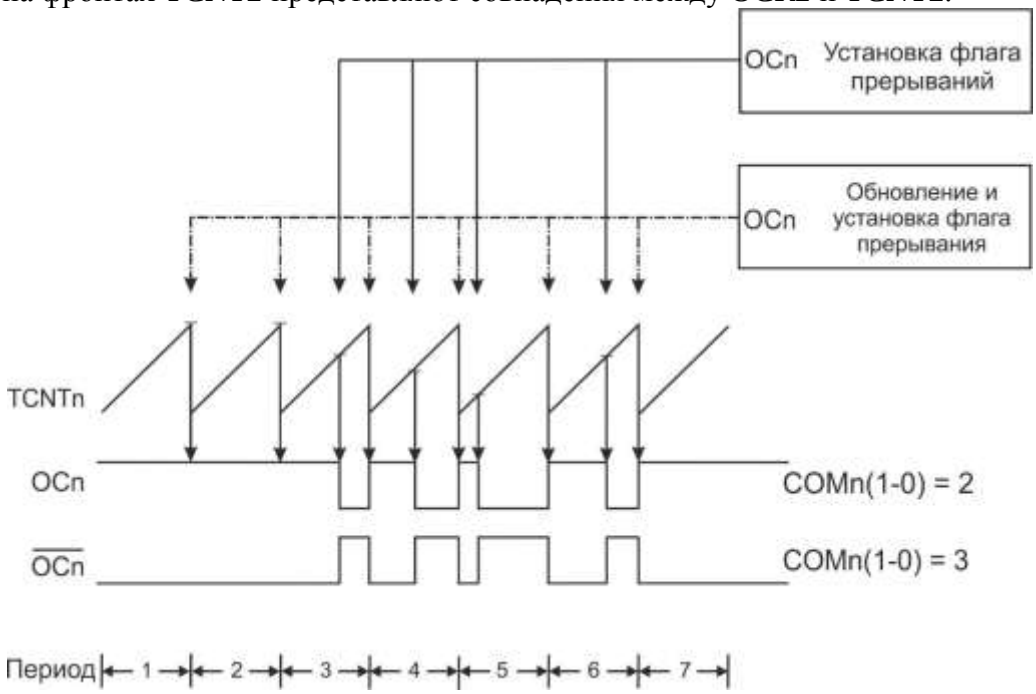


Рисунок 3.56 – Временная диаграмма для быстрого ШИМ режима

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

В быстром ШИМ режиме модуль сравнения разрешает генерацию ШИМ сигналов на выводе OC2. Установка разрядов COM2(1–0) в «10» дает неинвертированный ШИМ, а инвертированный ШИМ можно сформировать, установив COM2(1–0) в «11». Фактическое значение OC2 будет видно на выводе порта, если направление данных для вывода установлено как «выход». Форма сигнала ШИМ генерируется с помощью установки (или очистки) регистра OC2 при совпадении между OCR2 и TCNT2, и очистки (или установки) регистра OC2 во время тактового периода, когда счетчик очищается (изменяется с MAX на BOTTOM).

Частоту выходного ШИМ можно вычислить, используя уравнение

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

Переменная «N» представляет собой коэффициент предварительного деления (1, 8, 32, 64, 128, 256 или 1024).

Крайние значения для регистра OCR2 представляют особые случаи при генерации сигнала ШИМ на выходе в быстром ШИМ режиме. Если OCR2 устанавливается равным BOTTOM, то выходной сигнал будет представлять собой острый импульс для каждого тактового цикла таймера MAX+1. Если OCR2 устанавливается равным MAX, то это приводит к постоянному высокому или низкому логическому состоянию на выходе (в зависимости от полярности на выходе, которая установлена разрядами COM2(1–0)).

Частота формы сигнала на выходе (при 50 % рабочем режиме) в быстром ШИМ режиме может быть получена путем установки OC2 на переключение своего логического уровня при каждом совпадении COM2(1–0) = 1. Форма сигнала, генерируемая при этом, будет иметь максимальную частоту $f_{OC2} = f_{clk\_I/O}/2$ при установке OCR2 в ноль. Эта осо-

бенность аналогична переключению OC2 в режиме CTC, за исключением того, что двойная буферизация в модуле сравнения выхода в быстром ШИМ режиме включена.

Режим фазовой коррекции ШИМ

Режим фазовой коррекции ШИМ WGM2(1–0) = 1 обеспечивает возможность генерации сигнала ШИМ с фазовой коррекцией и высокой разрешающей способностью. Режим фазовой коррекции ШИМ основан на работе с использованием двух фронтов. Счетчик ведет повторяемый подсчет от BOTTOM к MAX, а затем от MAX к BOTTOM. В неинвертированном режиме сравнения OC2 очищается при совпадении между TCNT2 и OCR2 при инкременте и устанавливается при совпадении при декременте. В инвертированном режиме сравнения по выходу операция инвертируется. Принцип работы по двум фронтам имеет пониженную максимальную рабочую частоту по сравнению с одним фронтом. В то же время благодаря симметричной особенности ШИМ режимов по двум фронтам, подобные режимы предпочтительны для использования при управлении двигателями.

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

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

В режиме ШИМ с фазовой коррекцией модуль сравнения обеспечивает генерацию импульсов ШИМ на выводе OC2. Установка разрядов COM2(1-0) в значение «10» вызовет неинвертированный ШИМ режим. Инвертированный выход ШИМ может генерироваться путем установки COM2(1–0) в «11».

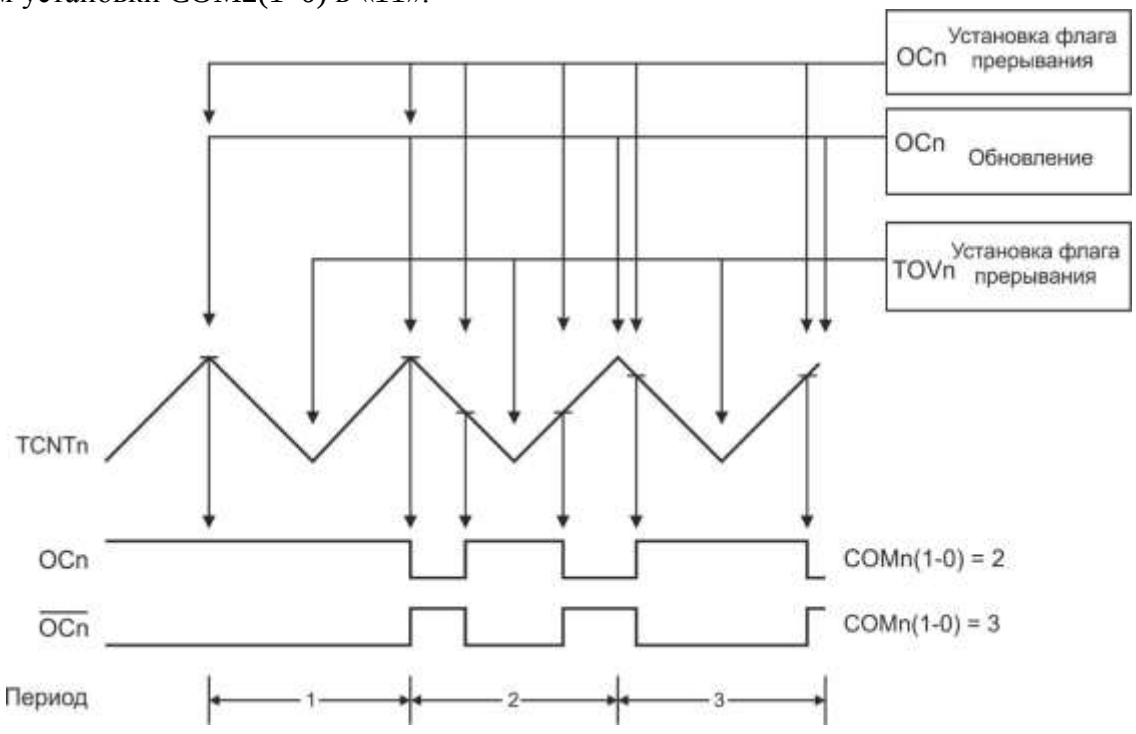


Рисунок 3.57 – Временная диаграмма, режим ШИМ с фазовой коррекцией

Фактическое значение OC2 будет доступно на выводе порта только в случае, когда направление данных для вывода порта будет установлено как «выход». ШИМ сигнал генерируется путем сброса (или установки) регистра OC2 при совпадении между OCR2 и TCNT2, когда значение счетчика возрастает, и при установке (или сбросе) регистра OC2 при совпадении между OCR2 и TCNT2, когда значение счетчика уменьшается. Частота ШИМ для выхода может быть вычислена с помощью формулы

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

Переменная «N» представляет собой коэффициент предварительного деления (1, 8, 32, 64, 128, 256 или 1024).

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

Временные диаграммы таймера/счетчика

Таймер/счетчик имеет синхронный принцип работы, и тактовый импульс таймера (clk_{T2}) с учетом этого показан на следующих рисунках как сигнал синхронизации. Рисунки содержат информацию о том, когда устанавливаются флаги прерывания.

На рисунке 3.58 приведены временные параметры по основному режиму работы таймера/счетчика. На рисунке приведена последовательность счета, близкая к значению MAX во всех режимах, отличающихся от режима ШИМ с фазовой коррекцией.

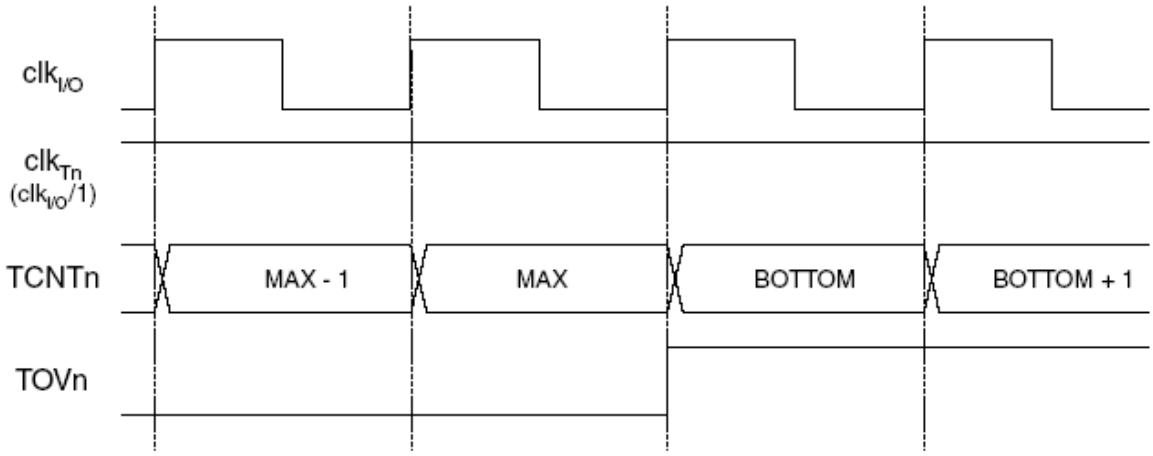


Рисунок 3.58 – Временная диаграмма таймера/счетчика без предварительного деления частоты

На рисунке 3.59 показаны те же временные параметры, но с предварительным делением частоты.

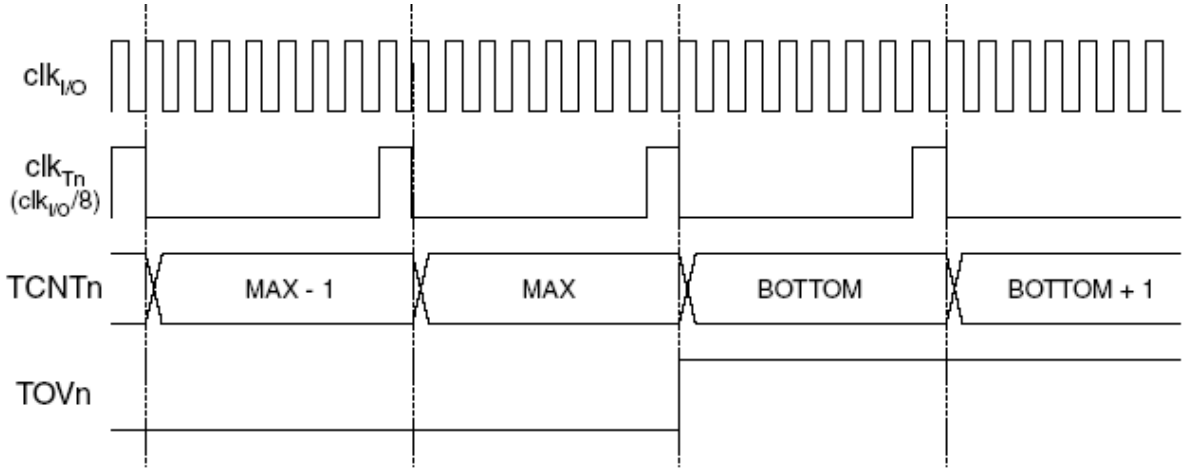


Рисунок 3.59 – Временная диаграмма таймера/счетчика с предварительным делением частоты ($f_{clk\_I/O}/8$)

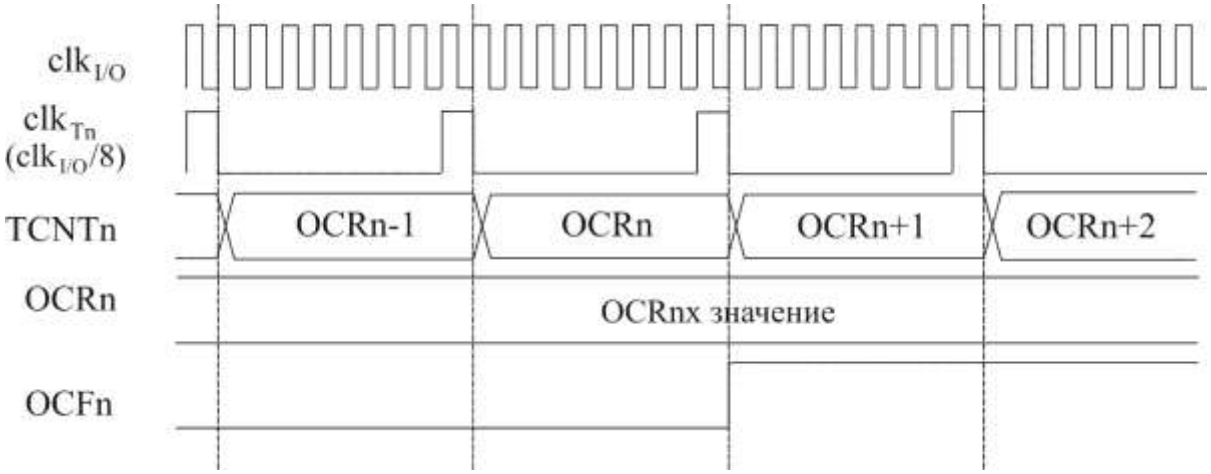


Рисунок 3.60 – Временная диаграмма таймера/счетчика с установкой OCF2 и предварительным делением частоты ($f_{clk\_I/O}/8$)

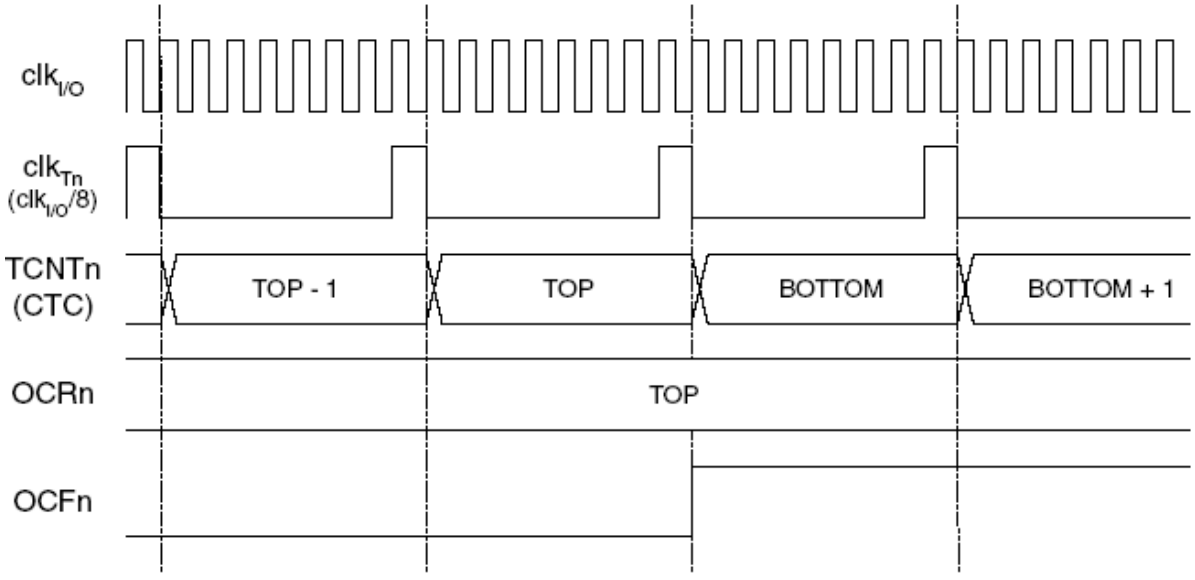


Рисунок 3.61 – Временная диаграмма таймера/счетчика, сброс таймера в режиме сравнения/совпадения с предварительным делением частоты ($f_{clk\_I/O}/8$)

3.13.13 Описание регистров 8-разрядного таймера/счетчика

Регистр управления таймера/счетчика – TCCR2

Бит	7	6	5	4	3	2	1	0	
	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	TCCR2
Чт./Зап.	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 – принудительное совпадение

Разряд FOC2 активируется только тогда, когда разряд WGM20 задает режим без использования ШИМ. В то же время для совместимости с будущими устройствами этот разряд должен быть установлен в ноль, когда записывается TCCR2 при работе в режиме ШИМ. При записи логической единицы в разряд FOC2 на модуль генерации сигнала подается команда на немедленное совпадение. Выход OC2 изменяется в соответствии с установкой его разрядов COM2(1–0). Следует обратить внимание на то, что разряд FOC2 реализуется в виде строба. С учетом этого, значение, присутствующее в разрядах COM2(1–0), определяет результат принудительного совпадения.

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

Разряд FOC2 всегда считывается как ноль.

Разряды 6, 3 – режим генерации сигнала произвольной формы WGM2(1–0)

Эти разряды управляют последовательностью счета для счетчика, источником максимального значения счетчика (TOP), а также используемым типом генерации сигнала. Модуль таймера/счетчика поддерживает следующие режимы работы: нормальный, режим сброса счетчика при совпадении (CTC), а также два типа режимов ШИМ (смотри таблицу 3.49).

Т а б л и ц а 3.49 – Описание разряда режима работы генерации формы сигнала [1)]

Режим	WGM21 (CTC2)	WGM20 (PWM2)	Режим работы таймера/счетчика	TOP	Обновление OCR2	Установка флага TOV2
0	0	0	Нормальный	0xFF	Немедленное	MAX
1	0	1	PWM, коррекция фазы	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR2	Немедленное	MAX
3	1	1	Быстрый ШИМ	0xFF	TOP	MAX

[1)] Наименования разрядов CTC2 и PWM2 в настоящее время считаются устаревшими. Следует использзовать определения WGM2(1–0), в то же время функциональность и расположение этих разрядов совместимы с прежними версиями таймера.

Разряды 5, 4: COM2(1–0) – режим сравнения/совпадения выхода

Эти разряды управляют поведением вывода «выход сравнения» (OC2). Если устанавливается один или оба разряда COM2(1–0), то выход переопределяет нормальную функцию порта входа/выхода. В то же время необходимо обратить внимание на то, что разряд регистра направления (DDR), соответствующий выводу OC2, должен быть установлен так, чтобы активировать выходной драйвер.

Когда к выводу подсоединен OC2, то функция разрядов COM2(1–0) зависит от установки разряда WGM2(1–0). В таблице 3.50 показана функциональность разряда COM2(1–0), когда разряды WGM2(1–0) устанавливаются в нормальный режим работы или в режим CTC (не режим ШИМ).

Т а б л и ц а 3.50 – Режим сравнения выхода, режим без ШИМ

COM21	COM20	Описание
0	0	Нормальная работа порта, OC2 отсоединен
0	1	Переключение OC2 при совпадении
1	0	Сброс OC2 при совпадении
1	1	Установка OC2 при совпадении

В таблице 3.51 приведена функциональность разряда COM2(1–0), когда разряды WGM2(1–0) устанавливаются в быстрый режим ШИМ.

Т а б л и ц а 3.51 – Режим сравнения выхода, быстрый режим ШИМ [1)]

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

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

В таблице 3.52 показывается функциональность разряда COM2(1–0), когда разряды WGM2(1–0) установлены в режим фазовой коррекции ШИМ.

Т а б л и ц а 3.52 – Режим сравнения по выходу, режим фазовой коррекции ШИМ

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

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

Разряды 2–0: CS2(2–0) – выбор синхронизации

Три разряда выбора синхронизации выбирают источник синхронизации для работы таймера/счетчика.

Т а б л и ц а 3.53 – Описание разряда выбора синхронизации

CS22	CS21	CS20	Описание
0	0	0	Нет источника синхронизации (таймер/счетчик остановлен)
0	0	1	clk_{T2S}/ (Без деления частоты)
0	1	0	clk_{T2S} /8 (От устройства предварительного деления частоты)
0	1	1	clk_{T2S} /32 (От устройства предварительного деления частоты)
1	0	0	clk_{T2S} /64 (От устройства предварительного деления частоты)
1	0	1	clk_{T2S} /128 (От устройства предварительного деления частоты)
1	1	0	clk_{T2S} /256 (От устройства предварительного деления частоты)
1	1	1	clk_{T2S} /1024 (От устройства предварительного деления частоты)

Регистр таймера/счетчика – TCNT2

Бит	7	6	5	4	3	2	1	0	
	TCNT2(7–0)								TCNT2
Чт./Зап.	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 обеспечивает прямой доступ как для операции считывания, так и записи в 8-разрядный счетный регистр блока таймера/счетчика. Запись в блоки регистра TCNT2 блокирует совпадение при сравнении для следующего сигнала синхронизации таймера. Изменение значения счетчика (TCNT2) во время работы счетчика создает риск отсутствия совпадения при сравнении между TCNT2 и регистром OCR2.

Регистр сравнения выхода – OCR2

Бит	7	6	5	4	3	2	1	0	
	OCR2(7–0)								OCR2
Чт./Зап.	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 выхода содержит 8-разрядную величину, которая непрерывно сравнивается со значением счетчика (TCNT2). Совпадение может использоваться для генерации прерывания при сравнении по выходу или для генерации сигнала на выводе OC2.

Асинхронный режим

Отличительной особенностью таймера/счетчика T2 является его возможность работать в асинхронном режиме. В этом режиме на вход предделителя поступает сигнал с вывода TOSC1, что позволяет использовать таймер/счетчик в качестве часов реального времени. Источником сигнала может быть как кварцевый резонатор частотой 32 768 Гц, подключаемый к выводам TOSC1 и TOSC2 микроконтроллера, так и внешняя схема. Несмотря на то, что тактовый генератор таймера/счетчика настроен на частоту 32 768 Гц, частота сигнала от внешней схемы может лежать в пределах от 0 до 256 кГц. При этом частота внешнего сигнала должна быть в четыре раза меньше частоты тактового сигнала микроконтроллера.

Для определения момента действительного изменения регистров TCNT2, OCR2 и TCCR2, а также для переключения таймера/счетчика в асинхронный режим, предназначен регистр ASSR, расположенный по адресу $22 ($42).

Бит	7	6	5	4	3	2	1	0	
					AS2	TCN2UB	OCR2UB	TCR2UB	ASSR
Чт./Зап.	Ч	Ч	Ч	Ч	Ч/З	Ч	Ч	Ч	
Начальное значение	0	0	0	0	0	0	0	0	

Таблица 3.54 – Регистр состояния асинхронного режима ASSR

Бит	Описание
AS2	Асинхронный таймер/счетчик 2 Когда AS2 очищается, T/C2 тактируется от $clk_{I/O}$. Когда AS2 устанавливается, T/C2 тактируется от кварцевого генератора, подключенного к выводу TOSC1. Когда значение AS2 изменяется, содержимое TCNT2, OCR2, TCCR2 может измениться
TCN2UB	Флаг занятости при обновлении TCNT2 Когда T/C2 работает в асинхронном режиме и записывается TCNT2, этот бит устанавливается. Когда TCNT2 обновляется содержимым временного регистра, этот бит аппаратно очищается. Логический ноль в этом бите отображает, что TCNT2 готов для обновления новым значением
OCR2UB	Флаг занятости при обновлении OCR2 Когда T/C2 работает в асинхронном режиме и записывается OCR2, этот бит устанавливается. Когда OCR2 обновляется содержимым временного регистра, этот бит аппаратно очищается. Логический ноль в этом бите отображает, что OCR2 готов для обновления новым значением
TCR2UB	Флаг занятости при обновлении TCCR2 Когда T/C2 работает в асинхронном режиме и записывается TCCR2, этот бит устанавливается. Когда TCCR2 обновляется содержимым временного регистра, этот бит аппаратно очищается. Логический ноль в этом бите отображает, что TCCR2 готов для обновления новым значением

Непосредственная запись в регистры TCNT2, OCR2 и TCCR2 в асинхронном режиме синхронизируется с тактовым сигналом таймера/счетчика. При записи числа в любой из указанных регистров оно сохраняется в специальном временном регистре: своем для каждого регистра таймера/счетчика. А пересылка содержимого временного регистра в рабочий регистр таймера/счетчика осуществляется по третьему после записи положительному фронту сигнала на выводе TOSC1. Соответственно запись нового значения можно производить только после пересылки содержимого временного регистра в регистр таймера/счетчика.

Если запись производится в любой из трех регистров T/C2, когда установлен флаг занятости при обновлении содержимого регистра, обновляемое значение может быть нарушено и может произойти непреднамеренное прерывание.

Механизм чтения регистров T/C2 различен. Когда читается TCNT2, читается действительное значение таймера. Когда читается OCR2 или TCCR2, читается значение временного регистра.

Необходимо отметить, что при переключении между синхронным и асинхронным режимами содержимое регистров таймера/счетчика может быть изменено. Чтобы этого избежать, рекомендуется придерживаться следующей последовательности действий:

- запретить прерывания от таймера/счетчика T2 (разряды TOIE2 и OCIE2 регистра TIMSK);
- выбрать соответствующий источник тактирования записью значения в AS2;
- записать новые значения в регистры TCNT2, OCR2 и TCCR2;
- в случае переключения в асинхронный режим – ждать, пока флаги TCN2UB, OCR2UB и TCR2UB не будут сброшены;
- очистить флаги прерываний T/C2;
- разрешить прерывания (если требуется).

Осциллятор оптимизирован для использования с часовым кварцем (32768 Гц). Подача внешнего тактового сигнала на TOSC1 может привести к некорректной операции T/C2. Частота ЦПУ должна быть по меньшей мере в четыре раза больше частоты осциллятора.

Когда записывается значение в один из регистров TCNT2, OCR2, TCCR2, значение передается во временный регистр и защелкивается после двух положительных фронтов на TOSC1. Пользователь не должен записывать новое значение до того, как содержимое временного буфера будет передано по назначению. Каждый из трех упомянутых регистров имеет индивидуальный временный регистр, это означает, что запись в TCNT2 не помешает записи в OCR2. Для детектирования, что передача в регистр назначения имела место, служит регистр ASSR.

После входа в режим микропотребления или расширенного ожидания после записи TCNT2, OCR2, TCCR2 пользователь должен ждать, пока содержимое записываемого регистра обновлено, если T/C2 используется для пробуждения устройства. Иначе микроконтроллер войдет в спящий режим до того, как обновления регистров вступят в силу. Это особенно важно, если прерывание OC2 используется для пробуждения устройства, поскольку функция выхода сравнения запрещена во время записи в OCR2 или TCNT2. Если цикл записи не закончен и микроконтроллер войдет в спящий режим до того, как OCR2UB вернется в ноль, устройство никогда не примет прерывание сравнения/совпадения, и микроконтроллер никогда не проснется.

Если T/C2 используется для пробуждения из режима микропотребления или расширенного режима ожидания, должны быть приняты меры предосторожности, если пользователь хочет выйти и снова зайти в один из этих режимов: логике прерывания необходим один такт TOSC1 для сброса. Если время между пробуждением и повторным входом в спящий режим меньше, чем один такт TOSC1, прерывание не произойдет и устройство никогда не проснется. Если пользователь сомневается, достаточное ли время прошло до повторного входа в спящий режим, следующий алгоритм должен быть выполнен, чтобы гарантировать, что прошел один такт TOSC1:

1 Записать значения в TCNT2, OCR2, TCCR2.

2 Дождаться сброса соответствующих флагов занятости при обновлении в ASSR.

3 Войти в режим микропотребления или режим расширенного ожидания.

Когда выбирается асинхронный режим, часовой осциллятор для T/C2 всегда запущен, кроме режимов хранения и ожидания. После сброса по включению питания или пробуждения из режима хранения или режима ожидания, пользователь должен быть готов к тому, что осциллятору может потребоваться время (вплоть до 1 секунды) на стабилизацию работы. Пользователю рекомендуется подождать 1 секунду перед использованием T/C2 после включения питания или пробуждения из режима хранения или режима ожидания. Содержимое всех регистров T/C2 должно считаться утерянным после пробуждения из режима хранения или режима ожидания из-за нестабильного тактового сигнала до начала работы, вне зависимости от источника тактового сигнала.

Описание пробуждения из режима микропотребления или режима расширенного ожидания, когда T/C2 работает в асинхронном режиме: когда срабатывает условие прерывания, процесс пробуждения начинается на следующем такте тактового сигнала таймера, т. о. таймер/счетчик всегда инкрементирован по меньшей мере на единицу до того, как ЦПУ прочитает его значение. После пробуждения микроконтроллер останавливается в течение четырех тактов, выполняется подпрограмма обработки прерывания и затем выполняется инструкция, следующая за командой SLEEP.

Чтение TCNT2 сразу после пробуждения из режима микропотребления может дать некорректный результат. Поскольку TCNT2 тактируется асинхронным сигналом, чтение TCNT2 должно быть выполнено после того, как регистр синхронизируется сигналом $clk_{I/O}$. Синхронизация выполняется на каждый нарастающий фронт TOSC1. Когда устройство пробуждается из режима микропотребления и сигнал $clk_{I/O}$ снова становится активным, будет читаться предыдущее значение TCNT2 до следующего нарастающего фронта TOSC1. Фаза тактового сигнала TOSC1 после пробуждения из режима микропотребления непредсказуема, поскольку зависит от времени пробуждения. Рекомендуется следующая процедура чтения TCNT2 после пробуждения:

1 Запись любого значения либо в OCR2, либо в TCCR2.

2 Ожидание сброса соответствующего флага в ASSR.

3 Чтение TCNT2.

Во время асинхронных операций синхронизация флагов прерываний для асинхронного таймера занимает три процессорных такта плюс один такт сигнала тактирования таймера. Значение таймера т. о. увеличивается по меньшей мере на единицу до того, как ЦПУ сможет прочитать значение таймера, приводящее к установке флага перерывания. Вывод «выход сравнения» изменяется по сигналу тактирования таймера и не синхронизируется тактовым сигналом ЦПУ.

Регистр маски прерывания таймера/счетчика – TIMSK

Бит	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Чт./Зап.	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 при совпадении сравнения на выходе таймера/счетчика

Если разряд OCIE2 и разряд «I» в регистре состояния установлен, то разрешается прерывание совпадения таймера/счетчика 2. Соответствующее прерывание выполняется, если происходит совпадение при сравнении в таймере/счетчике 2 (т. е. когда разряд OCF2 установлен в регистре флага прерывания таймера/счетчика – TIFR).

Разряд 6: TOIE2 – разрешение прерывания переполнения таймера/счетчика

Если разряд TOIE2 и разряд «I» в регистре состояния установлен, то разрешается прерывание переполнения таймера/счетчика 2. Соответствующее прерывание выполняется при появлении переполнения в таймере/счетчике 2 (т. е. когда разряд TOV2 устанавливается в регистре флага прерывания таймера/счетчика – TIFR).

Регистр флага прерывания таймера/счетчика – TIFR

Бит	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
Чт./Зап.	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 сравнения выхода

Разряд OCF2 устанавливается, когда происходит совпадение при сравнении между таймером/счетчиком 2 и данными с OCR2 – регистре сравнения по выходу. OCF2 очищается аппаратным способом при выполнении соответствующего вектора прерываний. В качестве альтернативы OCF2 очищается путем записи логической единицы в разряд флага. Когда устанавливается разряд «I» в SREG, OCIE2 (разрешение прерывания совпадения при сравнении таймера/счетчика 2) и OCF2, то выполняется прерывание при сравнении совпадения таймера/счетчика 2.

Разряд 6: TOV2 – флаг переполнения таймера/счетчика 2

Разряд TOV2 устанавливается, когда происходит переполнение таймера/счетчика 2. TOV2 очищается аппаратным способом при выполнении соответствующего вектора прерывания. В качестве альтернативы TOV2 очищается путем записи логической единицы в разряд флага. Когда устанавливаются разряды SREG «I», TOIE2 (разрешение прерывания переполнения таймера/счетчика 2) и разряд TOV2, то выполнится прерывание при переполнении таймера/счетчика 2. В режиме фазовой коррекции ШИМ этот разряд устанавливается, когда таймер/счетчик 2 изменяет направление счета при 0x00.

Устройство предварительного деления таймера/счетчика 2

Источник тактирования таймера/счетчика 2 называется clk_{T2S}. По умолчанию, этот сигнал подключен к основному системному сигналу $clk_{I/O}$. Установка бита AS2 в регистре ASSR приводит к тому, что таймер/счетчик начинает тактироваться от асинхронного сигнала вывода TOSC1. Это разрешает использовать Т/С2 как счетчик реального времени. Когда AS2 установлен, выводы TOSC1 и TOSC2 отключены от порта С. Кварц может быть подключен между выводами TOSC1 и TOSC2, чтобы работать в качестве независимого источника тактирования Т/С2. Осциллятор оптимизирован для использования часового кварца (32 768 Гц). Использование внешнего источника тактирования на TOSC1 не рекомендуется.

Для Т/С2 возможен выбор следующих коэффициентов предделения частоты: $clk_{T2S}/8$, $clk_{T2S}/32$, $clk_{T2S}/64$, $clk_{T2S}/128$, $clk_{T2S}/256$, $clk_{T2S}/1024$. Дополнительно clk_{T2S} может быть равен нулю. Установка бита PSR2 в регистре SFIOR очищает предделитель. Это позволяет пользователю работать с определенным коэффициентом предделения (смотри рисунок 3.62).

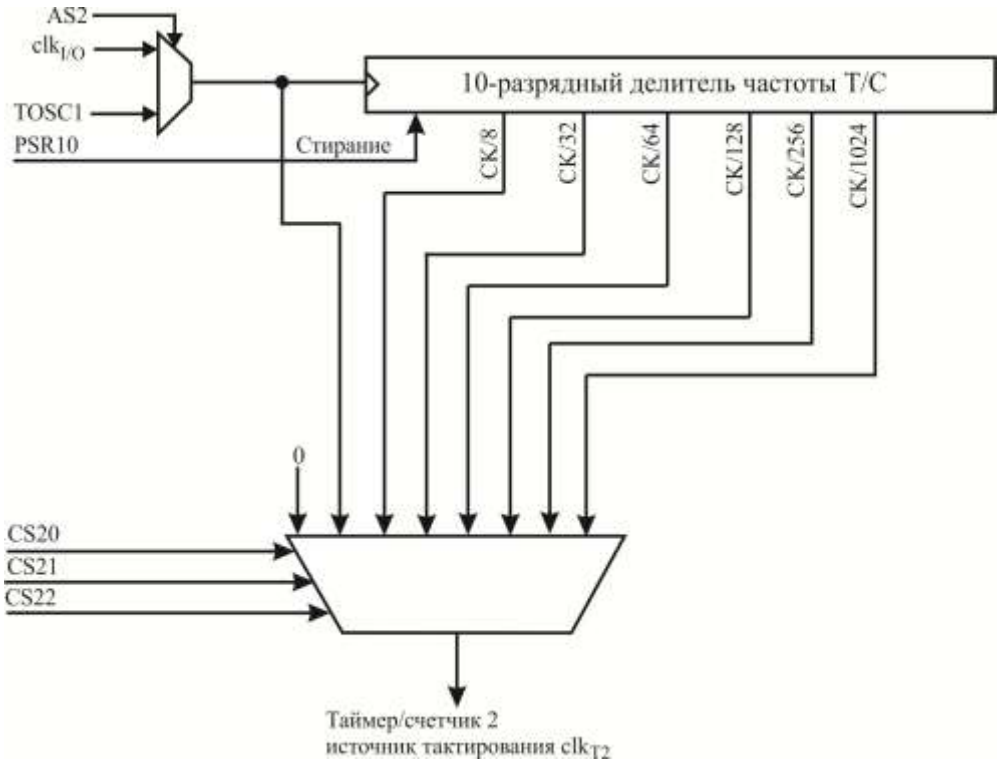


Рисунок 3.62 – Предделитель таймера/счетчика 2

Регистр специальных функций – SFIOR

Бит	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10	SFIOR
Чт./Зап.	Ч/З	Ч/З	Ч/З	Ч	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 1: PSR2 – сброс предделителя Т/С2

Когда этот бит устанавливается, предделитель Т/С2 будет сброшен. Этот бит будет очищен аппаратно после завершения сброса. Запись логического нуля в разряд не будет иметь эффекта. Этот бит всегда читается, как ноль, если Т/С2 тактируется от внутреннего источника тактирования ЦПУ. Если этот бит записывается когда Т/С2 работает в асинхронном режиме, бит останется установленным до тех пор, пока предделитель не будет сброшен.

3.14 Последовательный периферийный интерфейс (SPI)

Последовательный периферийный интерфейс, реализованный в микроконтроллере, имеет два назначения. Прежде всего через него может быть осуществлено программирование микроконтроллера (так называемый режим последовательного программирования). Использование интерфейса SPI в этом качестве будет описано в главе «Программирование».

Вторым назначением интерфейса является организация высокоскоростного обмена данными между микроконтроллером и различными периферийными устройствами, такими как цифровые потенциометры ЦАП/АЦП, Flash ПЗУ и др. Посредством этого интерфейса также может производиться обмен данными между несколькими микроконтроллерами. Использование интерфейса SPI в качестве высокоскоростного канала связи и рассматривается в данном подразделе.

При обмене данными по интерфейсу SPI микроконтроллер AVR может работать как в режиме Master, так и в режиме Slave. При этом пользователь может задать следующие параметры:

- скорость передачи (четыре программируемых значения);
- формат передачи (от младшего разряда к старшему или наоборот).

Дополнительной возможностью подсистемы SPI является «пробуждение» микроконтроллера из режима холостого хода при поступлении данных.

3.14.1 Функционирование модуля

Структурная схема модуля SPI приведена на рисунке 3.63.

Модуль SPI использует четыре вывода микроконтроллера. Как и для большинства прочих периферийных устройств эти выводы являются линиями порта ввода-вывода общего назначения.

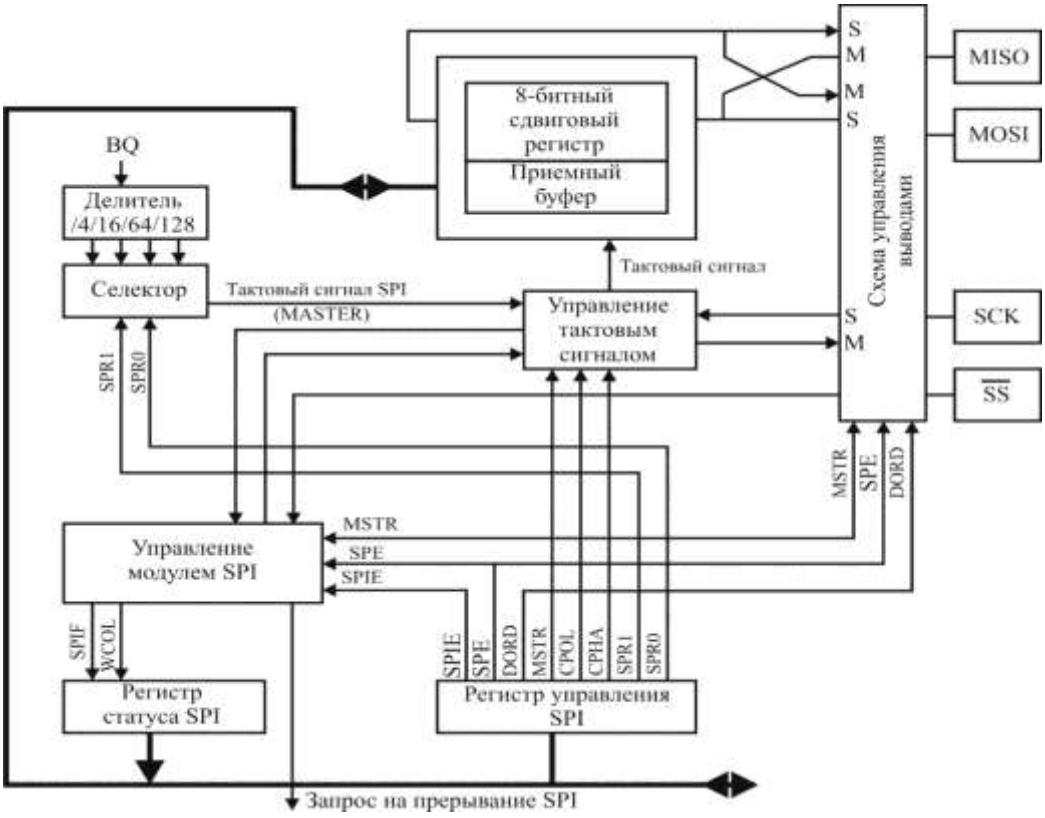


Рисунок 3.63 – Структурная схема модуля SPI

Таблица 3.55 – Выводы, используемые модулем SPI

Вывод	Направление, ведущий SPI	Направление, ведомый SPI
MOSI	Определено пользователем	Вход
MISO	Вход	Определено пользователем
SCK	Определено пользователем	Вход
SS#	Определено пользователем	Вход

Примечание – Смотрите 3.8.8 «Альтернативные функции порта B» для детального описания, как определить направление выводов SPI.

Как видно из таблицы 3.55, в некоторых случаях пользователь должен самостоятельно задать режим работы вывода, используемого модулем SPI , в соответствии с его назначением (см. далее по тексту). Причем возможность управления внутренними подтягивающими резисторами выводов, работающих как входы, сохраняется независимо от способа управления их режимом работы.

Для управления модулем SPI предназначен регистр управления SPCR, расположенный по адресу $0D ($2D). Формат этого регистра приведен ниже, краткое описание функций разрядов регистра приведено в таблице 3.56. Подробно использование различных разрядов регистра будет описано далее.

Бит	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Чт./Зап.	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	

Таблица 3.56 – Регистр SPCR

Разряд	Название	Описание
7	SPIE	Разрешение прерывания от SPI
6	SPE	Включение/выключение SPI
5	DORD	Порядок передачи данных
4	MSTR	Выбор режима работы («ведущий»/«ведомый»)
3	CPOL	Полярность тактового сигнала
2	CPHA	Фаза тактового сигнала
1–0	SPR1–SPR0	Скорость передачи

Контроль состояния модуля осуществляется с помощью регистра состояния SPSR (доступен только для чтения), расположенного по адресу $0E ($2E). Формат этого регистра приведен ниже, назначение его разрядов описано в таблице 3.57.

Бит	7	6	5	4	3	2	1	0	
	SPIF	WCOL	-	-	-	-	-	SPI2X	SPSR
Чт./Зап.	Ч	Ч	Ч	Ч	Ч	Ч	Ч	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Таблица 3.57 – Описание регистра SPSR

Разряд	Название	Описание
7	SPIF	Флаг прерывания от SPI. Данный флаг устанавливается по окончании передачи очередного байта. Если флаг SPIE регистра SPCR установлен и прерывания разрешены, одновременно с установкой флага генерируется прерывание от SPI. Также флаг SPIF устанавливается при переводе микроконтроллера из режима «ведущий» в режим «ведомый» посредством вывода SS#. Флаг сбрасывается аппаратно либо при старте подпрограммы обработки прерывания, либо после чтения регистра состояния SPI с последующим обращением к регистру данных
6	WCOL	Флаг конфликта записи. Данный флаг устанавливается при попытке записи в регистр данных во время передачи очередного байта. Флаг сбрасывается аппаратно после чтения регистра статуса SPSR с последующим обращением к регистру данных SPI
5–1	-	Зарезервированы, читаются как ноль
0	SPI2X	Удвоение скорости. Когда этот бит установлен, скорость работы будет удвоена, если устройство работает в режиме «ведущий». Это значит, что минимальный SCK период будет равен двум периодам тактового сигнала ЦПУ. Когда SPI сконфигурирован как «ведомый», устройство гарантирует работу при частоте в четыре или более раз превышающей частоту тактового сигнала ЦПУ

Передаваемые данные записываются, а принимаемые – считываются из регистра данных SPDR, расположенного по адресу $0F ($2F). Запись в этот регистр инициирует начало передачи, а при его чтении считывается содержимое приемного буфера сдвигового регистра. Поэтому этот регистр можно назвать буфером между регистровым файлом микроконтроллера и сдвиговым регистром модуля SPI.

Соединение двух микроконтроллеров (ведущий-ведомый) по интерфейсу SPI показано на рисунке 3.65. Вывод SCK ведущего микроконтроллера является выходом тактового сигнала, а ведомого микроконтроллера – входом.

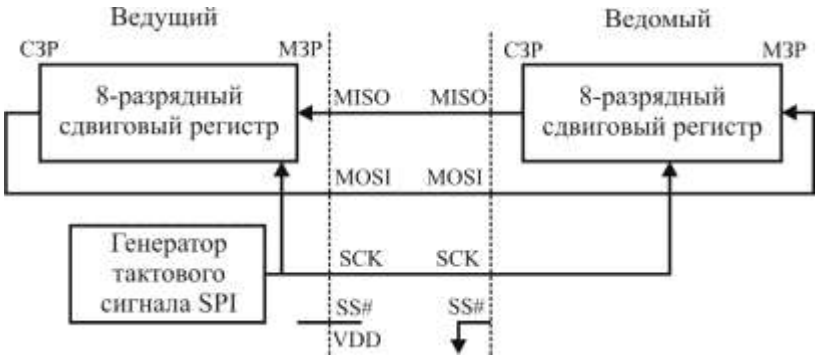


Рисунок 3.65 – Соединение микроконтроллеров по интерфейсу SPI

Перед выполнением обмена необходимо прежде всего разрешить работу модуля SPI. Для этого следует установить разряд SPE регистра SPCR. Режим работы определяется состоянием разряда MSTR этого регистра: если разряд установлен в «1», микроконтроллер работает в режиме «ведущий», если сброшен в «0» – в режиме «ведомый».

Передача данных осуществляется следующим образом. При записи в регистр данных SPI ведущего микроконтроллера запускается генератор тактового сигнала модуля SPI, и данные начинают поразрядно выдаваться на вывод MOSI и, соответственно, поступать на вывод MOSI ведомого микроконтроллера. Порядок передачи разрядов данных определяется состоянием разряда DORD регистра SPCR. Если разряд установлен в «1», первым передается младший разряд байта, если же сброшен в «0» – старший разряд. После выдачи последнего разряда текущего байта генератор тактового сигнала останавливается с одновременной установкой в «1» флага «Конец передачи» (SPIF). Если прерывания от модуля SPI разрешены (флаг SPIE регистра SPCR установлен в «1»), генерируется запрос на прерывание. При подключении к ведущему устройству нескольких ведомых, что разрешено спецификацией SPI, выбор конкретного ведомого устройства осуществляется подачей на его вход SS# сигнала НИЗКОГО уровня.

Образно говоря, два сдвиговых регистра ведомого и ведущего устройств можно считать одним распределенным 16-разрядным циклическим сдвиговым регистром, как показано на рисунке 3.65. Одновременно с передачей данных от ведущего к ведомому происходит передача и в обратном направлении. Таким образом, в каждом цикле сдвига происходит обмен данными между устройствами.

В модуле используется одинарная буферизация при передаче и двойная – при приеме. Это означает, что готовый для передачи байт данных не может быть записан в регистр данных SPI до окончания предыдущего цикла обмена. При попытке изменить содержимое регистра данных во время передачи устанавливается в «1» флаг WCOL регистра SPSR. Сбрасывается этот флаг после чтения регистра SPSR с последующим обращением к регистру данных SPI. Соответственно во время приема принятый байт должен быть прочитан из регистра данных SPI до того, как в сдвиговый регистр поступит последний разряд следующего байта. В противном случае первый байт будет потерян.

3.14.2 Режимы передачи данных

Спецификация интерфейса SPI предусматривает четыре режима передачи данных. Эти режимы различаются соответствием между фазой (момент считывания сигнала) тактового сигнала SCK, его полярностью и передаваемыми данными. Всего существует четыре такие комбинации, определяемые состоянием разрядов CPHA и CPOL регистра SPCR (см. таблицу 3.58).

Таблица 3.58 – Задание режима передачи данных

Разряд	Описание
CPOL	Полярность тактового сигнала. «0» – генерируются импульсы положительной полярности, при отсутствии импульсов на выводе присутствует низкий логический уровень. «1» – генерируются импульсы отрицательной полярности, при отсутствии импульсов на выводе присутствует высокий логический уровень
CPHA	Фаза тактового сигнала. «0» – обработка данных производится по переднему фронту импульсов сигнала SCK (для CPOL = 0 – по нарастающему, а для CPOL = 1 – по спадающему фронту). «1» – обработка производится по заднему фронту импульсов сигнала SCK

Соответствующие этим режимам форматы обмена данными через SPI приведены на рисунках 3.66 и 3.67 (передача ведется от старшего разряда к младшему). Частота тактового сигнала SCK и, соответственно, скорость передачи данных по интерфейсу определяются состоянием разрядов SPR1–SPR0 регистра SPCR (см. таблицу 3.59). Речь идет о микроконтроллере, работающем в режиме «ведущий», т. к. именно он является источником

тактового сигнала. Для устройства, находящегося в режиме «ведомый», состояние этих разрядов безразлично.

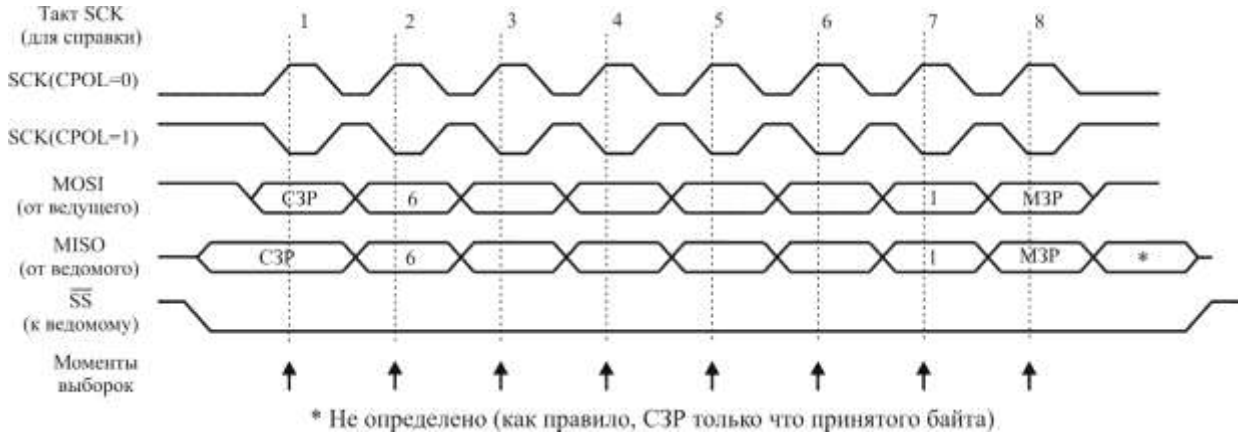


Рисунок 3.66 – Передача данных при CPHA = 0 и DORD = 0

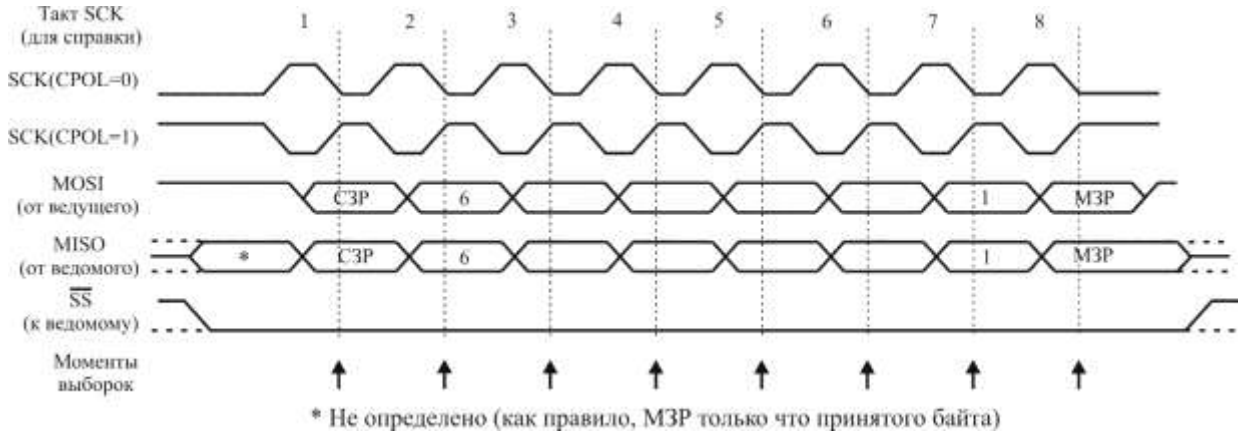


Рисунок 3.67 – Передача данных при CPHA = 1 и DORD = 0

Таблица 3.59 – Задание частоты тактового сигнала

SPR1	SPR0	Частота сигнала SCK
0	0	$f_{clk/4}$
0	1	$f_{clk/16}$
1	0	$f_{clk/64}$
1	1	$f_{clk/128}$

3.14.3 Использование вывода SS#

Этот вывод предназначен для выбора активного ведомого устройства и в режиме «ведомый» всегда является входом. При подаче на него напряжения НИЗКОГО уровня модуль SPI активируется и вывод MOSI переключается в режим вывода данных (если это задано пользователем). Остальные выводы модуля SPI являются в этом режиме входами. А при подаче на вывод SS# напряжения ВЫСОКОГО уровня, все выводы модуля SPI переключаются в режим ввода данных. При этом модуль переходит в неактивное состояние и прием данных не производится.

Следует помнить, что каждый раз, когда на вывод SS# подается напряжение ВЫСОКОГО уровня, происходит сброс модуля SPI. Соответственно, если изменение состояния этого вывода произойдет во время передачи данных, то прием и передача немедленно прекратятся, а передаваемый и принимаемый байты будут потеряны.

Если же микроконтроллер находится в режиме Master (разряд MSTR регистра SPCR установлен в «1»), направление передачи данных через вывод SS# определяется пользователем. Если вывод сконфигурирован как выход, он работает как линия вывода общего назначения и не влияет на работу модуля SPI. Если же он сконфигурирован как вход, то для обеспечения нормальной работы модуля SPI на него должен быть подан сигнал ВЫСОКОГО уровня. Дело в том, что подача на этот вход сигнала НИЗКОГО уровня от какой-либо внешней схемы будет воспринята модулем SPI как выбор данного микроконтроллера в качестве ведомого и, соответственно, начало передачи ему данных. Во избежание конфликта на шине система SPI в таких случаях выполняет следующие действия:

1 Флаг MSTR регистра SPCR сбрасывается, и микроконтроллер переключается в режим Slave. Как следствие, выводы MOSI и SCK начинают функционировать как входы.

2 Устанавливается флаг SPIF регистра SPSR, генерируя запрос на прерывание от SPI. Если прерывания от SPI разрешены и флаг «I» регистра SREG установлен в «1», происходит запуск подпрограммы обработки прерывания.

Таким образом, если ведущий микроконтроллер использует передачу данных, управляемую прерыванием, и существует вероятность подачи на вход SS# сигнала НИЗКОГО уровня, в подпрограмме обработки прерывания от SPI обязательно должна осуществляться проверка состояния флага MSTR.

При обнаружении сброса этого флага он должен быть программно установлен обратно в «1» для обратного перевода микроконтроллера в режим Master.

3.15 Последовательный синхронно-асинхронный приемопередатчик UART

Универсальный синхронный и асинхронный последовательный приемопередатчик предназначен для организации гибкой последовательной связи.

Отличительные особенности:

- полнодуплексная работа (раздельные регистры последовательного приема и передачи);
- асинхронная или синхронная работа;
- «ведущее» или «ведомое» тактирование в синхронном режиме работы;
- высокая разрешающая способность генератора скорости связи;
- поддержка формата передаваемых данных с 5, 6, 7, 8 или 9 битами данных и одним или двумя стоп-битами;
- аппаратная генерация и проверка бита паритета (четность/нечетность);
- определение переполнения данных;
- определение ошибки в структуре посылки;
- фильтрация шума с детекцией ложного старт-бита и цифровым ФНЧ;
- три раздельных прерывания по завершении передачи, освобождении регистра передаваемых данных и завершении приема;
- режим многопроцессорной связи;
- режим удвоения скорости связи в асинхронном режиме.

3.15.1 Краткий обзор

В состав UART входят три основных блока: тактовый генератор, передатчик и приемник. Регистры управления используются всеми блоками. Логика тактового генератора состоит из логики синхронизации, связанной с внешним тактовым входом (используется в подчиненном режиме), и генератора скорости связи. Вывод XCK (синхронизация передачи) используется только в режиме синхронной передачи. Передатчик состоит из одного буфера записи, последовательного сдвигового регистра, генератора паритета и управляющей логики, которая поддерживает различные форматы последовательной посылки. Буфер записи позволяет непрерывно передавать данные без каких-либо задержек между переда-

чей посылок. Приемник является более сложным блоком UART, т. к. в его состав входят модули обнаружения данных и синхронизации. Модули обнаружения необходимы для асинхронного приема данных. Помимо модулей обнаружения в приемник входят: устройство проверки паритета, сдвиговый регистр и двухуровневый приемный буфер (UDR). Приемник поддерживает те же последовательные форматы, что и передатчик, и может определить ошибку в посылке (кадре), переполнение данных и ошибку паритета.

Логика генерации тактовых импульсов формирует основную синхронизацию приемника и передатчика. UART поддерживает четыре режима работы синхронизации: нормальная асинхронная, асинхронная с удвоением скорости, ведущая синхронная и подчиненная синхронная. Бит UMSEL в регистре С управления и статуса (UCSRC) позволяет выбрать асинхронную или синхронную работу. Удвоение скорости (только в асинхронном режиме) управляется битом U2X в регистре UCSRA. При использовании синхронного режима (UMSEL = 1) соответствующий бит в регистре направления данных для вывода XCK (DDR_XCK) задает: будет синхронизация внутренней («ведущий» режим) или внешней (подчиненный режим). Вывод XCK активен только при использовании синхронного режима.

На рисунке 3.68 показана функциональная схема логики синхронизации.

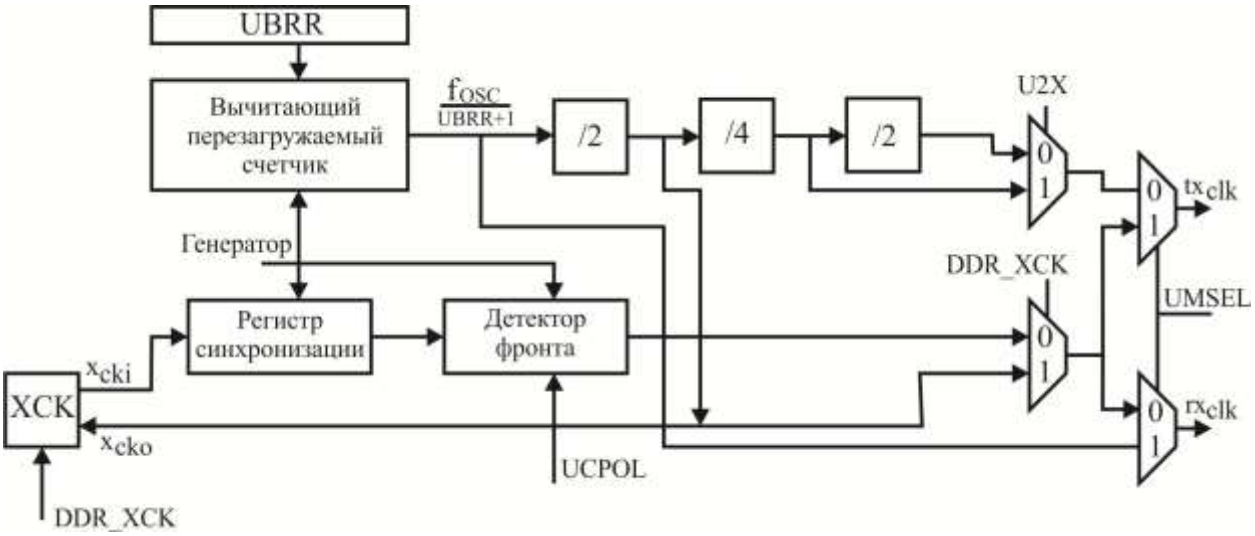


Рисунок 3.68 –Функциональная схема логики синхронизации UART

Описание сигналов:

- tx_{clk} – синхронизация передатчика (внутренний сигнал);
- rx_{clk} – основная синхронизация приемника (внутренний сигнал);
- x_{cki} – вход от вывода XCK (внутренний сигнал). Используется для синхронной подчиненной работы;
- x_{cko} – выход синхронизации к выводу XCK (внутренний сигнал). Используется в «ведущем» синхронном режиме;
- f_{osc} – вывод частоты BQ (системная синхронизация).

Генерация внутренней синхронизации – генератор скорости связи

Внутренняя синхронизация используется для асинхронного и «ведущего» синхронного режимов работы.

Регистр генератора скорости связи (UBRR) и связанный с ним вычитающий счетчик функционируют как программируемый предделитель или генератор скорости связи. Вычитающий счетчик тактируется системной синхронизацией (f_{osc}) и перезагружается значением из регистра UBRR всякий раз при достижении нулевого значения или после записи регистра UBRRL. Тактовый сигнал генерируется всякий раз при достижении счетчиком

нулевого значения. Данный тактовый сигнал является тактовым выходом генератора скорости связи (равен $f_{osc}/(UBRR + 1)$). Передатчик делит частоту генератора скорости связи на 2, 8 или 16 в зависимости от режима работы. Модули обнаружения синхронизации и данных приемника подключены непосредственно к тактовому выходу генератора скорости связи. Цифровой автомат модулей обнаружения использует 2, 8 или 16 состояний в зависимости от режима, задаваемого битами UMSEL, U2X и DDR_XCK.

В таблице 3.60 даны выражения для вычисления скорости связи (в битах в секунду) и вычисления значений UBRR для каждого из рабочих режимов при использовании внутренне генерируемого тактового источника.

Таблица 3.60 – Выражения для вычисления установок регистра скорости связи

Режим работы	Выражение для вычисления скорости связи	Выражение для вычисления значения UBRR
Нормальный асинхронный режим (U2X=0)	$BAUD = f_{OSC}/16(UBRR+1)$	$UBRR = (f_{OSC}/16BAUD) – 1$
Асинхронный режим с удвоением скорости (U2X=1)	$BAUD = f_{OSC}/8(UBRR+1)$	$UBRR = (f_{OSC}/8BAUD) – 1$
Синхронный «ведущий» режим	$BAUD = f_{OSC}/2(UBRR+1)$	$UBRR = (f_{OSC}/2BAUD) – 1$

Примечания
1 BAUD – скорость связи (в битах в секунду, бод).
2 f_{OSC} – частота синхронизации системного генератора.
3 UBRR – содержимое регистров UBRRH и UBRRL (0–4095).
4 Скорость связи представлена в битах в секунду (бод).

3.15.2 Работа с удвоением скорости связи (U2X)

Скорость передачи данных может быть удвоена, если установить бит U2X в регистре UCSRA. Установка данного бита оказывает действие только в асинхронном режиме. При использовании синхронного режима необходимо установить нулевое значение данного бита.

Установка данного бита приводит к уменьшению коэффициента деления частоты генератора скорости связи с 16 до 8, тем самым удваивая скорость асинхронной связи. Следует обратить внимание, что в этом случае приемник сокращает количество выборок с 16 до 8 при обнаружении синхронизации и данных, поэтому, при использовании данного режима, необходимо использовать более точные установки скорости связи и более стабильный тактовый источник. Для передатчика удвоение скорости не связано с какими-либо ограничениями.

3.15.3 Внешняя синхронизация

Внешняя синхронизация используется в синхронном приемном режиме работы (см. рисунок 3.70).

Во избежание возможности возникновения метастабильности вход внешней синхронизации с вывода XCK связан с регистром синхронизации. Выход регистра синхронизации проходит через детектор фронтов, а только затем используется приемником и передатчиком. На данный процесс затрачивается два такта синхронизации ЦПУ, и поэтому максимальная частота внешней синхронизации на выводе XCK ограничивается выражением.

$$f_{XCK} < f_{OSC} / 4.$$

Частота f_{OSC} зависит от стабильности системного источника синхронизации. В связи с этим рекомендуется учесть некоторый запас для предотвращения возможности потери данных из-за колебаний частоты.

3.15.4 Режим синхронной связи

Если используется режим синхронной связи (UMSEL = 1), то вывод XCK используется или как вход синхронизации («ведомый» или «подчиненный» режим) или как выход синхронизации («ведущий» режим). Зависимость между тактовыми фронтами и выборкой данных или изменением данных одна и та же. Основной принцип работы заключается в том, что выборка вводимых данных (на RXD) осуществляется фронтом XCK, который противоположен фронту, по которому происходит изменение выходных данных (на TXD).

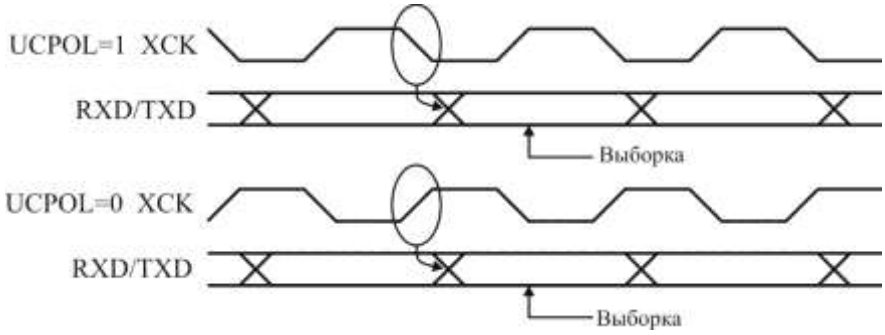


Рисунок 3.69 – Временная диаграмма для синхронного режима XCK

Бит UCPOL регистра UCRSC выбирает какой фронт XCK используется для выборки данных, а какой для изменения данных. На рисунке 3.69 показано, что при UCPOL = 0 изменение данных происходит по нарастающему фронту XCK, а выборка – по падающему фронту XCK. Если установлен бит UCPOL = 1, то изменение данных происходит по падающему фронту XCK, а выборка – по нарастающему фронту XCK.

Последовательная посылка состоит из бит данных, бит синхронизации (старт и стоп-биты), а также опционального бита паритета для поиска ошибок. UART поддерживает все 30 комбинаций следующих форматов посылок:

- 1 старт-бит;
- 5, 6, 7, 8 или 9 бит данных;
- без паритета, с битом четности, с битом нечетности;
- 1 или 2 стоп-бита.

Посылка начинается со старт-бита, а за ним следует передача бит данных, начиная с самого младшего разряда. Затем следует передача остальных бит данных (максимальное число бит данных – девять), которая заканчивается передачей старшего разряда данных. Если разрешена функция контроля паритета, то сразу после бит данных передается бит паритета, а затем стоп-биты. После завершения передачи посылки имеется возможность либо передавать следующую посылку, либо перевести линию связи в состояние ожидания (высокий уровень). На рисунке 3.70 показана возможность сочетания форматов посылки. Наличие прямоугольной скобки указывает на опциональность данного формата посылки.

St – старт-бит имеет всегда низкий уровень;
0–8 – номера бита данных;
P – бит паритета: четность или нечетность;
Sp1, Sp2 – стоп-бит имеет всегда высокий уровень;
IDLE – состояние ожидания, в котором приостановлена передача на RXD или TXD.
В состоянии ожидания на линии должен быть высокий уровень.

Рисунок 3.70 – Сочетание форматов посылки

Формат посылки, который используется UART, задается битами UCSZ(2–0), UPM(1–0) и USBS в регистрах UCSRB и UCSRC. Приемник и передатчик используют одни и те же установки форматов. Следует обратить внимание, что изменение установок любого из этих бит может привести к повреждению текущего сеанса связи как для приемника, так и для передатчика.

Биты выбора длины передаваемых данных UCSZ(2–0) определяют из скольких бит данных состоит посылка. Биты режима паритета UART (UPM(1–0)) разрешают передачу/контроль бита паритета и устанавливают тип паритета: четность, нечетность. Выбрать один или два стоп-бита позволяет бит выбора стоп-бита UART (USBS). Приемник игнорирует второй стоп-бит. Флаг ошибки посылки FE позволяет выявить ситуацию, когда первый стоп-бит равен 0.

Вычисление бита паритета (четности)

Бит паритета вычисляется путем выполнения логической операции исключающего ИЛИ над всеми битами данных. Если используется нечетность, то результат этой операции инвертируется. Сказанное отражено в следующих выражениях:

$$P_{ЧЕТН} = d_{n-1} \wedge \ldots \wedge d_3 \wedge d_2 \wedge d_1 \wedge d_0 \wedge 0,$$
$$P_{НЕЧЕТН} = d_{n-1} \wedge \ldots \wedge d_3 \wedge d_2 \wedge d_1 \wedge d_0 \wedge ,1$$

где $P_{ЧЕТН}$ – бит четного паритета;
$P_{НЕЧЕТН}$ – бит нечетного паритета;
d_n – n-ый бит данных в посылке.

После разрешения бит паритета передается между последним битом данных и первым стоп-битом.

Перед началом сеанса связи необходимо выполнить инициализацию UART. Процесс инициализации обычно состоит из установки скорости связи, задания формата посылки и разрешения работы передатчика и приемника. Если используется управление связью по прерываниям, то во время инициализации необходимо, чтобы был сброшен флаг общего разрешения прерываний (т. е. необходимо запретить все прерывания).

Если необходимо выполнить повторную инициализацию UART, например, для изменения скорости связи или формата посылки, то необходимо убедиться, что во время инициализации передача приостановлена. Флаг TXC может использоваться для проверки завершения работы передатчика, а флаг RXC – для проверки отсутствия в приемном буфере несчитанных данных. Следует обратить внимание, что при использовании флага TXC, он должен сбрасываться программно перед началом каждой передачи (перед записью в UDR).

В следующих примерах показаны функции для простой инициализации UART на Ассемблере и Си. В примерах предполагается, что используются управление связью по опросу флагов состояния (не по прерываниям) и фиксированный формат посылки. Ско-

рость связи выступает как параметр функции. Для примера на ассемблере предполагается, что параметр скорости связи записан перед вызовом функции в регистры r17–r16.

Пример кода на Ассемблере [1)]

```
UART_Init:
; Установка скорости связи
out UBRRH, r17
out UBRRL, r16
; Разрешение работы приемника и передатчика
ldi r16, (1<<RXEN)|(1<<TXEN)
out UCSRB,r16
; Установка формата посылки: 8 бит данных, 2стоп-бита
ldi r16, (1<<URSEL)|(1<<USBS)|(3<<UCSZ0)
out UCSRC,r16
ret
```

Пример кода на Си [1)]

```
void UART_Init( unsigned int baud )
(
/* Установка скорости связи */
UBRRH = (unsigned char)(baud>>8);
UBRRL = (unsigned char)baud;
/* Разрешение работы передатчика и приемника */
UCSRB = (1<<RXEN)|(1<<TXEN);
/* Установка формата посылки: 8 бит данных, 2 стоп-бита */
UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
)
```

[1)] В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода, необходимо заменить инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции «LDS» и «STS» в сочетании с «SBRS», «SBRC», «SBR» и «CBR».

Более совершенные процедуры инициализации могут использовать расширенный интерфейс функции, где в качестве параметров выступают, например, формат посылки, отключение прерываний и т. д. Однако, в большинстве приложений используются фиксированные установки скорости связи и управляющих регистров, поэтому для них представленные примеры могут быть непосредственно включены в основную программу или к процедурам инициализации других модулей ввода-вывода.

3.15.5 Передача данных – передатчик UART

Работа передатчика UART разрешается путем установки бита разрешения передачи (TXEN) в регистре UCSRB. После разрешения функция вывода TXD, как обычного порта, меняется на функцию выхода последовательной передачи данных. Скорость связи, режим работы и формат посылки должны быть установлены однократно перед началом какой-либо передачи. Если используется синхронная работа, то функция вывода XCK также заменяется на альтернативную – синхронизация передачи.

Передача посылок с 5–8 битами данных

Начало передачи инициируется записью передаваемых данных в буфер передатчика. ЦПУ может загрузить буфер передатчика путем записи в регистр UDR, расположенный в памяти ввода-вывода. Буферизованные данные в буфере передатчика будут перемещены в сдвиговый регистр после того, как он будет готов к отправке новой посылки. Запись в

сдвиговый регистр новых данных происходит в состоянии ожидания (когда передача завершена) или сразу после завершения передачи последнего стоп-бита предыдущей посылки. Если в сдвиговый регистр записаны новые данные, то начинается передача одной посылки на скорости, определенной в регистре скорости связи, битом U2X или XCK в зависимости от выбранного режима работы.

В следующих примерах представлены простые функции передачи через UART, использующие опрос флага освобождения регистра данных (UDRE). Если используется посылка с менее чем 8 бит данных, то старшие биты, записанные в UDR, игнорируются. Перед вызовом данной функции должна быть выполнена инициализация UART. Для кода на Ассемблере предполагается, что передаваемые данные записаны в регистр R16 перед вызовом процедуры.

Пример кода на Ассемблере [1)]

```
UART_Transmit:
; Ожидание освобождения буфера передатчика
sbis UCSRA,UDRE
rjmp UART_Transmit
; Помещение данных (r16) в буфер, отправка данных
out UDR,r16
ret
```

Пример кода на С [1)]

```
void UART_Transmit( unsigned char data )
(
/* Ожидание освобождения буфера передатчика  */
while ( !( UCSRA & (1<<UDRE)) );
/* Помещение данных в буфер, отправка данных */
UDR = data;
)
```

Отправка посылок с 9 битами данных

Если необходимо передавать 9 бит данных (UCSZ = 7), то девятый бит данных должен быть записан в бит TXB8 регистра UCSRB перед тем, как младший байт будет записан в UDR. В следующих примерах показаны функции для передачи 9 бит данных. Для кода на Ассемблере предполагается, что отправляемые данные предварительно записаны в регистры r17–r16.

Пример кода на Ассемблере [2)]

```
UART_Transmit:
; Ожидание освобождения буфера передатчика
sbis UCSRA,UDRE
rjmp UART_Transmit
; Копирование 9-го бита из r17 в TXB8
cbi UCSRB,TXB8
sbrc r17,0
sbi UCSRB,TXB8
; Помещение мл. байта данных (r16) в буфер, отправка данных
out UDR,r16
ret
```

Пример кода на Си [2)]

```
void UART_Transmit( unsigned int data )
(
/* Ожидание освобождения буфера передатчика */
while ( !( UCSRA & (1<<UDRE)) );
/* Копирование 9-го бита в TXB8 */
```

```
UCSRB &= ~(1<<TXB8);
if ( data & 0x0100 )UCSRB |= (1<<TXB8);
/* Помещение данных в буфер, отправка данных */
UDR = data;
```

[1)] В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода, необходимо заменить инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции «LDS» и «STS» в сочетании с «SBRS», «SBRC», «SBR» и «CBR».

Перед загрузкой новых данных для передачи в данной функции осуществляется ожидание освобождения буфера передатчика путем опроса флага UDRE.

[2)] Данные функции записаны как функции общего назначения. Они оптимизированы под статическое содержимое UCSRB, т. е. когда после инициализации в регистре UCSRB изменяется только бит TXB8. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода, необходимо заменить инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции «LDS» и «STS» в сочетании с «SBRS», «SBRC», «SBR» и «CBR».

Девятый бит данных может использоваться для индикации адреса посылки в многопроцессорном режиме связи или в других протоколах.

3.15.6 Флаги и прерывания передатчика

Передатчик UART имеет два флага, которые индицируют его состояние: флаг освобождения регистра данных UART (UDRE) и флаг завершения передачи (TXC). Оба флага могут использоваться для генерации прерываний. Флаг освобождения регистра данных (UDRE) индицирует готовность буфера передатчика принять новые данные. Данный бит устанавливается при освобождении передающего буфера и сбрасывается, когда буфер передатчика содержит данные для передачи, но которые еще не были переданы в сдвиговый регистр. Для совместимости с будущими микроконтроллерами в данный бит необходимо записывать ноль во время записи в регистр UCSRA.

Если записать логическую 1 в бит разрешения прерывания по освобождению регистра данных в регистре UCSRB, то прерывание по освобождению регистра данных UART будет выполняться всякий раз, когда устанавливается бит UDRE (с учетом того, что общие прерывания разрешены). UDRE сбрасывается при записи в UDR. Если используется управление связью по прерываниям, то в процедуре обработки прерывания по освобождению регистра данных необходимо или записать новые данные в регистр UDR для сброса флага UDRE, или выключить прерывание по освобождению регистра данных. В противном случае при выходе из процедуры обработки прерывания сразу возникнет новое прерывание.

Флаг завершения передачи (TXC) принимает единичное значение, если вся посылка в сдвиговом регистре передатчика была полностью сдвинута и в буфере передатчика отсутствуют новые данные для передачи. Флаг TXC автоматически сбрасывается при переходе на вектор обработки прерывания по завершении передачи или программно сбрасывается путем записи в него логическую 1. Флаг TXC полезно использовать при организации полудуплексной связи, где имеется необходимость перевода передающей стороны в режим приема после завершения передачи, тем самым достигая освобождения шины.

Если разрешено прерывание по завершении передачи (TXCIE=1) в регистре UCSRB, то прерывание по завершению передачи UART выполняется всякий раз, когда флаг TXC принимает единичное состояние (с учетом, что активно общее разрешение прерываний). При переходе на вектор обработки прерывания по завершении передачи UART нет необходимости сбрасывать флаг TXC, т. к. это происходит автоматически.

3.15.7 Генератор паритета

Генератор паритета вычисляет бит паритета для включения в состав последовательной посылки. Если бит паритета установлен (UPM1 = 1), то управляющая логика передатчика вставляет бит паритета между последним битом данных и первым стоп-битом в отправляемой посылке.

3.15.8 Отключение передатчика

Отключение передатчика после сброса TXEN наступит только тогда, когда завершится текущая и ожидаемая передача, т. е. когда в сдвиговом регистре передатчика и буфере передатчика не будет данных для передачи. После отключения передатчика отменяется альтернативное назначение вывода TXD.

3.15.9 Прием данных – приемник UART

Работа приемника UART разрешается, если записать логическую 1 в бит разрешения работы приемника (RXEN) в регистре UCSRB. После разрешения работы приемника обычное назначение вывода RXD заменяется на альтернативное: вход последовательного ввода данных приемника UART. Скорость связи, режим работы и формат посылки должны быть установлены однократно перед началом выполнения приема данных. Если используется синхронная работа, то вывод XCK будет использоваться для синхронизации связи.

3.15.10 Прием посылок с 5–8 битами данных

Приемник начинает прием данных только после определения действительного старт-бита. Выборка следующих за старт-битом бит данных происходит с частотой, равной скорости связи или частотой сигнала XCK, и размещается в сдвиговом регистре приемника. Второй стоп-бит приемником игнорируется. После получения первого стоп-бита, т. е. когда последовательная посылка полностью принята и находится в сдвиговом регистре приемника, содержимое сдвигового регистра перемещается в приемный буфер. Приемный буфер считывается при чтении регистра ввода-вывода UDR.

В следующих примерах приведены простые функции для организации приема данных, которые основаны на опросе состояния флага завершения приема (RXC). Если используется формат посылки с числом бит менее восьми, то после считывания содержимого UDR старшие неиспользуемые разряды будут обнулены. Перед вызовом данных функций должна быть выполнена инициализация UART.

Пример кода на Си [1)]

```
unsigned int UART_Receive( void )
(
unsigned char status, resh, resl;
/* Ожидание окончания приема данных */
while ( !(UCSRA & (1<<RXC)) );
/* Опрос статусных бит и 9-го бита данных перед чтением  данных из буфера */
status = UCSRA;
resh = UCSRB;
resl = UDR;
/* Если ошибка, то возврат -1 */
if ( status & (1<<FE)|(1<<DOR)|(1<<UPE) )
return -1;
```

```
/* Выделение  9-го бита данных перед выходом */
resh = (resh >> 1) & 0x01;
return ((resh << 8) | resl);
)
```

[1)] В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода, необходимо заменить инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции «LDS» и «STS» в сочетании с «SBRS», «SBRC», «SBR» и «CBR».

В данных функциях считываются все регистры ввода-вывода в файл регистров перед выполнением каких-либо вычислений. Такой подход позволяет наиболее оптимально использовать заполняемость буфера, т. к. буфер становится свободным для приема новых данных, как только это станет возможным.

3.15.11 Флаг и прерывание по завершению приема

Приемник UART имеет один флаг, который индицирует состояние приемника.

Флаг завершения приема (RXC) сигнализирует о наличии несчитанных данных в приемном буфере. Данный флаг равен «1», если имеются несчитанные данные, и равен «0», если буфер приемника свободен (т. е. не содержит каких-либо несчитанных данных). Если приемник отключается (RXEN = 0), то приемный буфер будет сброшен и флаг RXC примет нулевое значение.

Если установлен бит разрешения прерывания по завершению приема (RXCIE) в регистре UCSRB, то при установке флага RXC программа переходит на вектор обработки данного прерывания (при условии, что активно общее разрешение прерываний). Если используется организация связи с управлением по прерываниям, то при выполнении процедуры обработки запроса на прерывание по завершению приема необходимо считать данные из UDR, чтобы сбросить флаг RXC. В противном случае новое прерывание возникнет сразу после выхода из текущего.

3.15.12 Флаги ошибок приемника

Приемник UART имеет три флага ошибок: ошибка посылки (кадра) FE, переполнение данных DOR и ошибка паритета UPE. Данные флаги входят в состав регистра UCSRA. Общим свойством данных флагов является то, что они хранятся в приемном буфере вместе с той посылкой данных, для которой они отражают состояние ошибок. С учетом этого, необходимо следить, чтобы флаги ошибок считывались из регистра UCSRA перед чтением данных из приемного буфера (UDR), т. к. после чтения из UDR изменяется состояние буфера. Другим сходством флагов ошибок является невозможность программно повлиять на их состояние. Однако в целях совместимости с UART последующих микроконтроллеров, во время записи регистра UCSRA в позициях флагов ошибок необходимо указывать нулевые значения. Ни один из флагов ошибок не может вызвать прерывание.

Флаг ошибки посылки (кадра) FE индицирует состояние первого стоп-бита сохраненной в приемном буфере посылки. Флаг FE равен нулю, если стоп-бит имел корректное значение (логическая 1), и равен единице, если некорректное, т. е. ноль. Данный флаг может использоваться для выявления условия рассинхронизации, обрыва связи и манипуляции над протоколом связи. Флаг FE не изменяется при установке бита USBS в регистре UCSRC, т. к. приемник игнорирует все стоп-биты за исключением первого. Для совместимости с последующими микроконтроллерами в позиции данного бита необходимо указывать ноль во время записи в регистр UCSRA.

Флаг переполнения данных (DOR) сигнализирует о потере данных из-за переполнения приемного буфера. Переполнение данных возникает, если приемный буфер заполнен

(две посылки), в сдвиговом регистре ожидает считывания только что принятая посылка и обнаружен новый старт-бит. Если флаг DOR установлен, то значит одна или более последовательных посылок потеряны между последним и следующим считанными значениями из UDR. Для совместимости с будущими микроконтроллерами в позицию данного бита необходимо всегда записывать логический 0 во время записи в регистр UCSRA. Флаг DOR сбрасывается, если принятая посылка была успешно перемещена из сдвигового регистра в приемный буфер.

Флаг ошибки паритета (UPE) сигнализирует, что во время приема посылки была обнаружена ошибка паритета. Если контроль паритета отключен, то данный флаг всегда имеет нулевое значение. Для совместимости с новыми разработками микроконтроллеров в позицию данного бита необходимо всегда записывать ноль во время записи в регистр UCSRA.

3.15.13 Устройство проверки паритета

Устройство проверки паритета UART становится активным после установки бита режима паритета UPM1. Тип контроля паритета: четность или нечетность задается битом UPM0. После активизации устройство проверки паритета вычисляет паритет принятых данных и сравнивает полученное значение с принятым вместе с этими данными в одной посылке битом паритета. Результат сравнения запоминается в приемном буфере вместе с принятыми данными и стоп-битом. Флаг ошибки паритета UPE может быть считан программно тогда, когда в посылке имеется ошибка паритета. Бит UPE устанавливается, если в посылке, которая может быть считана из приемного буфера, имеется ошибка паритета и во время приема этой посылки был разрешен контроль паритета (UPM1 = 1). Данный бит должен быть опрошен до считывания буфера приемника (UDR).

Отключение приемника

В отличие от передатчика отключение приемника происходит незамедлительно. При этом принимаемые данные будут потеряны. После отключения (т. е. когда RXEN = 0) приемник далее не поддерживает альтернативные настройки вывода порта RXD. Приемный буфер FIFO сбрасывается после отключения приемника, следовательно, оставшиеся в нем данные будут потеряны.

Сброс приемного буфера

Приемный буфер FIFO сбрасывается после отключения приемника, т. е. полностью освобождается от своего содержимого. Несчитанные данные будут потеряны. Если буфер необходимо очистить в процессе нормальной работы, например, при возникновении ошибок, то необходимо считывать содержимое UDR пока не очиститься флаг RXC. В следующем примере показано как очистить буфер приемника.

Пример кода на Ассемблере[1)]

```
UART_Flush:
sbis UCSRA, RXC
ret
in r16, UDR
rjmp UART_Flush
```

Пример кода на Си [1)]

```
void UART_Flush( void )
(
unsigned char dummy;
while ( UCSRA & (1<<RXC) ) dummy = UDR;
)
```

[1)] В примере предполагается, что подключен файл специфических заголовков. Для регистров ввода-вывода, которые расположены в области памяти расширенного ввода-вывода, необходимо заменить инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» на инструкции, осуществляющие доступ к расширенной памяти ввода-вывода. Обычно это инструкции «LDS» и «STS» в сочетании с «SBRS», «SBRC», «SBR» и «CBR».

3.15.14 Асинхронный прием данных

UART содержит блоки обнаружения данных и синхронизации для управления асинхронным приемом данных. Логика обнаружения синхронизации используется для синхронизации с внутренним генератором скорости связи для обеспечения возможности ввода последовательной посылки с выводов RXD. Логика обнаружения данных осуществляет выборку и фильтрацию (ФНЧ) каждого входящего бита данных, тем самым увеличивая помехоустойчивость приемника. Рабочий диапазон асинхронного приема определяется точностью встроенного генератора скорости связи, точностью скорости входящей посылки и размером посылки (количество бит).

Асинхронный поиск синхронизации

Логика обнаружения синхронизации стабилизирует во времени работу приемника с входящей последовательной посылкой. На рисунке 3.71 иллюстрируется процесс поиска старт-бита во входящей посылке. Частота выборок в 16 раз выше скорости связи для нормального режима и в восемь раз выше для режима удвоения скорости. Горизонтальные стрелки иллюстрируют возможный уход синхронизации в процессе выборки. Следует обратить внимание на более высокую рассинхронизацию во времени при использовании режима удвоения скорости (U2X = 1). Выборки, обозначенные номером 0, соответствуют состоянию ожидания на линии RXD (т. е. при неактивной связи).

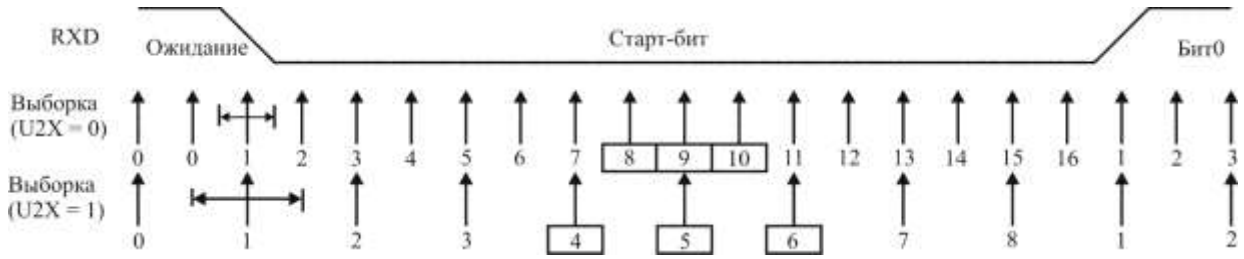


Рисунок 3.71 – Процесс поиска старт-бита во входящей посылке

Если логика обнаружения синхронизации определяет переход из высокого (состояние ожидания) к низкому (старт) состоянию на линии RXD, то инициируется последовательность действий по обнаружению старт-бита. Если принять, что выборка 1 означает первую выборку с нулевым значением, тогда по выборкам 8, 9, 10 в нормальном режиме и выборкам 4, 5, 6 в режиме удвоения скорости определяется действительность старт-бита (на рисунке эти выборки помещены в рамку). Если две или более из этих выборок имеют единичное состояние (принцип мажоритарного голосования), то старт-бит отклоняется как ложный, а приемник продолжит поиск следующего перехода из «1» в «0». Однако, если определен действительный старт-бит, то логика обнаружения синхронизации оказывается

синхронизированной, после чего вступит в силу логика обнаружения данных. Процесс синхронизации повторяется для каждого старт-бита.

Асинхронный поиск данных

После обнаружения старт-бита начинает работу логика обнаружения данных. Блок обнаружения данных использует цифровой автомат с 16 состояниями в нормальном режиме работы и с восемью состояниями в режиме удвоения скорости. На рисунке 3.72 показана выборка бита данных и бита четности. Для каждой выборки указан номер, который соответствует номеру состояния цифрового автомата блока обнаружения данных.

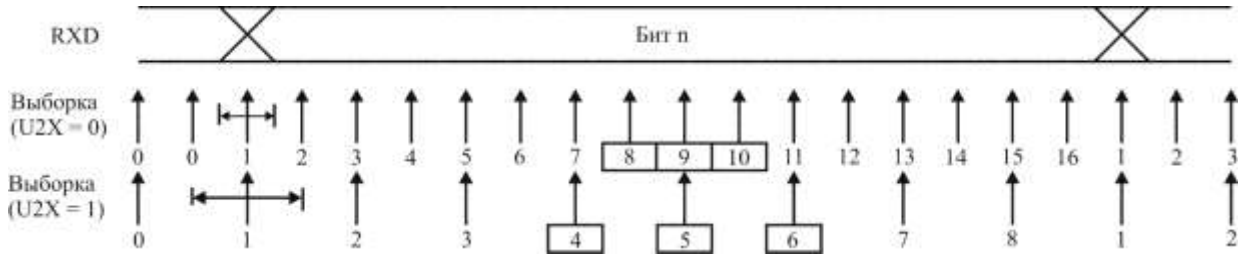


Рисунок 3.72 – Выборка бит данных и бита четности

Определение логического уровня принимаемого бита данных происходит с помощью мажоритарного голосования по трем выборкам, расположенным по центру принятого бита. Центральные выборки выделены на рисунке 3.72 путем размещения их в рамке. Процесс мажоритарного голосования состоит в следующем: если две или все три выборки имеют высокие уровни, то принятый бит фиксируется как логическая 1. Если две или три выборки имеют низкие уровни, то принятый бит фиксируется как логический 0. Процесс мажоритарного голосования, по сути, представляет собой фильтр низких частот для входящего сигнала с вывода RXD. Процесс обнаружения повторяется до полного завершения приема/посылки, в т. ч. первого стоп-бита. Следует обратить внимание, что приемник определяет только первый стоп-бит посылки, а второй игнорируется. На рисунке 3.73 отображен процесс выборки стоп-бита и начальный момент возможности обнаружения старт-бита следующей посылки.

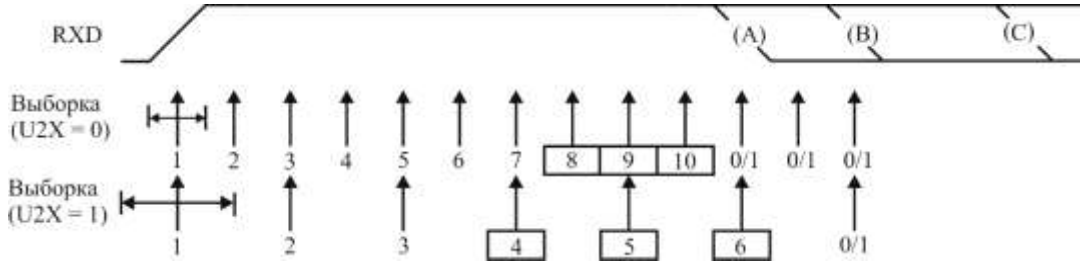


Рисунок 3.73 – Процесс выборки стоп-бита

Принцип мажоритарного голосования, рассмотренный на примере стоп-бита, аналогично распространяется и на другие биты в посылке. Если обнаруженный стоп-бит имеет нулевое значение, то устанавливается флаг ошибки посылки FE.

Новое изменение из «1» в «0» будет воспринято как стоп-бит новой посылки, если это изменение произошло после выборки последнего бита, используемого при мажоритарном голосовании. Для режима с нормальной скоростью первая выборка с низким уровнем может находиться в позиции, обозначенной A на рисунке 3.73. Для режима удвоения скорости появление низкого уровня допускается позже (точка B). Точка C соответствует полному завершению передачи стоп-бита. Использование раннего обнаружения стоп-бита влияет на рабочий диапазон приемника (допустимое расхождение частот при фиксированном формате посылки).

Рабочий диапазон асинхронной связи

Рабочий диапазон приемника зависит от расхождения между внутренне-генерируемой скоростью связи и скоростью принимаемых бит. Если передатчик отправляет посылки на более высокой или более низкой скорости или внутренне-генерируемая скорость связи приемника не соответствует основной частоте, то приемник окажется неспособным синхронизировать посылку по отношению к старт-биту.

Следующие выражения могут использоваться для вычисления отношения скорости принимаемых данных и внутренней скорости приемника.

$$R_{МИН} = \frac{(D+1)S}{S-1+D\cdot S+S_F}, R_{МАКС} = \frac{(D+2)S}{(D+1)\cdot S+S_M},$$

где D – сумма количества передаваемых бит данных и бит паритета, D = 5–10 бит;

S – количество выборок в секунду. S = 16/8 в режиме нормальной/удвоенной скорости;

S_F – номер первой выборки, используемой для мажоритарного голосования. S_F = 8/4 в режиме нормальной/удвоенной скорости;

S_M – номер центральной выборки. используемой при мажоритарном голосовании. S_M = 9/5 в режиме нормальной/удвоенной скорости;

$R_{мин}$ – отношение наименьшей скорости принимаемых данных к скорости приемника;

$R_{макс}$ – отношение наибольшей скорости принимаемых данных к скорости приемника.

В таблицах 3.61 и 3.62 приведен список максимально допустимых погрешностей при генерации скорости приемника. Следует обратить внимание, что режим нормальной скорости устойчив к более широким изменениям скорости связи.

Таблица 3.61 – Рекомендуемая максимальная погрешность генерации скорости связи приемника в режиме нормальной скорости (U2X = 0)

D (количество бит данных и паритета)	$R_{мин}$, %	$R_{макс}$, %	Общая максимальная 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 3.62 – Рекомендуемая максимальная погрешность генерации скорости связи приемника в режиме удвоенной скорости (U2X = 1)

D (количество бит данных и паритета)	$R_{мин}$, %	$R_{макс}$, %	Общая максимальная погрешность, %	Рекомендуемая максимальная погрешность приемника, %
5	94,12	105,66	+5,66/−5,88	± 3,0
6	94,92	104,92	+4,92/−5,08	± 2,5
7	95,52	104,35	+4,35/−4,48	± 2,0
8	96,00	103,90	+3,90/−4,00	± 2,0
9	96,39	103,53	+3,53/−3,61	± 1,5
10	96,70	103,23	+3,23/−3,30	± 1,5

Рекомендуемая погрешность генератора скорости связи приемника была выбрана исходя из того, что передатчик и приемник в совокупности определяют общую макси-

мальную погрешность. Имеется два возможных источника влияния на погрешность скорости связи приемника. Системная синхронизация (BQ) приемника всегда имеет некоторую нестабильность в зависимости от напряжения питания и температуры. При использовании кварцевого резонатора для генерации системной синхронизации как правило не возникает проблем, но при использовании керамических резонаторов частота синхронизации может изменяться более чем на 2 % в зависимости от характеристик выбранного резонатора. Второй источник влияния на погрешность является более управляемым.

Требуемую скорость связи не всегда удается получить путем деления частоты синхронизации на целое число. В этом случае необходимо выбрать такое значение UBRR, которое обеспечивает минимально возможную погрешность результирующей частоты.

3.15.15 Многопроцессорный режим связи

Установка бита многопроцессорного режима связи MPCM в регистре UCSRA активизирует функцию фильтрации входящих посылок приемником UART. Посылки, которые не содержат информации об адресе, игнорируются и не помещаются в приемный буфер. Это позволяет существенно уменьшить количество входящих посылок, подлежащих обработке ЦПУ в многопроцессорных системах, связь между процессорами в которых организована через одну последовательную шину. Значение бита MPCM не оказывает никакого влияния на работу передатчика, но при этом передатчик должен быть использован иначе, если используется режим многопроцессорной связи.

Если приемник настраивается на прием посылок с 5, 6, 7, 8 битами данных, то первый стоп-бит позволяет отличить назначение принятых данных: адрес или данные. Если приемник настроен на прием 9 бит данных, то значение девятого бита (RXB8) используется для идентификации адреса или данных. Если идентификатор типа посылки (первый стоп-бит или девятый бит данных) равен единице, то в посылке содержится адрес. В противном случае в посылке переданы данные.

Режим многопроцессорной связи позволяет нескольким подчиненным микроконтроллерам принимать данные от одного ведущего. При этом подчиненные микроконтроллеры по первой адресной посылке определяют к какому микроконтроллеру адресуется ведущий. Если один из подчиненных микроконтроллеров обнаруживает свой адрес, то следующие посылки данных он будет принимать в нормальном режиме, а остальные подчиненные микроконтроллеры эти данные игнорируют до тех пор, пока не будет обнаружена следующая адресная посылка.

3.15.16 Использование MPCM

Если микроконтроллер действует как ведущий, то он может использовать 9-битный формат данных в посылке (UCSZ = 7). Девятый бит данных (TXB8) устанавливается при передаче адресной посылки (TXB8 = 1) и сбрасывается при передаче посылки данных (TXB = 0). В этом случае подчиненные микроконтроллеры также должны устанавливать 9-битный формат.

Для обмена данными в многопроцессорном режиме связи необходимо использовать следующие процедуры:

1 Все подчиненные микроконтроллеры переводятся в многопроцессорный режим связи (MPCM =1 в UCSRA).

2 Ведущий МК отправляет адресную посылку, а все подчиненные принимают и считывают эту посылку. В подчиненных МК флаг RXC в регистре UCSRA устанавливается как обычно.

3 Каждый подчиненный МК считывает регистр UDR и определяет к кому адресуется ведущий МК. Адресуемый МК должен очистить бит MPCM в UCSRA, в противном случае он ожидает следующего адресного байта и сохраняет установки MPCM.

4 Адресуемый МК принимает все данные до следующей адресной посылки. Другие подчиненные МК, у которых бит MPCM остался установленным, будут игнорировать посылки данных.

5 После приема адресуемым МК последней посылки данных устанавливается бит MPCM и ожидается прием новой адресной посылки от ведущего МК. Далее процесс повторяется с пункта 2.

6 Использование 5-, 6-, 7-, 8-разрядных форматов данных возможно, но не удобно, т. к. приемник должен переключаться между n и (n + 1) форматами посылки. Это делает затруднительной полнодуплексную связь, т. к. передатчик и приемник используют общие установки формата. При использовании 5-, 6-, 7-, 8-разрядных данных в посылке передатчик должен использовать два стоп-бита, т. к. первый стоп-бит будет задействован для индикации типа посылки.

Не следует использовать инструкции «чтение-модификация-запись» (SBI и CBI) для установки или сброса бита MPCM. Бит MPCM находится в одной ячейке с флагом TXC, поэтому последний может быть случайно сброшен при выполнении инструкций SBI или CBI.

Описание регистров UART

Регистр данных UART–UDRn

Бит	7	6	5	4	3	2	1	0	
	RXBn(7–0)								UDRn чт.
	TXBn(7–0)								UDRn зап.
Чт./Зап.	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 UARTn расположены по одному и тому же адресу в области ввода-вывода, обозначенной как регистр данных UARTn или UDRn. Если выполнять запись по адресу регистра UDRn, то записываемые данные помещаются в буферный регистр данных передатчика TXBn. По аналогии, при чтении регистра UDRn извлекается содержимое буферного регистра данных приемника RXBn.

При использовании 5-, 6- или 7-битных форматов данных передатчик игнорирует, а приемник устанавливает нулевые значения неиспользуемых разрядов.

Запись в буфер передатчика можно выполнять, если установлен флаг UDREn в регистре UCSRAn. Данные, записанные в UDRn при сброшенном флаге UDREn, будут игнорированы передатчиком UARTn. Если выполнена запись в приемный буфер и при этом работа передатчика была разрешена, то после освобождения сдвигового регистра передатчик загрузит в него значение из буферного регистра. После этого выполняется передача данных на выводе TxDn. Приемный буфер организован как двухуровневый буфер FIFO (первый пришел – последний вышел). Буфер FIFO изменяет свое состояние, если выполнено чтение из приемного буфера. Вследствие такой организации буфера необходимо следить, чтобы по данному адресу не использовались инструкции «чтение-модификация-запись» (SBI и CBI). Также нужно быть внимательным при использовании инструкций тестирования бита (SBIC и SBIS), т. к. их выполнение может также изменить состояние буфера FIFO.

Регистр A управления и статуса UART–UCSRnA

Разряд	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
	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: RXCn – флаг завершения приема UART

Данный флаг устанавливается, если в приемном буфере содержатся несчитанные данные и сбрасывается, когда приемный буфер свободен (т. е. не содержит несчитанных данных). Если приемник отключается, то приемный буфер сбрасывается и, следовательно, флаг RXCn принимает нулевое значение. Флаг RXCn может использоваться для генерации прерывания по завершению приема (смотрите описание бита RXCIEn).

Разряд 6: TXCn – флаг завершения передачи UART

Данный флаг устанавливается, если вся посылка из сдвигового регистра передатчика полностью передана и в передающем буфере UDRn нет новых данных для передачи. Флаг TXCn автоматически сбрасывается при переходе на вектор прерывания по завершению передачи или сбрасывается программно путем записи логической 1 в позицию данного бита. Флаг TXCn может служить источником для генерации прерывания по завершению передачи (смотрите также описание бита TXCIEn).

Разряд 5: UDREn – флаг освобождения регистра данных UART

Флаг UDREn индицирует о готовности приемного буфера UDRn к приему новых данных. Если UDREn = 1, то буфер свободен и, следовательно, готов к записи. Флаг UDREn может служить источником для генерации прерывания по освобождению регистра данных (смотрите описание бит UDRIEn). UDREn устанавливается после сброса, индицируя о готовности передатчика.

Разряд 4: FEn – ошибка посылки

Данный бит устанавливается, если при приеме посылки, находящейся на выходе из приемного буфера, была определена ошибка в структуре посылки. Под ошибкой структуры в данном случае понимается нулевое значение первого стоп-бита в этой посылке. Значение данного бита действительно до чтения содержимого приемного буфера (UDRn). Флаг FEn принимает нулевое значение, если принятый стоп-бит имел правильное единичное значение. При записи в регистр UCSRnA в позиции данного бита необходимо указывать логический ноль.

Разряд 3: DORn – флаг переполнения данных

Данный бит устанавливается, если выявлено условие переполнения. Переполнение данных возникает, если заполнен приемный буфер (две посылки), новая посылка полностью принята в приемный сдвиговый регистр, а также обнаружен новый старт-бит. Значение данного бита действительно до чтения содержимого приемного буфера (UDRn). При записи в регистр UCSRnA в позиции данного бита необходимо указывать логический ноль.

Разряд 2: UPEn – ошибка паритета

Данный бит устанавливается, если следующая посылка в приемном буфере характеризуется ошибкой паритета, если во время приема этой посылки был разрешен контроль паритета (UPMn1 = 1). Данный бит имеет действительное значение до чтения приемного буфера (UDRn). При записи в регистр UCSRnA в позиции данного бита необходимо указывать логический ноль.

Разряд 1: U2Xn – удвоение скорости связи UART

Данный бит оказывает влияние только в асинхронном режиме связи. В синхронном режиме в данный бит необходимо записать логический 0. Запись в данный бит логической 1 уменьшает в два раза значение коэффициента деления скорости связи с 16 до 8, тем самым удваивая скорость передачи данных в асинхронном режиме.

Разряд 0: MPCMn – режим многопроцессорной связи

Данный бит разрешает режим многопроцессорной связи. Если в бит MPCMn записать логическую 1, то все входящие посылки, принимаемые приемником UART, будут игнорироваться, если они не содержат адресной информации. Установка бита MPCMn не влияет на работу передатчика.

Регистр B управления и статуса UARTn – UCSRnB

Разряд	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7: RXCIEn – разрешение прерывания по завершению приема

Запись в данный бит логическую 1 разрешает прерывание по флагу RXCn. Прерывание по завершению приема UARTn генерируется, если RXCIEn = 1, флаг общего разрешения прерываний I = 1 (в регистре SREG), а также установлен бит RXCn в регистре UCSRnA.

Разряд 6: TXCIE – разрешение прерывания по завершению передачи

Запись в данный бит логической 1 разрешает прерывание по флагу TXCn. Прерывание по завершению передачи UARTn генерируется, если TXCIEn = 1, флаг общего разрешения прерываний I = 1 (в регистре SREG), а также установлен бит TXCn в регистре UCSRnA.

Разряд 5: UDRIEn – разрешение прерывания по освобождению регистра данных UART

Установка данного флага разрешает прерывание по флагу UDREn. Прерывание по освобождению регистра данных генерируется, если бит UDRIEn = 1, флаг общего разрешения прерываний I = 1 (в регистре SREG) и установлен бит UDREn в регистре UCSRnA.

Разряд 4: RXENn – разрешение работы приемника

Запись в данный бит логической 1 приводит к разрешению работы приемника UARTn. При этом приемник формирует отключающий сигнал, который разрешает альтернативную функцию вывода RXDn. Отключение приемника приводит к сбросу приемного буфера, теряя при этом значения флагов FEn, DORn и UPEn.

Разряд 3: TXENn – разрешение работы передатчика

Запись в данный бит логической 1 разрешает работу передатчика UARTn. После этого передатчик генерирует отключающий сигнал, который активизирует альтернативную функцию вывода TXDn. Отключение передатчика (запись логического 0 в TXENn) вступит в силу только по завершении генерации посылки, т. е. когда освободятся и сдвиговый регистр и буфер передатчика. После отключения вывод TXDn возвращается к выполнению функции обычной линии ввода-вывода.

Разряд 2: UCSZn – формат данных

Бит UCSZn2 вместе с битами UCSZn(1–0) в регистре UCSRnC задаёт количество бит данных в посылке как для приемника, так и для передатчика.

Разряд 1: RXB8n – значение восьмого разряда принятых данных

RXB8n содержит значение девятого бита принятой посылки с 9-битным форматом. Данный бит необходимо считать прежде, чем будут считаны младшие 8 бит из регистра UDRn.

Разряд 0: TXB8n – восьмой разряд передаваемых данных

TXB8n содержит значение девятого бита данных для передачи посылки с 9-битным форматом. Данный бит необходимо записать перед тем, как будут записаны младшие разряды данных в UDRn.

Регистр C управления и статуса UART – UCSRnC

Разряд	7	6	5	4	3	2	1	0	
	URSEL	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7 – резервный бит

Этот бит выбирает доступ к регистрам UCSRC и UBRRH. Он читается, как «1», когда читается UCSRC. Этот бит должен быть установлен, когда осуществляется запись в регистр UCSRC, сброшен, когда осуществляется запись в регистр UBRRH.

Разряд 6: UMSELn – выбор режима UART

Данный бит позволяет переключаться между синхронным и асинхронными режимами последовательной связи.

Таблица 3.63 – Установки бита UMSELn

UMSELn	Режим связи
0	Асинхронный
1	Синхронный

Разряды 5, 4: UPMn(1–0) – режим паритета

Данные биты разрешают и устанавливают тип генерируемого и контролируемого паритета. После разрешения паритета передатчик автоматически генерирует и передает бит паритета в каждой посылке. Приемник генерирует бит паритета для принятых данных и сравнивает его со значением принятого в этой посылке бита паритета, а по результату сравнения устанавливает флаг ошибки паритета UPEn в регистре UCSRnA.

Таблица 3.64 – Установки бит UPMn

UPMn1	UPMn0	Режим паритета
0	0	Отключен
0	1	Резерв
1	0	Четность
1	1	Нечетность

Разряд 3: USBSn – выбор числа стоп-бит

Данный бит определяет сколько стоповых бит вставляет передатчик при генерации посылки. Приемник игнорирует эту настройку.

Таблица 3.65 – Установки бита USBSn

USBSn	Число стоп-бит
0	1 бит
1	2 бита

Разряды 2, 1: UCSZn(1–0) – формат данных

Биты UCSZn(1–0) вместе с UCSZn2 в регистре UCSRnB задают количество бит данных в посылке как для приемника, так и для передатчика.

Таблица 3.66 – Установки бит UCSZn

UCSZn2	UCSZn1	UCSZn0	Формат данных
0	0	0	5 бит
0	0	1	6 бит
0	1	0	7 бит
0	1	1	8 бит
1	0	0	Резерв
1	0	1	Резерв
1	1	0	Резерв
1	1	1	9 бит

Разряд 0: UCPOLn – полярность синхронизации

Данный бит используется только в синхронном режиме. Если используется асинхронный режим, то в данный бит необходимо записать логический 0. В синхронном режиме бит UCPOLn определяет соотношение между выборкой входящих данных и обновлением передаваемых данных и сигналом тактирования синхронной связи (XCKn).

Таблица 3.67 – Установки бит UCPOLn

UCPOLn	Изменение передаваемых данных на выходе TXDn	Выборка принимаемых данных на входе RXDn
0	Нарастающий фронт XCKn	Спадающий фронт XCKn
1	Спадающий фронт XCKn	Нарастающий фронт XCKрз

Регистры скорости связи UART – UBRRnL и UBRRnH

Разряд	7	6	5	4	3	2	1	0	
	-	-	-	-	UBRRn(11–8)				UBRRnH
	UBRRn(7–0)								UBRRnL
	Чт.	Чт.	Чт.	Чт.	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	
Примечание – Регистр UBRRnH недоступен в режиме совместимости с ATmega103.									

Разряды 15–12 – зарезервированные разряды

Данные разряды зарезервированы для будущего использования. Для совместимости с последующими разработками необходимо записать логический ноль в эти разряды во время записи в регистр UBRRnH.

Разряды 11–0: UBRRn(11–0) – регистр скорости связи UART

UBRR – 12-разрядный регистр, который задает значение скорости связи UART. Регистр UBRRnH содержит четыре старших разряда, а UBRRnL – восемь младших разрядов значения скорости UARTn. Если во время передачи или приема изменить скорость связи, то сеанс связи будет нарушен. Запись в регистр UBRRnL инициирует обновление предделителя скорости связи.

В таблицах 3.68 – 3.70 приведены примеры установок UBRR для генерации стандартных скоростей связи при типичных тактовых частотах микроконтроллера. Значения UBRR, которые дают результирующую скорость связи, отличаются не более, чем на 0,5 % от искомого значения. Более высокие погрешности также приемлемы, но приемник будет обладать меньшей помехоустойчивостью, особенно при передаче длинных посылок. Значения погрешностей вычислены по выражению

$$\delta = ((f_{ген} / 16(UBRR + 1)\, f_{связи}) - 1)100\ (\%),$$

где $f_{ген}$ – частота тактового генератора, Гц;
$f_{связи}$ – скорость связи, бит/с;
UBRR – значение регистра UBRR.

Таблица 3.68 – Примеры установок UBRR для типичных частот тактового генератора

Скорость связи, бит/с	$f_{ген}$ = 1,0 МГц				$f_{ген}$ = 1,8432 МГц				$f_{ген}$ = 2,0 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %
2400	25	0,2	51	0,2	47	0	95	0	51	0,2	103	0,2
4800	12	0,2	25	0,2	23	0	47	0	25	0,2	51	0,2
9600	6	-7,0	12	0,2	11	0	23	0	12	0,2	25	0,2
14,4К	3	8,5	8	-3,5	7	0	15	0	8	-3,5	16	2,1
19,2К	2	8,5	6	-7,0	5	0	11	0	6	-7,0	12	0,2
28,8К	1	8,5	3	8,5	3	0	7	0	3	8,5	8	-3,5
38,4К	1	-18,6	2	8,5	2	0	5	0	2	8,5	6	-7,0
57,6К	0	8,5	1	8,5	1	0	3	0	1	8,5	3	8,5
76,8К	-	-	1	-18,6	1	-25,0	2	0	1	-18,6	2	8,5
115,2К	-	-	0	8,5	0	0	1	0	0	8,5	1	8,5
230,4К	-	-	-	-	-	-	0	0	-	-	-	-
250К	-	-	-	-	-	-	-	-	-	-	0	0
Максимум[1)]	62,5 Кбит/с		125 Кбит/с		115,2 Кбит/с		230,4 Кбит/с		125 Кбит/с		250 Кбит/с	

[1)] UBRR = 0, погрешность = 0 %.

Таблица 3.69 – Примеры установок UBRR для типичных частот тактового генератора

Скорость связи, бит/с	$f_{ген}$ = 3,6864 МГц				$f_{ген}$ = 4,0 МГц				$f_{ген}$ = 7,3728 МГц			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %	UBRR	δ, %
1	2	3	4	5	6	7	8	9	10	11	12	13
2400	95	0	191	0	103	0,2	207	0,2	191	0	383	0
4800	47	0	95	0	51	0,2	103	0,2	95	0	191	0
9600	23	0	47	0	25	0,2	51	0,2	47	0	95	0
14,4К	15	0	31	0	16	2,1	34	-0,8	31	0	63	0
19,2К	11	0	23	0	12	0,2	25	0,2	23	0	47	0
28,8К	7	0	15	0	8	-3,5	16	2,1	15	0	31	0
38,4К	5	0	11	0	6	-7,0	12	0,2	11	0	23	0
57,6К	3	0	7	0	3	8,5	8	-3,5	7	0	15	0
76,8К	2	0	5	0	2	8,5	6	-7.0	5	0	11	0
115,2К	1	0	3	0	1	8,5	3	8.5	3	0	7	0
230,4К	0	0	1	0	0	8,5	1	8,5	1	0	3	0

Окончание таблицы 3.69

1	2	3	4	5	6	7	8	9	10	11	12	13
250К	0	-7,8	1	-7,8	0		1		1	-7,8	3	-7,8
0,5М	-	-	0	-7,8	-	0,0	0	0,0	0	-7,8	1	-7,8
1М	-	-	-	-	-	-	-	0,0	-	-	0	-7,8
						-		-				
Макси-мум[1)]	230,4К бит/с		460,8К бит/с		250К бит/с		0,5М бит/с		460,8К бит/с		921,6К бит/с	
[1)] UBRR = 0, погрешность = 0 %.												

Таблица 3.70 – Примеры установок UBRR для типичных частот тактового генератора

Скорость связи, бит/с	$f_{ген}$ = 8,0 МГц			
	U2X = 0		U2X = 1	
	UBRR	δ, %	UBRR	δ, %
2400	207	0,2	416	–0,1
4800	103	0,2	207	0,2
9600	51	0,2	103	0,2
14,4К	34	–0,8	68	0,6
19,2К	25	0,2	51	0,2
28,8К	16	2,1	34	–0,8
38,4К	12	0,2	25	0,2
57,6К	8	–3,5	16	2,1
76,8К	6	–7,0	12	0,2
115,2К	3	8,5	8	–3,5
230,4К	1	8,5	3	8,5
250К	1	0	3	0
0,5М	0	0	1	0
1М	–	–	0	0
Максимум[1)]	0,5 Мбит/с		1 Мбит/с	
[1)] UBRR = 0, погрешность = 0 %.				

3.16 Двухпроводной последовательный интерфейс TWI

Отличительные особенности:

- гибкий, простой, при этом эффективный последовательный коммуникационный интерфейс, требующий только две линии связи;
- поддержка как «ведущего», так и «ведомого» режима работы;
- возможность работы как приемника, так и как передатчика;
- 7-разрядное адресное пространство позволяет подключить к шине до 128 подчиненных устройств;
- поддержка многомастерного арбитрирования;
- скорость передачи данных до 400 КГц;
- выходы драйверов с ограниченной скоростью изменения сигналов;
- схема шумоподавления повышает стойкость к выбросам на линиях шины;
- программируемый адрес для подчиненного режима с поддержкой общего вызова;
- пробуждение микроконтроллера из режима сна при определении заданного адреса на шине.

3.16.1 Определение шины TWI

Двухпроводной последовательный интерфейс TWI идеально подходит для типичных применений микроконтроллера. Протокол TWI позволяет проектировщику системы внешне связать до 128 различных устройств через одну двухпроводную двунаправленную шину, где одна линия – линия синхронизации SCL, а вторая – линия данных SDA. В качестве внешних аппаратных компонентов, которые требуются для реализации шины, необходим только подтягивающий к плюсу питания резистор на каждой линии шины. Все устройства, которые подключены к шине, имеют свой индивидуальный адрес, а механизм определения содержимого шины поддерживается протоколом TWI (см. рисунок 3.74).

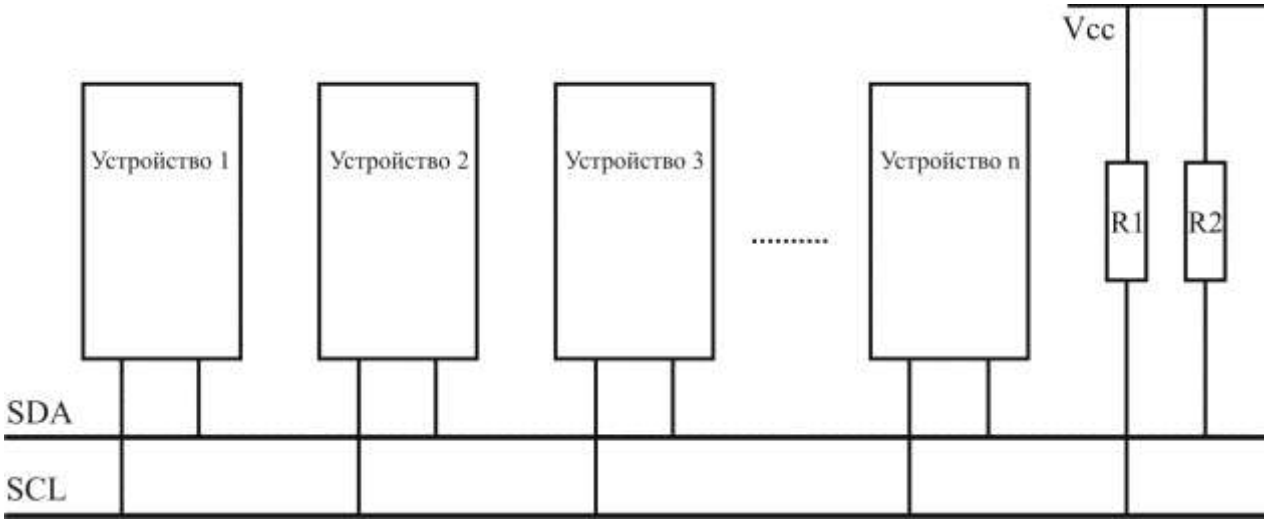


Рисунок 3.74 – Коммутация устройств на шинах SDA и SCL

3.16.2 Терминология TWI

Таблица 3.71 – Терминология TWI

Термин	Описание
Ведущий	Устройство, которое инициирует и прекращает сеанс связи. На стороне ведущего также генерируется сигнал синхронизации SCL
Ведомый	Устройство, которое адресуется ведущим устройством
Передатчик	Устройство, размещающее данные на шине
Приемник	Устройство, считывающее данные с шины

Как показано на рисунке 3.74, обе линии шины подключены к положительной шине питания через подтягивающие резисторы. Среди всех совместимых с TWI устройств в качестве драйверов шины используются транзистор или с открытым стоком, или с открытым коллектором. Этим реализована функция монтажного И, которая очень важна для двунаправленной работы интерфейса. НИЗКИЙ логический уровень на линии шины TWI генерируется, если одно или более из TWI-устройств выводит логический ноль. ВЫСОКИЙ уровень на линии присутствует, если все TWI-устройства перешли в третье высокоимпедансное состояние, позволяя подтягивающим резисторам задать уровень логической единицы. Следует обратить внимание, что при подключении к шине TWI нескольких AVR-микроконтроллеров для работы шины должны быть запитаны все из этих микроконтроллеров.

Количество устройств, которое может быть подключено к одной шине, ограничивается предельно допустимой емкостью шины (400 пФ) и 7-разрядным адресным пространством. Поддерживаются два различных набора технических требований, где один набор для шин со скоростью передачи данных ниже 100 кГц и один действителен для скоростей свыше 400 кГц.

3.16.3 Формат посылки и передаваемых данных

Передаваемые биты

Каждый передаваемый бит данных по шине TWI сопровождается импульсом на линии синхронизации. Уровень данных должен быть стабильным, когда на линии синхронизации присутствует логическая 1. Исключением для этого правила является генерация условий СТАРТа и СТОПа сеанса связи.

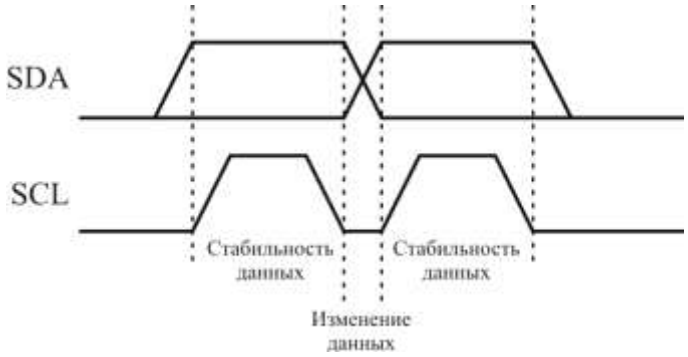


Рисунок 3.75 – Действительность данных

Условия СТАРТа и СТОПа

Ведущее устройство инициирует и заканчивает передачу данных. Передача инициируется, когда ведущий формирует условие СТАРТа на шине, и прекращается, когда ведущий формирует на шине условие СТОПа. Между условиями СТАРТа и СТОПа шина считается занятой и в этом случае никакой другой ведущий не может осуществлять управляющие воздействия на шине. Существуют особые случаи, когда новое условие СТАРТа возникает между условиями СТАРТа и СТОПа. Данный случай именуется как условие ПОВТОРНОГО СТАРТА и используется при необходимости инициировать ведущим новый сеанс связи, не теряя при этом управление шиной. После ПОВТОРНОГО СТАРТА шина считается занятой до следующего СТОПа. Это идентично поведению после СТАРТа, следовательно, при описании ссылка на условие СТАРТа распространяется и на ПОВТОРНЫЙ СТАРТ, если, конечно же, нет специального примечания. Как показано ниже, условия СТАРТа и СТОПа являются изменением логического уровня на линии SDA, когда на линии SCL присутствует логическая 1.

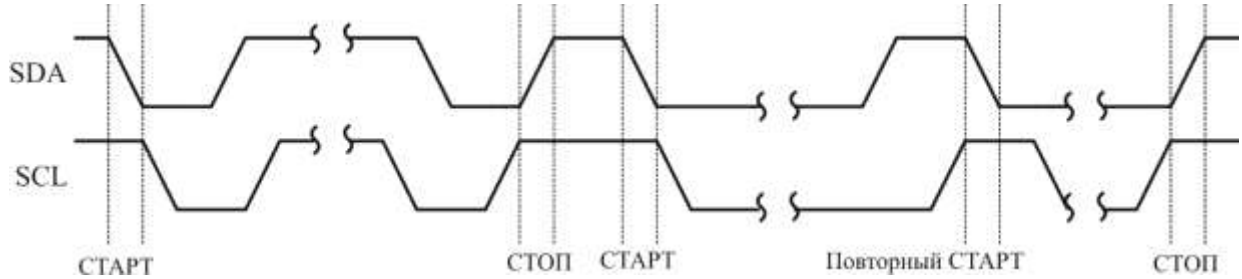


Рисунок 3.76 – Условия СТАРТа, ПОВТОРНОГО СТАРТА и СТОПа

3.16.4 Формат адресного пакета

Все передаваемые адресные пакеты по шине TWI состоят из 9 бит, в т. ч. 7 бит адреса, один бит управления для задания типа операции ЧТЕНИЕ/ЗАПИСЬ и один бит подтверждения. Если бит ЧТЕНИЕ/ЗАПИСЬ = 1, то будет выполнена операция чтения, иначе – запись. Если ведомый распознает, что к нему происходит адресация, то он должен сформировать низкий уровень на линии SDA на девятом цикле SCL (формирование бита подтверждения). 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 цикла подтверждения. Ведущий после этого может передать условие СТОПа или ПОВТОРНОГО СТАРТА для инициации новой передачи. Адресный пакет, состоящий из адреса ведомого устройства и бита ЧТЕНИЕ или ЗАПИСЬ, обозначим как ВЕДОМЫЙ_АДР+ЧТЕНИЕ (ПОДЧИН_АДР+ЧТЕНИЕ) или ВЕДОМЫЙ_ _АДР+ЗАПИСЬ (ПОДЧИН_АДР+ЗАПИСЬ), соответственно.

Старший разряд адресного байта передается первым. Нет никаких ограничений на выбор адреса ведомого устройства, за исключением адреса 0000 000, который зарезервирован для общего вызова.

При определении общего вызова все ведомые устройства должны ответить низким уровнем на линии SDA во время цикла подтверждения (ACK). Общий вызов необходимо использовать, если одно и тоже сообщение необходимо передать от ведущего к нескольким ведомым устройствам. Если вслед за битом ЗАПИСИ передан адрес общего вызова, то все ведомые устройства устанавливают низкий уровень на линии SDA для подтверждения общего вызова во время цикла подтверждения. Следующие пакеты данных будут приниматься всеми ведомыми устройствами, которые подтвердили общий вызов. Следует обратить внимание, что передача адреса общего вызова вслед за битом ЧТЕНИЕ бессмысленна, т. к. одновременное чтение нескольких подчиненных устройств одним ведущим невозможно (смотри рисунок 3.77).

Все адреса с форматом 1111xx необходимо зарезервировать для будущего использования.

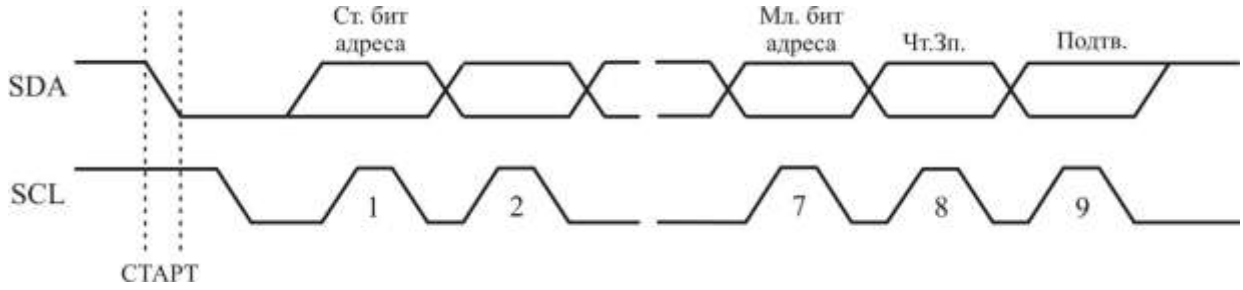


Рисунок 3.77 – Формат адресного пакета

3.16.5 Формат пакета данных

Все пакеты данных, передаваемые по шине TWI, состоят из 9 бит, в т. ч. 1 байт данных и бит подтверждения. Во время передачи данных ведущее устройство генерирует синхронизацию, а также условия СТАРТа и СТОПа, при этом на приемник возлагается подтверждение приема. Подтверждение (ПОДТВ) сигнализируется приемником выводом низкого уровня на линию SDA во время девятого такта сигнала SCL. Если приемник оставляет линию SDA в высоком состоянии, то это сигнализирует о том, что подтверждения не было (НЕТ ПОДТВ). После получения приемником последнего байта или если по каким-либо причинам нет возможности далее принимать данные, он должен информировать передатчик отправкой бита НЕТ ПОДТВ (нет подтверждения) после последнего байта. Старший бит данных передается первым (смотри рисунок 3.78).

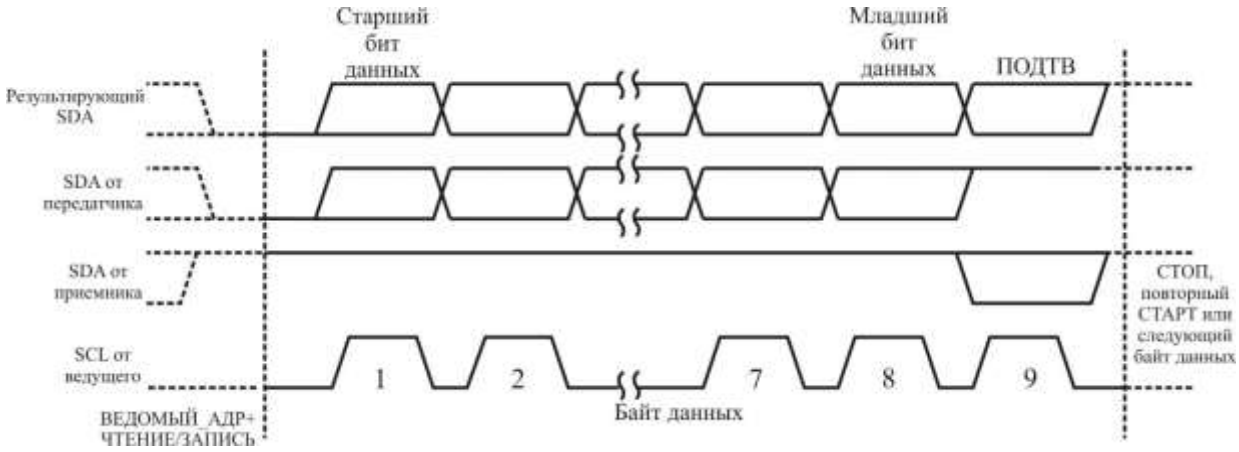


Рисунок 3.78 – Формат пакета данных

3.16.6 Сочетание пакетов адреса и данных во время сеанса связи

Сеанс связи обычно состоит из условия СТАРТа, ВЕДОМЫЙ_ _АДР+ЧТЕНИЕ/ЗАПИСЬ, одного или более пакетов данных и условия СТОПа. Передача пустого сообщения, которое состоит из условия СТАРТа, переданного вслед за условием СТОПа, является недопустимым. Следует обратить внимание, что монтажное «И» на линии SCL может использоваться для реализации подтверждения связи между ведущим и ведомым. Ведомый может продлить низкое состояние на линии SCL путем установки логического 0 на выводе SCL. Данный способ полезно использовать, если установленная скорость связи ведущим является повышенной по отношению к ведомому или если ведомому требуется дополнительное время на обработку между приемами данных. Ведомый, продлевающий низкое состояние на линии SCL, не будет оказывать влияние на длительность высокого состояния SCL, которая определяется ведущим. Как следствие, ведомый может снизить скорость передачи данных, продлевая рабочий цикл SCL.

На рисунке 3.79 показана типичная передача данных. Следует обратить внимание, что несколько байт данных могут быть переданы между условиями ВЕДОМЫЙ_ _АДР+ЧТЕНИЕ/ЗАПИСЬ и СТОП в зависимости от программного протокола, реализованного в прикладной программе.

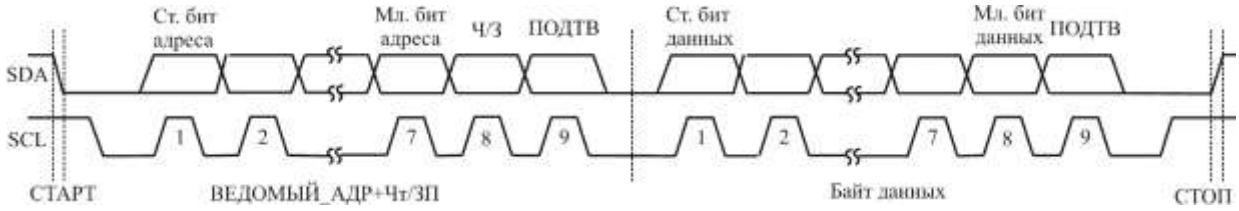


Рисунок 3.79 – Типичная передача данных

3.16.7 Системы многомастерных шин, арбитраж и синхронизация

Протокол TWI допускает размещение нескольких ведущих устройств на одной шине. Чтобы гарантировать нормальный процесс передачи, даже если два и более ведущих устройств инициируют передачу в одно и то же время, следует предпринять меры. Две проблемы, которые необходимо решить в многомастерных системах.

Алгоритм необходимо реализовать так, чтобы только одно ведущее устройство могло завершить передачу. Все остальные ведущие устройства должны прекратить передачу после обнаружения потери приоритета. Данный процесс выбора носит название арбитража. Если ведущее устройство обнаруживает потерю приоритета, то он должен сразу перейти в

«ведомый» режим, чтобы проверить не к нему ли обращается ведущее устройство, которое выиграло арбитраж. Факт начала одновременной передачи несколькими ведущими устройствами не должен обнаружиться ведомыми, т. е. передаваемые данные должны быть повреждены.

Разные ведущие устройства могут использовать разные частоты сигнала SCL. Необходимо придумать схему, которая позволяла бы синхронизировать тактовые сигналы всех ведущих устройств.

Использование принципа монтажного «И» позволяет решить обе эти проблемы. Соединение тактовых сигналов всех ведущих устройств по схеме монтажного «И» означает, что длительность результирующего единичного импульса будет равна длительности самого короткого единичного импульса в этом соединении. Длительность низкого уровня результирующего тактового сигнала равна длительности низкого уровня тактового сигнала того ведущего устройства, у которого она максимальная. Следует обратить внимание, что все ведущие устройства, подключенные к линии SCL, начинают счет времени окончания периодов с высоким и низким состояниями, когда результирующий сигнал SCL переходит в высокое или низкое состояние, соответственно (смотри рисунок 3.80).

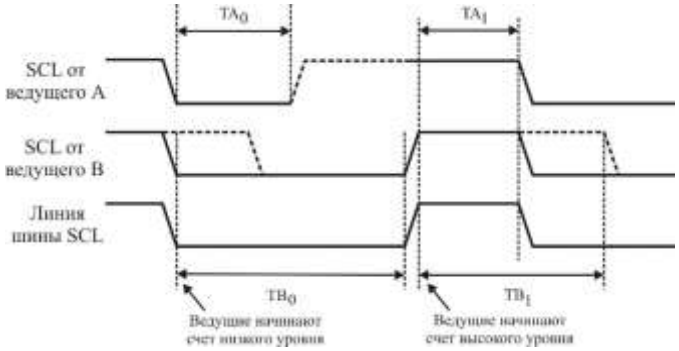


Рисунок 3.80 – Синхронизация на линии SCL между несколькими ведущими устройствами

Арбитраж реализован путем контроля состояния линии SDA всеми ведущими, выводящими данные. Если уровень, присутствующий на линии SDA не совпадает со значением, который передал ведущий, то данный ведущий теряет арбитраж. Арбитраж теряется только в том случае, если ведущий передал логическую 1, а фактически на линии SDA присутствовал логический 0. После потери ведущим приоритета, он незамедлительно переходит в «ведомый» режим для определения: не к нему ли адресуется выигравший арбитраж ведущий? Ведущие, которые проиграли процесс арбитража, могут продолжать генерировать тактовый сигнал до окончания передачи текущего пакета адреса или данных, но при этом они должны выводить высокий уровень на линию SDA. Арбитраж выполняется то тех пор, пока останется активным только один ведущий и для этого в ряде случаев может быть передано много бит. Если несколько ведущих пытаются адресоваться к одному и тому же ведомому, то процесс арбитража переносится на пакет данных (смотри рисунок 3.81).

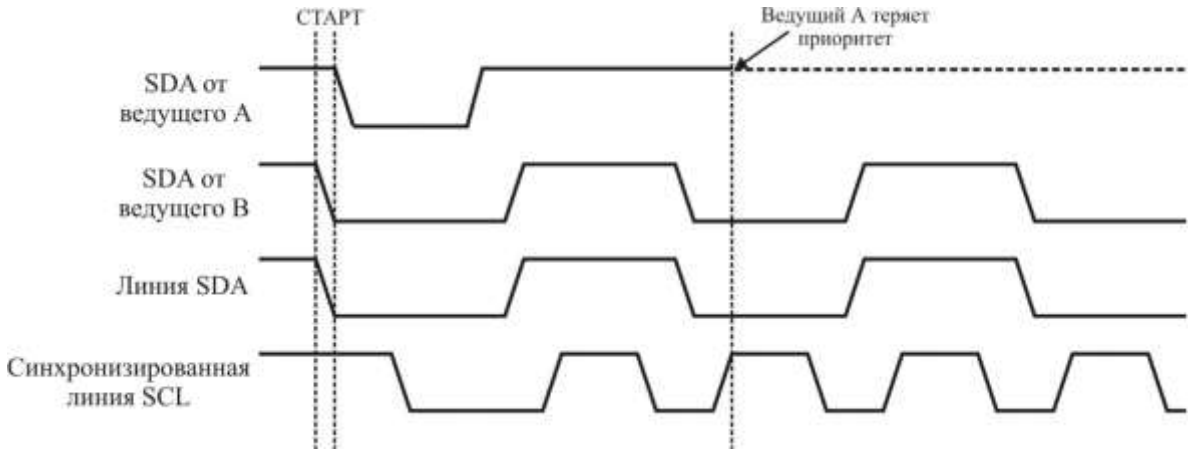


Рисунок 3.81 – Арбитраж двух ведущих

Арбитраж не выполняется между передачами: условия ПОВТОРНЫЙ СТАРТ и бита данных; условия СТОП и бита данных; условия ПОВТОРНЫЙ СТАРТ и СТОП.

Гарантирование невозможности возникновения данных условий возлагается на программное обеспечение пользователя. Этим подразумевается, что во многомастерных системах должно использоваться одинаковое сочетание пакетов данных и ВЕДОМЫЙ_ _АДР+ЧТЕНИЕ/ЗАПИСЬ. Или иначе: любой сеанс связи должен состоять из одинакового числа пакетов данных, в противном случае результат арбитража будет неопределенным.

3.16.8 Обзор модуля TWI

Модуль TWI состоит из нескольких подмодулей (см. рисунок 3.82). Все регистры, выделенные жирной линией на рисунке, доступны через шину данных микроконтроллера.

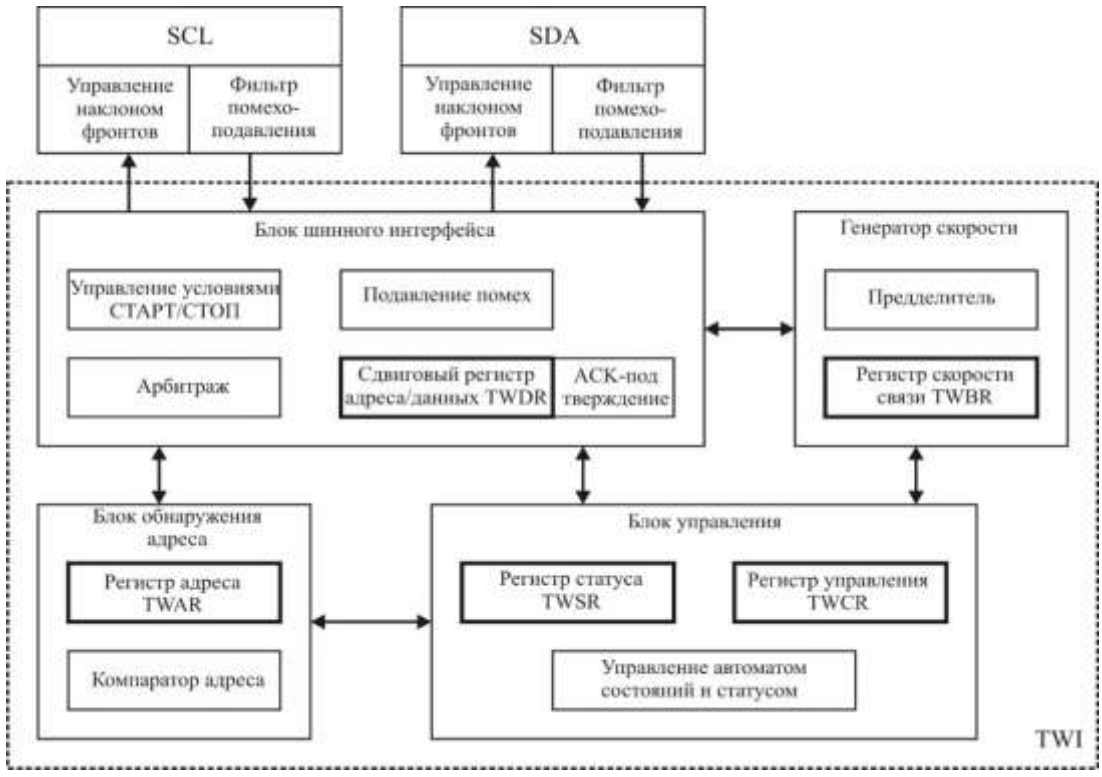


Рисунок 3.82 – Функциональная схема модуля TWI

Выводы SCL и SDA

Данные выводы связывают двухпроводной интерфейс микроконтроллера с остальными микроконтроллерами в системе. Драйверы выходов содержат ограничитель скорости изменения фронтов для выполнения требований к TWI. Входные каскады содержат блок подавления помех, задача которого состоит в игнорировании импульсов длительностью менее 50 нс. К каждой из этих линий можно подключить внутренний подтягивающий резистор путем установки разрядов PORTD.0 (SCL), PORTD.1 (SDA). Использование встроенных подтягивающих резисторов в ряде случаев позволяет отказаться от применения внешних.

Блок генератора скорости связи

Данный блок управляет периодом импульсов SCL в режиме ведущего устройства. Период SCL задается регистром скорости TWI (TWBR) и значением бит управления предделителем в регистре статуса TWI (TWSR). В «ведомом» режиме значения скорости или установки предделителя не оказывают влияния на работу, но частота синхронизации ЦПУ ведомого устройства должна быть минимум в 16 раз выше частоты SCL. Следует обратить внимание, что подчиненные могут продлевать длительность низкого уровня на линии SCL, тем самым уменьшая среднюю частоту синхронизации шины TWI. Частота SCL генерируется в соответствии с выражением

$$f_{SCL} = f_{ЦПУ} / (16 + 2\ (TWBR)\ 4^{TWPS}),$$

где *TWBR* – значение регистра скорости TWI;
TWPS – значение бит предделителя в регистре статуса TWI.

Примечание – значение TWBR должено быть не менее 10, если TWI работает в «ведущем» режиме. Если значение TWBR меньше 10, то ведущий может генерировать некорректное состояние на линиях SDA и SCL. Проблема возникает при работе в «ведущем» режиме при передаче условий СТАРТ+ВЕДОМЫЙ_АДР+ ЧТЕНИЕ/ЗАПИСЬ ведомому.

Блок шинного интерфейса

Данный блок содержит сдвиговый регистр адреса и данных (TWDR), контроллер СТАРТа/СТОПа и схему арбитража. TWDR содержит передаваемый байт адреса или данных или принятый байт адреса или данных. Помимо 8-разрядного регистра TWDR в состав блока шинного интерфейса также входит регистр, хранящий значение передаваемого или принятого бита НЕТ ПОДТВ. К данному регистру нет прямого доступа со стороны программного обеспечения. Однако во время приема он может устанавливаться или сбрасываться путем манипуляций с регистром управления TWI (TWCR). В режиме передатчика значение принятого бита НЕТ ПОДТВ можно определить по значению регистра TWSR.

Контроллер СТАРТа/СТОПа отвечает за генерацию и детектирование условий СТАРТ, ПОВТОРНЫЙ СТАРТ и СТОП. Контроллер СТАРТа/СТОПа позволяет обнаружить условия СТАРТ и СТОП, даже если микроконтроллер находится в одном из режимов сна. Этим обеспечивается возможность пробуждения микроконтроллера по запросу ведущего шины.

Если TWI инициировал передачу в качестве ведущего, то схема арбитража непрерывно контролирует передачу, определяя возможность дальнейшей передачи. Если TWI теряет приоритет, то блок формирует соответствующий сигнал блоку управления, который выполняет адекватные действия и генерирует соответствующий код состояния.

Блок обнаружения адреса

Блок обнаружения адреса проверяет равен ли принятый адрес значению 7-разрядного адреса из регистра TWAR. Если установлен бит разрешения обнаружения общего вызова TWGCE в регистре TWAR, то все входящие адресные биты будут дополнительно сравниваться с адресом общего вызова. При адресном совпадении подается сиг-

нал блоку управления, что позволяет выполнить ему необходимые действия. В зависимости от установки регистра TWCR подтверждение адреса TWI может происходить, а может и нет. Блок обнаружения адреса способен функционировать даже когда микроконтроллер переведен в режим сна, тем самым позволяя возобновить нормальную работу микроконтроллера по запросу ведущего шины. Если при адресном совпадении TWI в экономичном режиме микроконтроллера, т. е. когда инициируется возобновление работы микроконтроллера, возникает другое прерывание (например, INT0), то TWI прекращает работу и возвращается к состоянию холостого хода. Если возникновение данного эффекта нежелательно, то необходимо следить, чтобы во время обнаружения адресования, когда микроконтроллер находится в режиме «Выключен» (Power-down), было разрешено только одно прерывание.

Блок управления

Блок управления наблюдает за шиной TWI и генерирует отклики в соответствии с установками регистра управления TWI (TWCR). Если на шине TWI возникает событие, которое требует внимания со стороны программы, то устанавливается флаг прерывания TWINT. Следующим тактом обновляется содержимое регистра статуса TWI – TWSR, в котором будет записан код, идентифицирующий возникшее событие. Даная информация хранится в TWSR только тогда, когда установлен флаг прерывания TWI. Остальное время в регистре TWSR содержится специальный код статуса, который информирует о том, что нет информации о состоянии TWI. До тех пор пока установлен флаг TWINT, линия SCL остается в низком состоянии. Этим обеспечивается возможность завершить программе все задачи перед продолжением сеанса связи.

Флаг TWINT устанавливается в следующих ситуациях:

- после передачи условия СТАРТ/ПОВТОРНЫЙ_СТАРТ;
- после передачи ВЕДОМЫЙ_АДР+ЧТЕНИЕ/ЗАПИСЬ;
- после передачи адресного байта;
- после потери приоритета;
- после того как TWI адресован собственным адресом или общим вызовом;
- после приема байта данных;
- после приема условия СТОП или ПОВТОРНЫЙ_СТАРТ в режиме подчиненной адресации;
- после возникновения ошибки по причине некорректного условия СТАРТ или СТОП.

3.16.9 Описание регистров TWI

Регистр скорости связи шины TWI – TWBR

Разряд	7	6	5	4	3	2	1	0	
	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7–0 – биты регистра скорости связи шины TWI

TWBR задает коэффициент деления частоты генератора скорости связи. Генератор частоты скорости связи – делитель частоты, который формирует сигнал синхронизации SCL в режиме «ведущий». В описании «Блок генератора скорости связи» приведена методика вычисления скоростей связи.

Регистр управления шиной TWI – TWCR

Разряд	7	6	5	4	3	2	1	0	
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	TWCR
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Регистр TWCR предназначен для управления работой TWI. Он используется для разрешения работы TWI, для инициации сеанса связи ведущего путем генерации условия СТАРТ на шине, для генерации подтверждения приема, для генерации условия СТОП и для СТОПа шины во время записи в регистр TWDR. Он также сигнализирует о попытке ошибочной записи в регистр TWDR, когда доступ к нему был запрещен.

Разряд 7: TWINT – флаг прерывания TWI

Данный бит устанавливается аппаратно, если TWI завершает текущее задание и ожидает реакции программы. Если бит «I» в SREG и бит TWIE в TWCR установлены, то микроконтроллер переходит на вектор прерывания TWI. Линия SCL остается в низком состоянии, пока установлен флаг TWINT. Флаг TWINT сбрасывается программно путем записи в него логической 1. Следует обратить внимание, что данный флаг сбрасывается не автоматически при переходе на вектор прерывания. Также нужно учесть, что очистка данного флага приводит к возобновлению работы TWI. Из этого следует, что программный сброс данного флага необходимо выполнить после завершения опроса регистров TWAR, TWSR и TWDR.

Разряд 6: TWEA – бит разрешения подтверждения

Бит TWEA управляет генерацией импульса подтверждения. Если в бит TWEA записана логическая 1, то импульс ПОДТВ генерируется на шине TWI, если выполняется одно из следующих условий:

- принят собственный адрес;
- принят общий вызов, когда установлен бит TWGCE в регистре TWAR;
- принят байт данных в режиме ведущего приемника или ведомого приемника;
- запись логического 0 в бит TWEA позволяет временно отключиться от двухпроводной последовательной шины. Для возобновления распознавания адреса необходимо записать в данный бит логическую 1.

Разряд 5: TWSTA – бит условия СТАРТ

Программист должен установить данный бит при необходимости установки режима «ведущий» на двухпроводной последовательной шине. TWI аппаратно проверяет доступность шины и генерирует условие СТАРТ, если шина свободна. Однако если шина занята, то TWI ожидает появления условия СТОП, а затем генерирует новое условие СТАРТ для перехвата состояния ведущего шины. TWSTA необходимо сбрасывать программно после передачи условия СТАРТ.

Разряд 4: TWSTO – бит условия СТОП

Установка бита TWSTO в режиме ведущего приводит к генерации условия СТОП на двухпроводной последовательной шине. Если на шине выполняется условие СТОП, то бит TWSTO сбрасывается автоматически. В подчиненном режиме установка бита TWSTO может использоваться для выхода из условия ошибки. В этом случае условие СТОП не генерируется, но интерфейс TWI возвращается к хорошо сконфигурированному безадресному подчиненному режиму и переводит линии SCL и SDA в высокоимпедансное состояние.

Разряд 3: TWWC – флаг ошибочной записи

Бит TWWC устанавливается при попытке записи в регистр данных TWDR, когда TWINT имеет низкий уровень. Флаг сбрасывается при записи регистра TWDR, когда TWINT = 1.

Разряд 2: TWEN – бит разрешения работы TWI

Бит TWEN разрешает работу TWI и активизирует интерфейс TWI. Если бит TWEN установлен, то TWI берет на себя функции управления линиями ввода-вывода SCL и SDA. При этом разрешается работа ограничителей скорости изменения фронтов и помехоподавляющих фильтров. Если данный бит равен нулю, то TWI отключается и все передачи прекращаются, независимо от состояния работы.

Разряд 1 – резервный бит

Данный бит является резервным и считывается как ноль.

Разряд 0: TWIE – разрешение прерывания TWI

Если в данный бит записана логическая 1 и установлен бит «I» в регистре SREG, то запрос на прерывание TWI будет генерироваться до тех пор, пока установлен флаг TWINT.

Регистр состояния TWI – TWSR

Разряд	7	6	5	4	3	2	1	0	
	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWS1	TWS0	TWSR
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Разряды 7–3: TWS – статус TWI

Данные 5 бит отражают статус логики блока TWI и двухпроводной последовательной шины. Различия в кодах состояния будут представлены далее в этом разделе. Следует обратить внимание, что считываемое значение из регистра TWSR содержит и 5-разрядный код состояния и 2-разрядное значение, управляющее предделителем. Программист должен маскировать биты предделителя во время проверки бит состояния. В этом случае проверка состояния не будет зависеть от настройки предделителя.

Разряд 2 – резервный бит

Данный бит является резервным и считывается как ноль.

Разряды 1, 0: TWPS – биты предделителя TWI

Данные биты отличаются полным доступом (чтение/запись) и позволяют управлять предделителем скорости связи (см. таблицу 3.72).

Таблица 3.72 – Предделитель скорости связи TWI

TWPS1	TWPS0	Значение предделителя
0	0	1
0	1	4
1	0	16
1	1	64

Формула для вычисления скорости связи представлена в 3.16.8 «Обзор модуля TWI» («Блок генератора скорости связи»). Значение бит TWPS1, TWPS 0 используется в ней.

Регистр данных шины TWI – TWDR

Разряд	7	6	5	4	3	2	1	0	
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Если TWI настроен на режим ведомого передатчика или приемника, то будет реагировать только на адрес, записанный в этот регистр (в семь старших разрядов TWAR). В остальных режимах ведущего данный регистр не используется. В многомастерных системах регистр TWAR устанавливается в том ведущем, к которому адресуются как к подчиненному другие ведущие шины.

Младший разряд регистра TWAR используется для разрешения обнаружения адреса общего вызова ($00). Специальный компаратор выполняет сравнение ведомого адреса (или адреса общего вызова) с принятым адресом. Если обнаруживается совпадение, то генерируется запрос на прерывание.

Разряды 7–1: TWA – регистр подчиненного адреса TWI

Данные семь бит составляют подчиненный адрес блока TWI.

Разряд 0: TWGCE – бит разрешения обнаружения общего вызова по шине TWI

После установки данного бита разрешается работа схемы обнаружения общего вызова, передаваемого по двухпроводной последовательной шине.

3.16.10 Рекомендации по использованию регистров TWI

TWI ориентирован на передачу данных в байтном формате с управлением по прерываниям. Прерывания возникают после обнаружения одного из событий на шине, например, прием байта или передача условия СТАРТ. Управление TWI по прерываниям позволяет освободить программное обеспечение на выполнение других задач во время передачи байта данных. Следует обратить внимание, что установка флага TWINT приводит к генерации запроса на прерывание только в том случае, когда установлен бит разрешения прерывания TWIE в регистре TWCR, а также разрешена работа прерываний установкой бита в регистре SREG. Если бит TWIE сброшен, то состояние TWINT должно отслеживаться программно для оценки ситуации на шине TWI.

После установки флага TWINT интерфейс TWI приостанавливает работу и ожидает реакции программы. В этом случае регистр статуса TWI (TWSR) содержит значение, которое индицирует текущее состояние шины TWI. Исходя из этого, программа задает дальнейшее поведение шины TWI, манипулируя регистрами TWCR и TWDR.

На рисунке 3.83 показан простой пример подключения к шине TWI. Здесь предполагается, что мастер желает передать один байт данных подчиненному. Данное описание весьма общее, а более подробно объяснение приводится далее в этом подразделе.

Первым шагом работы регистра TWI является передача условия СТАРТ. Это инициируется путем записи специфического значения в TWCR. О значении, которое необходимо записать, будет сказано позже. Однако необходимо следить, чтобы в записываемом в регистр значении был установлен бит TWINT. Запись логической 1 в TWINT сбрасывает этот флаг. TWI не начнет работу до тех пор, пока будет установлен флаг TWINT в регистре TWCR. Сразу после сброса TWINT начинается передача условия СТАРТ.

После передачи условия СТАРТ устанавливается флаг TWINT в регистре TWCR, а содержимое TWSR обновляется значением кода состояния, индицирующего об успешной передаче условия СТАРТ.

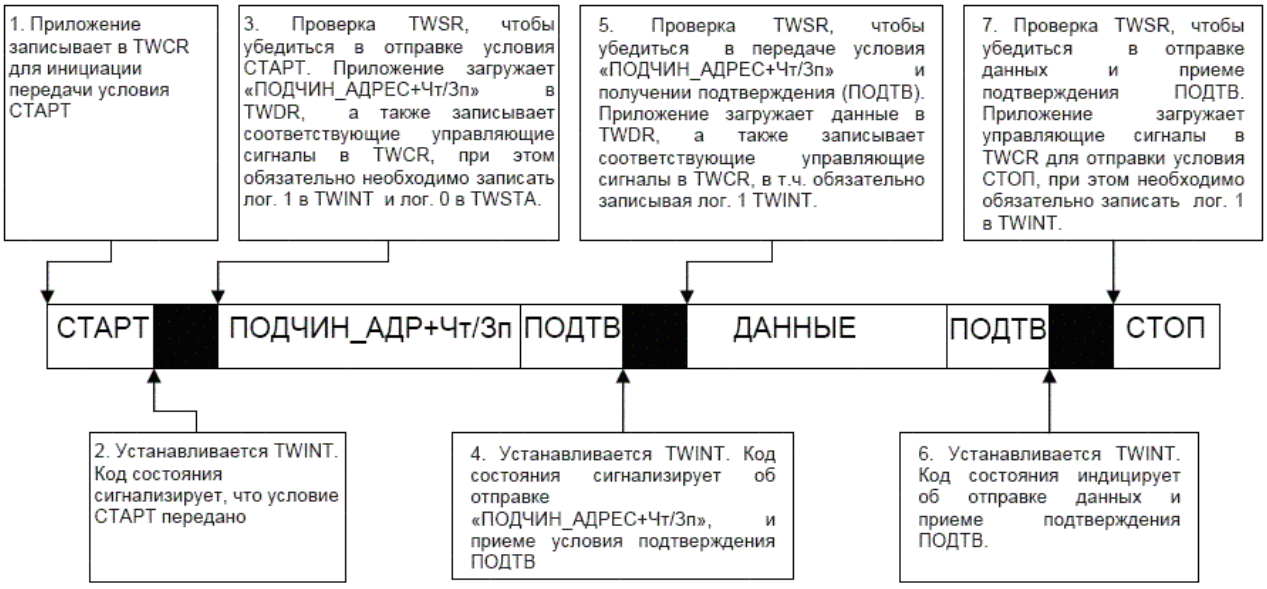


Рисунок 3.83 – Последовательность обслуживания TWI при типичной передаче

В программе необходимо выполнить проверку значения TWSR, чтобы убедиться в том, что условие СТАРТ было успешно передано. Если TWSR индицирует другую ситуацию, то программа выполняет особые действия, например, вызывает процедуру обработки ошибочных ситуаций. Если код состояния имеет ожидаемое значение, то выполняется загрузка условия ВЕДОМЫЙ_АДР + ЗАПИСЬ в TWDR. Необходимо помнить, что TWDR используется для хранения как адреса, так и данных. После загрузки в TWDR желаемого значения ВЕДОМЫЙ_АДР + ЗАПИСЬ в регистр TWCR должно быть записано специфическое значение, которое служит командой для передачи значения ВЕДОМЫЙ _АДР + ЗАПИСЬ, хранящегося в TWDR. Какое именно значение необходимо записать будет сказано позже. Однако необходимо следить, чтобы в записываемом в регистр значении был установлен бит TWINT. Запись логической 1 в TWINT приводит к сбросу этого флага. TWI не начнет работу до тех пор, пока установлен бит TWINT в регистре TWCR. Сразу после сброса флага TWINT инициируется передача адресного пакета.

После передачи адресного пакета устанавливается флаг TWINT в регистре TWCR, а содержимое регистра TWSR обновляется кодом состояния, индицирующего успешность передачи адресного пакета. В коде состояния также отражается было ли подтверждение приема адресного пакета со стороны подчиненного или нет.

Выполняется программная проверка значения TWSR, чтобы убедиться в успешности передачи адресного пакета и что бит подтверждения имеет ожидаемое значение. Если TWSR индицирует иную ситуацию, то при необходимости выполняются особые действия, например, вызывается процедура обработки ошибочных ситуаций. Если же код состояния имеет ожидаемое значение, то программа записывает пакет данных в TWDR. Впоследствии в регистр TWCR записывается специфическое значение, которое служит командой для TWI и вызывает аппаратную передачу данных, записанных в TWDR. Необходимо учесть, что в записываемом значении должен быть установлен бит TWINT. Запись логической 1 в TWINT приводит к сбросу этого флага. TWI не начнет работу до тех пор пока будет установлен бит TWINT в регистре TWCR. Сразу после сброса TWINT начинается передача пакета данных.

После передачи пакета данных устанавливается флаг TWINT в регистре TWCR, а содержимое регистра TWSR обновляется значением кода состояния, который сигнализирует об успешной передачи пакета данных. В коде состояния также отражается: было ли принято подтверждение от подчиненного или нет.

Выполняется программная проверка значения в TWSR, чтобы убедиться в успешности передачи пакета данных и в том, что бит ПОДТВ имеет ожидаемое значение. Если TWSR индицирует иную ситуацию, то программа выполняет особые действия, в т. ч. вы-

зывает процедуру обработки прерывания. Если код состояния имеет ожидаемое значение, то выполняется запись специального значения в TWCR, которое служит командой для TWI и инициирует передачу условия СТОП. Какое именно значение необходимо записать, сказано далее. Однако следует учесть, что во время записи должна быть произведена установка бита TWINT. Запись логической 1 в TWINT приводит к очистке этого флага. TWI не начнет работу до тех пор, пока установлен бит TWINT в регистре TWCR. Сразу после сброса флага TWINT инициируется передача условия СТОП. Следует обратить внимание, что флаг TWINT не устанавливается по завершении передачи условия СТОП.

Несмотря на простоту изложенного примера, он показывает принципы, положенные в основу любой передачи через TWI. Из вышеизложенного можно сделать следующие выводы:

- по завершении работы регистра TWI устанавливается флаг TWINT и далее ожидается реакция со стороны программы. Линия SCL находится в низком состоянии, пока сброшен флаг TWINT;

- если флаг TWINT установлен, то пользователь может обновлять любой из регистров TWI значением, которое относится к следующему этапу работы шины TWI. Например, в TWDR загружается значение, которое необходимо передать на следующем цикле шины;

- после обновления всех регистров TWI и завершения других задач выполняется запись в TWCR. Во время записи TWCR необходимо, чтобы был установлен бит TWINT. В этом случае запись логической 1 в TWINT приведет к сбросу данного флага. TWI выполняет действия в соответствии с установкой регистра TWCR.

Далее приводится пример на языке «Ассемблере» и «Си». В примере предполагается, что все символьные обозначения определены в присоединенном файле.

Таблица 3.73

Номер по порядку	Пример кода на языке «Ассемблер»	Пример кода на языке «Си»	Комментарий
1	2	3	4
1	ldi r16, (1<<TWINT) | (1<<TWSTA)	(1<<TWEN) out TWCR, r16	TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN)	Передача условия СТАРТ
2	wait1: in r16, TWCR sbrs r16, TWINT rjmp wait1	while (!(TWCR & (1<<TWINT)))	Ожидание установки флага TWINT. Этим индицируется завершение передачи условия СТАРТ
3	in r16,TWSR andi r16, 0xF8 cpi r16, START brne ERROR	if ((TWSR & 0xF8) ! = START) ERROR()	Проверка кода состояния TWI. Маскирующий бит предделителя. Если код состояния не равен СТАРТ, то переход на ERROR
	ldi r16, SLA_W out TWDR, r16 ldi r16, (1<<TWINT) | (1<<TWEN) out TWCR, r16	TWDR = SLA_W; TWCR = (1<<TWINT) | (1<<TWEN)	Загрузка ВЕДОМЫЙ_ _АДР + ЗАПИСЬ в регистр TWDR. Сброс бита TWINT в TWCR для начала передачи адреса

Окончание таблицы 3.73

1	2	3	4
4	wait2: in r16, TWCR sbrs r16, TWINT rjmp wait2	while (!(TWCR & (1<<TWINT)))	Ожидание установки флага TWINT. Этим сигнализируется завершение передачи ВЕДОМЫЙ_АДР + ЗАПИСЬ и получение/неполучение подтверждения (ПОДТВ/НЕТ ПОДТВ)
5	in r16, TWSR andi r16, 0xF8 cpi r16, MT_SLA_ACK brne ERROR	if ((TWSR & 0xF8) ! = MT SLA ACK) ERROR()	Проверка значения регистра статуса. Маскирование бит предделителя. Если состояние отличается от MT SLA ACK, то переход на ERROR
	ldi r16, DATA out TWDR, r16 ldi r16, (1<<TWINT) | (1<<TWEN) out TWCR, r16	TWDR = DATA; TWCR = (1<<TWINT) | (1<<TWEN)	Загрузка данных в TWDR. Сброс флага TWINT в TWCR для начала передачи данных
6	wait3: in r16, TWCR sbrs r16,TWINT rjmp wait3	while (!(TWCR & (1<<TWINT)))	Ожидание установки флага TWINT. Этим индицируется, что данные были переданы и принято/не принято подтверждение (ПОТДВ/НЕТ ПОДТВ).
7	in r16,TWSR andi r16, 0xF8 cpi r16, MT DATA ACK brne ERROR	if ((TWSR & 0xF8) ! = MT DATA ACK) ERROR()	Проверка значения регистра статуса TWI. Маскирование бит предделителя. Если состояние отличается от MT DATA ACK, то переход на ERROR
	ldi r16, (1<<TWINT) | (1<<TWEN) | (1<<TWSTO) out TWCR, r16	TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO)	Передача условия СТОП

Примечание – Если фактический регистр расположен в расширенной памяти ввода-вывода, то инструкции «IN», «OUT», «SBIS», «SBIC», «CBI» и «SBI» необходимо заменить теми инструкциями, которые эффективны для данной области памяти. Обычно это инструкции «LDS» и «STS» в сочетании с «SBRS», «SBRC», «SBR» и «CBR».

Режимы передачи

Регистр TWI может работать в одном из четырех режимов работы. 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 двухпроводное последовательное ЭСППЗУ, а режим MR для считывания данных из ЭСППЗУ. Если в системе имеются другие ведущие, один из которых передает данные, то у остальных используется режим SR. Какой из режимов должен использоваться определяется программно.

Далее описан каждый из этих режимов. Возможные значения кодов состояния представлены рядом с рисунками, детализирующими процесс передачи данных в каждом из режимов. На рисунках используются следующие аббревиатуры:

- СТАРТ: условие СТАРТ;
- ПОВТ. СТАРТ: условие ПОВТОРНЫЙ СТАРТ;
- ЧТЕНИЕ: бит «Чтение» (высокий уровень на SDA);
- ЗАПИСЬ: бит «Запись» (низкий уровень на SDA);
- ПОДТВ.: бит подтверждения (низкий уровень на SDA);

- НЕТ ПОДТВ.: нет бита подтверждения (высокий уровень на SDA);
- ДАННЫЕ: 8-разрядный байт данных;
- СТОП: условие СТОП;
- ВЕДОМЫЙ_АДР (ПОДЧИН_АДР): ведомый (подчиненный) адрес.

На рисунке 3.85 окружности используются для индикации установки флага TWINT. Число, записанное внутри окружности, является кодом состояния из регистра TWSR, в котором замаскированы биты предделителя. В данном состоянии ожидается действие со стороны программы для завершения передачи TWI. Передача TWI приостанавливается до тех пор, пока программно не будет сброшен флаг TWINT.

После установки флага TWINT по значению кода состояния из регистра TWSR определяется, какое действие выполнить программе. В таблицах 3.74 – 3.77 представлена информация о том, какие программные действия должны быть предприняты при различных значениях кода состояния. Следует обратить внимание, что в таблицах биты предделителя замаскированы нулевыми значениями.

Режим ведущего передатчика

В режиме ведущего передатчика (MT) байты данных передаются ведомому приемнику (SR) (смотри рисунок 3.84). Для ввода режима ведущего необходимо передать условие СТАРТ. Формат следующего адресного пакета определяет какой режим вводится: ведущий передатчик (MT) или ведущий приемник (MR). Если передается ВЕДОМЫЙ_ _АДР + ЗАПИСЬ, то вводится режим MT (ведущий передатчик), а если В_АДР + ЧТЕНИЕ, то вводится режим MR (ведущий приемник). Все упоминаемые в этом разделе коды состояния в позиции бит предделителя имеют нулевые значения.

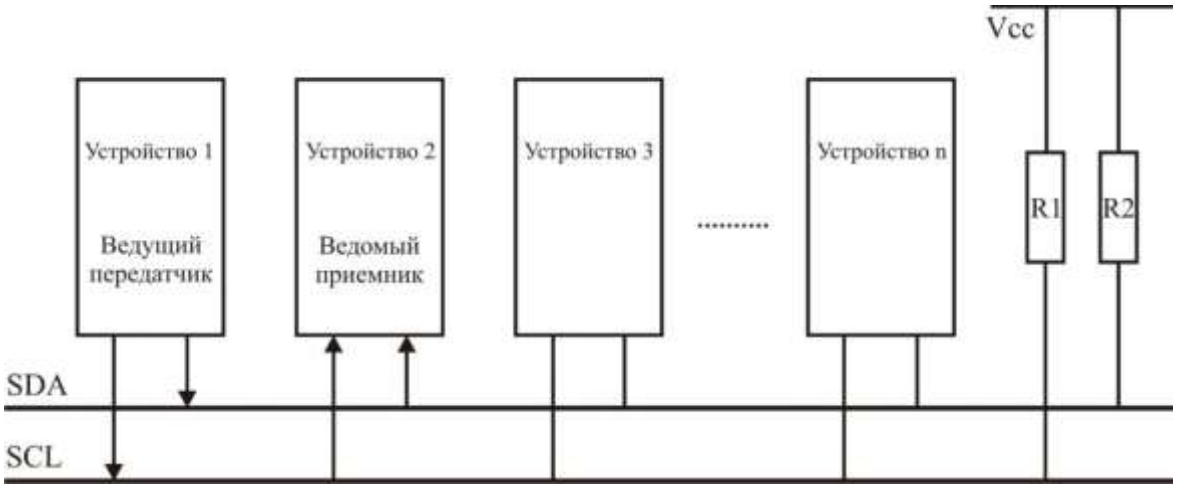


Рисунок 3.84 – Передача данных в режиме ведущего передатчика

Передача условия СТАРТ инициируется путем записи в TWCR следующего значения:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

Для разрешения работы двухпроводного последовательного интерфейса необходимо установить бит TWEN. Запись логической 1 в TWSTA инициирует передачу условия СТАРТ, а запись логической 1 в TWINT приводит к сбросу флага TWINT. После записи данного значения TWI тестирует двухпроводную последовательную шину и генерирует условие СТАРТ сразу после освобождения шины. После передачи условия СТАРТ аппаратно устанавливается флаг TWINT, а в регистр TWSR помещается код состояния $08 (см. таблицу 3.74). Для перевода в режим ведущего передатчика необходимо передать ВЕДО-

МЫЙ_АДР + ЗАПИСЬ. Это выполняется путем записи значения ВЕДОМЫЙ_АДР + ЗАПИСЬ в регистр TWDR. После этого необходимо сбросить флаг TWINT (путем записи в него логической 1) для продолжения сеанса связи. Данное выполняется путем записи следующего значения в TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	0	x	1	0	x

После передачи ВЕДОМЫЙ_АДР + ЗАПИСЬ и приема бита подтверждения флаг TWINT снова устанавливается, а в регистр TWSR помещается код состояния, который может иметь несколько значений. В режиме ведущего код состояния может быть $18, $20 или $38. Для каждого из этих кодов состояний необходимо выполнить адекватные действия, что отражено в таблице 3.74.

После успешной передачи ВЕДОМЫЙ_АДР + ЗАПИСЬ должен быть передан пакет данных. Его передача инициируется записью байта данных в TWDR. Доступ на запись в TWDR разрешен только тогда, когда флаг TWINT равен 1. В противном случае доступ блокируется и устанавливается флаг ошибочной записи TWWC в регистре TWCR. После обновления TWDR необходимо сбросить бит TWINT (путем записи в него логической 1) для продолжения сеанса связи. Данное можно выполнить путем записи следующего значения в регистр TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	0	x	1	0	x

Данная последовательность повторяется до тех пор, пока не будет передан последний байт. После этого генерируется условие СТОП или ПОВТОРНЫЙ СТАРТ. Условие СТОП генерируется путем записи следующего значения TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	1	x	1	0	x

Условие ПОВТОРНЫЙ СТАРТ генерируется путем записи следующего значения в TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

После передачи условия ПОВТОРНОГО СТАРТА (состояние $10) двухпроводной последовательный интерфейс может обращаться к тому же подчиненному устройству или же к новому, при этом не требуется передача условия СТОП. Таким образом, ПОВТОРНЫЙ СТАРТ полезно использовать для смены подчиненного устройства в режимах ведущий передатчик и ведущий приемник без потери управления шиной.

Таблица 3.74 – Коды состояния в режиме ведущего передатчика

Код состояния (TWSR), биты предделителя	Состояние двупроводной последовательной шины	Программные действия					Следующее действие, выполняемое микросхемой
		В/из TWDR	В TWCR				
			STA	STO	TWINT	TWEA	
$08	Передано условие СТАРТ	Загрузка ВЕДОМЫЙ_ _АДР + ЗАПИСЬ	0	0	1	x	Принимается ВЕДОМЫЙ_ _АДР + ЗАПИСЬ; Принимается ПОДТВ. или НЕТ ПОДТВ.
$10	Передано условие ПОВТОРНЫЙ СТАРТ	Загрузка ВЕДОМЫЙ_ _АДР + ЗАПИСЬ	0	0	1	x	Передается ВЕДОМЫЙ_ _АДР + ЗАПИСЬ; Принимается ПОДТВ или НЕТ ПОДТВ.
		или ВЕДОМЫЙ_ _АДР + ЧТЕНИЕ	0	0	1	x	Передается ВЕДОМЫЙ_ _АДР + ЧТЕНИЕ; Переход в режим ведущего приемника
$18	Передано ВЕДОМЫЙ_ _АДР + + ЗАПИСЬ и принято подтверждение	Загрузка байта данных или	0	0	1	x	Передается байт данных, принимается или не принимается подтверждение
		Действие без загрузки TWDR или	1	0	1	x	Передается ПОВТОРНЫЙ СТАРТ
		Действие без загрузки TWDR или	0	1	1	x	Передается условие СТОП и сбрасывается флаг TWSTO
		Действие без загрузки TWDR	1	1	1	x	Вслед за условием СТОП передается условие СТАРТ и сбрасывается флаг TWSTO
$20	Передано ВЕДОМЫЙ_ _АДР + + ЗАПИСЬ и принято НЕТ подтверждения	Загрузка байта данных или	0	0	1	x	Передается байт данных, принимается или не принимается подтверждение
		Действие без загрузки TWDR или	1	0	1	x	Передается ПОВТОРНЫЙ СТАРТ
		Действие без загрузки TWDR или	0	1	1	x	Передается условие СТОП и сбрасывается флаг TWSTO
		Действие без загрузки TWDR	1	1	1	x	Вслед за условием СТОП передается условие СТАРТ и сбрасывается флаг TWSTO
$28	Передается байт данных, принимается подтверждение	Загрузка байта данных или	0	0	1	x	Передается байт данных, принимается или не принимается подтверждение
		Действие без загрузки TWDR или	1	0	1	x	Передается ПОВТОРНЫЙ СТАРТ
		Действие без загрузки TWDR или	0	1	1	x	Передается условие СТОП и сбрасывается флаг TWSTO
		Действие без загрузки TWDR	1	1	1	x	Вслед за условием СТОП передается условие СТАРТ и сбрасывается флаг TWSTO
$30	Передается байт данных, принимается НЕТ подтверждения	Загрузка байта данных или	0	0	1	x	Передается байт данных, принимается или не принимается подтверждение
		Действие без загрузки TWDR или	1	0	1	x	Передается ПОВТОРНЫЙ СТАРТ
		Действие без загрузки TWDR или	0	1	1	x	Передается условие СТОП и сбрасывается флаг TWSTO
		Действие без загрузки TWDR	1	1	1	x	Вслед за условием СТОП передается условие СТАРТ и сбрасывается флаг TWSTO

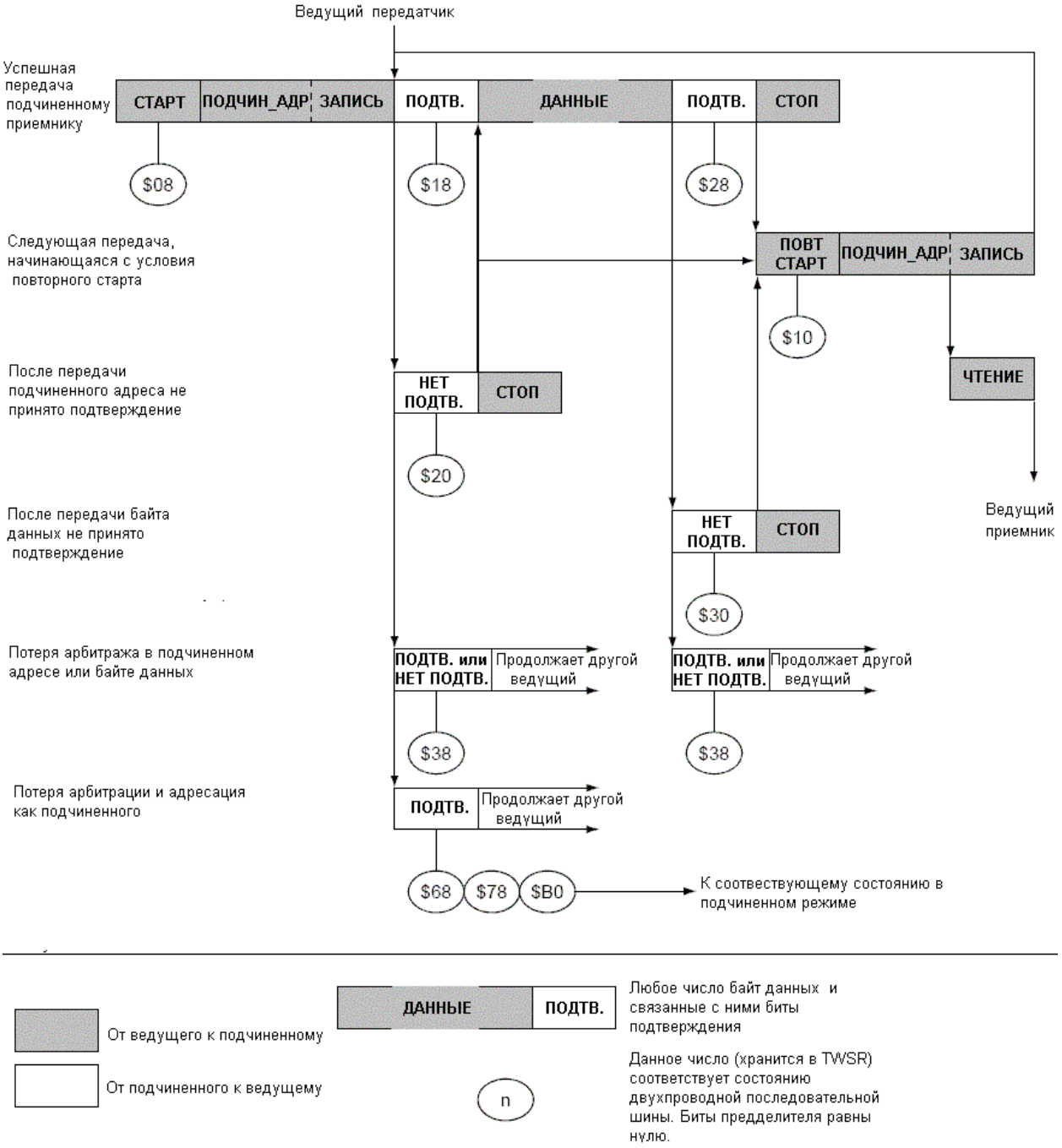


Рисунок 3.85 – Форматы и состояния в режиме ведущего передатчика

Режим ведущего приемника

В режиме ведущего приемника принимается несколько байт данных от ведомого приемника (см. рисунок 3.86). Для ввода режима ведущего необходимо передать условие СТАРТ. Формат следующего адресного пакета определит, будет ли введенный режим ведущий передатчик или приемник. Если передается ВЕДОМЫЙ_АДР + ЗАПИСЬ, то вводится режим ведущий передатчик, если же передается ВЕДОМЫЙ_АДР + ЧТЕНИЕ, то вводится режим ведущий приемник. Во всех кодах состояния, приведенных в этом подразделе, не учитываются биты предделителя, и они равны нулю.

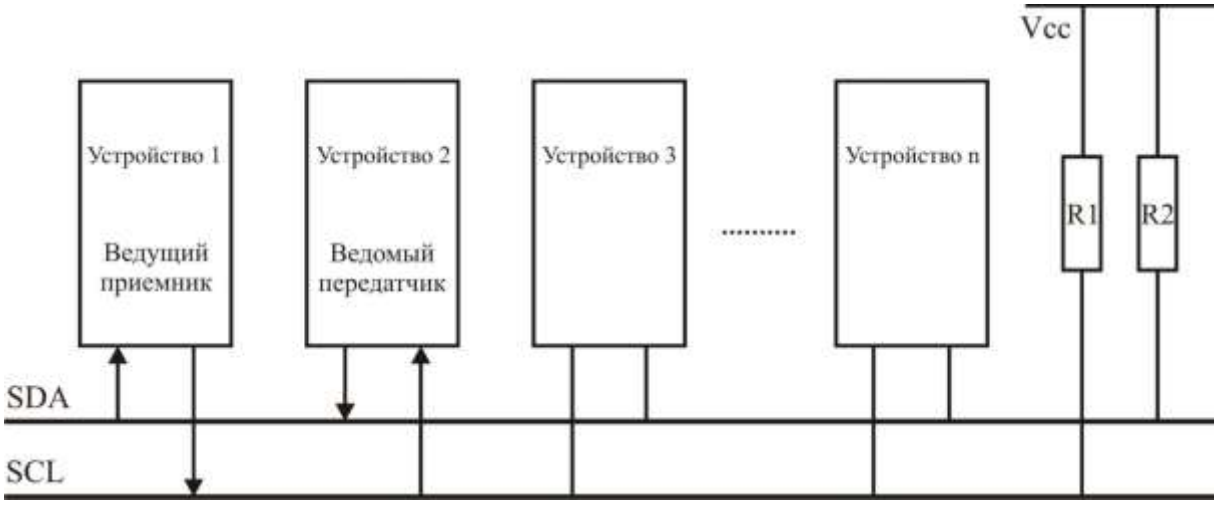


Рисунок 3.86 – Передача данных в режиме ведущего приемника

Передача условия СТАРТ инициируется путем записи следующего значения в TWCR:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

Для разрешения работы двухпроводного последовательного интерфейса необходимо установить бит TWEN. Передача условия СТАРТ инициируется записью логической 1 в TWSTA. Для сброса флага TWINT необходимо записать в него логическую 1. TWI выполнит генерацию условия СТАРТ только после тестирования шины и определения ее освобождения. После передачи условия СТАРТ флаг TWINT устанавливается аппаратно, а в регистр TWSR помещается код состояния $08 (см. таблицу 3.75). Для ввода режима ведущий приемник необходимо передать ВЕДОМЫЙ_АДР + ЧТЕНИЕ, выполняемое путем записи значения ВЕДОМЫЙ_АДР + ЧТЕНИЕ в TWDR. После этого необходимо сбросить флаг TWINT (путем записи в него логической 1) для продолжения сеанса связи. Для этого в регистр TWCR необходимо поместить следующее значение:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	0	x	1	0	x

После передачи ВЕДОМЫЙ_АДР + ЧТЕНИЕ и приема бита подтверждения устанавливается снова флаг TWINT, а в регистр TWSR помещается код состояния, который может иметь несколько значений: $38, $40 или $48. Действия, которые выполняются при каждом из этих значений, представлены в таблице 3.75. Принятые данные хранятся в регистре TWDR после аппаратной установки флага TWINT. Данная последовательность повторяется до приема последнего байта. После этого ведущий приемник информирует подчиненный передатчик отправкой НЕТ ПОДТВ. (нет подтверждения) после приема последнего принятого байта данных. Сеанс связи завершается генерацией условия СТОП или ПОВТОРНЫЙ СТАРТ. Условие СТОП генерируется путем записи в регистр TWCR следующего значения:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	0	1	x	1	0	x

Условие ПОВТОРНЫЙ СТАРТ генерируется путем записи в TWCR следующего значения:

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	1	x	1	0	x	1	0	x

После генерации условия ПОВТОРНЫЙ СТАРТ (состояние $10) двухпроводной последовательный интерфейс может обращаться к тому же подчиненному или к новому подчиненному без генерации условия СТОП. ПОВТОРНЫЙ СТАРТ позволяет ведущему переключаться между подчиненными, режимом ведущего передатчика и ведущего приемника без потери управления над шиной (см. рисунок 3.87).

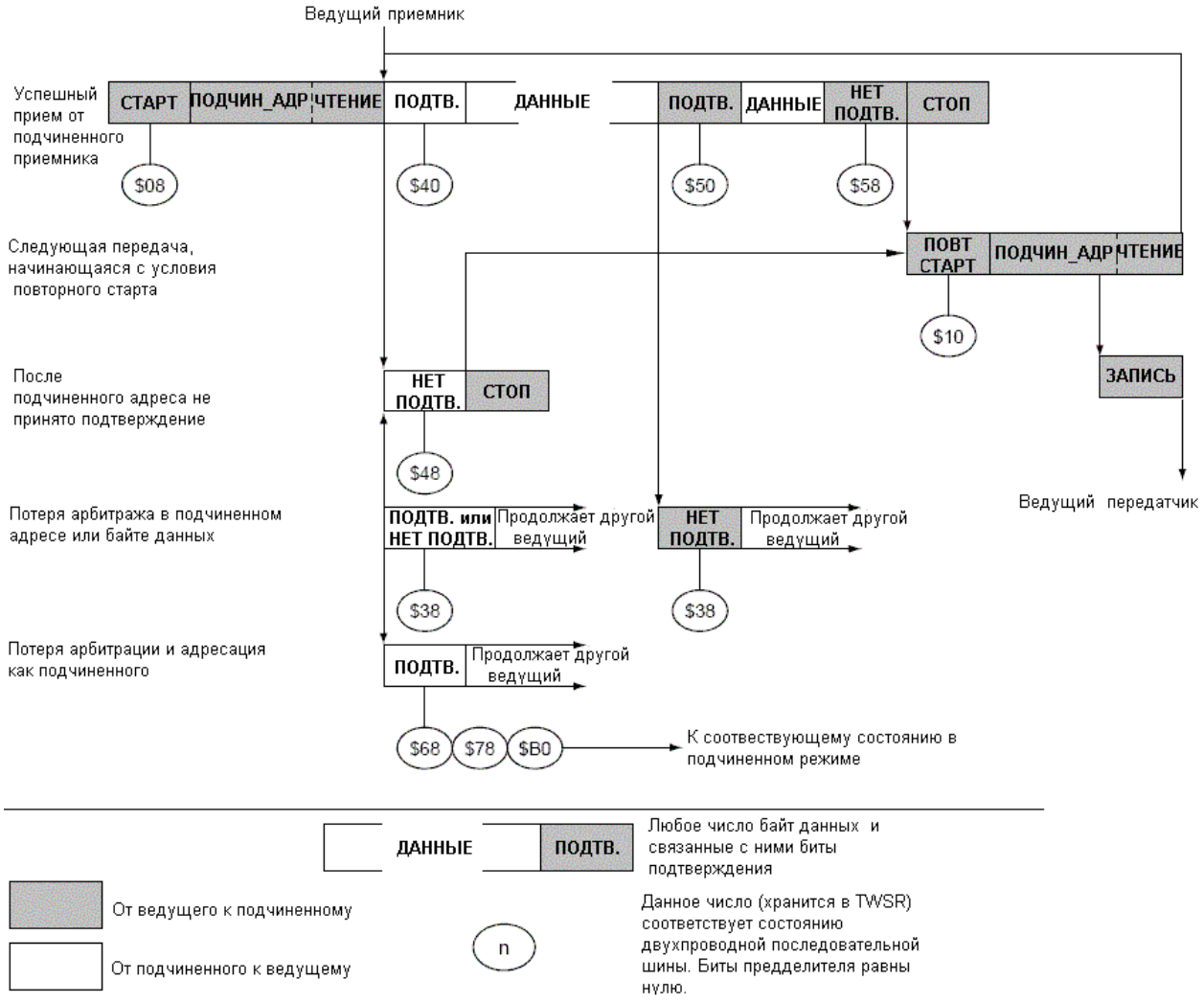


Рисунок 3.87 – Форматы и состояния в режиме ведущего приемника

Таблица 3.75 – Коды состояния для режима ведущий приемник

Код состояния (TWSR), биты предделителя	Состояние двупроводной последовательной шины	Программные действия					Следующее действие, выполняемое микросхемой
		В/из TWDR	В TWCR				
			STA	STO	TWINT	TWEA	
$08	Передано условие СТАРТ	Загрузка ВЕДОМЫЙ_АДР + + ЧТЕНИЕ	0	0	1	x	Передается ВЕДОМЫЙ_ _АДР + ЧТЕНИЕ Принимается ПОДТВ. или НЕТ ПОДТВ.
$10	Передано условие ПОВТОРНЫЙ СТАРТ	Загрузка ВЕДОМЫЙ_АДР + + ЧТЕНИЕ	0	0	1	x	Передается ВЕДОМЫЙ_ _АДР + ЧТЕНИЕ Принимается ПОДТВ. или НЕТ ПОДТВ.
		Или ВЕДОМЫЙ_ _АДР + ЗАПИСЬ	0	0	1	x	Передается ВЕДОМЫЙ_ _АДР + ЗАПИСЬ Переключение в режим ведущий передатчик
$38	Потеря приоритета во время передачи ВЕДОМЫЙ_ АДР + + ЧТЕНИЕ или нет бита НЕТ ПОДТВ.	Действия без загрузки TWDR	0	0	1	x	Шина освобождается и устанавливается неадресованный ведомый режим
		Действия без загрузки TWDR	1	0	1	x	Условие СТАРТ передается после освобождения шины
$40	Передано ВЕДОМЫЙ_АДР + + ЧТЕНИЕ или принято ПОДТВ.	Действия без загрузки TWDR	0	0	1	0	Принимается байт данных, возвращается бит НЕТ ПОДТВ.
		Действия без загрузки TWDR	0	0	1	1	Принимается байт данных, возвращается НЕТ ПОДТВ.
$48	Передано ВЕДОМЫЙ_АДР + + ЧТЕНИЕ или принято НЕТ ПОДТВ.	Действия без загрузки TWDR	1	0	1	x	Передается ПОВТОРНЫЙ СТАРТ
		Действия без загрузки TWDR	0	1	1	x	Передается условие СТОП и сбрасывается флаг TWSTO
		Действия без загрузки TWDR	1	1	1	x	Вслед за условием СТОП передается условие СТАРТ и сбрасывается флаг TWSTO
$50	Принят байт даннх и возвращено подтверждение	Чтение байта данных	0	0	1	0	Принимается байт данных, возвращается бит НЕТ ПОДТВ.
		Чтение байта данных	0	0	1	1	Принимается байт данных, возвращается бит ПОДТВ.
$58	Принят байт даннх и возвращено НЕТ подтверждения	Чтение байта данных	1	0	1	x	Передается ПОВТОРНЫЙ СТАРТ
		Чтение байта данных	0	1	1	x	Передается условие СТОП и сбрасывается флаг TWSTO
		Чтение байта данных	1	1	1	x	Вслед за условием СТОП передается условие СТАРТ и сбрасывается флаг TWSTO

Режим ведомого приемника

В режиме ведомого приемника принимается несколько байт данных от ведущего передатчика (смотри рисунки 3.88, 3.89). Во всех кодах статуса, приведенных в этом подразделе, не учитываются биты предделителя и они равны нулю.

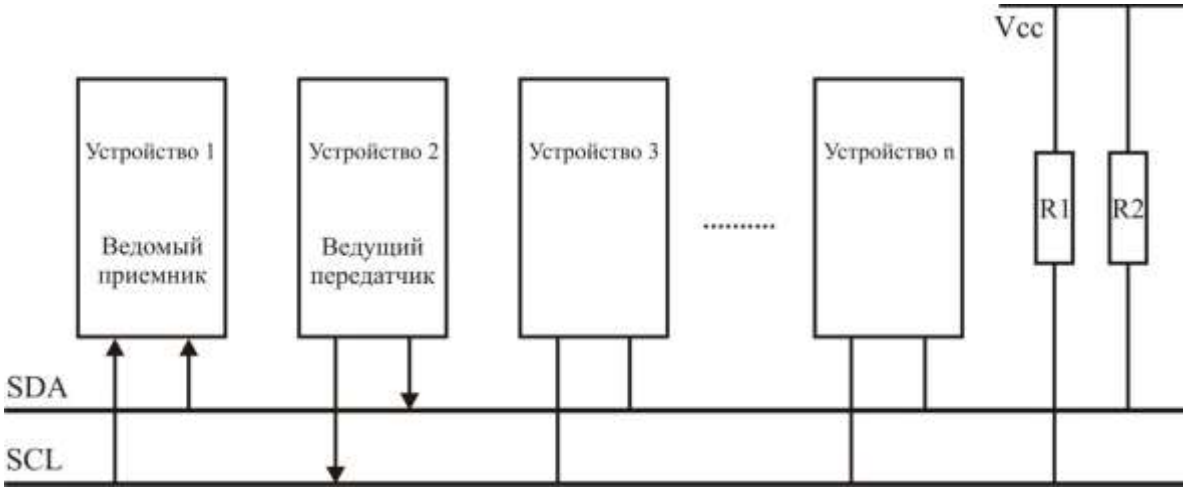


Рисунок 3.88 – Передача данных в режиме ведомого приемника

Для ввода режима ведомого приемника необходимо выполнить инициализацию регистров TWAR и TWCR следующим образом:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Значение	Собственный подчиненный адрес устройства							

Старшие семь разрядов образуют адрес. С помощью него определяется к какому двухпроводному интерфейсу адресуется ведущий. Если в младшем разряде записана логическая 1, то TWI будет отвечать на адрес общего вызова ($00). В противном случае он игнорирует адрес общего вызова.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	0	1	0	0	0	1	0	x

Для разрешения работы TWI необходимо записать логическую 1 в TWEN. Для разрешения подтверждения собственно ведомого адреса или адреса общего вызова записывается 1 в TWEA. Битам TWSTA и TWSTO необходимо присвоить нулевое значение.

После инициализации TWAR и TWCR схема TWI ожидает получения собственного адреса (или, если разрешено, адреса общего вызова), а вслед за ним - бита направления данных. Если бит направления равен «0» (запись), то TWI переходит в режим ведомый приемник, в противном случае вводится режим ведомый передатчик. После приема собственного адреса и бита записи устанавливается флаг TWINT, а в регистр TWSR помещается код статуса. По коду статуса определяется какие программные действия необходимо предпринять. В таблице 3.76 собрана информация о предпринимаемых действиях для каждого возможного значения кода состояния. Режим подчиненного приемника также вводится, если теряется приоритет, когда TWI находился в режиме ведущего (смотрите состояния $68 и $78).

Если бит TWEA сбросить во время передачи, то TWI ответит НЕТ ПОДТВ. («1») на линии SDA после приема следующего байта данных. Данное свойство может использоваться для сигнализации состояния, когда ведомый не может больше принимать байты данных. Если TWEA равен нулю, то TWI не подтверждает свой адрес. Однако последовательная шина остается под контролем, и функция распознавания адреса может быть активизирована в любой момент путем установки бита TWEA. Это означает, что бит TWEA можно использовать для временной изоляции TWI от двухпроводной последовательной шины.

Таблица 3.76 – Коды состояния для режима ведомый приемник

Код состояния (TWSR)	Состояние двупроводной последовательной шины	Программные действия					Следующее действие, выполняемое микросхемой
		В/из TWDR	В TWCR				
			STA	STO	TWINT	TWEA	
1	2	3	4	5	6	7	8
$60	Принимается собственный ВЕДОМЫЙ_АДР + + ЗАПИСЬ, возвращается подтверждение	Действия без загрузки TWDR	x	0	1	0	Принимается байт данных и возвращается НЕТ подтверждения
		Действия без загрузки TWDR	x	0	1	1	Принимается байт данных и возвращается подтверждение
$68	Потеря приоритета во время передачи ВЕДОМЫЙ_АДР+ЧТЕНИЕ, принят собственный ВЕДОМЫЙ_АДР + + ЗАПИСЬ, возвращено подтверждение	Действия без загрузки TWDR	x	0	1	0	Принимается байт данных и возвращается НЕТ подтверждения
		Действия без загрузки TWDR	x	0	1	1	Принимается байт данных и возвращается подтверждение
$70	Принят адрес общего вызова, возвращено подтверждение	Действия без загрузки TWDR	x	0	1	0	Принимается байт данных и возвращается НЕТ подтверждения
		Действия без загрузки TWDR	x	0	1	1	Принимается байт данных и возвращается подтверждение
$78	Потеря приоритета во время передачи ВЕДОМЫЙ_АДР + ЧТЕНИЕ/ /ЗАПИСЬ, приняты данные, возвращено подтверждение	Действия без загрузки TWDR	x	0	1	0	Принимается байт данных и возвращается НЕТ подтверждения
		Действия без загрузки TWDR	x	0	1	1	Принимается байт данных и возвращается подтверждение
$80	Адресация собственным ВЕДОМЫЙ_ АДР + + ЗАПИСЬ, приняты данные, возвращено подтверждение	Чтение байта данных	x	0	1	0	Принимается байт данных и возвращается НЕТ подтверждения
		Чтение байта данных	x	0	1	1	Принимается байт данных и возвращается подтверждение
$88	Адресация собственным ВЕДОМЫЙ_АДР+ + ЗАПИСЬ, приняты данные, возвращено НЕТ подтверждения	Чтение байта данных	0	0	1	0	Переключение в неадресованный «ведомый режим», не распознается собственный адрес или адрес общего вызова
		Чтение байта данных	0	0	1	1	Переключение в неадресованный «ведомый» режим, распознается собственный адрес или адрес общего вызова, если установлен TWGCE
		Чтение байта данных	1	0	1	0	Переключение в неадресованный «ведомый» режим, не распознается собственный адрес или адрес общего вызова, передается условие СТАРТ после освобождения шины
		Чтение байта данных	1	0	1	1	Переключение в неадресованный «ведомый» режим, распознается собственный адрес или адрес общего вызова, если установлен TWGCE, передается условие СТАРТ после освобождения шины

Окончание таблицы 3.76

1	2	3	4	5	6	7	8
$90	Адресация общим вызовом, приняты данные, возвращено подтверждение	Чтение байта данных	x	0	1	0	Принимается байт данных и возвращается НЕТ подтверждения
		Чтение байта данных	x	0	1	1	Принимается байт данных и возвращается подтверждение
$98	Адресация общим вызовом, приняты данные, возвращено НЕТ подтверждения	Чтение байта данных	0	0	1	0	Переключение в неадресованный «ведомый» режим, не распознается собственный адрес или адрес общего вызова
		Чтение байта данных	0	0	1	1	Переключение в неадресованный «ведомый» режим, распознается собственный адрес или адрес общего вызова, если установлен TWGCE
		Чтение байта данных	1	0	1	0	Переключение в неадресованный «ведомый» режим, не распознается собственный адрес или адрес общего вызова, передается условие СТАРТ после освобождения шины
		Чтение байта данных	1	0	1	1	Переключение в неадресованный «ведомый» режим, распознается собственный адрес или адрес общего вызова, если установлен TWGCE, передается условие СТАРТ после освобождения шины
$A0	Принято условие СТОП или ПОВТОРНЫЙ СТАРТ во время адресации как ведомого	Нет действий	0	0	1	0	Переключение в неадресованный «ведомый» режим, не распознается собственный адрес или адрес общего вызова
		Нет действий	0	0	1	1	Переключение в неадресованный «ведомый» режим, распознается собственный адрес или адрес общего вызова, если установлен TWGCE
		Нет действий	1	0	1	0	Переключение в неадресованный «ведомый» режим, не распознается собственный адрес или адрес общего вызова, передается условие СТАРТ после освобождения шины
		Нет действий	1	0	1	1	Переключение в неадресованный «ведомый» режим, распознается собственный адрес или адрес общего вызова, если установлен TWGCE, передается условие СТАРТ после освобождения шины

Первым шагом работы регистра TWI является передача условия СТАРТ. Это инициируется путем записи специфического значения в TWCR. О значении, которое необходимо записать, будет сказано позже. Однако необходимо следить, чтобы в записываемом в регистр значении был установлен бит TWINT. Запись логической 1 в TWINT сбрасывает

этот флаг. TWI не начнет работу до тех пор, пока будет установлен флаг TWINT в регистре TWCR. Сразу после сброса TWINT начинается передача условия СТАРТ.

После передачи условия СТАРТ устанавливается флаг TWINT в регистре TWCR, а содержимое TWSR обновляется значением кода состояния, индицирующего об успешной передаче условия СТАРТ.

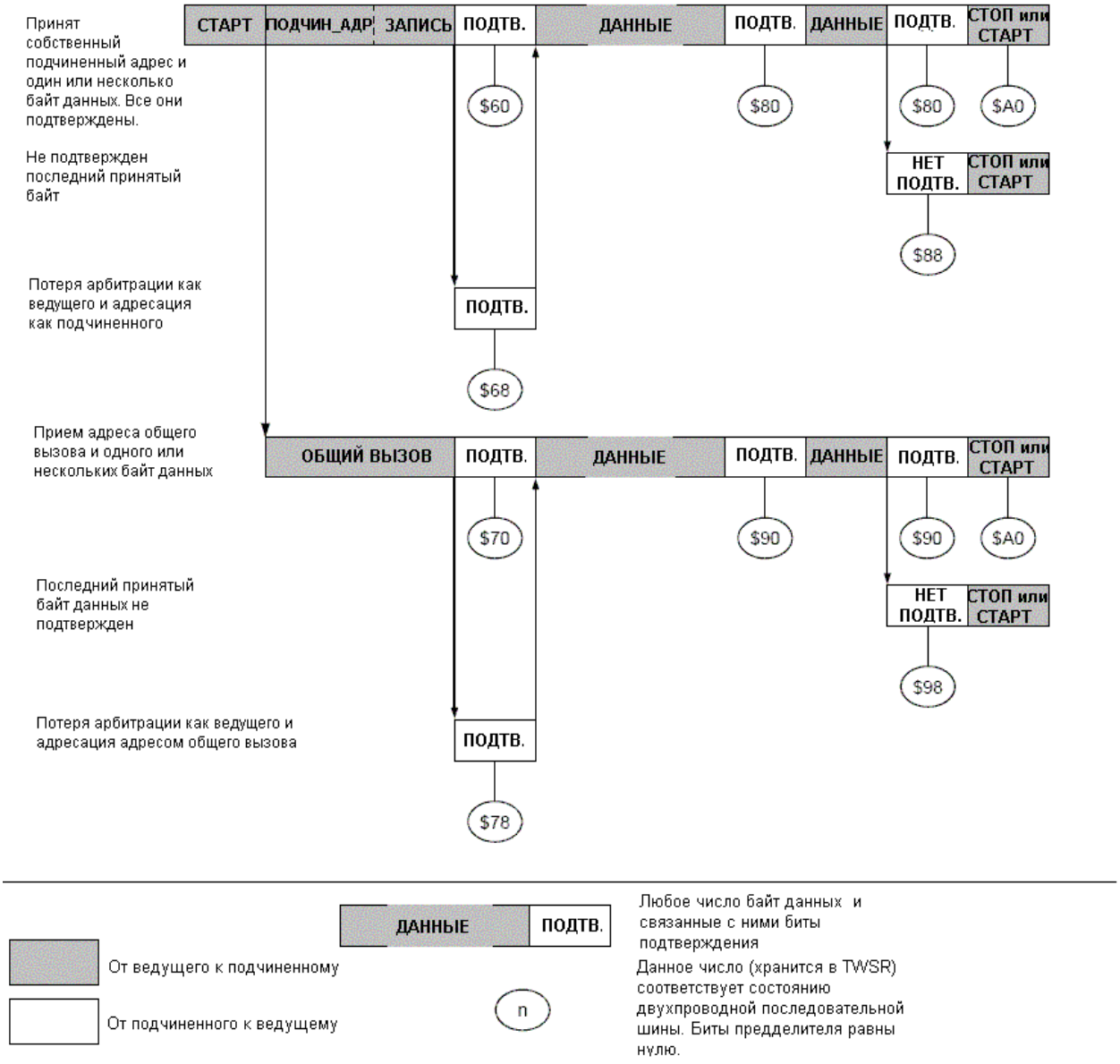


Рисунок 3.89 – Форматы и состояния в режиме ведомого приемника

Режим ведомого передатчика

В режиме ведомого передатчика выполняется передача нескольких байт данных ведущему приемнику (см. рисунки 3.90, 3.91). Во всех кодах статуса, приведенных в этом разделе, не учитываются биты предделителя и они равны нулю.

Для ввода режима ведомого передатчика необходимо инициализировать регистры TWAR и TWCR следующим образом:

TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Значение	Собственный подчиненный адрес устройства							

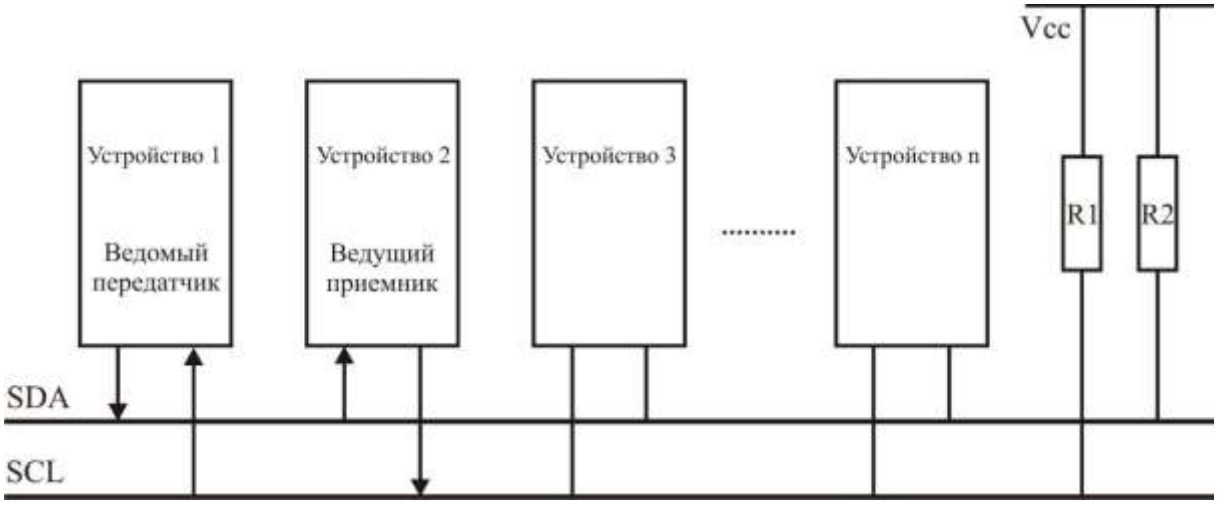


Рисунок 3.90 – Передача данных в режиме ведомого передатчика

Старшие семь разрядов образуют адрес. С помощью него определяется к какому двухпроводному интерфейсу адресуется ведущий. Если в младшем разряде записана логическая 1, то TWI будет отвечать на адрес общего вызова ($00). В противном случае он игнорирует адрес общего вызова.

TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Значение	0	1	0	0	0	1	0	x

Для разрешения работы TWI необходимо записать логическую 1 в TWEN. Для разрешения подтверждения собственного адреса или адреса общего вызова записывается единица в TWEA. Битам TWSTA и TWSTO необходимо присвоить нулевое значение.

После инициализации TWAR и TWCR схема TWI ожидает получения собственного адреса (или, если разрешен, адреса общего вызова), а вслед за ним – бита направления данных. Если бит направления равен «1» (чтение), то TWI переходит в режим ведомый передатчик, иначе вводится режим ведомый приемник. После приема собственного адреса и бита записи устанавливается флаг TWINT, а в регистр TWSR помещается код статуса. Код статуса позволяет определить, какие программные действия необходимо выполнить. Подробности по использованию кодов статуса представлены в таблице 3.77. Режим ведомый передатчик также вводится, если теряется приоритет, когда TWI находился в режиме ведущего (см. состояние $B0). Если бит TWEA обнулить во время передачи, то TWI передаст последний байт. Вводится состояние $C0 или состояние $C8 в зависимости от того принял ПОДТВ. или НЕТ ПОДТВ. ведущий приемник за последним байтом. TWI переходит в неадресованный подчиненный режим и далее игнорирует ведущего, если тот продолжает передачу. Таким образом, ведущий приемник принимает все «1» как последовательные данные. Состояние $C8 вводится, если ведущий требует передачи дополнительных байт данных (путем передачи ПОДТВ.), даже если подчиненный передал последний байт (TWEA равен нулю и ожидается прием НЕТ ПОДТВ. от ведущего).

Пока TWEA равен нулю, TWI не отвечает на собственный адрес. Однако последовательная шина остается под контролем и функция распознавания адреса может быть активизирована в любой момент путем установки бита TWEA. Это означает, что бит TWEA можно использовать для временной изоляции TWI от двухпроводной последовательной шины.

Во всех режимах сна, кроме холостого хода, синхронизация TWI отключается. Если бит TWEA установлен, то интерфейс останется способным подтверждать прием своего собственного адреса или адреса общего вызова за счет использования сигнала синхронизации шины в качестве тактового источника. При обнаружении запроса микроконтроллер

выходит из режима сна, при этом линия SCL остается на низком логическом уровне в процессе пробуждения и до сброса флага TWINT (записью в него «1»). Далее выполняется прием данных обычным способом при обычной системной синхронизации. Необходимо учесть, что если для микроконтроллера выбрано большое время запуска, то линия SCL может оказаться длительно в низком состоянии и заблокировать другой обмен информацией.

После пробуждения регистр данных TWDR не отражает последний байт, присутствовавший на шине во время выхода из указанных выше режимов сна.

Таблица 3.77 – Коды состояния для режима ведомый передатчик

Код состояния (TWSR), биты предделителя	Состояние двупроводной последовательной шины	Программные действия					Следующее действие, выполняемое микросхемой
		В/из TWDR	В TWCR				
			STA	STO	TWINT	TWEA	
1	2	3	4	5	6	7	8
$A8	Принимается собственный ВЕДОМЫЙ_ _АДР + ЧТЕНИЕ, возвращается подтверждение	Загрузка байта данных	x	0	1	0	Передается последний байт данных и должно быть передано НЕТ подтверждения
		Загрузка байта данных	x	0	1	1	Передается последний байт данных и должно быть передано подтверждение
$B0	Потеря приоритета во время передачи ВЕДОМЫЙ_АДР + + ЧТЕНИЕ/ЗАПИСЬ, принят собственный ВЕДОМЫЙ_АДР + + ЧТЕНИЕ, возвращено подтверждение	Загрузка байта данных	x	0	1	0	Передается последний байт данных и должно быть передано НЕТ подтверждения
		Загрузка байта данных	x	0	1	1	Передается последний байт данных и должно быть передано подтверждение
$B8	Передан байт данных, принято подтверждение	Загрузка байта данных	x	0	1	0	Передается последний байт данных и должно быть передано НЕТ подтверждения
		Загрузка байта данных	x	0	1	1	Передается последний байт данных и должно быть передано подтверждение
$C0	Передан байт данных, принято НЕТ подтверждения	Действия без загрузки TWDR	0	0	1	0	Переключение в неадресованный «ведомый» режим, не распознается собственный адрес или адрес общего вызова
		Действия без загрузки TWDR	0	0	1	1	Переключение в неадресованный «ведомый» режим, распознается собственный адрес или адрес общего вызова, если установлен TWGCE
		Действия без загрузки TWDR	1	0	1	0	Переключение в неадресованный «ведомый» режим, не распознается собственный адрес или адрес общего вызова, передается условие СТАРТ после освобождения шины
		Действия без загрузки TWDR	1	0	1	1	Переключение в неадресованный «ведомый» режим, распознается собственный адрес или адрес общего вызова, если установлен TWGCE, передается условие СТАРТ после освобождения шины

Окончание таблицы 3.77

1	2	3	4	5	6	7	8
$C8	Передан последний байт данных, принято подтверждение	Действия без загрузки TWDR	0	0	1	0	Переключение в неадресованный «ведомый» режим, не распознается собственный адрес или адрес общего вызова
		Действия без загрузки TWDR	0	0	1	1	Переключение в неадресованный «ведомый» режим, распознается собственный адрес или адрес общего вызова, если установлен TWGCE
		Действия без загрузки TWDR	1	0	1	0	Переключение в неадресованный «ведомый» режим, не распознается собственный адрес или адрес общего вызова, передается условие СТАРТ после освобождения шины
		Действия без загрузки TWDR	1	0	1	1	Переключение в неадресованный «ведомый» режим, распознается собственный адрес или адрес общего вызова, если установлен TWGCE, передается условие СТАРТ после освобождения шины

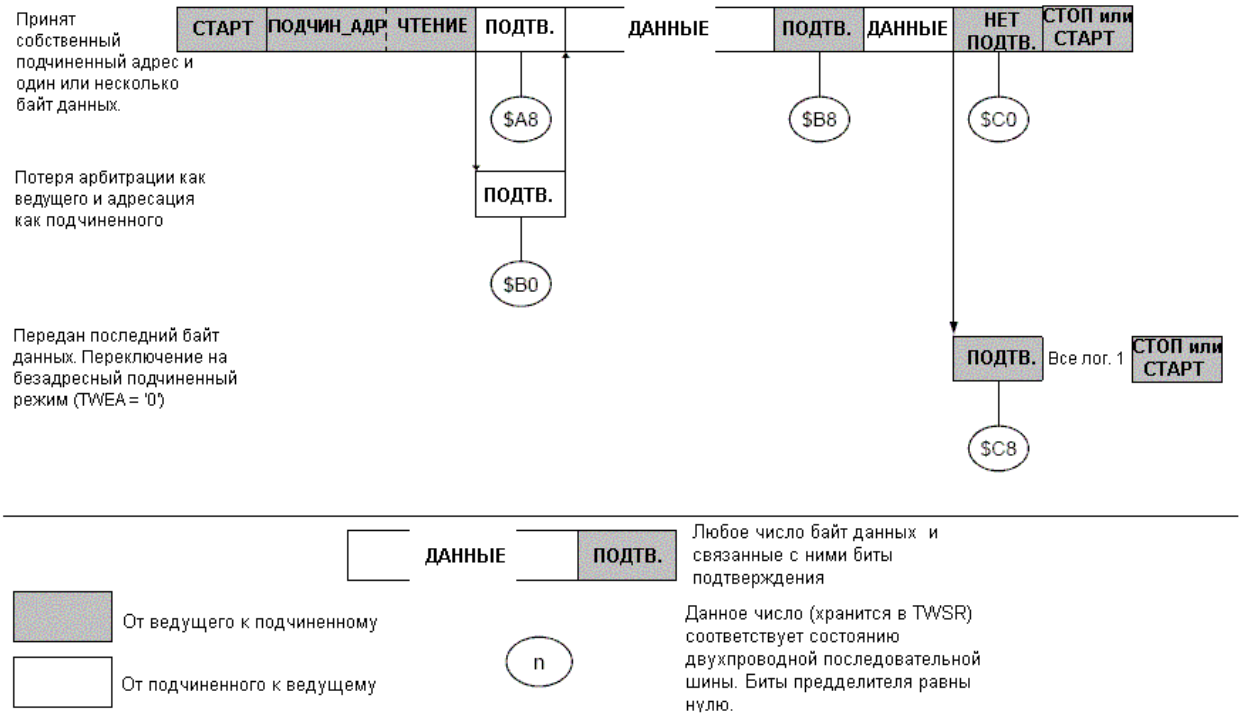


Рисунок 3.91 – Форматы и состояния в режиме ведомого передатчика

Прочие состояния

Имеются несколько кодов состояний, которые отличаются от упомянутых выше (см. таблицу 3.77). Состояние $F8 индицирует, что нет доступной информации, т. к. не установлен флаг TWINT. Это может произойти между другими состояниями и когда TWI не участвует в последовательной передаче данных.

Состояние $00 индицирует, что во время последовательной передачи данных на шине возникла ошибка. Ошибка возникает, если условия СТАРТ или СТОП возникают в неверной позиции формата посылки. Примеры таких неточных позиций могут существовать во время передачи адресного байта, байта данных и бита подтверждения. После воз-

никновения ошибки устанавливается флаг TWINT. Для выхода из состояния ошибки необходимо установить флаг TWSTO и сбросить TWINT путем записи в него «1». Это приводит к переводу TWI в безадресный режим и к сбросу флага TWSTO (другие биты в TWCR не затрагиваются). Линии SDA и SCL освобождаются, и условие СТОП не передается.

Сочетание нескольких режимов

В некоторых случаях сочетаются несколько режимов TWI для обеспечения желаемого действия. В качестве примера рассмотрим чтение данных из последовательного ЭСППЗУ. Обычно такой сеанс связи организовывается в такой последовательности:

- инициируется сеанс связи;
- в ЭССПЗУ отправляется инструкция с указанием адреса считываемой ячейки;
- выполняется чтение;
- завершается сеанс связи.

Следует обратить внимание, что данные передаются как от ведущего к подчиненному (ведомову), так и обратно, от подчиненного к ведущему. Ведущий инструктирует подчиненного какую ячейку он желает считать, для чего используется режим ведущий передатчик. В дальнейшем данные передаются подчиненным, что требует использования режима ведущий приемник. Следовательно, направление передачи данных изменяется. Ведущий должен сохранить управление над шиной на каждом из этапов, а каждый из шагов должен быть выполнен как элементарное действие. Если данный принцип нарушить в многомастерной системе, то другой ведущий может обратиться к ЭСППЗУ на шагах 2 и 3 и изменить указатель данных. Это приведет к тому, что ведущий считывает данные из ячейки с неверным адресом. Таким образом, направление передачи данных необходимо изменять только передачей условия ПОВТОРНЫЙ СТАРТ между передачей адресного байта и приемом байта. Передачей ПОВТОРНОГО СТАРТА ведущий сохранит свое «господство» на шине. На рисунке 3.92 представлена структура потока данной передачи.

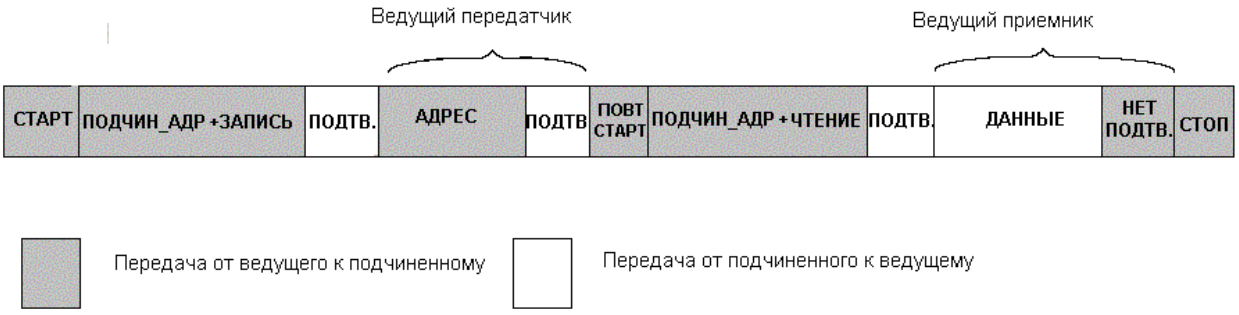


Рисунок 3.92 – Сочетание нескольких режимов TWI для обмена данными с последовательным ЭСППЗУ

Если к одной шине подключено несколько ведущих, то передача может быть инициирована одновременно одним или несколькими из них. Стандарт TWI гарантирует, что в таких ситуациях разрешается передача только одному ведущему, при этом не будет происходить потеря данных. Пример арбитража такой ситуации представлен ниже, где два ведущих пытаются передавать данные ведомому приемнику.

Ниже приведены несколько различных сценариев, возникающих в процессе арбитража:

1 Два или более ведущих выполняют идентичную связь с одним и тем же ведомым. В этом случае ни один ведомый и ни один из ведущих не узнает об этой конфликтной ситуации (смотри рисунок 3.93).

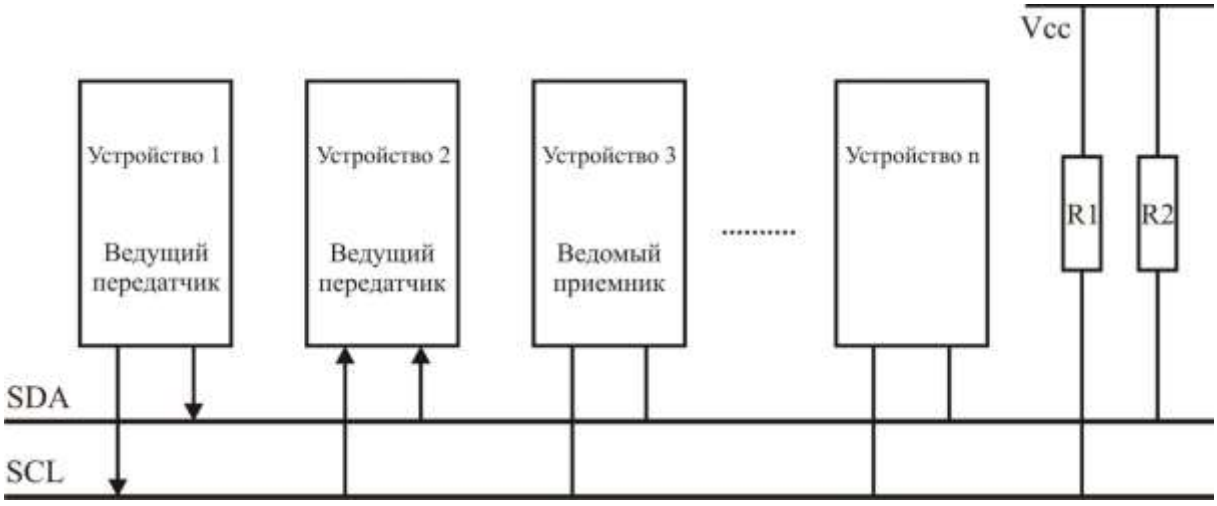


Рисунок 3.93 – Пример арбитража

2 Два или более ведущих обращаются к тому же ведомому с различными данными или битом направления данных. В этом случае возникает арбитраж во время передачи бита ЧТЕНИЕ/ЗАПИСЬ или же бит данных. Одни ведущие выводят на SDA логическую 1, а другие выводят логический 0. Последние теряют приоритет. Ведущие, которые проиграли арбитраж, переходят в неадресованный «ведомый» режим или ожидают освобождения шины и затем передают новое условие СТАРТ, что зависит от программных действий.

3 Два или более ведущих обращаются к различным ведомым. В этом случае арбитраж возникает во время передачи бит ВЕДОМЫЙ_АДР. Одни ведущие выводят на SDA логическую 1, а другие выводят логический 0. Последние теряют приоритет. Ведущие, которые проиграли арбитраж во время передачи ВЕДОМЫЙ_АДР, переходят в «ведомый» режим для проверки: не обращается ли к ним выигравший арбитраж «ведущий». Если такая адресация действительно выполняется, то они переключаются в режим ведомый приемник или ведомый передатчик в зависимости от значения бита ЧТЕНИЕ/ ЗАПИСЬ. Если адресации не было, то они перейдут в неадресованный «ведомый» режим или будут ожидать освобождения шины и после этого передадут новое условие СТАРТ (задается программно). Данные действия обобщены на рисунке 3.94. Возможные коды состояний представлены внутри окружностей.

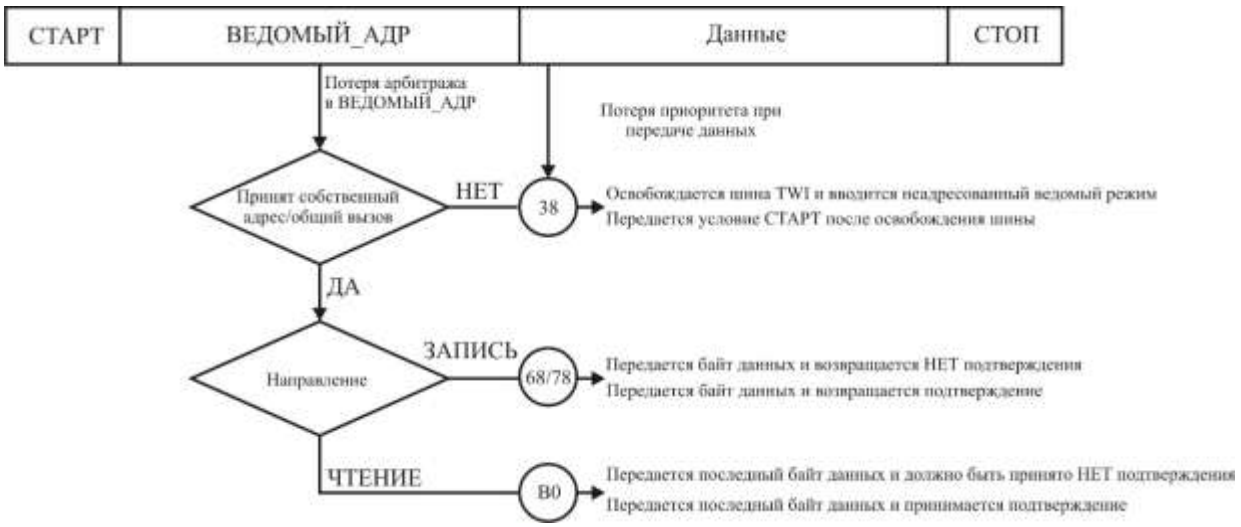


Рисунок 3.94 – Возможные коды состояний при потере арбитрации

3.17 Аналоговый компаратор

Аналоговый компаратор сравнивает уровни напряжений на неинвертирующем входе AIN0 и инвертирующем входе AIN1. Если напряжение на неинвертирующем входе AIN0 превышает напряжение на инвертирующем входе AIN1, то выход аналогового компаратора ACO принимает единичное состояние. Выход компаратора может быть настроен для использования в качестве источника входного сигнала для схемы захвата фронтов таймера-счетчика 1. Кроме того, компаратор может генерировать собственный запрос на обработку прерывания. Пользователь может выбрать несколько событий, по которым возникает прерывание: нарастающий, спадающий фронт на выходе компаратора или любое его изменение. Функциональная схема компаратора и связанной с ним логики представлена на рисунке 3.95. Параметры работы компаратора представлены в таблице 3.79.

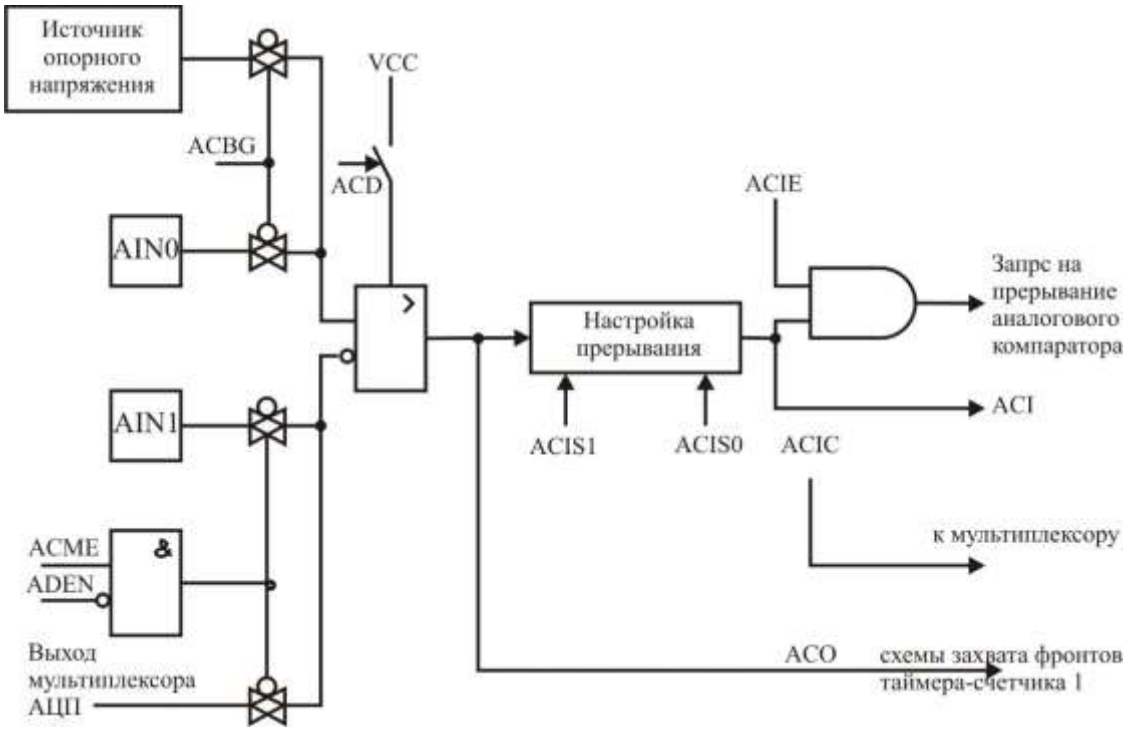


Рисунок 3.95 – Функциональная схема аналогового компаратора

Регистр специальных функций ввода-вывода – SFIOR

Разряд	7	6	5	4	3	2	1	0	
	TSM	-	-	-	ACME	PUD	PSR0	PSR10	SFIOR
	Чт/Зап	Чт	Чт	Чт	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 3: ACME – выбор мультиплексора на входе аналогового компаратора

Если выключен аналогово-цифровой преобразователь (ADEN = 0 в регистре ADCSRA) и в данный разряд записана логическая 1, то к инвертирующему входу аналогового компаратора подключен выход аналогового мультиплексора АЦП. Запись в данный разряд логического 0 приведет к подключению инвертирующего входа аналогового компаратора к выводу микроконтроллера AIN1.

Регистр состояния и управления аналогового компаратора ACSR:

Разряд	7	6	5	4	3	2	1	0	
	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
	Чт/Зап	Чт/Зап	Чт	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Разряд 7: ACD – отключение аналогового компаратора

Запись в данный разряд логической 1 приводит к отключению питания от аналогового компаратора. Данный разряд можно устанавливать в любой момент при необходимости отключения аналогового компаратора. Его использование позволяет снизить энергопотребление в активном режиме и режиме холостого хода. Перед изменением бита ACD необходимо отключить прерывание по аналоговому компаратору путем сброса бита ACIE в регистре ACSR. В противном случае может возникнуть прерывание после изменения значения данного бита.

Разряд 6: ACBG – подключение источника опорного напряжения к аналоговому компаратору

После установки данного бита к неинвертирующему входу компаратора подключается источник опорного напряжения. После сброса данного разряда неинвертирующий вход компаратора связан с выводом AIN0 микроконтроллера.

Разряд 5: ACO – выход аналогового компаратора

Данный бит выхода аналогового компаратора связан непосредственно с выходом ACO через цепь синхронизации. Синхронизация реализована как временная задержка на 1, 2 машинных цикла.

Разряд 4: ACI – флаг прерывания аналогового компаратора

Данный разряд устанавливается аппаратно, при возникновении события в соответствии с установками бит ACIS1 и ACIS0. Запрос на обработку прерывания аналогового компаратора выполняется, если установлены биты ACIE и «I» в регистре SREG. ACI сбрасывается аппаратно при переходе на соответствующий вектор обработки прерывания. Альтернативно бит ACI можно сбросить программно путем записи логической 1 в данный флаг.

Разряд 3: ACIE – разрешение прерывания аналогового компаратора

Если в данный разряд записана логическая 1 и установлен бит «I» в регистре статуса, то прерывание по аналоговому компаратору активизируется. Запись в данный разряд логического 0 приводит к отключению данного прерывания.

Разряд 2: ACIC – подключение аналогового компаратора к схеме захвата фронтов

Установка данного разряда приводит к разрешению совместной работы схемы захвата фронтов таймера-счетчика 1 и аналогового компаратора. В этом случае выход аналогового компаратора непосредственно подключен к входному каскаду схемы захвата фронтов, позволяя к компаратору добавить функции подавления шумов и настройки фронтов прерывания по захвату фронта таймером-счетчиком 1. После записи в данный разряд логического 0 связь между аналоговым компаратором и схемой захвата фронтов разрывается. Для активизации прерывания схемы захвата фронтов таймера-счетчика 1 по срабатыванию аналогового компаратора необходимо установить бит TICIE1 в регистре маски прерывания таймера (TIMSK).

Разряды 1, 0: ACIS1, ACIS0 – выбор события прерывания аналогового компаратора

Данные разряды определяют какое событие приводит к генерации запроса на прерывание аналогового компаратора. Варианты установок данных разрядов и их назначение:

ACIS1	ACIS0	Событие
0	0	Прерывание по любому изменению на выходе компаратора
0	1	Зарезервировано
1	0	Прерывание по спадающему фронту на выходе компаратора
1	1	Прерывание по нарастающему фронту на выходе компаратора

Перед изменением бит ACIS1/ACIS0 необходимо отключить прерывание по аналоговому компаратору путем сброса бита разрешения прерывания в регистре ACSR. В противном случае может возникнуть прерывание при изменении значений данных бит.

Мультиплексированный вход аналогового компаратора

Имеется возможность использовать выводы ADC(7–0) в качестве неинвертирующих входов аналогового компаратора. Для организации такого ввода используется мультиплексор АЦП и, следовательно, в этом случае АЦП должен быть отключен. Если установлен бит разрешения подключения мультиплексора к аналоговому компаратору (бит ACME в SFIOR) и выключен АЦП (ADEN = 0 в регистре ADCSRA), то состояние разрядов MUX(2–0) регистра ADMUX определяют какой вывод микроконтроллера подключен к неинвертирующему входу аналогового компаратора (смотри таблицу 3.79). Если ACME сброшен или установлен ADEN, то в качестве неинвертирующего входа аналогового компаратора используется вывод микроконтроллера AIN1.

Таблица 3.78 – Мультиплексированный вход аналогового компаратора

ACME	ADEN	MUX(2–0)	Неинвертирующий вход аналогового компаратора
0	X	xxx	AIN1
1	1	xxx	AIN1
1	0	000	ADC0
1	0	001	ADC1
1	0	010	ADC2
1	0	011	ADC3
1	0	100	ADC4
1	0	101	ADC5
1	0	110	ADC6
1	0	111	ADC7

Таблица 3.79 – Параметры работы аналогового компаратора

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма		
		не менее	тип	не более
1 Напряжение смещения по входу аналогового компаратора, мВ, $U_{\#VCC} = U_{\cap VCC} = 5{,}0$ В; 3,3 В; $V_{IN} = U_{\cap VCC}/2$	V_{ACIO}	–	–	40
2 Токи утечки по входу аналогового компаратора, нА, $U_{\#VCC} = U_{\cap VCC} = 5{,}0$ В; $V_{IN} = U_{\cap VCC}/2$	I_{ACLK}	–50	–	50
3 Задержка распространения аналогового компаратора, нс, $U_{\#VCC} = U_{\cap VCC} = 4{,}0$ В	t_{ACID}	–	500	–

3.18 Аналого-цифровой преобразователь

Отличительные особенности:

- 10-разрядное разрешение;
- интегральная нелинейность 0,5 младшего разряда;
- абсолютная погрешность ±2 младшего разряда;
- время преобразования (65 – 260) мкс;
- частота преобразования до 15 тысяч преобразований в секунду при максимальном разрешении;
- 8 мультиплексированных однополярных входов;
- 7 дифференциальных входных каналов;
- 2 дифференциальных входных канала с опциональным усилением на 10 и 200;
- представление результата с левосторонним или правосторонним выравниванием в 16-разрядном слове;
- диапазон входного напряжения АЦП от 0 В до $U_{\cap VCC}$;
- выборочный внутренний источник опорного напряжения на 2,56 В;
- режимы одиночного преобразования и автоматического перезапуска;
- прерывание по завершении преобразования АЦП;
- механизм подавления шумов в режиме сна.

Микросхема содержит 10-разрядный АЦП последовательного приближения. АЦП связан с 8-канальным аналоговым мультиплексором, восемь однополярных входов которого связаны с линиями порта F. Общий вывод входных сигналов должен иметь потенциал 0 В (т. е. связан с ∩OV). АЦП также поддерживает ввод 16 дифференциальных напряжений. Два дифференциальных входа (ADC1, ADC0 и ADC3, ADC2) содержат каскад со ступенчатым программируемым усилением: 0 дБ (1x), 20 дБ (10x) или 46 дБ (200x). Семь дифференциальных аналоговых каналов используют общий инвертирующий вход (ADC1), а все остальные входы АЦП выполняют функцию неинвертирующих входов. Если выбрано усиление 1x или 10x, то можно ожидать 8-разрядное разрешение, а если 200x, то 7-разрядное.

АЦП содержит УВХ (устройство выборки-хранения), которое поддерживает на постоянном уровне напряжение на входе АЦП во время преобразования. Функциональная схема АЦП показана на рисунке 3.96.

АЦП имеет отдельный вывод питания ∩VCC (аналоговое питание). Напряжение $U_{\cap VCC}$ не должно отличаться более чем на ± 0,3 В от $U_{\#VCC}$.

В качестве внутреннего опорного напряжения может выступать напряжение от внутреннего ИОН на 2,56 В или напряжение $U_{\cap VCC}$. Если требуется использование внешнего ИОН, то он должен быть подключен к выводу ∩REF с подключением к этому выводу блокировочного конденсатора для улучшения шумовых характеристик.

Параметры работы АЦП представлены в таблице 3.80.

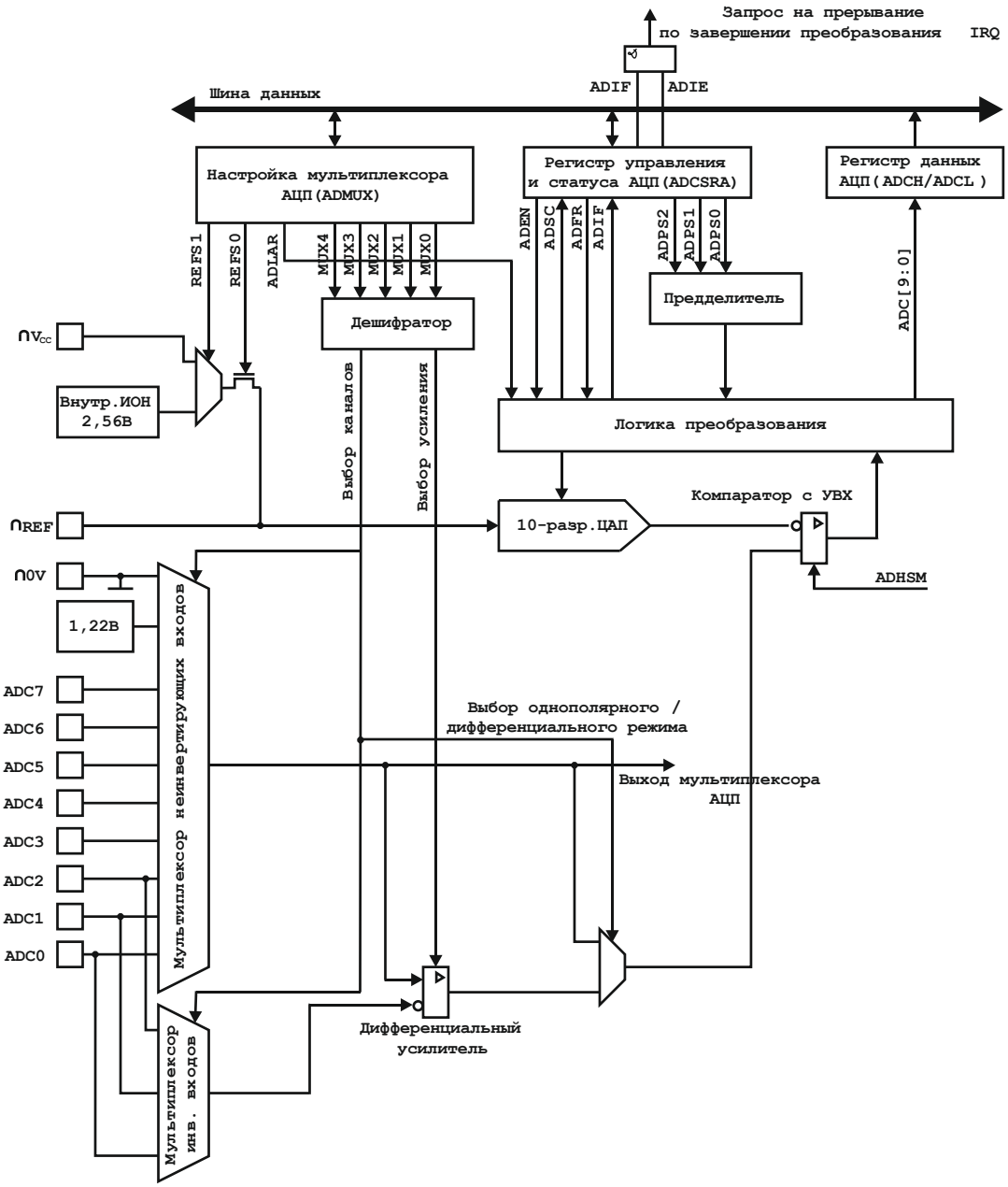


Рисунок 3.96 – Функциональная схема АЦП

3.18.1 Принцип действия АЦП

АЦП преобразовывает входное аналоговое напряжение в 10-разрядный код методом последовательных приближений. Минимальное значение соответствует уровню ∩OV, а максимальное уровню ∩REF минус 1 МЗР. К выводу ∩REF опционально может быть подключено напряжение $U_{\cap VCC}$ или внутренний ИОН на 1,22 В путем записи соответствующих значений в биты REFSn в регистр ADMUX. Несмотря на то, что ИОН на 2,56 В находится внутри микроконтроллера, к его выходу может быть подключен блокировочный конденсатор для снижения чувствительности к шумам, т. к. он связан с выводом ∩REF.

Канал аналогового ввода и каскад дифференциального усиления выбираются путем записи бит MUX в регистр ADMUX. В качестве однополярного аналогового входа АЦП может быть выбран один из входов ADC0–ADC7, а также ∩OV и выход фиксированного

источника опорного напряжения 1,22 В. В режиме дифференциального ввода предусмотрена возможность выбора инвертирующих и неинвертирующих входов к дифференциальному усилителю.

Если выбран дифференциальный режим аналогового ввода, то дифференциальный усилитель будет усиливать разность напряжений между выбранной парой входов на заданный коэффициент усиления. Усиленное таким образом значение поступает на аналоговый вход АЦП. Если выбирается однополярный режим аналогового ввода, то каскад усиления пропускается.

Работа АЦП разрешается путем установки бита ADEN в ADCSRA. Выбор опорного источника и канала преобразования невозможно выполнить до установки ADEN. Если ADEN = 0, то АЦП не потребляет ток, поэтому при переводе в экономичные режимы сна рекомендуется предварительно отключить АЦП.

АЦП генерирует 10-разрядный результат, который помещается в пару регистров данных АЦП: ADCH и ADCL. По умолчанию результат преобразования размещается в младших 10 разрядах 16-разрядного слова (выравнивание справа), но может быть опционально размещен в старших 10 разрядах (выравнивание слева) путем установки бита ADLAR в регистре ADMUX.

Практическая полезность представления результата с выравниванием слева существует, когда достаточно 8-разрядное разрешение, т. к. в этом случае необходимо считать только регистр ADCH. В другом же случае необходимо первым считать содержимое регистра ADCL, а затем ADCH, чем гарантируется, что оба байта являются результатом одного и того же преобразования. Как только выполнено чтение ADCL, блокируется доступ к регистрам данных со стороны АЦП. Это означает, что если считан ADCL и преобразование завершается перед чтением регистра ADCH, то ни один из регистров не может модифицироваться и результат преобразования теряется. После чтения ADCH доступ к регистрам ADCH и ADCL со стороны АЦП снова разрешается.

АЦП генерирует собственный запрос на прерывание по завершении преобразования. Если между чтением регистров ADCH и ADCL запрещен доступ к данным для АЦП, то прерывание возникнет, даже если результат преобразования будет потерян.

Запуск преобразования

Одиночное преобразование инициируется 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.

Альтернативно преобразование может быть инициировано автоматически различными источниками. Автоматическое инициирование включается установкой в ADC бита автоматического включения триггера ADATE в регистре ADCSRA. Источник запуска выбирается установкой бита выбора ADTS в SFIOR (см. описание битов ADTS для списка источников запуска). Когда положительный фронт происходит на выбранном пусковом сигнале, делитель частоты ADC сброшен и преобразование запущено. Этот метод обеспечивает преобразование в фиксированных интервалах. Если пусковой сигнал все еще будет установлен, когда преобразование завершится, то новое преобразование не будет запущено. Если другой положительный фронт произойдет на пусковом сигнале во время преобразования, то фронт будет проигнорирован. Обратите внимание на то, что флаг прерывания будет установлен, даже если определенное прерывание будет отключено или глобальный бит разрешения прерывания в SREG очищен. Преобразование может таким образом быть инициировано, не вызывая прерывание. Однако, флаг прерывания должен быть очищен, чтобы инициировать новое преобразование до следующего прерывания. Схема запуска преобразования приведена на рисунке 3.97.

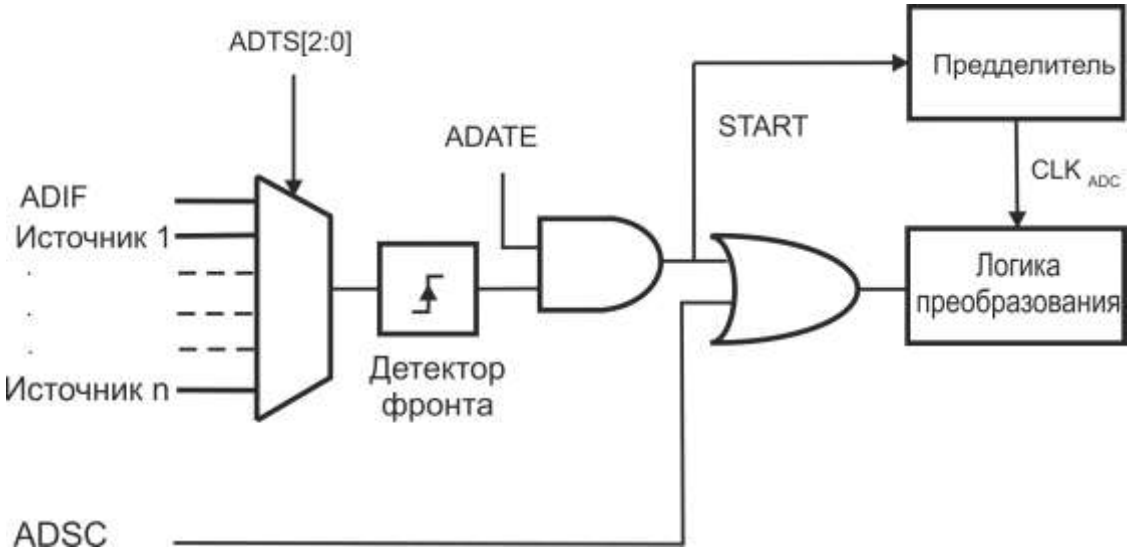


Рисунок 3.97 Схема запуска преобразования

Следует использовать флаг прерывания ADC, поскольку источник запуска заставляет запустить новое преобразование, как только преобразование закончилось. ADC работает в свободном режиме, постоянно выбирая и обновляя регистр данных ADC. Первое преобразование должно быть запущено при записи логической единицы в бит ADSC в ADCSRA. В этом режиме ADC выполнит последовательные преобразования, независимо от того очищен флаг прерывания ADIF или нет.

В автономном режиме работы АЦП непрерывно оцифровывает аналоговый сигнал и обновляет свой регистр данных. Автономный режим 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.

3.18.2 Предделитель АЦП и временная диаграмма преобразования

Предделитель АЦП приведен на рисунке 3.98, диаграммы – на рисунках 3.99 – 3.101.

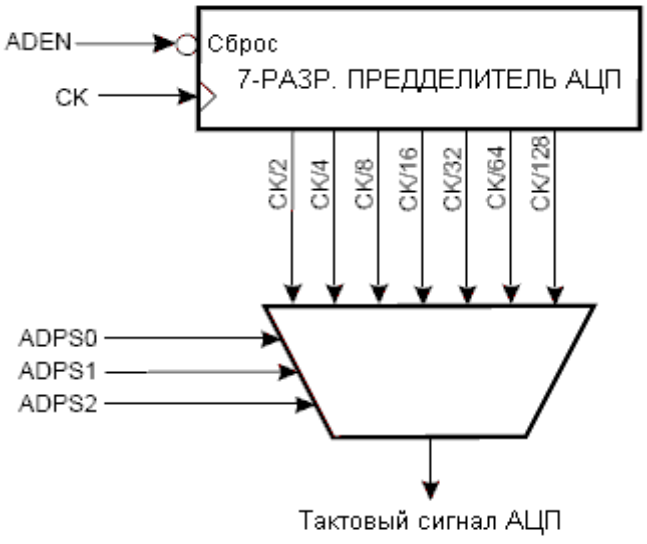


Рисунок 3.98 – Предделитель АЦП

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 свыше 200 кГц.

Модуль АЦП содержит предделитель, который формирует производные частоты свыше 100 кГц по отношению к частоте синхронизации ЦПУ. Коэффициент деления устанавливается с помощью бит ADPS в регистре ADCSRA. Предделитель начинает счет с момента включения АЦП установкой бита ADEN в регистре ADCSRA. Предделитель работает пока бит ADEN = 1 и сброшен, когда ADEN = 0.

Если инициируется однополярное преобразование установкой бита ADSC в регистре ADCSRA, то преобразование начинается со следующего нарастающего фронта тактового сигнала АЦП.

Нормальное преобразование требует 13 тактов синхронизации АЦП. Первое преобразование после включения АЦП (установка ADEN в ADCSRA) требует 25 тактов синхронизации АЦП за счет необходимости инициализации аналоговой схемы.

После начала нормального преобразования на выборку-хранение затрачивается полтора такта синхронизации АЦП, а после начала первого преобразования – 13,5 тактов. По завершении преобразования результат помещается в регистры данных АЦП и устанавливается флаг ADIF. В режиме одиночного преобразования одновременно сбрасывается бит ADSC. Программно бит ADSC может быть снова установлен и новое преобразование будет инициировано первым нарастающим фронтом тактового сигнала АЦП.

В режиме автоматического перезапуска новое преобразование начинается сразу по завершении предыдущего, при этом ADSC остается в высоком состоянии.

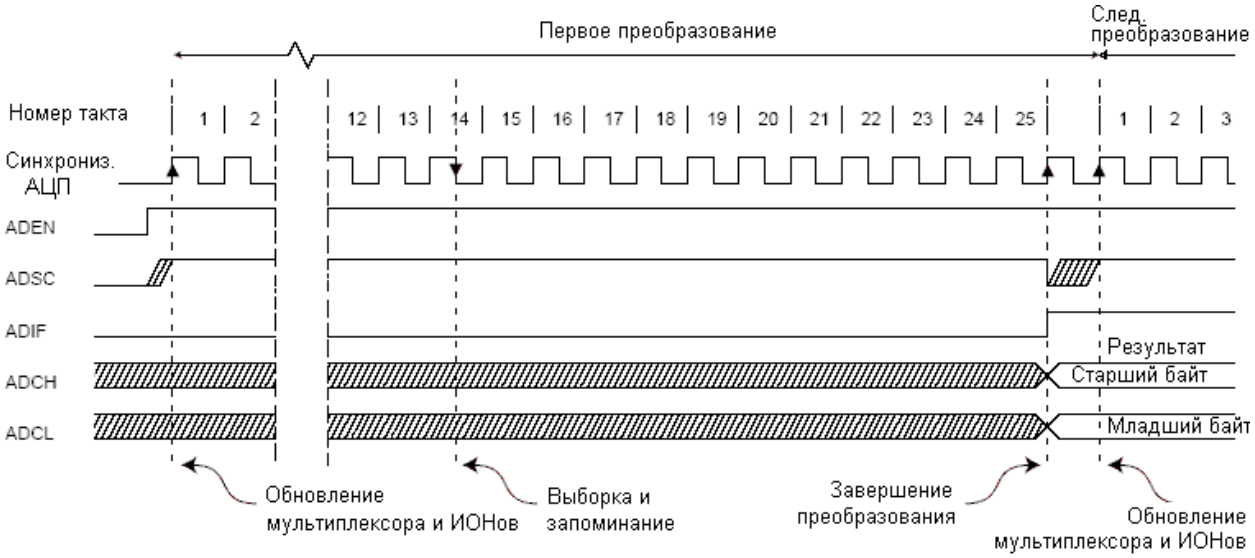


Рисунок 3.99 – Временная диаграмма работы АЦП при первом преобразовании в режиме одиночного преобразования

Рисунок 3.100 – Временная диаграмма работы АЦП в режиме одиночного преобразования

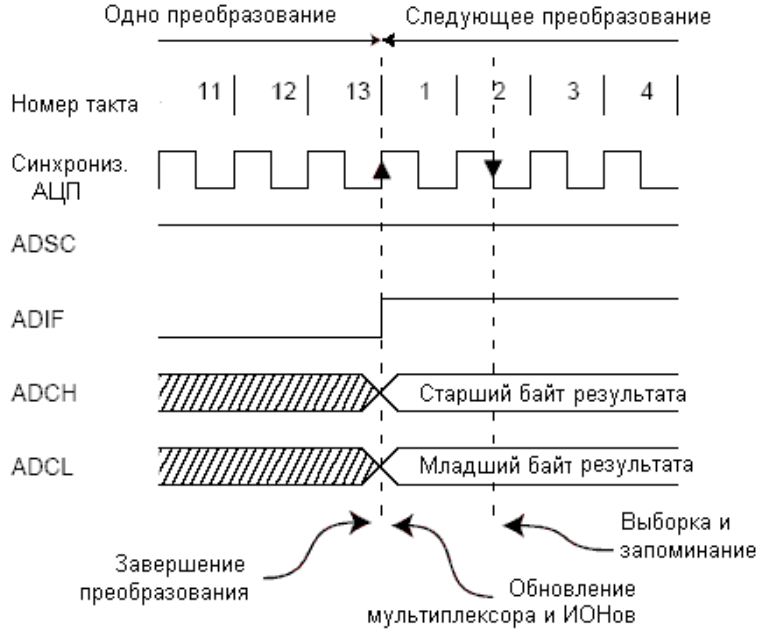


Рисунок 3.101 – Временная диаграмма работы АЦП в режиме автоматического перезапуска

3.18.3 Каналы дифференциального усиления

Если используются каналы дифференциального усиления, то необходимо принять во внимание некоторые особенности.

Дифференциальные преобразования синхронизированы по отношению к внутренней синхронизации $СК_{АЦП2}$, частота которого равна половине частоты синхронизации АЦП. Данная синхронизация выполняется автоматически интерфейсом АЦП таким образом, чтобы выборка-хранение инициировалась определенным фронтом $СК_{АЦП2}$. Если преобразование (все одиночные преобразования и первое преобразование в режиме автоматического перезапуска) инициировалось пользователем, когда $СК_{АЦП2}$ находился в низком логическом состоянии, то его длительность будет эквивалента однополярному преобразованию (13 тактов синхронизации АЦП). Если преобразование инициируется пользователем, когда $СК_{АЦП2}$ равен логической 1, оно будет длиться 14 тактов синхронизации АЦП

вследствие работы механизма синхронизации. В режиме автоматического перезапуска новое преобразование инициируется сразу по завершении предыдущего, а т. к. в этот момент $СК_{АЦП2}$ равен логической 1, то все преобразования, которые были автоматически перезапущены (т. е. все, кроме первого), будут длиться 14 тактов синхронизации АЦП. Усилительный каскад оптимизирован под частотный диапазон до 4 кГц для любых коэффициентов усиления. Усиление сигналов более высоких частот будет нелинейным. Поэтому, если входной сигнал содержит частотные составляющие выше частотного диапазона усилительного каскада, то необходимо установить внешний фильтр низких частот. Следует обратить внимание, что частота синхронизации АЦП не связана с ограничением по частотному диапазону усилительного каскада. Например, период синхронизации АЦП может быть 6 мкс, при котором частота преобразования канала равна 12 тысячам преобразований в секунду, независимо от частотного диапазона этого канала.

Изменение канала или выбор опорного источника

Биты MUXn и REFS(1–0) в регистре ADMUX поддерживают одноступенчатую буферизацию через временный регистр. Этим гарантируется, что новые настройки канала преобразования и опорного источника вступят в силу в безопасный момент для преобразования. До начала преобразования любые изменения канала и опорного источника вступают в силу сразу после их модификации. Как только начинается процесс преобразования, доступ к изменению канала и опорного источника блокируется, чем гарантируется достаточность времени на преобразование для АЦП. Непрерывность модификации возвращается на последнем такте АЦП перед завершением преобразования (перед установкой флага ADIF в регистре ADCSRA). Следует обратить внимание, что преобразование начинается следующим нарастающим фронтом тактового сигнала АЦП после записи ADSC. Таким образом, пользователю не рекомендуется записывать новое значение канала или опорного источника в ADMUX до первого такта синхронизации АЦП после записи ADSC.

Особые меры необходимо предпринять при изменении дифференциального канала. Как только осуществлен выбор дифференциального канала, усилительному каскаду требуется 125 мкс для стабилизации нового значения. Следовательно в течение первых после переключения дифференциального канала 125 мкс не должно стартовать преобразование. Если же в этот период преобразования все-таки выполнялись, то их результат необходимо игнорировать.

Такую же задержку на установление необходимо ввести при первом дифференциальном преобразовании после изменения опорного источника АЦП (за счет изменения бит REFS(1–0) в ADMUX).

Входные каналы АЦП

При переключении входного канала необходимо учесть некоторые рекомендации, которые исключат некорректность переключения:

- в режиме одиночного преобразования переключение канала необходимо выполнять перед началом преобразования. Переключение канала может произойти только в течение одного такта синхронизации АЦП после записи логической 1 в ADSC. Однако самым простым методом является ожидание завершения преобразования перед выбором нового канала;

- в режиме автоматического перезапуска канал необходимо выбирать перед началом первого преобразования. Переключение канала происходит аналогично – в течение одного такта синхронизации АЦП после записи логической 1 в ADSC. Но самым простым методом является ожидание завершения первого преобразования, а затем переключение канала. Поскольку следующее преобразование уже запущено автоматически, то следующий результат будет соответствовать предыдущему каналу. Последующие преобразования отражают результат для нового канала;

- при переключении на дифференциальный канал первое преобразование будет характеризоваться плохой точностью из-за переходного процесса в схеме автоматической

регулировки смещения. Следовательно, первый результат такого преобразования рекомендуется игнорировать.

3.18.4 Источник опорного напряжения АЦП

Источник опорного напряжения (ИОН) для АЦП ($V_{ИОН}$) определяет диапазон преобразования АЦП. Если уровень однополярного сигнала свыше $U_{Vион}$, то результатом преобразования будет 0x3FF. В качестве $U_{Vион}$ могут выступать $U_{\cap VCC}$, внутренний ИОН 2,56 В или внешний ИОН, подключенный к выводу ∩REF. ∩VCC подключается к АЦП через пассивный ключ. Внутреннее опорное напряжение 2,56 В генерируется внутренним эталонным источником VBG, буферизованным внутренним усилителем. В любом случае внешний вывод ∩REF связан непосредственно с АЦП, и поэтому можно снизить влияние шумов на опорный источник за счет подключения конденсатора между выводом ∩REF и общим. Напряжение $V_{ИОН}$ также может быть измерено на выводе ∩REF высокоомным вольтметром. Необходимо обратить внимание, что $V_{ИОН}$ является высокоомным источником и поэтому внешне к нему может быть подключена только емкостная нагрузка.

Если пользователь использует внешний опорный источник, подключенный к выводу ∩REF, то не допускается использование другой опции опорного источника, т. к. это приведет к шунтированию внешнего опорного напряжения. Если к выводу ∩REF не приложено напряжение, то пользователь может выбрать $U_{\cap VCC}$ и 2,56 В в качестве опорного источника. Результат первого преобразования после переключения опорного источника может характеризоваться плохой точностью и пользователю рекомендуется его игнорировать.

Если используются дифференциальные каналы, то выбранный опорный источник должен быть меньше уровня $U_{\cap VCC}$.

3.18.5 Подавитель шумов АЦП

АЦП характеризуется возможностью подавления шумов, которые вызваны работой ядра ЦПУ и периферийных устройств ввода-вывода. Подавитель шумов может быть использован в режиме снижения шумов АЦП и в режиме холостого хода. При использовании данной функции необходимо придерживаться следующей процедуры:

- следует убедиться, что работа АЦП разрешена и он не выполняет преобразования, выбирать режим одиночного преобразования и разрешить прерывание по завершении преобразования;
- ввести режим уменьшения шумов АЦП (или режим холостого хода). АЦП запустит преобразование как только остановится ЦПУ;
- если до завершения преобразования не возникает других прерываний, то АЦП вызовет прерывание ЦПУ и программа перейдет на вектор обработки прерывания по завершении преобразования АЦП. Если до завершения преобразования другое прерывание пробуждает микроконтроллер, то это прерывание обрабатывается, а по завершении преобразования генерируется соответствующий запрос на прерывание. АЦП остается в активном режиме пока не будет выполнена очередная команда sleep;
- следует обратить внимание, что АЦП не отключается автоматически при переводе во все режимы сна, кроме режима холостого хода и снижения шумов АЦП. Поэтому пользователь должен предусмотреть запись логического 0 в бит ADEN перед переводом в такие режимы сна во избежание чрезмерного энергопотребления. Если работа АЦП была разрешена в таких режимах сна и пользователь желает выполнить дифференциальное преобразование, то после пробуждения необходимо включить, а затем выключить АЦП для инициации расширенного преобразования, чем будет гарантировано получение действительного результата.

3.18.6 Схема аналогового входа

Схема аналогового входа для однополярных каналов представлена на рисунке 3.102. Независимо от того, какой канал подключен к АЦП, аналоговый сигнал, подключенный к выводу ADCn, нагружается емкостью вывода и входным сопротивлением утечки. После подключения канала к АЦП аналоговый сигнал будет связан с конденсатором выборки-хранения через последовательный резистор, сопротивление которого эквивалентно всей входной цепи.

АЦП оптимизирован под аналоговые сигналы с выходным сопротивлением не более 10 кОм. Если используется такой источник сигнала, то время выборки незначительно. Если же используется источник с более высоким входным сопротивлением, то время выборки будет определяться временем, которое требуется для зарядки конденсатора выборки-хранения источником аналогового сигнала. Рекомендуется использовать источники только с малым выходным сопротивлением и медленно изменяющимися сигналами, т. к. в этом случае будет достаточно быстрым заряд конденсатора выборки-хранения. Параметры работы АЦП – в таблице 3.80.

По отношению к каналам с дифференциальным усилением рекомендуется использовать сигналы с внутренним сопротивлением до нескольких сотен килоом. Следует предусмотреть, чтобы в предварительных каскадах формирования аналогового сигнала ко входу АЦП не вносились частоты выше $f_{АЦП}/2$, в противном случае результат преобразования может быть некорректным. Если вероятность проникновения высоких частот существует, то рекомендуется перед АЦП установить фильтр низких частот.

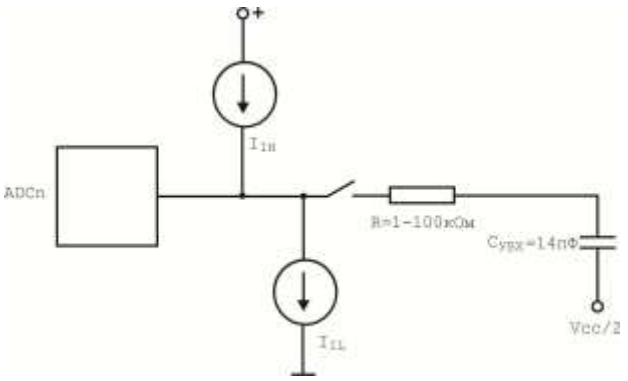


Рисунок 3.102 – Схема аналогового входа

Таблица 3.80 – Параметры работы АЦП

Наименование параметра, единица измерения	Режим измерения	Буквенное обозначение	Норма		
			не менее	Типовое	не более
1	2	3	4	5	6
Разрешение, бит	Несимметричный канал	–	–	10	–
	Дифференциальный канал (коэффициент усиления 1х, 10х)	–	–	8	–
	Дифференциальный канал (коэффициент усиления 200х)	–	–	7	–
Абсолютная точность, младший значащий разряд (МЗР)	Несимметричный канал $U_{\cap REF}$ = 4 В, частота АЦП = 200 кГц	–	–	1	–

Окончание таблицы 3.80

1	2	3	4	5	6
Дифференциальная нелинейность, МЗР	$U_{\cap REF}$ = 4 В	–	–	0,5	–
Смещение нуля, МЗР	$U_{\cap REF}$ = 4 В	–	–	1,0	–
Время преобразования, мкс	Режим непрерывного преобразования	–	65	–	260
Тактовая частота, кГц		–	50	–	200
Аналоговое напряжение питания, В		$U_{\cap VCC}$	$U_{\#VCC}$– –0,3	–	$U_{\#VCC}$ + 0,3
Опорное напряжение, В	Несимметричные каналы	$U_{\cap REF}$	2,0	–	$U_{\cap VCC}$
	Дифференциальные каналы		2,0	–	$U_{\cap VCC}$ – 0,2
Напряжение входных сигналов, В	Несимметричные каналы	U_{IN}	0		$U_{\cap REF}$
	Дифференциальные каналы		–	–	–
Напряжение внутреннего ИОН, В		U_{INT}	2,4	2,56	2,7
Входное сопротивление канала опорного напряжения, кОм		R_{REF}	6	10	13
Входное сопротивление аналогового входа, МОм		R_{AIN}	–	100	–

Характеристики АЦП приведены в таблицах 3.81, 3.82.

Таблица 3.81 – Характеристики АЦП, несимметричные каналы

Параметр, обозначение	Условия измерения	Мин.	Типовое	Макс.	Единица измерения
Разрешение	Одиночное преобразование	-	10	-	Биты
Абсолютная точность	Одиночное преобразование $U_{\cap REF}$ = 4 В, Ucc = 4 В, частота АЦП = 200 кГц	-	1,5	-	МЗР
	Одиночное преобразование $U_{\cap REF}$ = 4 В, Ucc = 4 В, частота АЦП = 1 МГц	-	3,0	-	МЗР
	Одиночное преобразование $U_{\cap REF}$ = 4 В, Ucc = 4 В, частота АЦП = 200 кГц Режим снижения шумов	-	1,5	-	МЗР
	Одиночное преобразование $U_{\cap REF}$ = 4 В, Ucc = 4 В, частота АЦП = 1 МГц, Режим снижения шумов	-	3,0	-	МЗР
Интегральная нелинейность	Одиночное преобразование $U_{\cap REF}$ = 4 В, U_{CC} = 4 В, частота АЦП = 200 кГц	-	0,75	-	МЗР
Дифференциальная нелинейность	Одиночное преобразование $U_{\cap REF}$ = 4 В, U_{CC} = 4 В, частота АЦП= 200 кГц	-	0,5	-	МЗР
Ошибка смещения	Одиночное преобразование $U_{\cap REF}$ = 4 В, U_{CC} = 4 В, частота АЦП = 200 кГц	-	1,0	-	МЗР
Время преобразования		13		260	мкс
Тактовая частота АЦП		50	-	200	кГц
Аналоговое напряжение питания, $U_{\cap VCC}$		$U_{\#VCC}$- -0,3	-	$U_{\#VCC}$+ +0,3	В
Опорное напряжение, $U_{\cap REF}$		2,0		$U_{\cap VCC}$	В
Напряжение входных сигналов, U_{IN}		0		$U_{\cap VCC}$	В
Напряжение внутреннего ИОН, U_{INT}		2,4	2,56	2,7	В
Входное сопротивление канала опорного напряжения, R_{REF}		-	28	-	кОм
Входное сопротивление аналогового входа, R_{AIN}		-	50	-	МОм

Таблица 3.82 – Характеристики АЦП, дифференциальные каналы

Параметр, обозначение	Условия измерения	Мин.	Типовое	Макс.	Единица измерения
Разрешение	Коэф. усиления = 1х	-	-	8	Биты
	Коэф. усиления = 10х	-	-	8	Биты
	Коэф. усиления = 200х	-	-	7	Биты
Абсолютная точность	Коэф. усиления = 1х $U_{\cap REF}$ = 4 В, U_{CC}= 5 В, Частота АЦП= (50 – 200) кГц	-	18	-	МЗР
	Коэф. усиления = 10х $U_{\cap REF}$ = 4 В, U_{CC} = 5 В, Частота АЦП= (50 – 200) кГц	-	18	-	МЗР
	Коэф. усиления = 200х $U_{\cap REF}$ = 4 В, U_{CC} = 5 В, Частота АЦП = (50 – 200) кГц	-	7	-	МЗР
Интегральная нелинейность	Коэф. усиления = 1х $U_{\cap REF}$ = 4 В, U_{CC} = 5 В, Частота АЦП = (50 –200) кГц	-	1,5	-	МЗР
	Коэф. усиления = 10х $U_{\cap REF}$ = 4 В, U_{CC} = 5 В, Частота АЦП = (50 – 200) кГц	-	2	-	МЗР
	Коэф. усиления = 200х $U_{\cap REF}$ = 4 В, U_{CC} = 5 В, Частота АЦП = (50 – 200) кГц	-	5	-	МЗР
Ошибка усиления	Коэф. усиления = 1х	-	4	-	%
	Коэф. усиления = 10х	-	4	-	%
	Коэф. усиления = 200х	-	1,5	-	%
Время преобразования		13	-	260	мкс
Тактовая частота АЦП		50	-	200	кГц
Аналоговое напряжение питания, $U_{\cap VCC}$		$U_{\#VCC}$ -0,3	-	$U_{\#VCC}$ +0,3	В
Опорное напряжение, $U_{\cap REF}$		2	-	4	В
Дифференциальное входное напряжение, U_{DIFF}		-$U_{\cap REF}$/ коэф. усил.	-	$U_{\cap REF}$/ коэф. усил.	В
Напряжение внутреннего ИОН, U_{INT}		2,4	2,56	2,7	В
Входное сопротивление канала опорного напряжения, R_{REF}		-	28	-	кОм
Входное сопротивление аналогового входа, R_{AIN}		-	50	-	МОм

Рекомендации по снижению влияния шумов на результат преобразования

Работа цифровых узлов внутри и снаружи микроконтроллера связана с генерацией электромагнитных излучений, которые могут негативно отразиться на точности измерения аналогового сигнала. Если точность преобразования является критическим параметром, то уровень шумов можно снизить, придерживаясь следующих рекомендаций:

- выполнять путь аналоговых сигналов как можно более коротко. Следить, чтобы аналоговые сигналы проходили над плоскостью (слоем) с аналоговой землей (экраном) и далеко от проводников, передающих высокочастотные цифровые сигналы;
- вывод ∩VCC необходимо связать с цифровым питанием #VCC через LC-цепь в соответствии с рисунком 3.103;
- следует использовать функцию подавления шумов АЦП, внесенных работой ядра ЦПУ;
- если какой-либо из выводов АЦП используется как цифровой выход, то чрезвычайно важно не допустить переключение состояния этого выхода в процессе преобразования.

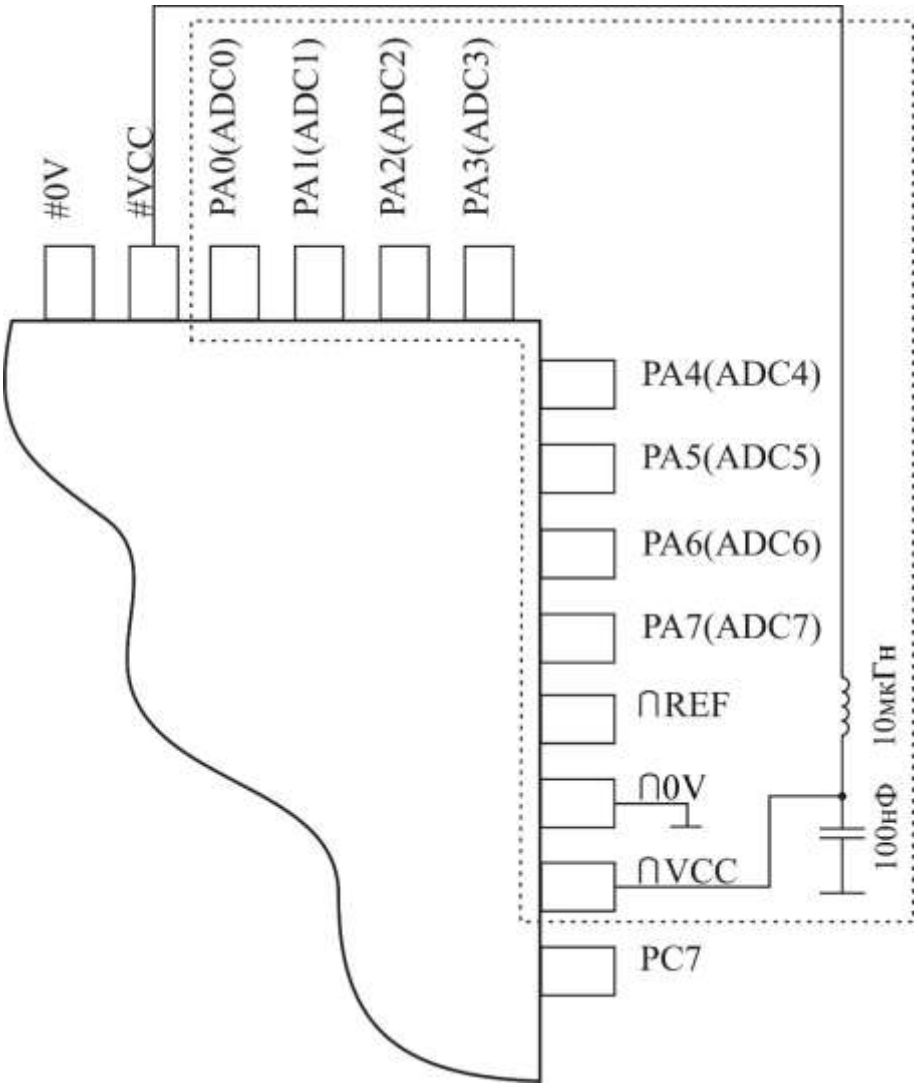


Рисунок 3.103 – Подключение питания АЦП

3.18.7 Методы компенсации смещения

Усилительный каскад имеет встроенную схему компенсации смещения, которая стремится максимально приблизить к нулю смещение дифференциального измерения.

Оставшееся смещение можно измерить, если в качестве дифференциальных входов АЦП выбрать один и тот же вывод микроконтроллера. Измеренное таким образом остаточное смещение можно программно вычесть из результата преобразования. Использование программного алгоритма коррекции смещения позволяет уменьшить смещение ниже одного младшего разряда.

Определения погрешностей аналогово-цифрового преобразования

Однополярный n-разрядный АЦП преобразовывает напряжение линейно между ∩OV и $V_{ИОН}$ с количеством шагов 2n (младших разрядов). Минимальный код = 0, максимальный код = 2n – 1. Основные погрешности преобразования являются отклонением реальной функции преобразования от идеальной. Ниже приведены примеры.

Смещение – отклонение первого перехода (с 0x000 на 0x001) по сравнению с идеальным переходом (т. е. при половине младшего разряда). Идеальное значение: «0» младший разряд.

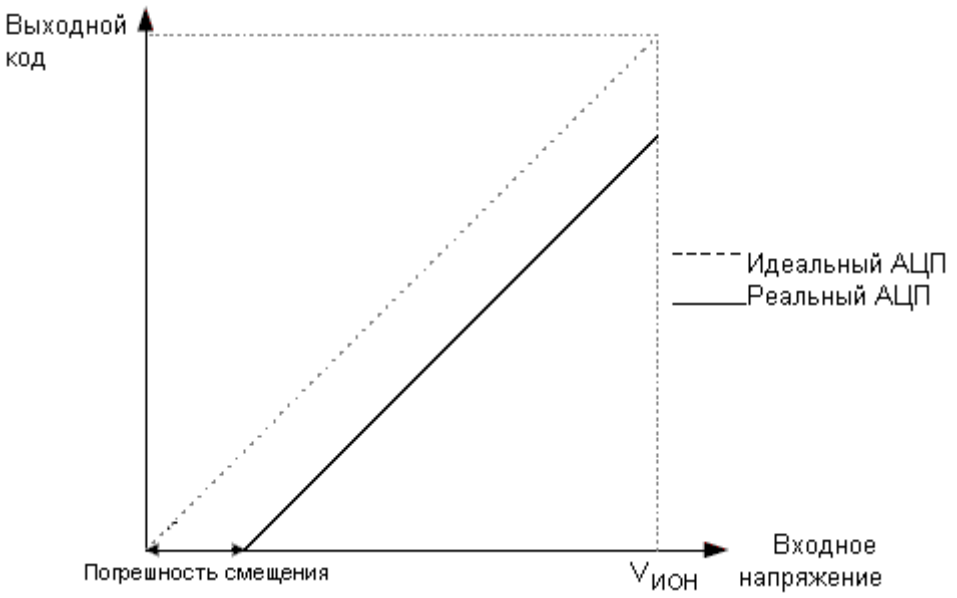


Рисунок 3.104 – Погрешность смещения

Погрешность усиления. После корректировки смещения погрешность усиления представляет собой отклонение последнего перехода (с 0x3FE на 0x3FF) от идеального перехода (т. е. отклонение при максимальном значении минус 1,5 младших разряда). Идеальное значение: «0» младших разрядов.

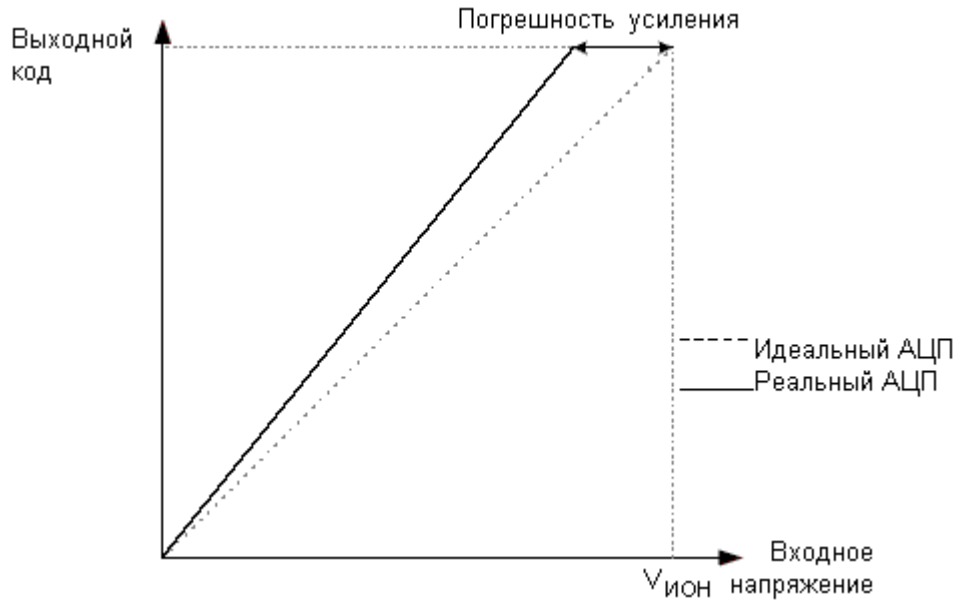


Рисунок 3.105 – Погрешность усиления

Интегральная нелинейность (ИНЛ). После корректировки смещения и погрешности усиления ИНЛ представляет собой максимальное отклонение реальной функции преобразования от идеальной для любого кода. Идеальное значение ИНЛ: «0» младший разряд.

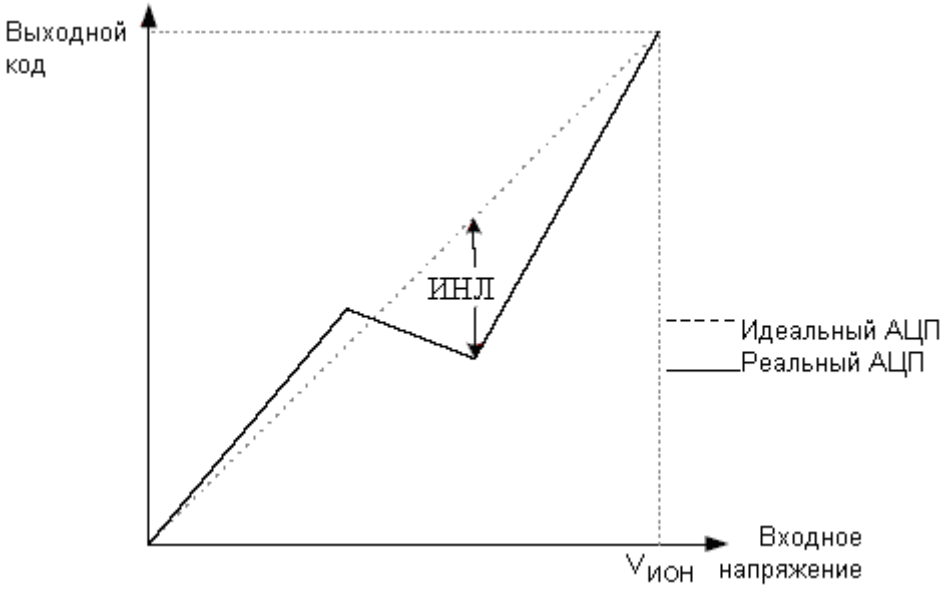


Рисунок 3.106 – Интегральная нелинейность (ИНЛ)

Дифференциальная нелинейность (ДНЛ) – максимальное отклонение между шириной фактического кода (интервал между двумя смежными переходами) от ширины идеального кода (1 младший разряд). Идеальное значение: «0» младший разряд.

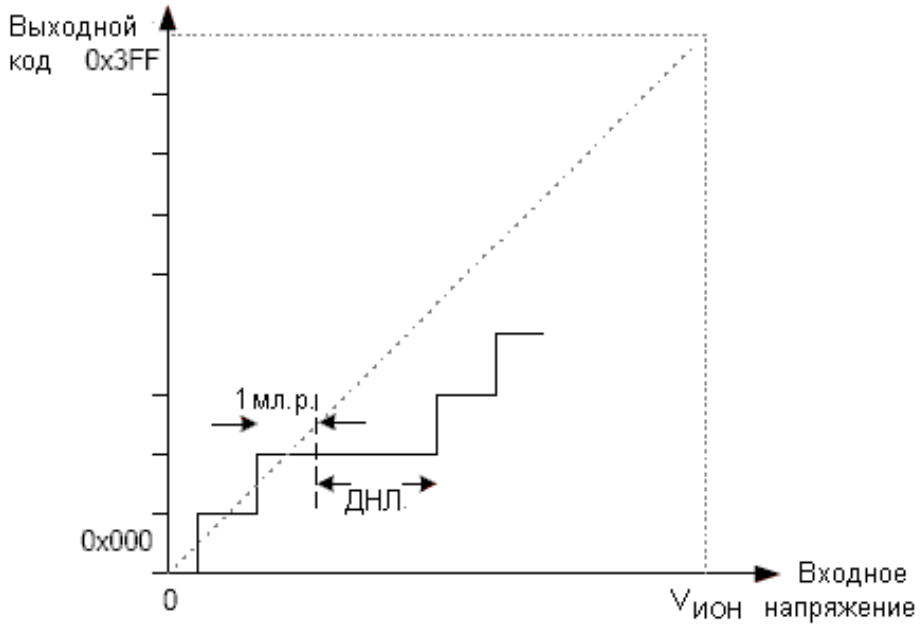


Рисунок 3.107 – Дифференциальная нелинейность (ДНЛ)

Погрешность квантования. Возникает из-за преобразования входного напряжения в конечное число кодов. Погрешность квантования – это интервал входного напряжения протяженностью один младший разряд (шаг квантования по напряжению), который характеризуется одним и тем же кодом. Всегда равен ±0,5 младшего разряда.

Абсолютная погрешность – максимальное отклонение реальной (без подстройки) функции преобразования от идеальной при любом коде. Является результатом действия нескольких эффектов: смещение, погрешность усиления, дифференциальная погрешность, нелинейность и погрешность квантования. Идеальное значение: ±0,5 младшего разряда.

Результат преобразования АЦП

По завершении преобразования (ADIF = 1) результат может быть считан из пары регистров данных АЦП (ADCL, ADCH).

Для однополярного преобразования результат 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 3.84 и 3.85).

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 регистре ADCH). Если данный бит равен логической единице, то результат отрицательный, если же он равен логическому нулю, то результат положительный. На рисунке 3.108 представлена функция преобразования АЦП в дифференциальном режиме.

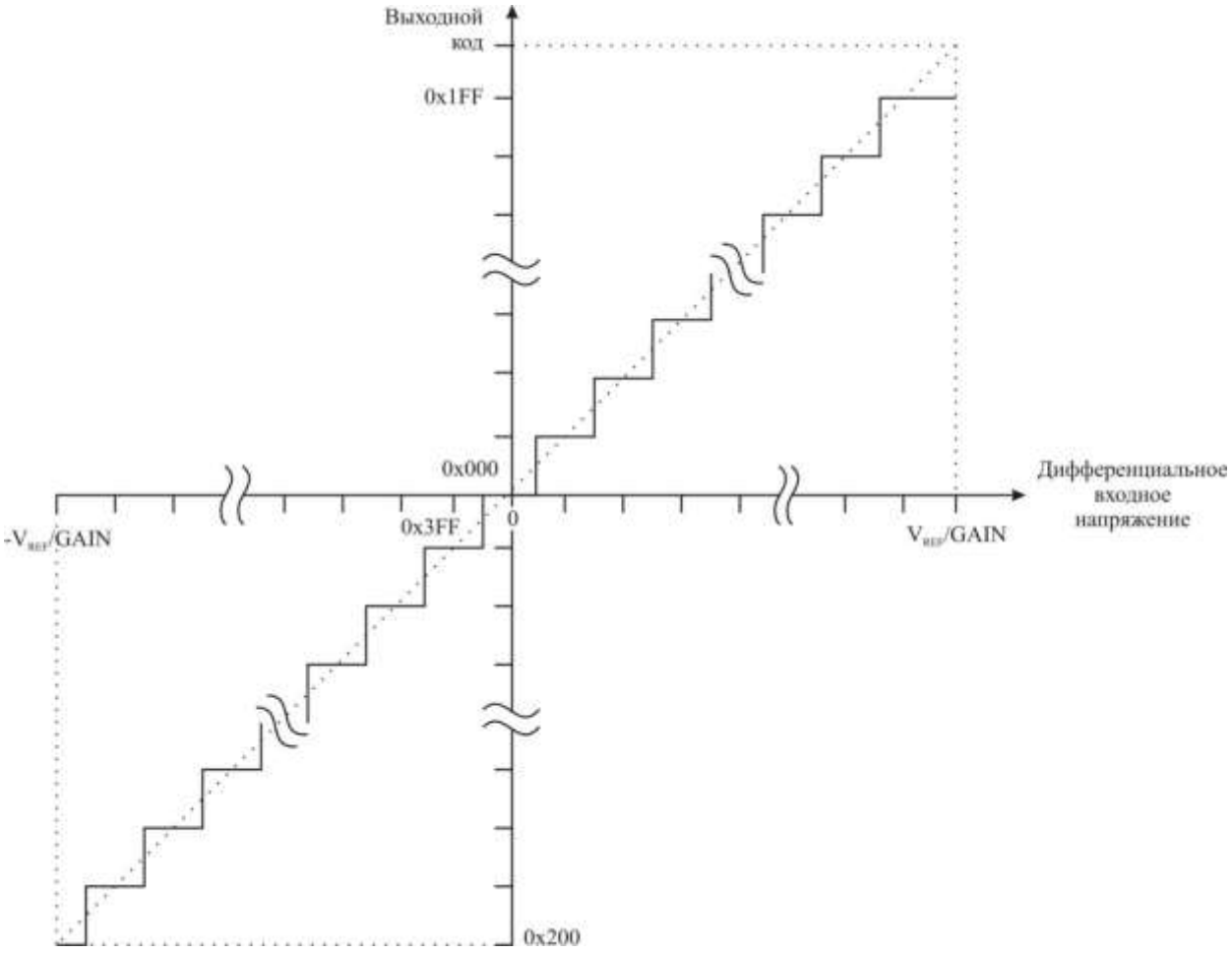


Рисунок 3.108 – Функция преобразования АЦП при измерении дифференциального сигнала

В таблице 3.83 представлены результирующие выходные коды для дифференциальной пары каналов (ADCn–ADCm) с коэффициентом усиления GAIN и опорным напряжением U_{REF}.

Таблица 3.83 – 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: REFS(1–0) – биты выбора источника опорного напряжения

Данные биты определяют, какое напряжение будет использоваться в качестве опорного для АЦП (см. таблицу 3.84). 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 3.84 – 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, если ADLAR = 0, то правосторонний. 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».

Разряды 4–0: MUX(4–0) – 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 3.85). 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 3.85 – Выбор входного канала и коэффициента усиления

MUX(4–0)	Однополярный 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			
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 (∩GND)			
* 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 ADEр 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 автономном режиме 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 инициализаци 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 автономного режима работы АЦП

Если в данный бит записать логическую единицу, то АЦП перейдет в автономный режим работы. 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: ADPS(2–0) – биты управления предделителем АЦП

Данные биты определяют коэффициент деления между частотой ЦПУ и частотой входной синхронизации АЦП (таблица 3.86).

Таблица 3.86 – 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 левостороннего выравнивания, и требуется точность не более восьми 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 левостороннего выравнивания. Если бит ADLAR сброшен (по умолчанию), то результат будет представлен в формате правостороннего выравнивания.

ADC(9–0) – результат преобразования АЦП

Данные биты представляют результат преобразования.

SFIOR

Бит	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10	SFIOR
Чтение/ Запись	Ч/З	Ч/З	Ч/З	Ч	Ч/З	Ч/З	Ч/З	Ч/З	
Начальное значение	0	0	0	0	0	0	0	0	

Бит 7–5: ADTS(2–0) – выбор автоматического запуска

Если в ADATE регистра ADCSRA записана 1, значение этих битов выбирает источник запуска преобразования ADC. Если бит ADATE очищен, то параметры битов ADTS(2–0) не важны. Преобразование будет запущено нарастающим фронтом выбранного флага прерывания. Обратите внимание на то, что переключение источника запуска из 0 в 1, генерирует положительный фронт на сигнале запуска. Если установлен бит ADEN в регистре ADCSRA, то преобразование начинается. Переключение в режим свободного доступа ADTS(2–0) = 0 не вызовет запуска события, даже если флаг прерывания установлен.

Таблица 3.87 – Выбор автоматического источника запуска

ADTS2	ADTS1	ADTS0	Источник запуска
0	0	0	Режим свободного запуска (автономный режим)
0	0	1	Аналоговый компаратор
0	1	0	Внешнее прерывание запрос 0
0	1	1	Таймер/счетчик 0 сравнение состояния
1	0	0	Таймер/счетчик 0 переполнение
1	0	1	Таймер/счетчик 1 сравнение состояния B
1	1	0	Таймер/счетчик 1 переполнение
1	1	1	Таймер/счетчик1 захват события

Бит 4: RES – резервный бит
Этот бит в ИС 1887ВЕ4У всегда читается как ноль.

3.19 Поддержка загрузчика – самопрограммирование

Для поддержки самопрограммирования вся область памяти программ логически разделена на две секции: секцию прикладной программы (Application Section) и секцию загрузчика (Boot Loader Section). Изменение памяти программ осуществляется программой-загрузчиком, расположенной в одноименной секции. Для загрузки нового содержимого памяти программ, а также для выгрузки старого содержимого, программа-загрузчик может использовать любой интерфейс передачи данных (UART, SPI, TWI), имеющийся в составе конкретного микроконтроллера. Следует отметить, что загрузчик может изменять содержимое обеих секций. Это позволяет ему модифицировать собственный код и даже удалять себя из памяти, если надобность в нем отпадет. Уровень доступа (чтение/запись) к каждой из секций задается пользователем с помощью ячеек защиты
BLB02–BLB01 и BLB12–BLB11.

Переход к программе-загрузчику может осуществляться различным образом. В частности, она может быть вызвана из основной программы командами RCALL/RJMP. Другим способом является перемещение вектора сброса в начало секции загрузчика. В этом случае запуск программы-загрузчика будет осуществляться автоматически после каждого сброса микроконтроллера. Положение вектора сброса определяется состоянием конфигурационной ячейки BOOTRST. Если в ней содержится «1», вектор сброса располагается в начале памяти программ по адресу $0000. При запрограммированнной ячейке, когда в ней содержится «0», вектор сброса располагается в начале секции загрузчика.

Размер секции загрузчика и размер секции прикладной программы задается с помощью двух конфигурационных ячеек BOOTSZ1–BOOTSZ0.

Таблица 3.88 – Конфигурация области загрузчика и области приложений в зависимости от бит BOOTSZ0 и BOOTSZ1

BOOTSZ1	BOOTSZ0	Размер области загрузчика, слова	Страниц	Область приложений	Область загрузчика	Окончание области приложений	Стартовый адрес области загрузчика
1	1	128	4	0x000–0xF7F	0xF80–0xFFF	0xF7F	0xF80
1	0	256	8	0x000–0xEFF	0xF00–0xFFF	0xEFF	0xF00
0	1	512	16	0x000–0xDFF	0xE00–0xFFF	0xDFF	0xE00
0	0	1024	32	0x000–0xBFF	0xC00–0xFFF	0xBFF	0xC00

В микроконтроллере вся память программ разбита на две области фиксированного размера, называемые «чтение во время записи» (RWW) и «нет чтения при записи» (NRWW). Отличие между этими областями заключается в различном поведении центрального процессора при изменении расположенных в них данных:

- во время выполнения операции стирания или записи страницы памяти программ, расположенной в области RWW, процессор может осуществлять чтение только из области NRWW;

- во время выполнения операции стирания или записи страницы памяти программ, расположенной в области NRWW, процессор останавливается до окончания этой операции.

Таким образом, во время изменения содержимого страницы памяти программ, расположенной в области RWW, чтение этой области запрещено. Попытка обратиться во время программирования к коду, находящемуся в области RWW (в результате выполнения команд RCALL/RJMP/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, которая всегда находится в области NRWW.

Для определения того, разрешено чтение из области RWW или нет, предназначен флаг RWWSB регистра SPMCR. Установленный в «1» флаг означает, что область RWW заблокирована для чтения. По окончании операции программирования флаг RWWSB должен быть сброшен программно (см. описание регистра SPMCR).

Напротив, код, расположенный в области NRWW, может быть считан во время изменения страницы памяти программ, расположенной в области RWW. А при изменении содержимого области NRWW процессор останавливается до завершения операции.

Таблица 3.89 – Ограничение на размер секции «Чтение во время записи»

Секция	Страниц	Адрес
Чтение во время записи (RWW)	96	0x000–0xBFF
Нет чтения во время записи (NRWW)	32	0xC00–0xFFF

3.19.1 Регистр управления SPMCR

Разряд	7	6	5	4	3	2	1	0	
	SPMIE	RWWSB	-	RWWSRE	BLBSET	PGWRT	PGERS	SPMEб	SPMCR
	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	Чт/Зап	
Начальное значение	0	0	0	0	0	0	0	0	

Бит 7: SPMIE – разрешение прерывания SPM

Когда в этот бит записана логическая единица и I бит в статусном регистре установлен, SPM прерывание разрешено. SPM готовность к прерыванию будет выполняться до тех пор, пока бит SPMEN в SPMCR очищен.

Бит 6: RWWSB – бит запрещения доступа к RWW секции

Когда самопрограммирование в RWW секции инициализировано, RWWSB бит аппаратно устанавливается. Когда он установлен, секция не может быть доступна. RWWSB бит очищается, когда в RWWSRE бит записывается логическая единица после того, как процедура самопрограммирования завершена. Альтернативно RWWSB будет автоматически очищен, если инициализирована операция загрузки страницы.

Бит 5: Res – резервированный бит

Бит 4: RWWSRE – чтение RWW секции разрешено

Когда программируется RWW секция, она блокирована для чтения. Для получения доступа к этой секции пользователь должен подождать окончания программирования. Таким образом, если RWWSRE бит записывается логической единицей в тоже время, когда и SPMEN, следующая SPM инструкция, проведенная в течение четырех циклов, разрешит доступ к RWW секции. RWW секция не может быть доступна, в то время пока Flash занята процедурой страничного стирания или записи. Если RWWSRE бит записан, в то время как Flash-память программируется, операция загрузки Flash-памяти будет игнорироваться и данные будут утеряны.

Бит 3: BLBSET – установка бит блокировки загрузчика

Если в этот бит одновременно с битом SPMEN записывается логическая единица, SPM инструкция, проведенная в следующие за установкой четыре такта, проведет установку бит блокировки загрузчика в соответствии с данными в регистре R0. Данные в R1 и адрес в Z-указателе будут игнорированы. BLBSET бит будет очищен до тех пор, пока установка бит блокировки загрузчика не будет завершена или если SPM инструкция не будет проведена в следующие за установкой SPMEN четыре цикла.

LPM инструкция, проведенная в течение трех циклов после установки BLBSET и SPMEN, будет читать либо биты блокировки, либо конфигурационные биты (в зависимости от Z0 в Z-указателе) в регистр назначения.

Бит 2: PGWRT – запись страницы

Если в этот бит одновременно с битом SPMEN записывается логическая единица, SPM инструкция, проведенная в следующие за установкой четыре такта, проведет запись страницы данными, записанными во временной буфер. Адрес страницы располагается в старших битах Z-указателя. Данные в R0 и в R1 будут игнорированы. Бит PGWRT будет аппаратно очищен после завершения процедуры записи или в случае, если SPM инструкция не будет проведена в следующие за установкой SPMEN четыре цикла. ЦПУ будет остановлен во время процедуры записи страницы, если адресуется NRWW секция.

Бит 1: PGERS – стирание страницы

Если в этот бит одновременно с битом SPMEN записывается логическая единица, SPM инструкция, проведенная в следующие за установкой четыре такта, проведет стирание страницы. Адрес страницы располагается в старших битах Z указателя. Данные в R0 и в R1 будут игнорированы. Бит PGERS будет аппаратно очищен после завершения процедуры стирания или в случае, если SPM инструкция не будет проведена в следующие за установкой SPMEN четыре цикла. ЦПУ будет остановлен во время процедуры записи страницы, если адресуется NRWW секция.

Бит 0: SPMEN – разрешение SPM

Этот бит разрешает SPM инструкцию в течение следующих четырех циклов. Если этот бит устанавливается вместе с RWWSRE, BLBSET, PGWRT или PGERS, следующая команда SPM будет иметь специальное значение. Если устанавливается только SPMEN, следующая команда SPM будет сохранять значение (R1–R0) во временной страничный буфер, адресуемый Z-указателем. Младшие биты Z-указателя игнорируются. SPMEN бит будет автоматически очищен после завершения SPM инструкции или в случае, если SPM инструкция не будет проведена в следующие за установкой SPMEN четыре цикла. Во время процедур стирания или записи страницы бит SPMEN будет установлен до завершения операции.

Запись в младшие пять разрядов регистра значений, отличных от «10001», «01001», «00101», «00011» и «00001», не вызывает никакого эффекта.

Следует обратить внимание на то, что во время записи в EEPROM-память изменение регистра SPMCR невозможно. Поэтому перед тем как записать какое-либо значение в регистр SPMCR, рекомендуется дождаться сброса флага EEWE регистра EECR. Для адресации памяти программ при использовании команды SPM используется индексный регистр Z, получаемый объединением двух старших регистров общего назначения R30 (младший байт) и R31 (старший байт). Поскольку память программ имеет страничную организацию, счетчик команд можно условно разбить на две части. Первая часть (младщие разряды) адресует ячейку на странице, а вторая часть определяет страницу. После запуска операции программирования содержимое регистра Z фиксируется и его можно использовать для других целей.

3.19.2 Изменение памяти программ

Изменение памяти программ осуществляется в следующей последовательности:

1 Заполнение буфера страницы новым содержимым.

Индексный регистр Z используется для адресации памяти программ. Биты (1–5) отвечают за адресацию слова в буфере, а биты (6–12) – за адресацию страницы в памяти программ (см. рисунок 3.109). Данные кода операции должны находиться в регистрах (R1–R0).

Загружаем адрес слова на странице и адрес страницы в памяти программ в индексный регистр Z. Выполняем команду SPM с установленным битом SPMEN. Загружаем следующий адрес слова в регистр Z. Выполняем команду SPM с установленным битом SPMEN. Повторяем данные операции до заполнения буфера.

2 Перенос содержимого буфера в память программ.

После загрузки последнего слова в буфер страницы выполняем команду SPM с установленными одновременно битами SPMEN и одним из битов PGWRT или PGERS, в зависимости от того, что необходимо сделать: либо записать, либо стереть данные на странице, при этом адрес страницы в памяти программ должен быть занесен в регистр Z (секция PCPAGE), а адрес и данные последнего занесенного слова не должны изменяться.

Следует заметить, что при записи данных в память программ необходимость в стирании страницы отсутствует, поэтому процедуру стирания страницы перед записью данных можно пропустить.

При стирании страницы памяти программ содержимое регистров R1 и R0 игнорируется. При записи страницы или слова в память программ в регистрах R1 и R0 содержится код операции.

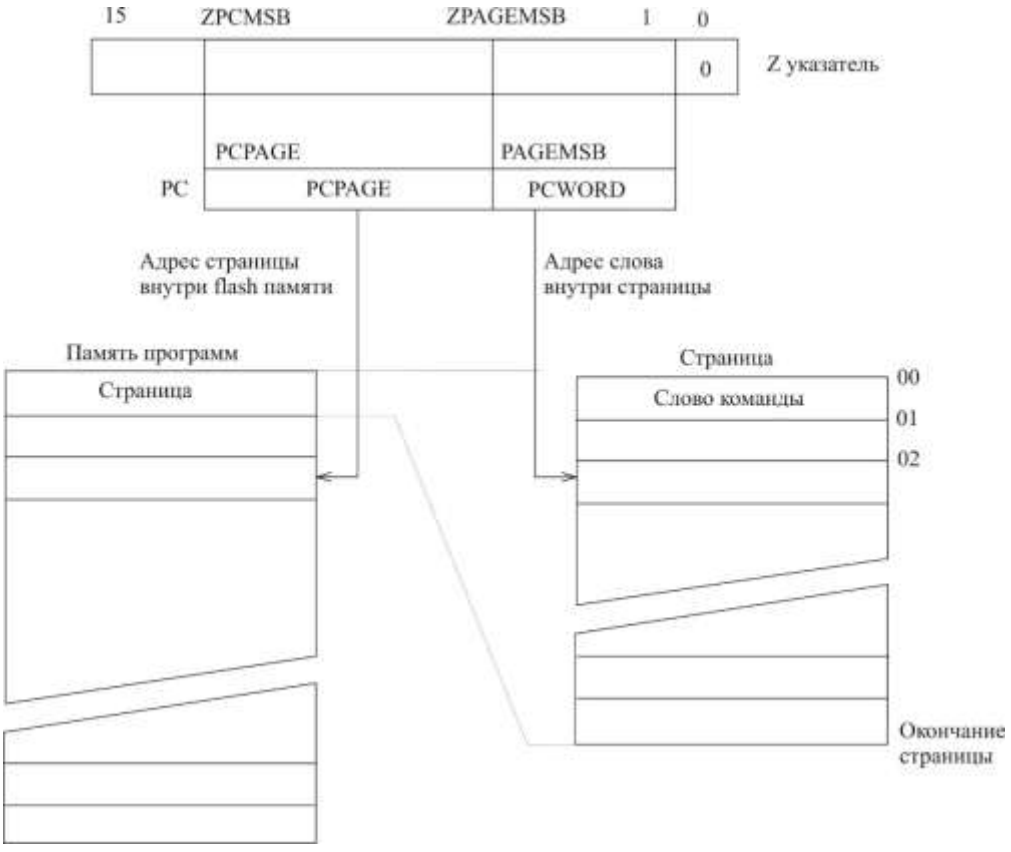


Рисунок 3.109 – Адресация к памяти программ через Z-указатель

Пример кода на Ассемблере

Запись страницы памяти программ данными из ОЗУ

```
;*****************************************
ldi     XH,high(RAMSTART)              ;Для адресации ОЗУ используем указатель
X
ldi     XL,low(RAMSTART)
clr     ZH                             ;указатель flash
clr     ZL
clr     count                          ;счетчик записанных слов
ldi     temp1,PROG_SIZE        ;размер программы в словах

save_rww:
        clr     looplo                 ;счетчик слов на странице

;передаем данные из ОЗУ в буфер страницы

load_buffer:
        ld      r0,X+
        ld      r1,X+
        ldi     spmcrval,(1<<SPMEN)
        rcall   Do_spm
        inc     count
        inc     looplo
        adiw    ZH:ZL,2
        cpi     looplo,$20
        brne    load_buffer
        sbiw    ZH:ZL,2

;записываем страницу

        ldi     spmcrval,(1<<PGWRT)|(1<<SPMEN)
        rcall   Do_spm
        adiw    ZH:ZL,2
        cp      count,temp1
        brlo    save_rww
;*************************************
```

Запись байта данных в EEPROM

```
;******************************
write_byte:
;ждем завершения записи в EEPROM
        sbic            EECR,EEWE
        rjmp            write_byte
;загружаем записываемые данные
        out             EEDR,Data
;установка адреса
        out             EEARH,XH
        out             EEARL,XL
;запись данных
        sbi             EECR,EEMWE
        sbi             EECR,EEWE
        ret
```

Чтение байта данных в EEPROM

```
;*****************************
read_byte:
;ждем завершения записи в EEPROM
        sbic            EECR,EEWE
        rjmp            read_byte
;установка адреса
        out             EEARH,XH
        out             EEARL,XL
;чтение данных
        sbi             EECR,EERE
        in              Data,EEDR
        ret
```

Выполнение команды SPM

```
;***************************************
Do_spm:
; ждем завершения предыдущей команды SPM
Wait_spm:
in      temp1, SPMCR
sbrc    temp1, SPMEN
rjmp    Wait_spm
; регистр spmcrval определяет действие команды SPM
; сохранение статуса, запрет прерываний
in       temp2, SREG
cli
; ждем завершения записи в EEPROM
Wait_ee:
sbic    EECR, EEWE
rjmp    Wait_ee
; выполнение SPM
out     SPMCR, spmcrval
spm
; восстановление регистра SREG (разрешение прерываний, если были разрешены)
out      SREG, temp2
ret
```

Выполнение стирания данных в странице с помощью команды SPM

Для выполнения стирания данных в странице установите адрес в Z-указателе, 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, что невозможно записывать данные более чем один раз по каждому адресу, если не стирать информацию во временном буфере.

Примечание – Если запись в ЭСППЗУ 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, 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 сегмент BLS, чтобы избежать возможности получения доступа прерыванием к сегменту RWW, когда этот сегмент блокирован для чтения. Подробности относительно перемещения прерываний см. в подразделе 3.7 «Прерывания».

Учет некоторых особенностей при обновлении сегмента программы начальной загрузки

Необходимо быть особенно осторожным, если пользователь разрешает обновление сегмента программы начальной загрузки, оставив бит BLB11 незапрограммированным.
Случайная запись в сам начальный загрузчик может повредить всю программу начальной загрузки, а дальнейшие обновления программ могут стать невозможными. Если нет необходимости изменять саму программу начальной загрузки, рекомендуется программировать бит BLB11, чтобы защитить начальный загрузчик от любых внутренних программных изменений.

Предотвращение чтения сегмента RWW во время самопрограммирования

Во время самопрограммирования (либо при стирании, либо при записи данных в страницу) сегмент RWW всегда блокирован для чтения. Пользовательская программа сама должна предотвратить возможность адресации данной области в течение операции самопрограммирования. Бит RWWSB в регистре SPMCSR будет установлен, пока сегмент RWW занят. Во время самопрограммирования таблицу векторов прерываний следует переместить в сегмент BLS, как описано в 3.7 «Прерывания», либо прерывания должны быть запрещены. Перед адресацией сегмента RWW после завершения программирования пользовательская программа должна сбросить бит RWWSB путем записи бита RWWSRE. См. ниже «Пример простого ассемблированного кода для программы начальной загрузки».

Установка битов защиты начального загрузчика с помощью команды SPM

Для установки битов защиты начального загрузчика внесите желаемые данные в регистр R0, запишите комбинацию «X0001001» в SPMCR и выполните команду SPM в течение четырех тактов после записи в регистр SPMCR. Единственными доступными битами защиты являются биты защиты загрузки BLB, которые могут предохранить сегмент приложения и сегмент начального загрузчика от любого обновления программ через микроконтроллер.

Бит	7	6	5	4	3	2	1	0
R0	1	1	BLB12	BLB11	BLB02	BLB01	1	1

См. таблицы 3.91 – 3.94, чтобы изучить влияние различных установок битов защиты на доступ к флэш-памяти.

Если какие-либо биты (5–2) в регистре R0 сброшены, то соответствующие биты защиты будут запрограммированы, когда выполнится команда SPM в течение четырех тактов после установки битов BLBSET и SPMEN в SPMCR. 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 флэш-памяти доступно для чтения во время данной операции.

Предотвращение записи в регистр SPMCR путем записи в ЭСППЗУ

Обратите внимание, что операция записи в ЭСППЗУ блокирует программирование флэш-памяти. Чтение конфигурационных битов и битов защиты из программы также не будет разрешено во время операции записи в ЭСППЗУ. Пользователю рекомендуется проверять бит статуса (EEWE) в регистре EECR и контролировать, что этот бит сбрасывается перед выполнением операции записи в регистр SPMCR.

Чтение конфигурационных битов и битов защиты из программы

Имеется возможность считывать 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 разделе описания инстркуций.

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 SPMCR. 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 3.88 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 3.87 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

Запрограммированные конфигурационные биты и биты защиты будут считываться как ноль. Незапрограммированные конфигурационные биты и биты защиты будут считываться как единица.

Предотвращение повреждения флэш-памяти

В периоды низкого напряжения #VCC программа во флэш-памяти может быть повреждена, так как напряжение питания слишком низкое для того, чтобы ЦПУ и флэш-память могли работать должным образом. В данном случае возникают точно такие же проблемы, что и в системах на уровне плат, которые используют флэш-память, и поэтому следует применять те же самые проектные решения.

Повреждение программы во флэш-памяти может быть вызвано двумя ситуациями, когда напряжение слишком низкое. Во-первых, правильная последовательность записи во флэш-память требует наличия определенного минимально необходимого напряжения для корректной работы. Во-вторых, само ЦПУ может выполнять команды неправильно, если напряжение питания слишком низкое для выполнения инструкций.

Повреждения флэш-памяти 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 программы начальной загрузки.

2 Удерживайте сигнал сброса микроконтроллера активным (на низком уровне) в периоды недостаточного напряжения питания. Это можно сделать путем включения внутреннего детектора питания BOD, если рабочее напряжение совпадает с уровнем детектирования. Если нет совпадения, то можно использовать внешнюю схему защиты сбросом при низком напряжении #VCC. Если сброс происходит во время выполнения операции записи, то данная операция будет завершена, при условии что напряжение питания будет достаточным.

3 Поддерживайте ядро микроконтроллера в режиме микропотребления в периоды низкого напряжения #VCC. Это предотвратит ЦПУ от попытки декодирования и выполнения инструкций, эффективно защищая регистр SPMCSR и, соответственно, флэш-память от непреднамеренной записи.

Время программирования флэш-памяти при использовании команды SPM

Калиброванный RC-генератор применяется для измерения времени доступа к флэш-памяти. В таблице 3.90 показывается типичное время программирования для доступа к флэш-памяти из ЦПУ.

Таблица 3.90 – Время программирования при использовании команды SPM

Виды программирования флэш-памяти с помощью команды SPM	Минимальное время программирования, мс	Максимальное время программирования, мс
Запись флэш-памяти: стирание данных в странице, запись в страницу, запись битов защиты	4	8

3.20 Программирование памяти

ИС 1887ВЕ4У поддерживает следующие режимы программирования:

- последовательное программирование при низком напряжении (по интерфейсу SPI);
- параллельное программирование при высоком напряжении.

Под «высоким» напряжением здесь понимается управляющее напряжение 12 В, подаваемое на вывод RESET# микроконтроллера для перевода последнего в режим программирования.

3.20.1 Защита кода и данных

Содержимое Flash-памяти (памяти программ), а также содержимое EEPROM-памяти (память данных) может быть защищено от записи и/или чтения посредством программирования ячеек защиты (Lock Bits). Возможные режимы защиты, соответствующие различным состояниям этих ячеек, приведены в таблице 3.91.

Таблица 3.91 – Биты блокировки

Биты блокировки	Номер бита	Описание	Значение по умолчанию
	7	-	1 (не запрограммирован)
	6	-	1 (не запрограммирован)
BLB12	5	Бит блокировки загрузчика	1 (не запрограммирован)
BLB11	4	Бит блокировки загрузчика	1 (не запрограммирован)
BLB02	3	Бит блокировки загрузчика	1 (не запрограммирован)
BLB01	2	Бит блокировки загрузчика	1 (не запрограммирован)
LB2	1	Бит блокировки	1 (не запрограммирован)
LB1	0	Бит блокировки	1 (не запрограммирован)

Таблица 3.92 – Режимы защиты бит блокировки LB

LB режим	Биты защиты памяти		Тип защиты
	LB2	LB1	
1	1	1	Память доступна для чтения и записи
2	1	0	Дальнейшее программирование Flash и EEPROM, бит конфигурации запрещено в параллельном и последовательном режиме
3	0	0	Дальнейшее программирование и верификация Flash и EEPROM, бит конфигурации запрещено в параллельном и последовательном режиме

Таблица 3.93 – Режимы защиты бит блокировки BLB0

BLB0 режим	Биты защиты памяти		Тип защиты
	BLB02	BLB01	
1	1	1	Нет ограничений для SPM и LPM доступа секции приложений
2	1	0	SPM не разрешена для записи в секцию приложений
3	0	0	SPM не разрешена для записи в секцию приложений и LPM, выполняемая из секции загрузчика, не разрешена для чтения из области приложений. Если векторы прерываний расположены в области загрузчика, они запрещены во время выполнения программы из области приложений
4	0	1	LPM, выполняемая из области загрузчика, не может производить чтение из области приложений. Если векторы прерываний расположены в области загрузчика, они запрещены во время выполнения программы из области приложений

Таблица 3.94 – Режимы защиты бит блокировки BLB1

BLB1 режим	Биты защиты памяти		Тип защиты
	BLB12	BLB11	
1	1	1	Нет ограничений для SPM и LPM доступа секции загрузчика
2	1	0	SPM не разрешена для записи в секцию загрузчика
3	0	0	SPM не разрешена для записи в секцию загрузчика и LPM, выполняемая из секции приложений, не разрешена для чтения из области загрузчика. Если векторы прерываний расположены в области приложений, они запрещены во время выполнения программы из области загрузчика
4	0	1	LPM, выполняемая из секции приложений, не разрешена для чтения из области загрузчика. Если векторы прерываний расположены в области приложений, они запрещены во время выполнения программы из области загрузчика

При использовании параллельного режима программирования в режимах 2 и 3 запрещается также изменение конфигурационных ячеек. Поэтому включение защиты следует выполнять в самую последнюю очередь, после программирования остальных областей памяти микроконтроллера.

В исходном (незапрограммированном) состоянии в этих ячейках содержится «1», после программирования – «0».

3.20.2 Конфигурационные ячейки

Таблица 3.95 – Старший конфигурационный байт

Старший конфигурационный байт	Номер бита	Описание	Значение по умолчанию
S8535C	7	Выбор режима совместимости с AT90S8535	1 (не запрограммирован)
WDTON	6	WDR всегда включен	1 (не запрограммирован)
SPIEN	5	Разрешение последовательного программирования и загрузки данных	0 (запрограммирован) SPI программирование разрешено
CKOPT [1)]	4	Опции осциллятора	1 (не запрограммирован)
EESAVE	3	EEPROM не очищается процедурой ChipErase	1 (не запрограммирован)
BOOTSZ1	2	Выбор размера массива области загрузчика	0 (запрограммирован)[2)]
BOOTSZ2	1	Выбор размера массива области загрузчика	0 (запрограммирован) [2)]
BOOTRST	0	Выбор вектора сброса	1 (не запрограммирован)

[1)] CKOPT бит функционально зависит от установки CKSEL бит.

[2)] Значение по умолчанию для битов BOOTSZ1, 2 выбирает максимальный размер массива области загрузчика.

Таблица 3.96 – Младший конфигурационный байт

Младший конфигурационный байт	Номер бита	Описание	Значение по умолчанию
BODLEVEL	7	Пороговый уровень BOD	1 (не запрограммирован)
BODEN	6	Разрешение работы BOD	1 (не запрограммирован)
SUT1	5	Выбор времени старта	1 (не запрограммирован)
SUT0	4	Выбор времени старта	0 (запрограммирован)
CKSEL3	3	Выбор источника тактирования	0 (запрограммирован)
CKSEL2	2	Выбор источника тактирования	0 (запрограммирован)
CKSEL1	1	Выбор источника тактирования	0 (запрограммирован)
CKSEL0	0	Выбор источника тактирования	1 (не запрограммирован)

Примечания

1 Значение по умолчанию SUT1, 0 выбирает максимальное время старта.

2 Значение по умолчанию CKSEL(3–0) выбирает внутренний RC генератор с частотой, равной 1 МГц.

Как следует из названия, конфигурационные ячейки (Fuse Bits) определяют ряд параметров конфигурации микроконтроллера. Эти ячейки расположены в отдельном адресном пространстве, доступном только при программировании.

Программирование бита S8535C будет изменять следующую функциональность:

1 Временная последовательность для изменения периода сторожевого таймера запрещена.

2 Двойная буферизация приемного регистра USART запрещена.

3.20.3 Команда Chip Erase

Выполнение команды Chip Erase непосредственно перед записью данных не обязательно. Данная команда необходима в случае использования битов защиты микроконтроллера, т. к. снятие битов защиты возможно только командой Chip Erase с одновременным удалением данных из памяти микроконтроллера.

После команды Chip Erase ячейки памяти программ и данных содержат значение 0x00.

3.20.4 Параллельное программирование

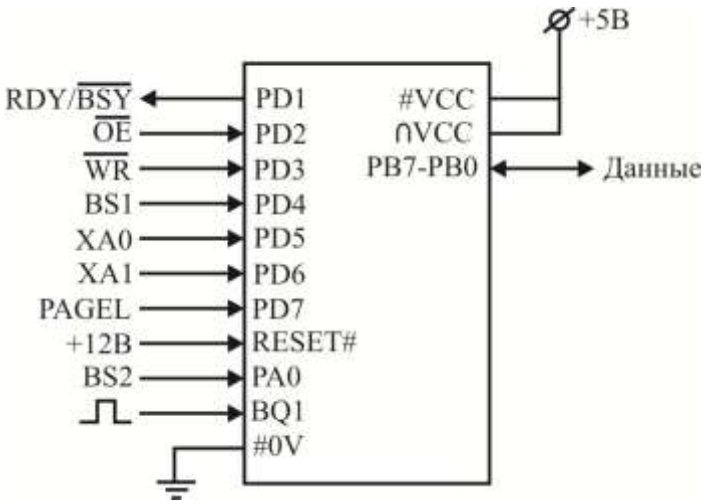


Рисунок 3.110 – Включение схемы при параллельном программировании

Таблица 3.97 – Управляющие сигналы программирования

Наименование сигналов	Имя вывода	Направление (I –вход, O –выход)	Функция
RDY/BSY#	PD1	O	0: Программирование 1: Устройство готово для новой команды
OE#	PD2	I	Чтение разрешено
WR#	PD3	I	Запись разрешена
BS1	PD4	I	Выбор байта 1: 0 – младший, 1 – старший
XA0	PD5	I	Бит действия 0 BQ
XA1	PD6	I	Бит действия 1 BQ
PAGEL	PD7	I	Бит страничной загрузки Flash и EEPROM
BS2	PA0	I	Выбор байта 2: 0 – младший, 1 – старший
DATA	PB7–PB0	I/O	Двунаправленная шина данных

Таблица 3.98 – Значения выводов при входе в режим программирования

Вывод	Символ	Значение
PAGEL	Разрешение программирования (3)	0
XA1	Разрешение программирования (2)	0
XA0	Разрешение программирования (1)	0
BS1	Разрешение программирования (0)	0

Таблица 3.99 – Кодирование XA1 и XA0

XA1	XA0	Действие, производимое на импульс BQ
0	0	Загрузка Flash и EEPROM адреса
0	1	Загрузка данных
1	0	Загрузка инструкции
1	1	Нет действия

Таблица 3.100 – Коды команд байта инструкций программирования

Байт команды	Выполняемое действие
1000 0000	Стирание памяти
0100 0000	Запись битов конфигурации
0010 0000	Запись битов блокировки
0001 0000	Запись Flash
0001 0001	Запись EEPROM
0000 0100	Чтение конфигурационных и блокировочных битов
0000 0010	Чтение Flash
0000 0011	Чтение EEPROM

Таблица 3.101 – Количество слов в странице и количество страниц в массиве Flash

Размер массива Flash	Размер страницы	PCWORD	Количество страниц	Количествово страниц PC	PCMSB
4К слов	32 слова	PC(4–0)	128	PC(11–5)	11

Таблица 3.102 – Количество слов в странице и количество страниц в массиве EEPROM

Размер массива EEPROM	Размер страницы	PCWORD	Коллчествово страниц	Количествово страниц PC	EEAMSB
1 Кбайт	4 байта	EEA(1–0)	256	PC(8–2)	8

Вход в режим программирования

Для входа в режим параллельного программирования выполняется следующий алгоритм:

1 Подать напряжение питания (5 В) и ожидать не менее 100 мкс.

2 Установить RESET# в низкое логическое состояние и переключить BQ не менее шести раз.

3 Установить выводы разрешения программирования в значение «0000» и ожидать не менее 100 нс.

4 Подать напряжение (11,5–12,5) В на RESET#. Любое действие на выводах разрешения программирования в течение 100 нс после подачи напряжения на RESET# приведет к сбою входа в режим программирования.

Следует заметить, если тактирование схемы сконфигурировано для работы от внешнего кварца или внешней RC цепочки, это может привести к невозможности работы с BQ. Для избежания этого должен быть выполнен следующий алгоритм:

1 Установить выводы разрешения программирования, указанные в таблице 3.98, в значение «0000».

2 Подать напряжение питания, подключить общий вывод одновременно с подачей (11,5–12,5) В на вывод RESET#.

3 Подождать 100 нс.

4 Перепрограммировать конфигурационные биты для тактирования схемы от внешнего тактового генератора (CKSEL(3–0) = 0b0000). Если биты блокировки запрограммированы, процедура Chip Erase (стирание памяти) должна предшествовать программированию конфигурационных битов.

5 Выйти из режима программирования, выполнив либо отключение питания, либо подачу низкого логического уровня на вывод RESET#.

6 Войти в режим программирования, используя первоначальный алгоритм, описанный выше.

Условия эффективного программирования

Загруженные команда и адрес остаются в устройстве в течение цикла программирования. Для эффективного программирования должны быть учтены следующие моменты:

- команду необходимо загружать только один раз при проведении неоднократных процедур записи/чтения;
- старший байт адреса должен быть загружен только при программировании или чтении новых 256 слов в массив Flash или новых 256 байт в массив EEPROM.

Стирание памяти (Chip Erase)

Процедура Chip Erase будет стирать целиком массивы Flash и EEPROM плюс биты блокировки. Биты блокировки не сбрасывают своих значений до тех пор, пока память программ полностью не стерта. Конфигурационные биты не изменяют своих значений. Процедура Chip Erase проводится перед программированием Flash и/или EEPROM.

Необходимо заметить, что процедура Chip Erase не производит стирания массива EEPROM в случае, если конфигурационный бит EESAVE запрограммирован.

Загрузка команды Chip Erase:

1 Установить XA1 и XA0 в значение «10». Это разрешит загрузку команды.

2 Установить BS1 в «0».

3 Установить данные в «1000 0000». Это команда для Chip Erase.

4 Подать на BQ1 положительный импульс. Это приведет к загрузке команды.

5 Подать отрицательный импульс на WR# – начало процедуры Chip Erase. RDY/BSY# перейдет в низкое логическое состояние.

6 Дождаться перехода RDY/BSY# в высокое логическое состояние перед подачей новой команды.

Программирование Flash

Массив Flash организован постранично. Когда программируется Flash-память, программируемые данные защелкиваются в буфер страницы. Это позволяет программировать страницу данных одновременно. Следующая процедура описывает шаги программирования массива Flash.

1 Загрузка команды «Write Flash»:

- установить XA1, XA0 в значение «10». Это разрешит загрузку команды «Write Flash»;
- установить BS0 в значение «0»;
- установить вход данных в «0001 0000». Это код команды Write Flash;
- подать положительный импульс на BQ1. Это приведет к загрузке команды.

2 Загрузка младшего байта адреса:

- установить XA1, XA0 в значение «00». Это разрешит загрузку адреса;

- установить BS0 в значение «0»;

- установить вход данных в значение младшего байта адреса;

- подать положительный импульс на BQ1. Это приведет к загрузке младшего байта адреса.

3 Загрузка младшего байта данных:

- установить XA1, XA0 в значение «01». Это разрешит загрузку данных;

- установить вход данных в значение младшего байта данных;

- подать положительный импульс на BQ1. Это приведет к загрузке младшего байта данных.

4 Загрузка старшего байта данных:

- установить BS0 в значение «1». Это значение выбирает старший байт данных;

- установить XA1, XA0 в значение «01». Это разрешит загрузку данных;

- установить вход данных в значение старшего байта данных;

- подать положительный импульс на BQ1. Это приведет к загрузке старшего байта данных.

5 Защелкивание данных:

- установить BS0 в значение «1». Это значение выбирает старший байт данных;

- подать на вход PAGEL положительный импульс. Это приведет к защелкиванию байта данных.

6 Повторить шаги со второго по пятый до тех пор, пока весь буфер не заполнится или до тех пор, пока все данные не будут загружены в страницу.

В то время как младшие биты адреса обращаются к слову внутри страницы, старшие биты обращаются к странице внутри массива Flash. Следует отметить, что если менее чем 8 бит требуется для адресации слова в странице, то наиболее значащие биты в младшем байте адреса использованы для адресации страницы, когда представлена страничная запись.

7 Загрузка старшего байта адреса:

- установить XA1, XA0 в значение «00». Это разрешит загрузку адреса;

- установить BS0 в значение «1»;

- установить вход данных в значение старшего байта адреса;

- подать положительный импульс на BQ1. Это приведет к загрузке старшего байта адреса.

8 Программирование страницы:

- установить BS0 в значение «0»;

- подать отрицательный импульс на WR# – начало процедуры записи. RDY/BSY# перейдет в низкое логическое состояние;

- дождаться перехода RDY/BSY# в высокое логическое состояние перед подачей новой команды.

9 Повторить шаги со второго по восьмой до тех пор, пока весь массив или все данные не будут запрограммированы.

10 Завершение страничного программирования:

- установить XA1, XA0 в значение «10». Это разрешит загрузку команды;

- установить вход данных в «0000 0000». Это код «холостой» команды;

- подать положительный импульс на BQ1. Это приведет к загрузке команды и внутренний сигнал записи будет сброшен.

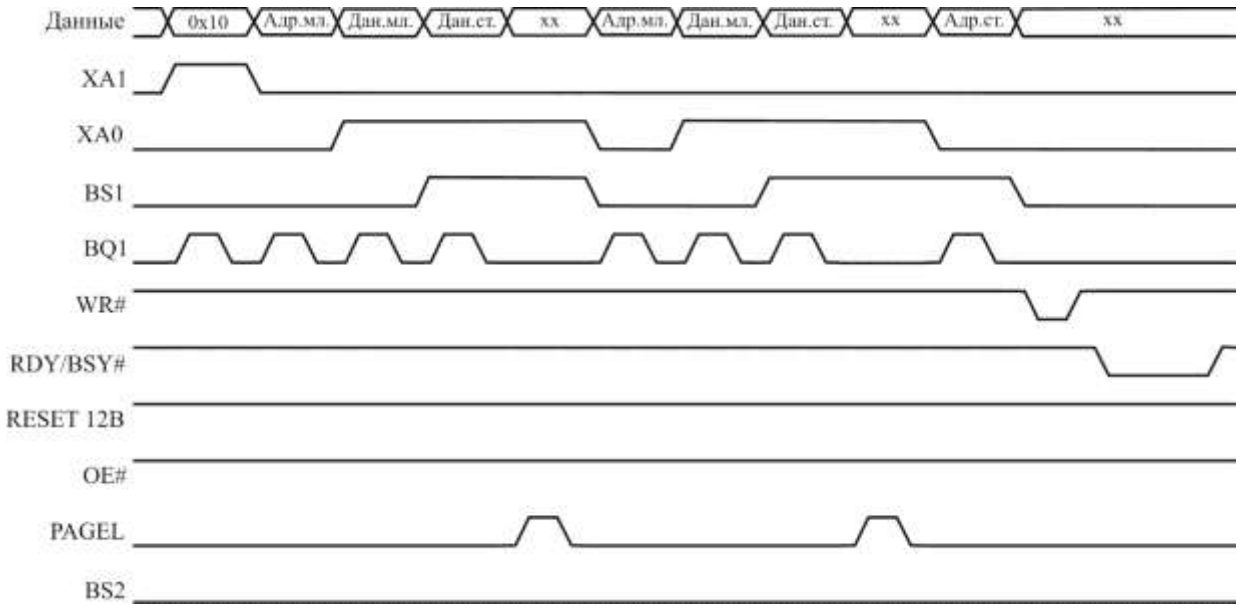


Рисунок 3.111 – Временная диаграмма программирования Flash-памяти

Программирование EEPROM

Память данных EEPROM организована в страничном режиме. Когда программируется массив EEPROM, данные защелкиваются в страничном буфере. Это позволяет программировать страницу данных одновременно. Ниже приведен алгоритм загрузки страницы EEPROM.

1 Загрузить команду «0001 0001».

2 Загрузить старший байт адреса.

3 Загрузить младший байт адреса.

4 Загрузить данные.

5 Выполнить защелкивание данных (подать положительный импульс PAGEL).

6 Повторить шаги с 3 по 5 до тех пор, пока вся страница не будет заполнена.

Для программирования массива EEPROM необходимо выполнить следующие шаги:

1 Установить BS0 в значение «1».

2 Подать отрицательный импульс на WR# – начало процедуры записи. RDY/BSY# перейдет в низкое логическое состояние.

3 Дождаться перехода RDY/BSY# в высокое логическое состояние перед подачей новой команды.

Диаграмма программирования массива EEPROM приведена на рисунке 3.112.

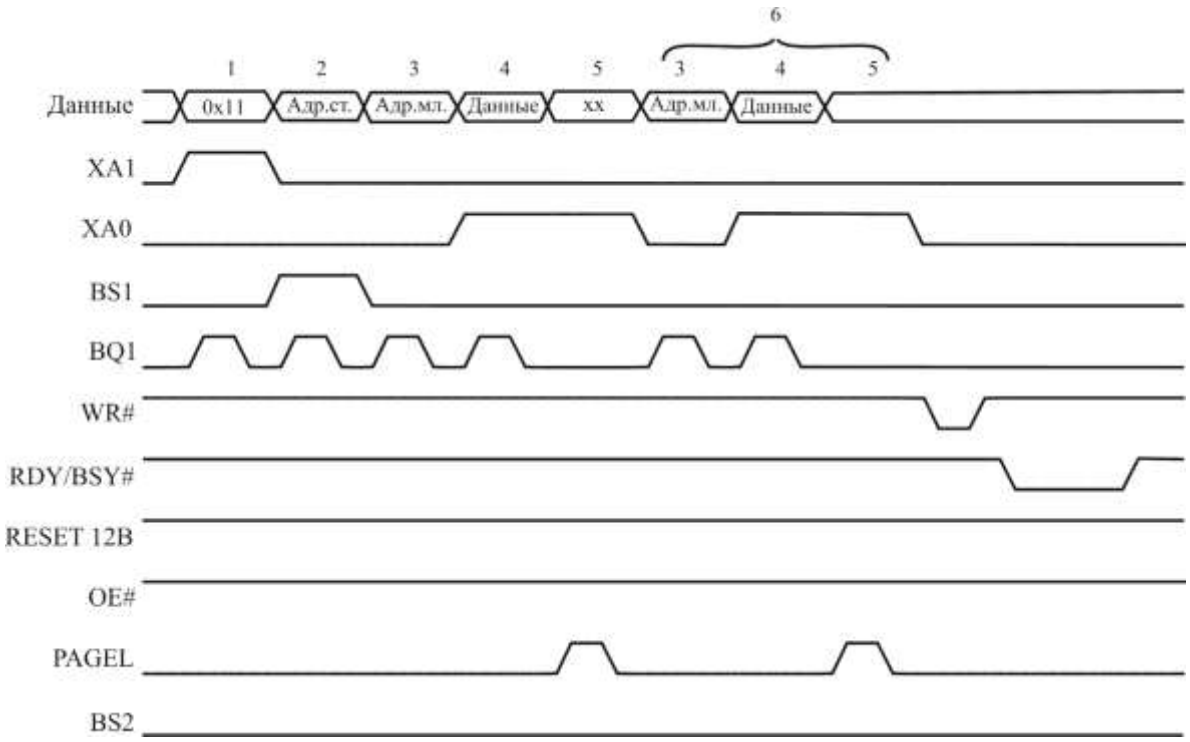


Рисунок 3.112 – Временная диаграмма программирования массива EEPROM

Чтение Flash-памяти

1 Загрузить команду «0000 0010».

2 Загрузить старший байт адреса.

3 Загрузить младший байт адреса.

4 Установить OE# в значение «0» и BS1 в значение «0». Младший байт слова Flash-памяти может быть прочитан на выводе DATA.

5 Установить BS1 в значение «1». Старший байт слова Flash памяти может быть прочитан на выводе DATA.

6 Установить OE# в значение «1».

Чтение массива EEPROM

1 Загрузить команду «0001 0011».

2 Загрузить старший байт адреса.

3 Загрузить младший байт адреса.

4 Установить OE# в значение «0» и BS1 в значение «0». Байт данных EEPROM-памяти может быть прочитан на выводе DATA.

5 Установить OE# в значение «1».

Программирование младшего конфигурационного байта

1 Загрузить команды «0100 0000» (см. рисунок 3.113).

2 Загрузить младший байт данных. Бит n = 0 программирует, а бит n = 1 стирает конфигурационный бит.

3 Установить BS1 в значение «0» и BS2 в значение «0». Это выберет младший байт данных.

4 Подать отрицательный импульс на WR# – начало процедуры записи – и ожидать, пока сигнал RDY/BSY# перейдет в высокое логическое состояние.

Программирование старшего конфигурационного байта

1 Загрузить команду «0100 0000» (см. рисунок 3.113).

2 Загрузить старший байт данных. Бит n = 0 программирует, а бит n = 1 стирает конфигурационный бит.

3 Установить BS1 в значение «1» и BS2 в значение «0». Это выберет старший байт данных.

4 Подать отрицательный импульс на WR# – начало процедуры записи – и ожидать, пока сигнал RDY/BSY# перейдет в высокое логическое состояние.

5 Установить BS1 в значение «0». Это выберет младший байт данных.

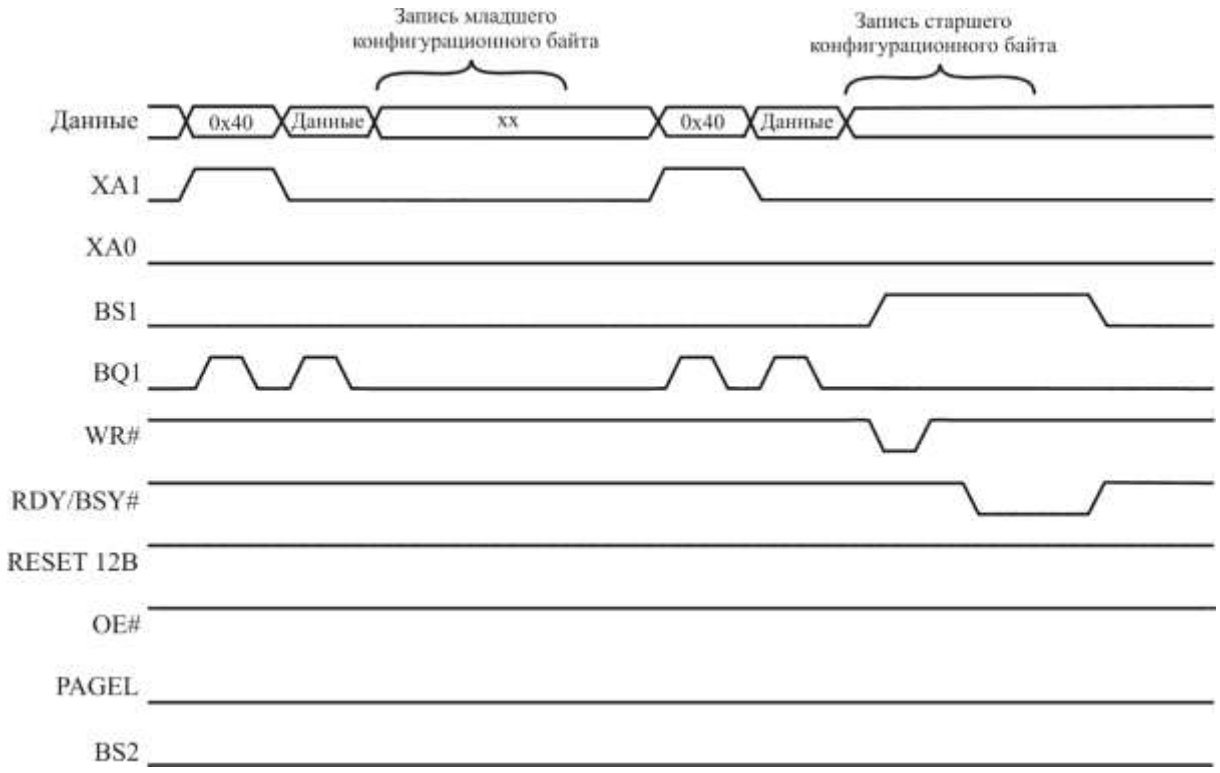


Рисунок 3.113 – Временная диаграмма программирования конфигурационных байтов

Программирование битов блокировки

1 Загрузить команду «0010 0000».

2 Загрузить младший байт данных. Бит n = 0 программирует бит блокировки.

3 Подать отрицательный импульс на WR# – начало процедуры записи, и ожидать, пока сигнал RDY/BSY# перейдет в высокое логическое состояние.

Чтение конфигурационных бит и бит блокировки

1 Загрузить команду «0000 0100» (см. рисунок 3.114).

2 Установить OE# в значение «0», BS2 в «0», BS1 в «0». Значения младших конфигурационных бит могут быть прочитаны на выводе DATA («0» означает запрограммированный бит).

3 Установить OE# в значение «0», BS2 в «1», BS1 в «1». Значения старших конфигурационных бит могут быть прочитаны на выводе DATA («0» означает запрограммированный бит).

4 Установить OE# в значение «0», BS2 в «0», BS1 в «1». Значения битов блокировки могут быть прочитаны на выводе DATA («0» означает запрограммированный бит).

5 Установить OE# в значение «1».

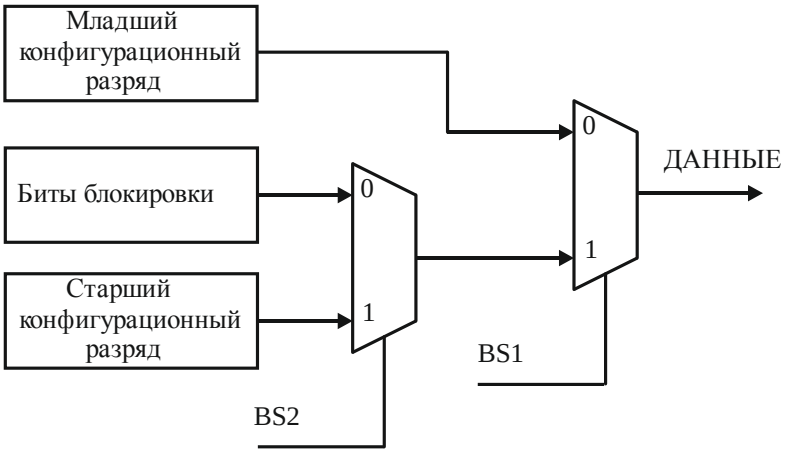


Рисунок 3.114 – Мультиплексирование конфигурационных битов и битов блокировки битами BS1 и BS2 при чтении на выводе «Данные»

Характеристики параллельного программирования

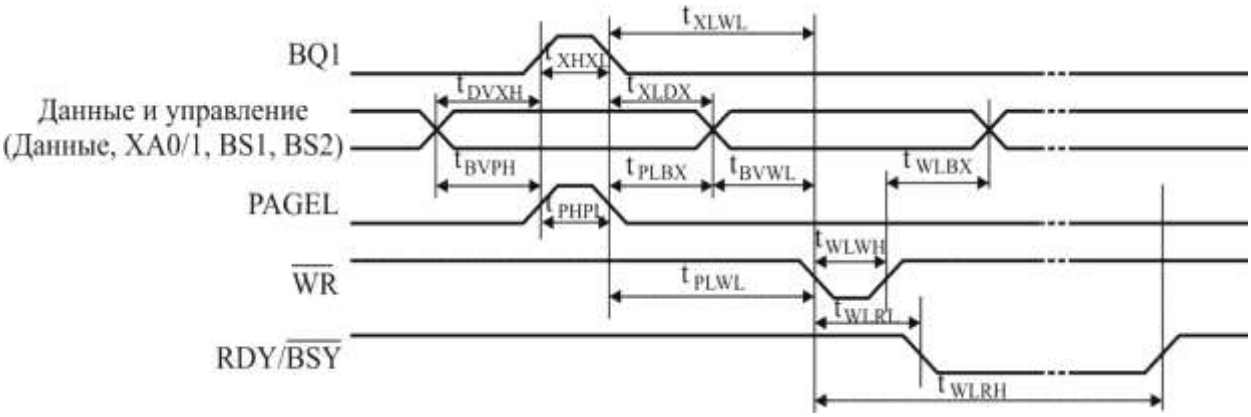


Рисунок 3.115 – Временная диаграмма управляющих сигналов при программировании, включая некоторые общие временные требования

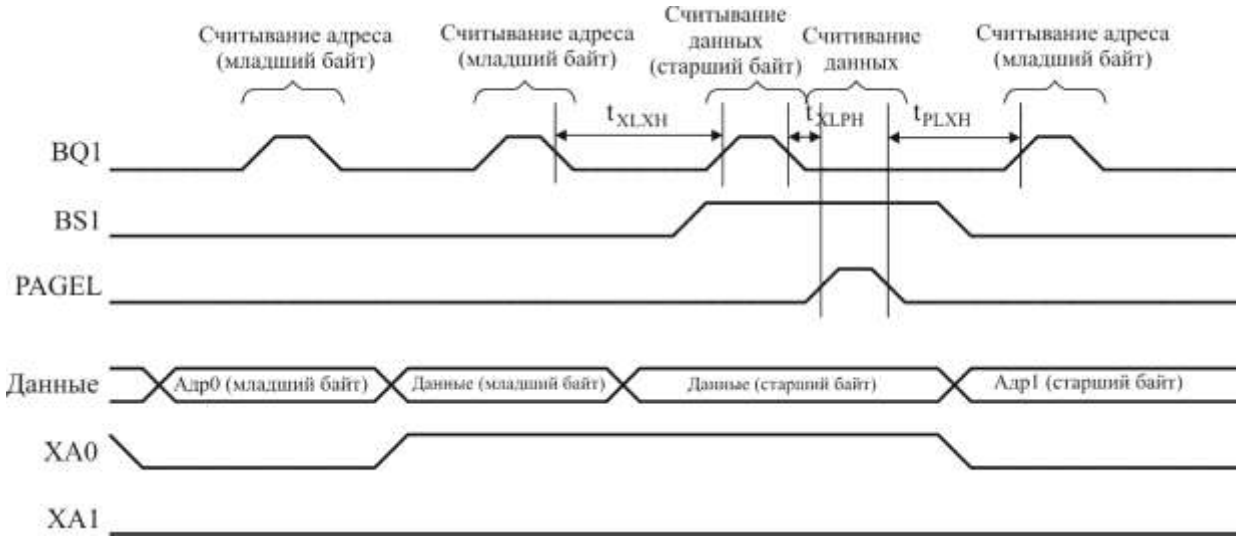


Рисунок 3.116 – Временные требования к последовательности загрузки

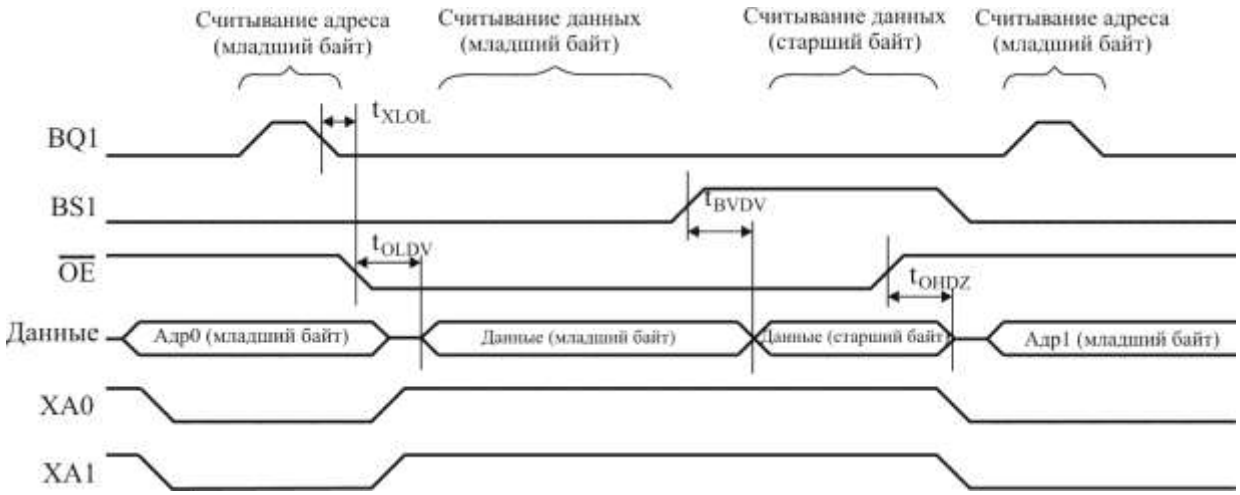


Рисунок 3.117 – Временные требования к последовательности чтения данных

Динамические характеристики, указанные на рисунках 3.115 – 3.117, представлены в таблице 3.103.

Таблица 3.103 – Динамические характеристики параллельного программирования

Символ	Параметр	Мин.	Тип.	Макс.	Ед. изм.
1	2	3	4	5	6
V_{PP}	Напряжение программирования	11,5	-	12,5	В
I_{PP}	Ток программирования		-	250	мкА
t_{DVXH}	Предустановка данных и управления к моменту перехода BQ1 в высокое логическое состояние	67	-	-	нс
t_{XLXH}	Интервал между двумя положительными фронтами BQ1	200	-	-	нс
t_{XHXL}	Ширина положительного импульса BQ1	150	-	-	нс
t_{XLDX}	Задержка данных и управления после перехода BQ1 в низкое логическое состояние	67	-	-	нс
t_{XLWL}	Интервал между переключением BQ1 в низкое логическое состояние и переключением WR# в низкое логическое состояние	0	-	-	нс
t_{XLPH}	Интервал между переключением BQ1 в низкое логическое состояние и переключением PAGEL в высокое логическое состояние	0	-	-	нс
t_{PLXH}	Интервал между переключением PAGEL в низкое логическое состояние и переключением BQ1 в высокое логическое состояние	150	-	-	нс
t_{BVPH}	Предустановка BS1 к моменту перехода PAGEL в высокое логическое состояние	67	-	-	нс
t_{PHPL}	Ширина положительного импульса PAGEL	150	-	-	нс
t_{PLBX}	Удержание BS1 после перехода PAGEL в низкое логическое состояние	67	-	-	нс
t_{WLBX}	Удержание BS2/1 после перехода WR# в низкое логическое состояние	67	-	-	нс
t_{PLWL}	Интервал между переключением PAGEL в низкое логическое состояние и переключением WR# в низкое логическое состояние	67	-	-	нс
t_{BVWL}	Предустановка BS1 к моменту перехода WR# в низкое логическое состояние	67	-	-	нс
t_{WLWH}	Ширина отрицательного импульса WR#	150	-	-	нс

Окончание таблицы 3.103

1	2	3	4	5	6
t_{WLRL}	Интервал между переключением WR# в низкое логическое состояние и переключением RDY/BSY# в низкое логическое состояние	0	-	1	мкс
t_{WLRH}	Интервал между переключением WR# в низкое логическое состояние и переключением RDY/BSY# в высокое логическое состояние [1)]	3,7	-	4,5	мс
t_{WLRH_CE} [2)]	Интервал между переключением WR# в низкое логическое состояние и переключением RDY/BSY# в высокое логическое состояние для процедуры Chip Erase [2)]	7,5	-	9	мс
t_{XLOL}	Интервал между переключением BQ1 в низкое логическое состояние и переключением OE# в низкое логическое состояние	0	-	-	нс
t_{BVDV}	Предустановка BS1 к моменту установки данных	0	-	250	нс
t_{OLDV}	Предустановка OE# в низкое логическое состояние к моменту установки данных	-	-	250	нс
t_{OHDZ}	Предустановка OE# в высокое логическое состояние к моменту переключения шины данных в состояние высокого импеданса	-	-	250	нс

[1)] t_{WLRH} распространяется на команды Write Flash, Write EEPROM, Write Fuse (запись бит конфигурации), Write Lock Bits (запись бит блокировки).

[2)] t_{WLRH_CE} используется для команды Chip Erase.

3.20.5 Программирование по последовательному каналу

Оба массива Flash и EEPROM могут быть запрограммированы, используя последовательный SPI интерфейс и подключение вывода RESET# к низкому логическому уровню (рисунок 3.118). Последовательный интерфейс включает выводы SCK, MOSI (вход) и MISO (выход). После переключения RESET# в низкое логическое состояние необходимо выполнить инструкцию Programming Enable перед проведением любых процедур программирования.

Таблица 3.104 – Назначение выводов при последовательном программировании

Символ	Вывод	Направление (I/O)	Описание
MOSI	PB5	I	Последовательный вход данных
MISO	PB6	O	Последовательный выход данных
SCK	PB7	I	Тактовый сигнал

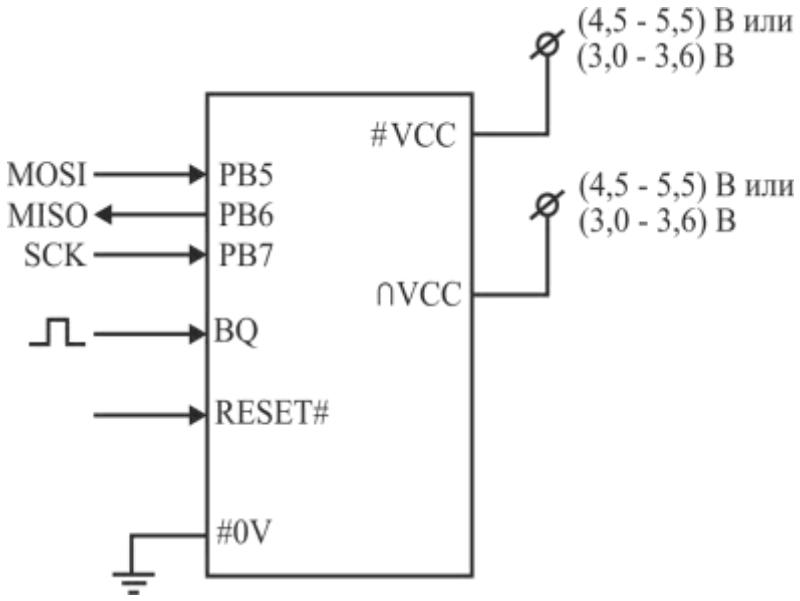


Рисунок 3.118 – Включение схемы при последовательном программировании и верификации

Примечания

1 Если микросхема тактируется внутренним осциллятором, нет необходимости подключать внешний источник тактовых импульсов к выводу BQ1.

2 ($U_{\#VCC}$ – 0,3) В < $U_{\cap VCC}$ < ($U_{\#VCC}$ + 0,3) В. Однако $U_{\cap VCC}$ должно всегда находиться внутри диапазона (4,5 – 5,5) В либо (3,0 – 3,6) В.

При программировании EEPROM процедура автостирания всегда следует перед проведением операции программирования, поэтому нет необходимости выполнения команды Chip Erase (только для последовательного программирования!).

В зависимости от конфигурации CKSEL должен быть представлен соответствующий тактовый сигнал. Минимальная длительность положительного и отрицательного полупериодов тактовых импульсов SCK должна соответствовать двум периодам тактового импульса ЦПУ.

Алгоритм последовательного программирования

При записи последовательных данных события происходят по возрастающему фронту SCK.

При чтении последовательных данных события происходят по спадающему фронту SCK.

Ниже представлена рекомендуемая последовательность шагов при проведении последовательного программирования и верификации памяти программ и данных.

1 Последовательность подачи напряжения питания: питание подается при наличии низкого потенциала на выводах RESET# и SCK.

2 Ожидать в течение 20 мс и затем подать команду Programming Enable для разрешения последовательного программирования.

3 Последовательное программирование не будет выполняться, если подключение не синхронизировано. При синхронизации второй байт (0x53) будет давать отклик в то время, когда будет загружаться третий байт команды Programming Enable. Корректен отклик или нет, все четыре байта инструкции должны быть переданы. Если второй байт (0x53) не вызвал отклика, следует подать положительный импульс RESET# и выполнить загрузку новой команды Programming Enable.

4 Flash-память программируется постранично. Страница памяти загружается побайтно подачей шести младших значимых бит адреса и данных вместе с командой Load Program Memory Page. Для уверенности корректной загрузки страницы, младший байт

данных должен быть загружен перед тем, как старший байт данных назначен для данного адреса. Страница памяти программ сохраняется при загрузке команды Write Program Memory Page и подаче восьми старших значимых бит адреса. Если опрос статуса программирования не производится, пользователь должен выждать время не менее t_{WD_FLASH} перед выдачей новой страницы данных. Доступ к последовательному интерфейсу программирования до того как завершится процедура записи во Flash, может привести к некорректному выполнению процесса программирования.

5 Массив EEPROM программируется побайтно назначением адреса и данных вместе с соответствующей командой Write. Ячейка памяти EEPROM предварительно автоматически стирается перед процедурой записи новых данных. Если опрос об окончании процедуры программирования не производится, пользователь должен выждать не менее t_{WD_EEPROM} перед выполнением программирования следующего байта.

6 Любая ячейка памяти может быть верифицирована, используя команду чтения Read, которая выдаст содержимое выбранного адреса на последовательный вывод MISO.

7 По завершении процесса программирования вывод RESET# должен быть установлен в высокое логическое состояние для начала нормального режима функционирования микросхемы.

8 Отключение питания (при необходимости):

- установить вывод RESET# в высокое логическое состояние;
- отключить напряжение питания от вывода #VCC.

Опрос данных Flash-памяти

Когда страница данных программируется во Flash-память, чтение адреса внутри страницы будет возвращать значение 0xFF. Когда устройство готово для загрузки следующей страницы, чтение будет возвращать корректные данные. Это используется для определения окончания процедуры записи страницы. Процедура опроса данных не работает для значения 0xFF, поэтому когда программируется это значение, пользователь должен выждать время не менее t_{WD_FLASH} перед тем, как программировать следующую страницу.

Опрос данных массива EEPROM

Когда программируется массив EEPROM, чтение программируемого в текущий момент байта будет возвращать значение 0xFF. Когда запись завершена, чтение будет возвращать корректные данные. Это используется для определения окончания процедуры записи страницы. Это не будет работать для значения 0xFF. В этом случае, опрос данных не может быть использован для значения 0xFF, и пользователь должен будет выждать не менее t_{WD_EEPROM} перед программированием следующего байта данных.

Таблица 3.105 – Минимальные времена задержки перед проведением следующего цикла программирования Flash и EEPROM памяти

Символ	Минимальное время задержки, мс
t_{WD_FLASH}	4,5
t_{WD_EEPROM}	9,0
t_{WD_ERASE}	9,0
t_{WD_FUSE}	4,5

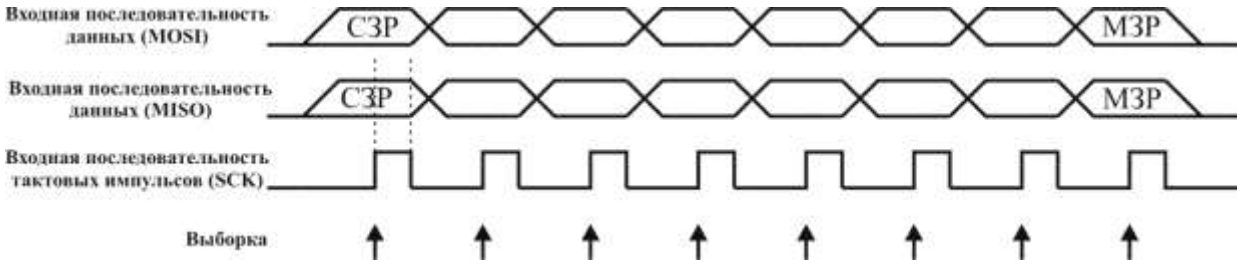


Рисунок 3.119 – Временная диаграмма выводов при последовательном программировании

Таблица 3.106 – Команды последовательного программирования

Команда	Формат инструкции				Операция
	Байт 1	Байт 2	Байт 3	Байт 4	
1	2	3	4	5	6
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Разрешение последовательного программирования после переключении, RESET# в «0»
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Стирание Flash и EEPROM массивов
Read program memory	0010 H000	0000 **aaaa**	**bbbb bbbb**	**oooo oooo**	Чтение H (старшего или младшего) байта данных памяти программ по адресу **a:b**
Load program memory	0100 H000	0000 xxxx	xxx**b bbbb**	**iiii iiii**	Запись H (старшего или младшего) байта данных **i** в страничный буфер по адресу **b**
Write program memory	0100 1100	0000 **aaaa**	**bbb**x xxxx	xxxx xxxx	Запись страницы памяти программ по адресу **a:b**
Read EEPROM memory	1010 0000	00xx xxx**a**	**bbbb bbbb**	**oooo oooo**	Чтение данных **i** из EEPROM памяти по адресу **a:b**
Write EEPROM memory	1100 0000	00xx xxx**a**	**bbbb bbbb**	**iiii iiii**	Запись данных **i** в память EEPROM по адресу **a:b**
Read Lock Bits	0101 1000	0000 0000	xxxx xxxx	xx**oo oooo**	Чтение бит блокировки: «0» – запрограммирован, «1» – незапрограммирован
Write Lock Bits	1010 1100	111x xxxx	xxxx xxxx	11**ii iiii**	Запись бит блокировки: «0» – программирование бита.
Write fuse bits	1010 1100	1010 0000	xxxx xxxx	**iiii iiii**	Запись младших конфигурационных бит. «0» – запрограммирован, «1» – незапрограммирован

Окончание таблицы 3.106

1	2	3	4	5	6
Write fuse high bits	1010 1100	1010 1000	xxxx xxxx	**iiii iiii**	Запись младших конфигурационных бит. «0» – запрограммирован, «1» – незапрограммирован
Read fuse bits	0101 0000	0000 0000	xxxx xxxx	**oooo oooo**	Чтение младших конфигурационных бит. «0» – запрограммирован, «1» – незапрограммирован
Read fuse high bits	0101 1000	0000 1000	xxxx xxxx	**oooo oooo**	Чтение младших конфигурационных бит. «0» – запрограммирован, «1» – незапрограммирован

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 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.

4.1.3 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.

4.1.4 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).

4.2 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, произво-

дится обращение по адресу, указанному в команде, относительно адреса, находящегося в индексном регистре.

Микроконтроллеры семейства Mega поддерживают 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 и с преддекрементом содержимое индексного регистра сначала уменьшается на единицу, а затем производится обращение по полученному адресу.

Микроконтроллеры семейства поддерживают 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».

Микроконтроллеры семейства поддерживают 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 1887ВЕ4У

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) кода 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 микроконтролера имеются команды как безусловного, так и условного переходов. Команды относительного перехода (RJMP), а также косвенного (IJMP) 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 SREG(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) 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 RCALL.

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 группу управления системой входят всего три команды:
- NOP – пустая команда;
- SLEEP – перевод микроконтроллера в режим пониженного энергопотребления;
- WDR – сброс сторожевого таймера.

Все команды этой группы выполняются за один машинный цикл.

6 Отладочные средства

6.1 Отладочное устройство КФДЛ.301411.243

Отладочное устройство КФДЛ.301411.243 для микроконтроллера 1887ВЕ4У предназначено для отладки и программирования систем, разрабатываемых на его основе, с использованием внутрисистемного программирования через последовательный интерфейс параллельного высоковольтного программирования.

Отладочное устройство, см. рисунок 6.1, представляет собой завершенный стартовый набор и систему для проектирования на основе микроконтроллера 1887ВЕ4У.

С его помощью разработчик сможет оперативно приступить к разработке программного кода и использовать широкие возможности стартового набора по макетированию и проверке новых решений.

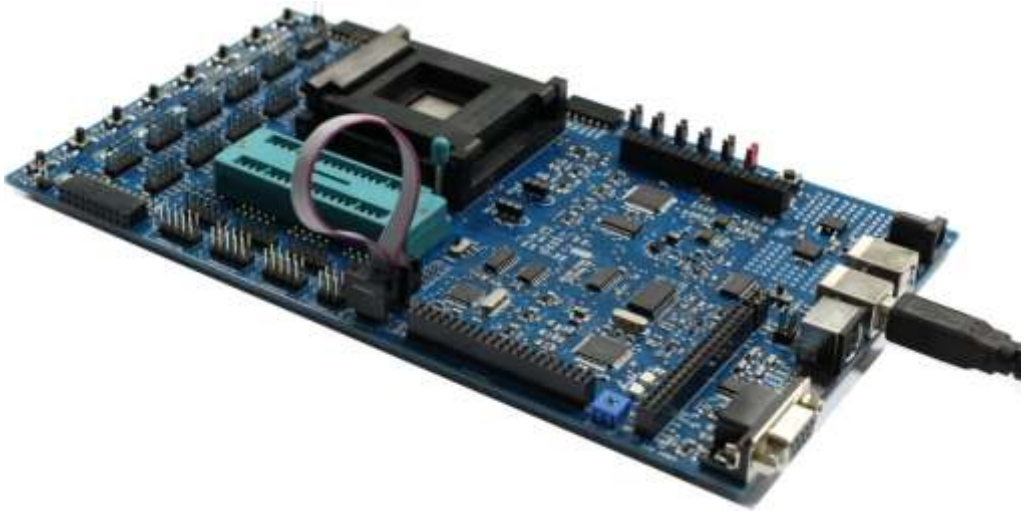


Рисунок 6.1 – Отладочное устройство КФДЛ.301411.243
для микроконтроллера 1887ВЕ4У

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 КФДЛ.301411.243 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».

Отладочная среда 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 1887ВЕ4У 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 1887ВЕ4У можно использовать другие программно-отладочные комплексы, такие как STK500, STK600, Atmel-ICE, AVRISP MKII и др.

STK500 – завершенный стартовый набор, средство программирования и система проектирования для Atmel AVR микроконтроллеров. STK500 дает пользователям AVR устройств полную свободу в разработке и тестировании проектов на AVR устройствах и их макетов.

Отличительные особенности отладочного комплекса STK500

- Работа под управлением AVR Studio.
- Последовательное внутрисистемное программирование.
- Внутрисистемное программирование во внешней целевой системе.
- Параллельное и последовательное программирование с повышенным напряжением.
- Интерфейс RS-232 для связи с ПК.
- Разъемы 8-, 20-, 28- и 40-выводные AVR устройств.
- Гибкость тактирования, питания и системного сброса.
- Светодиоды и кнопка для исследований.
- Все порты ввода-вывода легкодоступны на разъеме.
- RS-232 драйвер и разъем.
- Обновление выполняется из AVR Studio.
- Разъемы расширения для вставляемых модулей и области макетирования.
- Целевое напряжение (1,8–6,0) В, напряжение питания (9–12) В.

Внешний вид отладочного комплекса STK500 приведен на рисунке 6.2.

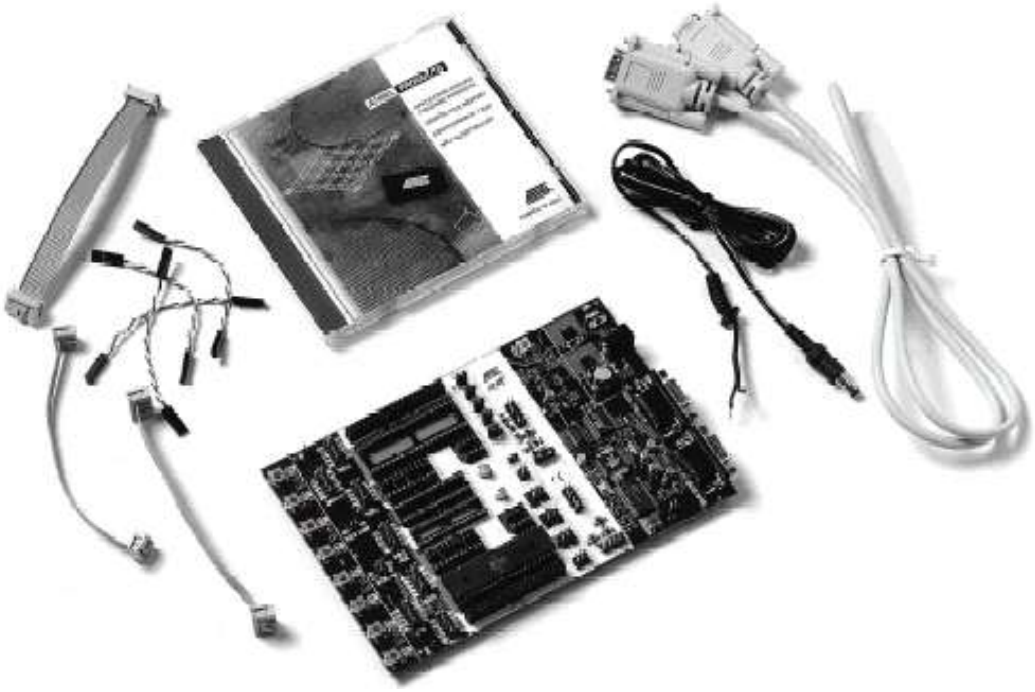


Рисунок 6.2 – Внешний вид отладочного комплекса STK500

STK500 поддерживает все режимы программирования AVR микроконтроллеров, в т. ч. программирование на панелях и внутрисистемное программирование в целевой системе.

Интерфейс программирования STK500 интегрирован в AVR Studio. Флэш-память, ЭППЗУ, а также биты конфигурации и защиты могут программироваться индивидуально или с помощью опции автоматического программирования последовательности.

Частота тактирования и напряжение питания также управляются из AVR Studio.

Программа для программирования под DOS также входит в комплект для эффективного программирования в производственной среде.

Активный код симулятора или эмулятора может быть легко загружен в STK500 выполнением щелчка мыши в AVR Studio.

Поддерживаются различные типы ИС, в том числе: AT90S8535, ATmega8535 и др.

Примечания

1 Устройство программно-отладочного комплекса STK500 поддерживается:

- через ISP программирование внешнего целевого устройства или с помощью модуля расширения STK501;
- через ISP программирование внешнего целевого устройства или с помощью модуля расширения STK502;
- через ISP программирование внешнего целевого устройства.

2 Также поддерживаются версии с пониженным питанием: поставляется с адаптером питания (100–240) В, 50/60 Гц.

Средство программирования разработки АО «НИИЭТ» – USB-программатор КФДЛ.301411.247

Для программирования ИС 1887ВЕ4У можно использовать USB-программатор КФДЛ.301411.247.

Программное обеспечение (ПО) работает в среде WindowsTM. Интерфейс пользователя – многооконный, см. рисунок 6.3.

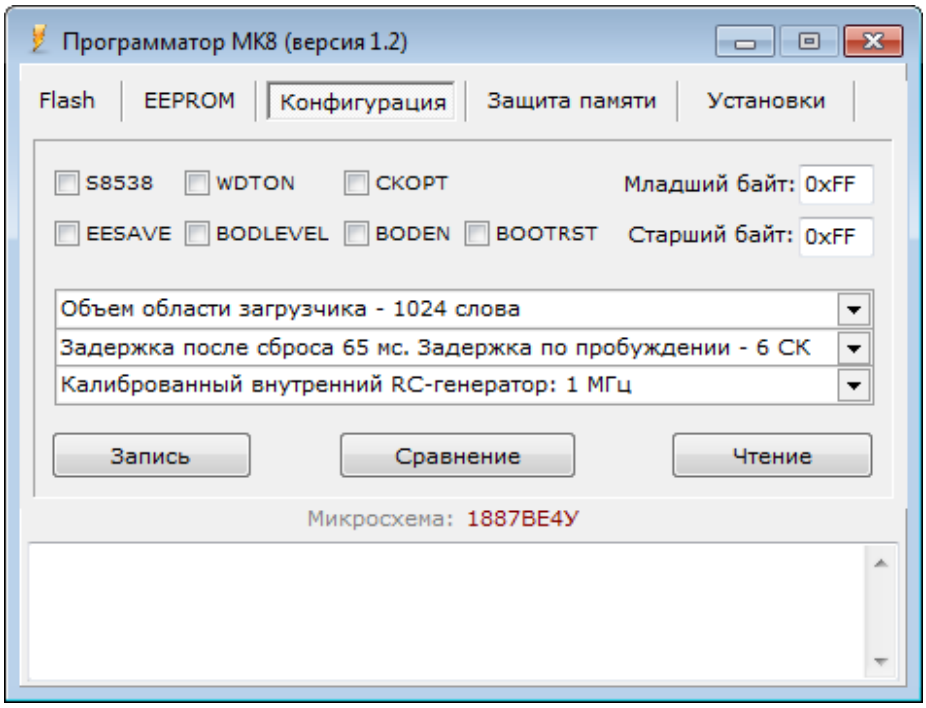


Рисунок 6.3 – Внешний вид ПО КФДЛ.301411.247 для ИС 1887ВЕ4У

ПО КФДЛ.301411.247 представляет собой архив «Программатор МК8.zip», состоящий из драйвера USB-порта и программы «Программатор МК8».

Интуитивно понятный интерфейс обеспечивает доступ ко всем режимам программирования, стирания и верификации памяти программ, данных, бит конфигурации и бит защиты информации.

Окно информации обеспечивает пользователя информацией и рекомендациями при работе с ПО.

Заключение

В настоящем руководстве пользователя приведено подробное описание архитектуры, функционального построения, системы команд и особенностей применения ИС 1887ВЕ4У, которая представляет собой 8-разрядный микроконтроллер с AVR RISC-архитектурой, тактовой частотой 8 МГц, содержащий 8 Кбайт Flash ЗУ программ, ЭСППЗУ (1К × 8) бит, ОЗУ (512 × 8) бит, 8-канальный 10-разрядный АЦП, последовательный периферийный интерфейс SPI, двухпроводной последовательный интерфейс TWI, последовательный порт UART.

Микроконтроллер предназначен для применения в системах управления робототехническими комплексами, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, телекоммуникационной технике и т. д.

Все значения электрических параметров микросхемы 1887ВЕ4У приведены в технических условиях АЕЯР.431280.537ТУ. Значения параметров, приведенные в настоящем техническом описании, являются справочными.

Приложение А
(обязательное)
Описание системы команд

В приложении А дано описание системы команд.
В таблице А.1 приведена группа команд логических операций.
В таблице А.2 приведена группа команд арифметических операций.
В таблице А.3 приведена группа команд операций над битами.
В таблице А.4 приведена группа команд пересылки данных.
В таблице А.5 приведена группа команд передачи управления.
В таблице А.6 приведена группа команд управления системой.

Таблица А.1 – Группа команд логических операций

Мнемоника	Описание	Операция	Циклы	Флаги
AND Rd, Rr	Логическое И двух РОН	Rd = Rd · Rr	1	Z, N, V, S
ANDI Rd, k	Логическое И РОН и константы	Rd = Rd · K	1	Z, N, V, S
EOR Rd, Rr	Исключающее ИЛИ двух РОН	Rd = Rd ^ Rr	1	Z, N, V, S
OR Rd, Rr	Логическое ИЛИ двух РОН	Rd = Rd | Rr	1	Z, N, V, S
ORI Rd, k	Логическое ИЛИ РОН и константы	Rd = Rd | K	1	Z, N, V, S
COM Rd	Перевод в обратный код	Rd = $FF – Rd	1	Z, N, V, C, S
NEG Rd	Перевод в дополнительный код	Rd = $00 – Rd	1	Z, N, V, C, S, H
CLR Rd	Сброс всех разрядов РОН	Rd = Rd ^ Rd	1	Z, N, V, S
SER Rd	Установка всех разрядов РОН	Rd = $FF	1	–
TST Rd	Проверка РОН на отрицательное и нулевое значение	Rd = Rd · Rd	1	Z, N, V, S
SWAP Rd	Обмен местами тетрад в РОН	Rd(3–0) = Rd(7–4) Rd(7–4) = Rd(3–0)	1	–

Таблица А.2 – Группа команд арифметических операций

Мнемоника	Описание	Операция	Циклы	Флаги
ADD Rd, Rr	Сложение двух РОН	Rd = Rd + Rr	1	Z, C, N, V, S, H
ADC Rd, Rr	Сложение двух РОН с переносом	Rd = Rd + Rr + C	1	Z, C, N, V, S, H
ADIW Rd, K	Сложение регистровой пары с константой	(Rd + 10):Rd = = (Rd + 1):Rd + K	2	Z, C, N, V, S
SUB Rd, Rr	Вычитание двух РОН	Rd = Rd – Rr	1	Z, C, N, V, S, H
SUBI Rd, K	Вычитание константы из РОН	Rd = Rd – K	1	Z, C, N, V, S, H
SBC Rd, Rr	Вычитание двух РОН с заемом	Rd = Rd – Rr – C	1	Z, C, N, V, S, H
SBCI Rd, K	Вычитание константы из РОН с заемом	Rd = Rd – K – C	1	Z, C, N, V, S, H
SBIW Rd, K	Вычитание константы из регистровой пары	(Rd + 1):Rd = = (Rd + 1):Rd – K	2	Z, C, N, V, S
DEC Rd	Декремент РОН	Rd = Rd – 1	1	Z, N, V, S
INC Rd	Инкремент РОН	Rd = Rd + 1	1	Z, N, V, S
ASR Rd	Арифметический сдвиг вправо	Rd(n) = Rd(n + 1), n = 0–6	1	Z, N, V, C, S
LSL Rd	Логический сдвиг влево	Rd(n + 1) = Rd(n), Rd(0) = 0, C = Rd(7)	1	Z, N, V, C, H, S
LSR Rd	Логический сдвиг вправо	Rd(n) = Rd(n + 1), Rd(7) = 0, C = Rd(0)	1	Z, N, V, C, S
ROL Rd	Сдвиг влево через перенос	Rd(0) = C, Rd(n + 1) = Rd(n), C = Rd(7)	1	Z, N, V, C, S, H
ROR Rd	Сдвиг вправо через перенос	Rd(7) = C, Rd(n) = Rd(n + 1), C = Rd(0)	1	Z, N, V, C, S
MUL Rd, Rr	Умножение беззнаковых чисел	R1:R0 = Rd × Rr	2	Z, C
MULS Rd, Rr	Умножение чисел со знаком	R1:R0 = Rd × Rr	2	Z, C
MULSU Rd, Rr	Умножение беззнакового числа на число со знаком	R1:R0 = Rd × Rr	2	Z, C
FMUL Rd, Rr	Умножение дробных беззнаковых чисел	R1:R0 = = (Rd × Rr) << 1	2	Z, C
FMULS Rd, Rr	Умножение дробных чисел со знаком	R1:R0 = = (Rd × Rr) << 1	2	Z, C
FMULSU Rd, Rr	Умножение дробного беззнакового числа и дробного числа со знаком	R1:R0 = = (Rd × Rr) << 1	2	Z, C

Таблица А.3 – Группа команд операций над битами

Мнемоника	Описание	Операция	Циклы	Флаги
CBR Rd, K	Сброс разряда(ов) РОН	Rd = Rd · ($FF – K)	1	Z, N, V, S
SBR Rd, K	Установка разряда(ов) РОН	Rd = Rd | K	1	Z, N, V, S
CBI P, b	Сброс разряда(ов) РВВ	I/O(P, b) = 0	2	-
SBI P, b	Установка разряда(ов) РВВ	I/O(P, b) = 1	2	-
BCLR s	Сброс флага	SREG(s) = 0	1	SREG(s)
BSET s	Установка флага	SREG(s) = 1	1	SREG(s)
BLD Rd, b	Загрузка разряда РОН из флага T	Rd(b) = T	1	-
BST Rd, b	Запись разряда РОН в флаг T	T = Rd(b)	1	T
CLC	Сброс флага переноса	C = 0	1	C
SEC	Установка флага переноса	C = 1	1	C
CLN	Сброс флага отрицательного числа	N = 0	1	N
SEN	Установка флага отрицательного числа	N = 1	1	N
CLZ	Сброс флага нуля	Z = 0	1	Z
SEZ	Установка флага нуля	Z = 1	1	Z
CLI	Общее запрещение прерываний	I = 0	1	I
SEI	Общее разрешение прерываний	I = 1	1	I
CLS	Сброс флага знака	S = 0	1	S
SES	Установка флага знака	S = 1	1	S
CLV	Сброс флага переполнения дополнения кода	V = 0	1	V
SEV	Установка флага переполнения дополнения кода	V = 1	1	V
CLT	Сброс флага T	T = 0	1	T
SET	Установка флага T	T = 1	1	T
CLH	Сброс флага половинного переноса	H = 0	1	H
SEH	Установка флага половинного переноса	H = 1	1	H

Таблица А.4 – Группа команд пересылки данных

Мнемоника	Описание	Операция	Циклы	Флаги
MOV Rd, Rr	Пересылка между РОН	Rd = Rr	1	-
MOVW Rd, Rr	Пересылка двухбайтовых значений	(Rd + 1):Rd = (Rr + 1):Rd	2	-
LDI Rd, K	Загрузка константы в РОН	Rd = K	1	-
LD Rd, X	Косвенное чтение	Rd = (X)	2	-
LD Rd, X+	Косвенное чтение с постинкрементом	Rd = (X), X = X + 1	2	-
LD Rd, –X	Косвенное чтение с преддекрементом	X = X – 1, Rd = (X)	2	-
LD Rd, Y	Косвенное чтение	Rd = (Y)	2	-
LD Rd, Y+	Косвенное чтение с постинкрементом	Rd = (Y), Y = Y + 1	2	-
LD Rd, –Y	Косвенное чтение с преддекрементом	Y = Y – 1, Rd = (Y)	2	-
LDD Rd, Y + q	Косвенное относительное чтение	Rd = (Y + q)	2	-
LD Rd, Z	Косвенное чтение	Rd = (Z)	2	-
LD Rd, Z+	Косвенное чтение с постинкрементом	Rd = (Z), Z = Z + 1	2	-
LD Rd, –Z	Косвенное чтение с преддекрементом	Z = Z – 1, Rd = (Z)	2	-
LDD Rd, Z + q	Косвенное относительное чтение	Rd = (Z + q)	2	-
LDS Rd, k	Непосредственное чтение из ОЗУ	Rd = (k)	2	-
ST X, Rr	Косвенная запись	(X) = Rr	2	-
ST X+, Rr	Косвенная запись с постинкрементом	(X) = Rr, X = X + 1	2	-
ST –X, Rr	Косвенная запись с преддекрементом	X = X – 1, (X) = Rr	2	-
ST Y, Rr	Косвенная запись	(Y) = Rr	2	-
ST Y+, Rr	Косвенная запись с постинкрементом	(Y) = Rr, Y = Y + 1	2	-
ST –Y, Rr	Косвенная запись с преддекрементом	Y = Y – 1, (Y) = Rr	2	-
STD Y + q, Rr	Косвенная относительная запись	(Y + q) = Rr	2	-
ST Z, Rr	Косвенная запись	(Z) = Rr	2	-
ST Z+, Rr	Косвенная запись с постинкрементом	(Z) = Rr, Z = Z + 1	2	-
ST –Z, Rr	Косвенная запись с преддекрементом	Z = Z – 1, (Z) = Rr	2	-
STD Z + q, Rr	Косвенная относительная запись	(Z + q) = Rr	2	-
STS k, Rr	Непосредственная запись в ОЗУ	(k) = Rr	2	-
LPM	Загрузка данных из памяти программ	R0 = (Z)	3	-
LPM Rd, Z	Загрузка данных из памяти программ	Rd = (Z)	3	-
LPM Rd, Z+	Загрузка данных из памяти программ с постинкрементом	Rd = (Z), Z = Z + 1	3	-
SPM	Запись в память программ	(Z) = R1:R0	-	-
IN Rd, P	Пересылка РВВ в РОН	Rd = I/O(P)	1	-
OUT P, Rr	Пересылка РОН в РВВ	I/O(P) = Rr	1	-
PUSH Rr	Сохранение байта в стеке	STACK = Rr	2	-
POP Rd	Извлечение байта из стека	Rd = STACK	2	-

Таблица А.5 – Группа команд передачи управления

Мнемоника	Описание	Операция	Циклы	Флаги
RJMP k	Относительный безусловный переход	PC = PC + k + 1	2	-
IJMP	Косвенный безусловный переход	PC(15–0) = Z	2	-
RCALL k	Относительный вызов подпрограммы	PC = PC + k + 1	3	-
ICALL	Косвенный вызов подпрограммы	PC = Z	3	-
RET	Возврат из подпрограммы	PC = STACK	4	-
RETI	Возврат из подпрограммы обработки прерывания	PC = STACK	4	I
CP Rd, Rr	Сравнение РОН	Rd – Rr	1	Z, N, V, C, S, H
CPC Rd, Rr	Сравнение РОН с учетом переноса	Rd – Rr – C	1	Z, N, V, C, S, H
CPI Rd, K	Сравнение РОН с константой	Rd – K	1	Z, N, V, C, S, H
CPSE Rd, Rr	Сравнение и пропуск следующей команды при равенстве	При Rd = Rr PC = PC + 2 (or 3)	1 / 2 / 3	–
SBRC Rr, b	Пропуск следующей команды, если разряд РОН сброшен	При Rr(b) = 0 PC = PC + 2 (or 3)	1 / 2 / 3	–
SBRS Rr, b	Пропуск следующей команды, если разряд РОН установлен	При Rr(b) = 1 PC = PC + 2 (or 3)	1 / 2 / 3	–
SBIC P, b	Пропуск следующей команды, если разряд РВВ сброшен	При I/O(P, b) = 0 PC = PC + 2 (or 3)	1 / 2 / 3	–
SBIS P, b	Пропуск следующей команды, если разряд РВВ установлен	При I/O(P, b) = 1 PC = PC + 2 (or 3)	1 / 2 / 3	–
BRBC s, k	Переход, если флаг регистра SREG сброшен	При SREG(s) = 0 PC = PC + k + 1	1 / 2	–
BRBS s, k	Переход, если флаг регистра SREG установлен	При SREG(s) = 1 PC = PC + k + 1	1 / 2	–
BRCS k	Переход по переносу	При C = 1 PC = PC + k + 1	1 / 2	–
BRCC k	Переход, если нет переноса	При C = 0 PC = PC + k + 1	1 / 2	–
BREQ k	Переход по равенству	При Z = 1 PC = PC + k + 1	1 / 2	–
BRNE k	Переход, если нет равенства	При Z = 0 PC = PC + k + 1	1 / 2	–
BRSH k	Переход по «больше или равно»	При C = 0 PC = PC + k + 1	1 / 2	–
BRLO k	Переход по «меньше»	При C = 1 PC = PC + k + 1	1 / 2	–
BRMI k	Переход по «отрицательное значение»	При N = 1 PC = PC + k + 1	1 / 2	–
BRPL k	Переход по «положительное значение»	При N = 0 PC = PC + k + 1	1 / 2	–
BRGE k	Переход по «больше или равно» (числа со знаком)	При (N ^ V = 0) PC = PC + k + 1	1 / 2	–
BRLT k	Переход по «меньше или равно» (числа со знаком)	При (N ^ V = 1) PC = PC + k + 1	1 / 2	–
BRHS k	Переход по половинному переносу	При H = 1 PC = PC + k + 1	1 / 2	–
BRHC k	Переход, если нет половинного переноса	При H = 0 PC = PC + k + 1	1 / 2	–

Окончание таблицы А.5

Мнемоника	Описание	Операция	Циклы	Флаги
BRTS k	Переход, если флаг T установлен	При T = 1 PC = PC + k + 1	1 / 2	–
BRTC k	Переход, если флаг T сброшен	При T = 0 PC = PC + k + 1	1 / 2	–
BRVS k	Переход по переполнению дополнительного кода	При V = 1 PC = PC + k + 1	1 / 2	–
BRVC k	Переход, если нет переполнения дополнительного кода	При V = 0 PC = PC + k + 1	1 / 2	–
BRID k	Переход, если прерывания запрещены	При I = 0 PC = PC + k + 1	1 / 2	–
BRIE k	Переход, если прерывания разрешены	При I = 1 PC = PC + k + 1	1 / 2	–

Таблица А.6 – Группа команд управления системой

Мнемоника	Описание	Операция	Циклы	Флаги
NOP	Нет операции	–	1	–
SLEEP	Переход в «спящий» режим	–	1	–
WDR	Сброс сторожевого таймера	–	1	–

Принятые обозначения

Регистр статуса (SREG):

SREG:	Регистр статуса
C:	Флаг переноса
Z:	Флаг нулевого значения
N:	Флаг отрицательного значения
V:	Флаг-указатель переполнения дополнения до двух
S:	N ^ 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:	Константа 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:	Регистры 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

Логические операции:

«|»	Логическое ИЛИ
«^»	Логическое исключающее ИЛИ
«·»	Логическое И

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 + 1):Rd, K	d ϵ{24, 26, 28, 30}, $0 \le K \le 63$	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#

C: Устанавливается, если результат $0000, в ином случае очищается
R15# · Rdh7
Устанавливается, если есть перенос из MSB результата, в ином случае очищается

R: (Результат) соответствует Rdh:Rdl после выполнения команды
(Rdh7–Rdh0 = R15–R8, Rdl7–Rdl0 = R7–R0)

Пример:

```
adiw r24, 1  ; Сложить 1 с r25:r24
adiw r30, 63 ; Сложить 63 с Z указателем (r31 : r30)
```

Слов: 1 (2 байта)

Циклов: 2

Команда AND – выполнить логическое 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	$16 \leq d \leq 31$, $0 \leq K \leq 255$	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.

Операция:

	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	$0 \le s \le 7$	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	$0 \leq d \leq 31$, $0 \leq b \leq 7$	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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 ≤ s ≤ 7, –64 ≤ k < +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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 значение положительное

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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 – 63 ≤ назначение ≤ PC + 64). 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 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	$0 \leq P \leq 31, 0 \leq b \leq 7$	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 постоянной K.

Операция:

(i) Rd <-- Rd · ($FF – K)

	Синтаксис	Операнды:	Счетчик программ:
(i)	CBR Rd, K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC <-- PC + 1

16-разрядный код операции:

Смотри команду

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 $0 \leq d \leq 31$ 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 · Rd7# · R7# + Rd7# · Rr7 · R7
Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R7
Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: R7# · R6# · R5# · R4# · R3# · R2# · R1# · R0#
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 · Rd7# · R7# + Rd7# · Rr7 · R7
Устанавливается, если в результате операции образуется переполнение дополнения до двух, в ином случае очищается

N: R7
Устанавливается, если в результате установлен MSB, в ином случае очищается

Z: R7# · R6# · R5# · R4# · R3# · R2# · R1# · R0# · Z
Предшествующее значение остается неизменным, если результатом является ноль, в ином случае очищается

C: Rd7# · Rr7 + Rr7 · R7 + R7 · Rd7#
Устанавливается, если абсолютное значение Rr плюс предшествовавший перенос больше абсолютного значения Rd, в ином случае очищается

R: (Результат) после выполнения команды

Пример:

```
                ; Сравнить r3 : r2 с r1 : r0
        cp r2, r0  ; Сравнить старший байт
        cpc r3, r1 ; Сравнить младший байт
        brne noteq ; Перейти, если не равно
                   . . .
noteq:  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	$16 \le d \le 31$, $0 \le K \le 255$	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	$0 \leq d \leq 31$, $0 \leq r \leq 31$	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

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 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) DEC Rd $0 \leq d \leq 31$ 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 $0 \leq d \leq 31$ 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 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.

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	$0 \le d \le 31$	PC<-- + 1
(ii)	LD Rd, X+	$0 \le d \le 31$	PC<-- + 1
(iii)	LD Rd, –X	$0 \le d \le 31$	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 пред-
декрементируется)
```

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.

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
(iiii)	Rd <-- (Y + q)		Y: Неизменен, q: смещение

	Синтаксис	Операнды:	Счетчик программ:
(i)	LD Rd, Y	$0 \le d \le 31$	PC <-- + 1
(ii)	LD Rd, Y+	$0 \le d \le 31$	PC <-- + 1
(iii)	LD Rd, –Y	$0 \le d \le 31$	PC <-- + 1
(iiii)	LDD Rd, Y + q	$0 \le d \le 31$ $0 \le q \le 63$	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 постинкремен-
тируется)
   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 преддекремен-
тируется)
   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.

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
(iiii)	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
(iiii)	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 Y
```

```
   ldi  r28, $20  ;Установить $20 в младший байт Y
   ld   r0, Y+    ;Загрузить в r0 содерж. SRAM по адресу $20 (Y постинкремен-
тируется)
   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 преддекремен-
тируется)
   ldd  r4, Y+2   ;Загрузить в r4 содержимое SRAM по адресу $24
```

Слов: 1 (2 байта)

Циклов: 2

Команда LDI – загрузить непосредственное значение

Описание:
Загружается 8-разрядная константа в регистр от 16 по 31

Операция:
(i) Rd <-- K

	Синтаксис	Операнды:	Счетчик программ:
(i)	LDI Rd, K	$16 \le d \le 31$, $0 \le K \le 255$	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.

	Операция:		Комментарий:
(i)	R0 <-- (Z)		Z: Неизменен, подразумеваемый регистр назначения R0
(ii)	Rd <-- (Z)		Z: Неизменен
(iii)	Rd <-- (Z)	Z <-- Z + 1	Z: Инкрементирован впоследствии

	Синтаксис	Операнды:	Счетчик программ:
(i)	LPM	None, R0 imlied	PC <-- PC + 1
(ii)	LPM Rd, Z	$0 \leq d \leq 31$	PC <-- PC + 1
(iii)	LPM Rd, Z+	$0 \leq d \leq 31$	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:

(i) C <-- b(7–0) <-- 0

	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:

(i) 0 --> b(7–0) --> C

	Синтаксис	Операнды:	Счетчик программ:
(i)	LSL 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
rcall  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 × 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#

R: (Результат) соответствует R1, R0 после выполнения команды

Пример:

```
mul r5, r4 ; Перемножить r5 и r4
movw r4, r0 ; Вернуть результат обратно в r5:r4
```

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	$0 \le d \le 31$	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 $80.

R: (Результат) соответствует Rd после выполнения команды

Пример:

```
            sub  r11, r0  ; Вычесть r0 из r11
            brpl positive ; Перейти, если результат положительный
            neg  r11      ; Выполнить дополнение до двух r11
positive:   nop           ; Перейти по назначению (пустая операция)
```

Слов: 1 (2 байта)

Циклов: 1

Команда NOP – выполнить холостую команду

Описание:
Команда выполняется за один цикл без выполнения операции.

Операция:

(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 | 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 | 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	$0 \leq r \leq 31, 0 \leq P \leq 63$	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	$0 \le d \le 31$	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:

```
          rcall routine ; Вызвать подпрограмму
          . . .
routine:  push r14      ; Сохранить r14 в стеке
          push r13      ; Сохранить r13 в стеке
          . . .
          pop  r13      ; Восстановить r13
          pop  r14      ; Восстановить r14
          ret           ; Вернуться из подпрограммы
```

Слов: 1 (2 байта)

Циклов: 2

Команда PUSH – поместить регистр в стек

Описание:
Команда помещает содержимое регистра Rd в стек.

Операция:
(i) STACK <-- Rr

	Синтаксис	Операнды:	Счетчик программ:
(i)	PUSH Rr	$0 \le d \le 31$	PC <-- PC + 1 SP <-- SP – 1

16-разрядный код операции:

1001	001d	dddd	1111

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
          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 (4 Кбайт). Адрес возврата (после выполнения команды RCALL) сохраняется в стеке (См. также команду RCALL).

Операция:

(i) PC <-- PC + k + 1 Приборы с 16-разрядным счетчиком команд, максимум 128 Кбайт памяти программ.

	Синтаксис	Операнды:	Счетчик программ:	Стек
(i)	RCALL k	$-2K \leq k \leq 2K$	PC <-- PC + k + 1	STACK <-- PC + 1 SP <-- SP – 2 (2 байта, 16 бит)

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 128 Кбайт памяти программ.

	Синтаксис	Операнды:	Счетчик программ:	Стек
(i)	RET	None	См. операцию	SP <-- SP + 2 (2 байта, 16 бит)

16-разрядный код операции:

1001	0101	0000	1000

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: 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 128 Кбайт памяти программ.

	Синтаксис	Операнды:	Счетчик программ:	Стек
(i)	RETI	None	См. операцию	SP <-- SP + 2 (2 байта, 16 бит)

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 – перейти относительно

Описание:

Команда выполняет относительный переход по адресу в пределах 2К слов (4 Кбайт) текущего состояния счетчика команд. В ассемблере вместо относительных операндов используются метки. Для AVR микроконтроллеров с памятью программ не превышающей 4К слов (8 Кбайт) данная команда может адресовать всю память программ.

Операция

(i) PC <-- PC + k + 1

	Синтаксис	Операнды:	Счетчик программ:	Стек
(i)	RJMP k	$-2K \leq k \leq 2K$	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:

(i) C–b(7–0) <-- C

	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: 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:

```
          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:

(i) C --> b(7–0) --> C

	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, лля 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	$0 \leq P \leq 31, 0 \leq b \leq 7$	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 1 слово

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 условия не соблюдены, нет пропуска, 2 – если условия соблюдены, выполняется пропуск

Команда SBIS – пропустить, если бит в регистре I/O установлен

Описание:

Команда проверяет состояние бита в регистре I/O и, если этот бит установлен, пропускает следующую команду. Данная команда работает с младшими 32 регистрами I/O (адреса с 0 по 31).

Операция:

(i) If I/O(P, b) = 1 then PC <-- 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 1 слово

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 условия не соблюдены, нет пропуска, 2 – если условия соблюдены, выполняется пропуск

Команда SBIW – вычесть непосредственное значение из слова

Описание:

Вычитание непосредственного значения (0–63) из пары регистров и размещение результата в паре регистров. Команда работает с четырьмя верхними парами регистров, удобна для работы с регистрами-указателями.

Операция:

(i) (Rd + 1):Rd <-- (Rd+1):Rd – K

Синтаксис	Операнды:	Счетчик программ:
(i) SBIW (Rd +1):Rd, K	d ϵ{24, 26, 28, 30}, 0 ≤ K ≤ 63	PC <-- PC + 1

16-разрядный код операции:ϵ

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  r24, 1  ; Вычесть 1 из r25:r24
sbiw  r28, 63 ; Вычесть 63 из Y указателя (r29 : r28)
```

Слов: 1 (2 байта)

Циклов: 2

Команда SBR – установить биты в регистре

Описание:

Команда выполняет установку определенных битов в регистре Rd. Команда выполняет логическое ИЛИ между содержимым регистра Rd и маской-константой K и размещает результат в регистре назначения Rd.

Операция:

(i) Rd <-- Rd | 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, 3F0 ; Установить биты 0 и 1 в r16
sbr r17, $F0 ; Установить старшие 4 бита в r17
```

Слов: 1 (2 байта)

Циклов: 1

Команда SBRC – пропустить, если бит в регистре очищен

Описание:

Команда проверяет состояние бита в регистре и, если этот бит очищен, пропускает следующую команду.

Операция:

(i) If Rr(b) = 0 then PC <-- PC + 2 (or 3) else PC <-- PC + 1

	Синтаксис	Операнды:	Счетчик программ:
(i)	SBRC Rr, b	$0 \le r \le 31, 0 \le b \le 7$	PC <-- PC + 1, если условия не соблюдены, нет пропуска PC <-- PC + 2, если следующая команда длиной в 1 слово

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 условия не соблюдены, нет пропуска, 2 – если условия соблюдены, выполняется пропуск

Команда SBRS – пропустить, если бит в регистре установлен

Описание:

Команда проверяет состояние бита в регистре и, если этот бит установлен, пропускает следующую команду.

Операция

(i) If Rr(b) = 1 then PC <-- PC + 2 (or 3) else PC <-- PC + 1

	Синтаксис	Операнды:	Счетчик программ:
(i)	SBRS Rr, b	$0 \le r \le 31, 0 \le b \le 7$	PC <-- PC + 1, если условия не соблюдены, нет пропуска PC <-- PC + 2, если следующая команда длиной в 1 слово

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 условия не соблюдены, нет пропуска, 2 – если условия соблюдены, выполняется пропуск

Команда SEC – установить флаг переноса

Описание:
Команда устанавливает флаг переноса (C) в регистре статуса (SREG)

Операция:
(i) C <-- 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	$16 \leq d \leq 31$	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 «спящий» режим

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:

Перевод микроконтроллера в «спящий» режим.

	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
sleep       ; Перевести MCU в режим sleep
```

Слов: 1 (2 байта)

Циклов: 1

Команда SPM – запись в память программ

Описание:

Команду SPM можно использовать для стирания страницы из памяти программы, для записи страницы в память программы (которая уже стерта) и для установки битов блокировки загрузчика. Память программы должна стираться по одной странице за один раз. При стирании памяти программы в качестве адреса страницы используются RAMPZ и Z-регистр. При записи в память программы RAMPZ и Z-регистр используются в качестве адреса страницы или слова, а в качестве данных (1) используется пара регистров R1–R0. При установке битов блокировки загрузчика в качестве данных используется пара регистров R1–R0. Эта инструкция может обращаться ко всей памяти программы.

Примечание – R1 определяет старший байт инструкции, а R0 определяет младший байт инструкции.

	Операция:	Комментарий:
(i)	(RAMPZ:Z) ← $ffff	Стереть страницу памяти программы
(ii)	(RAMPZ:Z) ← R1:R0	Записать слово программы в память
(ii)	(RAMPZ:Z) ← R1:R0	Запись временного буфера страницы
(iv)	(RAMPZ:Z) ← TEMP	Запись временного буфера страниц в память программы
(v)	BLBITS ← R1:R0	Установите биты блокировки загрузчика

	Синтаксис:	Операнды:	Счетчик программ:
(i)–(v)	SPM	None	PC <-- PC + 1

16-разрядный код операции:

1001	0101	1110	1000

Булевы выражения регистра статуса (SREG)

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Пример:

```
; В этом примере показана запись одной страницы
;- подпрограмма записывает одну страницу данных из ОЗУ во флэш-память
; начало размещения данных в ОЗУ определяет указатель Y
; начало размещения данных во флэш-памяти определяет указатель Z
;- обработка ошибок не включена
;- подпрограмма должна быть размещена в загрузочной области
; (по крайней мере, подпрограмма do_spm)
;- используемые регистры: r0, r1, temp1, temp2, looplo, loophi, spmcrval
; (temp1, temp2, looplo, loophi, spmcrval должны быть определены пользователем)
; хранение и восстановление регистров не включено в процедуру
; использование регистров может быть оптимизировано за счет размера кода
.equPAGESIZEB = PAGESIZE*2;PAGESIZEB - размер страницы в байтах, а не в словах
.org SMALLBOOTSTART
страница записи:
;стереть страницу
ldi spmcrval, (1<<PGERS) + (1<<SPMEN)
call do_spm
;перенести данные из ОЗУ в буфер страниц флэш-памяти
ldi looplo, low(PAGESIZEB) ; переменная цикла инициализации
 ldi loophi, high(PAGESIZEB) ; не требуется для PAGESIZEB<=256
```

```
wrloop: ldr0, Y+
ldi 1, Y+
ldi spmcrval (1<<SPMEN)
call do_spm

adiw ZH:ZL, 2
sbiw loophi : looplo, 2 ; используйте subi для PAGESIZEB<=256
brne wrloop

; выполнить запись страницы subi ZL, низкий (PAGESIZEB)
; указатель восстановления sbci ZH, high(PAGESIZEB)
; не требуется для PAGESIZEB<=256
ldi spmcrval, (1<<PGWRT) + (1<<SPMEN)
call do_spm

; перечитать и проверить, необязательно
ldi looplo, low(PAGESIZEB) ; переменная цикла инициализации
ldi loophi, high(PAGESIZEB) ; не требуется для PAGESIZEB<=256
subi YL, низкий (PAGESIZEB) ; восстановить указатель
sbci YH, высокий (PAGESIZEB)
rdloop : lpmr 0, Z+
ldi 1, Y+
cpser0, r1
jmperror
sbiw loophi : looplo, 2 ; используйте subi для PAGESIZEB<=256
brnerdloop
; возвращаться
ret

do_spm:
; input : spmcrval определяет действие SPM
; отключить прерывания, если они включены, сохранить статус
intemp2, SREG
cli
; проверить завершение предыдущего SPM
Wait : intemp1, SPMCR
sbrctemp1, SPMEN
rjmp wait
; временная последовательность SPM
out SPMCR, spmcrval
spm
; восстановить SREG (чтобы разрешить прерывания, если они изначально
были разрешены)
out SREG, temp2
ret
```

Слов: 1 (2 байта)

Циклы: зависит от операции

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.

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 постинкре-
ментируется)
  st    X, r1    ; Сохранить в r1 содержимое SRAM по адресу $21
  ldi   r26, $23 ; Установить $23 в младший байт X
  st    r2, X    ; Сохранить в r2 содержимое SRAM по адресу $23
  st    r3, -X   ; Сохранить в r3 содержимое SRAM по адресу $22 (X преддекре-
ментируется)
```

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.

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
(iiii)	(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 постин-
крементируется)
    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 предде-
крементируется)
    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.

Использование Z-указателя:

	Операция:		Комментарий:
(i)	(Z) <-- Rr		Z: Неизменен
(ii)	(Z) <-- Rr	Z <-- Z + 1	Z: Инкрементирован впоследствии
(iii)	Z <-- Z – 1	(Z) <-- Rr	Z: Предварительно декрементирован
(iiii)	(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
(iiii)	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 адресу $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 адресу $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 64 Кбайта. Команда STS использует для обращения к памяти выше 64 Кбайт регистр RAMPZ.

Операция:
(i) (k) <-- Rr

	Синтаксис	Операнды:	Счетчик программ:
(i)	STS k, Rr	$0 \leq r \leq 31, 0 \leq k \leq 65535$	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	$0 \leq d \leq 31$	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 И содержимого регистра с самим собой. Содержимое регистра остается неизменным.

Операция:

(i) Rd <-- Rd · Rd

	Синтаксис	Операнды:	Счетчик программ:
(i)	TST Rd	0 ≤ d ≤ 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)

Средние значения фактической частоты внутреннего RC генератора в диапазоне температур и напряжений

Б.1 Средние значения фактической частоты внутреннего RC генератора в диапазоне температур и напряжений 5 В ± 10 % см. таблицы Б.1 – Б.4.

Таблица Б.1 – Значение частоты при f_{RC} (номинальное) = 1 МГц

Т, °С	Значение частоты, кГц, при напряжении питания, В		
	4,5	5,0	5,5
–60	1052	–	1062
25	996	–	1004
85	952	–	960

Таблица Б.2 – Значение частоты при f_{RC} (номинальное) = 2 МГц

Т, °С	Значение частоты, кГц, при напряжении питания, В		
	4,5	5,0	5,5
–60	2128	–	2158
25	2002	–	2032
85	1904	–	1936

Таблица Б.3 – Значение частоты при f_{RC} (номинальное) = 4 МГц

Т, °С	Значение частоты, кГц, при напряжении питания, В		
	4,5	5,0	5,5
–60	4262	–	4362
25	3986	–	4086
85	3744	–	3856

Таблица Б.4 – Значение частоты при f_{RC} (номинальное) = 8 МГц

Т, °С	Значение частоты, кГц, при напряжении питания, В		
	4,5	5,0	5,5
–60	8526	–	8876
25	7844	–	8204
85	7232	–	7602

Б.2 Средние значения фактической частоты внутреннего RC генератора в диапазоне температур и напряжений 3,3 В ± 10 % см. таблицы Б.5 – Б.8.

Таблица Б.5 – Значение частоты при f_{RC} (номинальное) = 1 МГц

Т, °С	Значение частоты, кГц, при напряжении питания, В		
	2,97	3,3	3,63
–60	1060	1060	1064
25	992	1000	1004
85	940	948	952

Таблица Б.6 – Значение частоты при f_{RC} (номинальное) = 2 МГц

Т, °С	Значение частоты, кГц, при напряжении питания, В		
	2,97	3,3	3,63
–60	2112	2132	2148
25	1968	1988	2008
85	1852	1872	1888

Таблица Б.7 – Значение частоты при f_{RC} (номинальное) = 4 МГц

Т, °С	Значение частоты, кГц, при напряжении питания, В		
	2,97	3,3	3,63
- 60	4140	4208	4264
25	3812	3880	3936
85	3536	3604	3664

Таблица Б.8 – Значение частоты при f_{RC} (номинальное) = 8 МГц

Т, °С	Значение частоты, кГц, при напряжении питания, В		
	2,97	3,3	3,63
–60	7932	8188	8388
25	7116	7368	7572
85	6456	6700	6896

Приложение В
(справочное)

Схемы электрические отладочного устройства КФДЛ.301411.243 и расположение элементов на печатных платах

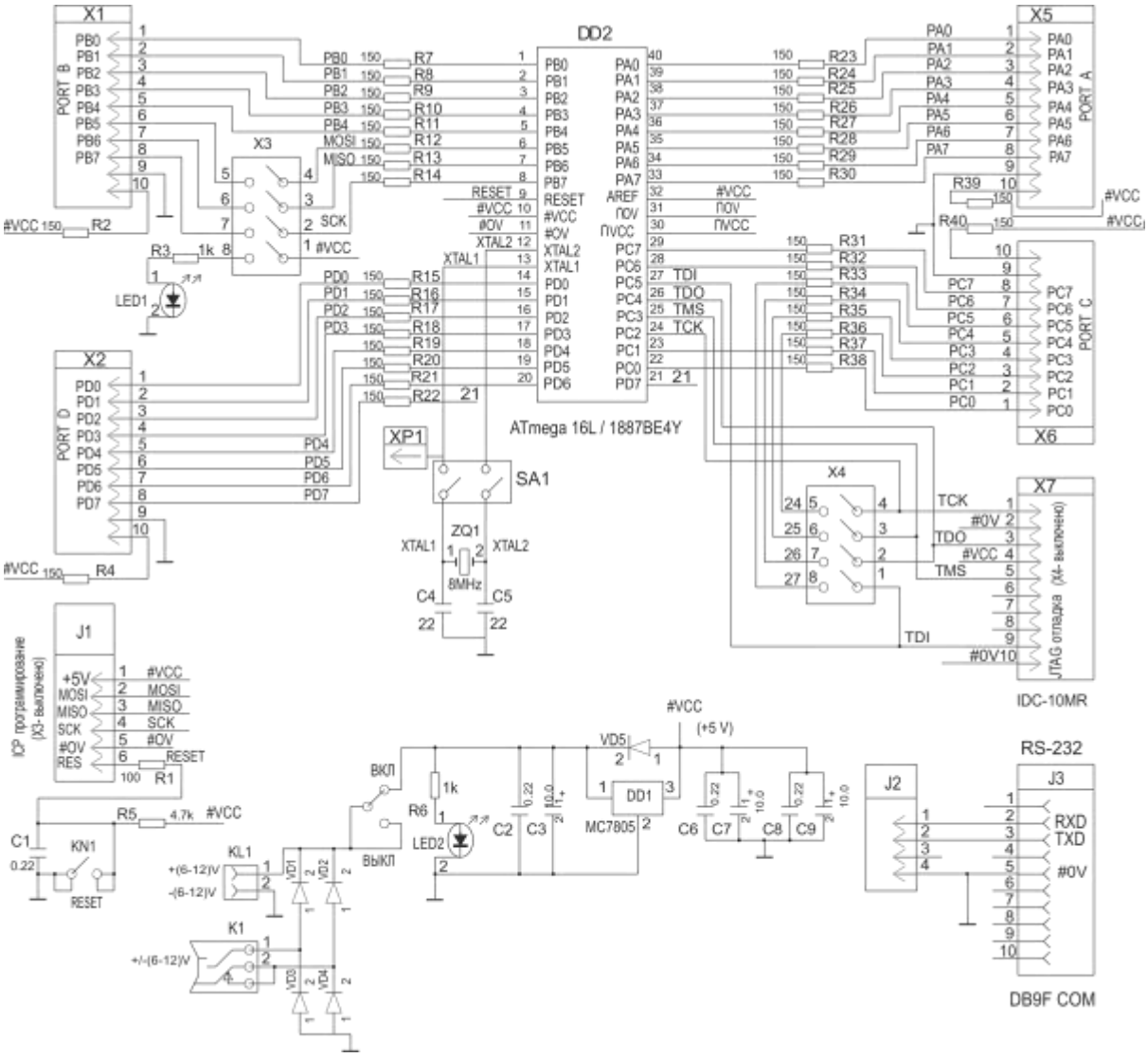


Рисунок В.1 – Схема электрическая основной платы с 40-выводной DIP-колодкой с нулевым усилием для установки микроконтроллеров 1887ВЕ4У / ATmega16 и макетным полем

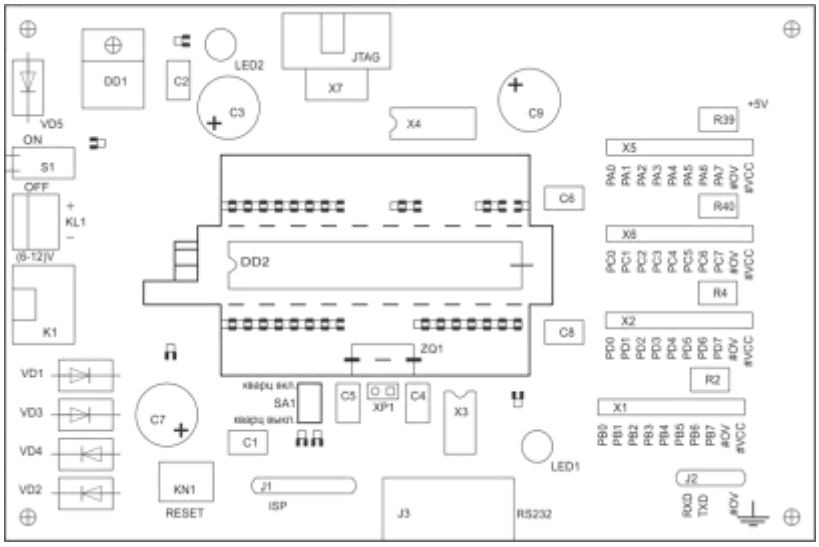


Рисунок В.2 – Расположение элементов на основной плате

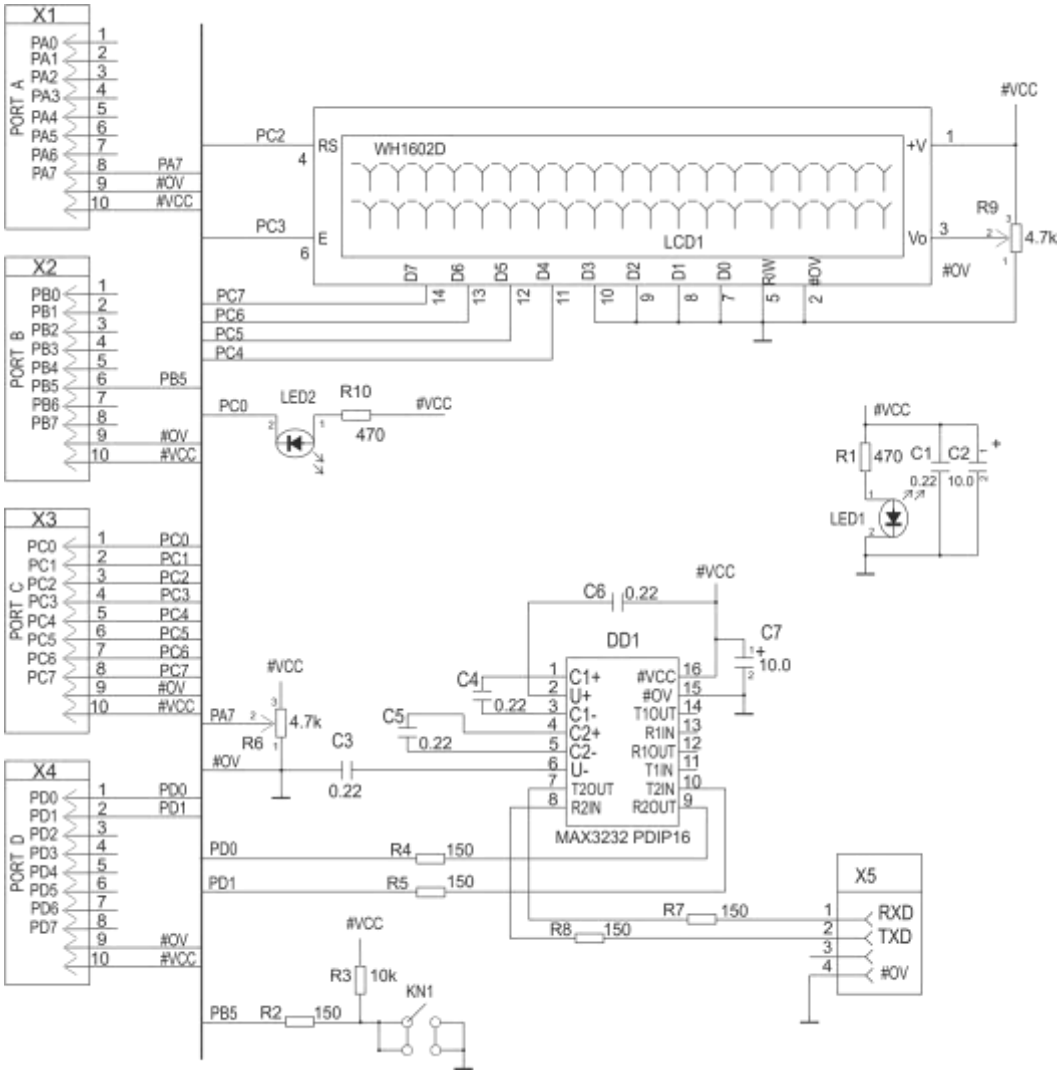


Рисунок В.3 – Схема электрическая платы расширения

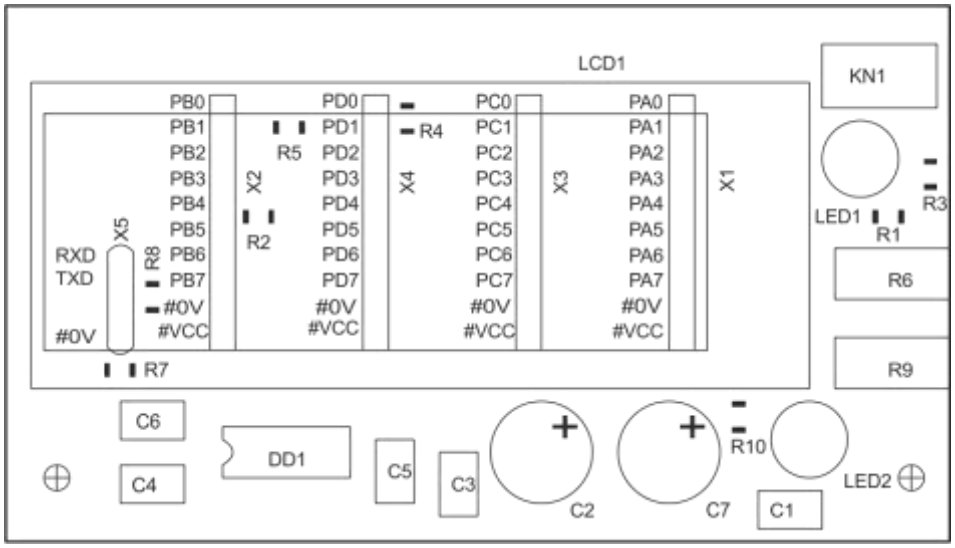


Рисунок В.4 – Расположение элементов на плате расширения

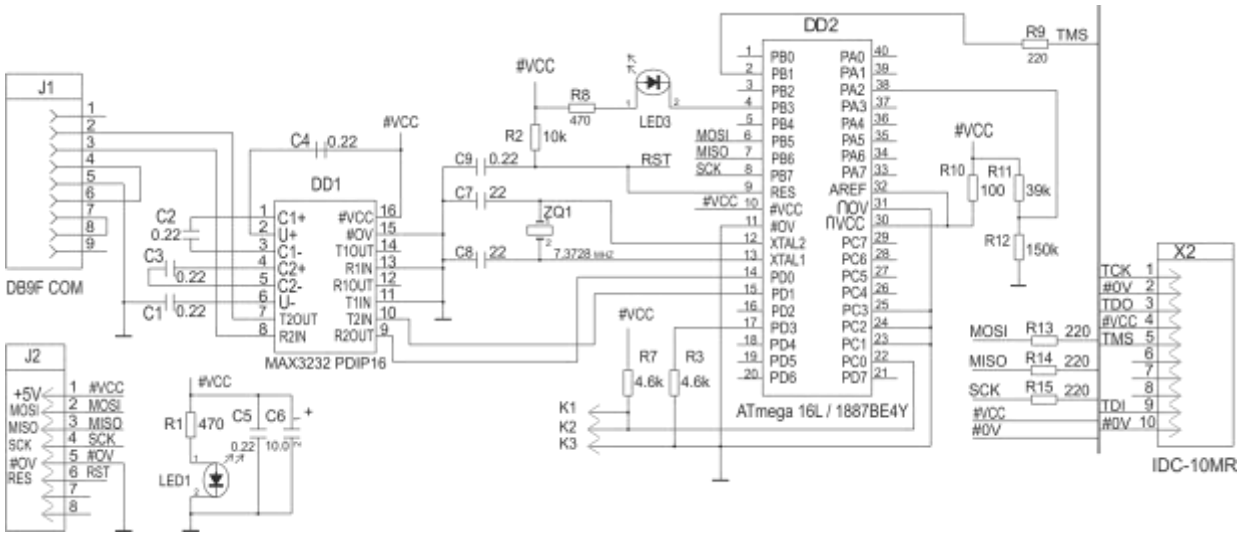


Рисунок В.5 – Схема электрическая RS232 AVR JTAGICE
(внутрисхемный отладчик и программатор для ATmega16)

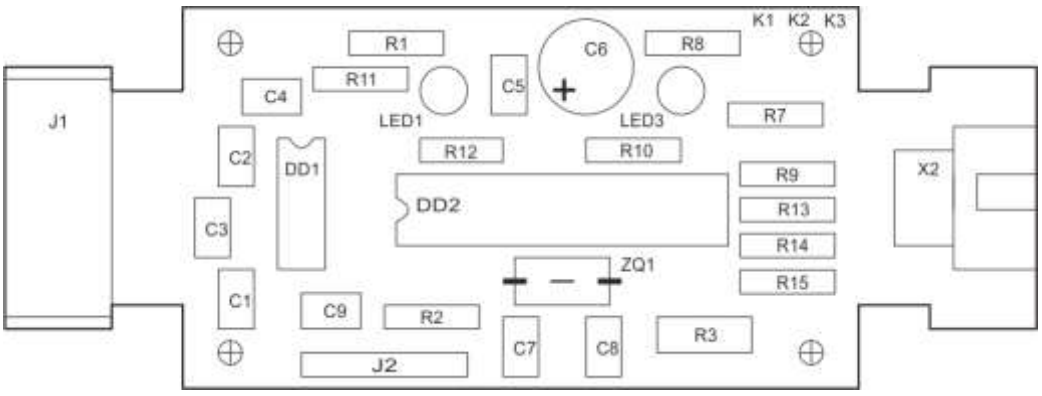
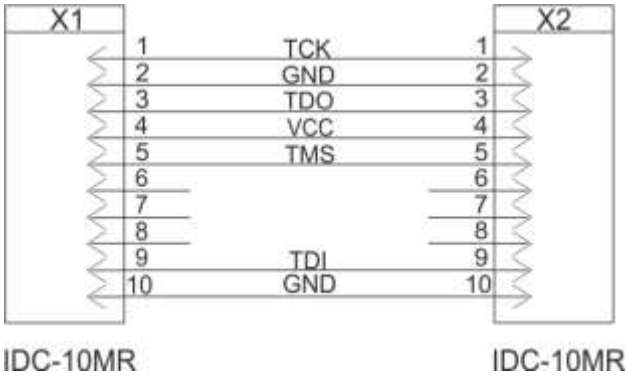


Рисунок В.6 – Расположение элементов на плате RS232 AVR JTAGICE



Длина жгута не более 150 мм

Рисунок В.7 – Схема электрическая жгута «RS232 AVR JTAGICE – Основная плата»

Приложение Г
(справочное)

Отличие микроконтроллеров 1887ВЕ4У и ATmega16

При программировании микроконтроллера 1887ВЕ4У должны учитываться приводимые ниже отличия. Необходимо также отметить, что прилагаемые компилятор или ассемблер, как правило, указанные различия контролируют.

Г.1 Объем встроенной памяти ИС приведен в таблице Г.1.

Т а б л и ц а Г.1 – Объем памяти

Объем памяти	1887ВЕ4У	ATmega16
Flash	8 Кбайт	16 Кбайт
ОЗУ	512 байт	1 Кбайт
ЭСППЗУ	1 Кбайт	512 байт

Г.2 Необходимо также учитывать, что в микроконтроллере ATmega16 область загрузчика (Boot Loader area) и стартовый адрес секции «нет чтения при записи» NRWW (No-Read-While-Write) также изменены, как показано в таблице Г.2.

Т а б л и ц а Г.2 – Объем памяти загрузчика и стартовый адрес секции NRWW

	1887ВЕ4У	ATmega16
Размер страницы Boot Loader Flash	32 слова	64 слова
Стартовый адрес секции NRWW	0xC00	0x1C00

Г.3 Векторы прерываний и относительных переходов/вызовов

Таблицы прерываний микросхем идентичны. В ИС 1887ВЕ4У используются векторы прерываний с одним словом, а в ATmega16 – с двумя словами (это обусловлено различием в объеме памяти). Поэтому инструкции JMP/CALLs должны быть изменены на RJMP/RCALLs при переходе на ИС 1887ВЕ4У.

Г.4 Конфигурационные биты

Два конфигурационных бита отличаются, как показано в таблице Г.3.

Таблица Г.3 – Конфигурационные биты микроконтроллеров

Бит		1887ВЕ4У	ATmega16
Старший конфигурационный байт	7	**S8535C**	**OCDEN**
	6	**WDTON**	**JTAGEN**
	5	SPIEN	SPIEN
	4	CKOPT	CKOPT
	3	EESAVE	EESAVE
	2	BOOTSZ1	BOOTSZ1
	1	BOOTSZ0	BOOTSZ0
	0	BOOTRST	BOOTRST
Младший конфигурационный байт	7	BODLEVEL	BODLEVEL
	6	BODEN	BODEN
	5	SUT1	SUT1
	4	SUT0	SUT0
	3	CKSEL3	CKSEL3
	2	CKSEL2	CKSEL2
	1	CKSEL1	CKSEL1
	0	CKSEL0	CKSEL0

Г.5 Параметры Watchdog

При переходе на ИС 1887ВЕ4У может понадобиться подстройка режима работы Watchdog.

Г.6 Дополнительная информация

AVR088: Migrating between ATmega8535 и ATmega16 (AVR8535 mega16.pdf).

Лист регистрации изменений								
Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	Номер документа	Подпись	Дата
	измененных	замененных	новых	аннулированных				
	-		-	-				